ISB Cancer Gateway in the Cloud Documentation

Release 2.0.0

the ISB-CGC team

Sep 09, 2022

GETTING STARTED

1 About the ISB-CGC Platform 3

2 ISB-CGC Data Overview 5

3 Quick-Start Guide 9

4 Getting Started with Analysis 11

5 How to Request Cloud Credits 13

6 Best Practices 15

7 Benefits of Using The Cloud 17

8 Cost Management 19

9 ISB-CGC Office Hours 21

10 Programs and Data Sets 23

11 ISB-CGC BigQuery Table Search 65

12 ISB-CGC BigQuery Tables 75

13 ISB-CGC Web Interface (Web App) 97

14 Cohort Builder/Data Explorer 145

15 Cancer Data File Browser 147

16 Mitelman Database 149

17 The *TP53* Database 151

18 The Synthetic Lethality Resource 153

19 Pipelines and APIs 155

20 Data Access and Security Overview 201

21 Accessing Controlled Data 203

22 ISB-CGC Notebooks 211

23 Statistical Notebooks 213

24 Machine Learning Notebooks 215

25 HTAN Notebooks 217

26 Tutorials and How-To Guides 219

27 Release Notes 397

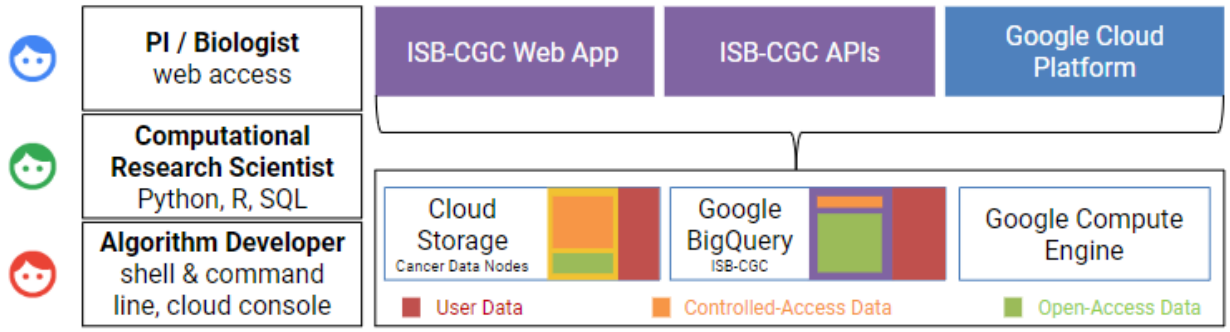
28 Quick Links 509

29 Frequently Asked Questions (FAQ) 513

30 Contact Us 521

Democratizing access to cancer data in the cloud

Contained within this documentation are descriptions of ISB-CGC features along with guides and tips for exploring data sets hosted on the Google Cloud Platform.



The ISB-CGC aims to serve the needs of a broad range of cancer researchers ranging from scientists or clinicians who prefer to use an interactive web-based application to access and explore the rich TCGA, TARGET, CCLE, COSMIC and other data sets, to computational scientists who want to write their own custom scripts using languages such as R or Python, accessing the data through APIs, and to algorithm developers who wish to spin up thousands of virtual machines to analyze hundreds of terabytes of sequence data.

– the ISB-CGC team

CHAPTER

ONE

ABOUT THE ISB-CGC PLATFORM

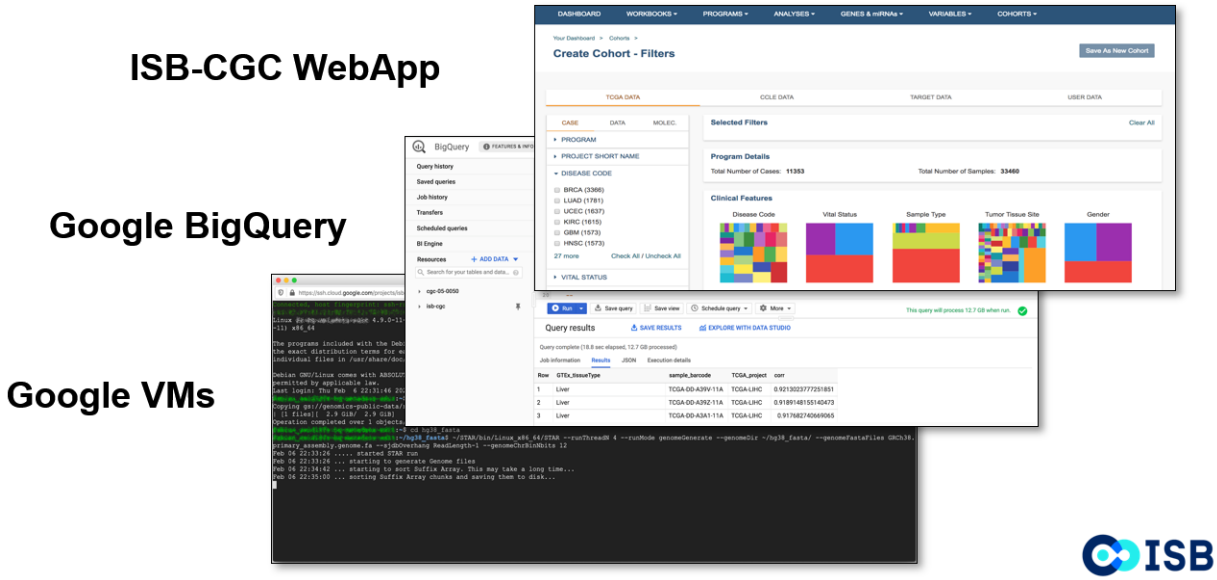
The ISB Cancer Gateway in the Cloud (ISB-CGC) is one of three National Cancer Institute (NCI) Cloud Resources tasked with bringing cancer data and computation power together through cloud platforms. It is a collaboration between the Institute for Systems Biology (ISB) and General Dynamics Information Technology Inc. (GDIT). Since starting in 2014 as part of NCI's Cloud Pilot Resource initiative, ISB-CGC has provided access to increasing amounts of cancer data in the cloud.

1.1 Exploring Cancer Data

The ISB-CGC Platform enables a wide range of users to bring their analysis tools to the data in the cloud, eliminating the need to download and store large data sets. Built with the Google Cloud Platform, it provides several entry points for exploring and analyzing cancer data:

- The **ISB-CGC Web Application** allows users to interactively create and explore cohorts of interest. It includes the functionality of the Cancer Data File Browser and the Cohort Builder/Data Explorer as well as other tools.
  - The **Cancer Data File Browser** allows users to explore a comprehensive selection of cancer related data files in Google Cloud Storage Buckets, such as raw sequencing, cancer nucleotide variation, pathology or radiology images.
  - The **Cohort Builder/Data Explorer** is a web interface which builds cohorts based on clinical demographics and molecular filters. Compare patient cohorts with various exploration tools including IGV viewer, image viewers, and analytical visualization.
- The **ISB-CGC API** gives users the ability to programmatically work with data such as cases, samples, cohorts, files and cloud projects.
- The **ISB-CGC BigQuery Table Search** is a discovery tool that allows the user to explore and search for ISB-CGC Google BigQuery tables.
- On the **Google Cloud Platform BigQuery Console**, ISB-CGC tables can be viewed and queried directly.
- **Python and R** can interface with the ISB-CGC tables, retrieving and analyzing data.
- Using **Google Compute Engines and VMs**, workflows can be run to perform data analysis.

Please see the USER GUIDE section to learn more about each of these tools and to see Jupyter and R Notebook examples. See the MORE INFORMATION section for tutorials, release notes, Frequently Asked Questions and more.



Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CHAPTER

TWO

ISB-CGC DATA OVERVIEW

ISB-CGC provides access to data from several research programs, such as **The Cancer Genome Atlas (TCGA)**, **Therapeutically Applicable Research to Generate Effective Treatments (TARGET)**, **Cancer Cell Line Encyclopedia (CCLE)** and **Catalogue of Somatic Mutations in Cancer (COSMIC)**. The full list is available here.

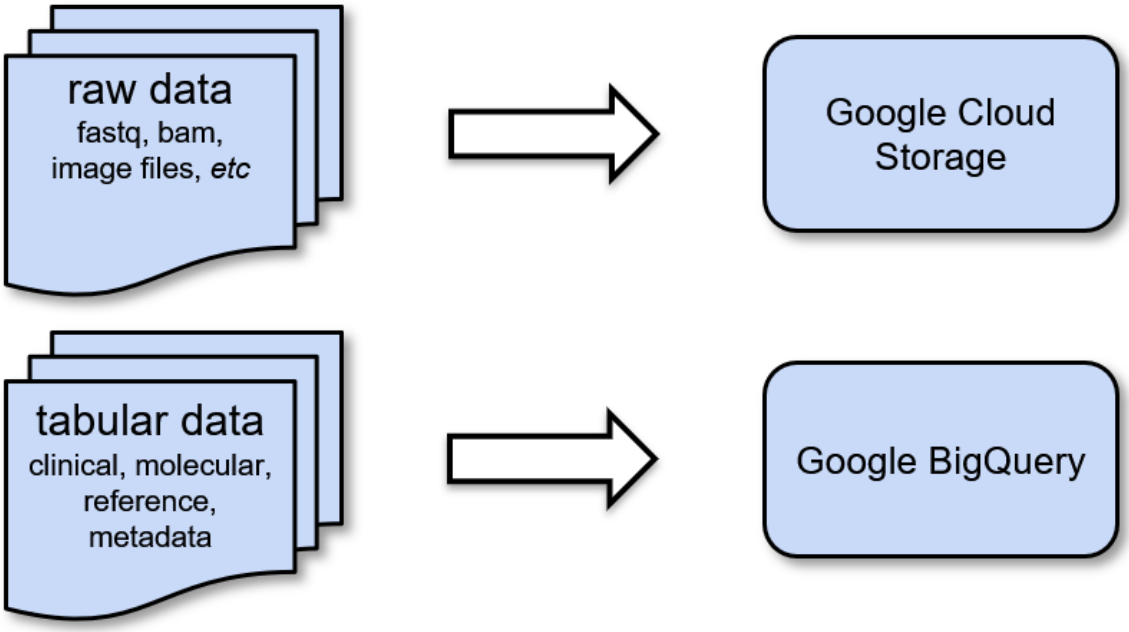
The majority of the data made available through ISB-CGC originates from NCI Genomic Data Commons (GDC). Users can access GDC data on the cloud through ISB-CGC. Users have access to both raw and processed data from cancer patients.

NCI Proteomics Data Commons (PDC) data is also available in ISB-CGC Google BigQuery tables.

In general, almost all raw data is controlled-access and is accessible through Google Cloud Storage buckets; only those users with proper authorization can access them. The GDC has established bioinformatics workflows/pipelines executed on the raw data to generate processed data. In this way, users can directly access the processed data without having to run compute-intensive workflows themselves. However, users who wish to run their own workflows/pipelines still have access to the raw data as well.

Processed data, however, are generally open-access. ISB-CGC allows users to utilize this processed data in two ways on the platform:

- **Google Cloud Storage:** Individual GDC processed data files are accessible through GDC Google Cloud Storage buckets; ISB-CGC provides pointers to these files.
- **Google BigQuery:** Processed data are consolidated by datatype (ex. Clinical, DNA Methylation, RNAseq, Somatic Mutation, etc.) and transformed into ISB-CGC Google BigQuery tables for ease of access and analysis. This novel approach allows our users to quickly analyze information from thousands of patients in our curated BigQuery tables.



2.1 Google Cloud Storage

Google Cloud Storage (GCS) is a cloud-based object-store that is used to store many types of (usually binary) data, typically processed by custom software pipelines. The data hosted by GDC is contained within Google Cloud Storage. Metadata stored within ISB-CGC BigQuery tables contains pointers to file locations in this GDC data.

2.2 Google BigQuery

Google BigQuery (BQ) is a columnar database ideal for storing tabular data. Its query speed is automatically scaled by multiprocessing. Data is accessed using a powerful SQL language interface.

ISB-CGC stores high-level clinical, biospecimen, and molecular data from the main NCI programs in the BigQuery projects isb-cgc-bq and isb-cgc. It also stores a large amount of metadata about files that are stored in the GDC Google Cloud Storage, as well as genome reference sources (*e.g.* GENCODE, miRBase, *etc.*). Most of these data sets and tables are completely *open access* and available to the research community.

2.3 Terms of Use

Using the data stored in ISB-CGC is subject to the terms of use of its origin.

- Genomic Data Commons (GDC) Data Access Policies
- Proteomics Data Commons (PDC) Data Use Guidelines
- The Cancer Imaging Archive (TCIA) Data Usage Policies and Restrictions
- The Cancer Research Data Commons (CRDC) Data Use Policy Statement

For reference and other tables, see the table description for specific information.

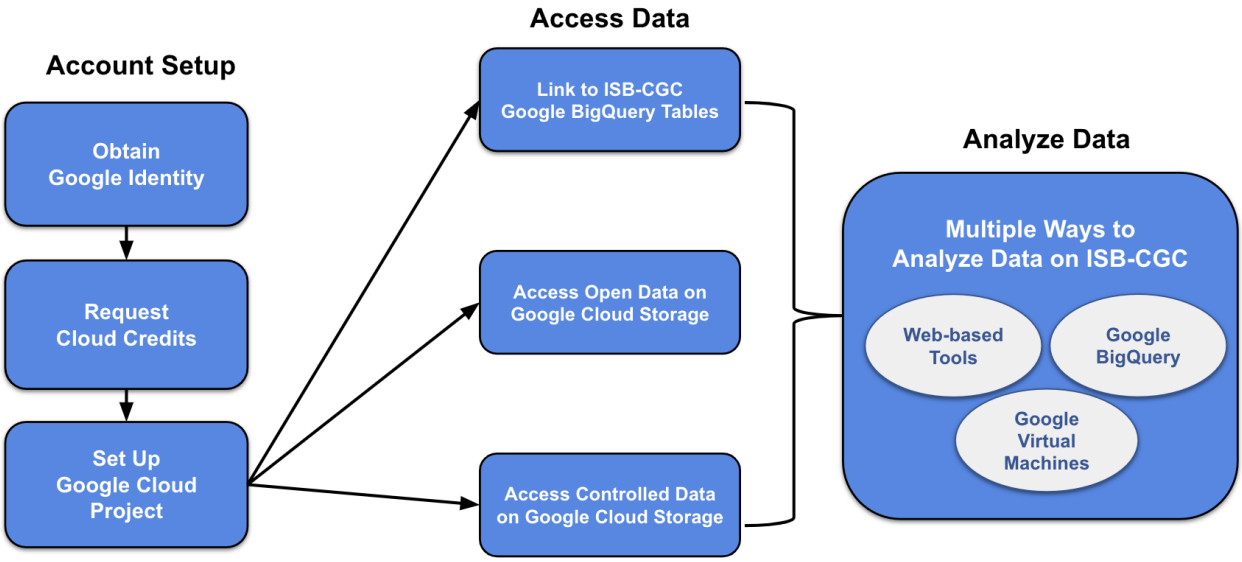
Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CHAPTER

THREE

QUICK-START GUIDE

ISB-CGC provides both interactive (through a web application) and programmatic access to data hosted by institutes such as the Genomic Data Commons (GDC) of the National Cancer Institute (NCI), and the Wellcome Trust Sanger Institute, leveraging many aspects of the Google Cloud Platform. To get started, you'll need a Google Cloud Project. Additionally, to access controlled data, you'll also need dbGaP authorization.



3.1 Google Cloud Project Setup and Data Access

A Google Cloud Project (GCP) is required to make use of all of the data, tools, and Google Cloud functionality.

Obtain a Google identity

- Do you or your institution already have a Google identity, such as a Gmail account? If so, you can proceed to the next step.
- If not, it only takes a minute to create a Google identity. You can even link a non-Gmail account (eg. scientist@nih.gov) as a Google identity by this method.

Request Google Cloud Credits

- Take advantage of a one-time $300 Google Credit.
- If you have already used this one-time offer (or there is some other reason you cannot use it), see this information about how to request ISB-CGC Cloud Credits.

Set up a Google Cloud Project

- See Google's documentation about how to create a Google Cloud Project.
- Learn about how to add members and roles to a project.
- If you'll be accessing controlled data, register the GCP project.
- Enable Required Google Cloud APIs

Connect to ISB-CGC's cancer data tables in Google BigQuery

- To obtain access to the ISB-CGC open access project tables in BigQuery, users can link these tables to their GCP project as described here.
- To obtain access to the ISB-CGC controlled access project tables in BigQuery, users can link these tables to their GCP project as described here.

Access open-access data

- All individual processed data files are accessible through GDC Google Cloud Storage buckets; ISB-CGC provides pointers to these files. Examples of how to find these URLs are in this section, on each Program's documentation page; these SQL queries can also be incorporated into notebooks or workflows.

Access controlled data (with proper authorization)

- To access controlled data (primarily raw data files in the GDC Google Cloud Storage buckets), users must first be authenticated by NIH (via the ISB-CGC web-app). Upon successful authentication, user dbGaP authorization will be verified. These two steps are required before the user's Google identity is added to the access control list (ACL) for the controlled data. At this time, this access must be renewed every 24 hours.

Getting Started with Analysis

Now you're ready to perform analysis. ISB-CGC offers web-based interactive interactive analysis, analysis with Google BigQuery and analysis using APIs and VMs. Please see the next section Getting Started with Analysis to learn more.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CHAPTER

FOUR

GETTING STARTED WITH ANALYSIS

ISB-CGC enables researchers to analyze cloud-based cancer data through a collection of powerful web-based tools and Google Cloud technologies. Learn more about the different analytical methods ISB-CGC users employ on their research projects.

4.1 Interactive web-based Cancer Data Analysis & Exploration

Explore and analyze ISB-CGC cancer data through a suite of graphical user interfaces (GUIs) that allow users to select and filter data from one or more public data sets (such as TCGA, CCLE, and TARGET), combine these with your own uploaded data and analyze using a variety of built-in visualization tools.

Cohort Builder/Data Explorer *Create and explore cohorts of interest*	• ISB-CGC Cohort Builder/Data Explorer Documentation • ISB-CGC Cohort Builder/Data Explorer
Interactive Pathology and Radiology Image Viewers *View images from cancer patients using integrated image viewers*	• ISB-CGC Image Viewers Documentation
Integrative Genomics Viewer (IGV) *Explore and visualize genomic data*	• ISB-CGC Integrative Genomics Viewer (IGV) Documentation
Cancer Data File Browser *Browse and identify files associated with cohorts of interest*	• ISB-CGC Cancer Data File Browser Documentation • ISB-CGC Cancer Data File Browser
Mitelman Database for Chromosome Aberrations and Gene Fusions in Cancer *Explore relationships between chromosomal changes and cancer*	• ISB-CGC Mitelman Database Documentation • ISB-CGC Mitelman Database
The *TP53* Database *Explore TP53 variant data that have been reported in the published literature or are available in other public databases.*	• ISB-CGC The *TP53* Database Documentation • ISB-CGC The *TP53* Database

4.2 Cancer data analysis using Google BigQuery

Processed data are consolidated by data type (ex. Clinical, DNA Methylation, RNAseq, Somatic Mutation, Protein Expression, etc.) from sources including the Genomics Data Commons (GDC) and Proteomics Data Commons (PDC) and transformed into ISB-CGC Google BigQuery tables. This allows users to quickly analyze information from thousands of patients in curated BigQuery tables using Structured Query Language (SQL). SQL can be used from the Google BigQuery Console but can also be embedded within Python, R and complex workflows, providing users with flexibility. The easy, yet cost effective, "burstability" of BigQuery allows you to, within minutes (as compared to days or weeks on a non-cloud based system), calculate statistical correlations across millions of combinations of data points.

BigQuery Table Search User Interface *Learn more about ISB-CGC hosted BigQuery tables*	• ISB-CGC BigQuery Table Search Documentation • ISB-CGC BigQuery Table Search
Google BigQuery Console *Use SQL to analyze and query ISB-CGC cancer data stored in Google's cloud-based data warehouse*	• ISB-CGC BigQuery Documentation • Google BigQuery Documentation • Google Cloud BigQuery Console
Notebooks *Seamlessly integrate ISB-CGC tables with R and Python to conduct robust analyses*	• ISB-CGC Notebook Documentation • ISB-CGC Statistical Notebook Documentation

4.3 Cancer data analysis using APIs & Google Cloud Virtual Machines

ISB-CGC enables the use of as many workflow technologies as possible through documentation, support, and necessary infrastructure.

ISB-CGC APIs *Programmatically access data and user-generated cancer patient cohort information*	• ISB-CGC API Documentation • ISB-CGC API
Connecting to GA4GH and Cloud Life Sciences APIs: *Easily connect to APIs from ISB-CGC*	• How to find a tool using GA4GH TRS Notebook • How to use a GA4GH tool using WES Notebook • Google API Documentation
Running workflows on ISB-CGC *Execute open-source and custom pipelines/algorithms on scalable virtual machines*	• ISB-CGC Workflow Documentation • We recommend tools such as the Google Cloud SDK, Google Compute Engine, Virtual Machines and Docker to assist your analyses.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CHAPTER

FIVE

HOW TO REQUEST CLOUD CREDITS

ISB-CGC offers free Google Cloud credits for researchers to try out our platform! To get trial funding in an ISB-CGC funded Google Cloud Platform (GCP) project, please send your request to request-gcp@isb-cgc.org.

Note: an ISB-CGC funded GCP project is not required to work with the ISB-CGC platform.

In your request, please:

- Describe your research goals in some detail
- Include information such as
- The type of data that you plan to use
- The algorithms and/or methods you plan to apply
- Estimate of the storage
- Computing costs you expect to incur
- Include if you have students or collaborators who will also be accessing the same cloud project

  Teams working on a single project should all use the same cloud project
- If you have previous experience using the Google Cloud Platform along with which specific components (*eg* Compute Engine, BigQuery, Cloud Datalab, *etc*).

All reasonable requests will receive an initial allocation of $300 towards storage and compute costs. We expect that this amount of funding will be more than enough for you to become familiar with the platform. If you expect that you will need additional funding to complete your planned research, this initial amount may be used to perform prototype analyses and to better estimate your total costs. At that time, you may request additional funding.

Please be aware that we will be monitoring your cloud resource usage on a daily basis and will alert you as you begin to approach your funding limit. If you exceed your allocation limit and we are not able to contact you by email for several days, we may need to take action to shut your project down which could cause you to lose work and data.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CHAPTER

SIX

BEST PRACTICES

6.1 Don't Download the Data

The ISB-CGC platform is one of NCI's Cancer Cloud Resources and our mission is to host cancer data (such as TCGA and TARGET data) in the cloud so that researchers around the world may work with data without needing to download and store the data at their own local institutions.

Remember those times when you had to wait weeks to download the data - you don't need to do that any more! The data is already on the cloud, so you can collaborate with other researchers much more easily. Be mindful that if you download data, you'll incur egress charges. Google egress charges information

6.2 Computing on the Cloud

Most of the same linux commands, scripts, pipelines/workflows, genomics software packages and docker containers that you run on your local machine can be executed on virtual machines on Google Cloud.

a. The basics and best practices on how to launch virtual machines (VMs) are described here in our documentation. **NOTE: When launching VMs, please maintain the default firewall settings.**

b. Compute Engine instances can run the public images for Linux and Windows Server that Google provides as well as private custom images that you can create or import from your existing systems.

   Be careful as you spin up a machine, as larger machines cost you more. If you are not using a machine, shut it down. You can always restart it easily when you need it.

   Example use-case: You would like to run Windows-only genomics software package on the TCGA data. You can create a Windows based VM instance.

c. More details on how to deploy docker containers on VMs are described here in Google's documentation: deploying containers

d. A good way to estimate costs for running a workflow/pipeline on large data sets is to test them first on a small subset of data.

e. There are different VM types depending on the sort of jobs you wish to execute. By default, when you create a VM instance, it remains active until you either stop it or delete it. The costs associated with VM instances are detailed here: compute pricing

f. If you plan on running many short compute-intensive jobs (for example indexing and sorting thousands of large bam files), you can execute your jobs on preemptible virtual machines. They are 80% cheaper than regular instances. preemptible vms

Example use-cases:

- Using preemptible VMs, researchers were able to quantify transcript levels on over 11K TGCA RNAseq samples for a total cost of $1,065.49.

  Tatlow PJ, Piccolo SR. A cloud-based workflow to quantify transcript-expression levels in public cancer compendia. Scientific Reports 6, 39259

- Also Broad's popular variant caller pipeline, GATK, was designed to be able to run on preemptible VMs.

6.3 Storage on the Cloud

The Google Cloud Platform offers a number of different storage options for your virtual machine instances: disks

a. Block Storage:

- By default, each virtual machine instance has a single boot persistent disk that contains the operating system. The default size is 10GB but can be adjusted up to 64TB in size. (Be careful! High costs here, spend wisely!)
- Persistent disks are restricted to the zone where your instance is located.
- Use persistent disks if you are running analyses that require low latency and high-throughput.

b. Object Storage: Google Cloud Storage (GCS) buckets are the most flexible and economical storage option.

- Unlike persistent disks, Cloud Storage buckets are not restricted to the zone where your instance is located.
- Additionally, you can read and write data to a bucket from multiple instances simultaneously.
- You can mount a GCS bucket to your VM instance when latency is not a priority or when you need to share data easily between multiple instances or zones.

  An example use-case: You want to slice thousands of bam files and save the resulting slices to share with a collaborator who has instances in another zone to use for downstream statistical analyses.

- You can save objects to GCS buckets including images, videos, blobs and unstructured data.

  A comparison table detailing the current pricing of Google's storage options can be found here: storage features

---

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CHAPTER

SEVEN

BENEFITS OF USING THE CLOUD

Working in the cloud is exceptionally scalable and versatile; you only use as much as you need, whether that's in terms of storage space or processing cores. Cloud-based data is easily read by massively parallel processes, expediting results. When you're done, resources disappear! You don't have idle resources sitting around collecting dust.

Don't be intimidated by the cloud! Scale your analyses using the data on ISB-CGC. If you've conducted bioinformatics before using the command line or SQL, this will be just as easy (if not easier) and we are also here to help. Email feedback@isb-cgc.org or visit our Community Notebooks page for guides and tutorials.

Most bioinformaticians today are likely accustomed to using the high performance compute (HPC) resources provided by their institution to conduct high-throughput bioinformatics analyses. Here's a breakdown on how the Google Cloud Platform compares to your institution's HPC resources.

	Your University's HPC Resource	Google Cloud Platform
Operating Systems	Linux, Windows	Virtual machines can run Linux and Windows
Compute	Virtual machines not determined by you	You can sign up with you own virtual machines*
Storage	Block Storage • Small storage is available in your home directory (usually around 1TB) • Some Scratch storage that is often deleted after a certain amount of time • Storage is usually a shared resource	Block Storage & Object Storage • Each virtual machine instance has a single boot persistent disk with a default size of 10GB that can be adjusted up to 64TB* • For storage that needs IO, consider persistent disks • Google Cloud Storage (GCS) buckets are the most flexible and economical storage option • You can save objects to GCS buckets including images, videos, blobs, and unstructured data
Pricing	Depends on the institution: • Institution provides basic HPC resources for researchers free of charge • PIs requiring larger-scale resources must purchase clusters and storage space	Pay as you go • You pay for the compute resources and storage that you use*
Do you have to wait?	Yes • Resources are shared among users • Scheduler systems used to schedule jobs based on resource availability	No • Once you've set up a Google Cloud Platform account, you can spin up a virtual machine and begin computing quickly
Is machine powerful enough?	Yes and no, depends on what you're trying to do; often it's a no	Compute resources and storage are unlimited, but you have to pay for it*
Accessing Cancer Genomics Data	Typically not stored on the HPC; you have to download to your local machine	Data is stored on the cloud
How to connect	Log in using Secure Shell protocol (SSH)	Log in using Secure Shell protocol (SSH)

***Be careful of costs**, See the Cost Management page for more information.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CHAPTER

EIGHT

COST MANAGEMENT

This section details a few use cases and their approximate costs in order to help users estimate cloud costs for their analyses.

8.1 Estimating Costs for Common Bioinformatics and Data Analysis Tasks

The following table summarizes order-of-magnitude costs for common data analysis tasks. For example an order-of-magnitude cost of $10 indicates that the cost can be up to $10, estimated from the given example notebooks. Estimated costs between $10 and $100 are reported as the next order of magnitude, $100:

Bioinformatics / Data Analysis Task	Dataset(s)	Tools	Approx. Cost (Max)
Identify differentially expressed genes Example: Breast Cancer Tumor vs. Normal	TCGA	BigQuery, Colab, Python, R	$1
Train a prediction model using gene expression data Example: Logistic Regression, Ovarian Cancer Chemo Response Example: Logistic Regression, Breast Cancer Tumor vs. Normal	TCGA	BigQuery, BigQuery ML, Colab, Python, R	$1 ($100) *
Train a linear regression model using gene expression data Example: Linear Regression, Kidney Cancer Survival	TCGA	BigQuery, BigQuery ML, Colab, Python	$1 ($100) *
Train a deep neural network (DNN) regression model using gene expression data Example: Regression w/ TensorFlow, Kidney Cancer Survival	TCGA	BigQuery, Colab, TensorFlow, Compute Engine w/ GPUs	$1 **
Analyze RNA-seq data using the GDC workflow Example: GDC RNA-seq CWL Workflow	TCGA	Compute Engine, Cloud Storage, CWL	$10 ***

- *BigQuery ML costs depend on data size. In these examples, a subset of data was extracted to a temporary table, which was used as input to BigQuery ML. This reduces costs substantially. If using all gene features of a TCGA dataset, costs can grow to the order of $100.
- **With small datasets, use of GPUs in Colab does not cost extra (unless using Colab Pro). However, if TensorFlow code is executed in a VM with GPUs, the hourly cost can range from $1 to $10.
- ***Cost per sample depends on sample size (i.e., number of reads) and processing time.
- BigQuery ML vs. TensorFlow w/ Compute Engine or Colab GPUs: When choosing between these tools for machine learning, consider the following guidelines:
  - TensorFlow w/ Compute Engine or Colab GPUs: Appropriate for data exploration or parameter tuning requiring multiple iterations of training and evaluation.
  - BigQuery ML: Appropriate for production deployment of machine learning models. For example, after optimizing model parameters, train and deploy the final model with BigQuery ML.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CHAPTER

NINE

ISB-CGC OFFICE HOURS

Do you need assistance with getting started? Questions on merging your research with cancer data in the cloud? Or possibly help with troubleshooting?

We have **virtual Office Hours on Tuesdays and Thursdays** for any questions on ISB-CGC functionality or data that you may have. We look forward to speaking with you.

Day of the Week	Time	Host	Link
Tuesday	2:00pm – 3:00pm Eastern	Poojitha Gundluru	http://meet.google.com/jkg-cxke-yzs
Thursday	11:00am – 12:00pm Eastern	Poojitha Gundluru	http://meet.google.com/jai-kgkg-sii

Note: If you are unable to join either meeting link, please email feedback@isb-cgc.org.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CHAPTER

TEN

PROGRAMS AND DATA SETS

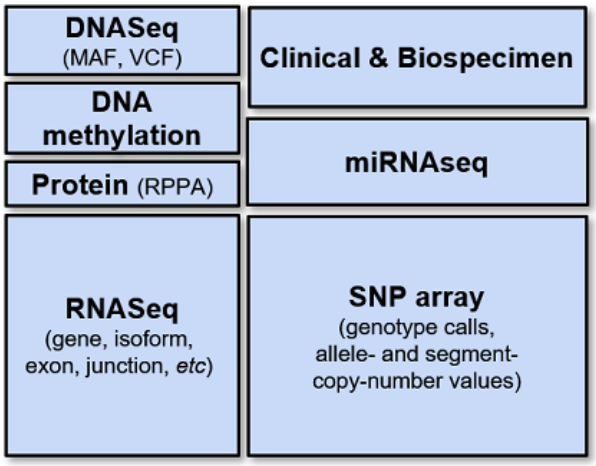
The National Cancer Institute (NCI) Genomic Data Commons (GDC) and Proteomics Data Commons (PDC) provide the cancer research community with data repositories that enables data sharing across cancer genomic and proteomic studies (known as Programs) in support of precision medicine.

The ISB-CGC started with The Cancer Genome Atlas (TCGA) data sets but has expanded to include other data sets from programs such as Therapeutically Applicable Research to Generate Effective Treatments (TARGET). Along with the NCI GDC and PDC data sets, ISB-CGC hosts data sets from programs such as Catalogue Of Somatic Mutations In Cancer (COSMIC) from the Wellcome Trust Sanger Institute. We are always interested in adding new data sets, so if you have any suggestions or requests for additional data, please let us know (feedback@isb-cgc.org).

10.1 Clinical, Biospecimen and Processed -Omics Data Sets

10.1.1 From Genomic Data Commons

Between ISB-CGC and the NCI GDC, there are many cancer data sets available on the Google Cloud Platform. ISB-CGC hosts some carefully curated, high-level clinical, biospecimen and molecular data sets and tables in Google BigQuery as well as radiology and pathology images in Google Cloud Storage. The GDC hosts several more data sets that include low-level sequencing data. For more information about the GDC, see the GDC Overview.



Clinical, biospecimen and processed -omics data (such as RNASeq, etc.) are available in the GDC Cloud Storage buckets, in ISB-CGC BigQuery tables and through ISB-CGC web tools. The table below lists each Program and where (through ISB-CGC) that you can find its data.

- Within the detailed documentation on each Program (click on the Program name), there is an example of how to use the metadata stored in ISB-CGC BigQuery tables to locate the Program's files on the GDC Google Cloud Storage buckets.
- To learn more about using this data with ISB-CGC web tools, go to the ISB-CGC Web Interface section of this document.
- To locate these tables in the ISB-CGC BigQuery project, use the ISB-CGC BigQuery Table Search.

Program	GDC Google Cloud Storage	ISB-CGC BigQuery Tables	ISB-CGC Cohort Builder
BEATAML	✓	✓	✓
CCLE	✓	✓	✓
CGCI	✓	✓	
CMI	✓	✓	
CPTAC	✓	✓	
CTSP	✓	✓	
Exceptional Responders		✓ **	
FM	✓	✓	✓
GENIE	✓	✓ *	
HCMI	✓	✓	
MMRF	✓	✓	✓
MP2PRT		✓ **	
NCICCR	✓	✓ *	
OHSU	✓	✓ *	✓
ORGANOID	✓	✓	
REBC	✓	✓ *	
TARGET	✓	✓	✓
TCGA	✓	✓	✓
TCGA Pathology and Radiology images	✓	✓	✓
TRIO	✓	✓ *	
VAREPOP	✓	✓	
WCDT	✓	✓	

*Clinical and metadata only available

**Clinical data only available

BEATAML1.0 Data Set

About the BEATAML1.0

The BEATAML1.0 data is from several studies focused on acute myeloid leukemia (AML) and the effect of different therapies such as the drug Crenolanib. The implementation of targeted therapies for AML was challenging due to two reasons. The first was due to the intricate mutational patterns within and across patients and the second, was a shortage of pharmacologic agents for most mutational events.

The Crenolanib drug was studied because it is a potent type I pan-FLT3 (GeneID:2322) inhibitor, and FLT3 mutations are associated with poor prognosis and commonly detected in AML patients.

About the BEATAML1.0 Data

The BEATAML1.0 consists of over 220 files with 56 phenotyped subjects, 672 tumor specimens collected from 562 cases, and over 36 TB of data. The data is made up of mainly BAM, VCF, TXT, and TSV files. The majority of the data is whole-exome sequencing along with RNA sequencing. The Project ID in the GDC is BEATAML1.0-CRENOLANIB and BEATAML1.0-COHORT.

For more information on the BEATAML1.0 data, please refer to these sites:

- BEATAML1.0-CRENOLANIB dbGaP site
- BEATAML1.0-COHORT dbGaP site
- GDC Data Portal

Accessing the BEATAML1.0 Data on the Cloud

Besides accessing the files on the GDC Data Portal, you can also access them from the GDC Google Cloud Storage Bucket, which means that you don't need to download them to perform analysis. ISB-CGC stores the cloud file locations in tables in the `isb-cgc-bq.GDC_case_file_metadata` data set in BigQuery.

- To access these metadata files, go to the Google BigQuery console.
- Perform SQL queries to find the BEATAML1.0 files. Here is an example:

```
SELECT active.*, file_gdc_url
FROM `isb-cgc-bq.GDC_case_file_metadata.fileData_active_current` as active, `isb-cgc-bq.
↪GDC_case_file_metadata.GDCfileID_to_GCSurl_current` as GCSurl
WHERE program_name = 'BEATAML1.0'
AND active.file_gdc_id = GCSurl.file_gdc_id
```

Accessing the BEATAML1.0 Data in Google BigQuery

ISB-CGC has BEATAML data, such as clinical and RNA-seq, stored in Google BigQuery tables. Information about these tables can be found using the ISB-CGC BigQuery Table Search with BEATAML selected for filter PROGRAM. To learn more about this tool, see the ISB-CGC BigQuery Table Search documentation.

The BEATAML tables are in project isb-cgc-bq. To learn more about how to view and query tables in the Google BigQuery console, see the ISB-CGC BigQuery Tables documentation.

- Data set `isb-cgc-bq.BEATAML` contains the latest tables for each data type.

- Data set `isb-cgc-bq.BEATAML_versioned` contains previously released tables, as well as the most current table.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CCLE Data Set

About the Cancer Cell Line Encyclopedia

The Cancer Cell Line Encyclopedia (CCLE) project is an effort to conduct a detailed genetic characterization of a large panel of human cancer cell lines. The CCLE provides public access analysis and visualization of DNA copy number, mRNA expression, mutation data and more, for 1000 cancer cell lines.

About the Cancer Cell Line Encyclopedia Data

The CCLE aligned reads (BAM files) are currently available in an open-access Cloud Storage bucket which you can browse here. CCLE data is also available in ISB-CGC Google BigQuery tables.

Accessing the Cancer Cell Line Encyclopedia Data on the Cloud

Besides accessing the files on the GDC Data Portal, you can also access them from the GDC Google Cloud Storage Bucket, which means that you don't need to download them to perform analysis. ISB-CGC stores the cloud file locations in tables in the `isb-cgc-bq.GDC_case_file_metadata` data set in BigQuery.

- To access these metadata files, go to the Google BigQuery console.
- Perform SQL queries to find the CCLE files. Here is an example:

```
SELECT legacy.*, file_gdc_url
FROM `isb-cgc-bq.GDC_case_file_metadata.fileData_legacy_current` as legacy, `isb-cgc-bq.
↪GDC_case_file_metadata.GDCfileID_to_GCSurl_current` as GCSurl
WHERE program_name = 'CCLE'
AND legacy.file_gdc_id = GCSurl.file_gdc_id
```

Accessing the CCLE Data in Google BigQuery

ISB-CGC has CCLE data, such as clinical, biospecimen, copy number segment, RMA Expression and somatic mutation, stored in Google BigQuery tables. Information about these tables can be found using the ISB-CGC BigQuery Table Search with CCLE selected for filter PROGRAM. To learn more about this tool, see the ISB-CGC BigQuery Table Search documentation.

The CCLE tables are in project isb-cgc-bq. To learn more about how to view and query tables in the Google BigQuery console, see the ISB-CGC BigQuery Tables documentation.

- Data set `isb-cgc-bq.CCLE` contains the latest tables for each data type.
- Data set `isb-cgc-bq.CCLE_versioned` contains previously released tables, as well as the most current table.

Note that some of the tables in the isb-cgc-bq project were migrated from the isb-cgc project. If you were using data sets `isb-cgc.ccle_201602_alpha`, `isb-cgc.CCLE_bioclin_v0` and `isb-cgc.CCLE_hg19_data_v0`, they still exist but are deprecated.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CGCI Data Set

About the Cancer Genome Characterization Initiative

The Cancer Genome Characterization Initiative is a series of studies sponsored by the Office of Cancer Genomics (OCG) at the National Cancer Institute (NCI). This program utilizes molecular characterization to uncover distinct features of rare cancers such as HIV+ associated cancers and rare pediatric cancers. The Burkitt Lymphoma Genome Sequencing Project (BLGSP) is one of the projects available through GDC. It explores genetic changes in patients with Burkitt lymphoma (BL) that could lead to better prevention, detection, and treatment of this rare and aggressive cancer.

About the Cancer Genome Characterization Initiative Data

CGCI data consists of 120 cases with RNA sequencing, miRNA sequencing, and whole-genome sequencing data. The NCI GDC houses all the clinical, biospecimen, and molecular characterization data with over 589 BAM, 339 TXT, 402 TSV, 237 BRC XML, 120 BRC PPS XML, and 93 BCR SSF XML files in around 50.28 TB of data. The Project ID in the GDC Data Portal is CGCI-BLGSP.

For more information on the CGCI data, please refer to these sites:

- CGCI Data Matrix
- dbGaP site
- GDC Data Portal

Accessing Cancer Genome Characterization Initiative Data on the Cloud

Besides accessing the files on the GDC Data Portal, you can also access them from the GDC Google Cloud Storage Bucket, which means that you don't need to download them to perform analysis. ISB-CGC stores the cloud file locations in tables in the `isb-cgc-bq.GDC_case_file_metadata` data set in BigQuery.

- To access these metadata files, go to the Google BigQuery console.
- Perform SQL queries to find the CGCI files. Here is an example:

```
SELECT active.*, file_gdc_url
FROM `isb-cgc-bq.GDC_case_file_metadata.fileData_active_current` as active, `isb-cgc-bq.
↪GDC_case_file_metadata.GDCfileID_to_GCSurl_current` as GCSurl
WHERE program_name = 'CGCI'
AND active.file_gdc_id = GCSurl.file_gdc_id
```

Accessing the CGCI Data in Google BigQuery

ISB-CGC has CGCI data, such as clinical, RNA-seq and masked somatic mutations, stored in Google BigQuery tables. Information about these tables can be found using the ISB-CGC BigQuery Table Search with CGCI selected for filter PROGRAM. To learn more about this tool, see the ISB-CGC BigQuery Table Search documentation.

The CGCI tables are in project isb-cgc-bq. To learn more about how to view and query tables in the Google BigQuery console, see the ISB-CGC BigQuery Tables documentation.

- Data set `isb-cgc-bq.CGCI` contains the latest tables for each data type.

- Data set `isb-cgc-bq.CGCI_versioned` contains previously released tables, as well as the most current table.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CMI Data Set

About the CMI

Count Me In (CMI) is a non-profit organization that is stewarded by four organizations: Emerson Collective, Broad Institute of MIT and Harvard, the Biden Cancer Initiative, and the Dana-Farber Cancer Institute. Count Me In works to engage patients via online platforms and share clinical and genomic data through public databases. Count Me In: The Metastatic Breast Cancer (MBC) Project is a patient-driven research initiative to accelerate metastatic breast cancer research. Count Me In: The Angiosarcoma (ASC) Project focuses on angiosarcoma, which is an exceedingly rare soft tissue sarcoma making it difficult to conduct large-scale research studies. Thus, this project demonstrates the feasibility of directly engaging patients to democratize research and create a large patient cohort.

About the CMI Data

The CMI consists of almost 5,000 files with 236 phenotyped subjects and over 16.09 TB of data. The data is made up of mainly BAM, VCF, TXT, and TSV files. The majority of the data is whole-exome sequencing along with RNA sequencing. The Project ID in the GDC is CMI-MBC and CMI-ASC.

For more information on the CMI data, please refer to these sites:

- CMI-MBC dbGaP site
- CMI-ASC dbGaP site
- GDC Data Portal

Accessing the CMI Data on the Cloud

Besides accessing the files on the GDC Data Portal, you can also access them from the GDC Google Cloud Storage Bucket, which means that you don't need to download them to perform analysis. ISB-CGC stores the cloud file locations in tables in the `isb-cgc-bq.GDC_case_file_metadata` data set in BigQuery.

- To access these metadata files, go to the Google BigQuery console.
- Perform SQL queries to find the CMI files. Here is an example:

```
SELECT active.*, file_gdc_url
FROM `isb-cgc-bq.GDC_case_file_metadata.fileData_active_current` as active, `isb-cgc-bq.
↪GDC_case_file_metadata.GDCfileID_to_GCSurl_current` as GCSurl
WHERE program_name = 'CMI'
AND active.file_gdc_id = GCSurl.file_gdc_id
```

Accessing the CMI Data in Google BigQuery

ISB-CGC has CMI data, such as clinical and RNA-seq, stored in Google BigQuery tables. Information about these tables can be found using the ISB-CGC BigQuery Table Search with CMI selected for filter PROGRAM. To learn more about this tool, see the ISB-CGC BigQuery Table Search documentation.

The CMI tables are in project isb-cgc-bq. To learn more about how to view and query tables in the Google BigQuery console, see the ISB-CGC BigQuery Tables documentation.

- Data set `isb-cgc-bq.CMI` contains the latest tables for each data type.
- Data set `isb-cgc-bq.CMI_versioned` contains previously released tables, as well as the most current table.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CPTAC Data Set

About the NCI Clinical Proteomic Tumor Analysis Consortium

The National Cancer Institute's Clinical Proteomic Tumor Analysis Consortium (CPTAC) is a national effort to accelerate the understanding of the molecular basis of cancer through the application of large-scale proteome and genome analysis or proteogenomics.

About the NCI Clinical Proteomic Tumor Analysis Consortium Data Set

From GDC

CPTAC data obtained through the Genomic Data Commons (GDC) consists of whole-genome sequencing, whole-exome sequencing, RNA sequencing, and miRNA sequencing. The program analyzed more than 700 cases. The GDC currently has controlled VCF, TSV, and BAM data available. The Project ID in the GDC Data Portal is CPTAC-2 and CPTAC-3.

For more information on the CPTAC data, please refer to these sites:

- CPTAC-2 dbGaP site
- CPTAC-3 dbGaP site
- GDC Data Portal

From PDC

ISB-CGC also has proteomic CPTAC data, obtained from the Proteomics Data Commons (PDC) API. This includes clinical and protein expression data for breast, ovarian, colon, liver, lung, uterine and other cancers.

The NCI CPTAC has generated a tremendous amount of valuable quantitative proteomics data derived from clinical cancer specimens and makes them publicly accessible to the community. We have imported the data into Google BigQuery, where they can be queried via SQL and easily joined with data tables from TCGA using the BigQuery interface or programmatically with the BigQuery API.

Which studies are available?

- CCRCC - Clear cell renal cell carcinoma
- GBM - glioblastoma multiforme
- HNSCC - Head and neck squamous cell carcinoma

- LUAD - lung adenocarcinoma
- UCEC - Uterine Corpus Endometrial Carcinoma
- Breast cancer
- Colon cancer
- Ovarian cancer

Most studies have both whole proteome as well as phosphoproteome. A few studies also have acetylome and glycoproteome data.

What processing of the raw data is available here?

- Most data have been processed by the original producers and presented in publications.
- The same raw data have been processed uniformly through the CPTAC Common Data Analysis Pipeline (CDAP).
- We provide here the results from the CDAP sourced from the PDC API.

Important considerations:

- All abundances are presented as log2 ratios as computed by the CDAP.
- Abundances are comparable within each study since the same reference was used within each study.
- However, different controls were used for different studies, and therefore extreme caution should be used when comparing abundance values between different studies.
- Some PDC datasets are embargoed, which means that the data may be examined prior to the end of the embargo period, but no manuscripts may be published until the embargo expires. Currently, ISB-CGC does not host any embargoed data in our BQ datasets.

Python Jupyter Notebooks showing examples of queries of PDC CPTAC data are available at:

- How do I explore CPTAC protein abundances?
- How do I compare protein and gene expression in CPTAC?
- Compute correlations of protein and gene expression in CPTAC
- Comparing protein expression from different pipelines - CPTAC

Accessing the NCI Clinical Proteomic Tumor Analysis Consortium Data on the Cloud

Besides accessing the GDC files on the GDC Data Portal, you can also access them from the GDC Google Cloud Storage Bucket, which means that you don't need to download them to perform analysis. ISB-CGC stores the cloud file locations in tables in the `isb-cgc-bq.GDC_case_file_metadata` data set in BigQuery.

- To access these metadata files, go to the Google BigQuery console.
- Perform SQL queries to find the CPTAC files. Here is an example to find CPTAC-2 GDC files:

```
SELECT active.*, file_gdc_url
FROM `isb-cgc-bq.GDC_case_file_metadata.fileData_active_current` as active, `isb-cgc-bq.
↪GDC_case_file_metadata.GDCfileID_to_GCSurl_current` as GCSurl
WHERE program_name = 'CPTAC' and project_short_name = 'CPTAC-2'
AND active.file_gdc_id = GCSurl.file_gdc_id
```

Here is an example to find CPTAC-3 GDC files:

```
SELECT active.*, file_gdc_url
FROM `isb-cgc-bq.GDC_case_file_metadata.fileData_active_current` as active, `isb-cgc-bq.
↪GDC_case_file_metadata.GDCfileID_to_GCSurl_current` as GCSurl
WHERE program_name = 'CPTAC' and project_short_name = 'CPTAC-3'
AND active.file_gdc_id = GCSurl.file_gdc_id
```

Accessing the CPTAC Data in Google BigQuery

ISB-CGC has GDC CPTAC data, such as clinical, RNA-Seq and somatic mutation, and PDC CPTAC data, such as clinical and protein expression, stored in Google BigQuery tables.

Information about these tables can be found using the ISB-CGC BigQuery Table Search with CPTAC2 and/or CPTAC3 selected for filter PROGRAM. To learn more about this tool, see the ISB-CGC BigQuery Table Search documentation.

The CPTAC tables are in project isb-cgc-bq.

- Data set `isb-cgc-bq.CPTAC` contains the latest tables for each data type.
- Data set `isb-cgc-bq.CPTAC_versioned` contains previously released tables, as well as the most current table.

Note that some data are part of a CPTAC2 retrospective study of TCGA data. These tables are labeled as both program CPTAC2 and TCGA and can be found by filtering for either. The tables are in project isb-cgc-bq.

- Data set `isb-cgc-bq.TCGA` contains the latest tables for each data type.
- Data set `isb-cgc-bq.TCGA_versioned` contains previously released tables, as well as the most current table.

In addition, there are some tables with CPTAC data derived from the 2017 paper Proteogenomics connects somatic mutations to signalling in breast cancer. These are in data set `isb-cgc.hg19_data_previews`. They are labeled with programs CPTAC2 and TCGA and source LIT (for literature).

To learn more about how to view and query tables in the Google BigQuery console, see the ISB-CGC BigQuery Tables documentation. Here is an example of a PDC CPTAC table viewed in the Google BigQuery console: quant_acetylome_prospective_breast_BI_pdc_current

---

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CTSP Data Set

About the Clinical Trials Sequencing Project

The Clinical Trials Sequencing Project (CTSP) is a joint collaboration from the National Cancer Institute (NCI) and the Division of Cancer Treatment and Diagnosis (DCTD) to promote the use of genomics in NCI-sponsored clinical trials and elucidate the molecular basis of response and resistance to therapies studied. Breast cancer, renal cell carcinoma, and diffuse large B-cell lymphoma are the cancer types that are currently under study.

About the Clinical Trials Sequencing Project Data

NCI utilized whole genome sequencing and/or whole-exome sequencing in conjunction with transcriptome sequencing to try to identify recurrent genetic alterations (mutations, deletions, amplifications, rearrangements) and/or gene expression signatures that would be important to the hypothesis(es) submitted by the investigators. The samples are processed and submitted for genomic characterization using pipelines and procedures established within The Cancer Genome Analysis (TCGA) project. There are 89 controlled access BAM files along with TSV files in the GDC.

For more information on the CTSP data, please refer to these sites:

- dbGaP site
- GDC Data Portal

Accessing the Clinical Trials Sequencing Project Data on the Cloud

Besides accessing the files on the GDC Data Portal, you can also access them from the GDC Google Cloud Storage Bucket, which means that you don't need to download them to perform analysis. ISB-CGC stores the cloud file locations in tables in the `isb-cgc-bq.GDC_case_file_metadata` data set in BigQuery.

- To access these metadata files, go to the Google BigQuery console.
- Perform SQL queries to find the CTSP files. Here is an example:

```
SELECT active.*, file_gdc_url
FROM `isb-cgc-bq.GDC_case_file_metadata.fileData_active_current` as active, `isb-cgc-bq.
↪GDC_case_file_metadata.GDCfileID_to_GCSurl_current` as GCSurl
WHERE program_name = 'CTSP'
AND active.file_gdc_id = GCSurl.file_gdc_id
```

Accessing the CTSP Data in Google BigQuery

ISB-CGC has CTSP data, such as clinical and RNA-seq, stored in Google BigQuery tables. Information about these tables can be found using the ISB-CGC BigQuery Table Search with CTSP selected for filter PROGRAM. To learn more about this tool, see the ISB-CGC BigQuery Table Search documentation.

The CTSP tables are in project isb-cgc-bq. To learn more about how to view and query tables in the Google BigQuery console, see the ISB-CGC BigQuery Tables documentation.

- Data set `isb-cgc-bq.CTSP` contains the latest tables for each data type.
- Data set `isb-cgc-bq.CTSP_versioned` contains previously released tables, as well as the most current table.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

Exceptional Responders Data Set

About Exceptional Responders

The Exceptional Responders Initiative is a pilot study to investigate the underlying molecular factors driving exceptional treatment responses of cancer patients to drug therapies.

About Exceptional Responders Data

Exceptional Responders has one project EXCEPTIONAL_RESPONDERS-ER with 84 cases spanning nine disease types and 20 primary sites. Data categories include sequencing reads, transcriptome profiling and simple nucleotide variation.

For more information on Exceptional Responders data, please refer to the site below:

- GDC Data Portal

Accessing the Exceptional Responders Data in Google BigQuery

ISB-CGC has Exceptional Responders data, such as clinical, stored in Google BigQuery tables. Information about these tables can be found using the ISB-CGC BigQuery Table Search with EXCEPTIONAL RESPONDERS selected for filter PROGRAM. To learn more about this tool, see the ISB-CGC BigQuery Table Search documentation.

The Exceptional Responders tables are in project isb-cgc-bq. To learn more about how to view and query tables in the Google BigQuery console, see the ISB-CGC BigQuery Tables documentation.

- Data set `isb-cgc-bq.EXC_RESPONDERS` contains the latest tables for each data type.
- Data set `isb-cgc-bq.EXC_RESPONDERS_versioned` contains previously released tables, as well as the most current table.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

FM Data Set

About the Foundation Medicine

The Foundation Medicine Adult Cancer Clinical Data Set (FM) was a study conducted by Foundation Medicine Inc. (FMI), which is a molecular information company that specializes in precision medicine. FMI has generated genomic profiles for thousands of cancer patients, which they designed to match each patient with a personalized treatment plan.

About the Foundation Medicine Data

FM data set consists of more than 18,000 unique solid tumor samples that underwent genomic profiling on a single uniform platform as part of standard clinical care. The data set is derived from the FoundationOne® genomic profiling assay version 2 that interrogates exonic regions of 287 cancer-related genes and selected introns from 19 genes known to undergo rearrangements in human cancer. The Genomic Data Commons (GDC) currently has VCF, TSV, and MAF data available. There are more than 36,008 VCF files, 84 TSV files, and 42 MAF files available with, 36,050 of them that are Controlled Access and 84 files, which are Open Access. The project identification in the GDC Data Portal is FM-AD.

For more information on the FM data, please refer to these sites:

- dbGaP site
- GDC Data Portal

Accessing the Foundation Medicine Data on the Cloud

Besides accessing the files on the GDC Data Portal, you can also access them from the GDC Google Cloud Storage Bucket, which means that you don't need to download them to perform analysis. ISB-CGC stores the cloud file locations in tables in the `isb-cgc-bq.GDC_case_file_metadata` data set in BigQuery.

- To access these metadata files, go to the Google BigQuery console.
- Perform SQL queries to find the FM files. Here is an example:

```
SELECT active.*, file_gdc_url
FROM `isb-cgc-bq.GDC_case_file_metadata.fileData_active_current` as active, `isb-cgc-bq.
↪GDC_case_file_metadata.GDCfileID_to_GCSurl_current` as GCSurl
WHERE program_name = 'FM'
AND active.file_gdc_id = GCSurl.file_gdc_id
```

Accessing the FM Data in Google BigQuery

ISB-CGC has FM data, such as clinical, stored in Google BigQuery tables. Information about these tables can be found using the ISB-CGC BigQuery Table Search with FM selected for filter PROGRAM. To learn more about this tool, see the ISB-CGC BigQuery Table Search documentation.

The FM tables are in project isb-cgc-bq. To learn more about how to view and query tables in the Google BigQuery console, see the ISB-CGC BigQuery Tables documentation.

- Data set `isb-cgc-bq.FM` contains the latest tables for each data type.
- Data set `isb-cgc-bq.FM_versioned` contains previously released tables, as well as the most current table.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

GENIE Data Set

About the AACR Project Genomics Evidence Neoplasia Information Exchange

The AACR Project Genomics Evidence Neoplasia Information Exchange contains data generated from an international pan-cancer registry to serve as an evidence base for the entire cancer community. Genomic and baseline clinical data from more than 40,000 tumors has been made available in the GDC, following the efforts of AACR's strategic and technical partners, Sage Bionetworks and Memorial Sloan Kettering Cancer Center.

About the AACR Project Genomics Evidence Neoplasia Information Exchange Data

The GENIE data set includes masked annotations, somatic mutations, gene level copy number scores, and transcript fusion analysis. The program analyzed more than 44,000 cases. The Genomic Data Commons (GDC) currently has MAF, TXT, and TSV controlled data available.

For more information on GENIE data, please refer to the site below:

- GDC Data Portal

Accessing the AACR Project Genomics Evidence Neoplasia Information Exchange Data on the Cloud

Besides accessing the files on the GDC Data Portal, you can also access them from the GDC Google Cloud Storage Bucket, which means that you don't need to download them to perform analysis. ISB-CGC stores the cloud file locations in tables in the `isb-cgc-bq.GDC_case_file_metadata` data set in BigQuery.

- To access these metadata files, go to the Google BigQuery console.
- Perform SQL queries to find the GENIE files. Here is an example:

```
SELECT active.*, file_gdc_url
FROM `isb-cgc-bq.GDC_case_file_metadata.fileData_active_current` as active, `isb-cgc-bq.
↪GDC_case_file_metadata.GDCfileID_to_GCSurl_current` as GCSurl
WHERE program_name = 'GENIE'
AND active.file_gdc_id = GCSurl.file_gdc_id
```

Accessing the GENIE Data in Google BigQuery

ISB-CGC has GENIE data, such as clinical, stored in Google BigQuery tables. Information about these tables can be found using the ISB-CGC BigQuery Table Search with GENIE selected for filter PROGRAM. To learn more about this tool, see the ISB-CGC BigQuery Table Search documentation.

The GENIE tables are in project isb-cgc-bq. To learn more about how to view and query tables in the Google BigQuery console, see the ISB-CGC BigQuery Tables documentation.

- Data set `isb-cgc-bq.GENIE` contains the latest tables for each data type.
- Data set `isb-cgc-bq.GENIE_versioned` contains previously released tables, as well as the most current table.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

HCMI Data Set

About the Human Cancer Models Initiative

The Human Cancer Models Initiative (HCMI) is a collaborative international consortium that is generating novel, next-generation, tumor-derived culture models annotated with genomic and clinical data. The collaborating institutions are the National Cancer Institute (NCI), Cancer Research UK (CRUK), Wellcome Sanger Institute (WSI), and foundation Hubrecht Organoid Technology (HUB). The four Cancer Model Development Centers (CMDCs), which are supported by the NCI as part of the HCMI, are Broad Institute of MIT and Harvard (BROAD), Cold Spring Harbor Laboratory (CSHL), Stanford University, and Weill Cornell Medical College.

About the Human Cancer Models Initiative Data

HCMI data consists of 23 cases with over 450 phenotyped subjects with whole-exome sequencing, RNA sequencing, and whole-genome sequencing data. The NCI GDC houses all the clinical, biospecimen, and molecular characterization data with over 460 VCF, 261 BAM, 123 TXT, 57 TSV, and 23 BRC XML files. The Project ID in the GDC Data Portal is HCMI-CMDC.

For more information on the HCMI data, please refer to these sites:

- NCI Cancer Model Development
- dbGaP site
- GDC Data Portal

Accessing the Human Cancer Models Initiative Data on the Cloud

Besides accessing the files on the GDC Data Portal, you can also access them from the GDC Google Cloud Storage Bucket, which means that you don't need to download them to perform analysis. ISB-CGC stores the cloud file locations in tables in the `isb-cgc-bq.GDC_case_file_metadata` data set in BigQuery.

- To access these metadata files, go to the Google BigQuery console.
- Perform SQL queries to find the HCMI files. Here is an example:

```
SELECT active.*, file_gdc_url
FROM `isb-cgc-bq.GDC_case_file_metadata.fileData_active_current` as active, `isb-cgc-bq.
↪GDC_case_file_metadata.GDCfileID_to_GCSurl_current` as GCSurl
WHERE program_name = 'HCMI'
AND active.file_gdc_id = GCSurl.file_gdc_id
```

Accessing the HCMI Data in Google BigQuery

ISB-CGC has HCMI data, such as clinical, RNA-seq and somatic mutation, stored in Google BigQuery tables. Information about these tables can be found using the ISB-CGC BigQuery Table Search with HCMI selected for filter PROGRAM. To learn more about this tool, see the ISB-CGC BigQuery Table Search documentation.

The HCMI tables are in project isb-cgc-bq. To learn more about how to view and query tables in the Google BigQuery console, see the ISB-CGC BigQuery Tables documentation.

- Data set `isb-cgc-bq.HCMI` contains the latest tables for each data type.
- Data set `isb-cgc-bq.HCMI_versioned` contains previously released tables, as well as the most current table.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

MMRF Data Set

About the Multiple Myeloma Research Foundation

The Multiple Myeloma Research Foundation (MMRF) seeks to provide resources and research for patients with multiple myeloma. MMRF began a ten-year study to track new Multiple Myeloma patients to create a rich data set which was led by researchers from the Dana Farber Cancer Institute, Celgene Corp., and the University of Arkansas for Medical Sciences. This trial was named CoMMpass.

About the Multiple Myeloma Research Foundation Data

The CoMMpass trial is a longitudinal observation study of 1000 newly diagnosed myeloma patients receiving various standard approved treatments. This trial aims to collect tissue samples and genetic information along with quality of life, disease, and clinical outcomes. CoMMpass data consists of 995 cases with RNA, whole-exome, and whole-genome sequencing data. The NCI GDC houses all the molecular characterization data with over 10,918 VCF, 6,577 BAM, 2,577 TXT, and 1718 TSV files in around 206.63 TB of data. The Project ID in the GDC Data Portal is MMRF-COMMPASS.

For more information on the MMRF data, please refer to these sites:

- dbGaP site
- GDC Data Portal

Accessing Multiple Myeloma Research Foundation Data on the Cloud

Besides accessing the files on the GDC Data Portal, you can also access them from the GDC Google Cloud Storage Bucket, which means that you don't need to download them to perform analysis. ISB-CGC stores the cloud file locations in tables in the `isb-cgc-bq.GDC_case_file_metadata` data set in BigQuery.

- To access these metadata files, go to the Google BigQuery console.
- Perform SQL queries to find the MMRF files. Here is an example:

```
SELECT active.*, file_gdc_url
FFROM `isb-cgc-bq.GDC_case_file_metadata.fileData_active_current` as active, `isb-cgc-bq.
↪GDC_case_file_metadata.GDCfileID_to_GCSurl_current` as GCSurl
WHERE program_name = 'MMRF'
AND active.file_gdc_id = GCSurl.file_gdc_id
```

Accessing the MMRF Data in Google BigQuery

ISB-CGC has MMRF data, such as clinical, stored in Google BigQuery tables. Information about these tables can be found using the ISB-CGC BigQuery Table Search with MMRF selected for filter PROGRAM. To learn more about this tool, see the ISB-CGC BigQuery Table Search documentation.

The MMRF tables are in project isb-cgc-bq. To learn more about how to view and query tables in the Google BigQuery console, see the ISB-CGC BigQuery Tables documentation.

- Data set `isb-cgc-bq.MMRF` contains the latest tables for each data type.
- Data set `isb-cgc-bq.MMRF_versioned` contains previously released tables, as well as the most current table.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

MP2PRT Data Set

About MP2PRT

The Molecular Profiling to Predict Response to Treatment (MP2PRT) program is part of the NCI's Cancer Moonshot Initiative. This study "Identification of Genetic Changes Associated with Relapse and/or Adaptive Resistance in Patients Registered as Favorable Histology Wilms Tumor on AREN03B2" performs genomic characterization on trio cases (normal tissue, tumor tissue at time of diagnosis, tumor tissue at time of relapse) from patients who relapsed with Favorable Histology Wilms Tumor.

About MP2PRT Data

The MP2PRT data set includes one project MP2PRT-WT with 52 cases. Data categories include sequencing reads, transcriptome profiling, simple nucleotide variation and copy number variation.

For more information on MP2PRT data, please refer to the site below:

- GDC Data Portal

Accessing the MP2PRT Data in Google BigQuery

ISB-CGC has MP2PRT data, such as clinical, stored in Google BigQuery tables. Information about these tables can be found using the ISB-CGC BigQuery Table Search with MP2PRT selected for filter PROGRAM. To learn more about this tool, see the ISB-CGC BigQuery Table Search documentation.

The MP2PRT tables are in project isb-cgc-bq. To learn more about how to view and query tables in the Google BigQuery console, see the ISB-CGC BigQuery Tables documentation.

- Data set `isb-cgc-bq.MP2PRT` contains the latest tables for each data type.
- Data set `isb-cgc-bq.MP2PRT_versioned` contains previously released tables, as well as the most current table.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

NCICCR Data Set

About the NCI Center for Cancer Research

The NCI Center for Cancer Research (NCICCR) conducted a study on the Genomic Variation in Diffuse Large B Cell Lymphomas (DLBCL) through an integrative analysis of genetic lesions in 574 diffuse large B cell lymphomas (DLBCL). The study investigated genomic structural variation, genetic alteration, and its effect on the development and biology of lymphomas by using high throughput sequencing, gene expression, and methylation status.

About the NCI Center for Cancer Research Data

There were around 489 cases that were phenotyped, contributing Authorized-Access, individual-level data. The Genomic Data Commons currently has around 957 controlled access BAM files available. The Project ID in the GDC Data Portal is NCICCR-DLBCL.

For more information on the NCICCR data, please refer to these sites:

- dbGaP site
- GDC Data Portal

Accessing the NCI Center for Cancer Research Data on the Cloud

Besides accessing the files on the GDC Data Portal, you can also access them from the GDC Google Cloud Storage Bucket, which means that you don't need to download them to perform analysis. ISB-CGC stores the cloud file locations in tables in the `isb-cgc-bq.GDC_case_file_metadata` data set in BigQuery.

- To access these metadata files, go to the Google BigQuery console.
- Perform SQL queries to find the NCICCR files. Here is an example:

```
SELECT active.*, file_gdc_url
FROM `isb-cgc-bq.GDC_case_file_metadata.fileData_active_current` as active, `isb-cgc-bq.
↪GDC_case_file_metadata.GDCfileID_to_GCSurl_current` as GCSurl
WHERE program_name = 'NCICCR'
AND active.file_gdc_id = GCSurl.file_gdc_id
```

Accessing the NCICCR Data in Google BigQuery

ISB-CGC has NCICCR data, such as clinical, stored in Google BigQuery tables. Information about these tables can be found using the ISB-CGC BigQuery Table Search with NCICCR selected for filter PROGRAM. To learn more about this tool, see the ISB-CGC BigQuery Table Search documentation.

The NCICCR tables are in project isb-cgc-bq. To learn more about how to view and query tables in the Google BigQuery console, see the ISB-CGC BigQuery Tables documentation.

- Data set `isb-cgc-bq.NCICCR` contains the latest tables for each data type.
- Data set `isb-cgc-bq.NCICCR_versioned` contains previously released tables, as well as the most current table.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

OHSU Data Set

About the Oregon Health & Science University

The Oregon Health & Science University contains data generated from chronic neutrophilic leukemia (CNL), atypical chronic myeloid leukemia (aCML), and unclassified myelodysplastic syndrome/myeloproliferative neoplasms (MDS/MPN-U), which are a group of rare, heterogeneous myeloid disorders.

About the Oregon Health & Science University Data

The data set consists of whole-exome and RNA sequencing on a cohort of over 100 cases of these rare hematologic malignancies. It presents the complete survey of the genomic landscape of these diseases to date. The Project ID in the GDC Data Portal is OHSU-CNL.

For more information on the OHSU data, please refer to these sites:

- dbGaP site
- GDC Data Portal

Accessing the Oregon Health & Science University Data on the Cloud

Besides accessing the files on the GDC Data Portal, you can also access them from the GDC Google Cloud Storage Bucket, which means that you don't need to download them to perform analysis. ISB-CGC stores the cloud file locations in tables in the `isb-cgc-bq.GDC_case_file_metadata` data set in BigQuery.

- To access these metadata files, go to the Google BigQuery console.
- Perform SQL queries to find the OHSU files. Here is an example:

```
SELECT active.*, file_gdc_url
FROM `isb-cgc-bq.GDC_case_file_metadata.fileData_active_current` as active, `isb-cgc-bq.
↪GDC_case_file_metadata.GDCfileID_to_GCSurl_current` as GCSurl
WHERE program_name = 'OHSU'
AND active.file_gdc_id = GCSurl.file_gdc_id
```

Accessing the OHSU Data in Google BigQuery

ISB-CGC has OHSU data, such as clinical, stored in Google BigQuery tables. Information about these tables can be found using the ISB-CGC BigQuery Table Search with OHSU selected for filter PROGRAM. To learn more about this tool, see the ISB-CGC BigQuery Table Search documentation.

The OHSU tables are in project isb-cgc-bq. To learn more about how to view and query tables in the Google BigQuery console, see the ISB-CGC BigQuery Tables documentation.

- Data set `isb-cgc-bq.OHSU` contains the latest tables for each data type.
- Data set `isb-cgc-bq.OHSU_versioned` contains previously released tables, as well as the most current table.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

ORGANOID Data Set

About the Pancreas Cancer Organoid Profiling Program

The Pancreas Cancer Organoid Profiling Program is from the Organoid Profiling Identifies Common Responders to Chemotherapy in Pancreatic Cancer study and contains data generated from a collection of patient-derived pancreatic normal and cancer organoids.

About the Pancreas Cancer Organoid Profiling Data Set

The data set consists of 70 cases and includes whole-genome, targeted exome, and RNA sequencing data on organoids as well as matched tumor and normal tissues. This data set is a valuable resource for pancreas cancer researchers, and those looking to compare primary tissue to organoid culture. The NCI GDC houses all the clinical, biospecimen, and molecular characterization data with over 130 VCF, 298 BAM, 165 TXT, and 110 TSV files in around 21.89 TB of data. The Project ID in the GDC Data Portal is ORGANOID-PANCREATIC.

For more information on the ORGANOID data, please refer to these sites:

- dbGaP site
- GDC Data Portal

Accessing Pancreas Cancer Organoid Profiling Data on the Cloud

Besides accessing the files on the GDC Data Portal, you can also access them from the GDC Google Cloud Storage Bucket, which means that you don't need to download them to perform analysis. ISB-CGC stores the cloud file locations in tables in the `isb-cgc-bq.GDC_case_file_metadata` data set in BigQuery.

- To access these metadata files, go to the Google BigQuery console.
- Perform SQL queries to find the ORGANOID files. Here is an example:

```
SELECT active.*, file_gdc_url
FROM `isb-cgc-bq.GDC_case_file_metadata.fileData_active_current` as active, `isb-cgc-bq.
↪GDC_case_file_metadata.GDCfileID_to_GCSurl_current` as GCSurl
WHERE program_name = 'ORGANOID'
AND active.file_gdc_id = GCSurl.file_gdc_id
```

Accessing the ORGANOID Data in Google BigQuery

ISB-CGC has ORGANOID data, such as clinical and RNA-seq, stored in Google BigQuery tables. Information about these tables can be found using the ISB-CGC BigQuery Table Search with ORGANOID selected for filter PROGRAM. To learn more about this tool, see the ISB-CGC BigQuery Table Search documentation.

The ORGANOID tables are in project isb-cgc-bq. To learn more about how to view and query tables in the Google BigQuery console, see the ISB-CGC BigQuery Tables documentation.

- Data set `isb-cgc-bq.ORGANOID` contains the latest tables for each data type.
- Data set `isb-cgc-bq.ORGANOID_versioned` contains previously released tables, as well as the most current table.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

REBC Data Set

About REBC

REBC studies comprehensive genomic characterization of radiation-related papillary thyroid cancer in the Ukraine after the 1986 Chernobyl nuclear power plan accident. This accident released radioactive contaminants into the surrounding areas in Ukraine, Belarus, and Russia, causing an increased occurrence of thyroid cancer among individuals who were children at the time of the accident or born not long afterwards.

About REBC

The REBC data set includes one project REBC-THYR with 440 cases. Data categories include sequencing reads, transcriptome profiling, simple nucleotide variation and copy number variation.

For more information on REBC data, please refer to the site below:

- GDC Data Portal

Accessing the REBC Data on the Cloud

Besides accessing the files on the GDC Data Portal, you can also access them from the GDC Google Cloud Storage Bucket, which means that you don't need to download them to perform analysis. ISB-CGC stores the cloud file locations in tables in the `isb-cgc-bq.GDC_case_file_metadata` data set in BigQuery.

- To access these metadata files, go to the Google BigQuery console.
- Perform SQL queries to find the REBC files. Here is an example:

```
SELECT active.*, file_gdc_url
FROM `isb-cgc-bq.GDC_case_file_metadata.fileData_active_current` as active, `isb-cgc-bq.
↪GDC_case_file_metadata.GDCfileID_to_GCSurl_current` as GCSurl
WHERE program_name = 'REBC'
AND active.file_gdc_id = GCSurl.file_gdc_id
```

Accessing the REBC Data in Google BigQuery

ISB-CGC has REBC data, such as clinical and metadata, stored in Google BigQuery tables. Information about these tables can be found using the ISB-CGC BigQuery Table Search with REBC selected for filter PROGRAM. To learn more about this tool, see the ISB-CGC BigQuery Table Search documentation.

The REBC tables are in project isb-cgc-bq. To learn more about how to view and query tables in the Google BigQuery console, see the ISB-CGC BigQuery Tables documentation.

- Data set `isb-cgc-bq.REBC` contains the latest tables for each data type.
- Data set `isb-cgc-bq.REBC_versioned` contains previously released tables, as well as the most current table.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

TARGET Data Set

About the Therapeutically Applicable Research to Generate Effective Treatments

The Therapeutically Applicable Research to Generate Effective Treatments (TARGET) program applied a comprehensive genomic approach to determine the molecular changes driving childhood cancers. Investigators formed a collaborative network to facilitate the discovery of molecular targets and translate those findings into the clinic. TARGET is managed by NCI's Office of Cancer Genomics and Cancer Therapy Evaluation Program.

About the Therapeutically Applicable Research to Generate Effective Treatments Data

The ISB-CGC currently hosts several TARGET data sets in BigQuery. TARGET controlled-access data is available to authorized users in Genomic Data Commons and open-access data includes RNA-seq and miRNA-seq expression levels, and is available in BigQuery, along with the open-access clinical and biospecimen information.

The TARGET data is available at the GDC in the legacy archive which contains over 10,000 files for over 5,000 cases. Virtually all of this data is low-level (and controlled-access) sequence data (including 1702 RNA-seq files, 765 miRNA-seq, with the remainder being WXS or WGS DNA-seq BAMs). Some of this data has been reprocessed and is available on the main GDC Data Portal. This newer dataset so far includes 33,402 files representing 6,197 cases and totaling over 200 TB. Over half of the files are controlled-access files, including BAM, VCF, and MAF file types, based on WXS, RNA-seq, and miRNA-seq data. The remaining files are open-access files, including RNA-seq and miRNA-seq quantification, as well as clinical and biospecimen supplement files.

BigQuery Therapeutically Applicable Research to Generate Effective Treatments Data

The open-access TARGET data hosted by the ISB-CGC Platform includes:

- Clinical (de-identified) and Biospecimen data: these data were originally provided in XML files (Level-1)
- Gene (mRNA) expression data: these data were originally provided as TSV files (Level-3)
- microRNA expression data: these data were originally provided as TSV files (Level-3)

The information scattered over thousands of XLSX and TSV files at the GDC is provided in a *much more accessible* form in a series of BigQuery tables.

For more information on TARGET data, please refer to the site below:

- GDC Data Portal

Accessing the Therapeutically Applicable Research to Generate Effective Treatments data on the Cloud

Besides accessing the files on the GDC Data Portal, you can also access them from the GDC Google Cloud Storage Bucket, which means that you don't need to download them to perform analysis. ISB-CGC stores the cloud file locations in tables in the `isb-cgc-bq.GDC_case_file_metadata` data set in BigQuery.

- To access these metadata files, go to the Google BigQuery console.
- Perform SQL queries to find the TARGET files. Here is an example:

```
SELECT active.*, file_gdc_url
FROM `isb-cgc-bq.GDC_case_file_metadata.fileData_active_current` as active, `isb-cgc-bq.
↪GDC_case_file_metadata.GDCfileID_to_GCSurl_current` as GCSurl
```

(continues on next page)

(continued from previous page)

```
WHERE program_name = 'TARGET'
AND active.file_gdc_id = GCSurl.file_gdc_id
```

Accessing the TARGET Data in Google BigQuery

ISB-CGC has TARGET data, such as clinical, biospecimen, miRNA and RNA-seq, stored in Google BigQuery tables. Information about these tables can be found using the ISB-CGC BigQuery Table Search with TARGET selected for filter PROGRAM. To learn more about this tool, see the ISB-CGC BigQuery Table Search documentation.

ISB_CGC also has controlled access TARGET VCF data in Google BigQuery tables; see here for more information.

The TARGET tables are in project isb-cgc-bq. To learn more about how to view and query tables in the Google BigQuery console, see the ISB-CGC BigQuery Tables documentation.

- Data set `isb-cgc-bq.TARGET` contains the latest tables for each data type.
- Data set `isb-cgc-bq.TARGET_versioned` contains previously released tables, as well as the most current table.

Note that some of the tables in the isb-cgc-bq project were migrated from the isb-cgc project. If you were using data sets `isb-cgc.TARGET_bioclin_v0` and `isb-cgc.TARGET_hg38_data_v0`, they still exist but are deprecated.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

TCGA Data Set

About The Cancer Genome Atlas

The Cancer Genome Atlas (TCGA) is a comprehensive and coordinated effort to accelerate the understanding of the molecular basis of cancer through the application of genome analysis technologies, including large-scale genome sequencing.

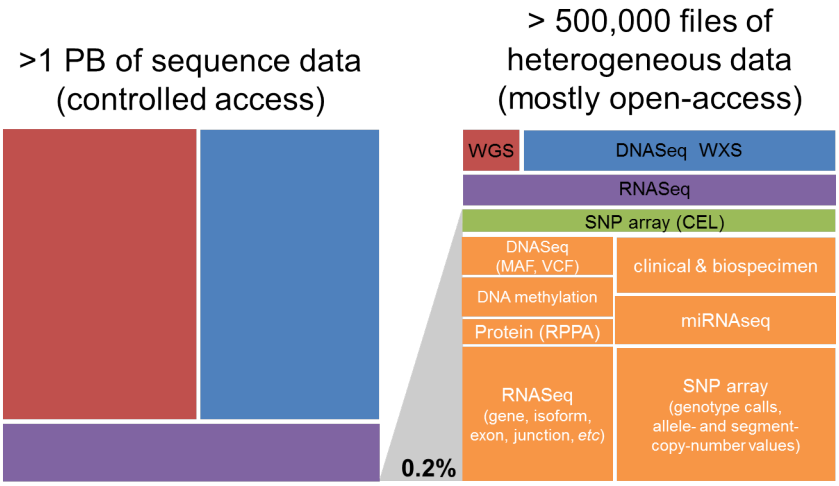
The overarching goal of TCGA is to improve our ability to diagnose, treat and prevent cancer. To achieve this goal in a scientifically rigorous manner, the National Cancer Institute (NCI) and the National Human Genome Research Institute (NHGRI) used a phased-in strategy to launch TCGA. A pilot project developed and tested the research framework needed to systematically explore the entire spectrum of genomic changes involved in more than 20 types of human cancer.

This massive effort was launched in 2006. The final samples were shipped in mid-2014, and analysis of the data produced by this program continues to this day.

About The Cancer Genome Atlas Data

The ISB-CGC hosts several TCGA datasets in BigQuery and more data is available through the Genomic Data Commons (GDC).

The vast majority (over 99%) of this **petabyte** of data consists of low-level sequence data, currently stored as files in the GDC (see figure below). Over the course of the TCGA project, this low-level (*"Level 1"*) data has been processed through a set of standardized pipelines and the resulting high-level (*"Level 3"*) data is frequently the data that is used in most downstream analyses. The ISB-CGC platform aims to make these different types of data accessible to the widest possible variety of users within the cancer research community.



The Cancer Genome Atlas Data Platforms

When working with any of the data types, it is important to also be aware of both the *platform* that was used to generate the underlying raw data as well as the *pipeline* that was used to process the data. For example, over the course of the TCGA study, DNA methylation data were obtained using first the Illumina HumanMethylation27 platform, and later using the HumanMethylation450 platform. Any analysis that combines data from these two platforms across a cohort of samples should take this into consideration. Another example where multiple platforms and/or pipelines were used to produce a single data type is the Level-3 gene expression data: most tumor samples were processed at UNC and the normalized gene-expression values are based on the RSEM method, while some tumor samples were processed at BCGSC and the normalized gene-expression values are based on RPKM.

The Cancer Genome Atlas Data Levels

For each *type* of data, there are typically three *levels* of data:

- Level 1 typically represents raw, unnormalized data
- Level 2 typically represents an intermediate level of processing and/or normalization of the data
- Level 3 typically represents aggregated, normalized, and/or segmented data

The results of integrative or pan-cancer analyses are sometimes referred to as "Level 4" data. More information about Data Level Classification can be found on the NCI page.

The Cancer Genome Atlas Data Types

The TCGA data set is unique in that the tumor samples were assayed using a standard set of platforms and pipelines in order to produce a comprehensive data set including:

- DNA sequencing of tumor samples and matched-normals (typically blood samples) in order to detect somatic mutations
- SNP array-based DNA copy-number and genotyping analysis of tumor samples and matched-normals
- DNA methylation of tumor samples
- messenger RNA (mRNA) expression analysis of the tumor samples to capture the gene expression profile

- microRNA (miRNA) expression profiling of the tumor samples

In addition, protein expression for a significant fraction (~20%) of all tumor samples was obtained using RPPA (reverse phase protein array).

Open-Access The Cancer Genome Atlas Data

The open-access TCGA data includes:

- Clinical (de-identified) and Biospecimen data: these data were originally provided in XML files (Level-1)
- Somatic mutation data: these data were originally provided in MAF files (Level-2)
- DNA copy-number segments: these data were originally provided as segmentation files (Level-3)
- DNA methylation data: these data were originally provided as TSV files (Level-3)
- Gene (mRNA) expression data: these data were originally provided as TSV files (Level-3)
- microRNA expression data: these data were originally provided as TSV files (Level-3)
- Protein expression data: these data were originally provided as TSV files (Level-3)
- TCGA Annotations data: annotations were originally obtained from the TCGA Annotations Manager, and can be found on the GDC Data Portal

The information scattered over tens of thousands of XML and TSV files at the GDC is provided in a *much more accessible* form in a series of BigQuery tables; see the *Accessing the TCGA Data in Google BigQuery* section below.

For more details, please see our Community Notebook Repository for tutorials and code examples in Python and R.

Controlled-Access The Cancer Genome Atlas Data

The controlled-access TCGA data includes:

- SNP array CEL files: these Level-1 data files were provided by the DCC and include over 22,000 files for both tumor and matched-normal samples
- VCF files: these Level-2 data files were provided by the DCC and include over 15,000 files produced by several different centers (primarily Broad and BCGSC)
- MAF files: these "protected" mutation files (Level-2) were provided by the DCC (note that these files were not generated uniformly for all tumor types)
- DNA-seq BAM files: these Level-1 data files were provided by CGHub
  - over 37,000 of these files are available in Google Cloud Storage (GCS)
  - roughly 90% of these BAM files contain exome data, the remaining 10% contain whole-genome data
  - BAM index (BAI) files are also available for all BAM files
- mRNA- and microRNA-seq BAM files: these Level-1 data files were provided by CGHub
  - over 13,000 mRNA-seq BAM files are available in GCS
  - over 16,000 miRNA-seq BAM files are available in GCS
- mRNA-seq FASTQ files: these Level-1 data files were provided by CGHub and include over 11,000 tar files

The Cancer Genome Atlas Data Repository History

Historically, the data was obtained from two former TCGA data repositories:

- **TCGA DCC**: the TCGA Data Coordinating Center which provided a **Data Portal** from which users could download open-access or controlled-access data. This portal provided access to all TCGA data *except* for the low-level sequence data.
- **CGHub**: the **Cancer Genomics Hub** was NCI's secure data repository for all TCGA BAM and FASTQ sequence data files.

In June of 2016, the official data repository for all TCGA and other NCI CCG data is the NCI's Genomic Data Commons (GDC). The original TCGA data, aligned to the hg19 human reference genome is available from the GDC's legacy archive while the new "harmonized" data, realigned to hg38 is available from the GDC's main data portal.

Accessing The Cancer Genome Atlas Data on the Cloud

Besides accessing the files on the GDC Data Portal, you can also access them from the GDC Google Cloud Storage Bucket, which means that you don't need to download them to perform analysis. ISB-CGC stores the cloud file locations in tables in the `isb-cgc-bq.GDC_case_file_metadata` data set in BigQuery.

- To access these metadata files, go to the Google BigQuery console.
- Perform SQL queries to find the TCGA files. Here is an example:

```
SELECT active.*, file_gdc_url
FROM `isb-cgc-bq.GDC_case_file_metadata.fileData_active_current` as active, `isb-cgc-bq.
↪GDC_case_file_metadata.GDCfileID_to_GCSurl_current` as GCSurl
WHERE program_name = 'TCGA'
AND active.file_gdc_id = GCSurl.file_gdc_id
```

Accessing the TCGA Data in Google BigQuery

ISB-CGC has TCGA data, such as clinical, biospecimen, miRNA and RNA-seq, stored in Google BigQuery tables. Information about these tables can be found using the ISB-CGC BigQuery Table Search with TCGA selected for filter PROGRAM. To learn more about this tool, see the ISB-CGC BigQuery Table Search documentation.

The TCGA tables are in project isb-cgc-bq. To learn more about how to view and query tables in the Google BigQuery console, see the ISB-CGC BigQuery Tables documentation.

- Data set `isb-cgc-bq.TCGA` contains the latest tables for each data type.
- Data set `isb-cgc-bq.TCGA_versioned` contains previously released tables, as well as the most current table.

Note that some of the tables in the isb-cgc-bq project were migrated from the isb-cgc project. If you were using data sets `isb-cgc.TCGA_bioclin_v0`, `isb-cgc.TCGA_hg19_data_v0` and `isb-cgc.TCGA_hg38_data_v0`, they still exist but are deprecated.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

TCGA Radiology and Pathology Image Data Set

The TCGA images from The Cancer Imaging Archive (TCIA) as well as the pathology and diagnostic images previously available from the Cancer Digital Slide Archive (CDSA) are available in open-access Google Cloud Storage (GCS) buckets and can be explored through the ISB-CGC Web App.

Metadata for these files can be found in ISB-CGC Google BigQuery tables. Information about these tables can be found using the ISB-CGC BigQuery Table Search with TCGA selected for filter PROGRAM and FILE METADATA selected for filter CATEGORY. To learn more about this tool, see the ISB-CGC BigQuery Table Search documentation.

Radiology Images

Over 1.4 million radiology image files in DICOM format, grouped together into over 20,000 ZIP files are available in a GCS bucket called gs://isb-tcia-open/. Each ZIP file may contain hundreds of images or just a single image.

The BigQuery metadata table, `isb-cgc-bq.TCGA.radiology_images_tcia_current` contains the full URLs to these ZIP files, *e.g.*:

```
gs://isb-tcia-open/images/TCGA-GBM/TCGA-06-5413/TCIA.image.1.3.6.1.4.1.14519.5.2.1.4591.
↪4001.275342915307453440215680715165.zip
```

The metadata table also includes the patient identifier in TCGA "barcode" format, *e.g.* TCGA-06-5413 (which is also part of the GCS URL). Other information available in the table includes the body part examined, image modality, patient age, etc.

Pathology Images

Note: All tissue slide images from the TCGA program are currently unavailable for viewing. (Diagnostic images will display.)

Over 30,000 TCGA tissue slide images in SVS format, are also available in GCS, in the open-access bucket gs://gdc-tcga-phs000178-open/.

These files were uploaded from the GDC legacy archive.

The BigQuery metadata table, `isb-cgc-bq.TCGA.slide_images_gdc_current` contains the full URLs to these SVS files, *e.g.*:

```
gs://gdc-tcga-phs000178-open/9c4b1b5c-b5cf-48f6-bf41-047ceb8c883c/TCGA-CR-7365-01A-01-
↪TS1.811bb2b7-66e3-4694-891b-10b436ec300d.svs
```

as well as image metadata and the TCGA case and sample "barcode" which can be used to join this table with other TCGA clinical, biospecimen and molecular data tables.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

TRIO Data Set

About TRIO

The Ukrainian National Research Center for Radiation Medicine Trio Study contains epidemiologic data of trios of parents (exposed to the radiation from the Chernobyl accident) and their unexposed offspring. The purpose of the study is to investigate the transgenerational effects following nuclear accidents to understand the consequences of parental exposure to ionizing radiation.

About the TRIO Data

The TRIO data set includes whole genome sequencing (WGS) sequencing reads for 339 cases in the project TRIO-CRU.

For more information on TRIO data, please refer to the site below:

- GDC Data Portal

Accessing the TRIO Data on the Cloud

Besides accessing the files on the GDC Data Portal, you can also access them from the GDC Google Cloud Storage Bucket, which means that you don't need to download them to perform analysis. ISB-CGC stores the cloud file locations in tables in the `isb-cgc-bq.GDC_case_file_metadata` data set in BigQuery.

- To access these metadata files, go to the Google BigQuery console.
- Perform SQL queries to find the TRIO files. Here is an example:

```
SELECT active.*, file_gdc_url
FROM `isb-cgc-bq.GDC_case_file_metadata.fileData_active_current` as active, `isb-cgc-bq.
↪GDC_case_file_metadata.GDCfileID_to_GCSurl_current` as GCSurl
WHERE program_name = 'TRIO'
AND active.file_gdc_id = GCSurl.file_gdc_id
```

Accessing the TRIO Data in Google BigQuery

ISB-CGC has TRIO data, such as clinical and metadata, stored in Google BigQuery tables. Information about these tables can be found using the ISB-CGC BigQuery Table Search with TRIO selected for filter PROGRAM. To learn more about this tool, see the ISB-CGC BigQuery Table Search documentation.

The TRIO tables are in project isb-cgc-bq. To learn more about how to view and query tables in the Google BigQuery console, see the ISB-CGC BigQuery Tables documentation.

- Data set `isb-cgc-bq.TRIO` contains the latest tables for each data type.
- Data set `isb-cgc-bq.TRIO_versioned` contains previously released tables, as well as the most current table.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

VAREPOP Data Set

About the VA Research for Precision Oncology Program

The Research for Precision Oncology Program (RePOP) is a research activity that established a cohort of Veterans diagnosed with cancer and had genomic analyses performed on their tumor tissue as part of the standard of care. All data relevant to a patient's cancer and cancer care was collected under RePOP, including patient demographics, comorbidities, genomic analysis, treatments, medications, lab values, imaging studies, and outcomes. All RePOP participants signed/verbal informed consent and signed HIPAA authorization to have their data stored and shared from RePOP's Precision Oncology Program Data Repository (PODR).

About the VA Research for Precision Oncology Program Data

The VARePOP data set consists of 7 cases with somatic mutation and targeted sequencing data. The Genomic Data Commons currently has controlled access BAM and VCF files. The Project ID in the GDC Data Portal is VAREPOP-APOLLO.

For more information on the VAREPOP data, please refer to these sites:

- dbGaP site
- GDC Data Portal

Accessing the VA Research for Precision Oncology Program on the Cloud

Besides accessing the files on the GDC Data Portal, you can also access them from the GDC Google Cloud Storage Bucket, which means that you don't need to download them to perform analysis. ISB-CGC stores the cloud file locations in tables in the `isb-cgc-bq.GDC_case_file_metadata` data set in BigQuery.

- To access these metadata files, go to the Google BigQuery console.
- Perform SQL queries to find the VAREPOP files. Here is an example:

```
SELECT active.*, file_gdc_url
FROM `isb-cgc-bq.GDC_case_file_metadata.fileData_active_current` as active, `isb-cgc-bq.
↪GDC_case_file_metadata.GDCfileID_to_GCSurl_current` as GCSurl
WHERE program_name = 'VAREPOP'
AND active.file_gdc_id = GCSurl.file_gdc_id
```

Accessing the VAREPOP Data in Google BigQuery

ISB-CGC has VAREPOP data, such as clinical, stored in Google BigQuery tables. Information about these tables can be found using the ISB-CGC BigQuery Table Search with VAREPOP selected for filter PROGRAM. To learn more about this tool, see the ISB-CGC BigQuery Table Search documentation.

The VAREPOP tables are in project isb-cgc-bq. To learn more about how to view and query tables in the Google BigQuery console, see the ISB-CGC BigQuery Tables documentation.

- Data set `isb-cgc-bq.VAREPOP` contains the latest tables for each data type.
- Data set `isb-cgc-bq.VAREPOP_versioned` contains previously released tables, as well as the most current table.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

WCDT Data Set

About the Genomic Characterization of Metastatic Castration Resistant Prostate Cancer

The overarching goal of the Genomic Characterization of Metastatic Castration-Resistant Prostate Cancer study is to illuminate molecular mechanisms of acquired resistance to therapeutic agents, and particularly androgen signaling inhibitors, in the treatment of metastatic castration-resistant prostate cancer (mCRPC).

About the Genomic Characterization of Metastatic Castration Resistant Prostate Cancer Data

West Coast Prostrate Cancer Dream Team (WCDT) data is available from the biopsies of castration-resistant prostate cancer metastases collected during the study. The data consists of 101 cases with over 202 whole-genome sequencing files and 792 RNA sequencing files consisting of 83TB of data. The Project ID in the GDC Data Portal is WCDT-MCRPC.

For more information on the WCDT data, please refer to these sites:

- dbGaP site
- GDC Data Portal

Accessing Genomic Characterization of Metastatic Castration Resistant Prostate Cancer Data on the Cloud

Besides accessing the files on the GDC Data Portal, you can also access them from the GDC Google Cloud Storage Bucket, which means that you don't need to download them to perform analysis. ISB-CGC stores the cloud file locations in tables in the `isb-cgc-bq.GDC_case_file_metadata` data set in BigQuery.

- To access these metadata files, go to the Google BigQuery console.
- Perform SQL queries to find the WCDT files. Here is an example:

```
SELECT active.*, file_gdc_url
FROM `isb-cgc-bq.GDC_case_file_metadata.fileData_active_current` as active, `isb-cgc-bq.
↪GDC_case_file_metadata.GDCfileID_to_GCSurl_current` as GCSurl
WHERE program_name = 'WCDT'
AND active.file_gdc_id = GCSurl.file_gdc_id
```

Accessing the WCDT Data in Google BigQuery

ISB-CGC has WCDT data, such as clinical and RNA-seq, stored in Google BigQuery tables. Information about these tables can be found using the ISB-CGC BigQuery Table Search with WCDT selected for filter PROGRAM. To learn more about this tool, see the ISB-CGC BigQuery Table Search documentation.

The WCDT tables are in project isb-cgc-bq. To learn more about how to view and query tables in the Google BigQuery console, see the ISB-CGC BigQuery Tables documentation.

- Data set `isb-cgc-bq.WCDT` contains the latest tables for each data type.
- Data set `isb-cgc-bq.WCDT_versioned` contains previously released tables, as well as the most current table.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

10.1.2 From Proteomics Data Commons

PDC protein expression data are available in ISB-CGC BigQuery tables. The table below lists each Program.

Program	PDC AWS Cloud Storage	ISB-CGC BigQuery Tables	ISB-CGC Cohort Builder
CBTN		✓	
CPTAC		✓	
Georgetown Proteomics Research Program		✓	✓
ICPC		✓	
Quantitative Digital Maps of Tissue Biopsies		✓	

CBTN Data Set

About the Children's Brain Tumor Network

The Children's Brain Tumor Network (CBTN) seeks to innovate discoveries, to pioneer new treatments and to support open science to improve the health of every child and young adult diagnosed with a brain tumor. Previously, it was named the Children's Brain Tumor Tissue Consortium (CBTTC).

About the Children's Brain Tumor Network Data Set

CBTN has the Pediatric Brain Cancer Pilot Study available at the Proteomics Data Commons (PDC). ISB-CGC has procured this data through the PDC API.

Accessing the CBTN Data in Google BigQuery

ISB-CGC has CBTN data, such as clincial and protein expression, stored in Google BigQuery tables. Information about these tables can be found using the ISB-CGC BigQuery Table Search with CBTTC selected for filter PROGRAM. To learn more about this tool, see the ISB-CGC BigQuery Table Search documentation.

The CBTTC tables are in project isb-cgc-bq. To learn more about how to view and query tables in the Google BigQuery console, see the ISB-CGC BigQuery Tables documentation.

- Data set `isb-cgc-bq.CBTTC` contains the latest tables for each data type.
- Data set `isb-cgc-bq.CBTTC_versioned` contains previously released tables, as well as the most current table.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

Georgetown Proteomics Research Program Data Set

About the Georgetown Proteomics Research Program

The Georgetown Proteomics Research Program is part of the Georgetown Lombardi Comprehensive Cancer Center at Georgetown University.

About the Georgetown Proteomics Research Program Data Set

The Georgetown Lung Cancer Proteomics Study is available at the Proteomics Data Commons (PDC). ISB-CGC has procured data from this study through the PDC API.

Accessing the Georgetown Proteomics Research Program Data in Google BigQuery

ISB-CGC has Georgetown Proteomics Research Program (GPRP) clinical data stored in Google BigQuery tables. Information about these tables can be found using the ISB-CGC BigQuery Table Search with GEORGETOWN PROTEOMICS selected for filter PROGRAM. To learn more about this tool, see the ISB-CGC BigQuery Table Search documentation.

The GPRP tables are in project isb-cgc-bq. To learn more about how to view and query tables in the Google BigQuery console, see the ISB-CGC BigQuery Tables documentation.

- Data set `isb-cgc-bq.GPRP` contains the latest tables for each data type.
- Data set `isb-cgc-bq.GPRP_versioned` contains previously released tables, as well as the most current table.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

ICPC Data Set

About the International Cancer Proteogenome Consortium

The International Cancer Proteogenome Consortium (ICPC) is a voluntary scientific organization that provides a forum for collaboration among some of the world's leading cancer and proteogenomic research centers. Launched in late 2016, the ICPC includes researchers from over a dozen countries sharing data and results of proteogenomic analysis.

About the International Cancer Proteogenome Consortium Data Set

ICPC has several studies available at the Proteomics Data Commons (PDC). ISB-CGC has procured data for the following studies through the PDC API:

- Proteogenomics of Gastric Cancer
- HBV-Related Hepatocellular Carcinima
- Oral Squamous Cell Carcinoma Study
- Academia Sinica LUAD100

Accessing the ICPC Data in Google BigQuery

ISB-CGC has ICPC clinical and protein expression data stored in Google BigQuery tables. Information about these tables can be found using the ISB-CGC BigQuery Table Search with ICPC selected for filter PROGRAM. To learn more about this tool, see the ISB-CGC BigQuery Table Search documentation.

The ICPC tables are in project isb-cgc-bq. To learn more about how to view and query tables in the Google BigQuery console, see the ISB-CGC BigQuery Tables documentation.

- Data set `isb-cgc-bq.ICPC` contains the latest tables for each data type.
- Data set `isb-cgc-bq.ICPC_versioned` contains previously released tables, as well as the most current table.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

Quantitative Digital Maps of Tissue Biopsies Data Set

About the Quantitative Digital Maps of Tissue Biopsies Program

The program and its data are described in the paper Rapid mass spectrometric conversion of tissue biopsy samples into permanent quantitative digital proteome maps.

About the Quantitative Digital Maps of Tissue Biopsies Data Set

The PCT SWATH Kidney Study is available at the Proteomics Data Commons (PDC). ISB-CGC has procured data from this study through the PDC API.

Accessing the Quantitative Digital Maps of Tissue Biopsies Data in Google BigQuery

ISB-CGC has Quantitative Digital Maps of Tissue Biopsies clinical data stored in Google BigQuery tables. Information about these tables can be found using the ISB-CGC BigQuery Table Search with QUANT MAPS TISSUE BIOPSIES selected for filter PROGRAM. To learn more about this tool, see the ISB-CGC BigQuery Table Search documentation.

The tables are in project isb-cgc-bq. To learn more about how to view and query tables in the Google BigQuery console, see the ISB-CGC BigQuery Tables documentation.

- Data set `isb-cgc-bq.Quant_Maps_Tissue_Biopsies` contains the latest tables for each data type.
- Data set `isb-cgc-bq.Quant_Maps_Tissue_Biopsies_versioned` contains previously released tables, as well as the most current table.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

10.1.3 From Other Sources

Program	GDC Google Cloud Storage	ISB-CGC BigQuery Tables	ISB-CGC Cohort Builder
COSMIC	No, the COSMIC database is maintained by the Wellcome Sanger Institute, UK	Yes, COSMIC data is in BigQuery for registered users. Learn more about how to gain access to the COSMIC data here	
Pan-Cancer Atlas		✓	
HTAN		✓	

COSMIC Data Set

About the Catalog Of Somatic Mutations In Cancer

The Catalogue Of Somatic Mutations In Cancer (COSMIC) is the world's largest and most comprehensive resource for exploring the impact of somatic mutations in human cancer. The COSMIC tables in BigQuery are produced in collaboration with the Wellcome Trust Sanger Institute to provide a new way to explore and understand the mutations driving cancer.

About the Catalog Of Somatic Mutations In Cancer Data

The BigQuery data sets contain *all* of the CSV and TSV files available for download from the COSMIC Download page for versions 85-92 except version 88. ISB-CGC will not be hosting any version of COSMIC data past version 92 due to licensing costs.

Please explore the tables at (after registering for access):

- isb-cgc-bq.COSMIC
- isb-cgc-bq.COSMIC_versioned

Note: the project isb-cgc contains versions 85-91 except 88 and project isb-cgc-bq contains versions 92

Accessing the Catalog Of Somatic Mutations In Cancer Data

To access the BigQuery tables, you will need to link your Google identity with a COSMIC account.

- **New COSMIC User:** Register for a new COSMIC account. During registration, fill in the 'Google ID' field with your base* Google Identity.

A COSMIC account and academic use of the data is free, though commercial use of the COSMIC data is subject to licensing fees. Please review the COSMIC terms for more information.

- **Registered COSMIC User:** After logging in, navigate to the Account Settings page and fill in the 'Google ID' field with your base* Google Identity.

Once you have linked your Google identity to a COSMIC account, ISB-CGC will obtain your Google Identity. After a short delay, you will have "viewer" access to the COSMIC tables in BigQuery. You will then be able to view the data sets in the BigQuery UI under the `isb-cgc-bq` Google Cloud project and query the tables with your own Google Cloud Project.

We also have tutorials on using the COSMIC data sets with BigQuery in our Community Notebook Repository that you can check out.

- Intro to COSMIC in BigQuery Notebook (Python)
- Exploring the COSMIC Cancer Gene Census (Python)

If you are new to using ISB-CGC Google BigQuery data sets, see the Quickstart Guide to learn how to obtain a Google identity and how to set up a Google Cloud Project. Additionally, we offer free cloud credits for cancer research; you can find out more here.

If you can't successfully run a query or see the COSMIC tables under the `isb-cgc-bq project`, please verify that the Google ID you have provided is a valid Google account. If you are still unable to run a query or view the data sets under the `isb-cgc-bq` Google Cloud Project, please contact us at feedback@isb-cgc.org.

* *e.g. the base account tb@mylab.org might have a longer-form alias like thomas.brown@mylab.org*

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

Pan-Cancer Atlas BigQuery Data

The Pan-Cancer Atlas BigQuery data set was produced in collaboration with the TCGA research network, the GDC, and the NCI. This rich data set allows for an integrated examination of the full set of tumors characterized in the robust TCGA dataset and provides a new way to explore and analyze the processes driving cancer.

The availability of Pan-Cancer Atlas data in BigQuery enables easy integration of this resource with other public data sets in BigQuery, including other open-access datasets made available by the ISB-CGC (see this and that for more details on other publicly accessible BigQuery data sets).

About Pan-Cancer Atlas Data

The Pan-Cancer Atlas BigQuery tables (accessed here) mirror most of the files shared by the Pan-Cancer Atlas initiative on the GDC PanCanAtlas Publications page.

The tables are generally unmodified uploads of the files in the GDC Pan-Cancer Atlas. The Filtered_* tables were annotated as appropriate with ParticipantBarcode, SampleBarcode, AliquotBarcode, SampleTypeLetterCode, SampleType and TCGA Study. Subsequently the tables were filtered using the Pan-Cancer Atlas whitelist (which is the list of TCGA barcodes included in the Pan-Cancer Atlas). Two exceptions are the (public) *MC3 MAF file* and the *TCGA-CDR resource*.

Use of the tables starting with Filtered_* is recommended.

See examples of statistical Jupyter notebooks using the Pan-Cancer Atlas data here.

Adding the Pan-Cancer Atlas tables to your workspace

If you are new to using ISB-CGC Google BigQuery data sets, see the Quickstart Guide to learn how to obtain a Google identity and how to set up a Google Cloud Project.

To add public BigQuery data sets and tables to your "view" in the Google BigQuery Console you need to know the name of the GCP project that owns the dataset(s). To add the publicly accessible ISB-CGC datasets (project name: `isb-cgc-bq`) which includes the Pan-Cancer Atlas data set (dataset name: `pancancer_atlas`) follow these steps. (Note that these tables also exist in project `isb-cgc`, but that ISB-CGC is migrating current data to project `isb-cgc-bq`. If you are using the pancancer_atlas tables in `isb-cgc`, they are still available for you.)

You should now be able to see and explore all of the Pan-Cancer Atlas tables and also tables of other ISB-CGC data sets. Clicking on the blue triangle next to a dataset name will open it and show the list of tables in the data set. Clicking on a table name will open up information about the table in main panel, where you can view the Schema, Details, or a Preview of the table.

Additional projects with public BigQuery data sets which you may want to explore (repeating the same process will add these to your BigQuery side-panel) include genomics-public-data and google.com:biggene.

You can also search for and learn about Pan-Cancer Atlas tables through the ISB-CGC BigQuery Table Search UI. Type 'pancancer' in the **Search** box in the upper right-hand corner to filter for them.

Pan-Cancer Atlas BigQuery Query Example

Ready to query? Follow the steps below to run a query in the Google BigQuery Console. More details are here.

- Login to your Google account (Chrome is the preferred browser);
- Go to the BigQuery Console.

Let's query using the MC3 somatic mutation table.

- Click on **COMPOSE NEW QUERY** button.
- Paste the sample query below into the text-box.
- Within a second or two you should see a green circle with a checkmark below the lower right corner of the New Query text-box. – If instead you see a red circle with an exclamation mark, click on it to see what your Syntax Error is.
- Once you do have the green circle, you can click on it to see a message like: "Valid: This query will process 76.3 MB when run."
- To execute the query, click on **RUN**!

```
WITH
mutCounts AS (
  SELECT
     COUNT(DISTINCT( Tumor_SampleBarcode )) AS CaseCount,
     Hugo_Symbol,
     HGVSc
  FROM
     `isb-cgc-bq.pancancer_atlas.Filtered_MC3_MAF_V5_one_per_tumor_sample`
  GROUP BY
     Hugo_Symbol,
     HGVSc
),
mutRatios AS (
  SELECT
     HGVSc,
     Hugo_Symbol,
     CaseCount,
     (CaseCount/SUM(CaseCount) OVER (PARTITION BY Hugo_Symbol)) AS ratio
  FROM
     mutCounts
)
SELECT  *
FROM
```

(continues on next page)

(continued from previous page)

```
   mutRatios
WHERE
   CaseCount>=10
   AND ratio>=0.2
   AND HGVSc is not null
ORDER BY
   ratio DESC
```

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

HTAN Data Set

About HTAN

The Human Tumor Atlas Network (HTAN) is focused on transitions in cancer. Funded by the National Cancer Institute (NCI) Cancer Moonshot initiative, its mission is to construct three-dimensional atlases of the dynamic cellular, morphological, and molecular features of human cancers as they evolve from precancerous lesions to advanced disease (Cell April 2020). Many HTAN studies focus on single cell and multiplex imaging modalities.

About the HTAN Data

HTAN data encompasses at least 11 atlases and 17 primary tumor sites. Data (Release 2) was extracted from the HTAN Data Portal via Synapse (https://humantumoratlas.org/data-download).

To explore HTAN data, please see the HTAN Data Portal.

Accessing the HTAN Data in Google BigQuery

ISB-CGC has HTAN data, such as single cell RNA-Seq, clinical, biospecimen, and metadata, stored in Google BigQuery tables. Information about these tables can be found using the ISB-CGC BigQuery Table Search with HTAN selected for filter PROGRAM. To learn more about this tool, see the ISB-CGC BigQuery Table Search documentation.

The HTAN tables are in project isb-cgc-bq. To learn more about how to view and query tables in the Google BigQuery console, see the ISB-CGC BigQuery Tables documentation.

- Data set `isb-cgc-bq.HTAN` contains the latest tables for each data type.
- Data set `isb-cgc-bq.HTAN_versioned` contains previously released tables, as well as the most current table.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

10.2 Reference Data Sets

ISB-CGC hosts reference tables in BigQuery with information that describes or annotates human or other genomes, or is necessary to work with data generated by specific platforms.

10.2.1 Reference Data

ISB-CGC Hosted Reference Data

To facilitate working with the TCGA and other program data tables that the ISB-CGC is hosting in BigQuery, additional reference data tables have been created. Others are hosted by Google Cloud Life Sciences. Suggestions for more are welcome at feedback@isb-cgc.org.

For additional details about each of these tables, please use the BigQuery Table Search. To find the reference tables, select **Genomic Reference Database** under **Category**.

Genome Reference Data

Reference data that describes or annotates the human or other genomes is described in this section. Reference data hosted by the ISB-CGC in BigQuery tables are available in the `isb-cgc.genome_reference` data set. Tables based on gene-sets such as Ensembl and GENCODE can be used to find the genomic coordinates and identifiers for genes of interest, to perform queries that join tables with gene-symbol based data to tables with genomic-coordinate based data or tables that use other gene identifiers, for example.

Program/Source	Description
ClinVar	• ClinVar contains reports of the relationships among human variations and phenotypes. • GRCh37 • GRCh38
Cytoband/UCSC	• Cytoband to Genomic Coordinate Conversion • liftOver_hg19_to_hg38 - This table provides a mapping of each hg19 position to the corresponding position in hg38, and can be used to perform a liftOver operation in BigQuery.
dbSNP	• dbSNP contains human single nucleotide variations, microsatellites, and small-scale insertions and deletions along with publication, population frequency, molecular consequence, and genomic and RefSeq mapping information for both common variations and clinical mutations • B150 GRCH37P13 • B151 GRCH37P13
Ensembl	• GRCh37: Release 75, the final build of the Ensembl gene-set mapped to GRCh37 • GRCh38: Release 87, the most recent Ensembl gene-set mapped to GRCh38
GENCODE	• GRCh37: Release 19, the final build of the GENCODE gene-set mapped to GRCH37 • GRCh38: Releases 22, 23, and 24 from GENCODE are all available (because the TCGA data has been reprocessed by at least one center using each of these three different releases)
Gene Ontology Consortium	• Tables based on GO annotations and the GO ontology.
Genome-Wide SNP Array	• The technical documentation for the Affymetrix Genome-Wide Human SNP Array 6.0 array can be found here.
gnomAD	• gnomAD aggregates and harmonizes both exome and genome sequencing data from a wide variety of large-scale sequencing projects. • GRCH37
ICD	• International Classification of Diseases • ICD-10 Chapters • ICD-10 Codes • ICD-O-3 Morphology • ICD-O-3 Site
Infinium	• Infinium EPIC HG19 and HG38 Manifests • Infinium HM27 HG19 and HG38 Manifests • Infinium HM450 HG19 and HG38 Manifests

Platform Reference Data

Some reference data is necessary to work with data generated by specific platforms such as the Illumina DNA Methylation array. The platform_reference data set contains information on the Illumina DNA Methylation Platform.

Program/Source	Description
GDC	• HG38 DNA Methylation - Most of the DNA Methylation data produced by the TCGA project was obtained using the Illumina Infinium HumanMethylation450 (aka 450k) BeadChip array. Some of the earlier tumor types were assayed on the older, 27k array.
Infinium	• Illumina DNA Methylation Annotation - Platform annotation information has been uploaded into BigQuery; each CpG locus is uniquely identified as described in this technical note and this unique identifier can be used to look up and cross-reference data between the TCGA DNA methylation data table and the platform annotation table.
Cytoband/UCSC	• DNA Methylation Annotation Liftover to HG38 Coordinates - The original Illumina-provided CpG coordinates have been *"lifted over"* from hg19 to hg38.

Genotype Tissue Expression (GTEx) Project Data

The GTEx_v7 data set contains tables with molecular and clinical data (gene read, gene expression, sample attributes, subject phenotype) loaded from the Genotype-Tissue Expression (GTEx) Project Data Portal on November 2017. See the GTEx Portal for more information.

University of California Santa Cruz (UCSC) TOIL RNA-seq recompute project Data

The Toil_recompute data set contains data made available by the UCSC TOIL RNA-seq recompute project. The goal of the project was to process ~20,000 RNA-seq samples to create a consistent meta-analysis of four datasets free of computational batch effects. This is best used to compare TCGA cohorts to TARGET or GTEx cohorts. For more details, see the Zena Browser Data Pages.

Other Reference Data Sources

Google Cloud Life Sciences maintains a list of publicly available data sets, including **Reference Genomes**, the **Illumina Platinum Genomes**, information about the **Tute Genomics Annotation** table, *etc*.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

10.3 File Metadata Data Sets

ISB-CGC hosts metadata tables in BigQuery with information that points to the raw and processed cancer data in the NCI GDC Google Cloud Storage buckets.

10.3.1 Case and File Metadata

The ISB-CGC hosts several metadata tables in Google BigQuery to help users find GDC files in Google Cloud Storage (GCS) or PDC files in Amazon Web Services (AWS) cloud storage. Preview and query these tables from the BigQuery web UI or scripting languages such as R and Python, or the command-line using the cloud SDK utility bq.

For additional details about each of these tables, please use the BigQuery Table Search. To find the metadata tables, select **File Metadata** under **Category**.

Below, the '#' represents the GDC release number and should be replaced by it when using the tables, for example: *isb-cgc-bq.GDC_case_file_metadata_versioned.caseData_r28*. The metadata is split up into several tables per GDC release as follows in the isb-cgc-bq project. (Older metadata is in the isb-cgc project and follows a slightly different table naming format.)

Table	Description
caseData_r#	List of all of the cases in GDC
fileData_active_r#	List of the currently active cases in GDC along with information related to those cases
fileData_legacy_r#	Same as the previous table but with legacy data instead
aliquot2caseIDmap_r#	"helper" table to map between identifiers at different levels of aliquot data. The intrinsic hierarchy is program > project > case > sample > portion > analyte > aliquot
slide2caseIDmap_r#	"helper" table to map between identifiers at different levels of tissue slide data. The intrinsic hierarchy is program > project > case > sample > portion > slide
GDCfileID_to_GCSurl_r#	Gives the Google Cloud Storage location for each file
per_sample_file_metadata_hg19_gdc_r# or per_sample_file_metadata_hg38_gdc_r#	Provides file ids and other metadata for samples. Information is stored in these tables by program and these tables are in the respective program data set.

PDC metadata file and case metadata are stored in data sets isb-cgc-bq.PDC_metadata_versioned and isb-cgc-bq.PDC_metadata.

Table	Description
file_associated_entity_mapping_V#	List of PDC entitites mapped to cases and file IDs
file_metadata_V#	Gives the AWS location for each file, study information, as well as an embargo date if it applies

For examples of querying the metadata tables, please see the ISB-CGC Community Notebook GitHub Repository.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

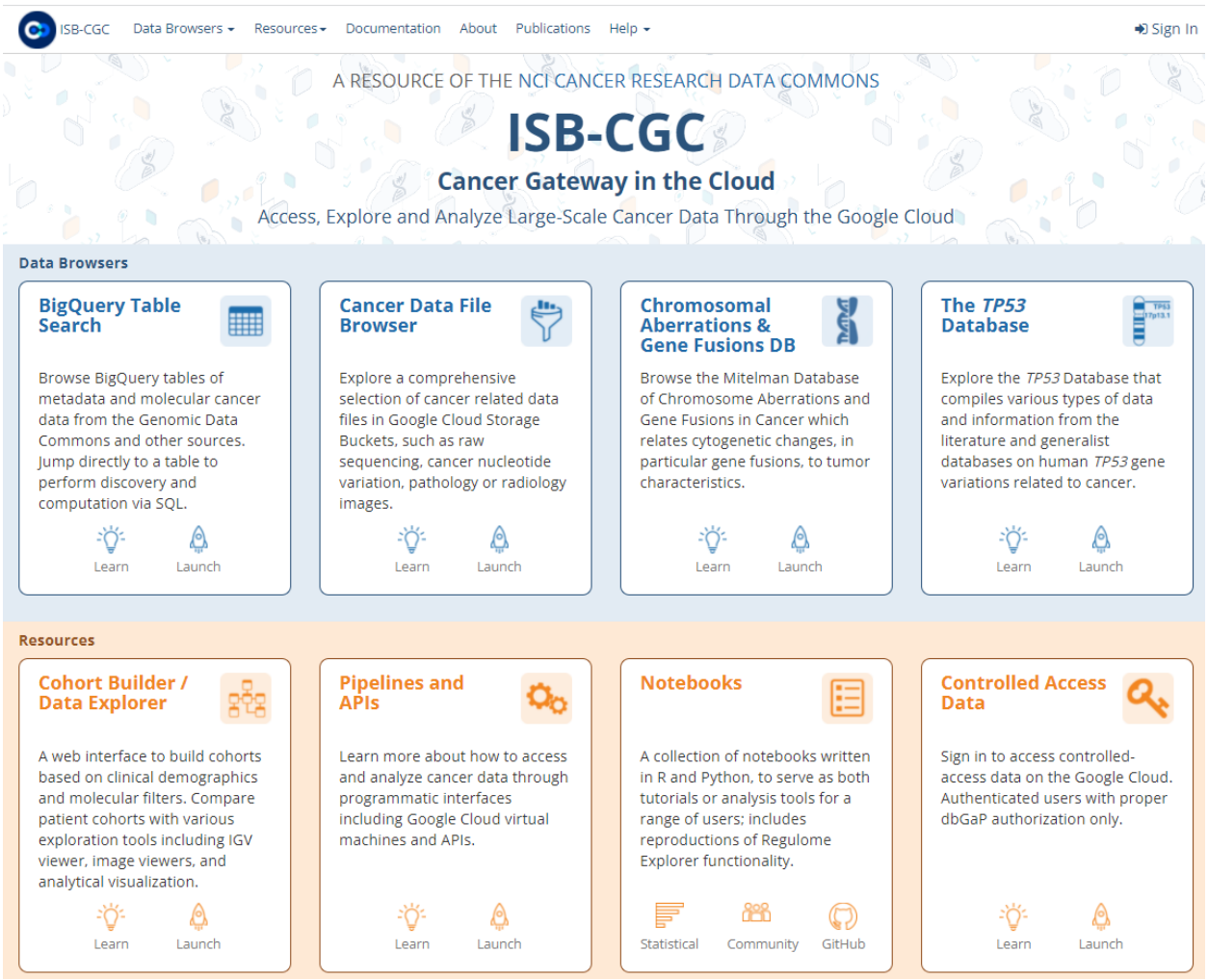
CHAPTER

ELEVEN

ISB-CGC BIGQUERY TABLE SEARCH

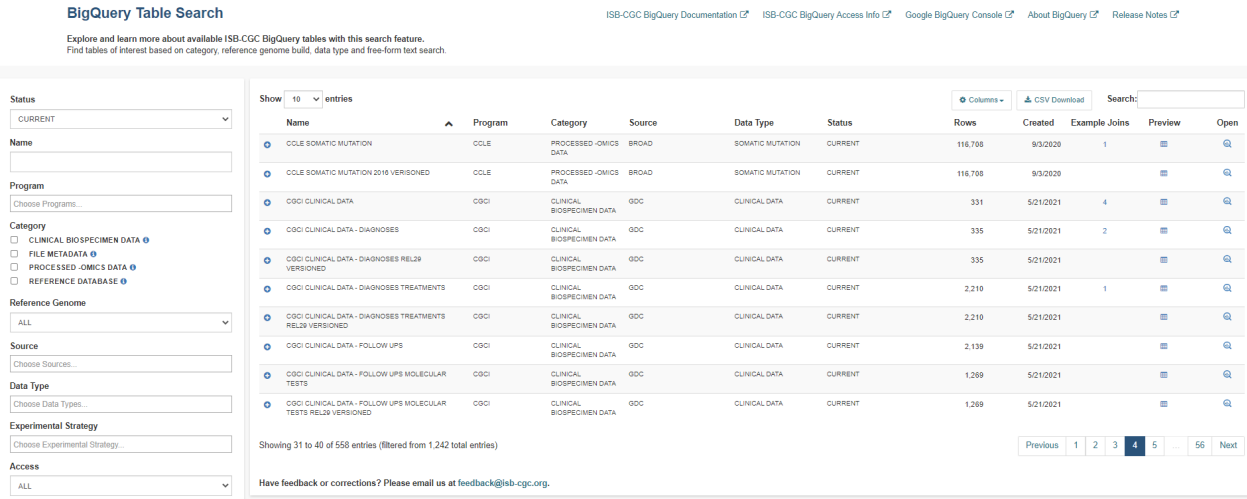
The ISB-CGC BigQuery Table Search UI (https://isb-cgc.appspot.com/bq_meta_search/) is a discovery tool that allows users to explore and search for ISB-CGC hosted BigQuery tables. It can be accessed directly from the ISB-CGC homepage (https://isb-cgc.org/) by clicking on **Launch** in the **BigQuery Table Search** box or selecting **BigQuery Table Search** from the **Data Browsers** drop down menu on the main menu bar.

Note: Users are not required to have a Google Cloud Platform (GCP) project or an account to learn more about the tables hosted by ISB-CGC.



Currently, ISB-CGC hosts open access BigQuery tables containing data for over 25 research programs and for over 15

data types. Each table has been curated to include detailed table and field descriptions as well as table labels allowing users to search for BigQuery tables of interest using a free-form text search or via available filters.



Links to various helpful documentation pages are available, including Google BigQuery's documentation and ISB-CGC's BigQuery documentation and the ISB-CGC Release Notes.

11.1 Filters

The search filters consist of a combination of multi-select dropdown lists, checkboxes and free-form text fields.

Selecting multiple items within a multi-select dropdown list or checkbox filter will perform a Boolean "AND" on those selections and bring back any data that match any of the selected items. For example, selecting Data Type BIOSPECIMEN and CLINICAL will display both biospecimen and clinical data.

Selecting multiple filters will perform a Boolean "OR" on those selections and bring back only data that fits all criteria. For example, selecting Data Type BIOSPECIMEN and Source of CCLE will only display CCLE biospecimen data.

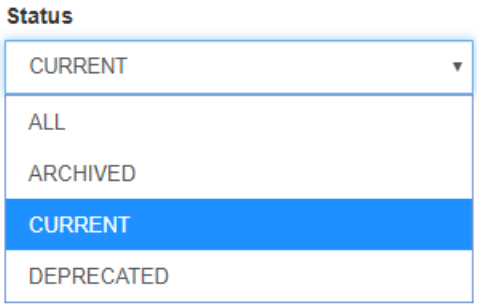
Information about each filter is detailed below.

Status

We are committed to providing the most up-to-date information in our BigQuery tables but realize that at times researchers need to reference older versions of data. Each table is assigned a status based on the following criteria:

- Tables with the most up-to-date available information are given a status of **current**
- Tables with older versions of data are given a status of **archived**
- Tables that have data that is no longer supported are **deprecated**

By default, the Status filter is set to Current.



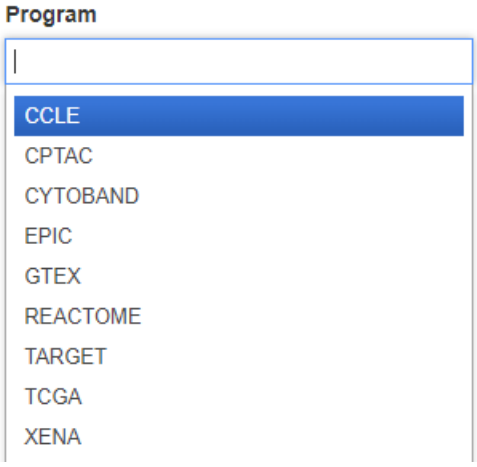
Name

The **Name** filter is a free-form text field; the user can type all or a portion of the name into the field to perform the search. It will match against the Name column.

Note that this Name field is not the Table ID (which is used in SQL queries) but is a **Friendly Name**; that is, a descriptive, user-friendly name for the table.

Program

Filter the BigQuery tables by programs such as CCLE, TARGET and TCGA by using the **Program** filter. Click the Program box to see the dropdown list and click on a program to select it. Additional programs can be selected by clicking in the Program box again.

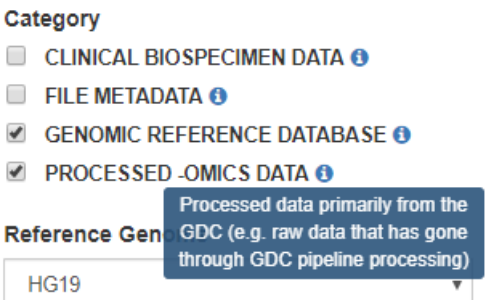


Categories

The tables are grouped into four high-level categories:

- **Clinical Biospecimen Data**: Patient case and sample information (includes clinical tables with patient demographic data, and biospecimen data with detailed sample information)
- **File Metadata**: Information about raw data files including Google Cloud Storage Paths (includes tables with information about files available at the GDC, including GCS paths, creation dates, sizes, etc.)
- **Genomic Reference Database**: Genomic information that can be used to cross-reference against processed-omics data tables (examples include COSMIC, ClinVar, cytoBand, dbSNP, Ensembl, Ensembl2Reactome)
- **Processed-omics Datasets**: Processed data primarily from the GDC (i.e. raw data that has gone through GDC pipeline processing e.g. gene expression, miRNA expression, copy number, somatic mutations, methylation)

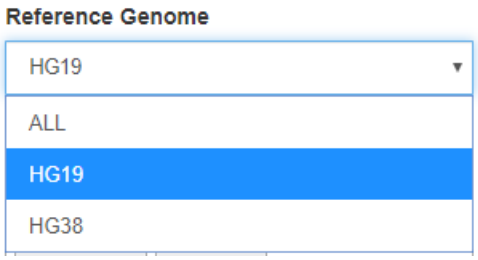
Click on one or more checkboxes to select categories. Hovering the cursor over the information icon will display a short description of the category.



Reference Genome Build

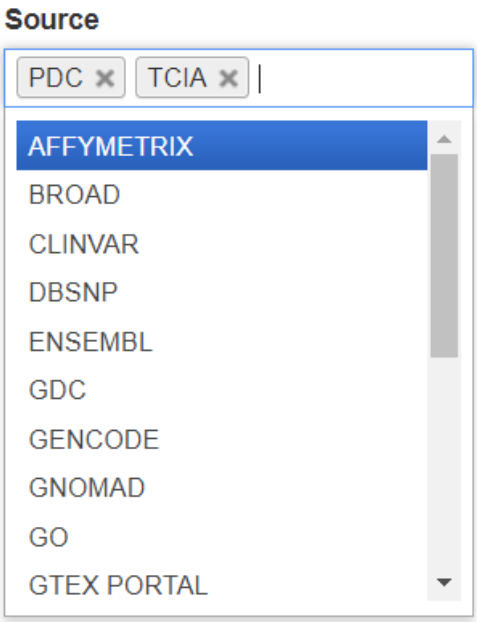
Filter for tables that contain data for hg19 or hg38. In a few cases, there are tables which contain information from both genome builds; for example, tables that include liftover coordinates between the reference builds.

By default, the **Reference Genome** filter is set to ALL.



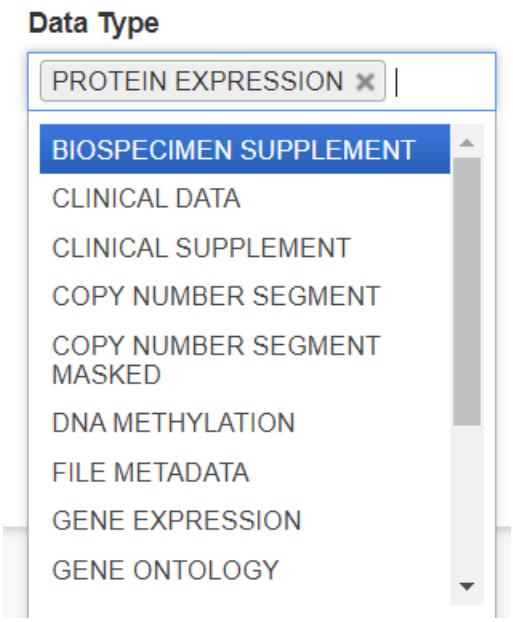
Source

Search through the sources of the data in our BigQuery tables by using the **Source** filter. Click the Source box to see the dropdown list and click on a source to select it. Additional sources can be selected by clicking in the Source box again.



Data Type

The **Data Type** filter also allows you to filter for data types of interest. Like Source, multiple Data Types can be selected.

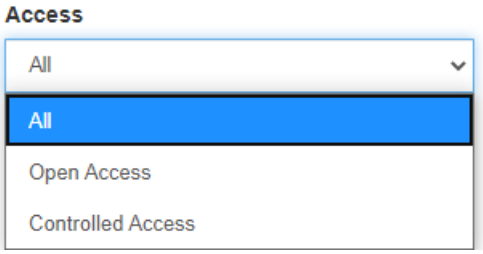


Experimental Strategy

The **Experimental Strategy** filter also allows you to filter for experimental strategies of interest. Multiple Experimental Strategies can be selected.

Access

The **Access** filter has options of All, Open Access and Controlled Access. Controlled Access tables will be displayed with a Lock icon to the right of the table name. Controlled Access data cannot be previewed, but can be opened in the Google BigQuery Console, if the user has the required permissions.



More Filters

The **Show More Filters** button can be used to display **BQ Project**, **BQ Dataset**, **BQ Table**, **Table Description**, **Labels** and **Field Name** filters. Except for BQ Project, these are free-form text fields; the user can type all or a portion of the name into the field to perform the query. For instance, for all datasets which have "alpha" in the name, type "alpha" into the field.

These fields are most useful for users already familiar with the BigQuery tables.

Labels

Each table was tagged with labels relating to the status, program, reference genome build, source, data type, experimental strategy and access. Users can search on any of these labels on the Labels filter field. Users can find the **Labels** search filter under the **Show More Filters** option.

The labels for a table can be viewed when the blue plus sign (+) to the left of the table row is clicked. See the screen shot in the Schema section below.

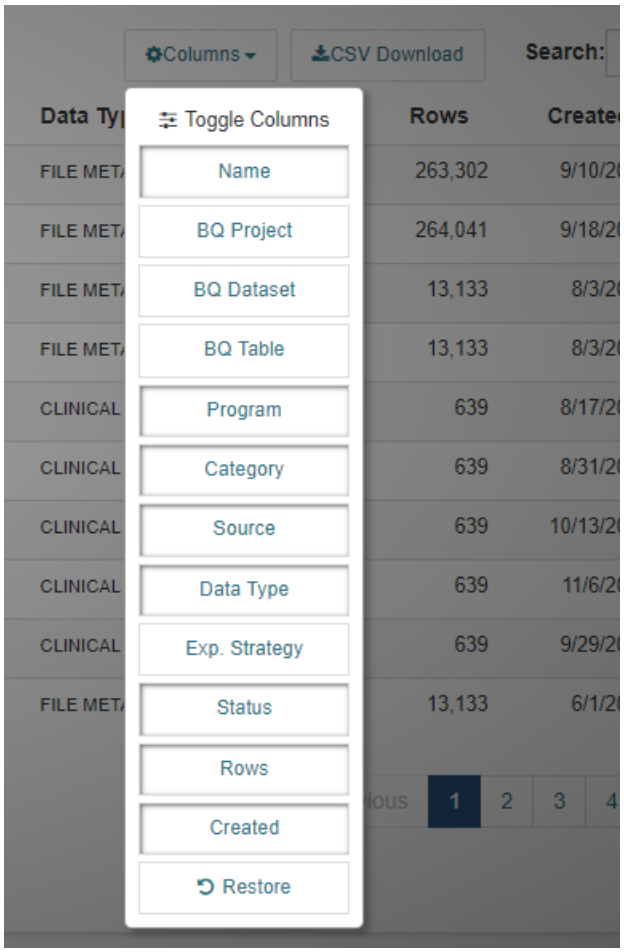
11.2 Search Results

By default, each row will display the Name, Category, Source, Data Type, Status, number of rows, and Created Date of the table.

Click on the column header to sort the displayed results by that column.

Columns Selector

Columns can be added or removed from the display by using the Columns selector. For instance, the Bq Project, BQ Dataset and BQ Table are not initially displayed, but they can be added to the display.



Search Box

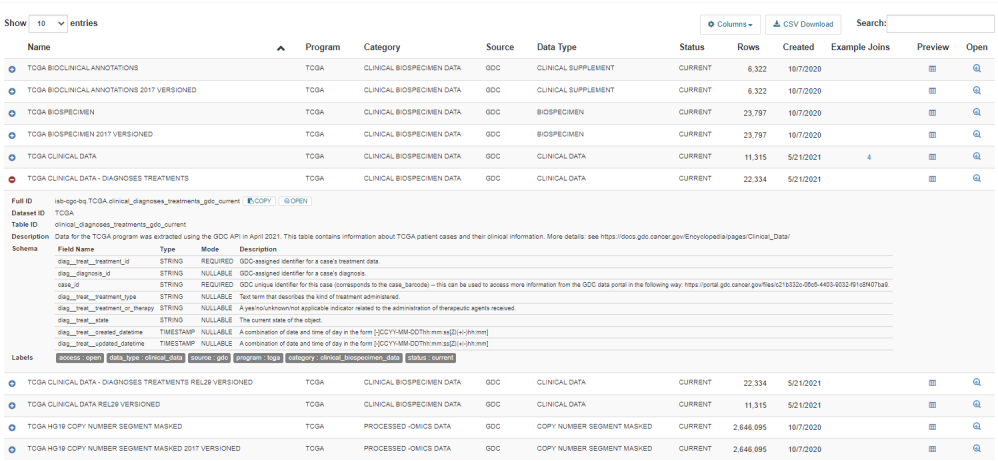
To further filter the results, use the **Search** box above the results, on the right-hand side. This is a free-form text field; the user can type all or a portion of the search item into the field to perform the query. This searches all fields in the table.

Export

To export the results of your search to a file in Comma Separated Values (CSV) format, click the **CSV Download** button.

11.2.1 Schema Description

For detailed table information, click on the blue plus sign (+) on the left-hand side.



The following information is displayed:

- **Full ID** - This is the Project, Dataset ID, and Table ID concatenated with periods between them. The Full ID is used in SQL queries.
- **Dataset ID** - The BigQuery dataset of the table. A data set is a group of related tables.
- **Table ID** - The BigQuery table ID.
- **Description** - A description of the table, which includes information such as how the data was created, its source, data type, and contents.
- **Schema** - The schema displays the Field Name, Type, Mode and Field Description for each field in the table.
- **Labels** - Labels are table metadata describing the source, data type, reference genome build, status, and access of the table data.

Copy button

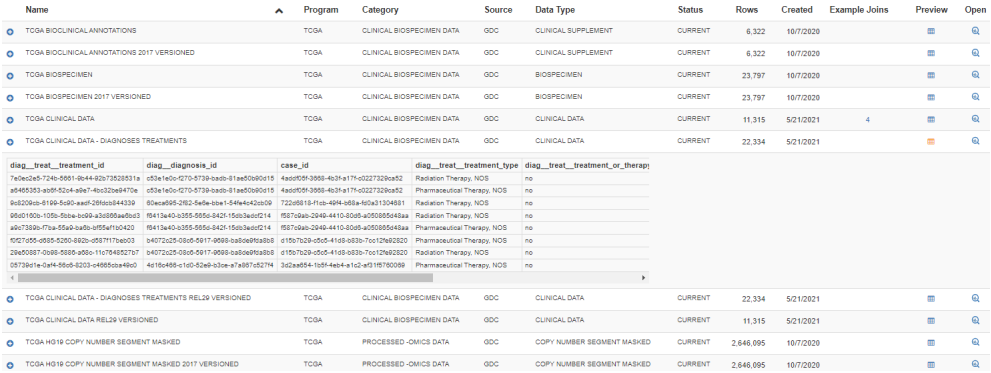
Next to the Full ID is a **Copy** button. When the user clicks this, the Full ID is copied to the clipboard. The Full ID can then be pasted into an SQL query within the BigQuery Query editor.

Open button

Next to the Copy button is an **Open** button. Clicking on this button opens the table in the BigQuery Google Cloud Platform Console. For more details, see the **Table Access in Google BigQuery** section below.

11.2.2 Table Preview

A few rows of the data in a BigQuery table can be viewed by clicking on the **Preview** button on the right-hand side. This feature allows the user to get a better idea of the contents and format of the data.



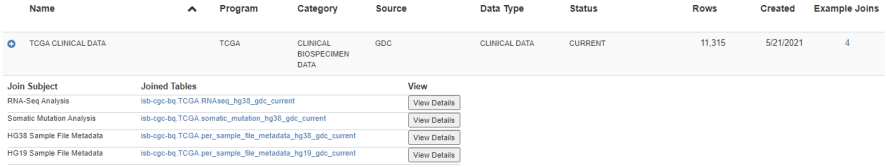
Name	Program	Category	Source	Data Type	Status	Rows	Created	Example Joins	Preview	Open
TCGA BIOCLINICAL ANNOTATIONS	TCGA	CLINICAL BIOSPECIMEN DATA	GDC	CLINICAL SUPPLEMENT	CURRENT	6,322	10/7/2020			
TCGA BIOCLINICAL ANNOTATIONS 2017 VERSIONED	TCGA	CLINICAL BIOSPECIMEN DATA	GDC	CLINICAL SUPPLEMENT	CURRENT	6,322	10/7/2020			
TCGA BIOSPECIMEN	TCGA	CLINICAL BIOSPECIMEN DATA	GDC	BIOSPECIMEN	CURRENT	23,797	10/7/2020			
TCGA BIOSPECIMEN 2017 VERSIONED	TCGA	CLINICAL BIOSPECIMEN DATA	GDC	BIOSPECIMEN	CURRENT	23,797	10/7/2020			
TCGA CLINICAL DATA	TCGA	CLINICAL BIOSPECIMEN DATA	GDC	CLINICAL DATA	CURRENT	11,315	5/21/2021	4		
TCGA CLINICAL DATA - DIAGNOSES TREATMENTS	TCGA	CLINICAL BIOSPECIMEN DATA	GDC	CLINICAL DATA	CURRENT	22,334	5/21/2021			

diag__treat__treatment_id	diag__diagnosis_id	case_id	diag__treat__treatment_type	diag__treat__treatment_or_therap
7e9ec2e5-724b-5661-9b44-92b73528531a	c53e1e0c-f270-5739-badb-81ae50b90d15	4addf05f-3668-4b3f-a17f-c0227329ca52	Radiation Therapy, NOS	no
a6465353-ab5f-52c4-a9e7-4bc32be9470e	c53e1e0c-f270-5739-badb-81ae50b90d15	4addf05f-3668-4b3f-a17f-c0227329ca52	Pharmaceutical Therapy, NOS	no
9c8209cb-8199-5c90-aacf-26fdcb844339	60eca695-2f62-5e6e-bbe1-54fe4c42cb09	722d5818-f1cb-49f4-b68a-fd0a31304881	Radiation Therapy, NOS	no
96d0160b-105b-5bbe-bc99-a3d869ae6bd3	f8413e40-b355-565c-842f-15db3edcf214	f587c9ab-2940-4410-80d5-a050865d48aa	Radiation Therapy, NOS	no
a9c7389b-f7ba-55a9-ba6b-bf55ef1b0420	f8413e40-b355-565c-842f-15db3edcf214	f587c9ab-2940-4410-80d5-a050865d48aa	Pharmaceutical Therapy, NOS	no
f0f27c55-d685-5250-992c-d587f17beb03	b4072c25-08c6-5917-9699-ba6ce9fda9b8	d15b7b29-c5c5-41d8-b83b-7cc12fe92820	Pharmaceutical Therapy, NOS	no
29e50887-0b98-5b95-a56c-11c7d4852757	b4072c25-08c6-5917-9699-ba6ce9fda9b8	d15b7b29-c5c5-41d8-b83b-7cc12fe92820	Radiation Therapy, NOS	no
0573b01e-0af4-56c6-5203-c4665cca49c0	4d16c468-c1d0-52e9-b3ce-a7a987c52174	3d2aa554-1b5f-4ea4-a1c2-af31f5780069	Pharmaceutical Therapy, NOS	no

Name	Program	Category	Source	Data Type	Status	Rows	Created	Example Joins
TCGA CLINICAL DATA - DIAGNOSES TREATMENTS REL29 VERSIONED	TCGA	CLINICAL BIOSPECIMEN DATA	GDC	CLINICAL DATA	CURRENT	22,334	5/21/2021	
TCGA CLINICAL DATA REL29 VERSIONED	TCGA	CLINICAL BIOSPECIMEN DATA	GDC	CLINICAL DATA	CURRENT	11,315	5/21/2021	
TCGA HG19 COPY NUMBER SEGMENT MASKED	TCGA	PROCESSED -OMICS DATA	GDC	COPY NUMBER SEGMENT MASKED	CURRENT	2,646,095	10/7/2020	
TCGA HG19 COPY NUMBER SEGMENT MASKED 2017 VERSIONED	TCGA	PROCESSED -OMICS DATA	GDC	COPY NUMBER SEGMENT MASKED	CURRENT	2,646,095	10/7/2020	

11.2.3 Example Joins

The **Example Joins** column specifies the number of example SQL join queries, for the table on that row, which are provided by the BigQuery Table Search. Clicking on the number will display a list of the examples.



Name	Program	Category	Source	Data Type	Status	Rows	Created	Example Joins
TCGA CLINICAL DATA	TCGA	CLINICAL BIOSPECIMEN DATA	GDC	CLINICAL DATA	CURRENT	11,315	5/21/2021	4

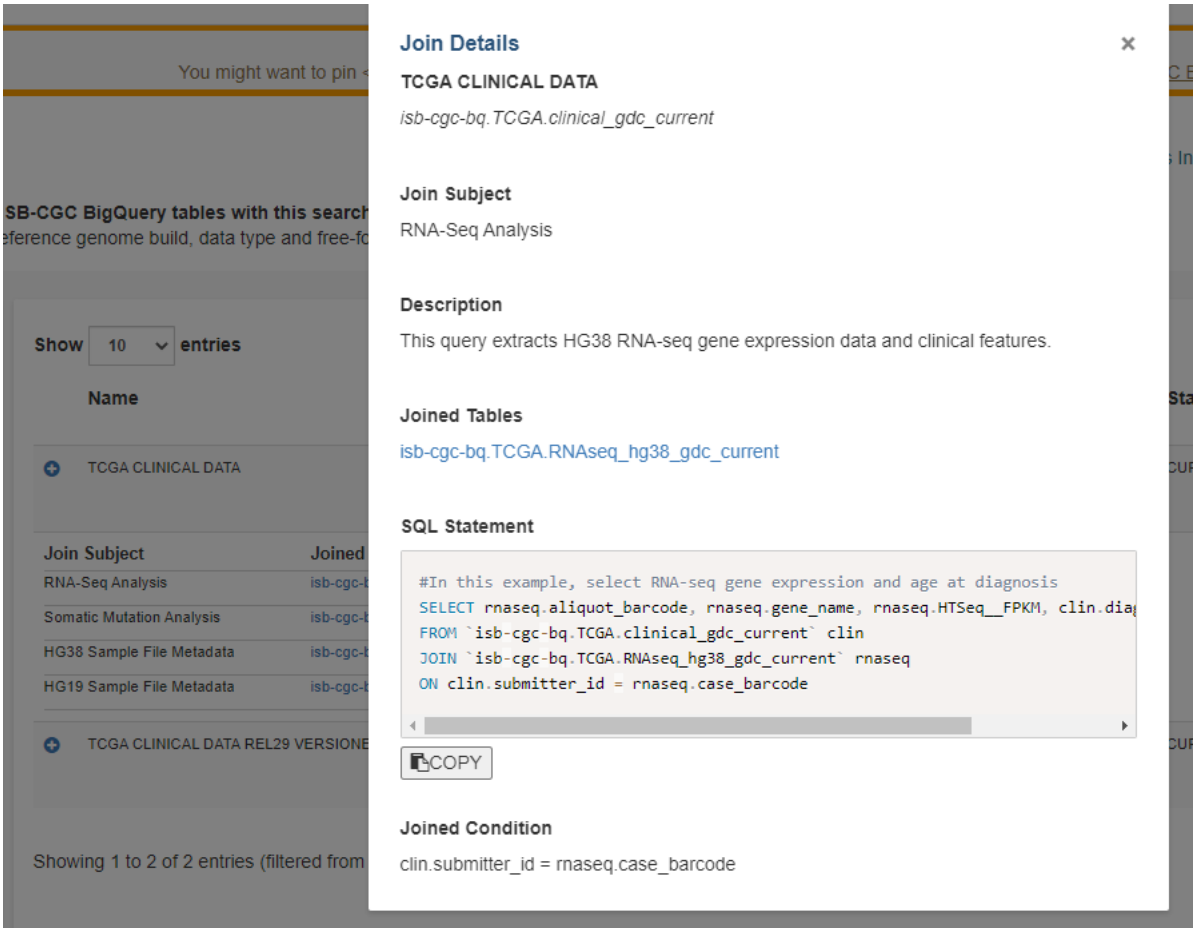
Join Subject	Joined Tables	View
RNA-Seq Analysis	isb-cgc-bq.TCGA.RNAseq_hg38_gdc_current	View Details
Somatic Mutation Analysis	isb-cgc-bq.TCGA.somatic_mutation_hg38_gdc_current	View Details
HG38 Sample File Metadata	isb-cgc-bq.TCGA.per_sample_file_metadata_hg38_gdc_current	View Details
HG19 Sample File Metadata	isb-cgc-bq.TCGA.per_sample_file_metadata_hg19_gdc_current	View Details

The following information is displayed:

- **Join Subject** - This is the topic of the query.
- **Joined Tables** - Here, the tables which are joined with the table in the row are listed.
- **View** - The View Details button takes you to a screen which displays the SQL statement and a more detailed description of the query.

Join Details

Clicking on the **View** button displays the **Join Details** screen.



The following information is displayed:

- **Table identification** - Both the table name and the table Full ID are displayed.
- **Join Subject** - This is the topic of the query.
- **Description** - Here, the query is described in more detail. For instance, it will describe what kind of data is extracted.
- **Joined Tables** - Tables which are joined with the main table of interest are listed here. The table name is also a link, in case you would like to easily learn more about the joined table. Clicking on it will open up the ISB-CGC BigQuery Search in another tab, with the table information in the query results.
- **SQL Statement** - This is the SQL statement for the joined tables.
- **COPY** - Clicking this button copies the SQL Statement to your clipboard. You can then directly copy the SQL query into the Google Cloud Platform BigQuery Console, a Jupyter notebook, or anywhere that you would like. These queries can be run as they are, or you can tailor them to your needs.
- **Joined Condition** - There are the fields being joined between the tables.

11.3 Table Access in Google BigQuery

To access the BigQuery tables in Google Cloud Console directly from the Table Search UI, simply click on the **Open** button on the right-hand side.

Note:

- If you have previously accessed the Google Cloud Platform and have a Google Cloud Platform project already set up, this button will automatically open up the table in the Google BigQuery Console as depicted in the image below.
- If you have never accessed Google Cloud Platform, you will be presented with a Google login page. You can use any Google ID to log in. Instructions on how to create a Google identity if you don't already have one can be found here. You will be prompted to create a project, free of charge. Once you create the project, you will be directed to the BigQuery table you wished to open in the Google BigQuery Cloud Platform Console.

Google Cloud Platform's free tier allows users to access many common Google Cloud resources including BigQuery free of charge and query up to 1 TB of data per month for free.

Please see the following ISB-CGC documentation pages for guidance:

- How to create a Google Cloud Platform (GCP) project
- How to link ISB-CGC BigQuery tables to your Google Cloud Platform (GCP) project

---

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CHAPTER

TWELVE

ISB-CGC BIGQUERY TABLES

Google BigQuery (BQ) is a massively-parallel analytics engine ideal for working with tabular data. Leveraging the power of BigQuery, we have made the information scattered over tens of thousands of XML and tabular data files in legacy and active archives at the NCI GDC and PDC much more accessible in the form of open-access BigQuery tables.

We have made the ability to explore and learn more about the ISB-CGC hosted BigQuery tables easy via an interactive BigQuery Table Search UI (https://isb-cgc.appspot.com/bq_meta_search/). Users can find tables of interest based on program, category, reference genome build, data type and free-form text search.

Using SQL in the Google BigQuery Console, in Juypter notebooks or in R, users with Google Cloud Platform (GCP) projects can analyze patient, biospecimen, and molecular data for many cancer programs such as TCGA, TARGET, CCLE, GTEx from ISB-CGC's BigQuery tables.

Note that dbGaP authorization is not required to access most tables.

Additional Support

For more information about Google BigQuery, see the following Google support pages:

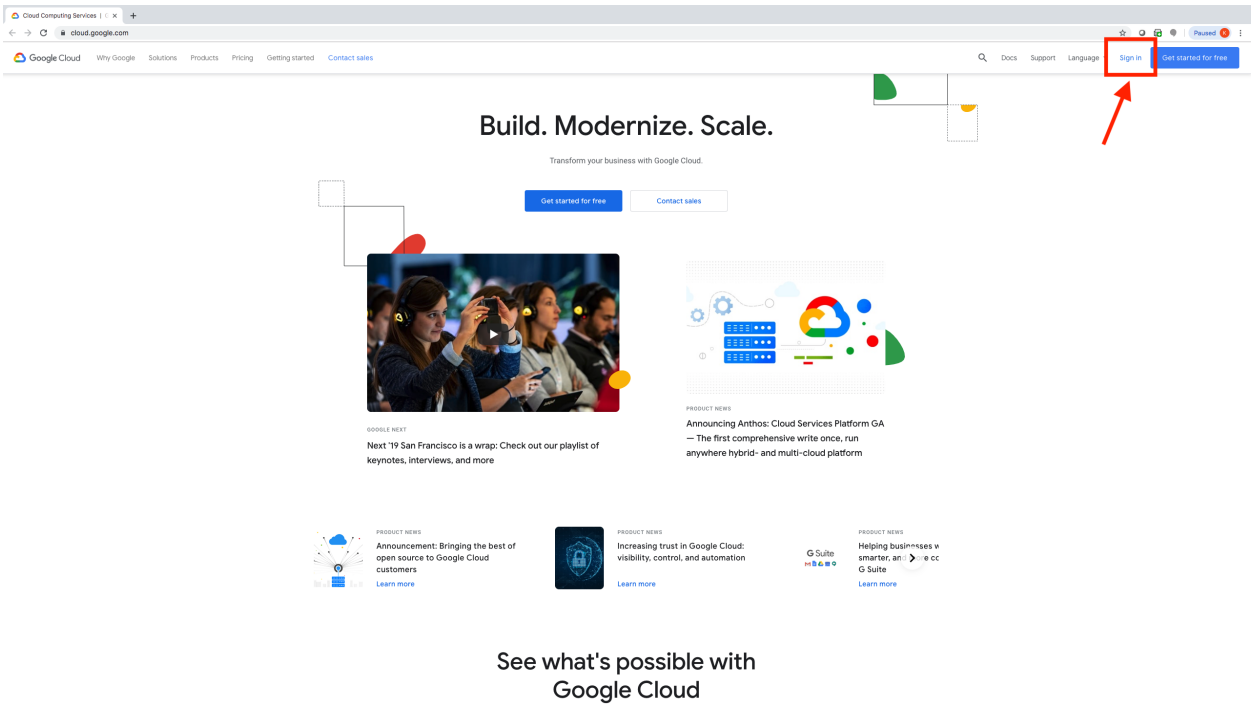
- Google BigQuery documentation
- What is BigQuery?

12.1 BigQuery on Google Cloud Platform

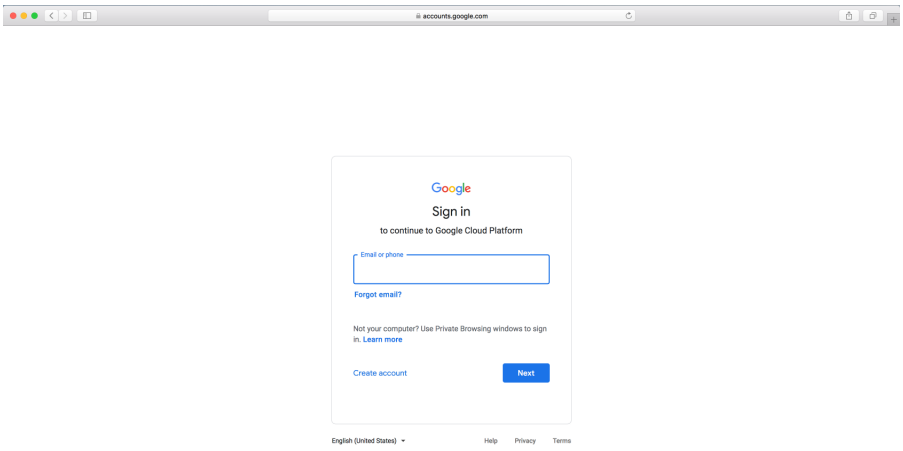
In order to use BigQuery, you must have access to a Google Cloud Platform (GCP) project. Your GCP project must be associated with a billing account in order to gain full access to all of products and services that make up the Google Cloud. Contact us at **request-gcp@isb-cgc.org** for more information on how to to request cloud credits.

Additionally, you will need a Google account identity (freely available with a new account or by linking to an existing email account).

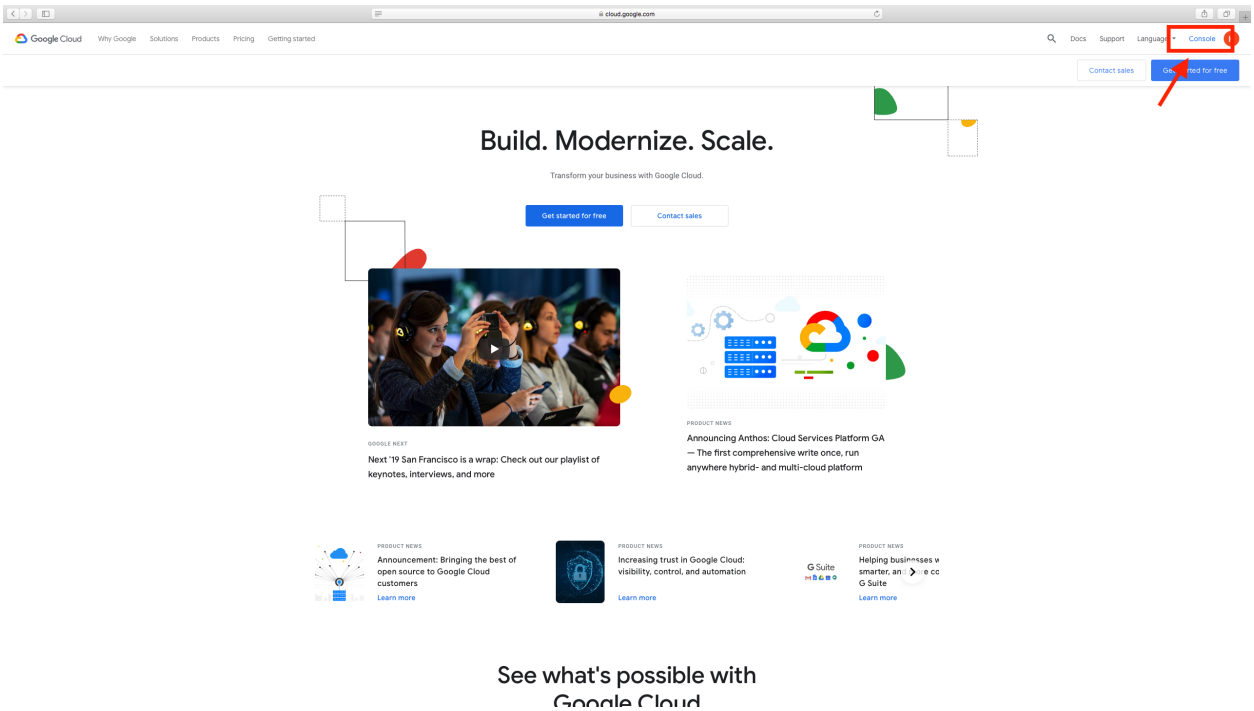
When first logging into the Google Cloud Platform, you will be presented with this page:



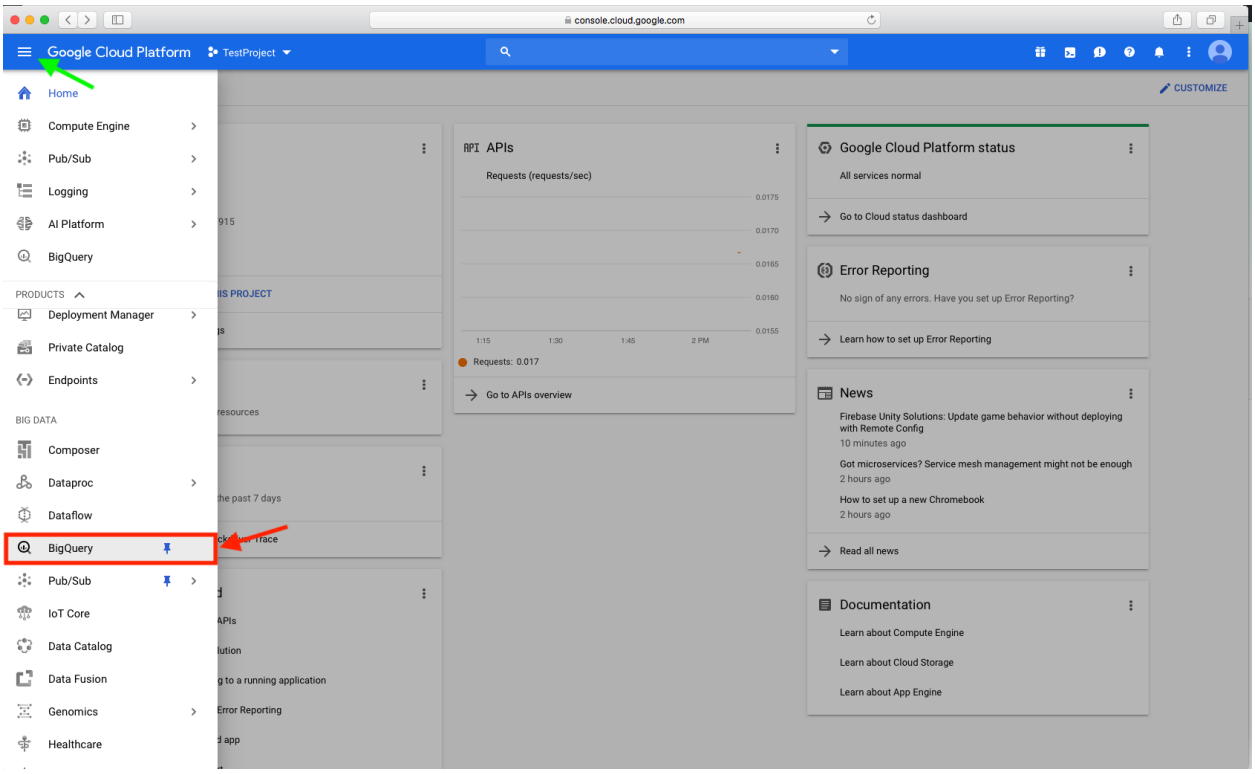
You will be presented with the sign in page, prompting you to enter a Google account log in and password:



Once you sign in, click on Console at the top of the screen (see arrow in image below) to access a full range of Google cloud products and services including BigQuery.



At the home button, scroll down to open BigQuery.



Have feedback or corrections? Please email us at feedback@isb-cgc.org.

12.2 ISB-CGC BigQuery Projects

ISB-CGC has two open-access Google BigQuery projects. To quickly access the ISB-CGC tables from your project on the Google BigQuery Console, you'll need to link to these projects. This process, known as "pinning a project", is described here.

- **isb-cgc** - This project has been in use since ISB-CGC's inception.
- **isb-cgc-bq** - This is a new project as of July 2020. It will hold all new ISB-CGC tables, and many of the tables in the isb-cgc project will be migrated here over time.

12.2.1 isb-cgc project

The isb-cgc project contains all of the ISB-CGC BigQuery tables created before July 2020.

Tables in isb-cgc will be retired and labeled as deprecated as we copy them over to the new project. Table descriptions will include the new table location. Eventually they will be turned into only views (with no preview ability) to ensure that existing references will continue to work correctly. Many older tables with light usage may remain in isb-cgc and not be copied over; tables with no logged recent usage may be deleted. When using the BigQuery Table Search UI to find these retired tables, select Status of **Deprecated**.

Many tables will continue to have the status of **Current**, at least for the time being, until they are copied to the new project. In addition, there are tables with the status of **Archived** in the isb-cgc project and more may become archived. **Archived** indicates that the table contains an older version of data; a newer version of the same data exists in another table.

12.2.2 isb-cgc-bq project

The isb-cgc-bq project contains all new ISB-CGC BigQuery tables created after July 1, 2020 as well as tables that have been migrated from project isb-cgc. It features a more intuitive data set and table organization, as well as consistent table naming both within and across cancer research programs.

This new project is a work in progress. The migration of existing tables from the isb-cgc project will be occurring over time, and will not be all at once. See the Migration to Project isb-cgc-bq Release Notes to find out which tables have been migrated as of this time.

All new tables will be created in this project.

isb-cgc-bq Data Set and Table Organization

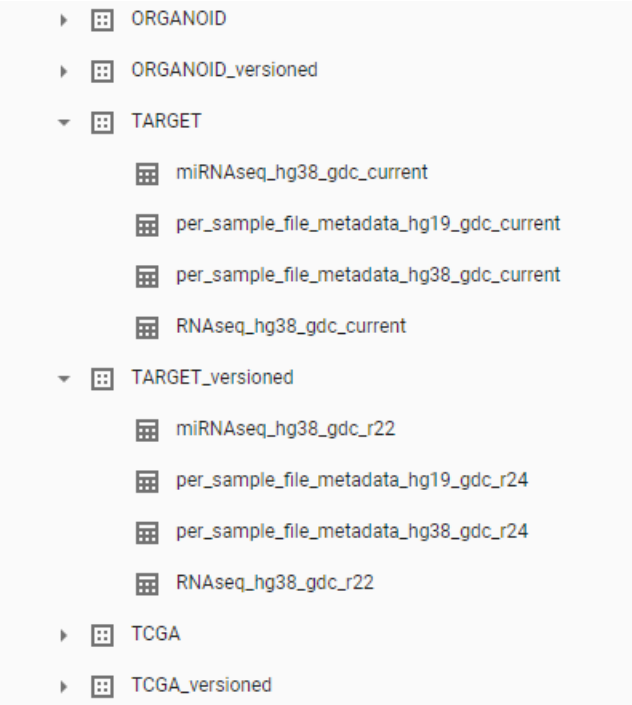
Each Program has two data sets, one containing the most current data that ISB-CGC has, and one containing versioned tables, which serves as an archive of previously released tables.

As new data releases occur, the data in the "_current" tables will be replaced with this new data. If you want the most up-to-date data, use these tables in your queries. However, if you want to ensure that your queries create a reproducible result, use a table from the "_versioned" data set. The most current data is also in this data set; however, the name of the table will end with the release number or year and not "current".

See below for more details.

Data Set Name	Data Set Contents	Table Name Format	Table Status
<Program>	Latest tables for each data type (ex. miRNA Expression, File Metadata) that ISB-CGC has, per Program	Data Type, Reference Genome, Source, Current. Ex. `TARGET.miRNAseq_hg38_gdc_current`	When using the BigQuery Table Search UI to find these tables, select Status of **Current**.
<Program>_versioned	Previously released tables, as well as the most current table	Data Type, Reference Genome, Source, Release Number or Year. Ex. `TARGET_versioned.miRNAseq_hg38_gdc_r22`. Here, the name of the most current table will end with the release number or year and not "current".	Previously released tables have status of **Archived**. The most current table has the status of **Current**.

See below for a snapshot of the isb-cgc-bq data set and table organization in the Google BigQuery Console.

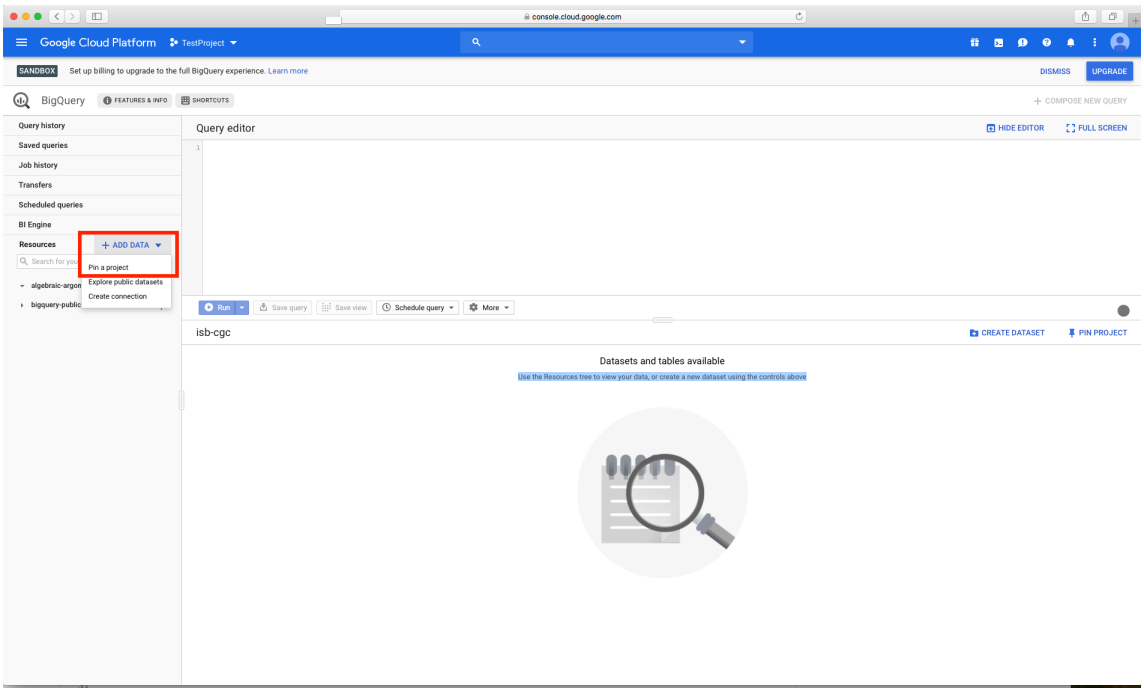


Have feedback or corrections? Please email us at feedback@isb-cgc.org.

12.3 Linking to ISB-CGC BigQuery tables

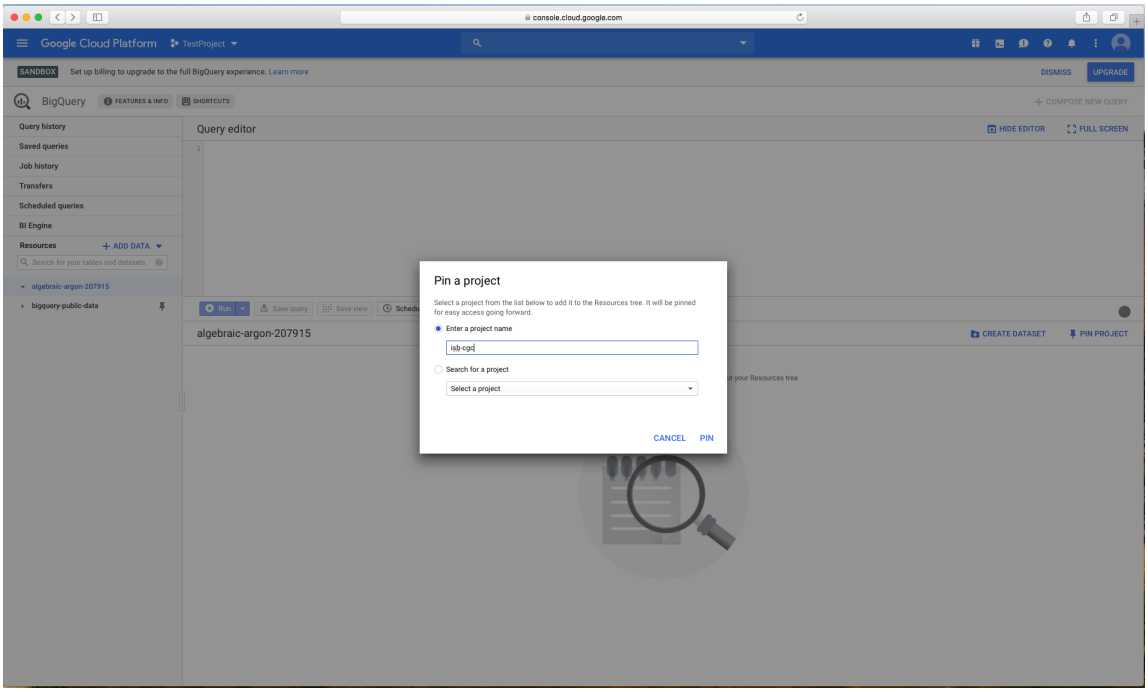
Follow the images below to link the ISB-CGC BigQuery tables in projects **isb-cgc** and **isb-cgc-bq** to your Google Cloud Project. Click on image to zoom in.

When you access BigQuery from your Google Cloud Platform Console, you will see an "Add Data" box with a "Pin a Project option".



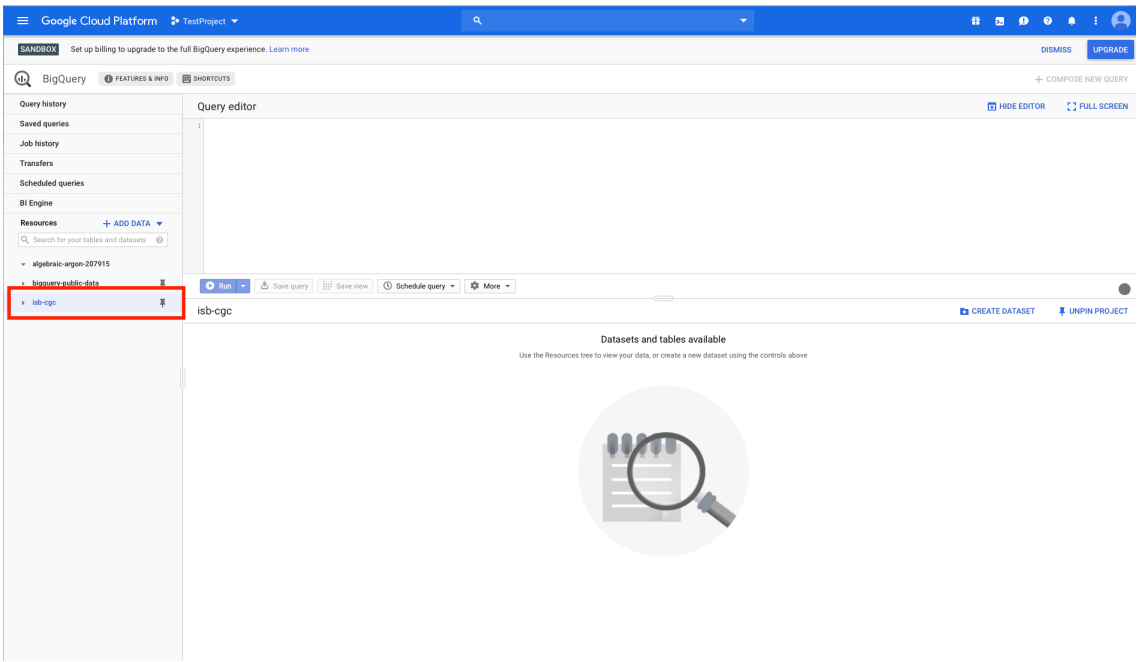
When you click on "Pin a Project", you will be presented with a pop-up box that allows you to either enter a project name or to select one from a list. Choose the "Enter a Project Name" and enter in "isb-cgc" and then click "Pin".

Note: If you are have Editor Tabs enabled, the "Pin a Project" options are a little different. When you click on "Add Data", select "Pin a Project" and then "Enter Project Name" from the menu. Then enter the project name in the pop-up box and click "Pin".



You will now see the isb-cgc open access BigQuery tables on the left-hand side pinned to your project. Repeat these steps for "isb-cgc-bq".

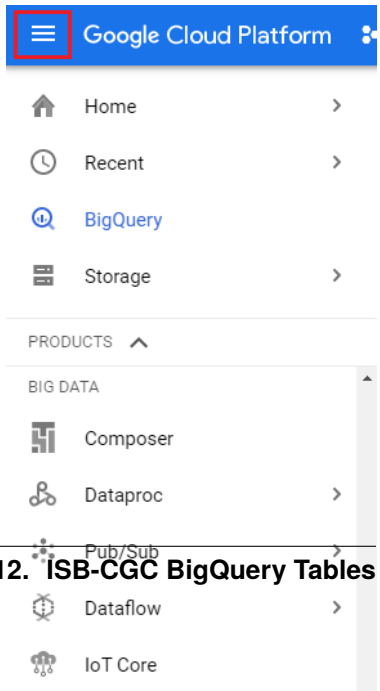
Note: If the data sets and tables within the project don't display immediately, refresh your screen until they do. They may take a couple of minutes to appear.



Have feedback or corrections? Please email us at feedback@isb-cgc.org.

12.4 BigQuery SQL Examples

You can write SQL queries to retrieve data from ISB-CGC BigQuery tables directly in the Google BigQuery console. To get to the console from within the Google Cloud Platform, click the Navigation menu in the upper left-hand corner. Expand PRODUCTS and find BigQuery in the BIG DATA section. (If you pin BigQuery, BigQuery will also display in the upper part of the navigation menu, making it easier to find next time.)



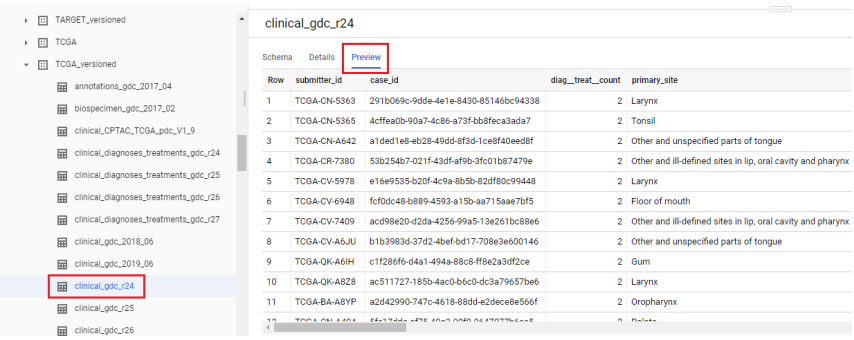
These instructions from Google will tell you more about using the BigQuery console.

12.4.1 Query versus Preview

Here is a simple query which retrieves all columns in a table.

```
SELECT *
FROM `isb-cgc-bq.TCGA_versioned.clinical_gdc_r24`
LIMIT 1000
```

You can use the "Preview" feature in the BigQuery web UI, at no cost, instead of doing a SELECT * which will do a full table scan! See the picture below.



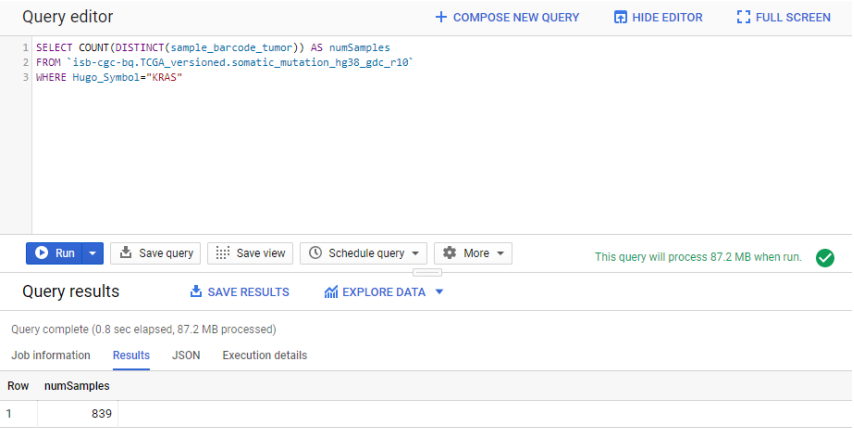
12.4.2 Simple Query Examples

Let's start with a few simple examples to get some practice using BigQuery. You can copy and paste any of the SQL queries on this page into the BigQuery web console at https://console.cloud.google.com/bigquery.

1. How many mutations have been observed in KRAS?

```
SELECT COUNT(DISTINCT(sample_barcode_tumor)) AS numSamples
FROM␣
↪`isb-cgc-bq.TCGA_versioned.somatic_mutation_hg38_gdc_r10`
WHERE Hugo_Symbol="KRAS"
```

The screenshot below shows the query in the "Query Editor" box, and the results down below. Just click on the "RUN QUERY" button to run the query. Notice the green checkmark indicating that the SQL query syntax looks good.



2. What other information is available about these KRAS mutant tumors?

In addition to answering the question above, this next query also illustrates usage of the **WITH** construct to create an intermediate table on the fly, and then use it in a follow-up **SELECT**:

```
WITH temp1 AS (
   SELECT
     project_short_name,
     sample_barcode_tumor,
     Hugo_Symbol,
     Variant_Classification,
     Variant_Type,
     SIFT,
     PolyPhen
   FROM  `isb-cgc-bq.TCGA_versioned.somatic_mutation_hg38_gdc_r10`
   WHERE Hugo_Symbol="KRAS"
   GROUP BY
     project_short_name,
     sample_barcode_tumor,
     Hugo_Symbol,
     Variant_Classification,
     Variant_Type,
     SIFT,
     PolyPhen )
SELECT
   COUNT(*) AS num,
   Hugo_Symbol,
   Variant_Classification,
   Variant_Type,
   SIFT,
   PolyPhen
FROM temp1
GROUP BY
   Hugo_Symbol,
   Variant_Classification,
   Variant_Type,
   SIFT,
   PolyPhen
ORDER BY num DESC
```

Query editor

```
WITH temp1 AS (
   SELECT
     project_short_name,
     sample_barcode_tumor,
     Hugo_Symbol,
     Variant_Classification,
     Variant_Type,
     SIFT,
     PolyPhen
   FROM  `isb-cgc-bq.TCGA_versioned.somatic_mutation_hg38_gdc_r10`
   WHERE Hugo_Symbol="KRAS"
   GROUP BY
     project_short_name,
     sample_barcode_tumor,
     Hugo_Symbol,
     Variant_Classification,
     Variant_Type,
     SIFT,
     PolyPhen )
```

Run | Save query | Save view | Schedule query | More

Query results | SAVE RESULTS | EXPLORE DATA

Query complete (1.0 sec elapsed, 255.1 MB processed)

Job information | Results | JSON | Execution details

Row	num	Hugo_Symbol	Variant_Classification	Variant_Type	SIFT	PolyPhen
1	208	KRAS	Missense_Mutation	SNP	deleterious(0)	benign(0.361)
2	176	KRAS	Missense_Mutation	SNP	deleterious(0)	possibly_damaging(0.479)
3	94	KRAS	Missense_Mutation	SNP	deleterious(0.04)	probably_damaging(0.993)
4	76	KRAS	Missense_Mutation	SNP	deleterious(0.04)	possibly_damaging(0.506)
5	41	KRAS	Missense_Mutation	SNP	deleterious(0.02)	possibly_damaging(0.773)
6	37	KRAS	Missense_Mutation	SNP	deleterious(0.03)	benign(0.376)
7	30	KRAS	3'Flank	SNP	null	null
8	24	KRAS	Missense_Mutation	SNP	deleterious(0.03)	possibly_damaging(0.644)

3. What are the most frequently observed mutations and how often do they occur?

```
WITH temp1 AS (
   SELECT
     sample_barcode_tumor,
     Hugo_Symbol,
     Variant_Classification,
     Variant_Type,
     SIFT,
     PolyPhen
   FROM `isb-cgc-bq.TCGA_versioned.somatic_mutation_hg38_gdc_r10`
   GROUP BY
     sample_barcode_tumor,
```

(continues on next page)

(continued from previous page)

```
    Hugo_Symbol,
    Variant_Classification,
    Variant_Type,
    SIFT,
    PolyPhen)
SELECT
  COUNT(*) AS num,
  Hugo_Symbol,
  Variant_Classification,
  Variant_Type,
  SIFT,
  PolyPhen
FROM temp1
GROUP BY
  Hugo_Symbol,
  Variant_Classification,
  Variant_Type,
  SIFT,
  PolyPhen
ORDER BY num DESC
```

Query editor

```
1  WITH temp1 AS (
2     SELECT
3       sample_barcode_tumor,
4       Hugo_Symbol,
5       Variant_Classification,
6       Variant_Type,
7       SIFT,
8       PolyPhen
9     FROM `isb-cgc-bq.TCGA_versioned.somatic_mutation_hg38_gdc_r10`
10    GROUP BY
11      sample_barcode_tumor,
12      Hugo_Symbol,
13      Variant_Classification,
```

Run | Save query | Save view | Schedule query | More

Query results | SAVE RESULTS | EXPLORE DATA

Query complete (13.9 sec elapsed, 217.7 MB processed)

Job information | Results | JSON | Execution details

Row	num	Hugo_Symbol	Variant_Classification	Variant_Type	SIFT	PolyPhen
1	2977	TTN	Missense_Mutation	SNP	null	null
2	1488	MUC16	Missense_Mutation	SNP	null	unknown(0)
3	1431	TTN	Silent	SNP	null	null
4	1022	TP53	Missense_Mutation	SNP	deleterious(0)	probably_damaging(1)
5	985	MUC16	Silent	SNP	null	null
6	949	PCLO	Missense_Mutation	SNP	null	unknown(0)
7	892	TTN	Intron	SNP	null	null
8	659	MUC2	RNA	SNP	null	null
9	583	MUC5B	Missense_Mutation	SNP	null	unknown(0)
10	565	BRAF	Missense_Mutation	SNP	deleterious(0)	probably_damaging(0.967)
11	539	TP53	Nonsense_Mutation	SNP	null	null
12	535	DST	Intron	SNP	null	null

12.4.3 Querying from more than one table (Joining)

Q: For bladder cancer patients who have mutations in the CDKN2A (cyclin-dependent kinase inhibitor 2A) gene, what types of mutations are they, what is their gender, vital status, and days to death - and for three downstream genes (MDM2 (MDM2 proto-oncogene), TP53 (tumor protein p53), CDKN1A (cyclin-dependent kinase inhibitor 1A)), what are the gene expression levels for each patient?

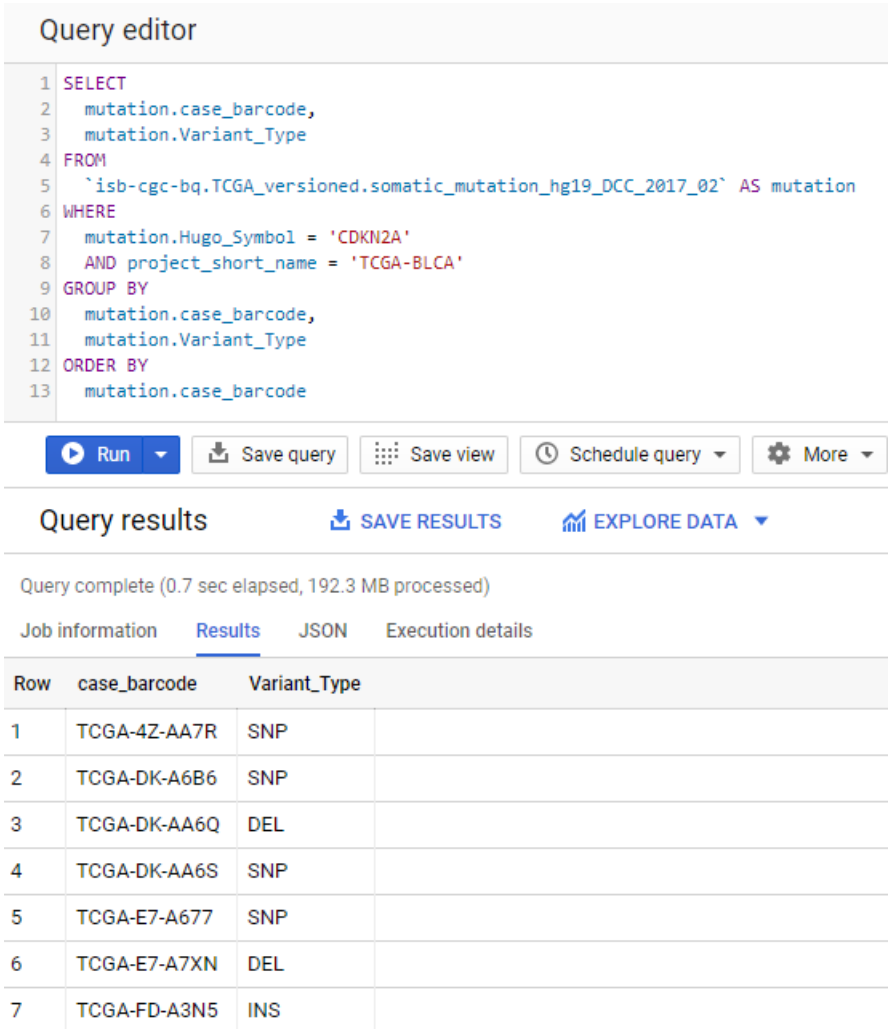
This question was chosen as an interesting example because the p53/Rb pathway is commonly involved in bladder cancer (see TCGA Network paper "Comprehensive Molecular Characterization of Urothelial Bladder Carcinoma", Figure 4).

This is a complex question that requires information from four tables. We will build up this complex query in three steps.

Step 1

Find the patients with bladder cancer who have mutations in the CDKN2A gene, and display the patient ID and the type of mutation.

```
SELECT
  mutation.case_barcode,
  mutation.Variant_Type
FROM
  `isb-cgc-bq.TCGA_versioned.somatic_mutation_hg19_DCC_2017_02` AS mutation
WHERE
  mutation.Hugo_Symbol = 'CDKN2A'
  AND project_short_name = 'TCGA-BLCA'
GROUP BY
  mutation.case_barcode,
  mutation.Variant_Type
ORDER BY
  mutation.case_barcode
```



We now have the list of patients who have a mutation in the CDKN2A gene and the type of mutation.

Notice that we have named the "isb-cgc-bq.TCGA_versioned.somatic_mutation_hg19_DCC_2017_02" table "muta-

tion" using the AS statement. This is useful for easier reading and composing of complex queries.

Step 2

Bring in the patient data from the ISB-CGC TCGA Clinical table so that we can see each patient's gender, vital status and days to death.

```
SELECT
  case_list.case_barcode AS case_barcode,
  case_list.Variant_Type AS Variant_Type,
  clinical.demo__gender,
  clinical.demo__vital_status,
  clinical.demo__days_to_death
FROM
  /* this will get the unique list of cases having the TP53 gene mutation in BRCA cases*/
  ( SELECT
    mutation.case_barcode,
    mutation.Variant_Type
  FROM
    isb-cgc-bq.TCGA_versioned.somatic_mutation_hg19_DCC_2017_02 AS mutation
  WHERE
    mutation.Hugo_Symbol = 'CDKN2A'
    AND project_short_name = 'TCGA-BLCA'
  GROUP BY
    mutation.case_barcode,
    mutation.Variant_Type
  ORDER BY
    mutation.case_barcode
    ) AS case_list /* end case_list */
JOIN
  isb-cgc-bq.TCGA.clinical_gdc_current AS clinical
ON
  case_list.case_barcode = clinical.submitter_id
```

Query editor

```
1  SELECT
2    case_list.case_barcode AS case_barcode,
3    case_list.Variant_Type AS Variant_Type,
4    clinical.demo__gender,
5    clinical.demo__vital_status,
6    clinical.demo__days_to_death
7  FROM
8    /* this will get the unique list of cases having the TP53 gene mutation in BRCA cases*/
9    ( SELECT
10     mutation.case_barcode,
11     mutation.Variant_Type
12   FROM
13     isb-cgc-bq.TCGA_versioned.somatic_mutation_hg19_DCC_2017_02 AS mutation
14   WHERE
15     mutation.Hugo_Symbol = 'CDKN2A'
16     AND project_short_name = 'TCGA-BLCA'
17   GROUP BY
```

Run | Save query | Save view | Schedule query | More

Query results | SAVE RESULTS | EXPLORE DATA

Query complete (0.6 sec elapsed, 192.7 MB processed)

Job information | Results | JSON | Execution details

Row	case_barcode	Variant_Type	demo__gender	demo__vital_status	demo__days_to_death
1	TCGA-G2-A2EO	SNP	male	Dead	1804.0
2	TCGA-E7-A7XN	DEL	male	Alive	null
3	TCGA-GD-A3OS	SNP	female	Alive	null
4	TCGA-FD-A5BV	INS	female	Dead	163.0
5	TCGA-G2-A2ES	SNP	male	Dead	1004.0
6	TCGA-DK-AA6Q	DEL	female	Dead	413.0

We now have combined information from two tables through a join (inner join by default). The same information is stored in the case_barcode field in the mutations table and in the submitter_id in the clinical table, which enables us to join on them.

Step 3

Show the gene expression levels for the four genes of interest, and order them by case id (Case Barcode) and gene name (HGNC_gene_symbol).

```
SELECT
  genex.case_barcode AS case_barcode,
  genex.sample_barcode AS sample_barcode,
  genex.aliquot_barcode AS aliquot_barcode,
  genex.HGNC_gene_symbol AS HGNC_gene_symbol,
  clinical_info.Variant_Type AS Variant_Type,
  genex.gene_id AS gene_id,
  genex.normalized_count AS normalized_count,
  genex.project_short_name AS project_short_name,
```

(continues on next page)

(continued from previous page)

```
  clinical_info.demo__gender AS gender,
  clinical_info.demo__vital_status AS vital_status,
  clinical_info.demo__days_to_death AS days_to_death
FROM ( /* This will get the clinical information for the cases*/
  SELECT
    case_list.Variant_Type AS Variant_Type,
    case_list.case_barcode AS case_barcode,
    clinical.demo__gender,
    clinical.demo__vital_status,
    clinical.demo__days_to_death
  FROM
    /* this will get the unique list of cases having the CDKN2A gene mutation in bladder␣
↪cancer BLCA cases*/
    (SELECT
      mutation.case_barcode,
      mutation.Variant_Type
    FROM
      isb-cgc-bq.TCGA_versioned.somatic_mutation_hg19_DCC_2017_02 AS mutation
    WHERE
      mutation.Hugo_Symbol = 'CDKN2A'
      AND project_short_name = 'TCGA-BLCA'
    GROUP BY
      mutation.case_barcode,
      mutation.Variant_Type
    ORDER BY
      mutation.case_barcode
      ) AS case_list /* end case_list */
  INNER JOIN
    isb-cgc-bq.TCGA.clinical_gdc_current AS clinical
  ON
    case_list.case_barcode = clinical.submitter_id /* end clinical annotation */ ) AS␣
↪clinical_info
INNER JOIN
  isb-cgc-bq.TCGA_versioned.RNAseq_hg19_gdc_2017_02 AS genex
ON
  genex.case_barcode = clinical_info.case_barcode
WHERE
  genex.HGNC_gene_symbol IN ('MDM2', 'TP53', 'CDKN1A','CCNE1')
ORDER BY
  case_barcode,
  HGNC_gene_symbol
```

Query editor

```
1  SELECT
2    genex.case_barcode AS case_barcode,
3    genex.sample_barcode AS sample_barcode,
4    genex.aliquot_barcode AS aliquot_barcode,
5    genex.HGNC_gene_symbol AS HGNC_gene_symbol,
6    clinical_info.Variant_Type AS Variant_Type,
7    genex.gene_id AS gene_id,
8    genex.normalized_count AS normalized_count,
9    genex.project_short_name AS project_short_name,
10   clinical_info.demo__gender AS gender,
11   clinical_info.demo__vital_status AS vital_status,
12   clinical_info.demo__days_to_death AS days_to_death
13 FROM ( /* This will get the clinical information for the cases*/
14   SELECT
15     case_list.Variant_Type AS Variant_Type,
16     case_list.case_barcode AS case_barcode,
17     clinical.demo__gender,
18     clinical.demo__vital_status,
19     clinical.demo__days_to_death
20   FROM
```

Run | Save query | Save view | Schedule query | More

Query results SAVE RESULTS EXPLORE DATA

Query complete (1.7 sec elapsed, 20.7 GB processed)

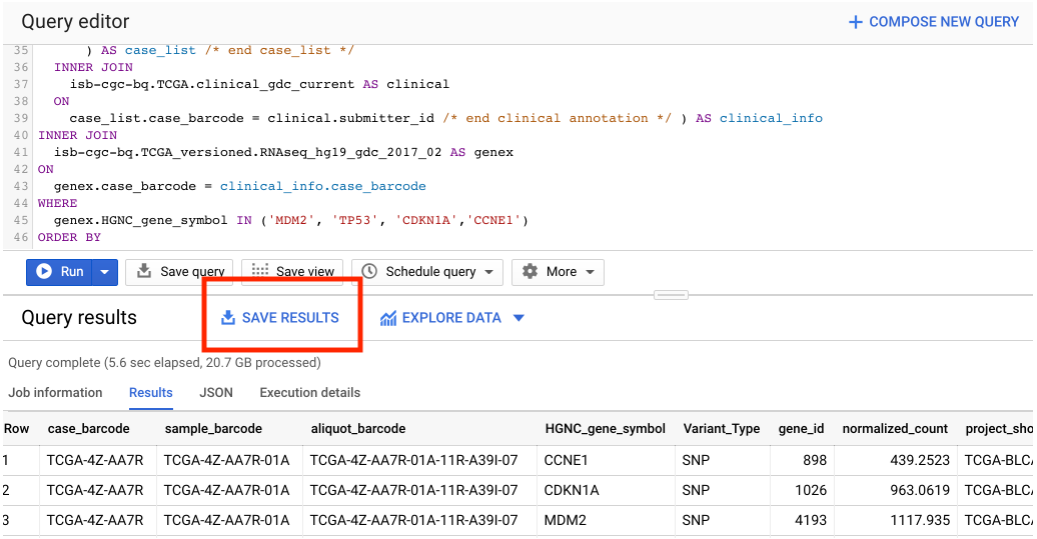
Job information Results JSON Execution details

Row	case_barcode	sample_barcode	aliquot_barcode	HGNC_gene_symbol	Variant_Type	gene_id	normalized_count
1	TCGA-4Z-AA7R	TCGA-4Z-AA7R-01A	TCGA-4Z-AA7R-01A-11R-A39I-07	CCNE1	SNP	898	439.2523
2	TCGA-4Z-AA7R	TCGA-4Z-AA7R-01A	TCGA-4Z-AA7R-01A-11R-A39I-07	CDKN1A	SNP	1026	963.0619
3	TCGA-4Z-AA7R	TCGA-4Z-AA7R-01A	TCGA-4Z-AA7R-01A-11R-A39I-07	MDM2	SNP	4193	1117.935
4	TCGA-4Z-AA7R	TCGA-4Z-AA7R-01A	TCGA-4Z-AA7R-01A-11R-A39I-07	TP53	SNP	7157	710.7254
5	TCGA-DK-A6B6	TCGA-DK-A6B6-01A	TCGA-DK-A6B6-01A-11R-A30C-07	CCNE1	SNP	898	211.9698
6	TCGA-DK-A6B6	TCGA-DK-A6B6-01A	TCGA-DK-A6B6-01A-11R-A30C-07	CDKN1A	SNP	1026	3700.8614
7	TCGA-DK-A6B6	TCGA-DK-A6B6-01A	TCGA-DK-A6B6-01A-11R-A30C-07	MDM2	SNP	4193	1660.7267
8	TCGA-DK-A6B6	TCGA-DK-A6B6-01A	TCGA-DK-A6B6-01A-11R-A30C-07	TP53	SNP	7157	284.4076

We now have all the data together in one table for further analysis. Note that the final join surrounds the previous join top and bottom. This is a common method of performing table joins.

12.4.4 Saving Query Results

You can download the results from a query in either CSV or JSON format, or save it for further analysis into a Google BigQuery table; see the options under SAVE RESULTS.



Running large queries combining multiple tables may be limited by cost and resources. If your query gets too complex it can take too long to run. Saving results as intermediate tables is a solution to these issues and can allow others to view and use them. Creating intermediate result tables can be a good approach to obtain the same result more quickly and at a lower cost.

12.4.5 SQL Functions

Standard SQL includes a large variety of built-in functions and operators including logical and statistical aggregate functions, and mathematical functions, just to name a few. User-defined functions (UDFs) are also supported and can be used to further extend the types of analyses possible in BigQuery. ISB-CGC offers a set of UDFs that implement commonly used statistical tests and methods in cancer research and bioinformatics. Please refer to this page for information on how to use the ISB-CGC UDFs.

12.4.6 Composing Queries Using the bq Command Line Tool

The **bq** command line tool is part of the cloud SDK and can be used to interact directly with BigQuery from the command line. The cloud SDK is easy to install and is available for most operating systems. It be can used to create and upload your own tables into BigQuery (if you have your own GCP project) as well as run queries at the command-line like this:

```
 bq query --use_legacy_sql=false \
'SELECT COUNT(DISTINCT(sample_barcode_tumor)) AS numSamples
  FROM `isb-cgc-bq.TCGA_versioned.somatic_mutation_hg38_gdc_r10`
  WHERE Hugo_Symbol="KRAS"'
```

12.4.7 Using BigQuery from R

There are a number of resources online as well as through ISB-CGC that demonstrate how to access BigQuery from R:

- BigQuery can be accessed from R using one of two powerful R packages; please refer to the documentation provided with these packages for more information:
  - bigrquery
  - dplyr
- If you have a GCP, you can use R with BigQuery through the Google AI plaform. Please refer to the Google documentation for more detail.
- Explore our Community Notebook Repository for examples on how to access BigQuery from R.

12.4.8 Using BigQuery from Python

- BigQuery client libraries are available that let you interact with BigQuery from Python or other languages.
- In addition, the pandas.io.gbq module provides a wrapper for BigQuery.
- Explore our Community Notebook Repository for examples on how to access BigQuery using Python.

12.4.9 Getting Help

Aside from the documentation, the best place to look for help using BigQuery and tips and tricks with SQL is StackOverflow. If you tag your question with `google-bigquery` your question will quickly get the attention of Google BigQuery experts.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

12.5 User Defined Functions

BigQuery now supports User Defined Functions (UDFs) in SQL and JavaScript that extend BigQuery for more specialized computations and that can be reused in notebooks and queries. To facilitate the analysis of cancer data, ISB-CGC offers a set of UDFs that implement commonly used statistical tests and methods in cancer research and bioinformatics. The UDFs are located in the `isb-cgc-bq.functions` data set, and the source code of the functions and examples of how to use them can be found in our Community Notebook GitHub Repository. The following table lists all the functions available in ISB-CGC.

UDF (click for details)	Description	Notebooks
kmeans	K-means method for clustering data	Python
p_fisherexact	p value of the Fisher exact test	
mannwhitneyu	Mann–Whitney U test	
kruskal_walis	Kruskal Walis test	Python
significance_level_ttest2	Significance level of the two sided T test	Python
complement_chisquare_cdf	One minus the CDF of the Chi Square distribution	
jstat_normal_cdf	CDF of the Normal distribution	

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

12.6 Usage Costs

There are two basic types of usages costs associated with BigQuery: **query costs** and **storage costs**.

12.6.1 Query Costs

The main costs associated with using BigQuery are the query costs. In BigQuery, queries are billed according to how much data is scanned during the course of the query, and the rate is $5 per TB, although the first **1 TB is free each month**. Queries can be more expensive as they become more computationally intensive.

While most of the cost is suprisingly low, it is always important to think carefully about your queries and to make them as efficient as possible. For example, if you want to derive summary information about all ~20,000 genes, you could do that with a single query that might cost a few pennies, or you might write a less-clever query that returns information only about a single gene and then programmatically loop over all genes, running that single-gene query 20,000 times. Your overall query costs using this less-clever approach, instead of being a few pennies would be several hundred dollars! This latter approach would also take significantly more time.

12.6.2 Storage Costs

You may want to upload your own data to BigQuery or to store results of your queries as new BigQuery tables. In BigQuery, storage costs are based on the amount of data stored. For example, ISB-CGC is hosting PanCancer Atlas tables in BigQuery and is paying for the storage costs (with support from NCI). The size of each PanCancer Atlas table is less than 1.5 GB and therefore costs less than $0.25 per year to store.

12.6.3 Additional Support

For more information, see the following Google support pages:

- Detailed information on Google pricing
- Estimating storage and query costs
- Best practices on how to control costs
- Keep an eye on your GCP expenses on your Google Cloud Platform Console home page

---

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CHAPTER

THIRTEEN

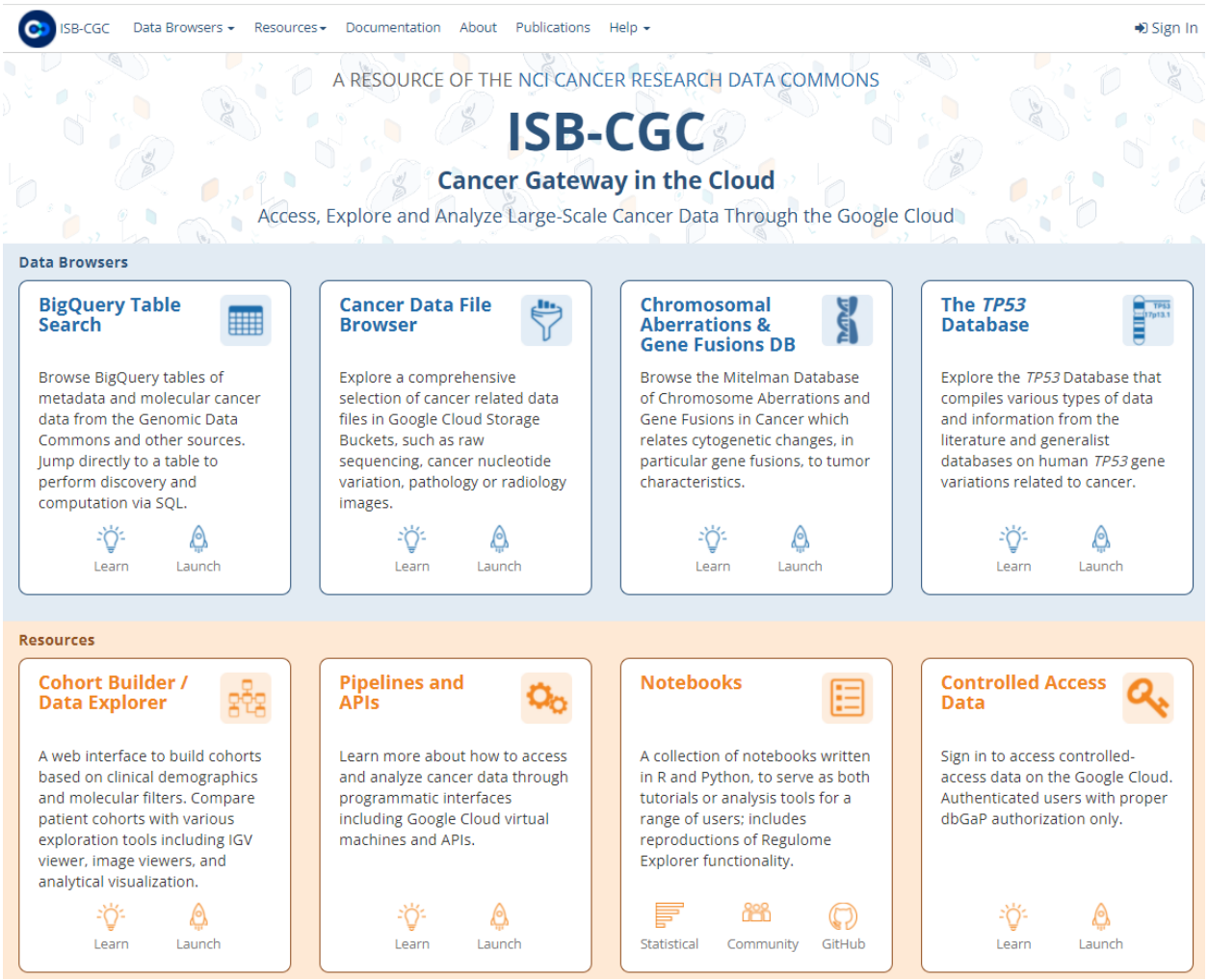
ISB-CGC WEB INTERFACE (WEB APP)

The ISB-CGC Web Interface (Web App) provides robust functionality for the user to analyze the ISB-CGC cancer data through a user interface. Without needing to use any programming, you can select and filter data from one or more public data sets (such as TCGA, CCLE, TARGET and BEATAML1.0), combine with your own uploaded data and analyze using a variety of built-in plot types. There is also a built-in Integrative Genomics Viewer (IGV) and Radiology Viewer.

Over time we will be updating and enhancing this web interface based on your feedback. We welcome your ideas and needs. Please use this link to provide them.

13.1 Login to Web App

The ISB-CGC Web App is accessed through a Google Account identity (freely available with a new account or by linking to an existing email account). If you are not logged into the ISB-CGC Web App, you will be presented with this page:



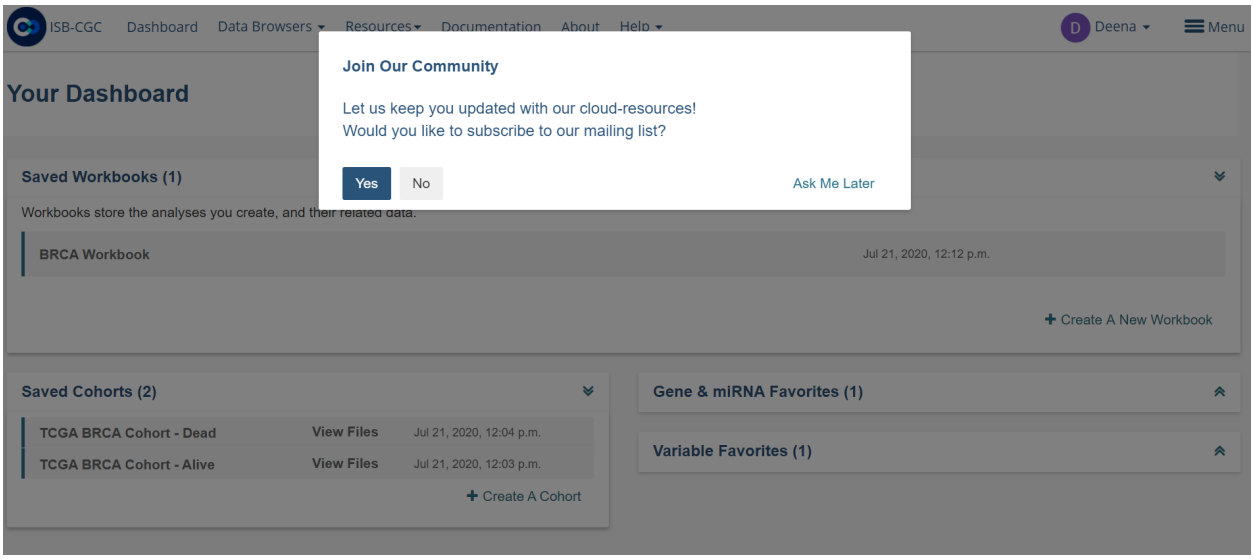
You login through the "Sign In" link in the upper right.

Also on this page are links to:

- ISB-CGC BigQuery Table Search
- Cancer Data File Browser
- Chromosomal Aberations & Gene Fusions (Mitelman) database
- The *TP53* Database
- Cohort Builder/Data Explorer
- Pipelines and APIs
- Notebooks
- Controlled Access Data
- Documentation
- ISB-CGC Publications and Citations

If your screen looks like this:

You have successfully logged into ISB-CGC Web App! Please subscribe for updates provided by ISB-CGC.

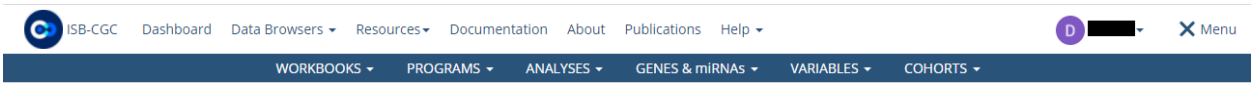


Have feedback or corrections? Please email us at feedback@isb-cgc.org.

13.2 Menu Bar

Clicking on the word Menu in the upper-right corner of your browser window (next to your name) will display the blue menu bar. You can make this menu bar disappear by clicking on the blue **X** or the word Menu again.

For additional details about each menu item, see the corresponding section in this user guide.



The **MENU** bar supplies links for the following Web App features:

- **DASHBOARD** - This menu item takes you to **Your Dashboard**, the Web App home page.
- **WORKBOOKS** - Workbooks store the Analyses you create, and their related data. You can create worksheets as you explore and analyze selected underlying data (i.e. Genes and miRNAs, Variables and Cohorts). From the Workbook menu item, you can select the following:
  - Saved Workbooks - Displays all your saved workbooks and allows you to edit, duplicate or delete the workbooks.
  - Create a New Workbook - Allows you to create a new workbook by selecting the data source and analysis type.
- **PROGRAMS** - This menu item provides a shortcut to view public programs which available in the ISB-CGC Web App.
  - Public Programs - Here you can:
    - View the public programs that are currently in the ISB-CGC Web App.
- **ANALYSES** - From here you can Create, Edit Details, Duplicate, Delete, or Share Analyses. You can use a specific analysis type to create a new workbook customized with the specific data (Genes and miRNAs, Variables, Cohorts) you have selected. The plot types that you can select are:
  - Bar Chart

  - Histogram
  - Scatter Plot
  - Violin Plot
  - Cubby Hole Plot
  - SeqPeek
  - OncoPrint
  - OncoGrid

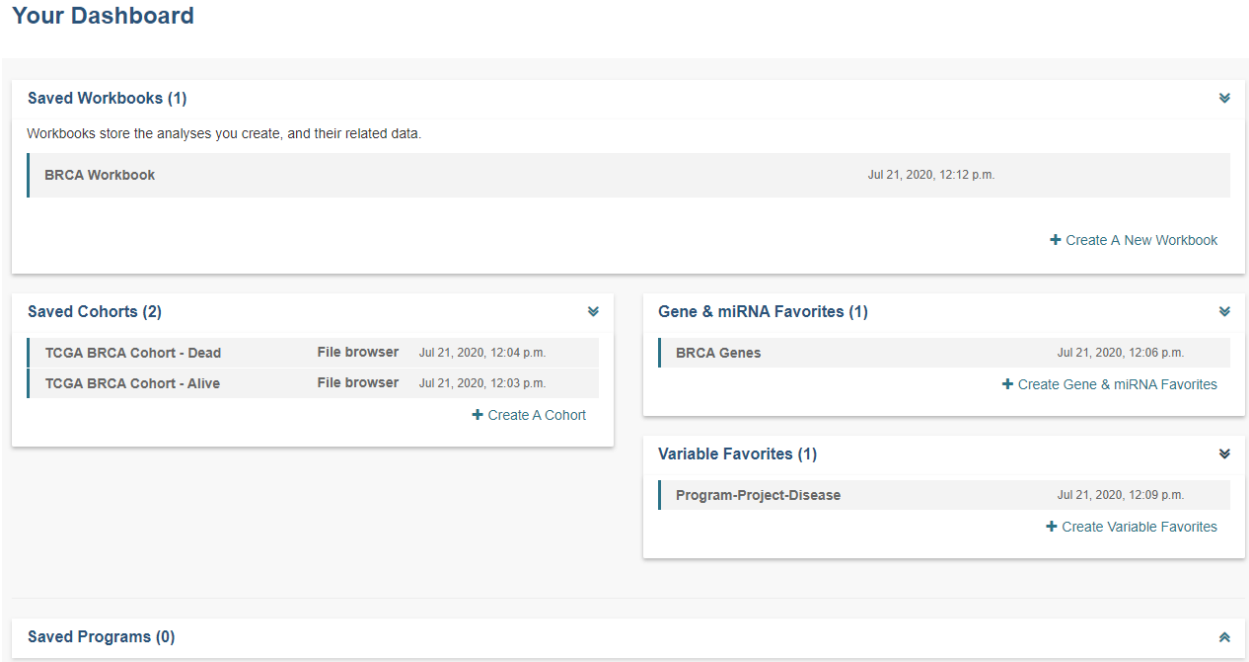
- **GENES & miRNA** - From this menu selection you can **Manage Gene and miRNA Favorites**, **Create Gene and miRNA Favorite(s)** or **Select Genes and miRNAs for a New Workbook**.
  - Manage Gene & miRNA Favorites - Here you can:
    * Edit or Delete a Saved Gene and miRNA Favorite(s)
    * Start a New Workbook
    * Create a New Gene and miRNA Favorites
  - Create Gene & miRNA Favorite - Here you can:
    * Create a Gene & miRNA Favorite for Analysis. To Create a New Gene and miRNA Favorite - You provide a name and select the Gene and/or miRNA. You can upload a stored Gene and miRNA List or type in Gene name or miRNA (**Note:** This will auto fill as you type in Gene name or miRNA name). To aid in Gene selection, you can access the HGNC portal (Hugo Gene Nomenclature Committee) via the "**View Gene Identifiers**" link under this Menu selection. Also, to aid in miRNA selection, you can access the miRBASE via the "**View miRNA Identifier**" link next to the View Gene Identifiers link.
  - Select Genes & miRNA for a New Workbook - This sub-menu has two features:
    * Apply to New Analysis - Select a Favorite(s) Gene and miRNA from the list shown of stored Favorites to Analyze
    * Add (+) Apply to New Analysis - Basically navigates back to the **Create Gene and miRNA Favorite** (See description above)
- **VARIABLES** - This menu item allows you to **Manage Variables Favorite** or **Create New Favorite**.
  - Manage Variable Favorites Lists - Shows your saved Variables as Favorites:
    * Edit
    * Delete
    * Start New Workbook - (Create a New Workbook using the selected Favorite Variables)
  - Create Favorite Variable(s) List - Here you "Name" your new favorite and select variables from four available data sources to incorporate in your analysis:
    * Common Variables
    * Favorite(s) Saved
    * Programs (Previously Uploaded and Saved)
  - Select Variables for a New Workbook - This sub-menu has two features:
    * Apply to New Worksheet - Select a Favorite(s) variables from the list shown of stored Favorites to Analyze
    * Add (+) Apply New Variable List - Basically navigates back to the **Create Variables Favorite** (See description above)

- **COHORTS** - Here you can **Manage Saved Cohorts**, select **Public Cohorts** and **Select Cohorts for a New Workbook** or **Create your First Cohort** if it's empty.
  - Manage Saved Cohorts - There are two tabs:
    - Saved Cohorts - Displays previously created cohorts which can be selected. If no cohorts exist, you can create your first Cohort here by selecting the "Create Your First Cohort" link displayed and selecting Donors and Data Types. Within Saved Cohorts you can:
      - Create a "New Workbook" from a saved Cohort
      - Delete a Saved Cohort
      - Set Operations (i.e., Union, Intersection or complement) from a Base or Subtracted Cohort.
    - Public Cohorts - Displays any public cohorts which can be selected.
      - Create a "New Workbook" from a saved Public Cohort
      - Set Operations (i.e., Union, Intersection or complement) from a Base or Subtracted Cohort.
  - Create a New Cohort - Allows you to create new cohorts using filters (such as Gender, disease code, sample type) and barcodes.
  - Public Cohorts - This menu item takes you to the same Public Cohorts page described above.
  - Select Cohorts for a New Workbook - This menu item takes you to the same Saved Cohorts and Public Cohorts pages described above.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

13.3 Dashboard

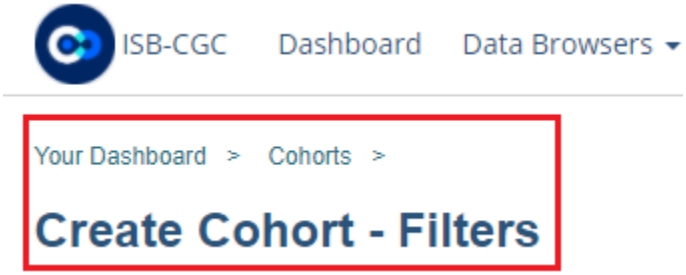
Upon signing in with a Google account identity, you will be presented with the following page:



This is your personal "Dashboard" where your Analyses, Gene and miRNA Lists, Variable Lists, Cohorts, and Saved Programs are readily accessible. Descriptions of how to use each component of this user interface are provided in the individual subsections of this user guide.

Multiple Sample Analyses can be grouped into Workbooks (and saved for later use, editing, and sharing). Workbooks are used to group together multiple related analyses, and can be used for sharing groups of analysis results with specific groups of people. For example, you may use one Workbook for an on-going study of gene mutations and pathways involved in Head and Neck Cancer (with one research group you are part of), and use a different Workbook for another on-going study with a different set of collaborators in which you are investigating survival-time after diagnosis for patients with different types of lung cancers. Think of workbooks as containers in which you can create and group related analyses, and which you can share with specific colleagues.

Breadcrumbs show you where you are in the Web App as you move from one section to another (figure below). These are live links, and can be used to rapidly navigate from one section of the interface to another.



Have feedback or corrections? Please email us at feedback@isb-cgc.org.

13.4 Workbooks

Workbooks store the analyses you create, and their related data. You can create multiple analyses in a workbook and store them on separate worksheets within the workbook. The worksheets you create to conduct analysis are based on the source data selected (i.e. Genes and miRNAs, Variables and Cohorts). Workbooks can be used to:

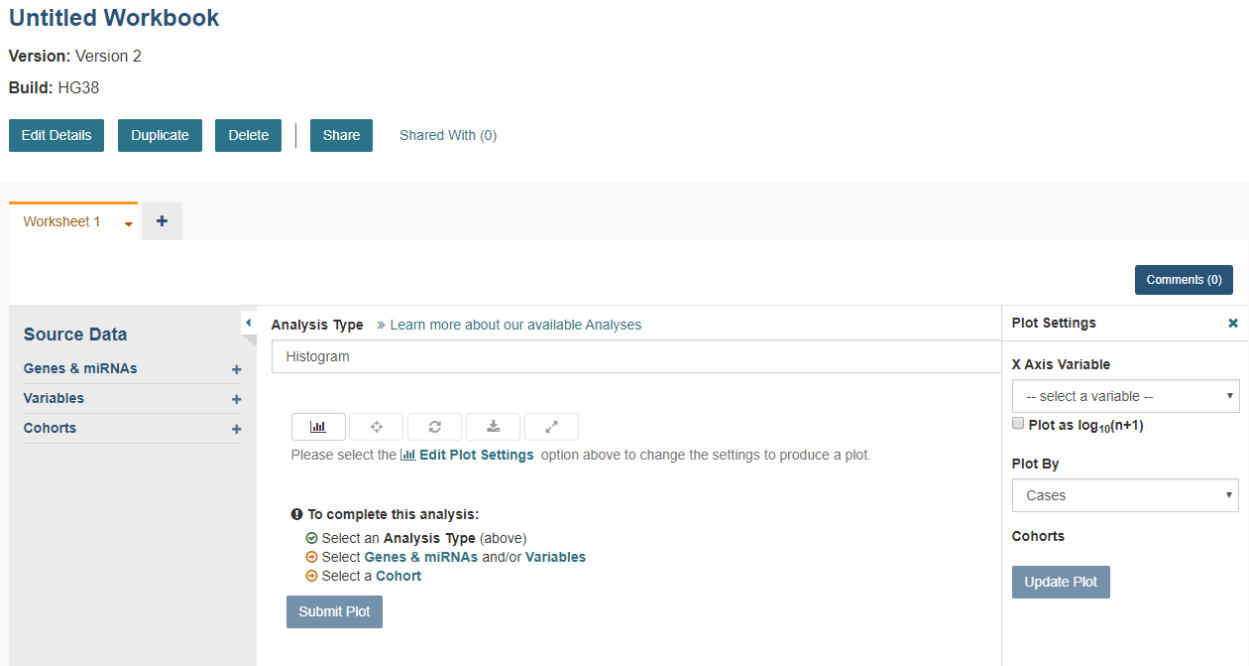
- Group together multiple related analyses.
- Share analysis results with specific groups of people.

For example, you can create a Workbook (i.e., Disease A) which consists of identifying gene mutations and pathways involved in Head and Neck Cancer (and share it with research Group A). And then you could create another Workbook (i.e., Disease B) with a different group of researchers (Group B) investigating the average time after diagnosis of death for different lung cancers. Think of workbooks as virtual "excel spreadsheets". Various related analyses can be created in individual worksheets ("Tabs" within the spreadsheet) and grouped together in one workbook (the overall spreadsheet).

13.4.1 Create a New Workbook

On **Your Dashboard**, there is a **Saved Workbooks** panel. This panel displays any previously created, saved workbooks. If you do not have any saved workbooks you will see "Workbooks store the analyses you create, and their related data." text in the panel. To create a new workbook, click on the **Create a New Workbook** link.

Selecting **Create a New Workbook** from the **WORKBOOKS** menu dropdown also displays a screen where you can create a new workbook.



Follow these steps to create a workbook:

1. From the Workbook creation panel, select an Analysis Type (i.e., Bar Chart, Histogram, Scatter Plot, Violin Plot, Cubby Hole Plot, SeqPeek, OncoPrint or OncoGrid).

   Analysis Type Description

   - **Bar Chart** - This chart is used to plot a single categorical feature for one or more cohorts. It generates vertical lines to represent the type of data being used. The X axis shows categorical information being used while the other axis (Y axis) displays categorical data chosen in the edit analysis settings.
   - **Histogram** - This chart is used to plot a single numerical feature for one or more cohorts. It generates vertical lines to represent the type of data being used. The X axis shows numerical information being used while the other axis (Y axis) displays numerical data chosen in the edit analysis settings.
   - **Scatter Plot** - This chart is used to plot two numerical features (X & Y axis) for one or more cohorts. Can also color code points by a single categorical feature.
   - **Violin Plot** - This chart is used to plot a categorical feature on the X axis versus a numerical feature on the Y axis. Points in the plot can be colored by another categorical feature.
   - **Cubby Hole Plot** - This chart is used to plot two categorical features. Boxes are colored by their related p-values.
   - **SeqPeek** - This visualization shows where somatic mutations have been observed on a linear representation of a specific protein. Each horizontal strip represents the protein, with data from different tumor types (aka cohorts or studies) shown stacked one on top of the other.

- **OncoPrint** - This chart is used to plot multiple genomic alterations (somatic mutation) events across a set of samples using color-coded glyphs. OncoPrint is developed and provided by cBioPortal.
- **OncoGrid** - This chart is used to visualize the top mutated genes across programs/projects and the number of cases affected. You can also view the mutation frequency, clinical data, data format types, number of gene sets and the number of cases affected.

Notes:

- A user has the option to make the axis logarithmic if the plot can display continuous numerical data, e.g. mRNA expression levels.
- For Violin Plot and Scatter Plot you can select multiple cohorts as your Color By Feature. This will cause the Legend to list all the cohorts that the sample is associated to. Please be aware you'll end up with lots of permutations if you have lots of samples that belong to many different cohorts.
- For OncoPrint, OncoGrid, and SeqPeek analyses, a default gene list is provided. Genes with consensus score of 6 or higher are added to the default gene list. (Ref: Bailey et al., Cell. 2018 Apr 5;173(2):371-385.e18. doi: 10.1016/j.cell.2018.02.06)

2. You will then select **Genes and miRNAs or Variables** (or, optionally both).

   Genes and miRNAs

   Selecting this link (or the '+' adjacent to it) displays the **Data Source | Gene & miRNA Favorites** screen showing previously created "Gene and miRNA Favorites". Click the **Apply to Worksheet** to apply or click **Apply New Gene & miRNA List** to create a new list and apply. Any Gene and miRNA List you create here will automatically be added to your Gene and miRNA Favorites list and can be selected for additional analysis later. (See Gene and miRNA Favorites for details.)

   Variables

   Selecting this link (or the '+' adjacent to it) displays the **Data Source | Variable Favorites** screen showing previously created "Variables Favorites". Click the **Apply to Worksheet** to apply or click **Apply New Variable List** to create a new list and apply. Any Variable Favorites you create here will automatically be added to your Variable Favorites and can be selected for additional analysis later. (See Variable Favorites for details.)

3. Select your **Cohort** - Cohorts allow the user to create custom groupings of the samples and/or cases that can be used for further analysis.

   Selecting this link (or the '+' adjacent to it) displays the **Data Source | Cohorts** screen showing previously created "Cohorts". Click the **Apply to Worksheet** to apply or click **Filter** or **Barcodes** button to create a new cohort and apply. Any Cohorts you create here will automatically be added to your Cohorts list and can be selected for additional analysis later.

   The user can also add multiple Cohorts to the worksheet if desired. More information about Cohorts can be found here.

4. Select **Edit Plot Settings** - This will display the **Plot Settings** panel displaying the applicable X & Y axis settings (i.e. Categorical or Numerical based on the analysis type selected). Depending on the analysis type selected (e.g., Bar chart, Histogram, Scatter Plot, Violin Plot, Cubby Hole Plot, SeqPeek, OncoPrint or OncoGrid) additional specifications may appear for selection.

5. Select **Toggle Sample Selection** - After a plot has been displayed, using the Toggle Sample Selection button allows you to create a smaller cohort from within the plot itself.

6. Select **Redraw** - After a plot has been displayed, using the Redraw button will reset the analysis to its original setting after being zoomed-in or moved.

7. Select **Download** - After a plot has been displayed, using the Download button will allow you to download the analysis as a SVG, PNG, or a JSON file.

8. Select **Toggle Full Screen** - After a plot has been displayed, using this button will display the plot to the full screen.

Note: If you wish to use your own data in graphing, please review the documentations on how to upload your own data and on how to graph your own data. Using your own data uses a slightly different approach than is described here.

13.4.2 Saved Workbooks

Selecting **Saved Workbooks** from the **WORKBOOKS** menu dropdown displays a screen which lists all of your saved workbooks, and information about the workbooks, including Version and Build, Name, number of Worksheets, Ownership and Last Updated.

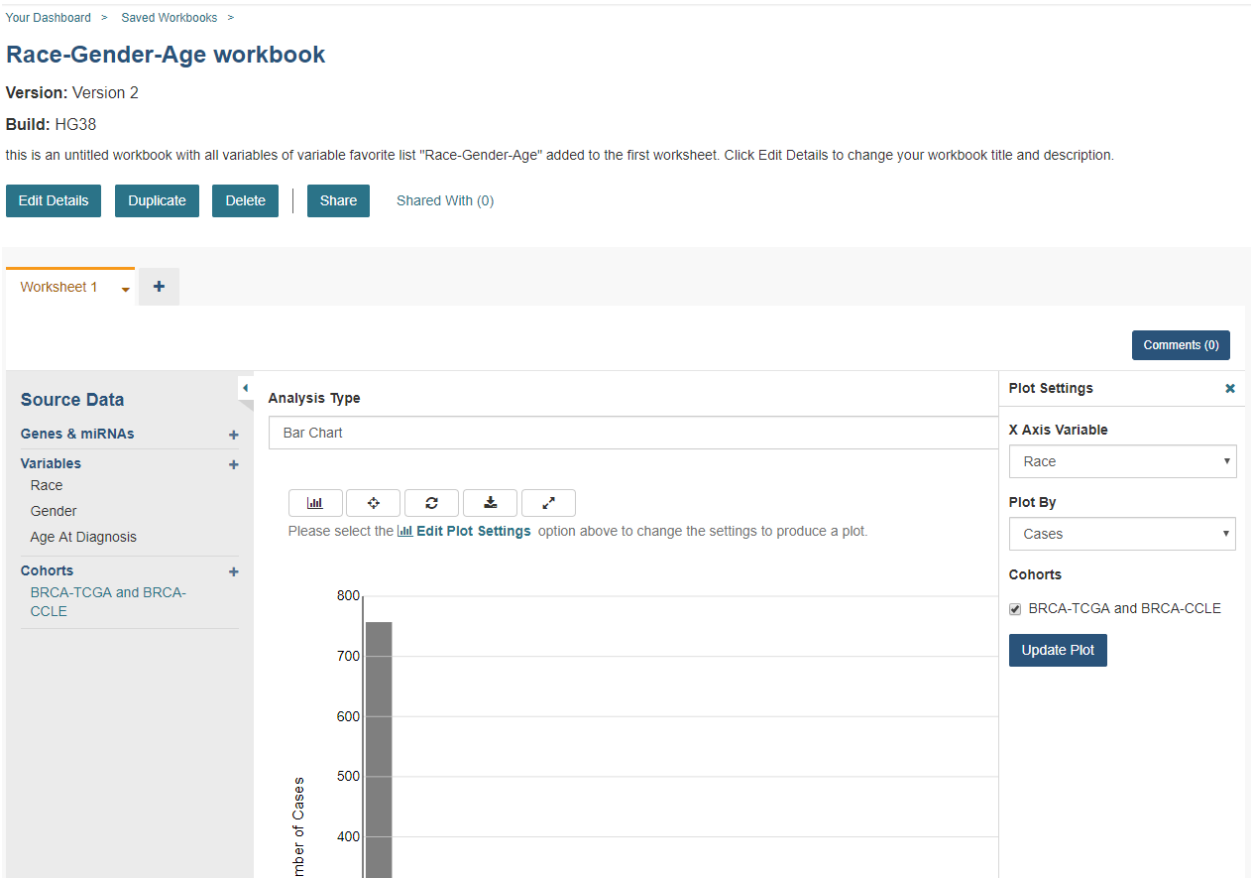
To the left of each Workbook, dropdown options allow you to Edit, Duplicate or Delete the Workbook.

- Edit - Selecting **Edit** displays a popup screen which allows you to update the Workbook name, build and description.
- Duplicate - Selecting **Duplicate** enables you to make a copy of the worksheet. Note that this will create a copy of the worksheet and reference the cohorts, variables, and gene lists used in the workbook, but will not make duplicates of the cohort, variables, and gene lists used in the workbook.
- Delete - This option will delete the workbook.

Clicking on the workbook **Name** will display the Workbook Details screen.

13.4.3 Workbook Details Screen

On the top of the Workbook Details Screen are the **Edit Details**, **Duplicate** and **Delete** buttons. They perform the same functions as described for the workbook dropdown menu options on the **Saved Workbooks** screen, described above.



Share a Workbook

Clicking the **Share** button allows you to share the workbook in the Web App with users you select by entering the user's email.

The User will receive an email message with a link to your shared workbook explaining that you want to share a workbook with them and that you have invited them to join. If the email address you entered is not registered with ISB-CGC, a message displays, "The following user emails could not be found; please ask them to log into the site first:(email entered)."

Manipulation of Workbooks and Worksheets

Creating A Worksheet - By selecting the "+" next to an existing worksheet, a user can create a new worksheet to create a new analysis. You can give the new worksheet a unique name and provide a worksheet description. This is an ideal way for the user to easily have access to different graphs with the same data in the same workbook.

Worksheet Drop Down Menu - The worksheet will have a drop down menu that allows the user to edit, duplicate or delete the worksheet. Click the downward pointing arrow next to the name of the worksheet that is open.

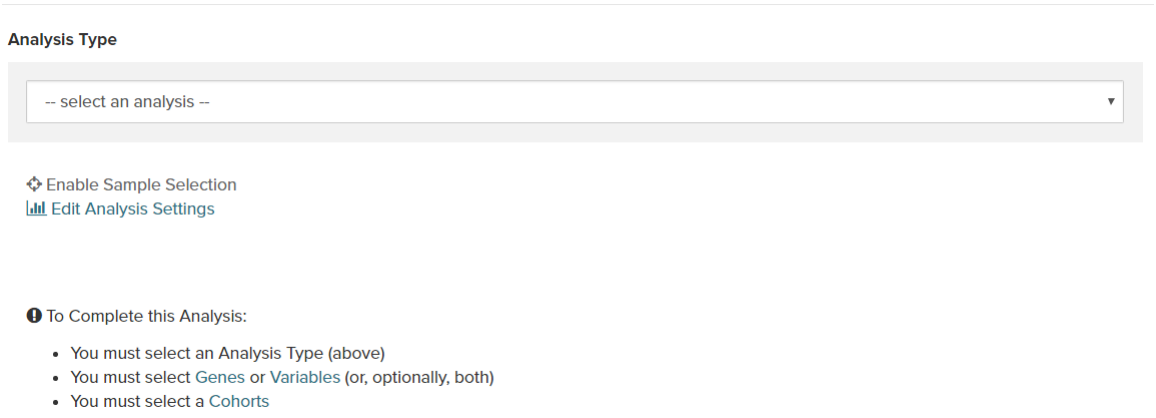
> **Edit Details** - This item allows the user to edit the name of the worksheet and also give a brief description on the worksheet being used for analysis. You can also change the build from HG19 to HG38 using this feature. Changing the build allows you to graph data from either builds.
>
> **Duplicate** - This item allows the user to create a duplicate worksheet in the workbook for further analysis and comparison.

Delete - This item will only appear when you are working with multiple worksheets. This will permanently delete the worksheet from the workbook.

Edit Plot Settings - This function allows you to select new Plot Settings for the selected analysis type.

Note: When selecting a gene or miRNA for either the X-axis or Y-axis variable, you will be prompted to select a specification. If you select Gene Expression you have the option of choosing a Select Feature. If you select the Copy Number specification you can choose a Value Filter. If you select the Protein specification you can select a Protein Filter. If you select the Mutation specification you can select a Value Filter. If you select a miRNA expression you can select a Select Feature.

Enable Sample section and Edit Analysis Settings - Enable Sample Selection (shown in the image below) allows you to select samples from displayed analysis and save that selection to a new cohort for further drill down analysis. The Edit Analysis Settings allows you to change the variables you wish to use for your analysis (varies by which analysis you choose). Finally, if you select miRNA you can select specification miRNA Expression and you will be prompted to select a feature.



Comment on a Workbook

Any user who owns or has had a workbook shared with them can comment on it. To open comments, use the **Comments** button at the top right. A right sidebar will appear and any previously comments will be shown.

On the bottom of the comments sidebar, you can create a new comment and save it. It should appear at the bottom of the list of comments.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

13.5 Programs

The Programs screen displays information about public programs available through the Web App.

13.5.1 Public Programs

Selecting **Public Program** from the **PROGRAMS** menu dropdown displays the **Programs** screen, **PUBLIC PROGRAMS** tab. This screen displays details about the public programs currently available in the Web App. It displays the number of projects, the ownership and the last date each program was updated.

Clicking the + adjacent to each program will display a list of all projects in the program, and their last updated dates.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

13.6 Analyses

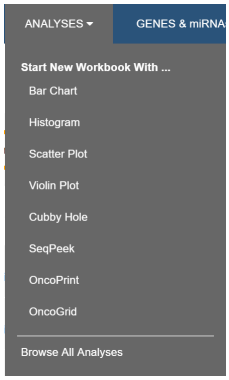
This feature allows you to create, edit details, duplicate, delete, or share analyses. You can customize new workbooks with selected data (Genes and miRNAs, Variables, Cohorts) and the following plot types:

- Bar Chart
- Histogram
- Scatter Plot
- Violin Plot
- Cubby Hole Plot
- SeqPeek
- OncoPrint
- OncoGrid

13.6.1 Start New Workbook With…<Plot Type>

Selecting **Start New Workbook With** and one of the above plot types from the **Analyses** menu dropdown displays a screen which enables you to create, edit details, duplicate, delete, or share analyses.

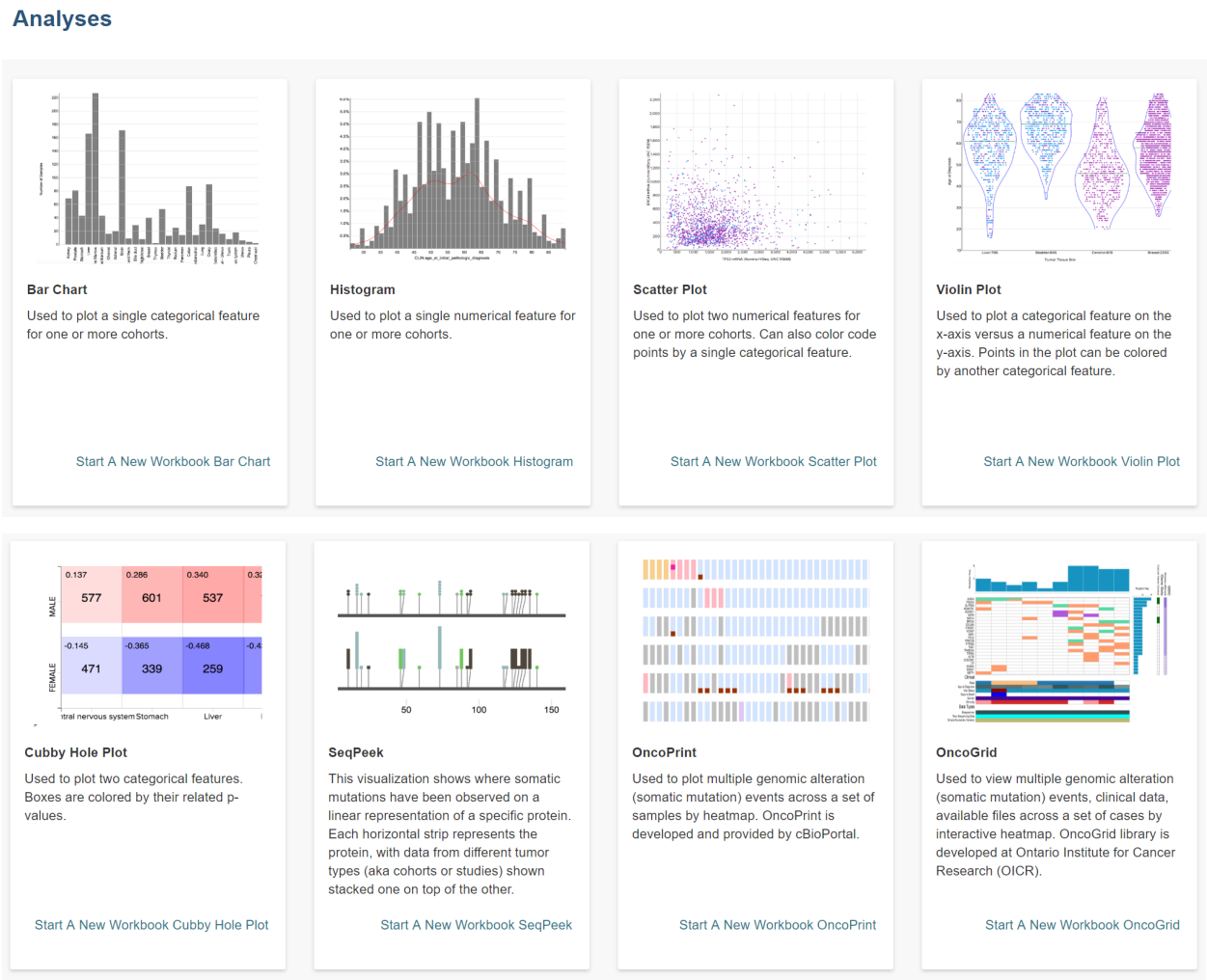
This is the same screen that is displayed when you choose to create a workbook using the **Create a New Workbook** link from **Your Dashboard** or the **WORKBOOK** menu, except that the **Analysis Type** field is prepopulated with your selected plot type.



13.6.2 Browse All Analyses

Selecting **Browse All Analyses** from the **Analyses** menu dropdown displays a screen which provides a visual example and a written description of each type of plot. This information can help you determine which type of plot would be useful in your analysis.

From here, click on the **Start a New Workbook <Plot Type>** link to go to a screen which enables you to build your analysis.



Have feedback or corrections? Please email us at feedback@isb-cgc.org.

13.7 Genes and miRNAs

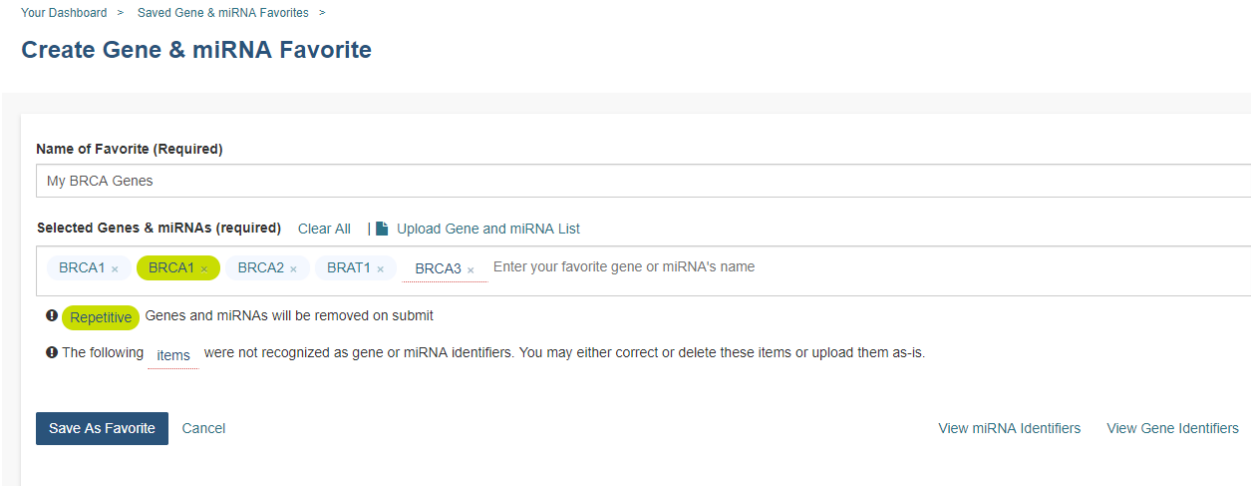
This feature allows you to create and manage Gene and miRNA lists for inclusion in workbooks and use in subsequent analyses.

13.7.1 Create a Gene & miRNA Favorite

Selecting **Create Gene & miRNA Favorites** from the **GENES & miRNAs** menu dropdown displays the **Create Gene & miRNA Favorites** screen.

To create a new Gene & miRNA Favorite:

- Name your new favorite; you can create many favorites and use them later when working with workbooks.
- Specify the Gene(s) and/or miRNA(s) to include in this list. You can do this by:
  - Uploading a pre-existing list using the Upload Gene and miRNA List link
  - Entering Genes and miRNAs by typing them into the input box (with auto-completion support).
    - To aid in Gene selection, you can access the HGNC portal (Hugo Gene Nomenclature Committee) via the View Gene Identifiers link.
    - To aid in miRNA selection, you can access the miRBase via the View miRNA Identifiers link.
    - If duplicate symbols are entered they will be marked for your deletion or automatically dropped when the list is saved. If an unrecognized item is entered it will also be flagged for your attention.
- Click **Save As Favorite**.



13.7.2 Manage Gene & miRNA Favorites

Selecting **Manage Gene & miRNA Favorites** from the **GENES & miRNAs** menu dropdown displays the **Saved Gene & miRNA Favorites** screen. This screen displays your saved Gene & miRNA Favorites and allows you to edit or delete them, as well as start a new workbook using your favorite.

Clicking on the **Create New Favorite** button will take you to the **Create Gene & miRNA Favorite** screen.

13.7.3 Select Genes & miRNAs for a New Workbook

Selecting **Select Genes & miRNAs for a New Workbook** from the **GENES & miRNAs** menu dropdown displays the **Data Source | Gene & miRNA Favorites** screen. This screen displays your saved Gene & miRNA Favorites and allows you to apply them to a new workbook.

- Check the box adjacent to your favorite and click the **Apply to New Worksheet** button to create a new workbook using your Gene & miRNA Favorite.
- Click the **Apply New Gene & miRNA List** button to create a new favorite. This takes you to the **Create Gene & miRNA Favorite** screen.

Resources for understanding and working with miRNAs and gene identifiers:

- The National Human Genome Research Institute (NHGRI) created the Talking Glossary of Genetic Terms to help everyone understand the terms and concepts used in genetic research. In addition to definitions, specialists in the field of genetics share their descriptions of terms, and many terms include images, animation and links to related terms.
- The miRBase created a microRNA database center to enable researchers to understand the published miRNA sequences and annotations.
- Hugo Gene Nomenclature Committee (HGCN)
- National Center for Biotechnology Information (NCBI)

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

13.8 Variables

Creating a variable favorites list is a way of creating custom groupings of the samples and/or cases that you are interested in analyzing further. For example, you can create a variable favorites list that spans across multiple projects, only contains samples for which certain types of data are available, or focuses on specific phenotypic characteristics. A Variable Favorites list can be included in a workbook.

13.8.1 Create Variable Favorites List

To create a variable list from **Your Dashboard**, click on the **Create Variable Favorites** link which will display the **Create Variable Favorite** screen.

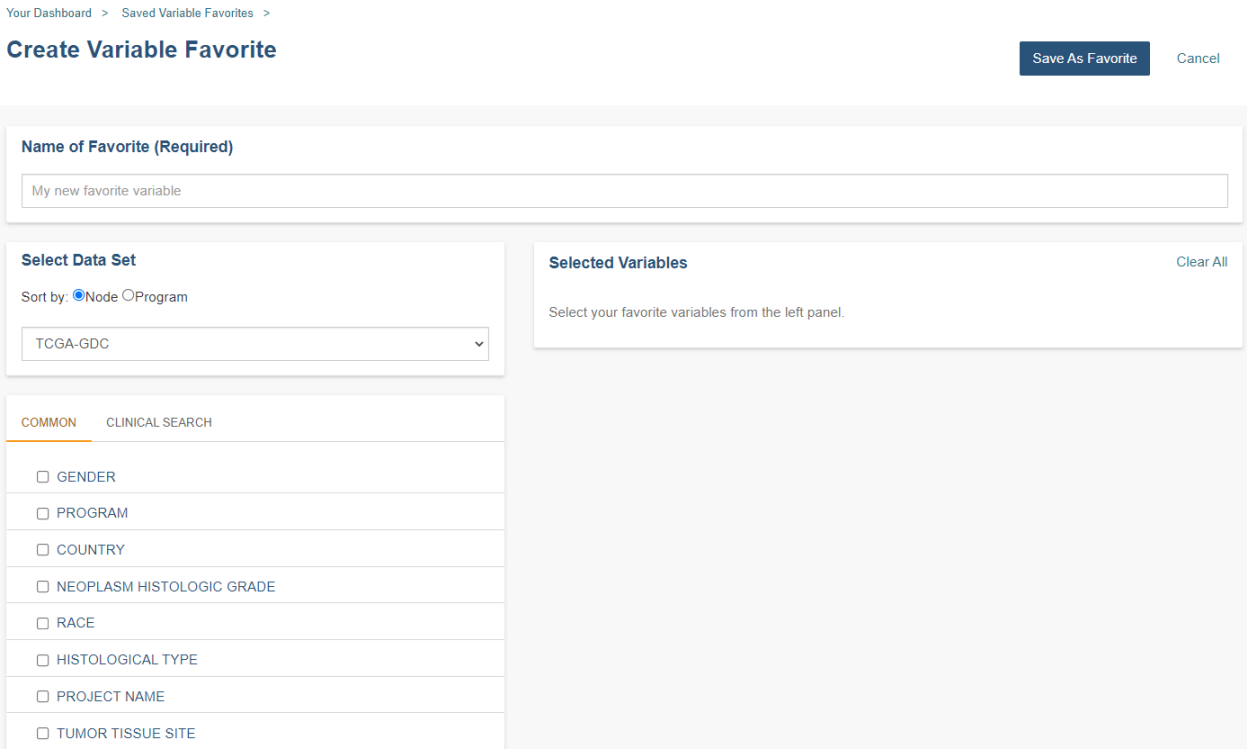
Or, from the menu dropdown, select **Create Variables Favorite List** from the **VARIABLES** menu dropdown.

To create a new Variable Favorite:

- Name your new favorite; you can create many favorites and use them later when working with workbooks.
- Select attributes and features for your variable list by performing one or more of these actions:
  - Select a data set (program) from the **Select Data Set** drop down list. This will display features for that data set under the **COMMON** and **CLINICAL SEARCH** tabs.
  - *Common Filter Selection* - Filters (attributes and features) that are fairly common across programs are displayed under the **COMMON** tab. Changing the data set will change the list of available filters.
    - Check the checkbox adjacent to each feature that you are interested in. They will display in the **Selected Variables** panel.

- *Clinical Filter Feature Search* - Click the **CLINICAL SEARCH** tab. This filter allows the user to search by any clinical feature in the selected data set (program). Changing the data set will change the list of available filters.
  - Enter one or more characters in the **Feature Search** field. A list of features containing these characters displays. Select a feature from the list and it will display in the **Selected Variables** panel.
- *Favorites Filter Selection* - From the Data Set drop down list, select **Favorites**. This displays your existing Variable Favorite lists, and their component features. These features can now be selected for a new Variable Favorite list by checking the checkbox adjacent to each feature that you are interested in. They will display in the **Selected Variables** panel.
- *User Data Filter Search*- From the Data Set drop down list, select **User Data-User**. This option allows you to select from filters that you have uploaded using the upload data functionality. It's separated by projects within your program; a drop down list will display the associated features.

- Verify that all your selected filters are displayed in the **Selected Variables** panel on the right-hand side. Clicking **Clear All** will remove all selected filters.
- Click **Save As Favorite** to save the Variable list.



13.8.2 Manage Variable Favorites List

Selecting **Manage Variable Favorites List** from the **VARIABLES** menu dropdown displays the **Saved Variable Favorites** screen. This screen displays your saved Variables Favorites and allows you to edit or delete them, as well as start a new workbook using your favorite.

- Editing a Variable Favorites List - Clicking the **Edit** button displays the **Edit Variable Favorite** screen, which shows all filters in the selected variable list. Any variables selected will be added to any existing variables in the list. Variables can also be removed from the favorite list. The title of the variable favorite list can be changed. To return to the previous view, you must either save any selected filters, or choose to cancel adding any new filters.

- Deleting a Variable Favorites List - Clicking the Delete button will delete the variable list.
- Apply To New Workbook button - Clicking on the **Apply to New Workbook** button will take you to a screen where you can create a new workbook using your variable list.

13.8.3 Select Variables for a New Workbook

Selecting **Variables for a New Workbook** from the **VARIABLES** menu dropdown displays the **Data Source | Variables** screen. This screen allows you to create a new workbook with the selected variables.

- Click the **Create New Workbook With Selected Variables** button to create a new workbook using your selected variables.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

13.9 Cohorts

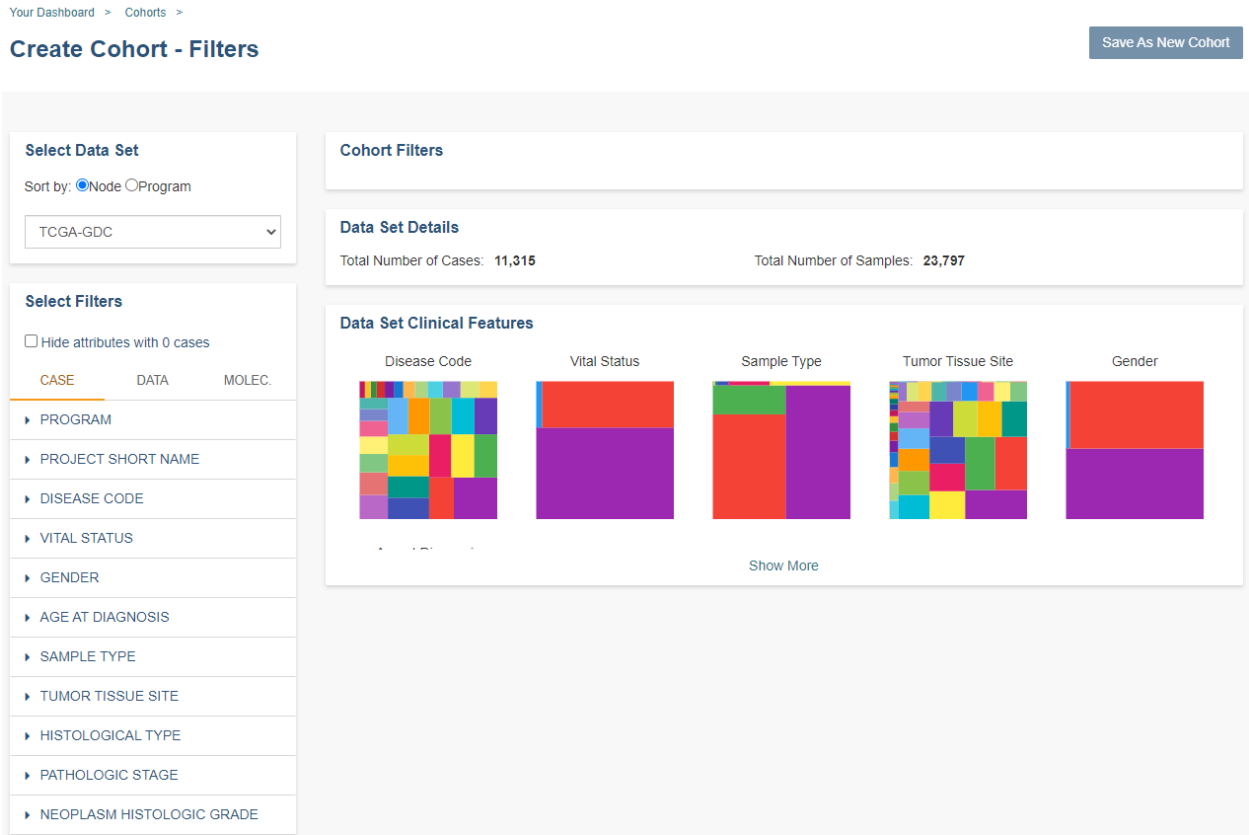
Cohorts are a way of creating custom groupings of the samples and/or cases that you are interested in analyzing further. You may frequently reuse a cohort in multiple analyses. Creating a "saved cohort" allows you to do this. If you have any existing saved cohorts, they will display here for you to view, edit and share.

13.9.1 Create a New Cohort

To create a cohort from **Your Dashboard**, click on the **Create a Cohort** link in the **Saved Cohorts** panel at the bottom of the screen and select either "Filter" or "Barcodes" from the dropdown. The Filter link will display the cohort creation page; filters are explained below. The Barcodes link will display a page where you can upload samples/cases barcodes and create a cohort from them.

If you already have saved cohorts, they will be listed in the **Saved Cohorts** panel. Click on the **Saved Cohorts** link in that panel and a page with the details of your saved cohorts will display. Alternatively, to go directly to a given cohort, click on its name and the cohort details page of that cohort will display.

You can also navigate to these functions by using the drop down options in the **COHORTS** item on the menu bar.



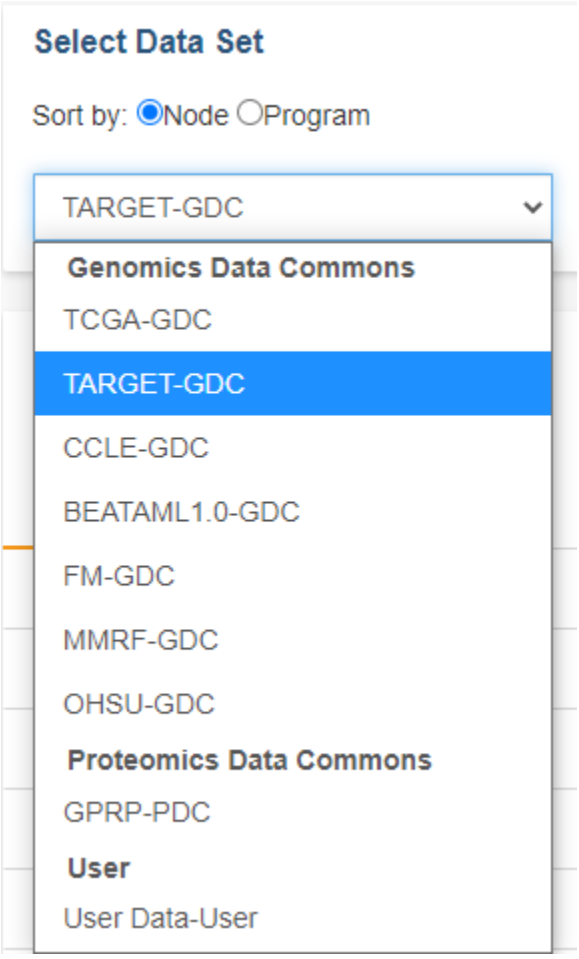
Cohort Creation - Filters

Using the list of data sets and filters on the left, you can select the attributes and features that interest you from ISB-CGC data or user data. You can create a cohort containing multiple programs.

Select Data Set

This panel in the top left of the screen allows you to pick the programs and user data sets that you want included in the cohort.

The drop down list will display the ISB-CGC data sets that the Web App is currently supporting, as well as your user data. By default, the list is sorted by Node (Genomics Data Commons, Proteomics Data Commons, User) with programs listed below each node header. The sort order can be changed by selecting Program next to **Sort By**.



Select Filters

When an ISB-CGC hosted data set is selected, appropriate filters will display under three tabs. All tabs are not available for all programs, but all programs will have some features available on the CASE tab.

- The CASE tab displays clinical and demographic features applicable to the selected program.
- The DATA tab displays data types (ex. Aligned Reads, Copy Number Segment Masked) applicable to the selected program.
- The MOLECULAR tab displays filters pertaining to mutations.

For USER DATA, there is one tab called "PROJECTS & STUDIES" which allow you to filter by the projects or studies you have uploaded to the system.

Click on a filter name to see the selection values. For example, when you click on "Vital Status", it expands and provides a list containing "Alive", "Dead", and "NA" as values you may choose.

Selected filters will display in the Cohort Filters panel. The Data Set Details panel will update the Total Number of Cases and the Total Number of Samples based on the selected filters.

Individual selections within a filter group are "ORed" together, meaning if any of the conditions are met, they will be in the results. On the other hand, filters are "ANDed" together, meaning that data must meet all filter criteria in order to be selected. There may be times where you have no cases and samples in the results, based on the combination of filters you have chosen.

- If you use AND and do not see the data you are expecting in the filter, try OR instead. AND is a more restrictive criteria requiring all filters to be met; OR is less restrictive, requiring only one criteria to be met for the data to display.
- You may want to consider adding the term "AND" or "OR" in your saved cohort title since the type of combination used in your cohort does not display in the filters list for a saved cohort.

Note: Hovering over the Disease Code name will display the disease code long name if it's part of the TCGA, CCLE, or TARGET data set.

Molecular Tab

The Molecular Tab is only available for TCGA data. It enables the user to filter by Gene Mutation Status, creating a cohort based on the presence of a mutation (of various types) in a gene or genes.

To combine multiple gene filters, select AND (requires all filters to be met for the data to be filtered) or OR (at least one criteria needs to be met for the data to be displayed). You can also filter by Genomic Build.

Programs & Projects Tab

The Programs & Projects Tab is only available for User Data. It displays the programs and projects that are part of the user data set.

Cohort Filters Panel

This panel displays the selected filters for the cohort. Filters are listed under the program name. If you click on the program name, the screeb will change to display the information for that program.

Selecting an X beside a single filter will remove that filter. Selecting **Clear All** in the top right of the panel will remove all the filters. Note that you cannot removed filters once the cohort has been saved. (See Set Operations below for more ways to add or remove filters from your cohorts.)

Data Set Details Panel

This panel shows the **Total Number of Samples** and **Total Number of Cases** for the currently displayed data set based on the selected filters. If there is a small "timer" icon, the calculation is taking place; the results should appear soon.

Data Set Clinical Features Panel

This panel shows a list of images (called "treemaps") that give a high level breakdown of the selected samples for a handful of features (ex. Disease Code, Vital Status, Gender, Sample Type, Age at Diagnosis, etc.) for the currently displayed data set based on the selected filters.

By using the **Show More** button, you can see additional tree maps. Mousing over an image shows the details of each specific section of the image and the number of samples associated with it.

Programs & Projects Panel

This panel displays a list of images (called "treemaps") similar to the Data Set Clinical Features panel, but is only available when the User Data tab is selected. This panel displays a high level breakdown of the projects and studies you have uploaded to the system. Hovering over the image will show details of that specific section of the image and the number of samples associated with it.

Saving the Cohort

Click the **Save as New Cohort** button when you are ready to save the cohort based on the filters you have set. You will be asked for a cohort name and the selected filters will be displayed. Enter the name and click the **Create Cohort** button.

NOTE: When working with multiple programs you will see a yellow notification box stating, "Your cohort contains samples from multiple programs. Please note that filters will only apply to samples from the program indicated by the tab they were chosen on - they will not apply to samples from other programs in this cohort."

Cohort Creation - Barcodes

This feature allows you upload or enter your own list of sample or cases barcodes from multiple programs. There is a blue **Show Instructions** button on both the **UPLOAD** and **ENTER** tabs.

Upload Tab

This feature allows uploading files with barcodes to create a cohort. Files must be in GDC Data Portal case manifest format, or in comma/tab-delimited case/sample/program format. The file can be a maximum of 32MB. Also, files must be in tab- or comma-delimited format (TSV or CSV) and have an extension of .txt, .csv, or .tsv. After selecting the file and uploading it, the entries will be validated. Any entries which are found to be invalid will be listed, and you can choose to omit them and continue with cohort creation, or select a new file for verification and upload.

GDC Data Portal Case Manifest Files

GDC Data Portal case manifests can be obtained on the 'Cases' tab of the Exploration section of the data portal here. JSON case manifests must have a .json extension, and will be validated against the GDC's JSON schema. The minimum required properties for each entry in the JSON file are the project object and the submitter_id field. The project object must include the project_id property. All other properties will be ignored.

TSV case manifests must have a .tsv extension, and must contain the first three columns of the GDC TSV case manifest in the following order: Case UUID, Case ID, Project. Any other columns will be ignored. Do not remove the header row of the TSV case manifest.

Because the GDC Data Portal case manifest entries are cases, all samples from a case will be included in the cohort.

Below are the instructions which display when the **Show Instructions** button is clicked.

Your Dashboard > Cohorts >

Create Cohort - Barcodes

UPLOAD ENTER

Upload a GDC Data Portal Case Manifest or a CSV/TSV file of barcodes and programs

Hide Instructions

Files must be in GDC Data Portal case manifest format, or in comma/tab-delimited case/sample/program format. After selecting the file and uploading it, the entries will be validated. Any entries which are found to be invalid will be listed, and you can choose to omit them and continue with cohort creation, or select a new file for verification and upload.

CSV/TSV Barcode Files

All entries must be either a valid `case barcode` ***or*** `sample barcode`. When a case barcode is provided, all samples from that case will be included in the cohort. Make a separate entry per sample if only specific samples from a case should be added to the cohort.

Barcode sets may be in tab-, comma-, or newline-separated format. Values may be placed in single or double quotes, and can be on a single line or on multiple lines.

Tab- or comma-delimited files must have an extension of `.txt`, `.csv`, or `.tsv`. Please do not include any column headers.

Example Barcode File Contents

```
TCGA-N9-A4Q4,TCGA-N7-A4Y8-01A,'Saos-2','Hs 863.T'

"TCGA-N7-A4Y8-01A"
'Hs 863.T', TARGET-51-PAJMFS

LAMA-84
LC-1F
LCLC-103H
TARGET-10-CAAABC-60.3A
TARGET-10-CAAABD-60.2A
TARGET-10-CAAABF-60.11C
```

GDC Data Portal Case Manifest Files

GDC Data Portal case manifests can be obtained on the 'Cases' tab of the Exploration section of the data portal.

JSON case manifests must have a .json extension, and will be validated against the GDC's JSON schema. The minimum required properties for each entry in the JSON file are the `project` object and the `submitter_id` field. The `project` object must include the `project_id` property. All other properties will be ignored.

TSV case manifests must have a .tsv extension, and must contain the Case ID and Project columns. Any other columns will be ignored. ***Do not remove the header row of the TSV case manifest.*** Please note that currently, TSV case manifests will only contain the current page of the 'Cases' Exploration tab.

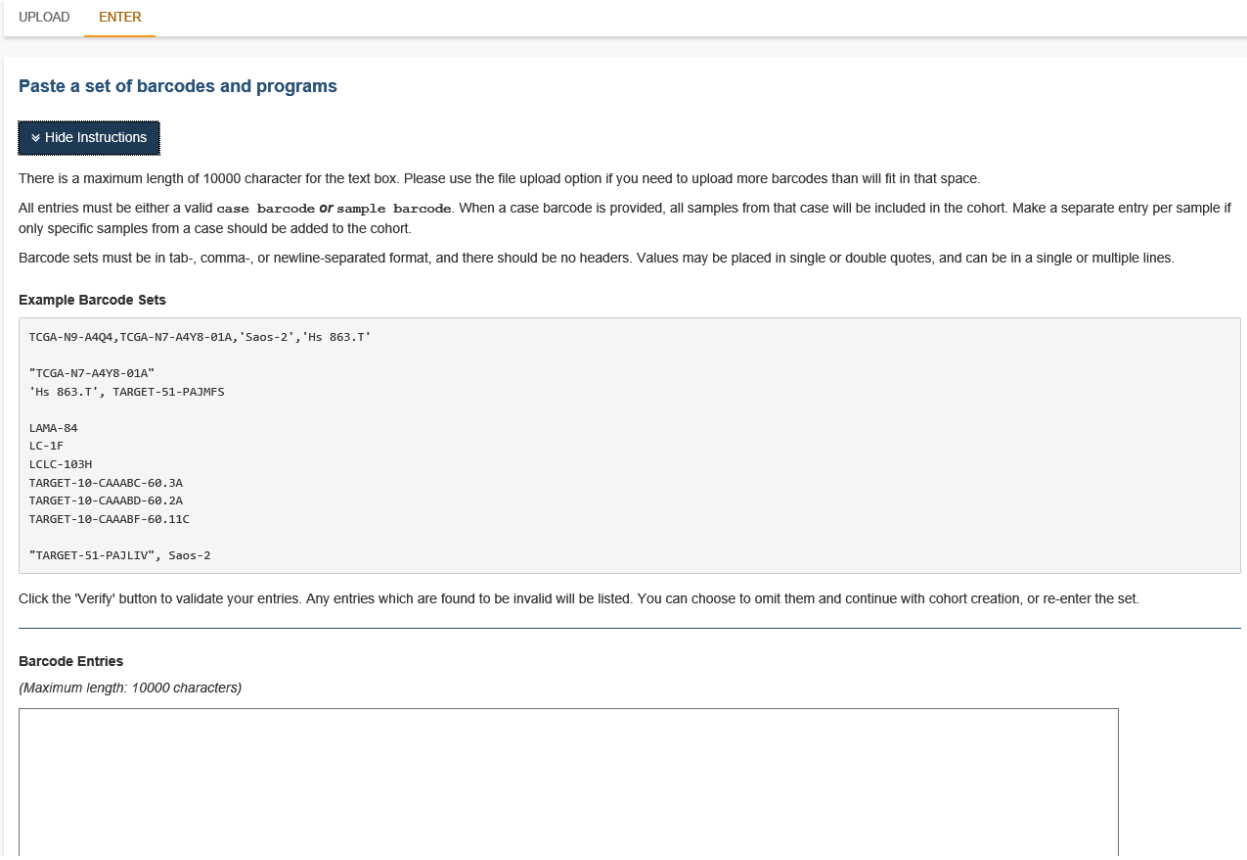
Because the GDC Data Portal case manifest entries are cases, all samples from a case will be included in the cohort.

Upload barcode list or case manifest

Enter Tab

This feature will allow you to manually input barcodes for cohort creation. There is a maximum length of 10000 characters for the text box. Please use the file upload option if you need to upload more barcodes than will fit in that space.

Below are the instructions which display when the **Show Instructions** button is clicked.



13.9.2 Manage Saved Cohorts

Selecting **Manage Saved Cohorts** from the **COHORT** menu dropdown displays the **Cohorts** screen, **SAVED COHORTS** tab. This screen displays your saved cohorts and allows you to view, edit, delete, set operations, and share them. In addition, you can start a new workbook using selected cohorts.

To view a cohort, click on the name of the cohort to display the cohort details. Alternately, you can view the cohort details by clicking on its name in the "Saved Cohorts" panel on the "Your Dashboard" page.

From **Cohorts** screen, **SAVED COHORTS** tab, you can perform the following functions. Except for Set Operations, these functions are described in detail in the Cohort Details Screen section, as they are also available there.

- New Workbook
- Delete
- Set Operations
- Share

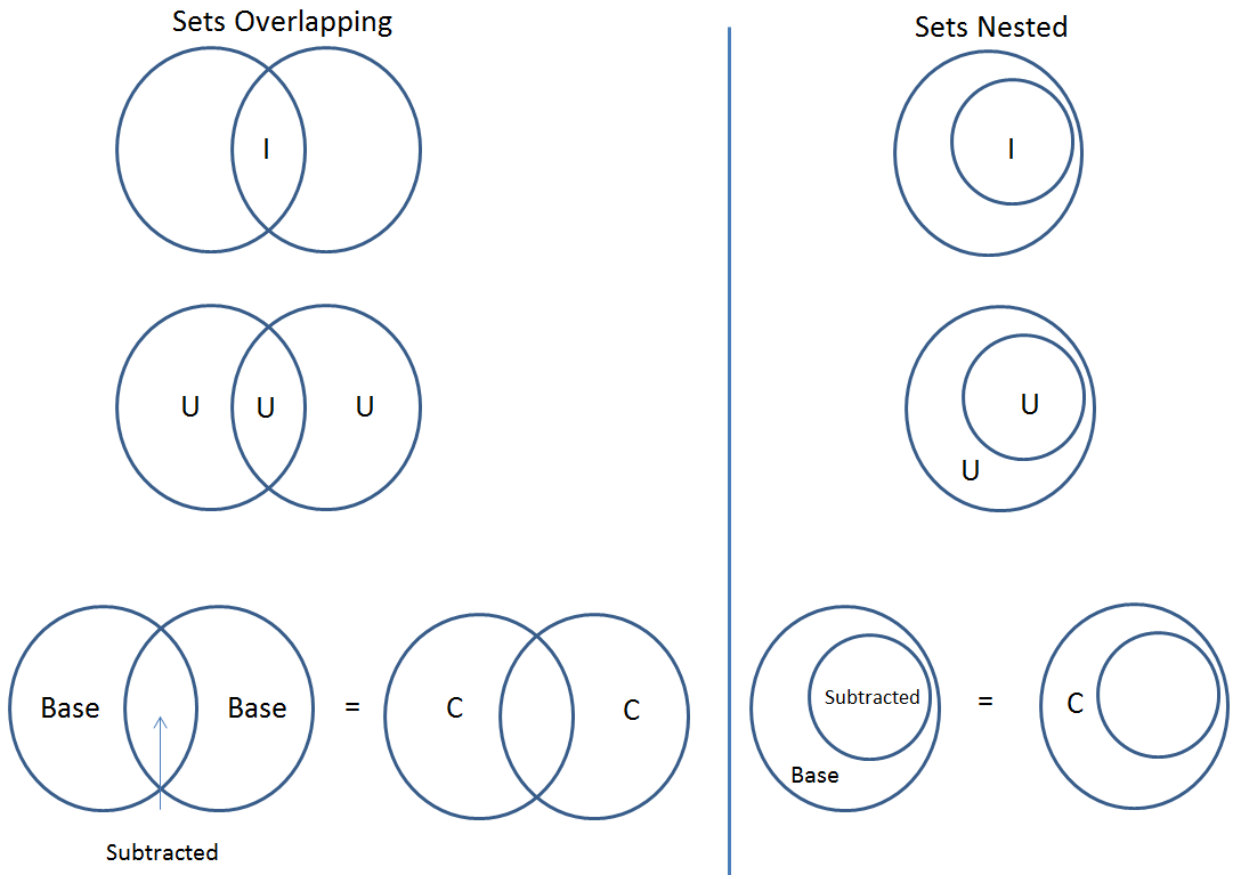
Set Operations

Clicking the **Set Operations** button displays a **New Cohort** screen where you can create new cohorts from two or more existing cohorts using the union, intersection or complement operations. The Set Operations button will only be available if at least two cohorts are selected on the **Cohorts** screen.

On the **New Cohort** screen, enter a name for the new cohort and select a set operation. The intersect and union operations can take any number of cohorts and in any order. The complement operation requires that there is a base cohort, from which the other cohorts will be subtracted. Click **Okay** to complete the set operation and create the new cohort.

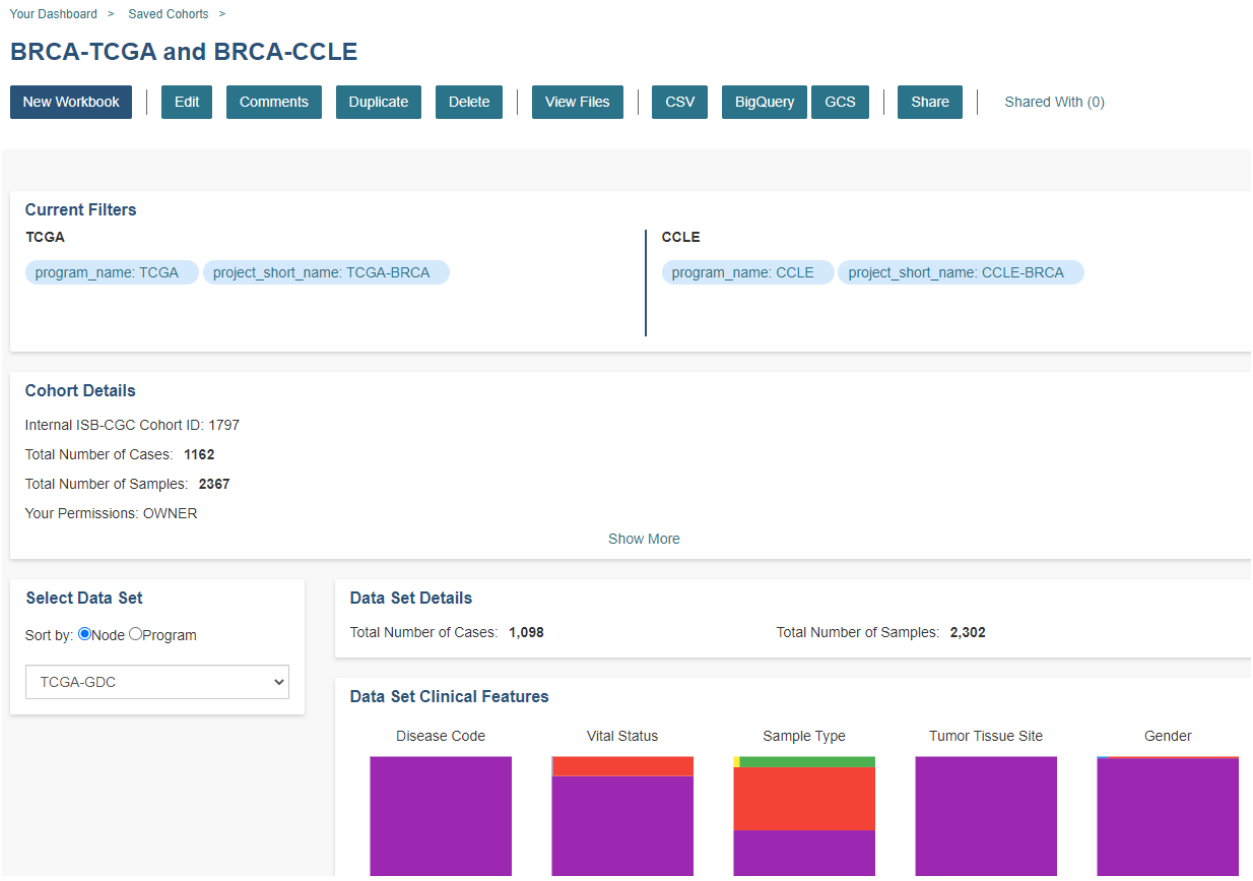
Note: To combine the user uploaded data and the ISB-CGC data, use the Set Operations function. This is possible because the list of barcodes is what is used to create the set operation. For example, to make a cohort of user data samples and ISB-CGC curated samples, Set Union must be used, and to filter user data which is an extension of TCGA or TARGET samples, Set Intersection must be used.

The figure below shows what the results of the set operations will be (represented by I for Intersect, U for Union, and C for Complement). There are two types of sets shown, those that overlap (on the left) and those that are nested (on the right). For the last row (complement operations), the "Subtracted" area is removed from the "Base" area to result in the Complement (C).



13.9.3 Cohort Details Screen

The cohort details screen displays the details of a specific cohort. The title of the cohort is displayed at the top of the page.



The screen is divided into the following sections:

Current Filters Panel

This panel displays current filters on this cohort or any of its ancestors. Saved filters cannot be removed, but new ones can be added using **Edit**.

Cohort Details Panel

This panel displays the Internal ISB-CGC Cohort ID (the identifier you use to access this cohort through the APIs), and the number of samples and cases in this cohort. The number of samples may be larger than the number of cases because some cases may have provided multiple samples. This panel also displays "Your Permissions" which can be either Owner or Reader, as well as Revision History. If you have edited the cohort, the filters that were used to originally create the cohort are displayed under the "Creation Filters" header. The newly applied filters (after original creation) are displayed under the "Applied Filters" header.

Select Data Set

This panel displays all the programs and user data sets that are included in the cohort; click on the drop down to see them.

By default, the list is sorted by Node (Genomics Data Commons, Proteomics Data Commons, User) with programs listed below each node header. The sort order can be changed by selecting Program next to **Sort By**. To see details about a program or data set, select it from the drop down list.

Data Set Details Panel

This panel shows the **Total Number of Samples** and **Total Number of Cases** for the currently displayed data set (selected from the Data Set drop down) based on the selected filters.

Data Set Clinical Features Panel

This panel shows a list of images (called "treemaps") that give a high level breakdown of the selected samples for a handful of features (ex. Disease Code, Vital Status, Gender, Sample Type, Age at Diagnosis, etc.) for the selected program.

By using the "Show More" button, you can see additional tree maps. Mousing over an image shows the details of each specific section of the image and the number of samples associated with it.

Cohort Details Screen functions:

Create a New Workbook

Clicking the **New Workbook** button brings you to a screen where you can create a new workbook using this cohort.

Edit a cohort

Clicking the **Edit** button displays the Filters panel. Any filters selected will be added to existing filters. To return to the previous view, save any newly selected filters using the **Save Changes** button, or cancel adding any new filters by clicking the **Cancel** link.

Comment on a cohort

Clicking the **Comments** button displays the Comments panel. Here anyone who can see this cohort (such as an owner or someone who has shared access to the cohort) can comment on it. Comments are shared with anyone who can view this cohort. They are ordered by newest on the bottom. Selecting the "X" on the Comments panel will close the panel.

Copy a cohort

To create a copy of the cohort, click on the **Duplicate** button. This will take you to a new copy of the cohort which has the same list of samples and cases; you will be the owner of the copy.

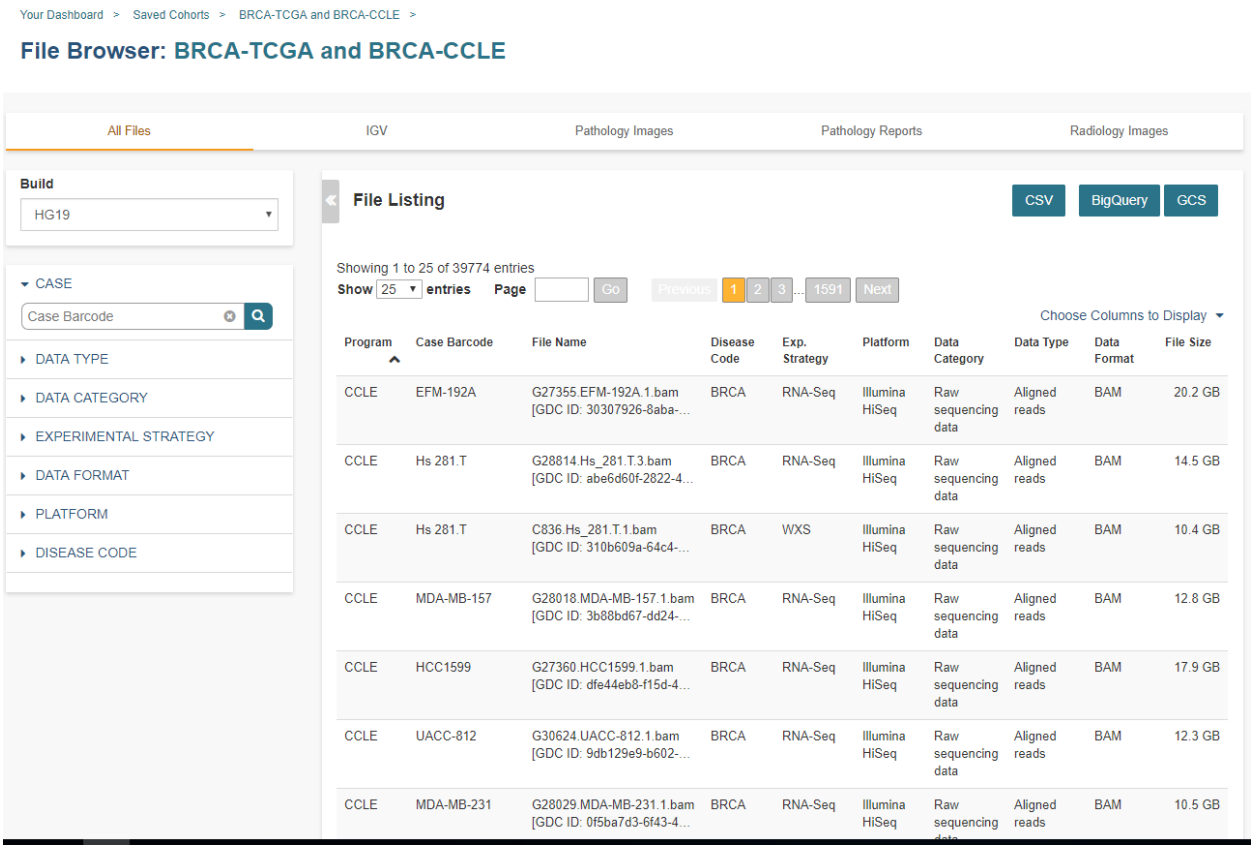
This is how you create a copy of another researcher's cohort that they have shared with you. (Note: If they later change their cohort, your cohort will not be updated; it will remain the same as it was at the time you duplicated it).

Delete a cohort

Click the **Delete** button to delete the cohort. Confirm by clicking the second **Delete** button presented.

File Browser

Clicking the **File Browser** button displays a screen with a list of data files associated with your current cohort. This list includes all files which are stored on the Google Cloud, including both controlled access and open access data.



You can use "Show", "Page", "Previous" and "Next" to navigate through the list. The columns are sortable by selecting the column header. You can select a subset of the default columns to show by using the "Choose Columns to Display" tool.

You can filter by Genomic Build (HG19 or HG38) and view which platforms and files are available for the build selected.

You can filter by full or partial Case Barcode on all tabs. To remove the search key word, click the "X" button adjacent to it. Filtering by Case Barcode updates the number to the right of all the other filters.

You may also filter by data type, data format, platform, disease code, disease strategy, and/or experimental strategy. Selecting a filter will update the associated list. The numbers next to the filter refers to the number of files available for that filter.

The tabs "IGV", "Pathology Images" and "Radiology Images" allow you to filter for files that show you respectively read-level sequence data (viewed using the IGV viewer), pathology images, and radiology images. Please note: only if you have authenticated as a dbGaP authorized user will you be able to select controlled access files to view in the IGV viewer (CCLE data does not require authorization to view the sequence data in the IGV viewer). Details of how to view Sequences, and Pathology and Radiology Images are provided below.

Viewing a Sequence

When available, sequences in a cohort can be viewed using the IGV viewer. To find those sequences that can be viewed, select the **IGV** link on the **File Browser** screen. The File Listing panel will display the files that can be viewed with the IGV viewer. Selecting the checkbox in the "View" column (maximum of file files) and clicking the **Launch IGV** button in the upper panel will display an IGV view of the selected sequence(s) data.

Controlled access files will be viewable by sequence ONLY if you have authenticated as a dbGaP-authorized user.

More information about Viewing a Sequence in the IGV Viewer.

Using the Image Pathology Viewer

Note: All tissue slide images from the TCGA program are currently unavailable for viewing. (Diagnostic images will display.)

When available, pathology images can be viewed using the caMicroscope tool (see more about caMicroscope provide here). These are the pathology images that are associated with TCGA samples. To find images that can be viewed, open a saved cohort and select the **File Browser** button. You can also select the **File Browser** link from the Dashboard Saved Cohorts panel. The files associated with your cohort will be shown. Click on **Pathology Images** to see a list of available pathology images. Hovering over the File Name and clicking on "Open in caMicroscope" will open the image file in a new tab using caMicroscope. (HINT: using a smaller cohort will provide faster response in creating the list of files available).

To zoom into the image, either click the left button or use your wheel to zoom in. Use your mouse to move around the image. To zoom out of the image, shift-slick the left mouse button or use your wheel to zoom out. Selecting caMicroscope at the top of page will send you to the caMicroscope homepage. If you hover over the Slide Barcode section on the top right hand side you will see metadata information listed.

Viewing a Radiology Image

To find images that can be viewed, open a saved cohort and select the **File Browser** button. You can also click the **File Browser** link from the Dashboard Saved Cohorts panel. The files associated with your cohort will be shown. Click the **Radiology Images** tab to view a list of available radiology images. Hovering over the Study Instance UID column and clicking on "Open in CHIF Viewer" will open the series Selection panel in a new tab using Osimis DICOM. (HINT: Using a smaller cohort will provide faster response in creating the list of files available.)

For a more detailed step-by-step process of Viewing Radiology Images using the Osimis DICOM viewer please go here.

Download File List as CSV

To download a list of files that are part of this cohort, select the **CSV** button in the upper right on the File Listing panel (on all tabs) on the **File Browser** screen.

The file contains the following information for each file:

- Case Barcode
- Sample Barcode
- Program
- Platform

- Experimental Strategy
- Data Category
- Data Type
- Data Format
- Genomic Data Commons(GDC) File UUID
- Google Cloud Storage(GCS) location
- Genomic Data Commons(GDC) Index
- Index File Google Cloud Storage(GCS) location
- File Size
- Access Type (open or controlled access)

Export File List to BigQuery

To export the File List to BigQuery, select the **BigQuery** button on the **File Browser** screen. You will need to have registered a Google Cloud Project and a BigQuery dataset to be able to export to BigQuery. More information on how to register a BigQuery Dataset can be found here. You can either make a new table or append to an existing table. You can also give the table a unique name; if left blank, a name will be provided for the table.

The table will contain the following information (for each of the data type tabs):

- row
- cohort_id
- case_barcode
- sample_barcode
- project_short_name
- date_added
- build
- gdc_file_uuid
- gdc_case_uuid
- platform
- exp_strategy
- data_category
- data_type
- data_format
- cloud_storage_location
- file_size_bytes
- index_file_gdc_uuid
- index_file_cloud_storage_location

Export File List to Google Cloud Storage

To export the File List to Google Cloud Storage (GCS), select the **GCS** button on the **File Browser** screen. You will need to have registered a Google Cloud Project and a GCS Object to be able to export to GCS. More information on how to register a GCS bucket can be found here. You can also give the object a unique name; if left blank, a name will be provided for the bucket. You will be able to select either CSV or JSON as the file format for exporting into Cloud Storage. All exported files are converted into zip files.

The file will contain the following information (for each of the data type tabs):

- sample_barcode
- case_barcode
- cloud_storage_location
- file_size_bytes
- platform
- data_type
- data_category
- exp_strategy
- data_format
- gdc_file_uuid
- gdc_case_uuid
- project_short_name
- cohort_id
- build
- index_file_storage_location
- index_file_gdc_uuid
- date_added

Cohort export to CSV

Click the **CSV** button to download the cohort in CSV format. The file will contain a list of sample and cases IDs in the cohort.

Cohort export to BigQuery

Clicking the **BigQuery** button allows you to create a new table or append to an existing table. You must have registered a BigQuery data set with a Google Cloud Project on the registered Google Cloud Projects details page. More information on how to register a BigQuery data set can be found here.

If a user wants to export a cohort to their own premade table, it is required to have the following columns:

```
{
      'fields': [
          {
              'name': 'cohort_id',
```

(continues on next page)

(continued from previous page)

```
            'type': 'INTEGER',
            'mode': 'REQUIRED'
        },{
            'name': 'case_barcode',
            'type': 'STRING',
            'mode': 'REQUIRED'
        },{
            'name': 'sample_barcode',
            'type': 'STRING',
            'mode': 'REQUIRED'
        },{
            'name': 'project_short_name',
            'type': 'STRING',
            'mode': 'REQUIRED'
        },{
            'name': 'date_added',
            'type': 'TIMESTAMP',
            'mode': 'REQUIRED'
        },{
            'name': 'case_gdc_uuid',
            'type': 'STRING'
        }
    ]
}
```

Note: You shouldn't ever set UUID to 'required' because sometimes a sample doesn't have a UUID, and the attempt to insert a 'null' will cause the cohort export to fail.

Cohort export to Cloud Storage

Clicking the **GCS** button allows you to save the details of the cohort in a specified Google Cloud Storage location. You must have a registered Google Cloud Storage (GCS) bucket with a Google Cloud Project on the registered Google Cloud Projects details page. More information on how to register a GCS bucket can be found here. You will be able to select either CSV or JSON as the file format for exporting into Cloud Storage. All exported files are converted into zip files.

Share a cohort

Clicking the **Share** button allows you to share the cohort in the Web App with users you select by entering the user's email.

If the email address you entered is not registered with ISB-CGC, a message displays, "The following user emails could not be found; please ask them to log into the site first:(email entered)."

13.9.4 Public Cohorts

Selecting **Public Cohorts** from the **COHORT** menu dropdown displays the **Cohorts** screen, **PUBLIC COHORTS** tab. This screen displays details about any public cohorts currently available in the Web App. It displays the cohort name, number of cases, number of samples and the last date each program was updated. Public cohorts can be used for "New Workbook" and "Set Operations".

To create new workbooks based on a public cohort, check the checkbox adjacent to the public cohort and click on the **New Workbook** button.

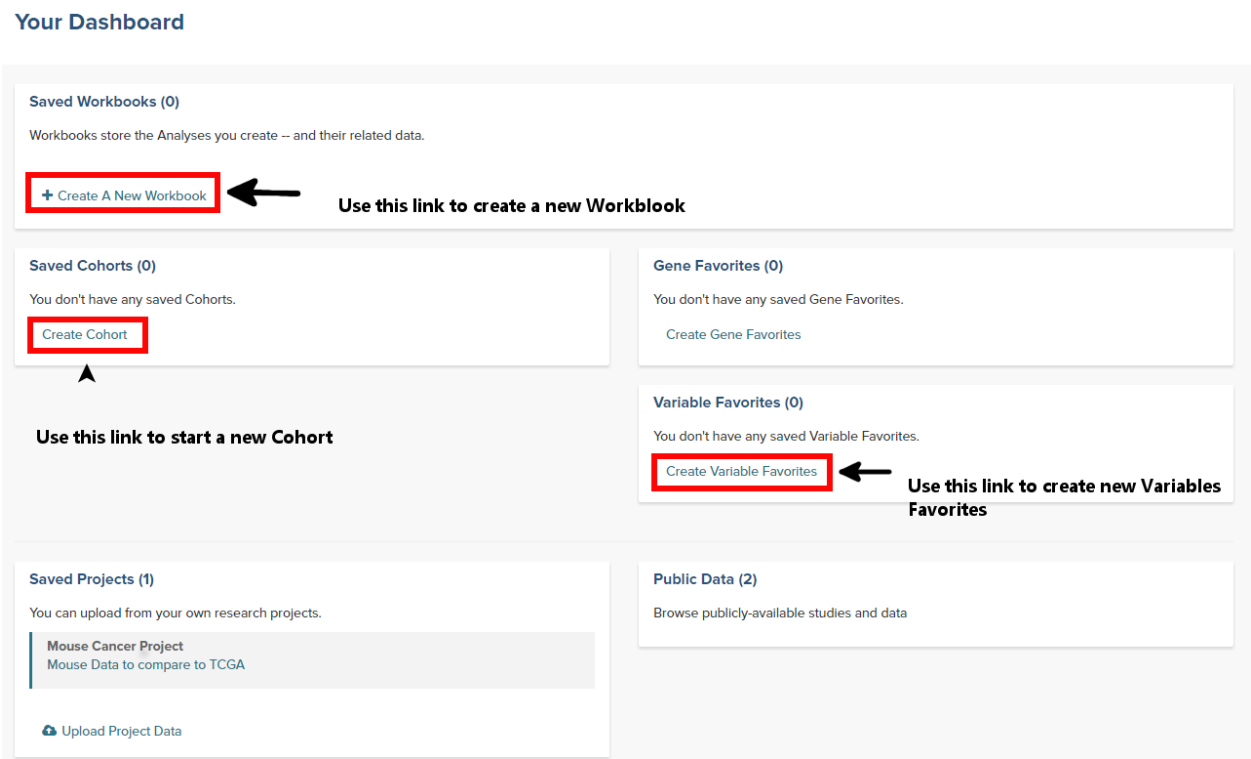
Have feedback or corrections? Please email us at feedback@isb-cgc.org.

13.10 Graphing User Data

Once a user has uploaded their own data to the Web App, that data can be visualized using the same graphing tools that are available for graphing TCGA, CCLE, TARGET and BEATAML1.0 data. However, the process for graphing user data is slightly different from how it is done for that data.

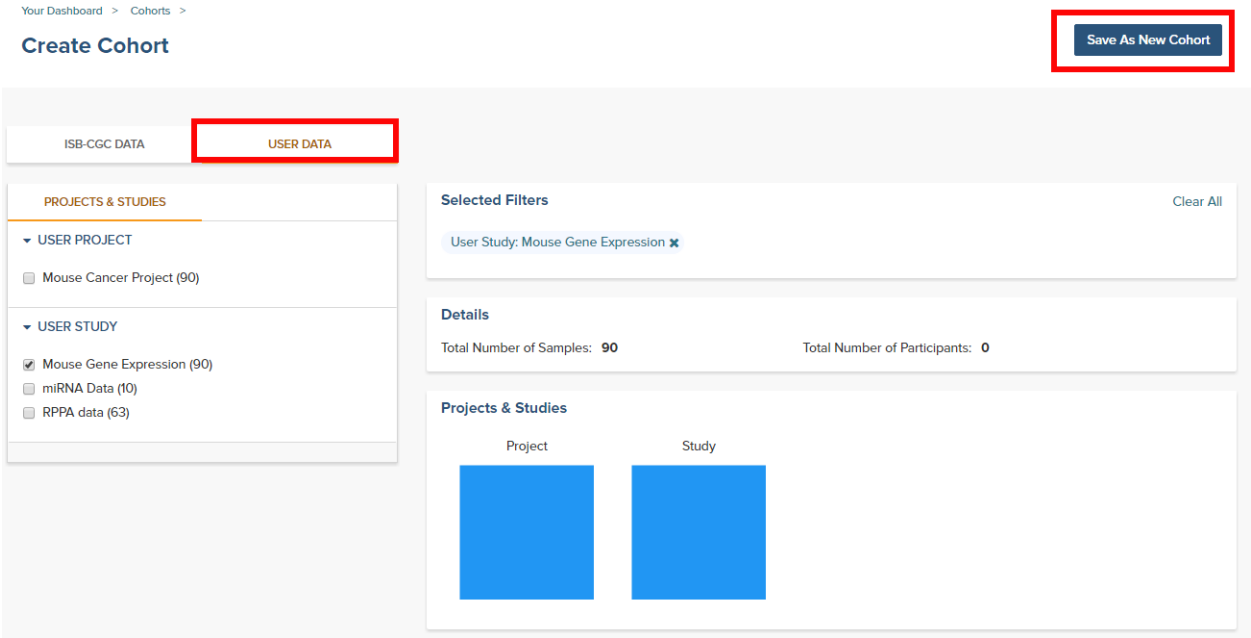
13.10.1 Important sections on the Web App Dashboard

The boxes in the figure below are links that are used to graph user data.



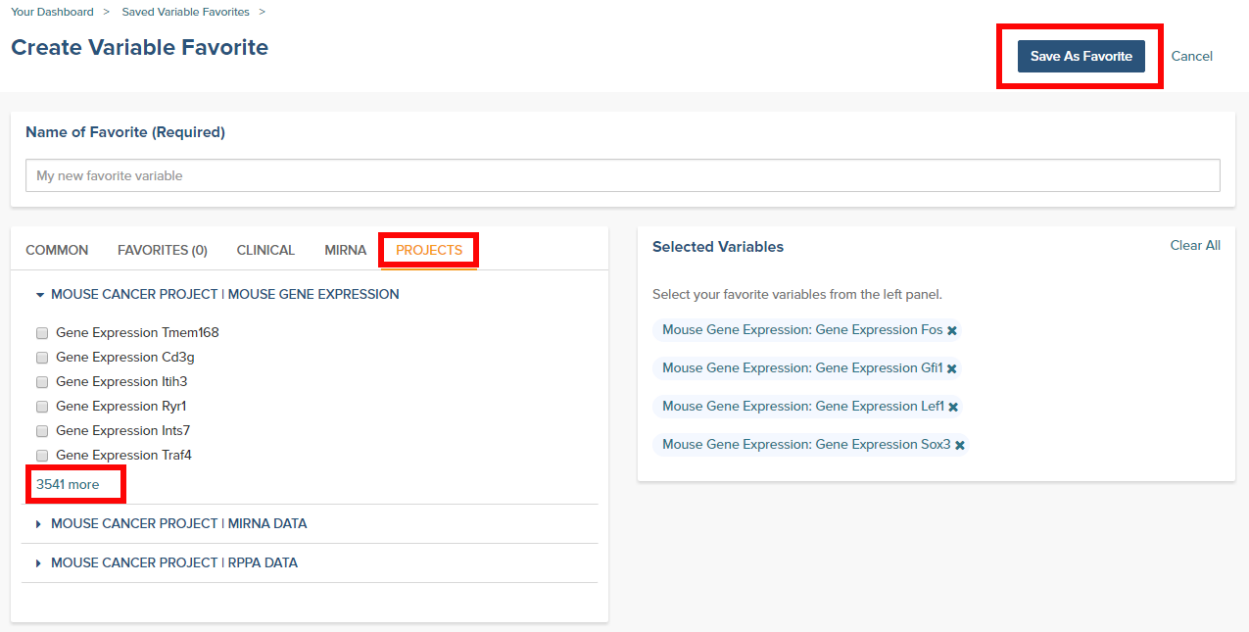
13.10.2 Step 1: Create a Cohort from your project

- From the Web App Dashboard, click on **Create Cohort**.
- Click on the **User Data** tab and select the project or study that will be the cohort.
- Save as a new cohort.



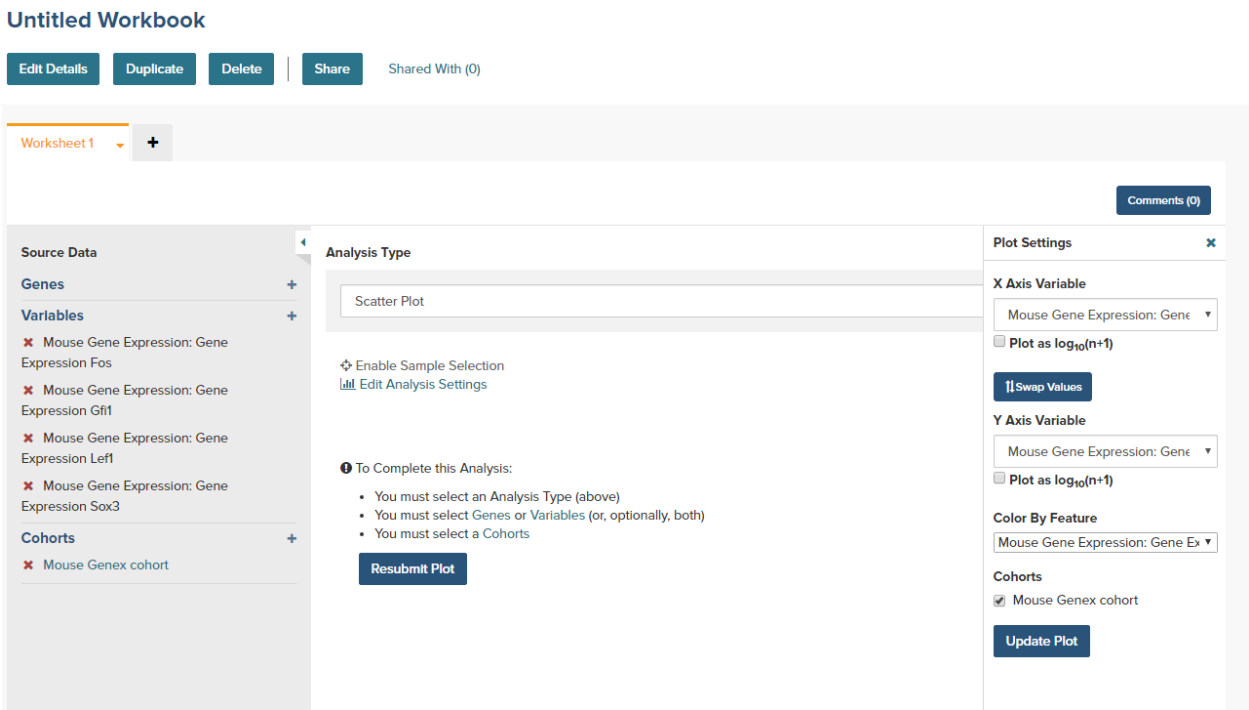
13.10.3 Step 2: Create a Variables Favorite

- From the Web App Dashboard, click on **Create Variable Favorites**.
- Click on the **Projects** tab to see the user supplied studies.
- Select the variables that will be available to graph. Note that if the study has a large number of selections, using the browser search function can help locate the item.
- Give the variables a name and click on the **Save as Favorite** button.

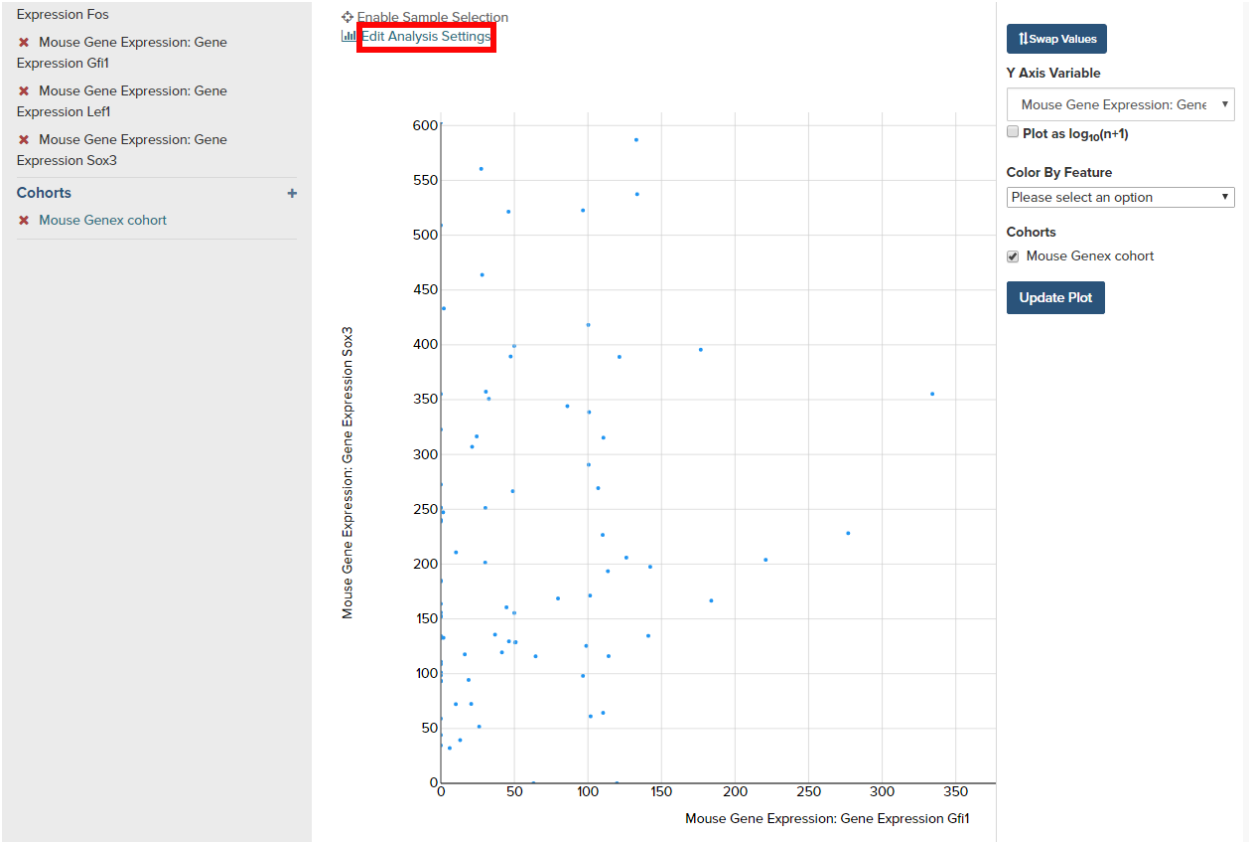


13.10.4 Step 3: Graph the Favorites in a Workbook

- From the Web App Dashboard, click on **Create a new Workbook**.
- Under the **Source Data** heading, select the Variables and Cohorts that you wish to use in the graph. In each case you will be brought to a page listing all of the available Variables or Cohorts. Simply select the desired ones and then click the **Add to Workbook** button.
- Under the **Analysis Type** heading, select the appropriate graph type. This will cause a window to slide in from the right.
- Fill in the X and Y axis variables, select a variable to use for coloring and finally select the cohort to use.



- Click on the **Update Plot** button to have the system gather the data and generate the plot.
- If changes need to be made to the plot, click on the **Edit Analysis Settings** link to bring back the graph dialog box.



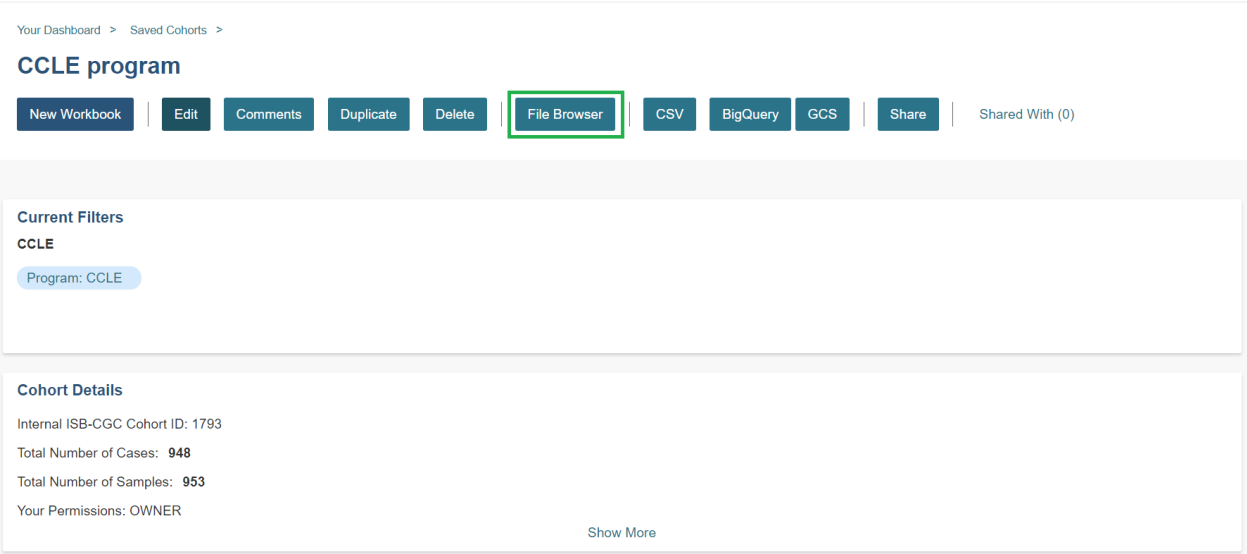
Have feedback or corrections? Please email us at feedback@isb-cgc.org.

13.11 Integrative Genomics Viewer (IGV)

IGV is a widely used interactive tool for exploring genomic data. A web-based version is integrated into the ISB-CGC Web App, and the IGV desktop version can also be used to access cancer data in Google Cloud Storage (GCS). For more information about IGV, please follow the links in the Acknowledgments section at the bottom of this page.

13.11.1 Accessing the IGV Browser from the Web App

To access IGV, first select a cohort and then go to the cohort file list page by clicking on the "File Browser" button at the top of the page.

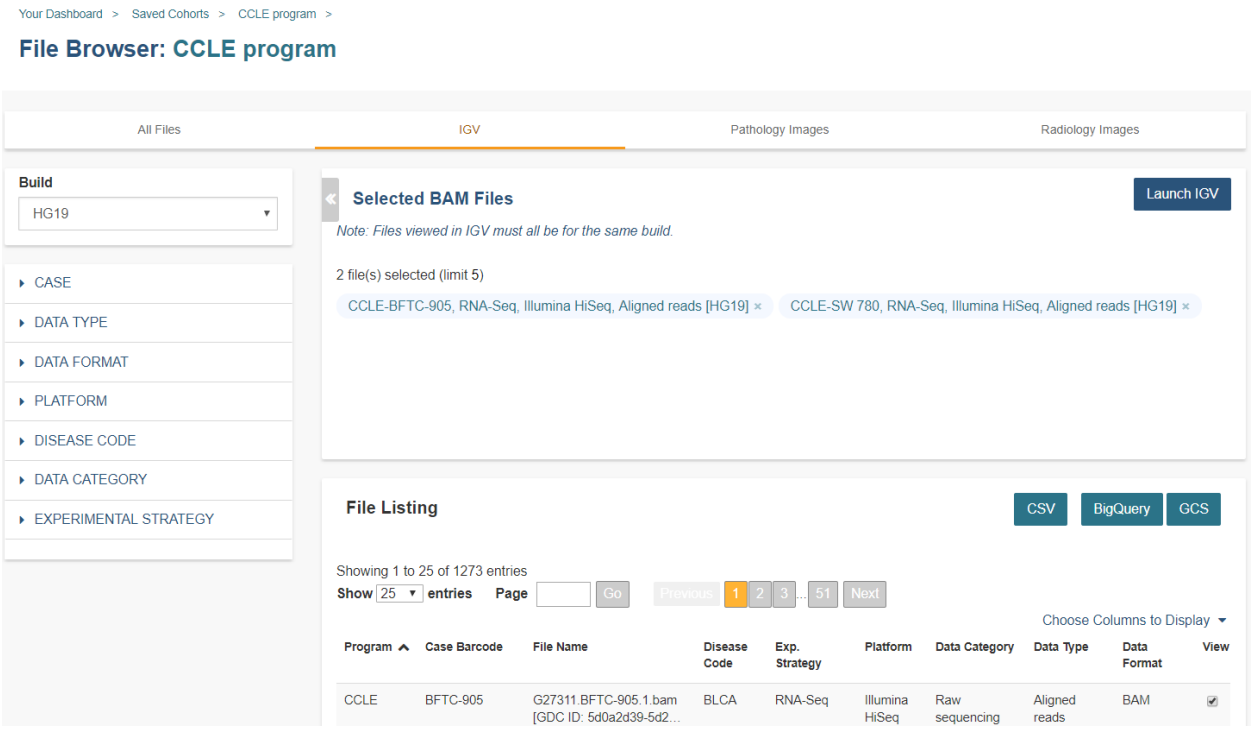


On the File Browser page, click on IGV in the top menu bar.

The resulting file list can be filtered using the Build (HG19 or HG38) and the other filters listed on the left. Click the View checkbox (far right column) for each file that you want to view in IGV. Sometimes the checkbox cannot be checked; here are some reasons why:

- Many files viewable in IGV may require that the user have dbGaP authorization to view controlled access data. If the user has been authenticated and authorized through the user details page, the user will be able to select files. Otherwise the cursor will be disabled when the user hovers over a checkbox. Open source data such as the CCLE project do not require dbGaP authorization and can be viewed by any authenticated user.
- Only a maximum of five files can be selected for viewing at a time.

To view the selected files in the IGV Browser, click on the "Launch IGV" button in the upper right of the window.



NOTES:

- You will only be able to view controlled access sequence files if you have logged in as a registered dbGaP authorized user.
- You will need to disable your browser pop-up blocker to view files with IGV. If you see a 403 error when using the IGV viewer, the pop-up blocker is the cause of that error. Turn off the blocker and try again.

13.11.2 Using IGV Desktop Application to View Aligned Reads in Google Cloud Storage

You can also download and use the IGV desktop application to view aligned reads stored in BAM files in Google Cloud Storage. To do this, download the most recent version of IGV. After launching IGV, go to the "Settings" menu to enable the Google Menu item in the application (directions on how to do this).

To load BAM files from ISB-CGC Google Cloud Storage, use the "File" > "Load from URL…" menu item in the IGV application, entering the path to the bam file in GCS. Paths to BAM files stored by ISB-CGC can be found using the `cohorts().cloud_storage_file_paths()` and `samples().cloud_storage_file_paths()` APIs described here.

NOTE:

- You will only be able to view controlled access sequence files if you have logged in as a registered dbGaP authorized user.

13.11.3 Acknowledgments

The copyright to the Integrative Genomics Viewer is held by the Broad Institute, and the software has been released under the MIT License. For more information about IGV please see the IGV home page or the IGV github repo.

We are grateful to the IGV team for their assistance in integrating IGV into the ISB-CGC Web App.

Robinson J T, Thorvaldsdottir H, Winckler W, Guttman M, Lander E S, Getz G & Mesirov J P, *Integrative genomics viewer*, Nature Biotechnology 29, 24-26 (2011).

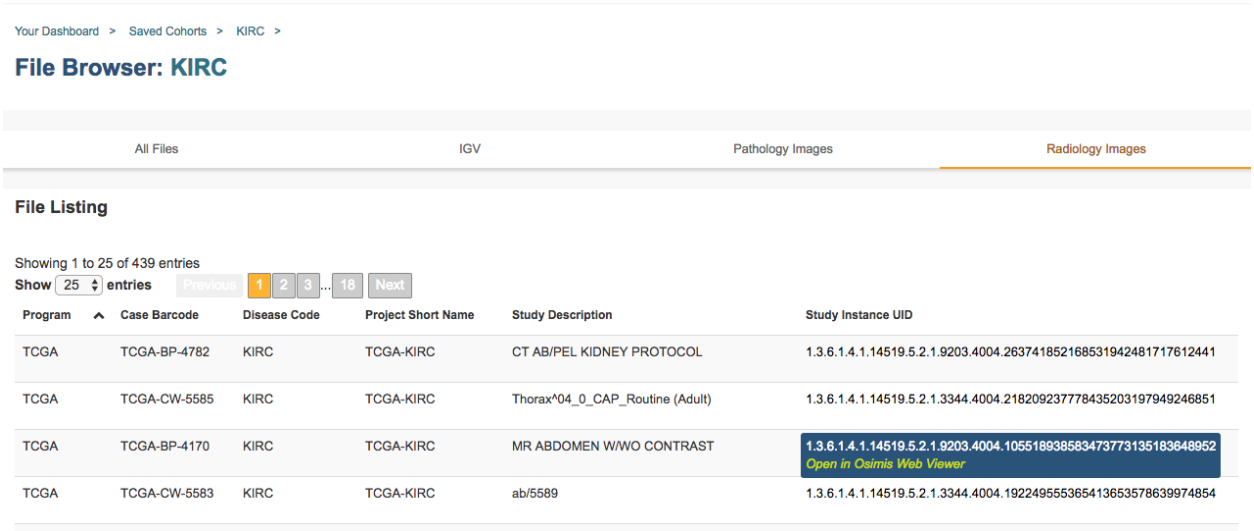
Thorvaldsdottir H, Robinson J T, Mesirov J P, *Integrative Genomics Viewer (IGV): high-performance genomics data visualization and exploration*, Briefings in Bioinformatics 14, 178-192 (2013).

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

13.12 Radiology Viewer

Radiology images are viewed in an Osimis Web Viewer, a plug-in to the Orthanc Image Server (Orthanc). The ISB-CGC web application uses an instance of Orthanc to manage radiology files for the purpose of viewing. Currently only DICOM formatted files from TCGA samples are available for viewing. It may be helpful to review the DICOM Model of the Real World to understand the relationship between patient DICOM studies, DICOM series and DICOM instances.

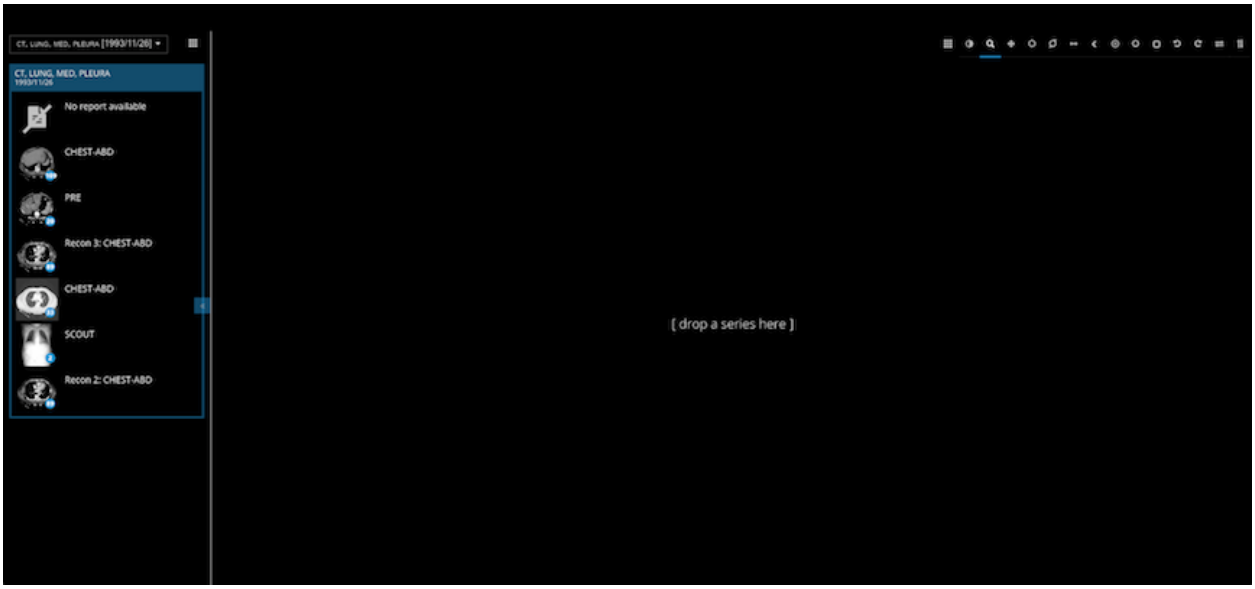
The ISB-CGC web application File Browser page presents a table of DICOM studies associated with patients in some cohort.



13.12.1 Viewer Components

Clicking on a study in the table opens an Osimis Web Viewer in a new tab:

All the DICOM series which comprise the selected study are shown as thumbnail images in the Series Selection Panel. Note that it can take several seconds for these thumnails to appear. By default, the thumbnails are laid out in a grid pattern. This can be changed to a list format by clicking on the **List Display** button above the thumbnails. The list format displays a description of each series.



Change back again to the grid pattern by clicking on the **Grid Display** button .

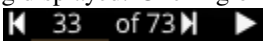
To the lower right of each thumbnail is a small blue circle in which is displayed the number of DICOM instances which comprise the corresponding DICOM series. In addition, when your cursor hovers over a thumbnail, the viewer cycles through the instances comprising the series. (Note these low resolution images are loaded in the background and may not be available for cycling immediately after the viewer window opens.)

To view a larger rendering of a series, drag its thumbnail into the viewport. The first instance of the series is immediately displayed.

At the same time, at the bottom of the viewport you will notice a grid comprised of a series of rectangles corresponding to the instances in the series. The color of the tabs indicates the following:

- Black: The corresponding instance is not yet available for viewing
- Red: A reduced resolution image of the instance is available for viewing
- Green: The full resolution image of the instance is available for viewing

Typically, the viewer loads reduced resolution images for all series as quickly as possible. It loads full resolution images only when a series is dragged into the viewport.

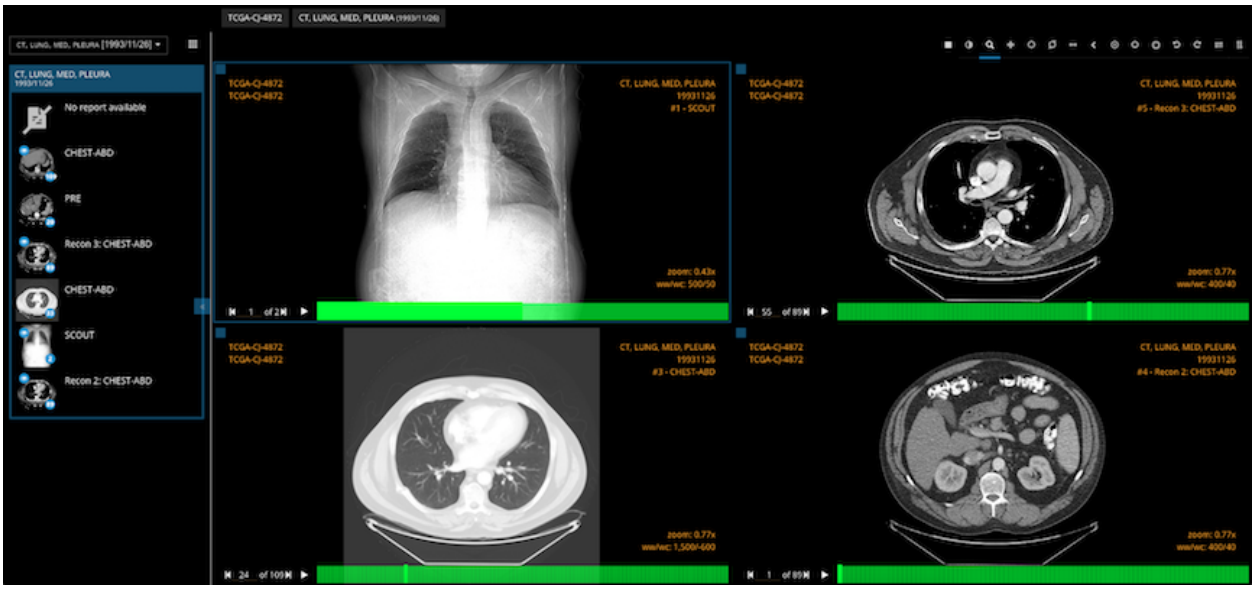
When instance images have been loaded, you can scroll through the instances using your mouse's thumbwheel or equivalent. As you scroll, the grid at the bottom of the screen highlights the instance currently being displayed. Clicking on a rectangle in the grid causes the corresponding instance to be displayed. The **Play Controls** 

in the lower left corner of the main window enable you to single step forward or backward through the series, and to cycle through the series repeatedly. A frame rate slider pops up when you hover over the play button.

13.12.2 Viewing Functions

A set of buttons above the viewport provides a range of functions.

The **Layout** button controls subdividing the viewport for the simultaneous display of one, two or four series. Drag a series into any of the subviewports to display it. Clicking in a subviewport gives it focus for mousewheel and cursor drag operations.



Of the remaining buttons, some are modal, changing the effect of the cursor drag function. A blue line underscores the currently selected mode. Other buttons immediately perform some operation on the subviewport that has focus.

- The **Invert Color** button immediately inverts the colors of the series in the (sub)viewport having focus.
- The **Zoom** button is modal. When selected, dragging the cursor with mouse button depressed expands or contracts the series in the (sub)viewport having focus. Expansion/contraction is around the cursor position at which dragging begins.
- The **Pan** button is modal. When selected, dragging the cursor with mouse button depressed causes panning of the series in the (sub)viewport having focus.
- The **Windowing Presets** button operates both modally and immediately. Hovering the cursor over the button displays a list of windowing presets, one of which can be selected by clicking on it. The selection immediately sets Window Width (WW) and Window Center (WC) values for the series in the (sub)viewport having focus. The WW,WC value pair specifies a linear conversion from stored pixel values to values to be displayed. See here for further information on Window Center and Window Width.

  DICOM instances generally include WW,WC value pairs and these are used by default. Other WW,WC value pairs that may be appropriate for specific cases can be selected on the pop-up. The *Preset #1* selection restores WW,WC to the DICOM setting.

  The Windowing Presets button operates modally when clicked. In this mode, dragging the cursor left or right in a (sub)viewport changes the Window Width value applied to the series in that (sub)viewport. Dragging the cursor up or down in a (sub)viewport changes the Window Center value applied to the series in that (sub)viewport.
- The **Magnifying Glass** button is modal. Hovering the cursor over the button displays a pop-up containing two sliders that control the magnification level and size of a virtual magnifying glass. When selected, dragging the cursor with mouse button depressed opens a virtual magnifying glass that displays a magnified rendering of the underlying image in the region of the cursor.
- The **Length Measurement** button is modal. When selected, the distance in physical units between two points in an instance can be measured. To perform a measurement, click the mouse button once with the cursor over some point of interest, and then again over a second point of interest. Alternatively, depress and hold the mouse button while the cursor is over the first point of interest, then release the mouse button while the cursor is over the second point of interest. A line joining the two points and its length are displayed. The line will scale if the series is zoomed in or out.

A length measurement can be moved by clicking on it and dragging. To remove a length measurement, drag it or an endpoint outside of the extent of the between instance. Note that if you have "zoomed in" on an instance, its extent may be much larger than the (sub)viewport in which it is displayed. This can make it difficult to drag the measure outside of the extent of the instance. In this case it may be necessary to "zoom out" in order to be able to drag the measure outside of the extent of the instance.

A length measurement is only visible on the instance on which it was made. There is currently no support for saving length measurements.

- The **Angle Measurement** button is modal. When selected, the angle between features in an instance can be measured. To perform a measurement, click on a point of interest in an instance. A pair of lines are displayed. Drag the end points of the lines as needed to form the angle to be measured. The angle between the lines is displayed continuously as any endpoint is dragged.

  An angle measurement can be moved by clicking on one of the lines and dragging it while holding down the mouse button. To remove an angle measurement, drag it or an endpoint outside of the extent of the instance. Note that if you have "zoomed in" on an instance, its extent may be much larger than the (sub)viewport in which it is displayed. This can make it difficult to drag the measure outside of the extent of the instance. In this case it may be necessary to "zoom out" in order to be able to drag the measure outside of the extent of the instance.

  An angle measurement is only visible on the instance on which it was made. There is currently no support for saving angle measurements.

- The **Pixel Probe** button is modal. When selected, clicking on a point in an instance displays a circle at the probe point, the X and Y location of the pixel relative to the top left corner of the instance, and the intensity or color of the selected pixel. The value of color instance pixels is specified in RGB coordinates. For monochrome instances, both a Stored Pixel value (SP) and a Modality Pixel value (MO) are displayed. The MO values is calculated as `SP * RescaleSlope + RescaleIntercept`, where RescaleSlope and RescaleIntercept are DICOM values of the instance.

  A pixel probe can be moved by clicking on the probe indicator and dragging it while holding down the mouse button. To remove a pixel probe, drag it outside of the extent of the instance. Note that if you have "zoomed in" on an instance, its extent may be much larger than the (sub)viewport in which it is displayed. This can make it difficult to drag the measure outside of the extent of the instance. In this case it may be necessary to "zoom out" in order to be able to drag the measure outside of the extent of the instance.

  A pixel probe is only visible on the instance on which it was made. There is currently no support for saving pixel probes.

- The **Elliptical ROI** button is modal. When selected, click on an instance and drag either of the small circles to configure an elliptical region of interest. The area, in pixels, of the ellipse is displayed near the ellipse. On monotone instances, the mean and standard deviation of the intensities of the pixels within the ellipse are also displayed.

  An ellipse can be moved by clicking on its border and dragging it while holding down the mouse button. To remove an elliptical ROI, drag the ellipse or one of its control points outside of the extent of the instance. Note that if you have "zoomed in" on an instance, its extent may be much larger than the (sub)viewport in which it is displayed. This can make it difficult to drag the ROI outside of the extent of the instance. In this case it may be necessary to "zoom out" in order to be able to drag the ROI outside of the extent of the instance.

  An elliptical ROI is only visible on the instance on which it was made. There is currently no support for saving elliptical ROIs.

- The **Rectangle ROI** button is modal. When selected, click on an instance and drag either of the small circles to configure a rectangular region of interest. The area, in pixels, of the rectangle is displayed near the rectangle. On monotone instances, the mean and standard deviation of the intensities of the pixels within the rectangle are also displayed.

A rectangle can be moved by clicking on its border and dragging it while holding down the mouse button. To remove a rectangular ROI, drag the rectange or one of its control points outside of the extent of the instance. Note that if you have "zoomed in" on an instance, its extent may be much larger than the (sub)viewport in which it is displayed. This can make it difficult to drag the ROI outside of the extent of the instance. In this case it may be necessary to "zoom out" in order to be able to drag the ROI outside of the extent of the instance.

A rectangular ROI is only visible on the instance on which it was made. There is currently no support for saving rectangular ROIs.

- The **Rotate Left** button immediately performs a ninety degree left rotation of the image in the (sub)viewport that has focus.
- The **Rotate Right** button immediately performs a ninety degree right rotation of the image in the (sub)viewport that has focus.
- The **Flip Horizontally** button immediately performs a flip about the Y axis of the image in the (sub)viewport that has focus.
- The **Flip Vertically** button immediately performs a flip about the X axis of the image in the (sub)viewport that has focus.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

13.13 Registering a Google Cloud Project

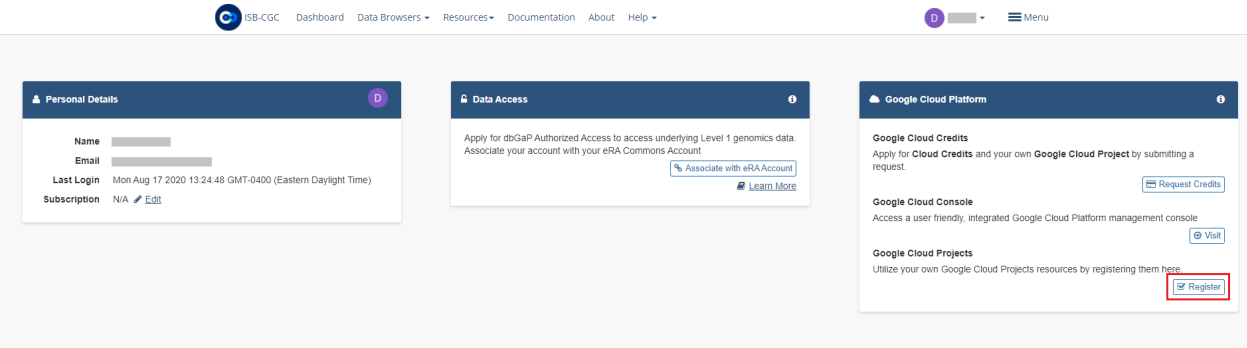
This section will show you how to register a Google Cloud Project (GCP), which you can use to store data from ISB-CGC. Users need to have access to a Google Cloud Project to perform the steps in this section. If you don't, see the the ISB-CGC Quick-Start Guide.

To allow flexibility while working with different research teams and different processes, you can have many GCPs registered with ISB-CGC.

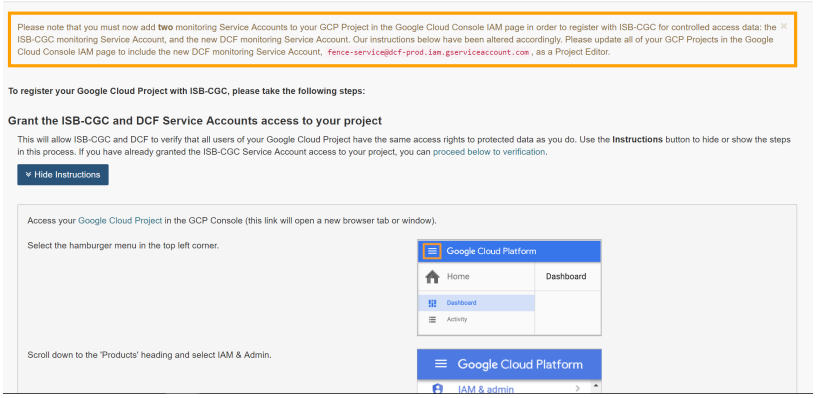
13.13.1 Registering your Google Cloud Project

Click on screen shots to enlarge them.

To register your Google Cloud Projectwith ISB-CGC, go to the Account Details page. After signing into the ISB-CGC Web App, either select the "persona" icon next to your login name or select **Account Details** from the drop down menu under your login name, which takes you to the following page:

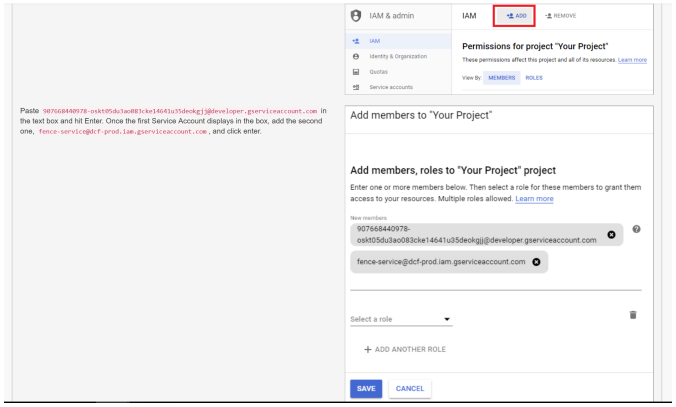


Click the **Register** button in the Google Cloud Platform section. That takes you to the following page:

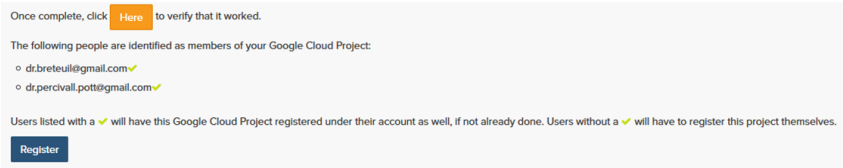


The instructions will walk you through how to add the necessary ISB-CGC and DCF service accounts to your project. Go to the Google Cloud Platform and follow these steps. You can hide the instructions by selecting the blue **Instructions** button.

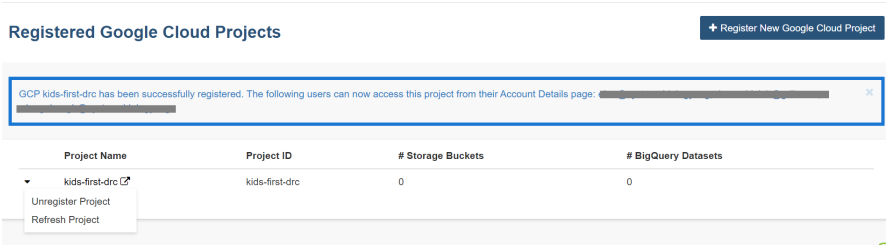
Please be sure to add both service accounts listed below. If you don't add both service accounts you will run into issues. Then return to the ISB-CGC Register a Google Cloud Project page, enter your Google Cloud Project ID and, click **Verify**.



Once you have completed these steps, a listing of the Google Cloud Project members will display:



Click the **Register** button to go to the next screen:

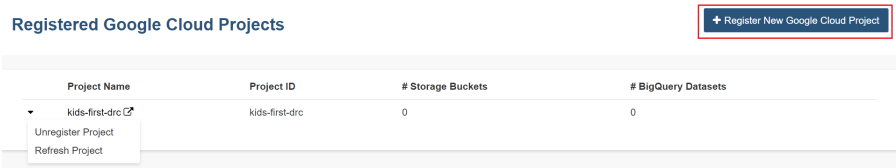


13.13.2 Managing your Google Cloud Projects

You can add or delete Google Cloud Projects by following the instructions below.

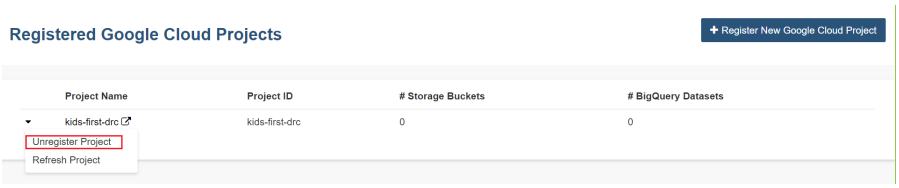
Adding additional Google Cloud Projects

To register additional Google Cloud Projects, select the **+ Register New Google Cloud Project** button from the "Registered Google Cloud Projects" page (see screenshot below).



Deleting Google Cloud Projects

To unregister a GCP, select the **Unregister Project** button from the drop down menu beside the project on the "Registered Google Cloud Projects" page (see screenshot below).



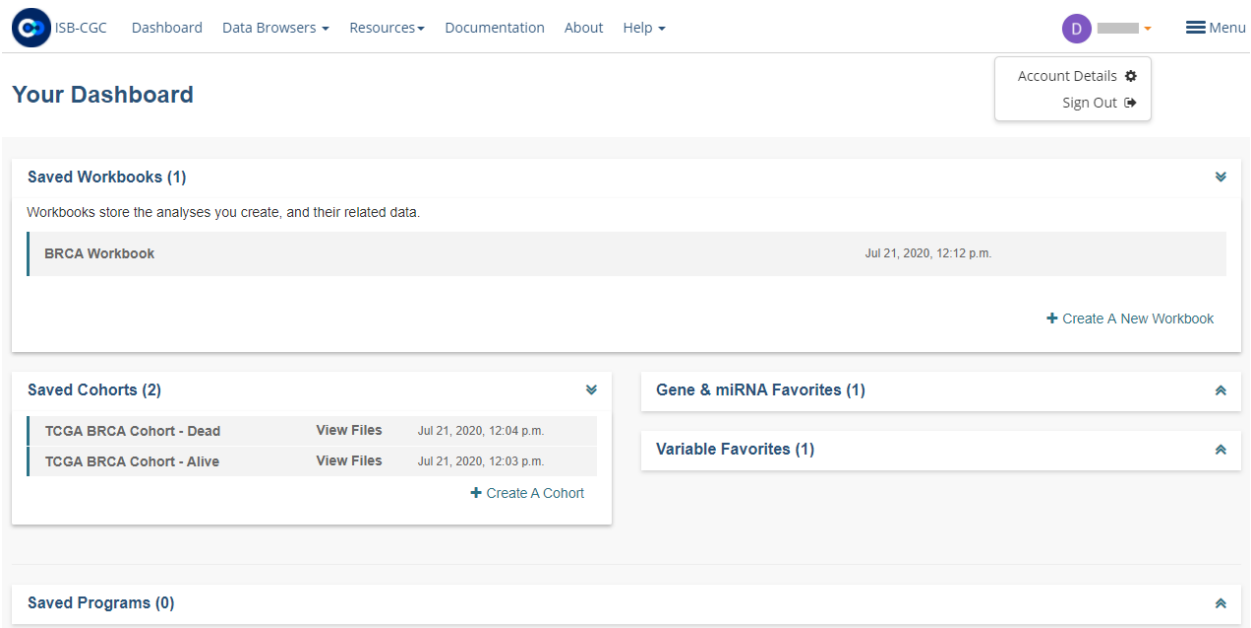
13.13.3 Registering Cloud Storage Buckets and BigQuery Datasets

Registering a Google Cloud Storage Bucket and a BigQuery Dataset is a prerequisite for storing data downloaded from the Web App to your own Google Cloud location. (Please note: The names of the buckets and data sets are case sensitive.)

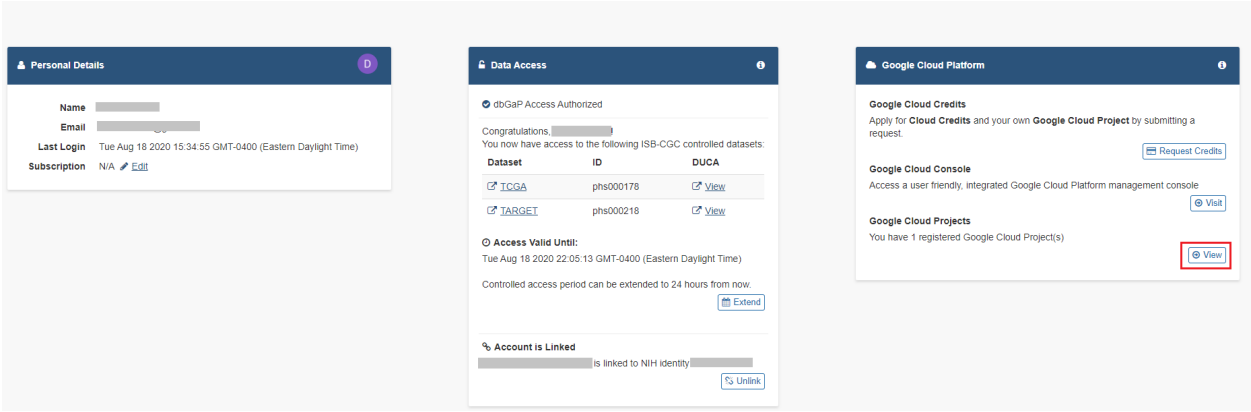
How To Register Buckets and Datasets

Once you have created a bucket and a dataset in the Google Cloud Console of your Google Cloud Project, you will need to register them with your project name using the Web App.

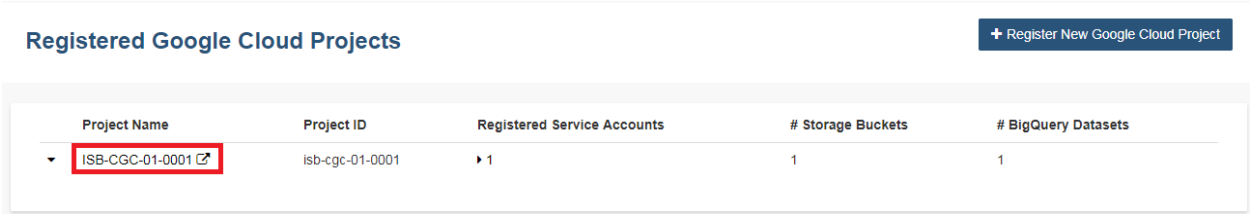
Step 1: Click on your user icon in the upper right or **Account Details** from the drop down menu under your name.



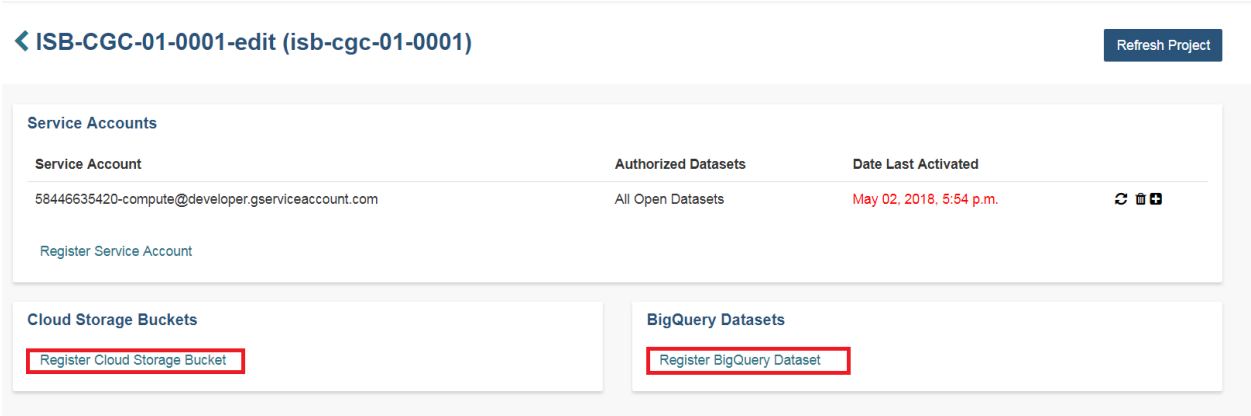
Step 2: Click on the **View** button under **Google Cloud Projects**.



Step 3: Click on the project you wish to use. If you have not registered a project, follow the instructions above.



Step 4: Use the "Register Cloud Storage Bucket" or "Register BigQuery Dataset" links to add buckets and datasets as needed.



Have feedback or corrections? Please email us at feedback@isb-cgc.org.

Data used by the Web App

The Web App performs its data retrieval and counts on ISB-CGC Google BigQuery tables which are based on the latest GDC data release. This means that you will see current data, but that the same queries in the Web App could produce different results if they were run during different time periods, when the Web App was based on different GDC data releases.

Sharing Cohorts between the Web App and the API

Cohorts are one of the central concepts used when analyzing large datasets. Cohorts can be created either in the Web App or via the ISB-CGC REST API. What may not be as clear is that cohorts created by one of the systems can be viewed and used in the other. In other words, you can create a cohort using the API and use it in the Web App or you can create a cohort in the Web App and use it in the API. This can give users significant flexibility in creating and sharing their cohorts.

Choosing a Web Browser

The Web App was optimized for use with the Google Chrome web browser. Most of the functionality should work with recent versions of other web browsers (e.g. Firefox, Safari, Internet Explorer). If you find an issue and you are not using Chrome, please try using Chrome to see if the issue appears to be browser specific.

Web App Time Zone

Also please note the system is set in Pacific time, so if you see some inconsistencies with the time in the workbooks or cohorts you generated, it could be due to this.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CHAPTER

FOURTEEN

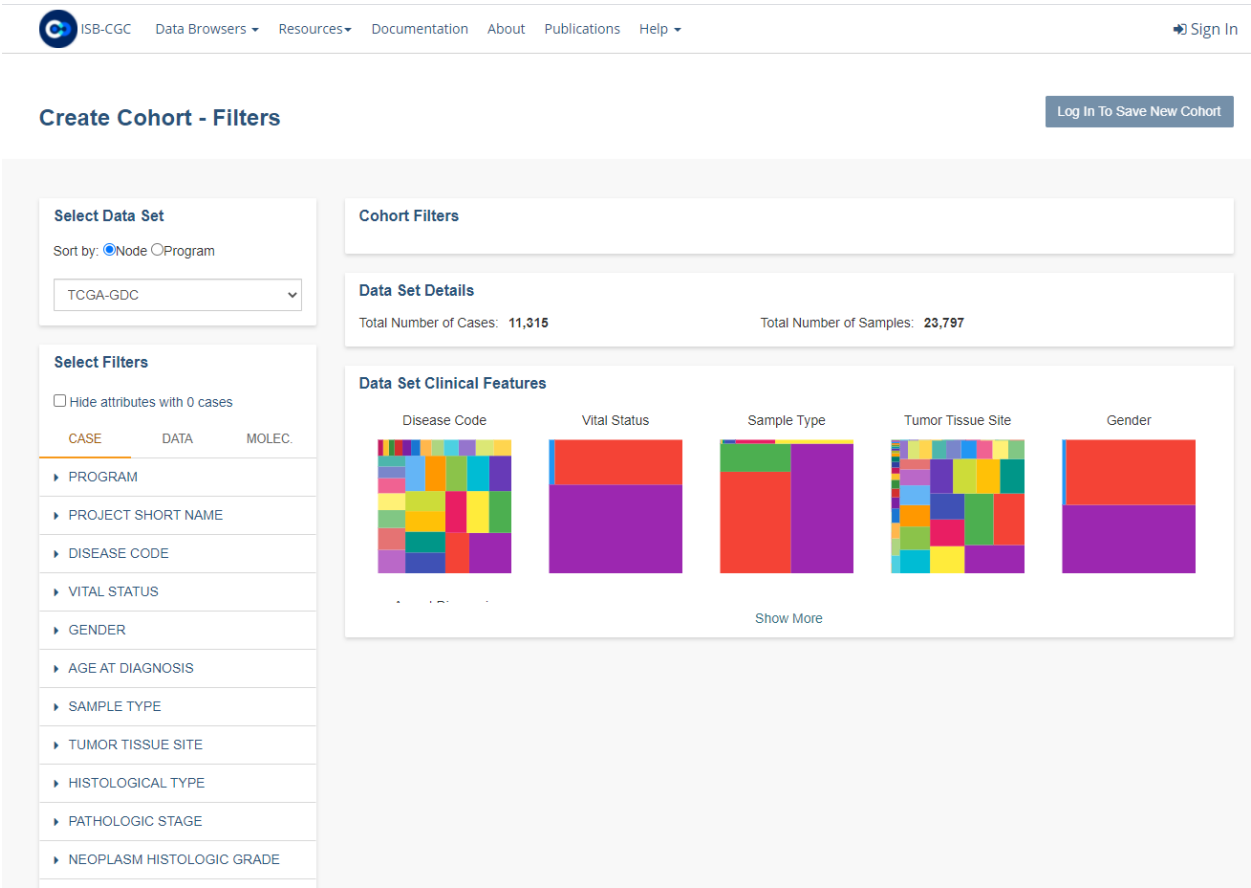
COHORT BUILDER/DATA EXPLORER

Cohorts are a way of creating custom groupings of the samples and/or cases that you are interested in analyzing further. The Cohort Builder/Data Explorer is an ISB-CGC web interface which allows you to build cohorts based on clinical demographics and molecular filters. Compare patient cohorts with various exploration tools including IGV viewer, image viewers, and analytical visualization.

Selecting **Cohort Builder/Data Explorer** from the **Resources** drop down menu on the ISB-CGC home screen will display the **Create Cohorts - Filters** screen. Another way to get to this screen is to click on the **Launch** icon in the **Cohort Builder/Data Explorer** box in the **Resources** section of the ISB-CGC home page.

You will be able to use the available filters to create a cohort, without needing to log into the ISB-CGC Web Application. Except for the ability to save cohorts, this screen has the same functionality as the one that you navigate to when selecting the **COHORTS - Create a New Cohort - Filters** option after signing into the Web App. To learn more about this screen, see the Cohorts documentation.

You may want to frequently reuse a cohort in multiple analyses. Creating a "saved cohort" allows you to do this. If this is the case, click on the **Login to Save New Cohort** button.



Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CHAPTER

FIFTEEN

CANCER DATA FILE BROWSER

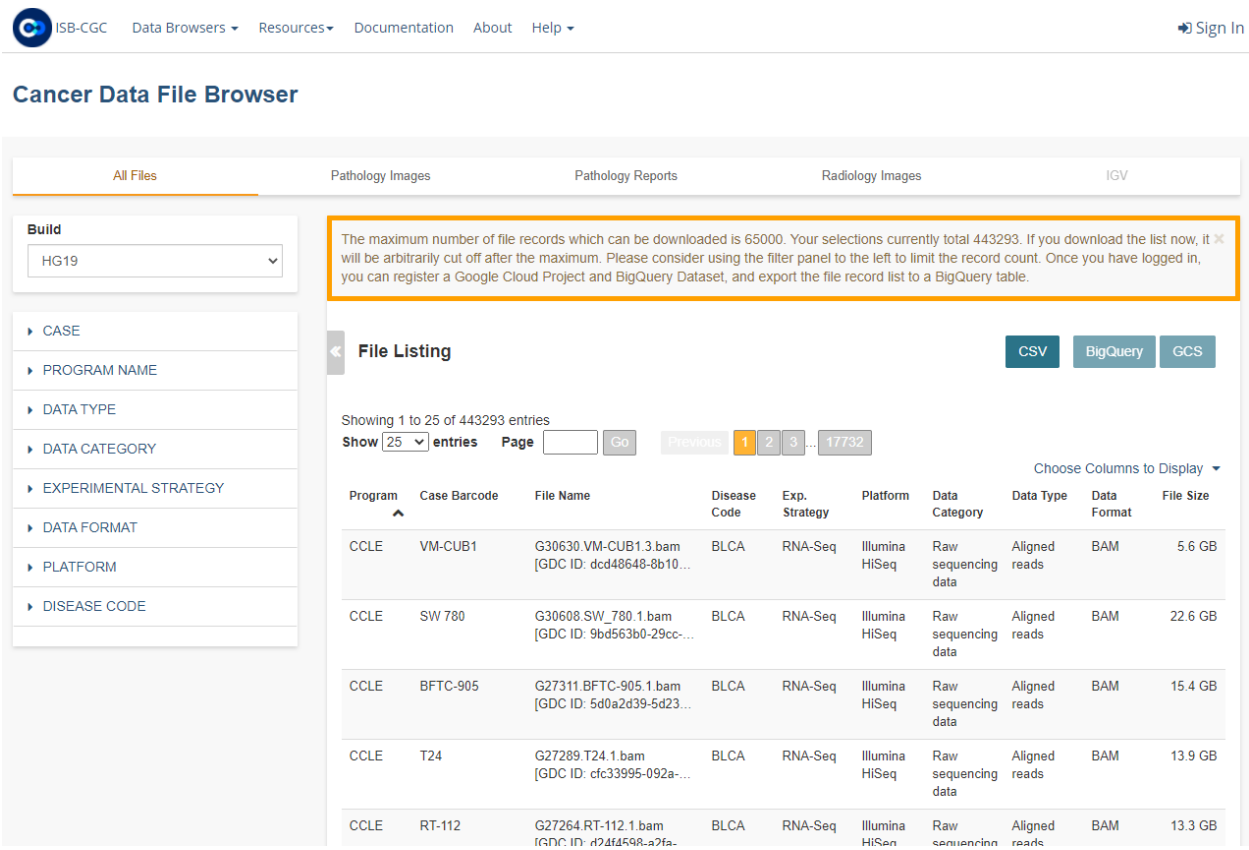
The Cancer Data File Browser is an ISB-CGC web interface which allows you to explore a comprehensive selection of cancer related data files in Google Cloud Storage Buckets, such as raw sequencing, cancer nucleotide variation, pathology or radiology images.

Selecting **Cancer Data File Browser** from the **Data Browsers** drop down menu on the ISB-CGC home screen will display the **Cancer Data File Browser** screen. Another way to get to this screen is to click on the **Launch** icon in the **Cancer Data File Browser** box in the **Data Browsers** section of the ISB-CGC home page.

You will be able to use the available filters to select a file record list. Click on the CSV button to download this list which includes barcodes and GCS locations, without needing to log into the ISB-CGC Web Application. Except for the ability to save output results to a Google BigQuery table or to a Google Cloud Storage Bucket (GCS), this screen has the same functionality as the one that you navigate to when selecting the **File Browser** button from the **Saved Cohorts** screen, after signing into the Web App. To learn more about this screen, see the Cohorts File Browser documentation.

Note that the maximum number of file records that can be downloaded is 65000. You'll need to use the filters to get the file listing results below this number.

If you decide to log into the ISB-CGC Web App (using **Sign In** in the upper right-hand corner), you can register a Google Cloud Project and BigQuery data set and export the file record list to a BigQuery table or a Google Cloud Storage Bucket.



Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CHAPTER

SIXTEEN

MITELMAN DATABASE

The Mitelman Database of Chromosome Aberrations and Gene Fusions in Cancer is devoted to genes, chromosomes, and cancer. The Mitelman Database is supported by NCI (National Cancer Institute), the Swedish Cancer Society and the Swedish Childhood Cancer. The Mitelman Database is available from ISB-CGC to access and search the data.

The information in the Mitelman Database relates cytogenetic changes and their genomic consequences, in particular gene fusions, to tumor characteristics, based either on individual cases or associations. All the data have been manually culled from the literature by Felix Mitelman in collaboration with Bertil Johansson and Fredrik Mertens. The database is updated quarterly in January, April, July, and October.

16.1 Using the Mitelman Database

It can be accessed from the ISB-CGC homepage (https://isb-cgc.org/) by clicking on **Launch** in the **Chromosomal Aberrations & Gene Fusions DB** box or selecting **Chromosomal Aberrations & Gene Fusions DB** from the **Data Browsers** drop down menu on the main menu bar. It can also be accessed directly from https://mitelmandatabase.isb-cgc.org.

The user queries the database by parameters such as topography, morphology, gene characteristics, cytogenetic aberrations, and journal references. There are five searchers available:

- Cases Cytogenetics Searcher
  - allows you to query the individual patient cases using fields such as the aberration, breakpoint, morphology, and topography
- Gene Fusions Searcher
  - finds studies pertaining to gene rearrangements, in particular gene fusions, detected either as a consequence of cytogenetic aberrations or identified by sequencing
- Clinical Associations Searcher
  - searches studies pertaining to clinical associations of cytogenetic aberrations and/or gene rearrangements.
- Recurrent Chromosome Aberrations Searcher
  - provides a way to search for structural and numerical abnormalities that are recurrent, i.e., present in two or more cases with the same morphology and topography
- References Searcher
  - queries only the references themselves, i.e., the references from the individual cases and the molecular biology and clinical associations

Until June 2022, the resulting genetic location information retrieved from the database was only displayed in karyotypes. Now, genomic coordinates are also displayed. Thanks to procedures incorporated from the web-based tool CytoConverter, karyotypes are converted to genomic coordinates and can be optionally viewed by the Mitelman Database user.

The user has the option of viewing the genomic coordinate information for either individual karyotypes or for multiple karyotypes in a search result. For individual karyotypes, the corresponding chromosome and its start and end position are given. In addition, the type of imbalance (gain or loss) is noted. For multiple karyotypes in the search results, net imbalances across the selected group are displayed in chart, ideogram or tabular format; information includes the chromosome affected, start and end positions, and whether the segment has been lost or gained.

16.2 How to Cite

To cite the use of the Mitelman Database, authors should cite the following source:

> Mitelman Database of Chromosome Aberrations and Gene Fusions in Cancer (2022). Mitelman F, Johansson B and Mertens F (Eds.), https://mitelmandatabase.isb-cgc.org

In addition, when using information about chromosomal gains and losses (found on the Karyotype Info and Overall Chromosomal Imbalances pages), please also cite the following:

> CytoConverter: a web-based tool to convert karyotypes to genomic coordinates. Wang, J., LaFramboise, T. BMC Bioinformatics 20, 467 (2019). https://doi.org/10.1186/s12859-019-3062-4

16.3 More Information

For more information, please see the Mitelman Database About page and User Guide.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CHAPTER

SEVENTEEN

THE *TP53* DATABASE

The *TP53* Database (https://tp53.isb-cgc.org) compiles *TP53* variant data that have been reported in the published literature since 1989 or are available in other public databases. Database releases are identified by a number. The following data are available:

- *TP53* functional and structural data including validated polymorphisms
- *TP53* tumor variants in sporadic cancers
- *TP53* germline variants in cancer patients, families with cancers and control populations
- *TP53* gene status in human cell-lines
- Mouse models with engineered p53
- Experimentally-induced variants of *TP53*

The *TP53* Database is meant to be a source of information on *TP53* variants for a broad range of scientists and clinicians who work in different research areas:

- **Basic research**, to study the structural and functional aspects of the p53 protein and the *TP53* gene
- **Molecular pathology of cancer**, to understand the clinical significance of *TP53* variants identified in cancer patients
- **Molecular epidemiology of cancer**, to analyze the links between specific exposures and *TP53* variant patterns in order to make inferences about possible causes of cancer
- **Molecular genetics**, to analyze genotype/phenotype relationships

The database includes various annotations on the predicted or experimentally assessed functional impact of *TP53* variants, clinicopathologic characteristics of tumors and demographic and life-style information on patients. This information is useful to compile tumor-specific variant patterns and to draw hypotheses on the nature of the molecular events involved in *TP53* mutagenesis and allows for the analysis of genotype/phenotype relationships.

Detailed information on data and annotations available is provided in the User Manual.

The ongoing project involves:

- Performing regular review of the literature on *TP53* variants
- Extracting *TP53* data from genetic and genomic databases
- Developing standard annotations of *TP53* variants
- Performing research on *TP53* variants, their patterns, origins and clinical impacts.

17.1 How to Cite

When using the database, authors should cite the following source:

> The *TP53* Database (R20, July 2019): https://tp53.isb-cgc.org

and refer to de Andrade K.C. et al. in the bibliography as below:

> de Andrade, K.C., Lee, E.E., Tookmanian, E.M. et al. The *TP53* Database: transition from the International Agency for Research on Cancer to the US National Cancer Institute. *Cell Death Differ* (2022). https://doi.org/10.1038/s41418-022-00976-3

17.2 More Information

For information regarding releases, credits and disclaimers, please see the *TP53* About page.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CHAPTER

EIGHTEEN

THE SYNTHETIC LETHALITY RESOURCE

Synthetic lethal interactions (SLIs), genetic interactions in which the simultaneous inactivation of two genes leads to a lethal phenotype, are promising targets for therapeutic intervention in cancer, as exemplified by the recent success of PARP inhibitors in treating BRCA1/2-deficient tumors. We present SL-Cloud, a cloud-based integrated resource to facilitate the prediction of context-specific SLIs. This resource addresses two main challenges related to SLI inference: the need to wrangle and preprocess large multi-omic datasets and the multiple comparable prediction approaches available.

SL-Cloud provides a cloud-based data access platform coupled with software and well documented computational notebooks that reimplement published synthetic lethality (SL) inference algorithms to facilitate novel investigation into synthetic lethality. In addition, we provide general purpose functions that support these prediction workflows, e.g. saving data in BigQuery tables. We anticipate that users can leverage the resources provided in this project to conduct highly customizable analysis based on their cancer type of interest and particular context.

More information about SL-Cloud can be found in the following links:

- Notebooks in MyBinder
- GitHub Repository
- Article describing SL-cloud

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CHAPTER

NINETEEN

PIPELINES AND APIS

19.1 APIs

ISB-CGC provides programmatic access to cancer data (both open and controlled-access) and metadata stored on the Google Cloud Platform through a combination of ISB-CGC APIs and Google APIs. Access to ISB-CGC metadata and user-data such as patient cohort definitions is provided through the ISB-CGC API. For more details on Google Cloud APIs, please read the extensive in-depth Google Cloud APIs documentation.

19.1.1 ISB-CGC APIs

ISB-CGC provides a Swagger API, which interacts with the ISB-CGC Web App data, including user-generated data such as cohorts. We provide API calls pertaining to specific samples, cases, files, cohorts, and users. The syntax for all of these is available on the ISB-CGC API v4.0 UI webpage.

The ISB-CGC APIs can also be used via Python and R. We have tutorial notebooks available in our Community Notebook Repository.

- Python Notebook
- R Notebook

Some example uses of the ISB-CGC API are:

- Obtaining detailed metadata about a particular patient or sample
- Creating (or retrieving a previously saved) cohort of patients and samples
- Retrieving a cohort's file manifest using the cohort ID or specific filters
- Register, refresh, and unregister a specified Google Cloud Project (GCP)

Note: APIs calling user-generated data (such as your cohorts) require identity credentials.

Authorization

Some of the APIs - such as for programs, samples, and cases - can be accessed without authorization. APIs that call on information saved in a user's account, such as the cohorts and GCP APIs, require account authorization.

In order to access the APIs that require ISB-CGC authorization, you will need to generate a credentials file on your local machine or on your VM. To load your credentials into your command line interface:

1. Clone the ISB-CGC-API scripts GitHub repository to your local machine.
2. Run the isb_auth.py script either through the command line or within Python.
3. If you are running the ISB-CGC APIs on a VM, upload the file generated by the above process.

ISB-CGC API v4.0 UI

The ISB-CGC API v4.0 UI displays details about the syntax for each call and also provides an interface to test requests.

To generate a subset of ISB-CGC hosted data with your desired characteristics, we have provided tools to generate cohorts of patients. In addition to the BigQuery command line, users may create and share cohorts using the ISB-CGC Web App and then access them using the Swagger UI API.

Make a Request

As mentioned before, some of the API calls will require authentication - denoted by a small lock symbol. This can be done by using the 'Authorize' button at the top right of the page.

For a quick demonstration of the syntax of an API call, one can test the POST/samples request. This API request has the following syntax:

```
{
 "barcodes": [
 <barcode 1>,
 <barcode 2>,
 ...,
 <barcode n>,
 ]
}
```

TCGA samples are easily selected by using the 16-character barcode, i.e. TCGA-B9-7268-01A, while patients are identified using the 12-character prefix of the sample barcode, in this case TCGA-B9-7268. Other data sets such as CCLE may use other naming conventions.

The value in the Parameters field can be edited by selecting 'Try it out'. One can change the default sample barcodes or leave them. The request can be run by selecting 'Execute'.

Request Response

Swagger UI submits the request and shows the curl code that was submitted. The 'Response body' section will display the response to the request. The expected format of the response for the above request is shown below:

```
{
 "data": [
 {
  "samples": [
    {
         "data_details": [
           {
             <key 1>: <value 1>,
             <key 2>: <value 2>,
             ...,
             <key n>: <value n>,
           }
         ],
         "biospecimen_data": {
           <key 1>: <value 1>,
           <key 2>: <value 2>,
           ...,
           <key n>: <value n>,
         },
         "sample_barcode": "string",
         "case_barcode": "string"
       }
     ]
   }
 ],
 "code": 0,
 "barcodes_not_found": [
   "string"
 ],
 "total_found": 0,
 "notes": "string"
}
```

The JSON formatted response can be downloaded by selecting the 'Download' button.

Warning:

- Any special characters in the input field will cause the request to fail. e.g. spacing in input box.
- Please make sure to delete all fields not being used.
- Case barcode centric requests only pull file paths specific to case entries.
- Sample centric requests pull file paths specific to sample entries.
- Cohorts made using the Web App will differ in sample counts from cohorts made using BigQuery tables. The Web App takes into consideration samples which correspond to pathology slide images and this information is currently not in the BigQuery tables.

For any questions or feedback on the API, please do not hesitate to contact us at feedback@isb-cgc.org.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

19.2 Workflow Gallery

We have compiled a collection of tutorials and sample workflows designed to introduce users to running workflows (CWL, Nextflow, Snakemake, WDL, and GeneFlow) on the Google Cloud Platform (GCP). ISB-CGC does not make the choice for users but instead enables as many workflow technologies as possible through documentation, support, and where necessary, infrastructure.

19.2.1 Running Workflows on ISB-CGC

The ISB-CGC platform is intentionally designed to be a light-weight infrastructure above the Google Cloud Platform (GCP). As a result, our end-users have at their disposal all of the tools and technologies that the GCP has to offer. This includes full **sudo** acccess to Google Cloud Compute Engines and Virtual Machines. For more detailed information about GCP tools and technologies, please see their in-depth documentation page.

We have put together some basic documentation and workflow examples for ISB-CGC users who are both new to the GCP as well as to the commonly used workflow languages in -omics research. We have chosen the workflow languages of CWL, NextFlow, Snakemake, WDL, and GeneFlow. If there are other workflow languages you are interested in, please let us know and we'll put together some examples for them as well (email us at feedback@isb-cgc.org).

Getting Started

The links in this section: **(1)** help new users get familiar with the Google Cloud Platform tools and technologies (virtual machines and cloud storage) necessary to run workflows with your ISB-CGC Google Cloud Platform project, **(2)** provide a cheatsheet of the packages and dependencies required on the virtual machines to successfully execute the workflow examples provided below. **(3)** Calculating cloud costs before running large workflows is very crucial. We provide here some tips and tricks to determining costs before running workflows.

- Launching a Virtual Machine
- Creating a Google Cloud Storage Bucket
- VM Workflow Tools Installation Cheatsheet
- Workflow Cost Considerations

Launching a Google Cloud Virtual Machine (VM)

You can launch a virtual machine (which we will generally refer to as a VM) from the web-based Google Cloud Console or from the command line interface (CLI) using the Google Cloud SDK. We will describe both of these approaches below.

Prerequisites

1. Enable the following APIs in your Google Cloud Project:
   - Google Compute Engine
   - Google Cloud Storage
2. Check the IAM & Admin section of the Cloud Console to verify that you have "Editor" or "Owner" privileges in your Google Cloud Project, otherwise you won't be able to create any VMs.

Via the Google Cloud Console

An instructional video on how to launch a VM can be found on the Google Cloud Platform youtube page: How to launch a VM.

Try it yourself: On your Google Cloud Platform project console page:

- **Choose the Compute Engine option from the menu icon in the upper-left corner.**
- Choose the VM instances page.
  1. Note: the first time you visit the page, you will see two options: "Create Instance" or "Take the quickstart".) After that, you will see a page with a list of existing (running or stopped) VMs.
- Select the Create Instance option, and customize your instance preferences:
  1. **Name:** this name is relatively arbitrary, choose something that is meaningful to you;
  2. **Zone**: choose one of the us-east or us-central zones;
  3. **Machine type**: you can specify a VM with anywhere between 1 and 16 cores (aka vCPUs), and with up to 100 GB of RAM (you can try the "Customize" view if you prefer a more graphical approach); note that as you change the specifications of the VM, the estimated cost shown on this page will update;
  4. **Boot disk**: the default boot disk and OS will be shown, but you can change this as you wish: the "Change" button will result in a flyout panel where you can choose from a variety of Preconfigured images (Debian, CentOS, Ubuntu, RedHat, etc) or previously created images or disks; you can also choose between "standard disks" and faster (and more expensive) solid-state drives (SSDs), and specify the size of the disk (up to 64TB).
- Once you have all of the options set, you can click on the blue **Create** button.
  1. Creating the VM should take less than a minute, after which you will see it listed on the "VM instances" page, with the Name, Zone, Disk, Network, and External IP address shown. There is also an SSH button that you can use directly from the Console.

Launch a VM using the Command Line Interface (CLI)

The command line argument to create a new VM instance is **gcloud compute instances create**. The complete documentation can be found online or by typing **gcloud compute instances create –help** on the command line.

Here is a very simple command to create a VM:

gcloud compute instances create my-instance –machine-type g1-small

Some defaults can be obtained (if available) from your configuration settings. For example, if you don't want to have to specify the zone of the instances, you can set the compute/zone property, for example:

gcloud config set compute/zone us-central1-a

A list of zones can be fetched by running:

gcloud compute zones list

Accessing your new VM

Whether you have created your VM from the Console or using the CLI, you can find it and ssh to it, again using either the Console or the CLI:

- From the Console, go to Compute Engine > VM instances, and then click on the **SSH** button on the far-right of the row describing the specific VM you would like to connect to.
- Using the CLI, simply use the command **gcloud compute ssh** followed by the instance name.

Shutting down your VM

Remember that as long as your VM is running, whether or not *you* are actually doing anything with it, charges will be incurred. It is therefore a good idea to get in the habit of shutting down VMs as soon as you are finished with your work. They can easily be restarted an hour, day, or week later. Note that resources that are *attached* to a stopped VM (such as persistent disks) *will*, however continue to incur charges. Compared to the cost of the VM, though, the cost of a persistent disk is typically negligible: a 50 GB standard persistent disk only costs $2 per month, and 1 TB costs $40.

If you know that you won't ever need this specific VM again, or you don't want to continue paying for the persistent disk, or you would rather start a fresh VM with an updated OS next time, then you can go ahead and **delete** the VM rather than just stopping it.

From the command-line, the relevant commands are **gcloud compute instances stop** and **gcloud compute instances delete**.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

Creating a Google Cloud Storage Bucket

Why should you create Google Cloud Storage buckets (hereto referred to as buckets) when your virtual machine can also store your data? Because Google Cloud Storage buckets are much less expensive to maintain compared to VM disks. Please see this link for the most up-to-date pricing on the different storage options offered by the Google Cloud Platform.

Via the Google Cloud Console

Instructional video provided by the Google Cloud Platform: How to Create a Google Cloud Storage Bucket through the Console.

Try it out yourself:

1) Click on Cloud Storage browser on the left of the page
2) Click **Create a bucket**
3) Give the bucket a unique name, with all lowercase letters and no spaces.
4) Select region, location and click **Create**

Via gsutil commandline tool

Use the gsutil mb command:

```
$gsutil mb gs://[BUCKET_NAME]/
```

Where:

- [BUCKET_NAME] is the name you want to give your bucket, subject to naming requirements. For example, my-bucket.

You can set the following optional flags to have greater control over the creation of your bucket:

- p: Specify the project with which your bucket will be associated. For example, my-project.
- c: Specify the default storage class (https://cloud.google.com/storage/docs/storage-classes)) of your bucket. For example, NEARLINE.
- l: Specify the location (https://cloud.google.com/storage/docs/locations of your bucket. For example, US-EAST1.
- b: Enable uniform bucket-level access (https://cloud.google.com/storage/docs/uniform-bucket-level-access) for your bucket.

Accessing data in your bucket by gsutil

```
my-cloud-bucket
|_______myFile.txt
|_______myOtherFile.txt
```

To list what in your bucket:

```
$gsutil ls gs://my-cloud-bucket/
```

output:

```
gs://my-cloud-bucket/myFile.txt
gs://my-cloud-bucket/myOtherFile.txt
```

Accessing data in your bucket via GCSFuse

If you have access to a bucket and want to "clone" that bucket to your VM instance, gcsfuse can mount that bucket/or a sub-directory of that bucket to your VM directory of your choice.

Pros:

- Data can be accessed without using **gsutil** or **gs://** address, i.e your bucket data become local to your VM instance

Cons:

- Since gcsfuse will actually download the data to your directory, make sure your VM instance has enough available storage to clone data from the bucket
- Consumes available storage space
- May slow down your performance

(1) How-to instructional video

(2) Step-by-Step installation guide

GCSFuse Quicktips

Mount a bucket to your folder:

```
$gcsfuse bucketname myfolder/to/mount
```

Mount a subdirectory from your bucket to your VM folder:

```
$gcsfuse --only-dir subdirectory bucketName myFolder/to/mount
```

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

VM Workflow Tools Installation Cheat Sheet

When working with a new Virtual Machine (VM), installing software, packages, and dependencies is usually required, and the process can be cumbersome. This cheat sheet was created with running workflows on Google Cloud VM in mind. It contains quick shortcuts to install common software, dependencies, and quick fixes.

NEXTFLOW

Install:

```
$ export NXF_VER=20.01.0
$ export NXF_MODE=google
$ curl https://get.nextflow.io | bash
```

SNAKEMAKE

Step 1 install Miniconda:

Installer | Instructions

Note: After "conda init fish" step **restarting your VM command line** is needed. In addition, if the conda command is not found, try: $export PATH=./miniconda3/bin/:$PATH

Step 2 install Snakemake:

Instructions

Installer:

```
$ wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
```

Snakemake environment:

Create and activate Environment for Snakemake from a file (yml/yaml):

```
$ conda env create --name <yourEnvironmentName> --file environment.yaml
$ source activate <yourEnvironmentName>
```

Update current environment:

```
$ conda env update -f environment.yml
```

Note: For more conda commands, visit: Conda Cheat sheet.

WDL

Installers for Cromwell and Womtool

```
$ wget https://github.com/broadinstitute/cromwell/releases/download/52/cromwell-52.jar
$ wget https://github.com/broadinstitute/cromwell/releases/download/52/womtool-52.jar
```

CWL

```
$ sudo apt-get install python-pip
$ pip install --upgrade pip
$ pip install cwltool
```

GENEFLOW

Instructions and Source Code

Install Python3 and Pip (see below), then install GeneFlow in a virtual environment with the following:

```
$ python3 -m venv gf
$ source gf/bin/activate
$ pip3 install geneflow
```

Be sure to always activate the GeneFlow Python virtual environment before using the GeneFlow command line.

Common dependencies

Java

```
$ sudo apt install default-jre
```

Python

```
$ sudo apt-get update
$ sudo apt-get install python3.6
```

Pip install/Python-pip

```
$ sudo apt-get install python-pip
$ pip install --upgrade pip
```

DOCKER

Install:

For Ubuntu:

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
↪$(lsb_release -cs) stable"
$ sudo apt-get update
$ apt-cache policy docker-ce
$ sudo apt-get install -y docker-ce
```

For Debian:

```
$ curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -
$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/debian
↪$(lsb_release -cs) stable"
$ sudo apt-get update
$ apt-cache policy docker-ce
$ sudo apt-get install -y docker-ce
```

Check docker status:

```
$ sudo systemctl status docker
```

Check docker installation:

```
$ docker run hello-world
```

If docker doesn't run, try the fix right below or read more here.

Permission denied

error prompt:

```
docker: Got permission denied while trying to connect to the Docker daemon socket at unix
```

Try:

```
$ sudo groupadd docker
$ sudo usermod -aG docker ${USER}
close VM and reopen
```

DOCKER Daemon not running:

```
$ sudo service docker start
$ sudo dockerd
```

Graphviz

```
$ sudo apt-get install graphviz
```

Git/github

```
$ sudo apt install git
```

Wget

```
$ sudo apt-get install wget
```

Subversion

```
$ sudo add-apt-repository universe
$ sudo apt update
$ sudo apt install subversion
```

GCSFUSE

Mount a bucket to your folder:

```
$ gcsfuse bucketname myfolder/to/mount
```

Mount a subdirectory from your bucket to your VM folder:

```
$ gcsfuse --only-dir subdirectory bucketName myFolder/to/mount
```

Set PATH for executable file

```
$ export PATH=~/where/you/install/theProgram:$PATH
```

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

Workflow Costs

Before running your workflow, it's important to consider the factors involved in the cost. Below are some ways to watch out for costs when working on the Google Cloud.

- Google Compute Engine charges are based on a pricing sheet.
  - Check this periodically as pricing is routinely updated.
- The pricing is totaled up from:
  - Disk size
  - Machine type memory
  - Network Usage
  - Price depends on the configuration
- Google provides an on site preview of costs and discounts per machine type:
  - Google Platform Pricing Calculator
- For more cost-saving tips and tricks, check out our Best Practices documentation.
- If you're looking to run large jobs, contact us (feedback@isb-cgc.org) and we'll help optimize your cloud costs.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

Example introductory bioinformatics workflows

We have put together these examples to **(1)** demonstrate the syntax of these workflow languages **(2)** illustrate them with two commonly used bioinformatics and -omics use-cases (RNAseq and Blast) **(3)** allow users to adapt these examples to their own use-cases.

- Snakemake RNA-seq
- CWL RNA-seq
- Nextflow RNA-seq
- GeneFlow RNA-seq
- Nextflow Blast
- CWL Blast

Your First Workflow on Google Cloud Virtual Machine (Snakemake RNA-seq)

This Snakemake RNA-seq workflow maps read-pairs to a reference genome and quantify the produce to produce a transcript. Snakemake workflows are Python based, extended by declarative code which defines rules. The rules indicate how the output files are created from the input files.

Requirements:

- Anaconda/Miniconda

Note: The requirements above are crucial to running this workflow. Please make sure you have them installed properly prior to running this workflow.

Download this tutorial:

```
$ sudo add-apt-repository universe
$ sudo apt update
$ sudo apt install subversion

#cloning this tutorial
$ svn checkout https://github.com/isb-cgc/RunningWorkflows-on-the-GoogleCloud/trunk/
↪Snakemake-RNAseq
```

Starting folder:

```
Snakemake-RNAseq
|_________data
|           |_________ggal_gut_1.fq
|           |_________ggal_gut_2.fq
|           |_________ggal.gtf
|           |_________reference.fa
|_________step1
|           |_________indexing.smk
|           |_________Snakefile
|_________step2
|           |_________therest.smk
|           |_________Snakefile
|_________environment.yml
```

Setting up your Conda environment using the environment.yml file

The file **environment.yml** contains a list of all the softwares required to run this tutorial

```
$ cd Snakemake-RNAseq
$ conda env create --name tutorial --file environment.yml
$ conda activate tutorial
```

Note: Syntax: conda env create –name <yourEnvironmentName> –file environment.yml. For this tutorial <yourEnvironmentName> = tutorial

It should look like this:

```
(base) thinh_vo@kv-test-snakemake:~/rnaseq-snake$ conda activate tutorial
(tutorial) thinh_vo@kv-test-snakemake:~/rnaseq-snake$
```

Congratulations! You are now ready to run your first workflow using snakemake.

Step 1: Creating Index files from reference genome using hisat2-build

From folder rnaseq-snake:

```
$ cd step1
```

Let's take a look at the 2 files in this folder: **Snakefile** and **indexing.smk**.

```
|_________step1
           |_________indexing.smk
           |_________Snakefile
```

Snakefile is the default name for snakemake's script. The "snakemake" command will automatically find a file name Snakefile and execute it; **indexing.smk** is a helper file that will be called within the Snakefile.

Let's run it by using:

```
$ snakemake
```

Snakemake will run and use hisat2 to build index files from file rnaseq-snake/data/**reference.fa**. All the index files will be moved to rnaseq-snake/step1/**reference**.

Let's take a quick look at the code:

```
#Snakefile
rule targets:
  input:
      "index.done"
include: "indexing.smk"
```

The first rule's input of a Snakefile (rule targets) will determine what's the output of that workflow. In this case, index.done is the target file that snakemake will look for. **Include: "indexing.smk"** will call the following helper script:

Note: Only the very first Snakefile rule's input behaves like this; other input will be the actual inputs of your workflow.

```
#Indexing.smk
rule indexing:
  input:
      "../data/reference.fa"
  output:
      touch('index.done')
  shell:
      """
      mkdir reference
      hisat2-build {input} index
      mv index.* reference/
      """
```

In **Indexing.smk** file we have an actual input "../data/reference.fa" and the output section tells snakemake to create an empty file "index.done", which is the file that the first rule will check to make sure that this helper script actually run. Then the shell script is executed as follows: a folder called reference got created, then Hisat2 created index files from the fasta file, and then all the index files got moved to the reference folder.

After **Step 1**:

```
rnaseq-snakemake
|_________data
|           |_________ggal_gut_1.fq
|           |_________ggal_gut_2.fq
|           |_________ggal.gtf
|           |_________reference.fa
|_________step1
|           |_________indexing.smk
|           |_________Snakefile
|           |_________[index.done]
|           |_________[reference]
|                          |_________[index.1.ht2]
|                          |               [(2-7)]
|                          |_________[index.8.ht2]
|_________step2
|           |_________therest.smk
|           |_________Snakefile
|_________environment.yml
```

Step 2: Creating the BAM file and the Transcript from reads and index files

Step 2 is similar to Step 1.

From folder step1, to run step 2:

```
$ cd ..
$ cd step2
$ snakemake
```

After **Step 2**:

```
rnaseq-snakemake
|_________data
|            |_________ggal_gut_1.fq
|            |_________ggal_gut_2.fq
|            |_________ggal.gtf
|            |_________reference.fa
|_________step1
|            |_________indexing.smk
|            |_________Snakefile
|            |_________index.done
|            |_________reference
|                          |_________index.1.ht2
|                          |              (2-7)
|                          |_________index.8.ht2
|_________step2
|            |_________therest.smk
|            |_________Snakefile
|            |_________[ggal_gut.cutadapt.sam]
|            |_________[e2t.ctab]
|            |_________[i_data.ctab]
|            |_________[i2t.ctab]
|            |_________[t_data.ctab]
|            |_________[e_data.ctab]
|            |_________[ggal_gut.tsv]
|            |_________[ggal_gut_ref.gtf]
|            |_________[ggal_gut_transcript.gtf]
|            |_________[ggal_gut.cutadapt.bam]
|            |_________[ggal_gut.cutadapt.bam.bai]
|_________environment.yml
```

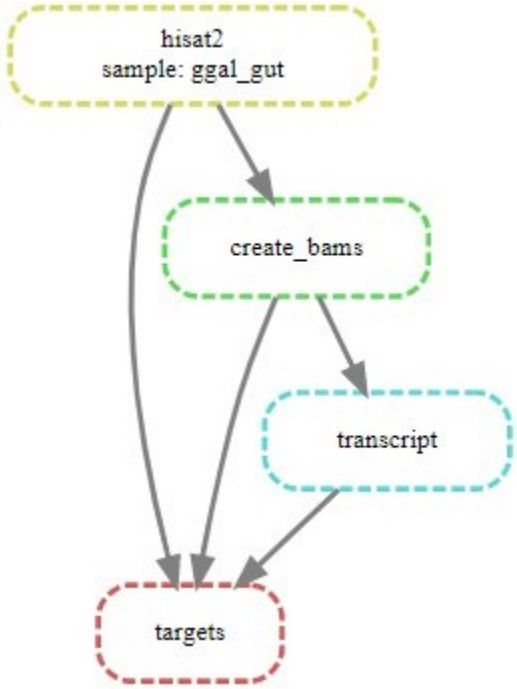
The script will call hisat2 , samtools, and stringtie to do the work. The same principle as step 1 applies to this step, the **Snakefile** will contains the final outputs, and call to the helper script **therest.smk**. **ggal_gut.cutadapt.sam** file will contains the sequence alignment data produced by mapping reads to the reference genome. **ggal_gut.cutadapt.bam** file will contain the compressed binary data from Sam. More description on ctab files, gtf outputs, and tsv of stringtie can be found here. The **ggal_gut_transcript.gtf** contains details of the transcripts that StringTie assembles from RNA-Seq data, while **ggal_gut.tsv** contains gene abundances.

Creating a visualization for your workflow

In the step2 folder:

```
$ snakemake --dag | dot -Tsvg > visual.svg
```

A file named **visual.svg** will be created in the same folder; it can be downloaded and open with any web browser. It should look like this:



About environment.yml

```
channels:
 - conda-forge
 - bioconda
 - main
 - r
dependencies:
 #snakemake and python will be included
 - snakemake-minimal =5.10.0
 - python =3.7.6
 #all other bioinformatics tools
 - samtools =1.9
 - bowtie2 =2.3.5.1
 - hisat2 =2.2.0
 - stringtie =2.1.2
 - gffread =0.11.7
 #visualization tool
 - graphviz =2.42.3e
```

To see the result of this workflow, you can check it here.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

Running CWL RNA-seq

This Common Workflow Language (CWL) RNA-seq workflow maps read-pairs to a reference genome and produces a transcript.

CWL enables the user to connect command line tools to create workflows; it is a specification and is therefore portable across platforms that support CWL.

Requirements:

- CWLtool
- Docker

Note: The requirements above are crucial to running this workflow. Please make sure you have them installed properly prior to running this workflow.

Download this tutorial:

```
$ sudo add-apt-repository universe
$ sudo apt update
$ sudo apt install subversion

#cloning this tutorial
$ svn checkout https://github.com/isb-cgc/RunningWorkflows-on-the-GoogleCloud/trunk/CWL-
↪RNAseq
```

To install Docker and CWL, see our VM Workflow Tools Installation Cheatsheet for instructions.

Starting folder **CWL-RNAseq** should look like this:

```
.
└── CWL-RNAseq
    ├── create_bam.cwl
    ├── create_transcript.cwl
    ├── CWL-RNAseq.cwl
    ├── CWL-RNAseq.yml
    ├── data
    │   ├── sample_1.fq
    │   ├── sample_2.fq
    │   ├── sample.fa
    │   └── sample.gtf
    ├── hisat2_align.cwl
    └── index_build.cwl
```

An overview of the main CWL files:

- **CWL-RNAseq.cwl** is the main cwl file that connects all other cwl tools and yml file together.
- **CWL-RNAseq.yml** is the file that contains all the inputs that are necessary to run the pipeline.
- **index_build.cwl** builds index files from a Fasta file, using Hisat2-build.
- **hisat2_align.cwl** builds a sam file from forward and reverse reads, and the indices built from previous step, using Hisat2.

- **create_bam.cwl** builds a bam file from the newly built sam file, using Samtools.
- **create_transcript.cwl** creates transcript from the bam file from previous step, using Stringtie.

Let's take a look at some example of the main file **CWL-RNAseq.cwl**: The first block describe what is required to run this workflow, more information on this CWL requirements can be found here. Docker usage is also described in the **hints** section.

```
#!/usr/bin/env cwl-runner

cwlVersion: v1.0
class: Workflow
requirements:
  SubworkflowFeatureRequirement: {}
  StepInputExpressionRequirement: {}
  InlineJavascriptRequirement: {}
  ShellCommandRequirement: {}

hints:
  DockerRequirement:
    dockerPull: kathrinklee/rna-seq-pipeline-hisat2
```

Below are the **inputs**, **outputs**, and **steps** blocks that come after. In **step1** the script **index_build.cwl** will be called, and its inputs (**in**) are taken from **inputs** section, the output (**out**) will be caught by the **outputs** (**step1/ht** and **step1/log** in this case). This declaration is important as it will decide which outputs to keep at the end of the step.

```
inputs:
  fasta_file: File
  out_name: string

outputs:
  index_files:
    type: Directory
    outputSource: step1/ht
  index_log:
    type: File
    outputSource: step1/log

steps:
  step1:
    run: index_build.cwl
    in:
      fasta_file: fasta_file
      out_name: out_name
    out:
      [ht, log]
```

Let's run it by using:

```
$ cwltool CWL-RNAseq.cwl CWL-RNAseq.yml
```

If you receive this error: "docker: Got permission denied while trying to connect to the Docker daemon socket at unix"

Try:

```
$ sudo groupadd docker
$ sudo usermod -aG docker ${USER}
close and reopen VM then run the script again
```

Let's take a look at the folder after cwltool finishes:

```
.
└── CWL-RNAseq
    ├── create_bam.cwl
    ├── create_transcript.cwl
    ├── CWL-RNAseq.cwl
    ├── CWL-RNAseq.yml
    ├── data
    │   ├── sample_1.fq
    │   ├── sample_2.fq
    │   ├── sample.fa
    │   └── sample.gtf
    ├── [final_ref.gtf]
    ├── [final_transcript.gtf]
    ├── [final.tsv]
    ├── hisat2_align.cwl
    ├── [hisat2_align_out]
    │   ├── [hisat2_align_out.log]
    │   └── [sample.sam]
    ├── [hisat2_build.log]
    ├── index_build.cwl
    ├── [sample]
    │   ├── [index.1.ht2]
    │   ├── [index.2.ht2]
    │   ├── [index.3.ht2]
    │   ├── [index.4.ht2]
    │   ├── [index.5.ht2]
    │   ├── [index.6.ht2]
    │   ├── [index.7.ht2]
    │   └── [index.8.ht2]
    └── [sample.bam]
```

The script will call hisat2 , samtools, and stringtie to do the work. **sample.sam** file will contains the sequence alignment data produced by mapping reads to the reference genome, **sample.bam** file will contains the compressed binary data from Sam. More description on gtf outputs, and tsv of stringtie can be found here. The **final_transcript.gtf** contains details of the transcripts that StringTie assembles from RNA-Seq data, while **final.tsv** contains gene abundances.

To see the result of this workflow, you can check it here.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

Running Nextflow RNA-seq

This Nextflow RNA-seq workflow maps read-pairs to a reference genome and produces a transcript.

By using software containers, Nextflow enables scalable and reproducible scientific workflows. Pipelines can be written in any common scripting language.

Requirements:

- Docker
- Java (required to install Nextflow)
- Nextflow
- Graphviz

To install Docker, Graphviz and Nextflow, see our VM Workflow Tools Installation Cheatsheet for instructions.

Note: The requirements above are crucial to running this workflow. Please make sure you have them installed properly prior to running this workflow.

Download this tutorial:

```
$ sudo add-apt-repository universe
$ sudo apt update
$ sudo apt install subversion

#cloning this tutorial
$ svn checkout https://github.com/isb-cgc/RunningWorkflows-on-the-GoogleCloud/trunk/
↪Nextflow-RNAseq
```

Running Nextflow

You should have a **Nextflow-RNAseq** directory :

```
Nextflow-RNAseq
          ├── data
          │   ├── sample_1.fq
          │   ├── sample_2.fq
          │   ├── sample.fa
          │   └── sample.gtf
          ├── main.nf
          └── nextflow.config
```

The file **main.nf** contains all the code to execute the workflow, and **nextflow.config** provides the name of the Docker image that contains all of the tools for this run.

Let's take a look at the contents of the file **main.nf**. The first block contains all the parameters (denoted by "params") that are required to input into the workflow.

```
#!/usr/bin/env nextflow

params.outdir = "$baseDir/results"
params.indexDir = "$baseDir/reference"
params.refGenome = "$baseDir/data/sample.fa"
params.forRead = "$baseDir/data/sample_1.fq"
params.revRead = "$baseDir/data/sample_2.fq"
params.gtfFile = "$baseDir/data/sample.gtf"
```

The process block contains 3 basic parts: **input**, **output**, and **command** (contained in the pairs of quotations """ some commands """). The variable **refGenome** with type **path** is created to hold the value of **params.refGenome**.

In the **command** (""") block, the hisat2 build command will be executed with the previously created variable **refGenome** denoted as **${refGenome}**. The name of the final output files from hisat2 build in this case will follow the pattern **index.#.ht2**.

In the output section, in order to catch the outputs of hisat2, the wildcard 'index*' was used, and those outputs will become a list assigned to the channel **index_ch*** variable. The **index_ch*** channel can be used by downstream processes. The other 3 processes have similar syntax.

```
process buildIndex {
 publishDir params.indexDir, mode: 'copy'
 echo true
 input:
 path refGenome from params.refGenome

 output:
 path 'index*' into index_ch

 """
 echo "Building Indices"
 hisat2-build ${refGenome} index
 """
}
```

To run:

```
#Assume the executable file "nextflow" is installed in the same directory with the␣
↪folder you downloaded "Nextflow-RNAseq".
$ ./nextflow run Nextflow-RNAseq
```

After the workflow finishes running, the folder should look like this:

```
Nextflow-RNAseq
          ├── data
          │   ├── sample_1.fq
          │   ├── sample_2.fq
          │   ├── sample.fa
          │   └── sample.gtf
          ├── main.nf
          ├── nextflow.config
          ├── [reference]
          │   ├── [index.1.ht2]
          │   ├── [index.2.ht2]
```

(continues on next page)

(continued from previous page)

```
│   ├── [index.3.ht2]
│   ├── [index.4.ht2]
│   ├── [index.5.ht2]
│   ├── [index.6.ht2]
│   ├── [index.7.ht2]
│   └── [index.8.ht2]
└── [results]
    ├── [final_ref.gtf]
    ├── [final_transcript.gtf]
    ├── [sample.bam]
    ├── [sample.sam]
    └── [sample.tsv]
```

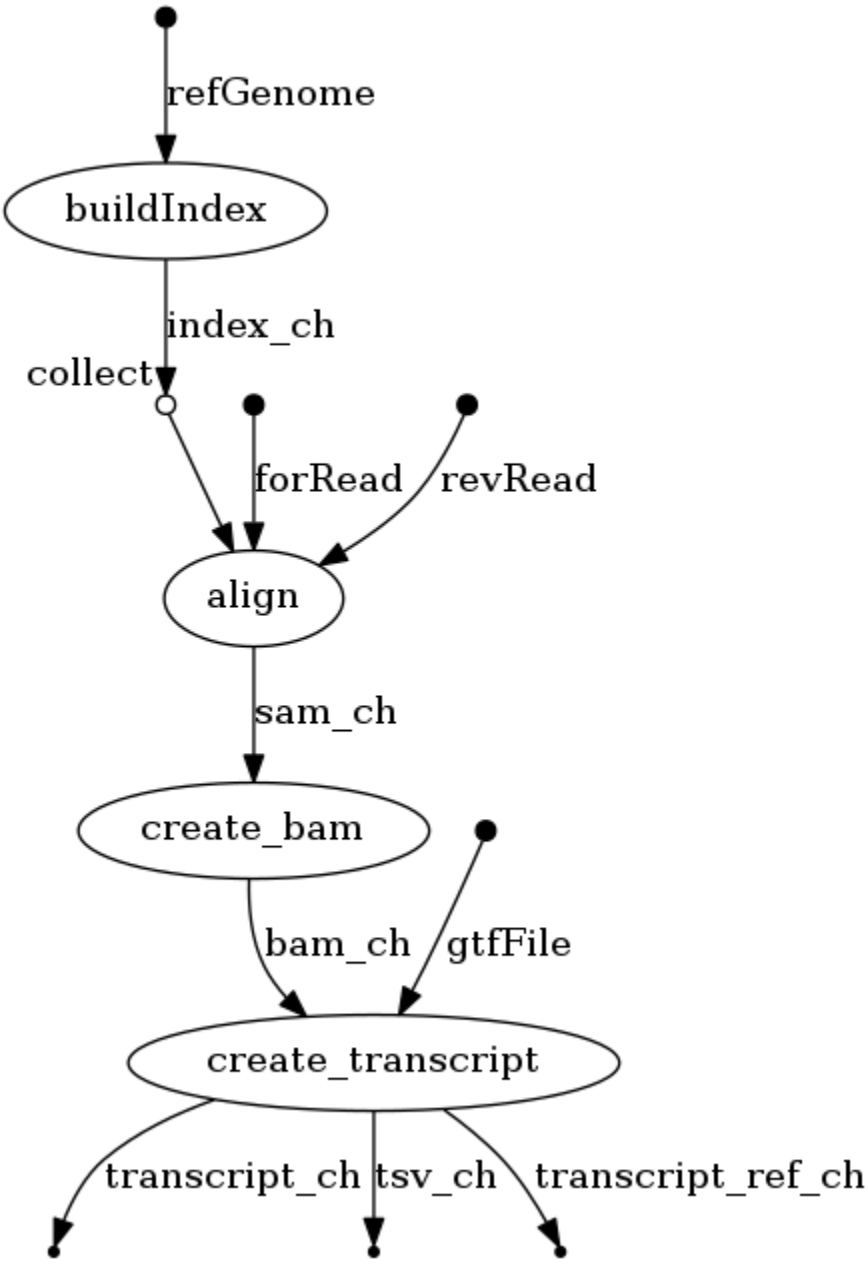
The script will call hisat2 , samtools, and stringtie to do the work. **sample.sam** file will contain the sequence alignment data produced by mapping reads to the reference genome, **sample.bam** file will contain the compressed binary alignment data. Additional information about the gtf and tsv outputs of StringTie can be found here. The **final_transcript.gtf** contains details of the transcripts that StringTie assembles from RNA-Seq data, while **final.tsv** contains gene abundances.

Running Nextflow with visualization

Use the following command:

```
#Assume the executable file "nextflow" is installed in the same directory with the␣
↪folder you downloaded "Nextflow-RNAseq".
$ ./nextflow run Nextflow-RNAseq -with-dag flowchart.png
```

An image file with the name **flowchart.png** will be available to download. It should look like this:



To see the result of this workflow, you can check it here.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

Running GeneFlow RNA-seq

This GeneFlow RNA-seq workflow maps read-pairs to a reference genome and produces a transcript.

GeneFlow enables modular and reproducible scientific workflows by leveraging reusable, containerized steps. Custom workflow steps can be implemented using either Docker or Singularity containers. Additional documentation for GeneFlow can be found here.

Requirements

- Docker
- Python 3
- GeneFlow

To install Docker, Python 3, and GeneFlow, see our VM Workflow Tools Installation Cheatsheet for instructions.

Note: The requirements above are crucial to running this workflow. Please make sure you have them installed properly prior to running this workflow.

Install the Workflow

```
$ sudo add-apt-repository universe
$ sudo apt update
$ sudo apt install subversion

# clone the workflow
$ svn checkout https://github.com/isb-cgc/RunningWorkflows-on-the-GoogleCloud/trunk/
↪GeneFlow-RNAseq

# install the workflow's apps
$ gf install-workflow --make-apps GeneFlow-RNAseq
```

Running GeneFlow

You should have a **GeneFlow-RNAseq** directory with the following contents:

```
GeneFlow-RNAseq
├── apps
│   ├── bam-sort
│   ├── hisat2-align
│   ├── hisat2-build
│   └── stringtie
├── data
│   ├── gtf
│   │   └── sample.gtf
│   ├── reads
│   │   ├── sample_1.fq
│   │   └── sample_2.fq
```

(continues on next page)

(continued from previous page)

```
│   └── reference
│       └── sample.fa
└── workflow.yaml
```

The file **workflow.yaml** contains the workflow definition, and the **apps** folder contains definitions for each of the workflow steps.

Let's take a look at the contents of the **workflow.yaml** file. The first block contains metadata, including workflow name, description, source repository, and version. The **final_output** block lists all steps whose output should be copied to the final output directory.

```
%YAML 1.1
---
gfVersion: v2.0
class: workflow

# metadata
name: HISAT2 StringTie Workflow
description: RNAseq workflow using HISAT2 and StringTie
git: https://github.com/isb-cgc/RunningWorkflows-on-the-GoogleCloud/GeneFlow-RNAseq
version: '0.1'

final_output:
- sort
- quantify

# inputs
inputs:
  reads:
    label: Input Directory
    description: Input directory containing FASTQ files
    type: Directory
    default: ./data/reads
    enable: true
    visible: true
  gtf:
    label: Input GTF
    description: GTF file describing transcriptome
    type: File
    default: ./data/gtf/sample.gtf
    enable: true
    visible: true
  reference:
    label: Reference Sequence FASTA
    description: Reference sequence FASTA file
    type: File
    default: ./data/reference/sample.fa
    enable: true
    visible: true

# parameters
parameters:
  threads:
```

(continues on next page)

(continued from previous page)

```
    label: CPU Threads
    description: Number of CPU threads for alignment
    type: int
    default: 2
    enable: false
    visible: true

# apps
apps:
  hisat2-build:
    git: https://github.com/geneflow-apps/hisat2-build-gf2.git
    version: '2.2.1-01'
  hisat2-align:
    git: https://github.com/geneflow-apps/hisat2-align-gf2.git
    version: '2.2.1-01'
  bam-sort:
    git: https://github.com/geneflow-apps/bam-sort-gf2.git
    version: '1.10-07'
  stringtie:
    git: https://github.com/geneflow-apps/stringtie-gf2.git
    version: '2.1.6-01'

# steps
steps:
  build:
    app: hisat2-build
    depend: []
    template:
      reference: ${workflow->reference}
      output: reference

  align:
    app: hisat2-align
    depend: [ "build" ]
    map:
      uri: ${workflow->reads}
      regex: (.*)_(R|)1(.*)\.((fastq|fq)(|\.gz))$
    template:
      input: ${workflow->reads}/${1}_${2}1${3}.${4}
      pair: ${workflow->reads}/${1}_${2}2${3}.${4}
      reference: ${build->output}/reference
      threads: ${workflow->threads}
      output: ${1}.sam

  sort:
    app: bam-sort
    depend: [ "align" ]
    map:
      uri: ${align->output}
      regex: (.*).sam
    template:
      input: ${align->output}/${1}.sam
```

(continues on next page)

(continued from previous page)

```
      output: ${1}.bam

  quantify:
    app: stringtie
    depend: [ "sort" ]
    map:
      uri: ${sort->output}
      regex: (.*).bam
    template:
      bam: ${sort->output}/${1}.bam
      gtf: ${workflow->gtf}
      output: ${1}
...
```

The **inputs** and **parameters** blocks define the inputs and parameters that need to be passed to the workflow upon execution. Some of these inputs and parameters are optional or have default values.

The **apps** block lists all apps used by the workflow and links to other, reusable source repositories for each app. Learn more about how each app works by following the Git repository links below:

- HISAT2 Build
- HISAT2 Align
- BAM Sort
- StringTie

The **steps** block defines the order of app execution as well as step dependencies for each app. It also defines how apps are chained together via their inputs and outputs.

To run the workflow:

```
# assuming the GeneFlow Python virtual environment has been activated, view the command␣
↪line help
$ gf help GeneFlow-RNAseq

# run the workflow
$ cd GeneFlow-RNAseq
$ gf run . -o output
```

After the workflow completes, the output folder should look similar to this:

```
output
└── geneflow-job-095ba2fe
    ├── quantify
    │   ├── _log
    │   │   ├── gf-0-quantify-sample.err
    │   │   ├── gf-0-quantify-sample.out
    │   │   ├── sample-stringtie.stderr
    │   │   └── sample-stringtie.stdout
    │   └── sample
    │       ├── e2t.ctab
    │       ├── e_data.ctab
    │       ├── i2t.ctab
    │       ├── i_data.ctab
```

(continues on next page)

(continued from previous page)

```
    │       ├── sample.tsv
    │       ├── sample_final_reference.gtf
    │       ├── sample_final_transcript.gtf
    │       └── t_data.ctab
    └── sort
        ├── _log
        │   ├── gf-0-sort-sample-bam.err
        │   ├── gf-0-sort-sample-bam.out
        │   └── sample.bam-samtools-sort.stderr
        └── sample.bam
```

The script will run Docker containers for hisat2, samtools, and stringtie to do the work. **sample.bam** contains the sequence alignment data produced by mapping reads to the reference genome **sample.bam**. Additional information about gtf and tsv outputs of stringtie can be found here. The **sample_final_transcript.gtf** contains details of the transcripts that StringTie assembles from RNA-Seq data, while **sample.tsv** contains gene abundances.

View the results of this workflow here.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

Running NEXTFLOW Blast

This workflow extracts contigs of interest from a mixed library of DNA; it uses Blastn and Python.

Requirements:

- Docker
- Java (required to install Nextflow)
- Python (to run the scripts)
- Nextflow
- Graphviz

To install Docker, Java, Python, Graphviz and Nextflow, see our VM Workflow Tools Installation Cheatsheet for instructions.

Note: The requirements above are crucial to running this workflow. Please make sure you have them installed properly prior to running this workflow.

Download this tutorial:

```
$ sudo add-apt-repository universe
$ sudo apt update
$ sudo apt install subversion

#cloning this tutorial
$ svn checkout https://github.com/isb-cgc/RunningWorkflows-on-the-GoogleCloud/trunk/
↪Nextflow-Blast
```

Rationale

The input of this workflow (**sample.fa**) is produced by genome assembly software. It contains contigs from multiple organisms (eukaryote, prokaryotes) because of the way the sample was prepared and sequenced. In this example, we are interested in the prokaryote genome only, and will separate their contigs from the eukaryote ones. In order to do that, Blastn will do sequence alignment between the contigs and the prepared blast database (files in the folder **db**). The output of Blastn will be a table, which maps the contig header to the appropriate species. Here, we also demonstrate how to incorporate the use of python helper scripts in our pipeline to extract the prokaryotes' headers (**scripts/Extract_Headers.py**), contigs (**scripts/Extract_Contigs.py**), and nucleotide count (**scripts/Count_Nucleotides.py**) from the original Fasta file using the Blastn result. The final output of the workflow will be 3 text files: **extracted_contigs.txt**, **Headers.txt**, **NucleoCount.txt**.

Running Nextflow

You should have a **Nextflow-Blast** directory :

```
Nextflow-Blast
    ├── data
    │   └── sample.fa
    ├── db
    │   ├── SampleDB.nhr
    │   ├── SampleDB.nin
    │   └── SampleDB.nsq
    ├── main.nf
    ├── nextflow.config
    └── scripts
        ├── Count_Nucleotides.py
        ├── Extract_Contigs.py
        └── Extract_Headers.py
```

The file **main.nf** contains all the code necessary to execute the workflow, and **nextflow.config** provides the name of docker image that contains all the tools for this run. To run:

```
#Assume the executable file "nextflow" is installed in the same directory with the␣
↪folder you download "Nextflow-Blast"
$ ./nextflow run Nextflow-Blast
```

Once complete, the folder should look like this:

```
Nextflow-Blast
      ├── data
```

(continues on next page)

(continued from previous page)

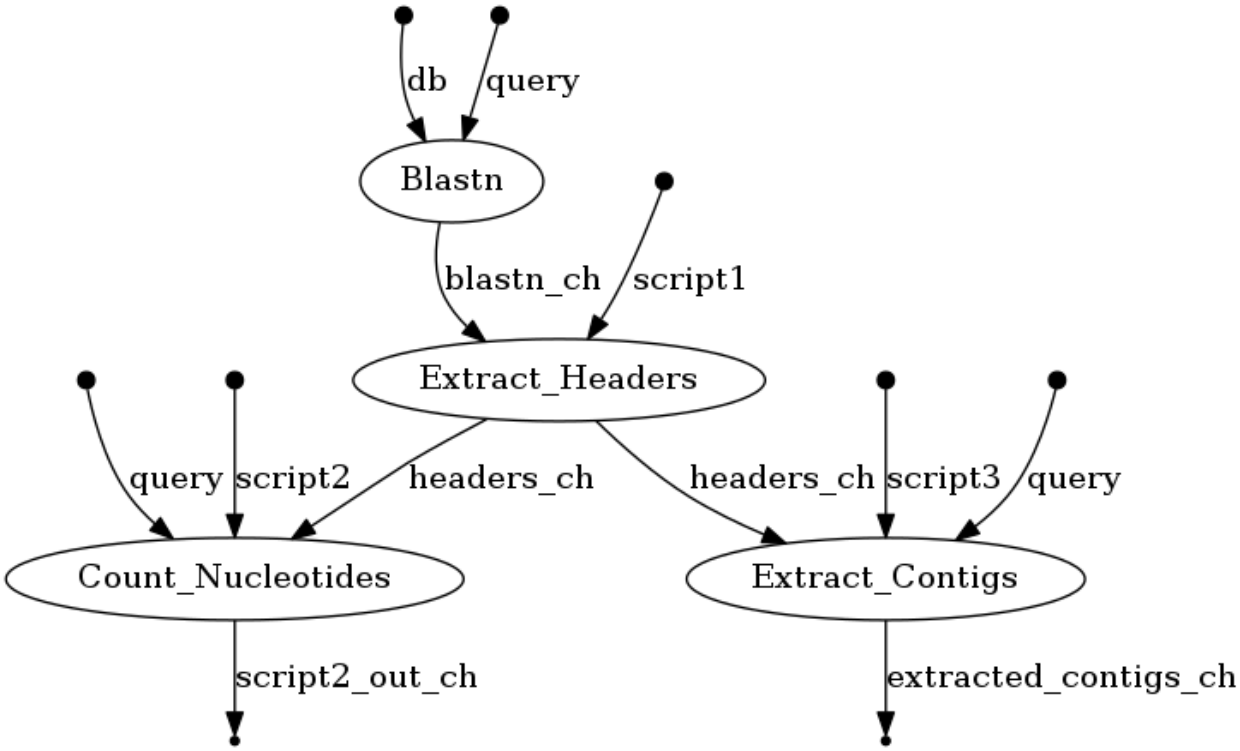
```
        │   └── sample.fa
        ├── db
        │   ├── SampleDB.nhr
        │   ├── SampleDB.nin
        │   └── SampleDB.nsq
        ├── main.nf
        ├── nextflow.config
        ├── [results]
        │   ├── [Blastn.out]
        │   ├── [extracted_contigs.txt]
        │   ├── [Headers.txt]
        │   └── [NucleoCount.txt]
        └── scripts
            ├── Count_Nucleotides.py
            ├── Extract_Contigs.py
            └── Extract_Headers.py
```

Running Nextflow with the Visualization Option

Use the following command:

```
#Assume the executable file "nextflow" is installed in the same directory with the␣
↪folder you download "Nextflow-Blast"
$ ./nextflow run Nextflow-Blast -with-dag flowchart.png
```

An image file with the name **flowchart.png** will be available to download. It should look like this:



To see the result of this workflow, you can check it here.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

Running CWL Blast

This workflow extracts contigs of interest from a mixed library of DNA; it uses Blastn and Python.

Requirements:

- CWLtool
- Docker
- Python (to run the scripts)

To install Docker, Python, and CWL, see our VM Workflow Tools Installation Cheatsheet for instructions.

Note: The requirements above are crucial to running this workflow. Please make sure you have them installed properly prior to running this workflow.

Download this tutorial:

```
$ sudo add-apt-repository universe
$ sudo apt update
$ sudo apt install subversion

#cloning this tutorial
$ svn checkout https://github.com/isb-cgc/RunningWorkflows-on-the-GoogleCloud/trunk/CWL-
↪Blast
```

Rationale

The input of this workflow (**sample.fa**) is produced by genome assembly software. It contains contigs from multiple organisms (eukaryotes, prokaryotes) because of the way the sample was prepared and sequenced. In this example, we are interested in the prokaryotic genomes only, and will separate their contigs from the eukaryotic ones. In order to do that, Blastn will do sequence alignment between the contigs and the prepared blast database (files in the folder **db**). The output of Blastn will be a table, which maps the contig headers to the appropriate species. Then we demonstrate how to incorporate the use of python helper scripts in our pipeline to extract the prokaryotes' headers (**scripts/Extract_Headers.py**), contigs (**scripts/Extract_Contigs.py**), and nucleotide count (**scripts/Count_Nucleotides.py**) from the original Fasta file using the Blastn result. The final output of the workflow will be 3 text files: **extracted_contigs.txt**, **Headers.txt**, **NucleoCount.txt**.

Running CWL

You should have a **CWL-Blast** directory :

```
CWL-Blast
├── blast.cwl
├── count_nucleotides.cwl
├── CWL-Blast.cwl
├── CWL-Blast.yml
├── data
│   └── sample.fa
├── db
│   ├── SampleDB.nhr
│   ├── SampleDB.nin
│   └── SampleDB.nsq
├── extract_contigs.cwl
├── extract_headers.cwl
└── scripts
    ├── Count_Nucleotides.py
    ├── Extract_Contigs.py
    └── Extract_Headers.py
```

Let's run it by using:

```
$ cd CWL-Blast
$ cwltool CWL-Blast.cwl CWL-Blast.yml
```

Let's take a look at the folder after cwltool finishes:

```
CWL-Blast
  ├── blast.cwl
  ├── Blastn.out
  ├── count_nucleotides.cwl
  ├── CWL-Blast.cwl
  ├── CWL-Blast.yml
  ├── data
  │   └── sample.fa
  ├── db
  │   ├── SampleDB.nhr
  │   ├── SampleDB.nin
  │   └── SampleDB.nsq
  ├── extract_contigs.cwl
  ├── extracted_contigs.txt
  ├── extract_headers.cwl
  ├── Headers.txt
  ├── NucleoCount.txt
  └── scripts
      ├── Count_Nucleotides.py
      ├── Extract_Contigs.py
      └── Extract_Headers.py
```

Full resulting files from running this workflow are deposited in the github repo here.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

Example workflows accessing data in Google Cloud Storage

We demonstrate here: **(1)** how to use the Google Cloud Storage FUSE (GCSFuse) tool to mount a Cloud Storage bucket to your virtual machine instance. Despite the fact that the Cloud storage buckets are object store, once mounted it behaves like a persistent disk. **(2)** how to use GCSFuse to mount a cloud storage bucket that contains CCLE open-access bam files to a virtual machine to run on your workflow of choice.

- Setting Up GCSFuse
- Running Nextflow pipeline on public BAM file from ISB-CGC
- Running Snakemake pipeline on public BAM file from ISB-CGC
- Running CWL pipeline on public BAM file from ISB-CGC
- Running WDL pipeline on public BAM file from ISB-CGC

Setting up GCSFuse

When you are running workflow on a virtual machine (VM), often your input files are stored in Google Cloud Storage buckets. One way to access them is to mount the Cloud Storage buckets as file systems on your VM. Google Cloud Storage FUSE (GCSFuse) allows you to mount Cloud Storage buckets to easily read and write from your VM to your Cloud Storage buckets. More detailed information can be found on the Google Cloud documentation page.

How-to video | Installing Page

Step 1: create a Virtual Machine (VM) instance big enough to hold your data

This guide recommends your VM be created with: **Ubuntu 16.04 LTS**, and with the **Allow full access to all Cloud APIs** option.

Note: It's very important to have a VM big enough, or your gcsfuse will not mount properly.

Step 2: installing gcsfuse

The following commands can be used to install gcsfuse:

```
$ sudo -i
$ cd /
$ cd opt
$ export GCSFUSE_REPO=gcsfuse-`lsb_release -c -s`
$ echo "deb http://packages.cloud.google.com/apt $GCSFUSE_REPO main" | sudo tee /etc/apt/
↪sources.list.d/gcsfuse.list
$ curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
$ sudo apt-get update
$ sudo apt-get install gcsfuse
#### Close the VM console and reopen #####
```

Step 3: mounting/ linking the target bucket to your VM directory

Making a directory to hold your bucket:

```
### at your home directory and not in opt ###
### in this tutorial yourNewDirectory will be testGcsfuse ###

$ mkdir <yourNewDirectory>
```

Note: To access restricted data with your Google credential, before going further, use this command: $gcloud auth application-default login

Mount a bucket to your folder:

```
$ gcsfuse <bucketname> <myfolder/to/mount>
```

Example: gs://gdc-ccle-open/ is the bucket you want to mount to your VM

```
$ gcsfuse gdc-ccle-open testGcsfuse
```

Mount a subdirectory from your bucket to your VM folder:

```
$ gcsfuse --only-dir <subdirectory> <bucketName> <myFolder/to/mount>
```

Example: you have a bam file with the address gs://gdc-ccle-open/692a845c-7957-41f2-b679-5434c69ba25b/G27328.Calu-6.1.bam, and you only need to mount the directory that holds that bam file:

```
$ gcsfuse --only-dir 692a845c-7957-41f2-b679-5434c69ba25b gdc-ccle-open testGcsfuse
```

You should see something like this:

```
Opening GCS connection...
Opening bucket...
Mounting file system...
File system has been successfully mounted.
```

```
(base) thinh_vo@kv-test:~$ cd testGcsfuse/
(base) thinh_vo@kv-test:~/testGcsfuse$ ls
G27328.Calu-6.1.bam
```

Step 4: running your workflow with a local VM directory

Write your workflow with the input pointing to that directory, as follows:

```
IDS, = glob_wildcards("/home/thinh_vo/testGcsfuse/{sample}.bam")
#this rule determine which is the output of this workflow
rule all:
    input:
        "final/final_gc_stats_out.txt"
#this task use sammtools to produce stats from bam files
rule samtools_stats_tool:
    input:
        "/home/thinh_vo/testGcsfuse/{sample}.bam"
```

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

Running Nextflow pipeline on public BAM file from ISB-CGC

This Nextflow workflow gathers GC content from a BAM file (or a list of BAM files) to a text file. By using software containers, Nextflow enables scalable and reproducible scientific workflows. Pipelines can be written in the most common scripting languages.

Requirements:

- Docker
- Java (required to install Nextflow)
- Gcsfuse
- Nextflow
- A public bam file from ISB-CGC at the address: gs://gdc-ccle-open/692a845c-7957-41f2-b679-5434c69ba25b/G27328.Calu-6.1.bam

To install Docker and Nextflow, see our VM Workflow Tools Installation Cheatsheet for instructions. To set up gcsfuse in order to get access to the BAM file, please visit Running Workflow with GCSFUSE.

Note: The requirements above are crucial to running this workflow. Please make sure you have them installed properly prior to running this workflow.

Download this tutorial:

```
$ sudo add-apt-repository universe
$ sudo apt update
$ sudo apt install subversion

#cloning this tutorial
$ svn checkout https://github.com/isb-cgc/RunningWorkflows-on-the-GoogleCloud/trunk/
↪Nextflow-GCgather
```

Running Nextflow

You should have a **Nextflow-GCgather** directory with one file called **Nextflow-GCgather.nf** inside. We are going to change the address in this file to the one you created in the **Running Workflow with GCS-FUSE** tutorial.

```
#go into the folder
$ cd Nextflow-GCgather
$ nano Nextflow-GCgather.nf
```

At the top of the file you will see this:

```
#!/usr/bin/env nextflow
myBamSample = Channel.fromPath('/home/thinh_vo/sample/*.bam')
```

Replace "/home/thinh_vo/sample/*.bam" with your new address from the gcsfuse tutorial for example: "/home/thinh_vo/testGcsfuse/*.bam". Now the script is ready to run with Nextflow.

```
#Go to where the Nextflow executable file was installed in this example. It will be␣
↪outside the Nextflow-GCgather directory.
#First, we get out of Nextflow-GCgather directory.
$ cd ..
#execute nextflow with docker image:
$ ./nextflow run Nextflow-GCgather/Nextflow-GCgather.nf -with-docker gcr.io/genomics-
↪tools/samtools
```

Note: This BAM file is quite large; it may take about 15 mins ~ 20 mins to run.

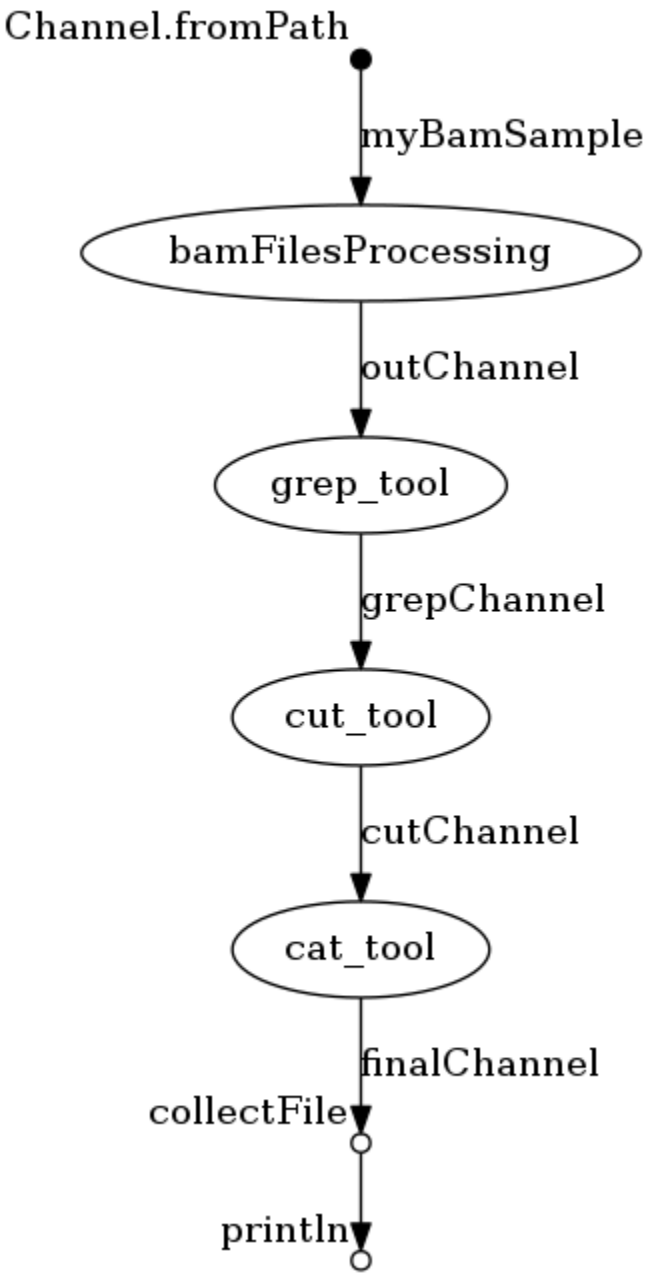
Once Nextflow is finished, the result will be on the screen, or you can find it at **Nextflow-GCgather/Sam_results/final_gc_stats_out.txt**.

Running Nextflow with visualization

You can use this command instead to run Nextflow; it will out put a visualization file named "flowchart.png".

```
$ ./nextflow run Nextflow-GCgather/Nextflow-GCgather.nf -with-dag flowchart.png
```

It should look like this:



To see the result of this workflow, you can check it here.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

Running SNAKEMAKE pipeline on public Bam file from ISB-CGC

This workflow gathers GC content from a bam file or a list of Bam files to a text file.

Requirements:

- Anaconda/Miniconda and how to activate the conda environment from a file
- Gcsfuse
- A public bam file from ISB-CGC at the address: gs://gdc-ccle-open/692a845c-7957-41f2-b679-5434c69ba25b/G27328.Calu-6.1.bam

To install and set up Anaconda/Miniconda, you can visit the first tutorial Your First Workflow on Google Cloud Virtual Machine, To set up gcsfuse in order to get access to the BAM file, please visit Running Workflow with GCSFUSE.

Note: The requirements above are crucial to running this workflow. Please make sure you have them installed properly prior to running this workflow

Download this tutorial:

```
$ sudo add-apt-repository universe
$ sudo apt update
$ sudo apt install subversion

#cloning this tutorial
$ svn checkout https://github.com/isb-cgc/RunningWorkflows-on-the-GoogleCloud/trunk/
↪Snakemake-GCgather
```

Running Snakemake

You should have a **Snakemake-GCgather** directory with 2 files **Snakefile** and **environment.yml** inside. We are going to activate the conda environment from the file (see **Your First Workflow on Google Cloud Virtual Machine** for more information) . Then we are going to change change the address in Snakefile to the one you created in the **Running Workflow with GCSFUSE** tutorial

```
#go into the folder
$ cd Snakemake-GCgather
$ nano Snakefile
```

At the top of the file you will see this:

```
#line 1:
IDS, = glob_wildcards("/home/thinh_vo/testGcsfuse/{sample}.bam")
#line 7-8:
input:
    "/home/thinh_vo/testGcsfuse/{sample}.bam"
```

Replace "/home/thinh_vo/testGcsfuse/{sample}.bam" with your new address from the gcsfuse tutorial for example: "/home/thinh_vo/testGcsfuse/{sample}.bam". Now the script is ready to run with Snakemake.

```
$ snakemake
```

Note: This Bam file is quite large, it may take about 15 mins ~ 20 mins to run.

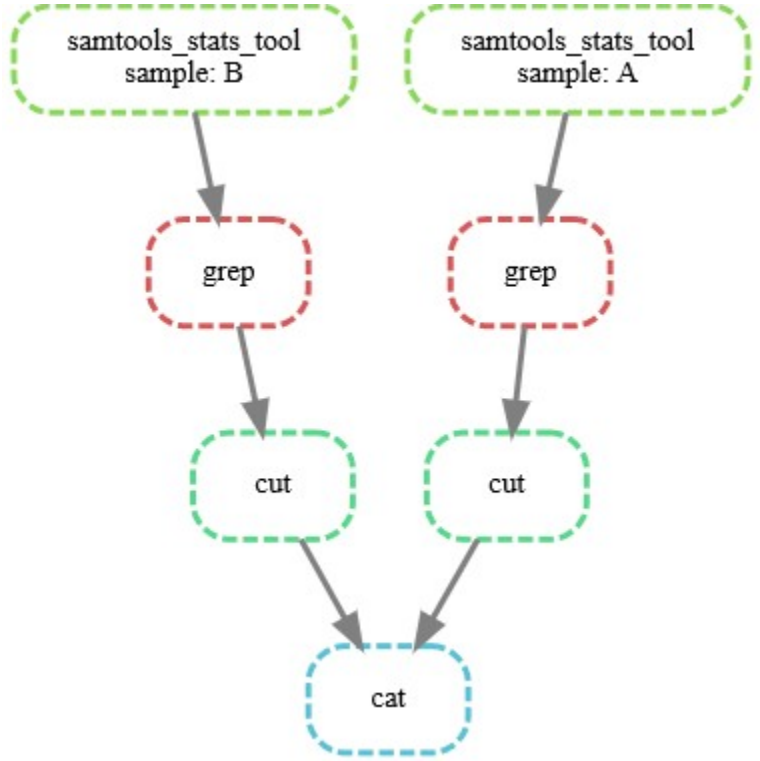
Once snakemake is finished, the result will be on the screen, or you can find it at **Snakemake-GCgather/final/final_gc_stats_out.txt**

Running Snakemake with visualization

You can use this command instead to run Snakemake, it will out put a visualization file named "flowchart.svg"

```
$ snakemake --dag | dot -Tsvg > flowchart.svg
```

It should look like this:



To see the result of this workflow, you can check it here

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

Running CWL pipeline on public Bam file from ISB-CGC

This workflow gathers GC content from a BAM file (or a list of BAM files) to a text file.

Requirements:

- Docker
- Gcsfuse
- CWLtool (CWL)
- A public bam file from ISB-CGC at the address: gs://gdc-ccle-open/692a845c-7957-41f2-b679-5434c69ba25b/G27328.Calu-6.1.bam

To install Docker and CWL, see our VM Workflow Tools Installation Cheatsheet for instructions. To set up gcsfuse in order to get access to the BAM file, please visit Running Workflow with GCSFUSE.

Note: The requirements above are crucial to running this workflow. Please make sure you have them installed properly prior to running this workflow.

Download this tutorial:

```
$ sudo add-apt-repository universe
$ sudo apt update
$ sudo apt install subversion

#cloning this tutorial
$ svn checkout https://github.com/isb-cgc/RunningWorkflows-on-the-GoogleCloud/trunk/CWL-
↪GCgather
```

Running CWLtool

You should have a **CWL-GCgather** directory with 6 files inside: 1 main workflow file "CWL-GCgather.cwl", 4 ***tools.cwl** and . We are going to change the address in **scatter_gather_pipeline.yml** file to the one you created in the **Running Workflow with GCSFUSE** tutorial

```
#go into the folder
$ cd CWL-GCgather
$ nano scatter_gather_pipeline.yml
```

At the top of the file you will see this:

```
filein:
  - {class: File, path: /opt/testGcsfuse/G27328.Calu-6.1.bam}
```

Replace "/home/thinh_vo/testGcsfuse/G27328.Calu-6.1.bam" with your new address from the gcsfuse tutorial for example: "/home/thinh_vo/testGcsfuse/G27328.Calu-6.1.bam". Now the script is ready to run with CWLtool. Save the change, then run the script with this command:

```
$ cwltool CWL-GCgather.cwl scatter_gather_pipeline.yml
```

If you receive this error: "docker: Got permission denied while trying to connect to the Docker daemon socket at unix"

Try:

```
$ sudo groupadd docker
$ sudo usermod -aG docker ${USER}
close and reopen VM then run the script again
```

Note: This Bam file is quite large, it may take about 15 mins ~ 20 mins to run.

Once CWLtool is finished, the result will be in the same folder called "final_output.txt"

To see the result of this workflow, you can check it here

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

Running WDL pipeline on public Bam file from ISB-CGC

This workflow gathers GC content from a BAM file (or a list of BAM files) to a text file.

Requirements:

- Java
- Graphviz
- Docker
- Gcsfuse
- WDL (Cromwell and Womtool)
- A public bam file from ISB-CGC at the address: gs://gdc-ccle-open/692a845c-7957-41f2-b679-5434c69ba25b/G27328.Calu-6.1.bam

To install Java, Docker, Graphviz and WDL, see our VM Workflow Tools Installation Cheatsheet for instructions. To set up gcsfuse in order to get access to the BAM file, please visit Running Workflow with GCSFUSE.

Note: The requirements above are crucial to running this workflow. Please make sure you have them installed properly prior to running this workflow.

Installing Cromwell and Womtool:

Download the **cromwell-XY.jar** and **womtool-XY.jar** at https://github.com/broadinstitute/cromwell/releases to your local machine

Note: "XY" in **cromwell-XY.jar** and **womtool-XY.jar** is the version of the software. At the time of this tutorial, the latest version is "51". For all the code from here "XY" will be "51" e.g. **cromwell-XY.jar** becomes **cromwell-51.jar**. Please change the code accordingly to reflect the version that you have.

Example using Wget:

```
$ wget https://github.com/broadinstitute/cromwell/releases/download/52/cromwell-52.jar
$ wget https://github.com/broadinstitute/cromwell/releases/download/52/womtool-52.jar
```

Download this tutorial:

```
$ sudo add-apt-repository universe
$ sudo apt update
$ sudo apt install subversion

#cloning this tutorial
$ svn checkout https://github.com/isb-cgc/RunningWorkflows-on-the-GoogleCloud/trunk/WDL-
↪GCgather
```

Running WDL

You should have a **WDL-GCgather** directory with 3 files inside. to simplify the command, I will move the **cromwell-51.jar** and **womtool-51.jar** into the *WDL-GCgather** directory:

```
$ mv *-51.jar WDL-GCgather/
```

We are going to change the address in **bamfiles.txt** file to the one you created in the **Running Workflow with GCS-FUSE** tutorial

```
#go into the folder
$ cd WDL-GCgather
$ nano bamfiles.txt
```

At the top of the file you will see this:

```
/home/thinh_vo/testGcsfuse/G27328.Calu-6.1.bam        bam1
```

Replace "/home/thinh_vo/testGcsfuse/G27328.Calu-6.1.bam" with your new address from the gcsfuse tutorial for example: "/home/thinh_vo/testGcsfuse/G27328.Calu-6.1.bam". Now the script is ready to run with WDL. Save the change, then run the script with this command:

```
$ java -jar cromwell-51.jar run gcstats.wdl -i gcstats.inputs
```

Note: This Bam file is quite large, it may take about 15 mins ~ 20 mins to run. Also make sure you have installed Java.

Once the Cromwell finished, the result will be located inside the folder called **cromwell-executions** (also they will announce right before they stop running where the output is (see image below))

It should look like this, the **final_gc_stats_out.txt** is the final output of this workflow :

```
thinh_vo@kv-testgit: ~/WDL-GCgather - Google Chrome
ssh.cloud.google.com/projects/isb-cgc-interns/zones/us-central1-a/instances/kv-testgit?authuser=0&hl=en_US&projectNumber=59953508660
[2020-07-02 00:03:03,72] [info] WorkflowExecutionActor-ecb519d7-5571-4804-ab63-
5f5796608569 [ecb519d7]: Workflow gcStats complete. Final Outputs:
{
  "gcStats.cat_tool.finalfile": "/home/thinh_vo/WDL-GCgather/cromwell-execution
s/gcStats/ecb519d7-5571-4804-ab63-5f5796608569/call-cat_tool/execution/final_gc
_stats_out.txt",
  "gcStats.grep_tool.grepout": ["/home/thinh_vo/WDL-GCgather/cromwell-execution
s/gcStats/ecb519d7-5571-4804-ab63-5f5796608569/call-grep_tool/shard-0/execution
/grep_out.txt"],
  "gcStats.cut_tool.cuttoolout": ["/home/thinh_vo/WDL-GCgather/cromwell-executi
ons/gcStats/ecb519d7-5571-4804-ab63-5f5796608569/call-cut_tool/shard-0/executio
n/cut_out.txt"],
  "gcStats.samtools_stats_tool.statsout": ["/home/thinh_vo/WDL-GCgather/cromwel
l-executions/gcStats/ecb519d7-5571-4804-ab63-5f5796608569/call-samtools_stats_t
ool/shard-0/execution/bam1_gc_stats.txt"]
}
[2020-07-02 00:03:03,79] [info] WorkflowManagerActor WorkflowActor-ecb519d7-557
1-4804-ab63-5f5796608569 is in a terminal state: WorkflowSucceededState
[2020-07-02 00:03:10,72] [info] SingleWorkflowRunnerActor workflow finished wit
h status 'Succeeded'.
{
```

```
thinh_vo@kv-testgit:~/WDL-GCgather$ ls
bamfiles.txt  cromwell-51.jar  cromwell-executions  [illegible]  File1.bam  gcstats.inputs  gcstats.wdl  womtool-51.jar
thinh_vo@kv-testgit:~/WDL-GCgather$ cd cromwell-executions/
thinh_vo@kv-testgit:~/WDL-GCgather/cromwell-executions$ ls
gcStats
thinh_vo@kv-testgit:~/WDL-GCgather/cromwell-executions$ cd gcStats/
thinh_vo@kv-testgit:~/WDL-GCgather/cromwell-executions/gcStats$ ls
01c9941f-ee13-4266-8934-978f18176a71
thinh_vo@kv-testgit:~/WDL-GCgather/cromwell-executions/gcStats$ cd 01c9941f-ee13-4266-8934-978f18176a71/
thinh_vo@kv-testgit:~/WDL-GCgather/cromwell-executions/gcStats/01c9941f-ee13-4266-8934-978f18176a71$ ls
[illegible]
thinh_vo@kv-testgit:~/WDL-GCgather/cromwell-executions/gcStats/01c9941f-ee13-4266-8934-978f18176a71$ cd call-cat_tool/
thinh_vo@kv-testgit:~/WDL-GCgather/cromwell-executions/gcStats/01c9941f-ee13-4266-8934-978f18176a71/call-cat_tool$ ls
[illegible]
thinh_vo@kv-testgit:~/WDL-GCgather/cromwell-executions/gcStats/01c9941f-ee13-4266-8934-978f18176a71/call-cat_tool$ cd execution/
thinh_vo@kv-testgit:~/WDL-GCgather/cromwell-executions/gcStats/01c9941f-ee13-4266-8934-978f18176a71/call-cat_tool/execution$ ls
docker_cid  final_gc_stats_out.txt  rc  script  script.background  script.submit  stderr  stderr.background  stdout  stdout.background
thinh_vo@kv-testgit:~/WDL-GCgather/cromwell-executions/gcStats/01c9941f-ee13-4266-8934-978f18176a71/call-cat_tool/execution$
```

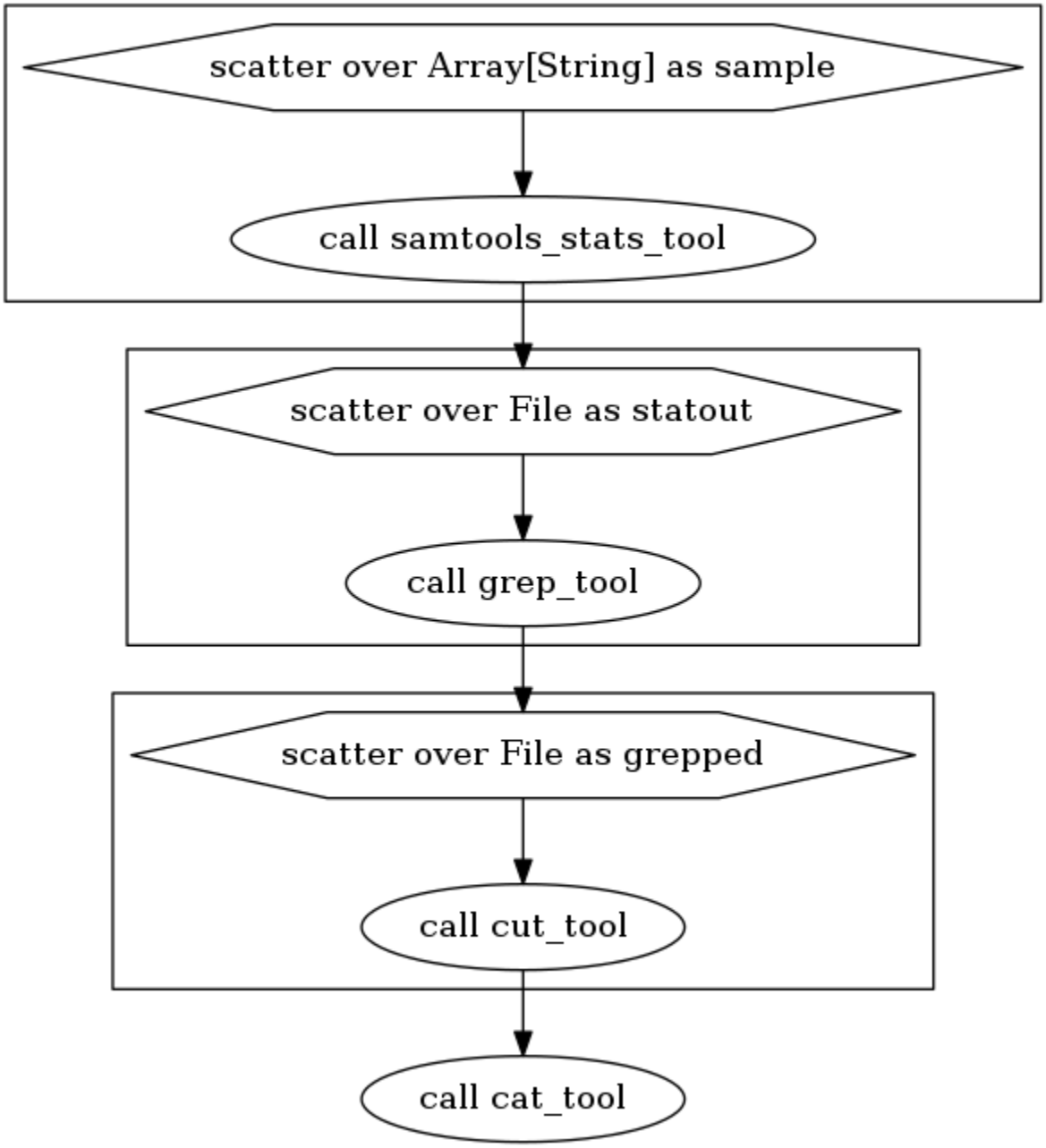
Creating visualization with Womtool

Run this command:

```
$ java -jar womtool-51.jar graph gcstats.wdl | dot -Tpng > visual.png
```

A file named "visual.png" will be created and ready to be downloaded

It should look like this:



To see the result of this workflow, you can check it here.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

19.3 St. Jude Bioinformatics Tools

The following bioinformatics tools and workflows developed by St. Jude have been containerized and made available for execution in the cloud. Each link below navigates to the tools' original documentation. If you would like guidance on how to run these on ISB-CGC, please attend our office hours or contact us (feedback@isb-cgc.org).

CICERO (Clipped-reads Extended for RNA Optimization) is an assembly-based algorithm to detect diverse classes of driver gene fusions from RNA-seq.	GitHub
RNAIndel calls coding indels from tumor RNA-Seq data and classifies them as somatic, germline, and artifactual.	GitHub
Teltale is a program that computes the fraction of telomeric reads in a BAM file.	GitHub
NetBID (Network-based Bayesian Inference of Drivers) is a data-driven system biology pipeline and toolkit for finding drivers from transcriptomics, proteomics and phosphoproteomics data, where the drivers can be either transcription factors (TF) or signaling factors (SIG).	GitHub

---

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CHAPTER

TWENTY

DATA ACCESS AND SECURITY OVERVIEW

20.1 Understanding Data Access Levels

- **Public Data** Sometimes the word "public" is misinterpreted as meaning "open". All of the TCGA data is *public* data, and much of it is *open*, meaning that it is accessible and available to *all* users; while some low-level TCGA data is *controlled* and restricted to authorized users.
- **Open-Access Data** Depending on how you categorize the data, *most* of the TCGA data is open-access data. This includes all de-identified clinical and biospecimen data, as well as all Level-3 molecular data including gene expression data, DNA methylation data, DNA copy-number data, protein expression data, somatic mutation calls, etc.
- **Controlled-Access Data** All low-level sequence data (both DNA-seq and RNA-seq), the raw SNP array data (CEL files), germline mutation calls, and a small amount of other data are treated as *controlled* data and require that a user is properly authenticated and have dbGaP authorization prior to accessing these data.

Note that many public, open-access datasets may still be **restricted** in various ways. Typically, a **License** document containing explicit terms of use will be associated with each dataset. Some institutions have their own licenses, though many uses one of the Creative Commons licenses. License terms apply to both data and source-code, so please be aware of the terms of a license whenever you plan to reuse data or source code produced by someone else. We recommend that you review the TCGA Publication Guidelines.

20.2 Understanding Data Security

Much of the low-level TCGA and TARGET data (including DNA and RNA reads, and SNP CEL files, for example) are classified as "controlled access data" and are under the control of the dbGaP Data Access Committee (DAC).

Investigator(s) requesting to receive genomic data in accordance with the NIH Genomic Data Sharing Policy are required to submit:

- a **data access request** (DAR)
- a **research use statement** (RUS)

Note: Requesters and institutional signing officials (SO) must have NIH eRA user IDs to begin this process. Visit the electronic Research Administration (eRA) for more information on registering for an NIH eRA account. NIH staff may utilize their NIH login. (See the dbGaP Data Access Request Portal for additional instructions.)

Additionally, they must:

- Submit a Data Use Certification (DUC) co-signed by the designated Institutional Official(s) at their sponsoring institution

- Protect data confidentiality (any data which has been designated "controlled" **must** be protected accordingly, unless prior release authorization is obtained from an NCI data custodian)
- Ensure that appropriate data security measures are in place

Google Cloud Platform and Access Control

In the context of Google Cloud Platform (GCP) projects, it is important to realize that all members of a GCP project must have at least read access to all data stored within that project, as well as to all virtual machines, boot disks, and persistent disks attached to that project.

Therefore, if a principle investigator (PI) establishes a GCP project (project-A) for the purposes of analyzing controlled data (*eg* performing mutation analysis on TCGA sequence data), then:

- *All* members of project-A must be authorized to view controlled data.
- The outputs of certain analyses performed on controlled data, if they are summary in nature, may no longer be controlled data and could be copied to a second GCP project (project-B) for further downstream analyses by researchers who are not authorized to view controlled data.
- Researchers who are not authorized to view controlled data could be made members of project-B, while users who *are* authorized could be members of both project-A *and* project-B.

Your Responsibilities

The PI and the PI's institution are *responsible* for and will be held *accountable* for ensuring the security of controlled data, not the cloud service provider. The Google Cloud Platform has been certified as FedRAMP compliant which means that it has been independently assessed and shown to meet all necessary FedRAMP security controls. This provides the assurance that the data security and access control mechanisms implemented by the Google Cloud Platform and made available to end users are sufficient to safeguard the data. However, it remains the PI's responsibility to ensure that these access control mechanisms are used appropriately and effectively within the context of the PI's GCP project.

You should think about securing controlled data within the context of your GCP project in the same way that you would think about securing controlled data that you might download to a file server or compute cluster at your own institution. Your responsibilities regarding the appropriate use of the data are the same in a cloud environment. For more information, please refer to the NIH Security Best Practices for Controlled-Access Data.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CHAPTER

TWENTYONE

ACCESSING CONTROLLED DATA

In ISB-CGC, you can gain access to controlled data via personal user credentials:

- Provides access to controlled data for 24 hours at a time;
- Uses your *personal* credentials;
- Example uses: the ISB-CGC Web App, R Studio or running short jobs on Google Compute Engine that complete in under 24 hours

Note: If you are looking to gain access to COSMIC data, please see the COSMIC documentation.

21.1 Prerequisites

You'll need the following before requesting controlled access via ISB-CGC:

- A Google identity;
- An NIH or electronic Research Administration (eRA) account;
- Database of Genotypes and Phenotypes (dbGaP) permission for each type of controlled access data of interest, linked to your NIH or eRA account;
- Your Google identify linked to your NIH/eRA account via the ISB-CGC Web App.

21.1.1 1) Google identity

If you don't have a Google identity yet, please see the ISB-CGC Quick-Start Guide.

21.1.2 2) NIH or eRA account

Intramural researchers can use their NIH log-in account, and extramural researchers will need to have a personal eRA account. Either way, the user's NIH/eRA account needs to be affiliated with their institution's eRA account. Your principal investigator (PI) or other authorized person can create your personal eRA account and link it to your institution's eRA account.

If you already have an NIH/eRA account, you can log into eRA at https://public.era.nih.gov/commons.

- If the Institution listed for you is not your current one, ask your PI to change it for you.
- If you are the PI or other authorized person, you can create, link and update accounts from here.

Visit electronic Research Administration (eRA) for more information on registering for a NIH eRA account.

21.1.3 3) Link your eRA (or NIH) account to dbGaP permissions.

Your principal investigator (PI) can link your NIH/eRA account to dbGaP permissions for selected controlled access data sets.

For more information on applying for dbGaP authorization to access controlled data, please see this dbGap How to Video: Apply for Controlled Access Data.

For additional information, refer to Tips for Preparing a Successful Data Access Request, and Understanding Data Security. Please be sure to review any pertinent Data Use Certification Agreements on dbGaP.

21.1.4 4) Link your NIH/eRA and Google identities.

Before you can access *any* controlled-data hosted by the ISB-CGC, you must first associate your Google identity (which you use to sign in to the ISB-CGC Web App and access the Google Cloud) with a valid NIH or eRA account associated with a dbGaP data-access request.

This is accomplished through the ISB-CGC Web App. Follow the directions on the How to link NIH/eRA and Google identities page.

During this process:

- You will first be redirected to an NIH login page, and once you have successfully authenticated, ISB-CGC will store an association between your NIH/eRA identity and your Google identity. (Note that this should be a one-to-one association.)
- Once you have authenticated, ISB-CGC will check which data sets (such as TCGA, TARGET controlled data, etc.) that you have been authorized (by dbGaP) to access. ISB-CGC obtains an updated whitelist for each of the hosted data sets from dbGaP every day. If you have just recently been granted access by dbGaP, there may be a 24 hour delay before you will be able to request access to this data on ISB-CGC.
- Once you have authenticated to NIH via the Web App, and your dbGaP authorization has been verified, the Google identity associated with your account will have access to the controlled data for 24 hours.

These prerequisite steps only need to be done once, unless your accounts become unlinked, you need access to another dbGaP controlled data set, or some other reason.

How to link your NIH/eRA & Google identities

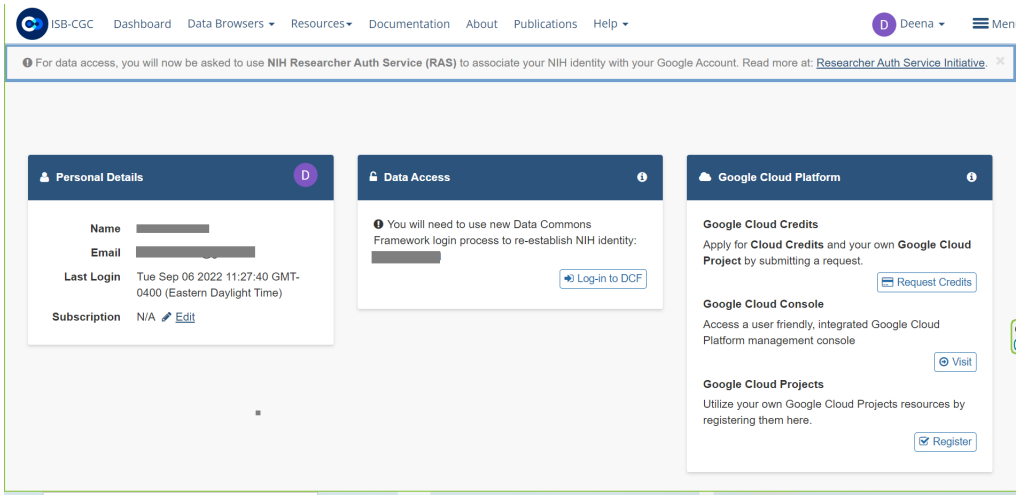
This section shows you how to associate your Google identity to your NIH or eRA identity through the Web App. (Your NIH/eRA identity is the one associated with dbGaP, authorizing you to work with controlled data.) This is a necessary step for gaining access to controlled data. When you are done, you'll be able to access controlled data via your personal user credentials.

At the bottom of this section, there are also instructions for pinning the ISB-CGC controlled access BigQuery project "isb-cgc-cbq" to your Google BigQuery Console.

Some screenshots can be enlarged by clicking on them.

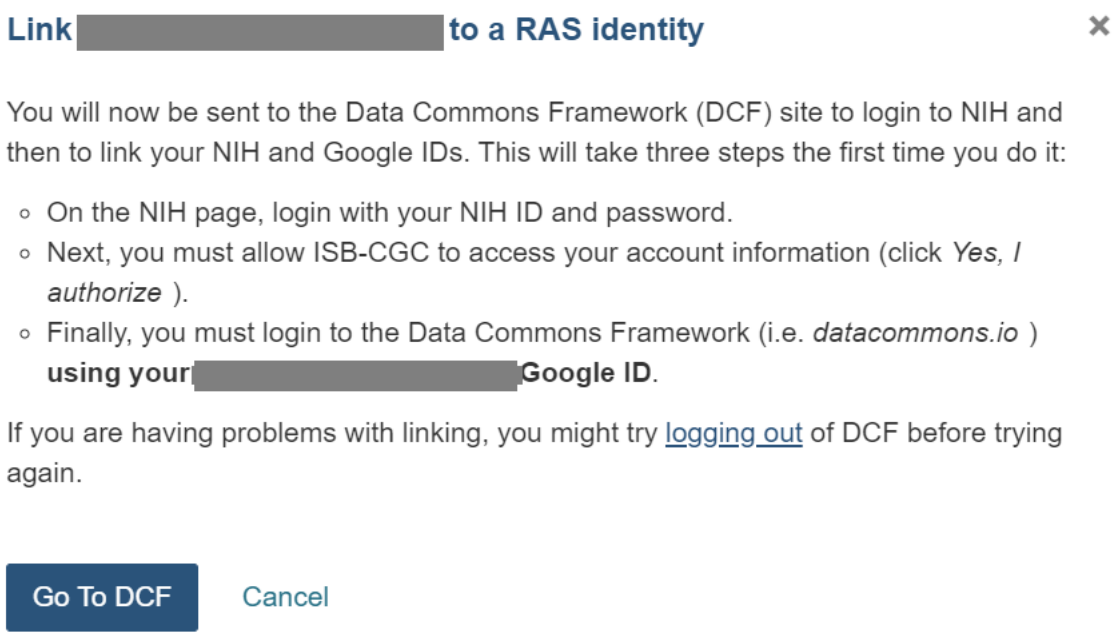
To link your NIH/eRA identity with your Google identity (the Google account you use to login to the ISB-CGC system), select the "persona" icon next to your login name after you have signed in to the ISB-CGC Web App. Or, you can click on the drop down menu next to your name, and click on **Account Details**.

You will then see the following page:

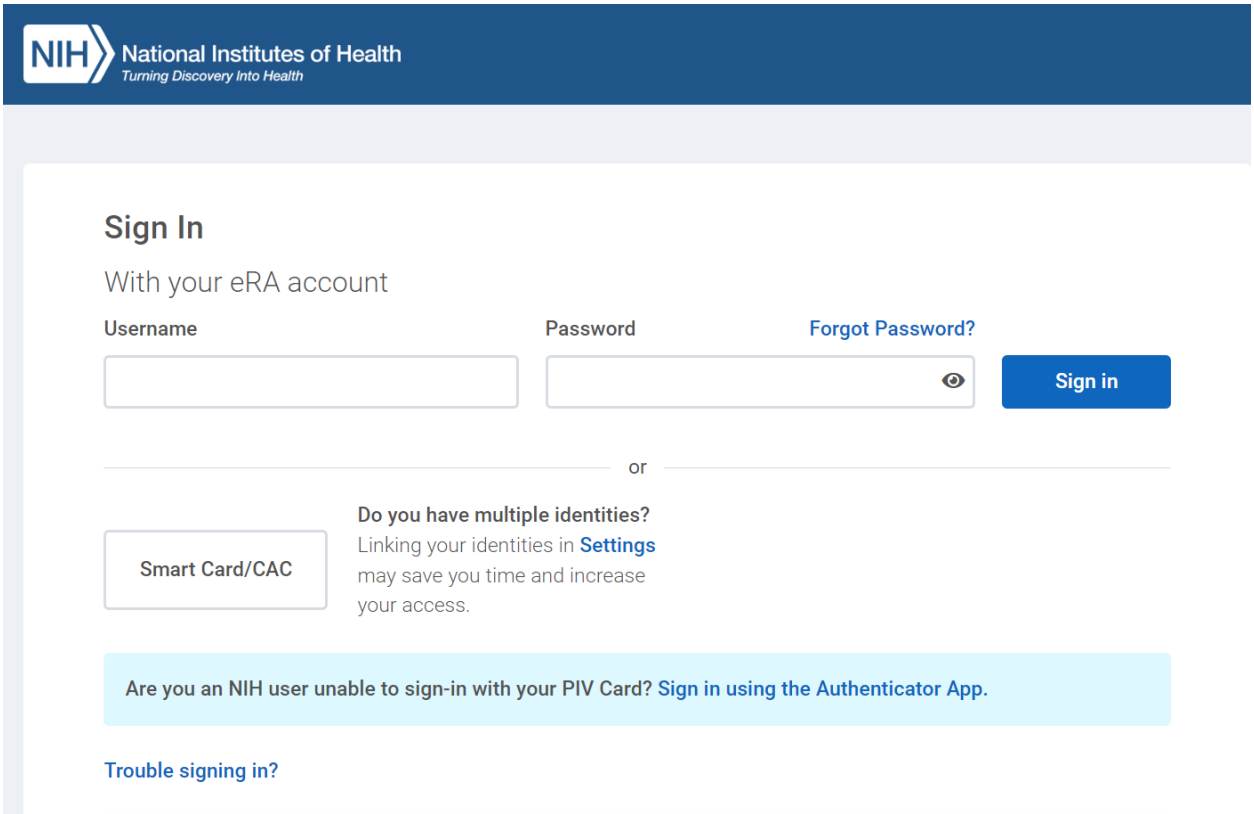


Click the **Log-in to DCF** button.

You will see a pop up describing all the steps needed to link your Google identity (email address) to your NIH identity via the Data Commons Framework (DCF). In this case, 'RAS identity' also refers to your eRA Commons account identity. Click on the **Go to DCF** button.

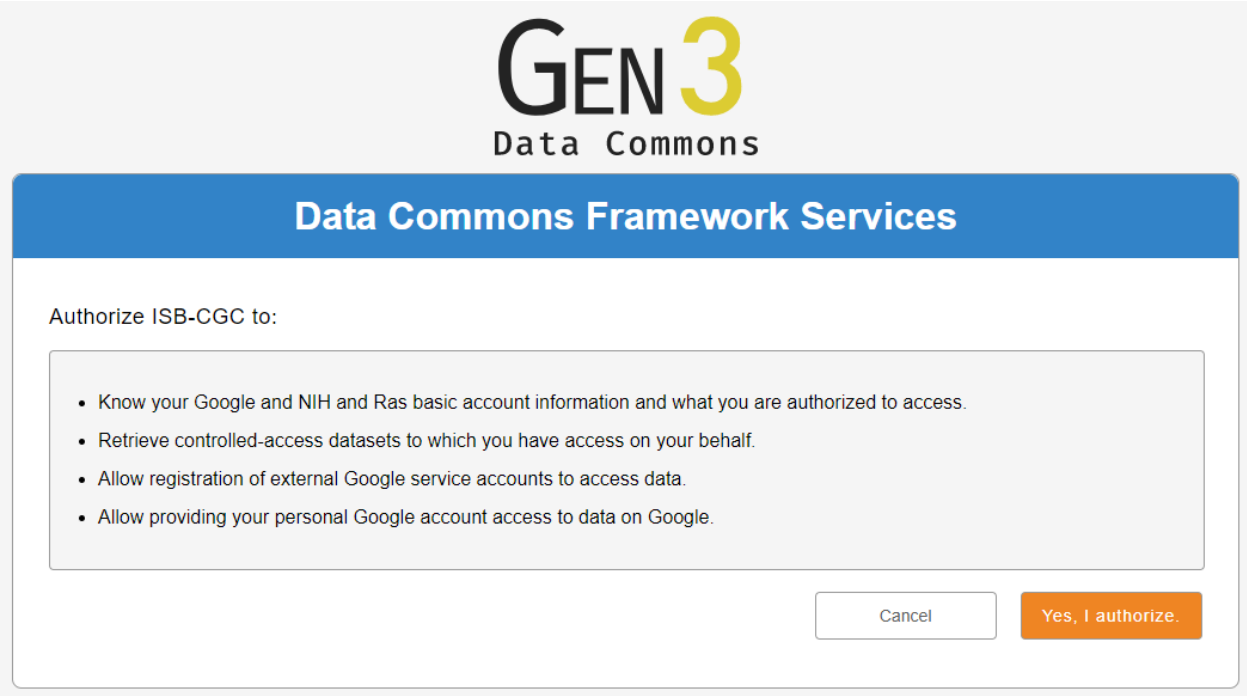


You will then be redirected to an NIH login page, in order to be authenticated by NIH:



If you have an eRA identification, use your eRA Commons username and password to log in. If you have an NIH PIV card, use the Smart Card Login.

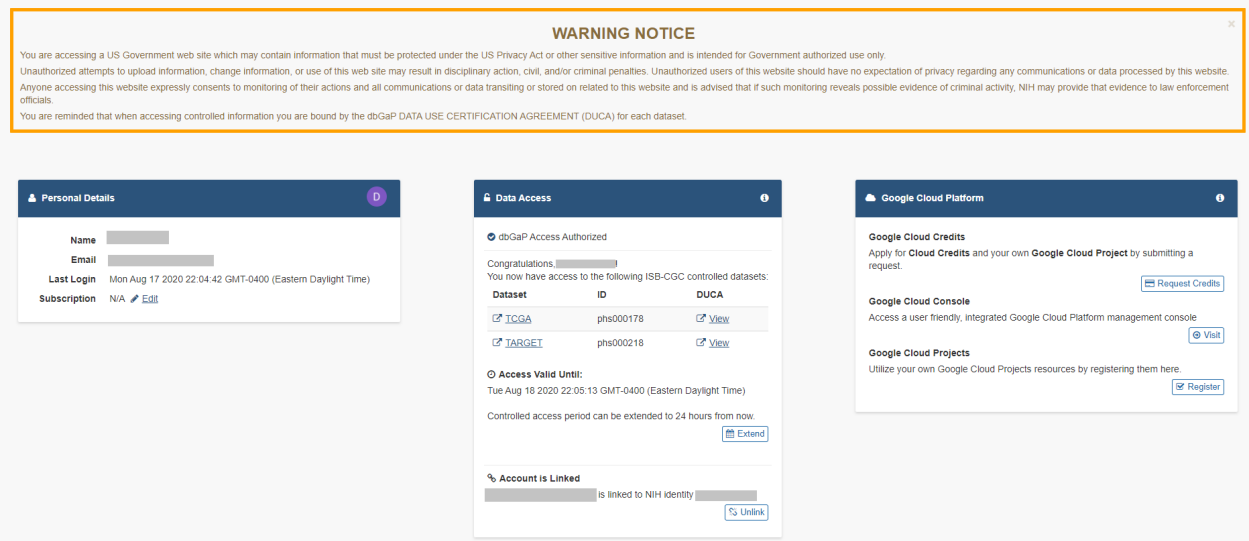
Once you have been authenticated by NIH, and your NIH identity has been verified to be on the current dbGaP whitelist, you will have access to controlled data for 24 hours.



Select the **Yes, I Authorize** button at the bottom right of the page to authorize the Data Commons Framework to associate your Google identity with controlled data.

When presented with a "Sign in with Google" page, select the email that you used when you logged into the ISB-CGC web application.

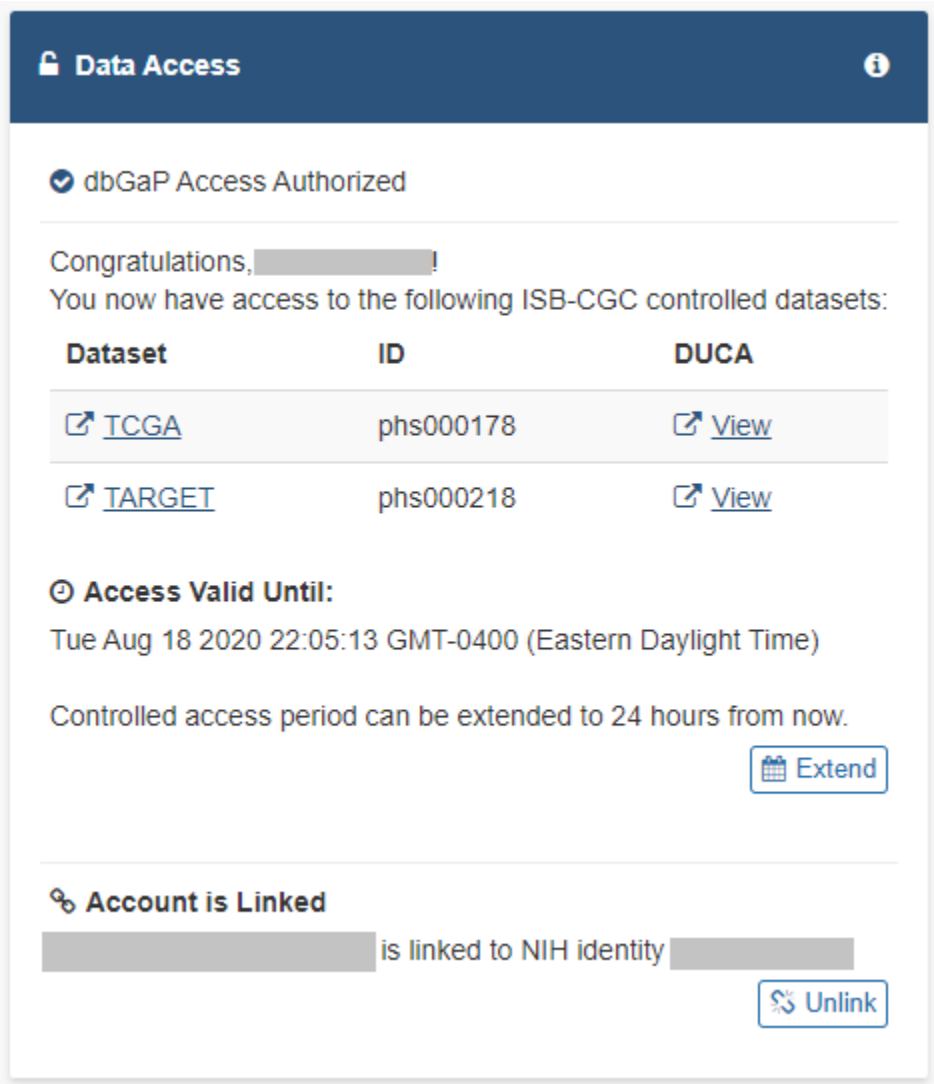
You will be redirected back to the ISB-CGC Web App. A Warning Notice displays, indicating that you must abide by the rules and regulations provided by the DUCA Use Agreement. In the **Data Access** panel, it will indicate "dbGaP Access Authorized".



Note that the ISB-CGC system will enforce a one-to-one relationship between NIH/eRA identities and Google identities. In other words, a single NIH or eRA identity may not be used to gain access to controlled data by multiple, different Google identities.

Unlink Google identity from NIH/eRA identity

If you need to *unlink* your eRA account from your Google account (for example if you want to change which Google identity you use to sign in to the ISB-CGC platform), you may do so by clicking on the **Unlink** button below "<GoogleID> is linked to NIH identity <NIH/eRA Commons ID>".



In the unusual circumstance that your NIH identity has been registered with another Google identity (*eg* with another Google identity that you have), you will see a message, "You tried to link your email address to NIH account <username>, but it is already linked to <Google identity email address>."

If this happens, please sign in with that other Google identity and "unlink" your NIH/eRA account from that identity, as described above. You will then be able to register your NIH/eRA account with the desired Google identity. If you are not able to resolve the issue, contact us at feedback@isb-cgc.org and we will help you resolve it.

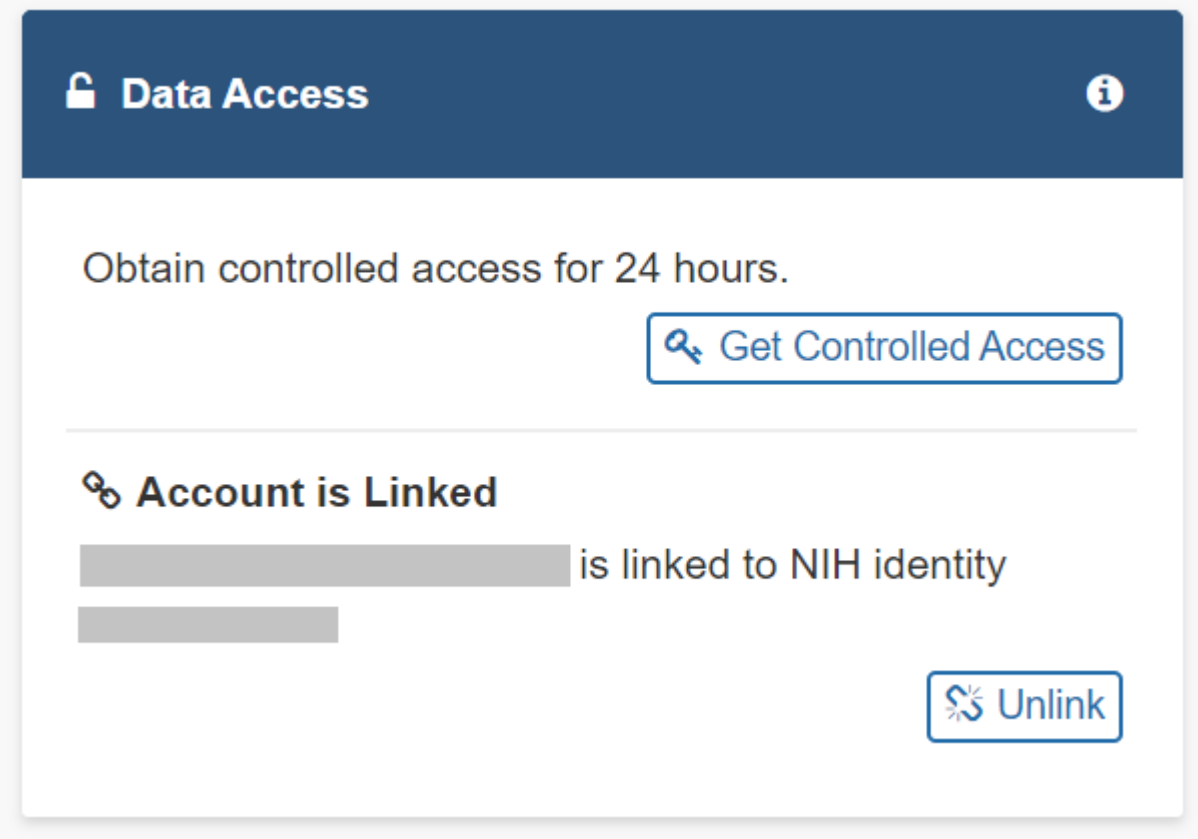
Extending Your Access by 24 hours

If you need to extend your access to controlled data for another 24 hours from now (*eg* if you have a compute job which is using these Google credentials to access controlled data and it is still running), click the **Extend** button on the Data Access panel on the Account Details screen (pictured above). Your access will be extended by 24 hours from the time that you click the button.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

21.2 Controlled Access Via Personal User Credentials

The first time that you perform the above steps, you are automatically granted controlled access via your personal user credentials. This access lasts for 24 hours, though it can be extended. Subsequently, to obtain access, sign into the Web App, click on your persona (or **Account Details** on the drop down menu next to your name). Click the **Get Controlled Access** button below **Obtain controlled access for 24 hours**.



Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CHAPTER

TWENTYTWO

ISB-CGC NOTEBOOKS

22.1 What's a notebook?

Notebooks provide an interface to an interactive analysis environment. They are a mix of code (usually R or Python), descriptive explanations, and visualizations. They're often used to demonstrate an analysis in a step by step fashion. We provide a set of notebooks below as tutorials for several frequently run analyses. You can run these through Jupyter Lab, R Studio, or Google Colaboratory.

22.2 I'm a novice, how do I...

Get started fast?	Python	R
Find GDC file locations?	Python	R
Plot a BigQuery result?	Python	R
Plot a heatmap using data in BigQuery?	Python	R
Work with cloud storage?	Python	
Create cohorts of patients?	Python	R
Use PyPika or dbplyr to build a query?	Python	R
Create a complex cohort?	Python	R
Join multiple tables?	Python	
Get started working with the COSMIC datasets?	Python	
Convert a .bam file to a .fastq file with samtools?	Python	
Find a GA4GH Tool Repository Service (TRS) tool?	Python	
Run workflow execution service (WES) tools?	Python	
Use the ISB-CGC APIs?	Python	R
Explore CPTAC protein abundances?	Python	
Compare protein and gene expression in CPTAC?	Python	

22.3 I'm an advanced user, how do I…

Make a BigQuery table from an NCBI GEO data set?	Python	
Compare cohorts with survival analysis and feature comparison?	Python	R
Run an ANOVA with BigQuery?*	Python	R
Score gene sets in BigQuery?*	Python	R
Correlate gene expression and copy number variation?	Python	
Compute gene-gene expression correlation using BigQuery?	Python	
Create randomized subsets of patients using BigQuery?	Python	R
Convert a 10X scRNA-seq bam file to fastq with dsub?	Python	
Quantify 10X scRNA-seq gene expression with Kallisto and BUStools?	Python	
Compute Nearest Centroid Classification using BigQuery?	Python	R
Analyze data in the COSMIC Cancer Gene Census dataset?	Python	
Use a BigQuery user defined function to perform k-means clustering?	Python	
Compute correlations of protein and gene expression in CPTAC?	Python	
Compare protein expression from different pipelines using CPTAC data?	Python	
Calculate associations between radiomics tumor imaging features and gene expression?	Python	
Analyze the correlation between gene mutations and tumor imaging features?	Python	
Compare gene expression in tumor against gene expression in normal tissue?	Python	
Identify cancer pathways from the Reactome database that are related to a set of genes?	Python	
Integrate the Targetome and Reactome datasets to identify pathways affected by cancer drugs?	Python	

*Notebook inspired by a Query of the Month Blog post

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CHAPTER

TWENTYTHREE

STATISTICAL NOTEBOOKS

Integrated statistical analysis and exploration of multiple genomic and clinical data types provides researchers with a great possibility to expand our current knowledge of cancer. ISB-CGC offers a great source of diverse data types including gene expression, somatic mutations, clinical data, etc. We have developed a series of notebooks that use BigQuery to compute the statistical associations between different combinations of the data types available in ISB-CGC.

23.1 Bioinformatics notebooks

Significant correlations and their p-values using BigQuery	Python	
One-way ANOVA with BigQuery	Python	R
Score gene sets in BigQuery	Python	R
Nearest Centroid Classification using BigQuery	Python	R

23.2 Standard pairwise statistics

The following table lists notebooks that compute associations between pairs of data types available in ISB-CGC. They assess the statistical significance for an association using rank-ordered data and a statistical test appropriate to each data type pair depending on categorical or numerical categorization. The Regulome Explorer inspired notebook is a special notebook that allows computation of associations between all possible data types available in the TCGA dataset; more details are below.

Data type	Data type	Statistical test/notebook
Gene expression	Clinical	Kruskal-Wallis score
Gene expression	Somatic mutation	T-test score
Gene expression	Gene expression	Spearman Correlation
Somatic mutation	Clinical	Chi Square test
Somatic mutation	Somatic Mutation	Fisher's exact test
All types	All types	Regulome Explorer inspired notebook

23.3 Regulome Explorer Inspired Notebook

Regulome Explorer is a well-established web tool for the exploration and visualization of associations between clinical and molecular features of TCGA data. Regulome Explorer was developed in 2012 in close collaboration between the Institute for Systems Biology and the MD Anderson Cancer Center. It enables users to search and visualize precomputed statistical data filtered according to user-specified parameters. Although Regulome Explorer's broad functionality and high-quality graphics make it a valuable tool for exploring and visualizing 20 of the 33 TCGA data sets, it does not yet contain analysis of recent releases of TCGA and cannot be easily applied to data sets other than TCGA.

We developed a more flexible version, replicating capabilities of Regulome Explorer, as a Python notebook that uses Google Cloud resources. Rather than working with precomputed, fixed cohorts and fixed results, statistical analyses are dynamically performed in the cloud, with user defined patient cohorts. Moreover, the notebook can be extended so that users can analyze additional data sets available as part of the 'ISB-CGC BigQuery ecosystem' such as TCGA, TARGET, CCLE, COSMIC, and others. The notebook can be accessed in Regulome Explorer inspired notebook.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CHAPTER

TWENTYFOUR

MACHINE LEARNING NOTEBOOKS

Machine learning methods have enabled researchers to leverage and integrate the vast amounts of diverse cancer data to reveal new insights, develop better diagnostics, and improve therapy. ISB-CGC offers examples of how to use Google Cloud resources to train and use machine learning models for a variety of cancer applications and datasets.

How to build an RNA-seq logistic regression classifier	Python	R
How to build an RNA-seq logistic regression classifier with BigQuery ML	Python	
How to perform nearest centroid classification using BigQuery	Python	R
How to predict cancer survival with BigQuery ML	Python	
How to predict cancer survival with TensorFlow	Python	

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CHAPTER

TWENTYFIVE

HTAN NOTEBOOKS

HTAN is a National Cancer Institute (NCI)-funded Cancer Moonshot initiative to construct 3-dimensional atlases of the dynamic cellular, morphological, and molecular features of human cancers as they evolve from precancerous lesions to advanced disease (Cell April 2020).

Investigating HTAN scRNA-seq with BigQuery	Python	
Explore HTAN single cell RNA seq data	Python	
Explore HTAN Clinicial, Biospecimen, and Assay Metadata	Python	R

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CHAPTER

TWENTYSIX

TUTORIALS AND HOW-TO GUIDES

The links on this page connect to How-To guides, workshop materials, examples and other helpful tutorials. We encourage the community to provide feedback on these examples and also to add your own examples to enrich this public resource! Contact us at feedback@isb-cgc.org.

26.1 Video Tutorials

See the following page for ISB-CGC produced videos giving helpful tours through ISB-CGC:

- ISB-CGC Video Tutorials

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

26.2 Query of the Month

Welcome to the 'Query of the Month' where we'll be creating a collection of new and interesting queries to demonstrate the powerful combination of BigData from the NCI cancer programs like TCGA, and BigQuery from Google.

NOTE! We mostly spend time producing notebooks for our community collection. Check it out: https://github.com/isb-cgc/Community-Notebooks ReadTheDocs: https://isb-cancer-genomics-cloud.readthedocs.io/en/latest/sections/HowTos.html

Query of the Month is produced by the ISB-CGC team, with special effort by:

- David L Gibbs (david.gibbs (~ at ~) systemsbiology (~ dot ~) org)
- Kawther Abdilleh (kawther.abdilleh (~ at ~) gdit (~ dot ~) com)
- Sheila M Reynolds (sheila.reynolds (~ at ~) systemsbiology (~ dot ~) org)

26.2.1 Table of Contents

2019

- *July2019*: New notebooks added, cohorts and GEO data
- *June2019*: Community Notebooks launched!
- *February2019*: BigQuery in R - a refresher
- *January2019*: Bam slicing in a cloud hosted python notebook.

2018

- *December2018*: BigQuery Tips & Tricks
- *November2018*: Transform VCF (DNA variants) files to BigQuery.
- *October2018*: Jupyter notebooks & Dataproc clusters … in the cloud.
- *September2018*: R scripts in the cloud.
- *August2018*: Using BigQuery ML in a shiny app.
- *July2018*: First look: BigQuery ML.
- *June2018*: Processing bam files using WDL 'scatter and gather'.
- *May2018*: Processing bam files using CWL 'scatter and gather'.
- *April2018*: Running CWL workflows in the cloud.
- *March2018*: Machine learning classifer in BigQuery?! Top Scoring Pairs implementation.
- *February2018*: BioCircos shiny app, showing pairwise correlations within a pathway.
- *January2018*: Gene Set Scoring in BigQuery, using the new hg38 mutation tables.

2017

- *December2017*: BigQuery comparing TCGA samples to GTEx tissues with Spearman correlation.
- *November2017*: Run an R (or python) script in batch mode using dsub on the google cloud.
- *October2017*: Using plotly for visualization in Shiny apps. We implement an interatictive heatmap using heatmaply
- *September2017*: We implement a new statistical test in BigQuery: the one-way ANOVA.
- *August2017*: A small demo application using BigQuery as the backend for a Shiny app.
- *July2017*: Look at the BigQuery RECORD data type in methylation tables from the GDC.
- *May2017*: Continued from April: estimating the distance between samples based on shared mutations in pathways.
- *April2017*: BigQuery compute a similarity metric on overlapping mutations between samples. Uses MC3 mutation table and data from COSMIC.
- *March2017*: BigQuery to compute a pairwise distance matrix and a heatmap in R
- *February2017*: Using BigQuery, define K-means clustering as a user defined (javascript) function
- *January2017*: Comparing Standard SQL and Legacy SQL.

2016

- *December2016*: Spearman correlation in BigQuery to compare the new hg38 expression data to the hg19 data

Links

Resources: Helpful information!

July, 2019

This month we've added a couple new notebooks. Let us know if they're useful!

We have a notebook demonstrating how to create sample cohorts based on clinical characteristics. Second, a notebook demonstrating how to get a dataset from NCBI GEO, transform it to a 'tidy' format and make a BigQuery table.

How do I create cohorts of patients (Python)?

How do I make an NCBI GEO BigQuery table (Python)?

June, 2019

#*Community Notebooks launched!*

Community Notebooks

After a lengthy hiatus, we've returned with a new Community Notebooks repository! This is a central hub, where very specific questions (like an FAQ) can be answered using a notebook format. This should make it easier to move from example to application.

Our first few notebooks include:

How do I get started fast (Python)?

How do I get started fast (R)?

How do I plot a BigQuery result (Python)?

How do I plot a BigQuery result (R)?

How do I work with cloud storage (Python)?

Stay tuned for more notebooks! And, we would love to hear from you! Have a notebooks you'd like to share with world, *and attribution* (!) , please send an email to dgibbs <at> systemsbiology <dot> org.

February, 2019

In this QoTM, we'll look at the current open-access TCGA and TARGET datasets in BigQuery using R as our workspace. You can find in-depth descriptions of the BigQuery datasets in our documentation here:.

Note: You will need to have set up a Google Cloud Platform project to access/query the BigQuery tables in R, more info here:.

Here's our example case-study for month:

Your lab is interested in comparing the gene expression profiles of 2 non-coding RNAs, specifically the lncRNA ANRIL and the miRNA miRNA-21 across multiple cancer types. From the literature and your own research studies, expression of both ANRIL and. miRNA-21 have been implicated in many different cancer types.

In R:

- Let's look at the currently available ISB-CGC BigQuery tables to find tables that will allow us to compare the expression profiles of the non-coding RNAs.
- Once identified, let's query and explore the BigQuery tables that contain the information we need.
- Let's generate an interactive scatterplot!

Let's begin!

Within your R environment

```
install.packages("bigrquery")
install.packages("httpuv")
install.packages("ggplot2")
install.packages("reshape")
install.packages("scales")
install.packages("dplyr")


library(bigrquery)
library(httpuv)
library(ggplot2)
library(reshape)
library(scales)
library(dplyr)

##Let's investigate the publicly available ISB-CGC created TCGA and TARGET tables.
##There are a number of Level 3 datasets updated and available including:


#To list datasets associated with the ISB-CGC project in R:
list_datasets("isb-cgc")
[1] "CCLE_bioclin_v0"     "GDC_metadata"        "GTEx_v7"             "QotM"             ␣
↪  "TARGET_bioclin_v0"
[6] "TARGET_hg38_data_v0" "TCGA_bioclin_v0"     "TCGA_hg19_data_v0"   "TCGA_hg38_data_v0
↪"   "Toil_recompute"
[11] "ccle_201602_alpha"   "genome_reference"    "hg19_data_previews"  "hg38_data_
↪previews"  "metadata"
[16] "platform_reference"  "tcga_201607_beta"    "tcga_cohorts"        "tcga_seq_metadata
↪"

#Let's get the list of tables for the TCGA_hg38 dataset:
 list_tables("isb-cgc","TCGA_hg38_data_v0")
 [1] "Copy_Number_Segment_Masked"     "Copy_Number_Segment_Masked_r14" "DNA_Methylation"
 [4] "DNA_Methylation_chr1"           "DNA_Methylation_chr10"          "DNA_Methylation_
↪chr11"
 [7] "DNA_Methylation_chr12"          "DNA_Methylation_chr13"          "DNA_Methylation_
↪chr14"
```

(continues on next page)

(continued from previous page)

```
 [10] "DNA_Methylation_chr15"          "DNA_Methylation_chr16"          "DNA_Methylation_
↪chr17"
 [13] "DNA_Methylation_chr18"          "DNA_Methylation_chr19"          "DNA_Methylation_
↪chr2"
 [16] "DNA_Methylation_chr20"          "DNA_Methylation_chr21"          "DNA_Methylation_
↪chr22"
 [19] "DNA_Methylation_chr3"           "DNA_Methylation_chr4"           "DNA_Methylation_
↪chr5"
 [22] "DNA_Methylation_chr6"           "DNA_Methylation_chr7"           "DNA_Methylation_
↪chr8"
 [25] "DNA_Methylation_chr9"           "DNA_Methylation_chrX"           "DNA_Methylation_
↪chrY"
 [28] "Protein_Expression"             "RNAseq_Gene_Expression"         "Somatic_Mutation
↪"
 [31] "Somatic_Mutation_DR10"          "Somatic_Mutation_DR6"           "Somatic_
↪Mutation_DR7"
 [34] "miRNAseq_Expression"            "miRNAseq_Isoform_Expression"    "tcga_metadata_
↪data_hg38_220818"
 [37] "tcga_metadata_data_hg38_250718"


 #So let's remember our use-case: We want to compare the gene expression profiles of our␣
↪non-coding RNAs
 #of interest. ANRIL is a lncRNA and lncRNA expression is captured in the RNAseq_Gene_
↪Expression table and miRNA-21 is
 #a miRNA_seq_Expression table. Let's find get our RNAs expression from their respective␣
↪tables

  #To access and query the BigQuery tables, you'll need to first specify your project id:
  project <-"isb-cgc-02-0001"

  #Information about lnRNAs are in the gene expression tables..Let's compose a query on␣
↪the RNAseq_Gene_Expression dataset
  #for the lncRNA dataset

  #query the RNAseq BigQuery table that has info for lncRNAs
  sql1<-"SELECT case_barcode, project_short_name, gene_name,HTSeq__Counts FROM `isb-
  cgc.TCGA_hg38_data_v0.RNAseq_Gene_Expression` WHERE Ensembl_gene_id = 'ENSG00000240498
↪' ORDER BY
  project_short_name"
  data1 <- query_exec(sql1, project = project, use_legacy_sql = FALSE,max_pages = Inf)
  head(data1)

  #query the BigQuery miRNA expression table
  sql2 <-"SELECT project_short_name, case_barcode, mirna_id,read_count FROM `isb-cgc.
↪TCGA_hg38_data_v0.miRNAseq_Expression`     WHERE mirna_id = 'hsa-mir-21' ORDER BY␣
↪project_short_name"
  data2 <- query_exec(sql2, project = project, use_legacy_sql = FALSE,max_pages = Inf)
  head(data2)

  #we can join these tables in BigQuery and have one datatable, we can also create 2␣
↪tables and merge them right here in R.
```

(continues on next page)

(continued from previous page)

```
#let's merge these two data tables here in R by case_barcode.
merge_data = merge(data1,data2,by=c("case_barcode","project_short_name"))

#The expression values were computed using different tools for the different datasets,␣
↪so to compare the values we can        #normalize using the rescale function in R.

merge_data$HTSeq__Counts= rescale(merge_data$HTSeq__Counts,to=c(0,1))
merge_data$read_count = rescale(merge_data$read_count,to=c(0,1))

#Let's take a look at our merged table, we now have information for expression of the␣
↪CDKN2B-AS1 and hsa-mir-21 for          #almost 12,000 cases.
head(merge_data)

#We can create an interactive scatterplot plot using the function plotly in R to␣
↪compare CDKN2B-AS1 and miRNA-21             #expression across multiple cancer types.␣
↪This interactive plopt in R is called plotly.

p <- ggplot(merge_data, aes((HTSeq__Counts), (read_count), colour=project_short_name))␣
↪+ geom_point() + theme_classic() +     theme(legend.position="none") + labs(x = "ANRIL␣
↪normalized expression",y="miRNA-21 normalized expression")
ggplotly(p)
```

January, 2019

Bam slicing in the cloud

This month we're going to do some bam slicing in a cloud hosted jupyter notebook.

Regarding bam slicing, ISB-CGC documentation can be found here.

BAM files are central to almost all genomic analyses. Often, they are very large in size, especially for larger genomes like the human genome. Researchers may only be interested in small genomic regions, and so rather than download and deal with massive files, we can extract or "slice out" subsections of the BAM file. The high performance HTSlib library (release 1.4+) is used to manipulate high-throughput genomics data and is what allows users to perform BAM-slicing. HTSlib is central to the SAMtools package, a popular tool for NGS data manipulation http://www.htslib.org/doc/samtools.html .

In this post, we'll be using a python wrapper for SAMtools called PySAM.

In the Jupyter notebook (see link below), we demonstrate the following:

- How to invoke bash commands within a Jupyter environment.
- How to install packages/programs within a Jupyter environment
- How to use available BigQuery tables within ISB-CGC to query and identify Google Cloud Storage bucket locations for BAM files of interest
- How to use PySam to slice BAM files
- How to save slices in your bucket and retrieve them
- Brief example of working with reads

Link to the Jupyter notebook here .

How to invoke bash commands within a Jupyter environment.

We're finding the free jupyter notebooks offered from Google Colaboratory really useful and surprisingly flexible. The level of access to the operating system is quite good and allows us to run bash commands to work on the file level.

To run a command, in your colaboratory notebook, create a cell like:

```
!ls -lha
```

That's going to list out all the files in your environment. The 'bang' (exclaimation point) signals to the notebook to run this command using bash.

Another useful command is

```
!env
```

Which prints out all the environment variables. This is useful in a lot of cases, such as compiling software.

How to install packages/programs within a Jupyter environment

To install a new linux library, create a new cell in your colaboratory python notebook and run:

```
!sudo apt-get install libxml2
```

How to use available BigQuery tables

In the notebook, there's other 'magic commands' as well. Since it's a Google product, it's relatively straightforward to connect to other Google products like BigQuery. This is really cool, because if the free colaboratory notebook is short on raw compute power, BigQuery gives you the power of a cluster, and with the summary results you can do visualization and downstream analysis in the notebook.

To run an SQL query, we use some %%BigQuery magic

```
%%bigquery --project our-project-id df
SELECT * FROM `isb-cgc.TCGA_hg19_data_v0.tcga_metadata_data_hg19_18jul`
where
data_format = 'BAM'
AND disease_code = 'OV'
AND experimental_strategy = "WGS"
AND platform = 'ABI SOLiD'
LIMIT 5
```

Where 'our-project-id' is your google project id, and df is a variable that will store the results of the query. In this query, we're selecting all the available metadata for bam files (data_format = 'BAM') associated with ovarian cancer (disease_code = 'OV'). From a query like this, you can get a list of bam files stored in a google cloud bucket, and slice out a section of reads from each.

Using Pysam to slice bams

Pysam is a python wrapper around SAMtools which uses the HTSlib in reading and processing bams.

In order to read out from GCS (cloud buckets), HTSlib needs to be compiled with some additional functionality.

```
export HTSLIB_CONFIGURE_OPTIONS="--enable-gcs"
```

Then, to slice out a region on chromosome 7 between 140453130-140453140, we would:

```
export GCS_OAUTH_TOKEN=`gcloud auth application-default print-access-token`

./samtools view gs://gdc-ccle-open/0a109993-2d5b-4251-bcab-9da4a611f2b1/C836.Calu-3.2.
↪bam 7:140453130-140453140
```

In the python notebook, we do something very similar. We need to compile HTSlib gcs-enabled to read cloud-based files, which in turn requires installing a few extra libraries. Then we would:

```
samfile = pysam.AlignmentFile('gs://gdc-ccle-open/0a109993-2d5b-4251-bcab-9da4a611f2b1/
↪C836.Calu-3.2.bam', "rb")

for read in samfile.fetch('7', 140453130, 140453135):
  print(read)

samfile.close()
```

The AlignmentFile lets you fetch a AlignedSegment object, and you use that object to call many different methods. You can see the full API here.

For a couple quick examples of working with AlignedSegments, such as processing sequence data, check out the notebook!

How to save slices in your bucket and retrieve them

Once you have a list of slices for analysis, we can use all our Google cloud tools, like gsutil!

```
!gsutil ls gs://my_bucket_1/
```

Here we list out all the file in my bucket. We can also use gsutil to move items in and out of our notebook environment.

```
!gsutil cp my.bam gs://my_bucket_1/my.bam
```

That's it for this month, please let us know if you have questions, or have topics you'd like to see covered in later months!

December, 2018

BigQuery Tips & Tricks

Last month, we transformed a typical genomics file type (the vcf file format) into a BigQuery table. This month, we'll continue exploring how to load data into bigquery tables. Because genomics files are often very large in size, we'll also explore some tricks on how to partition tables to query to save both money and time!

I. Loading CSV Data into BigQuery

First, let's explore how to load csv files into big query. Here's an example of a common use-case:

A lab with which your group collaborates is generating large amounts of RNAseq gene expression data. They have been following the nice analyses your group has done using BigQuery and would like you to help them perform similar analyses. Thus far, they have saved all of their RNAseq expression data into CSV format. They need your help first loading their RNAseq data files into BigQuery.

Here, we'll learn how to load CSV files into BigQuery tables. We'll accomplish this with some very useful bq command-line tools and arguments.

Before starting:

– This tutorial assumes that you've already created a GCP project. If you don't already have one, instructions on how to set up one up can be found: here

– Ensure that you have Google Cloud SDK installed

Loading CSV files into Google Cloud Storage

You can load local files as well as files from Google Cloud Storage (GCS). For this exercise, let's make a GCS bucket to store our CSV files.

1. Make a bucket to store the RNAseq CSV

```
gsutil mb gs://RNAseq_CSVs
```

2. Copy the CSVs into your newly created storage bucket

```
gsutil cp *.csv gs://RNAseq_CSVs
```

Creating a BigQuery dataset

Creating tables and loading data via the BigQuery web-UI is good if you're only loading a small amount of data. It can be a tediously manual process though if you have more than a handful of files. We can create tables and load data using Google SDK's handy bq tool. bq is a python-based, command-line tool for BigQuery. https://cloud.google.com/bigquery/docs/bq-command-line-tool

Let's create a dataset that will hold our RNAseq data:

```
bq mk RNAseq_data
```

If successful, you will get the following message:

```
Dataset 'Your_Project_ID:RNAseq_data' successfully created
```

You can also list all of the datasets associated with your project using the following command:

```
bq ls
```

Generate schema for BigQuery table

A schema for a table is a description of its field names and types. BigQuery allows you to specify a table's schema when you load data into a table. We can create a schema file in JSON format. You can find a Python script (**createSchema.py**) to create a JSON schema for your table in our github examples-Python repository.

https://github.com/isb-cgc/examples-Python/tree/master/python

Usage: python createSchema.py <input-filename> <nSkip>

where nSkip specifies the # of lines skipped between lines that are parsed and checked for data-types; if the input file is small, you can leave set nSkip to be small, but if the input file is very large, nSkip should probably be 1000 or more (default value is 1000)

Loading Data in BigQuery

With the JSON schema file, we are now ready to load data into BigQuery. The bq load command is used to load data in BigQuery via the command-line.

```
# usage:
# bq --location=[LOCATION] load --source_format=[FORMAT] [DATASET].[TABLE] [PATH_TO_
↪SOURCE] [SCHEMA]

  bq load \\

  --source_format=CSV \\

  --skip_leading_rows=1 \\

  RNAseq_data.expressionFile \\                  # where it's going
```

(continues on next page)

(continued from previous page)

```
  gs://RNAseq_CSVs/ExpressionDataTable.csv \\   # the table in a bucket

  ExpressionDataTable.csv.json                  # the table schema
```

You can verify that the table loaded by showing the table properties with this command:

```
bq show RNAseq_data.expressionFile
```

II. BigQuery Table Clusters

The costs of using BigQuery center around how much of a table is read by the query. So, the same query applied to a small table versus a very large table will incur very different costs. It simply costs more to query a large table! In the past, we broke tables into many subtables to save costs and time. This was the case with the methylation tables where the entire thing consisted of 3.9 Billion rows (932 GB)! It's pretty expensive to query that table, so we broke it into many tables by chromosome. OK, but not entirely convenient to work with.

Now, there's a fairly simple step to accomplish the same thing, resulting in huge cost savings without changing your SQL or table schema! They're called 'clustered tables', which groups rows of your BigQuery table so that your query only reads the appropriate portions of your table. This means you can specify the cluster to be over chromosomes, and your query will only read the portion of the table associated with that chromosome. docs here

There's a number of different ways to partition your tables. For one, you can partition it at the time of 'ingestion'. What that means is that each time new data arrives, a new partition is created when the data is appended to a new table.

So let's look at an example using a table built from wikipedia (from Optimizing BigQuery: Cluster your tables, by *Felipe Hoffa*). This uses a query to select-*-everything from the table, and cluster it by wiki and title. The order matters in clusters (see notes below)! Clustered tables also have to be applied to partitioned tables. Below the table is being partitioned by a date.

```
CREATE TABLE `fh-bigquery.wikipedia_v3.pageviews_2017`

PARTITION BY DATE(datehour)

CLUSTER BY wiki, title

OPTIONS(
description="Wikipedia pageviews - partitioned by day,clustered by (wiki, title).
  Contact `*https://twitter.com/felipehoffa* <https://twitter.com/felipehoffa>`__",
  require_partition_filter=true)
AS SELECT * FROM `fh-bigquery.wikipedia_v2.pageviews_2017`
WHERE datehour > '1990-01-01' # nag
```

Now, *Felipe* notes:

- **CLUSTER BY wiki, title: Whenever people query using the wiki** column, BigQuery will optimize these queries. These queries will be optimized even further if the user also filters by title. If the user only filters by title, clustering won't work, as the order is important (think boxes inside boxes).
- **require_partition_filter=true: This option reminds my users to** always add a date filtering clause to their queries. That's how I remind them that their queries could be cheaper if they only query through a fraction of the year.

To use a clustered table, just GROUP BY on the clustered columns, then it's done automatically. Most often, you'll see a reduction in the amount of data read, but you can also see where the runtime is reduced, even if the amount of data read is the same.

```
SELECT wiki, title, SUM(views) views
FROM `fh-bigquery.wikipedia_v3.pageviews_2017`
WHERE DATE(datehour) BETWEEN '2017-06-01' AND '2017-06-30'
GROUP BY wiki, title
ORDER BY views DESC
LIMIT 10
```

without clustering

64.8s elapsed, 180 GB processed

with clustering

22.1 elapsed, 180 GB processed

So, in genomics data, this is an excellent technique to apply, and some experimentation might be necessary to find the best clustering schema for your work.

Let's try this on the 1000 genomes table from last month. That was a table of genomic data, produced from a VCF file from the Wellcome Trust 1000 Genomes project.

Earlier I had written a query to flatten the VCF table, we'll use that to partition, since some of the columns we'd like to use for partitioning and clustering are nested fields, which are incompatible. I saved that flat file to a new table 'flat1000genomes' with 2.7 *Billion* rows.

Partitioning tables (right now) only works with DATEs. So to get around that, we'll create a 'fake date'

see here.

```
CREATE TABLE
  `isb-cgc-02-0001.Daves_working_area.Clustered1000genomes`
PARTITION BY fake_date
CLUSTER BY chr, name
OPTIONS(
  description="1000 genomes partitioned by chr, cluster by call.name",
  require_partition_filter=true)
AS SELECT *, DATE('2018-12-14') fake_date FROM
  `isb-cgc-02-0001.Daves_working_area.flat1000genomes`
```

So here's a query that counts up variants within samples.

```
SELECT chr, name, alt1, COUNT( alt1 ) AS n
FROM
`isb-cgc-02-0001.Daves_working_area.flat1000genomes`
GROUP BY chr,name,alt1
ORDER BY n ASC
```

Query complete (8.6s elapsed, 40.8 GB processed)

```
SELECT chr, name, alt2, COUNT( alt2 ) AS n
FROM `isb-cgc-02-0001.Daves_working_area.Clustered1000genomes`
WHERE fake_date is not NULL
GROUP BY chr, name, alt2
ORDER BY n ASC
LIMIT 10
```

Query complete (3.6s elapsed, 61.2 GB processed)

That's more than 58% less time on ~50% more data!

November, 2018

Transforming VCF (DNA variants) files to BigQuery.

tldr;

Variant calls, as organized in vcf files, are central to almost all genomics and bioinformatics analyses. As genomics datasets continue to become larger in both size and complexity, as researchers we are often faced with the scenario of having to gain biological insights from hundreds and sometimes even thousands of VCF files at once. Google Genomics has developed a tool for transforming and processing VCF files in a scalable manner based on *Apache Beam* using *Dataflow* on the Google Cloud Platform. Using this transform pipeline, one can load hundreds of thousands of VCF files with millions of samples and billions of records into BigQuery.

In the near future, ISB-CGC will make available controlled-access VCFs, transformed into BigQuery tables, allowing users with approved authorization to harness the power of BigQuery to conduct powerful variant analyses. This month, we will explore variant analysis in BigQuery. We transform a VCF into a BigQuery table and perform queries to gain biological insights into variant data.

Here's some helpful links:

Loading and transforming VCF files into BigQuery,

Understanding the BigQuery Variants Schema,

Variant Transforms github,

Analyzing Variants in BigQuery

To start, you'll first need to configure your Cloud environment:

- A GCP project with billing
- Enable the *Cloud Genomics, Compute Engine, Cloud Storage, and Cloud Dataflow APIs*
- An existing *BigQuery dataset* and a *Cloud Storage bucket*.

Moving data into your GCS bucket:

If we have web addresses to the VCF files, we can use the cloud console (or gcloud) to transfer files directly to our GCS bucket.

For this exercise, we use a vcf file of chromosome 21 from the public 1000 genomes project already in GCS into a bigquery table: gs://genomics-public-data/1000-genomes-phase-3/vcf

VCF to BigQuery Transform:

The easiest way to run the VCF to BigQuery pipeline is to use the *docker* image and run it with the *Google Genomics Pipelines API* as it has the binaries and all dependencies pre-installed.

Run the script below and replace the following parameters:

- GOOGLE_CLOUD_PROJECT: This is your project ID that contains the BigQuery dataset.
- INPUT_PATTERN: A location in Google Cloud Storage where the VCF file are stored. You may specify a single file or provide a pattern to load multiple files at once. Please refer to the *Variant Merging* documentation if you want to merge samples across files. The pipeline supports gzip, bzip, and uncompressed VCF formats. However, it runs slower for compressed files as they cannot be sharded.
- OUTPUT_TABLE: The full path to a BigQuery table to store the output.
- TEMP_LOCATION: This can be any folder in Google Cloud Storage that your project has write access to. It's used to store temporary files and logs from the pipeline.

```
GOOGLE_CLOUD_PROJECT=your_project_id

INPUT_PATTERN=gs://Path_to_your_vcf_file

OUTPUT_TABLE=your_project_id:bq_dataset.bqtable

TEMP_LOCATION=gs://path_to_a_temp_folder

COMMAND="/opt/gcp_variant_transforms/bin/vcf_to_bq
   --infer_undefined_headers --allow_incompatible_records \\

   --project ${GOOGLE_CLOUD_PROJECT} \\

   --input_pattern ${INPUT_PATTERN} \\

   --output_table ${OUTPUT_TABLE} \\

   --temp_location ${TEMP_LOCATION} \\

   --job_name vcf-to-bigquery \\

   --runner DataflowRunner"

gcloud alpha genomics pipelines run \\

   --project "${GOOGLE_CLOUD_PROJECT}" \\

   --logging "${TEMP_LOCATION}/runner_logs_$(date+%Y%m%d_%H%M%S).log" \\

   --service-account-scopes https://www.googleapis.com/auth/cloud-platform \\**

   --zones us-central1-f \\

   --docker-image  gcr.io/gcp-variant-transforms/gcp-variant-transforms \\**

   --command-line "${COMMAND}"
```

Note the operation ID returned by the above script. You can track the status of your operation by running:

```
gcloud alpha genomics operations describe <operation-id>
```

The resulting transformed vcf table looks something like this:

Row	reference_name	start_position	end_position	reference_bases	alternate_bases.alt	alternate_bases.AC	alternate_bases.AF	names	quality
1	21	34466727	34466728	G	A	2	0.000394477		1.4384e-06
					C	4	0.000788955		

What are all those empty cells? Well, when you upload a vcf file, the schema automatically defines a lot of those fields

as type:'Record' that have mode:'Repeated'. This is compared to more common data types like 'Integer' or 'String'. A 'Record' or 'Struct' is a data structure that brings related items together as a list or a nested set of lists. The example given in the Google documentation is:

"In BigQuery, you can preserve the relationship between book and author without creating a separate author table. Instead, you create an author column, and you nest fields within it such as the author's first name, last name, date of birth, and so on. If a book has multiple authors, you can make the nested author column repeated." *https://cloud.google.com/bigquery/docs/nested-repeated*

In our case, we have a single genomic position, and within that position we can list different alternate variants. In the example, the reference G is replaced by either an A or a C, and you can see this list in the column names 'alternate_bases.alt', 'alternate_bases.AC', etc.

***Unnesting BigQuery tables to query repeated fields ***

Querying multiple independently repeated fields or calculating the cross product of such fields requires "flattening" of the BigQuery records. You may have seen error messages like "Cannot query the cross product of repeated fields …" from BigQuery in such scenarios. *Google Genomics* describes the workarounds for enabling such queries and exporting a flattened BigQuery table that can be directly used in tools that required a flattened table structure (e.g. for easier data visualization). BigQuery supports fields of type *ARRAY* for lists of values and fields of type *STRUCT* for hierarchical values. These field types are useful for representing rich data without duplication.

These type of tables can be pretty tricky, so…

***Let's dive in with some examples! ***

Colaboratory is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud! Colaboratory is a great way to work on analysis projects with a group. A great feature is that BigQuery takes all the heavy-duty compute to the cloud, and lets the notebook be used for documentation and visualization. We've provided a notebook with all code in shared Colab notebook here:

Just as with all Google Cloud Platform (GCP) products, to work with BigQuery in the Colaboratory notebook you must first set up a GCP Project. If you're starting up a new account you can get $300 in credit, plus there's a sizable amount of querying that's free every month. Detailed instructions on how to set up a GCP project can be found here in our documentation:

https://isb-cancer-genomics-cloud.readthedocs.io/en/latest/sections/HowToGetStartedonISB-CGC.html

Once you've set up your GCP project, insert your project ID into code within the Colaboratory notebook.

When you create a GCP, a billing account will be attached to it. Any charges incurred by BigQuery are billed to the attached billing account even if you're accessing data found in another project. Here's some information on BigQuery costs. You can keep an eye on your GCP expenses in your Google Cloud Platform Console home page.

BigQuery Colab Notebook

Thank you! Please let know if you have any questions.

October, 2018

Jupyter notebooks and Dataproc clusters.

Switching gears from last month when we learned how to start the RStudio server in the Google cloud, this month we'll discover how to quickly start up a Jupyter notebook. Secondly we'll make that notebook the front end of a Dataproc cluster, for some serious compute power.

In this tutorial we'll be following these tutorials: jupyter notebook and Dataproc with GCP.

Jupyter notebooks are popular as a workspace in data science that makes it easy to share work. From the Jupyter site: "The notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text."

From Google: Google Dataproc is a fast, easy-to-use, fully-managed cloud service for running Apache Spark and Apache Hadoop clusters in a simpler, more cost-efficient way. Operations that used to take hours or days take seconds or minutes instead, and you pay only for the resources you use (with per-second billing). Cloud Dataproc also easily integrates with other Google Cloud Platform (GCP) services, giving you a powerful and complete platform for data processing, analytics and machine learning.

Starting scripts

The first task is getting a VM up and running with Jupyter. Google has provided a set of bash scripts to make starting Jupyter notebooks more convenient. You can find this project on github. We will be passing the bucket address of the initialization script to *gcloud dataproc clusters create* to to all the needed installations on the VM.

Single node clusters

If you only want a single VM as the computational platform, then you can start a 'single node cluster'. Single node clusters have dataproc-role set to 'Master' and the dataproc-worker-count set to 0. Most of the initialization actions in this repository should work out of the box, as they run only on the master. Actions that run on all nodes of the cluster (such as cloud-sql-proxy) similarly work out of the box.

To create a single node cluster, we use the 'gcloud dataproc clusters create' command with a special argument '–single-node'.

```
gcloud dataproc clusters create <<args>> --single-node
```

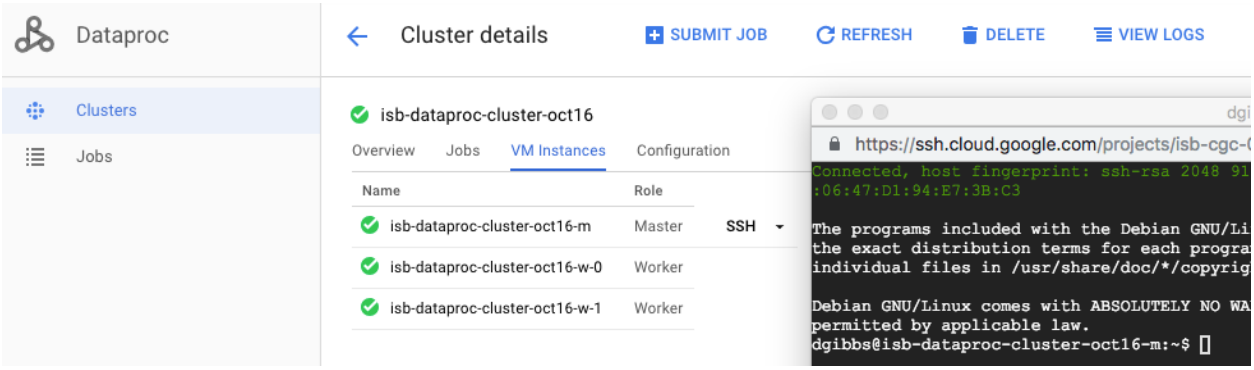
Multinode clusters

But let's suppose we'd like a few worker nodes, to start up that small cluster we use the command:

```
gcloud dataproc clusters create isb-dataproc-cluster-test2 \
    --metadata "JUPYTER_PORT=8124" \
    --initialization-actions gs://dataproc-initialization-actions/jupyter/jupyter.sh \
    --properties spark:spark.executorEnv.PYTHONHASHSEED=0,spark:spark.yarn.am.
↪memory=1024m \
    --worker-machine-type=n1-standard-4 \
    --master-machine-type=n1-standard-4
```

This, init script *jupyter2/jupyter2.sh* uses python2, and *jupyter/jupyter.sh* uses python3. In the metadata, it's also possible to add *JUPYTER_CONDA_PACKAGES=numpy:pandas:scikit-learn*, but at the moment I'm getting some errors with those options. Running that we get a return message..

```
Waiting on operation [projects/isb-cgc-02-0001/regions/global/operations/e2b39fb6-e139-
↪3028-b7bc-33e1c1ca352b].
Waiting for cluster creation operation...done.
Created [https://dataproc.googleapis.com/v1/projects/my-project-123/regions/global/
↪clusters/cluster-name-here] Cluster placed in zone [us-west1-b].
```



Pay attention to the zone where the cluster lives!

Scaling Clusters

After creating a Cloud Dataproc cluster, you can scale the cluster by increasing or decreasing the number of primary or secondary worker nodes in the cluster. You can scale a Cloud Dataproc cluster at any time, even when jobs are running on the cluster.

Because clusters can be scaled more than once, you might want to increase/decrease the cluster size at one time, and then decrease/increase the size later.

```
gcloud dataproc clusters update cluster-name \
  [--num-workers and/or --num-preemptible-workers] new-number-of-workers
```

Connecting to the notebook

This is actually one of the tricky parts.

Now that we have the VMs running, we need to connect our browser to the notebook (living in the cloud).

When using DataLab, one opens the network-connection to the notebook via the Google Cloud Console and Cloud shell in particular. However, here, we can't use the cloud shell to open the Jupyter notebook because it uses a different ssh tunnelling system. This is a clear case where using DataLab is going to be easier in the Google ecosystem.

If you log into the cloud console, find Dataproc and click on your cluster, under the cluster name is a link that brings up the commands to create an SSH tunnel to connect to a web interface.

Essentially, the two commands, ON A MAC, are:

```
gcloud compute ssh DATAPROC_CLUSTER_NAME-m \
  --project=GOOGLE_PROJECT_ID \
  --zone=ZONE -- -D 8124 -N  \
  &

"/Applications/Google Chrome.app/Contents/MacOS/Google Chrome" \
  --proxy-server="socks5://localhost:8124" \
  --user-data-dir="/tmp/DATAPROC_CLUSTER_NAME-m" http://DATAPROC_CLUSTER_NAME-m:8124
```

The two commands, ON UBUNTU LINUX, are:

```
gcloud compute ssh DATAPROC_CLUSTER_NAME-m \
  --project=GOOGLE_PROJECT_ID \
  --zone=ZONE -- -D 8124 -N  \
  &

/usr/bin/google-chrome \
  --proxy-server="socks5://localhost:8124" \
  --user-data-dir="/tmp/DATAPROC_CLUSTER_NAME-m" http://DATAPROC_CLUSTER_NAME-m:8124
```

note the '-m' and putting it in the background with '&' The first command opens an SSH tunnel to the server, and the second opens a browser window using the correct proxy and port. Just replace DATAPROC_CLUSTER_NAME with the name you gave your cluster in the *gcloud dataproc clusters create* call.

And that should open a chrome browser connection to Jupyter. Whew!

One more useful command is:

```
gcloud dataproc clusters describe ${DATAPROC_CLUSTER_NAME}  ### this is just FYI ###
```

Now, there is a Google provided *launch-jupyter-interface.sh* script, but I had a lot of issues with it. So I'm not sure I would recommend it yet.

Ready for work!

We can now create a new notebook. To make backups of our notebook, under the File menu, select 'save as', and here we can select one of the cloud buckets associated with our project.

You can download the notebook I made here

~~Let's try some BigQuery!~~

For the first bit of code, we'll import the libraries we need.

The python client library for the Google cloud can be found here.

```
import sys
!{sys.executable} -m pip install --upgrade google-cloud-bigquery

import pandas as pd
from pandas.io import gbq

print("Imports run.")
```

OK! Next cell:

```
query_job = client.query("""
WITH
  table1 AS (
  SELECT
    project_short_name,
    case_barcode,
    IF (gender = 'FEMALE',
      1,
      0) AS F,
    IF (gender = 'MALE',
      1,
      0) AS M
  FROM
    `isb-cgc.TCGA_bioclin_v0.Clinical`
  WHERE
    project_short_name = 'TCGA-SKCM'
  GROUP BY
    project_short_name,
    case_barcode,
    gender)
    --
    --
SELECT
  project_short_name,
  SUM(F) AS F_count,
  SUM(M) AS M_count
FROM
  table1
GROUP BY
  project_short_name
""")
```

(continues on next page)

(continued from previous page)

```
print('Running query...')
data = gbq.read_gbq(sql, project_id=projectId)

data
```

Ok, we run that cell, and notice it completes very quickly. Next we'll write a cell to get the results.

```
    for row in results:
print("{} : {} : {}".format(row.project_short_name, row.F_count, row.M_count))
```

"TCGA-BRCA : 1085 : 12"

Let's change that query and get results for all types of cancer in TCGA.

```
query_job = client.query("""
        WITH
          table1 AS (
          SELECT
            project_short_name,
            case_barcode,
            IF (gender = 'FEMALE',
              1,
              0) AS F,
            IF (gender = 'MALE',
              1,
              0) AS M
          FROM
            `isb-cgc.TCGA_bioclin_v0.Clinical`
          GROUP BY
            project_short_name,
            case_barcode,
            gender)
            --
            --
        SELECT
          project_short_name,
          SUM(M) AS M_count,
          SUM(F) AS F_count
        FROM
          table1
        GROUP BY
          project_short_name
        """)

results = query_job.result()
```

OK, that's working great. But what about Pandas you're saying?

Well, it turns out we can do the same thing with the pandas and pandas-gbq libraries.

This is nice, because then we can use these summarized results in visualizations or in further analysis.

```
In [15]: data_frame.shape
Out[15]: (33, 3)

In [16]: data_frame
Out[16]:
```

	project_short_name	M_count	F_count
0	TCGA-UCEC	0	548
1	TCGA-CESC	0	307
2	TCGA-OV	0	587
3	TCGA-UCS	0	57
4	TCGA-BRCA	12	1085
5	TCGA-CHOL	20	25
6	TCGA-DLBC	22	26

```
import matplotlib.pyplot as plt
plt.figure();
df2 = pandas.DataFrame(data_frame, columns=['M_count','F_count'])
df2.plot.bar();
```

OK, for the main work product here, we will define a cohort, save that cohort into a new BigQuery table (not download it!), and run a spark job that fits a model.

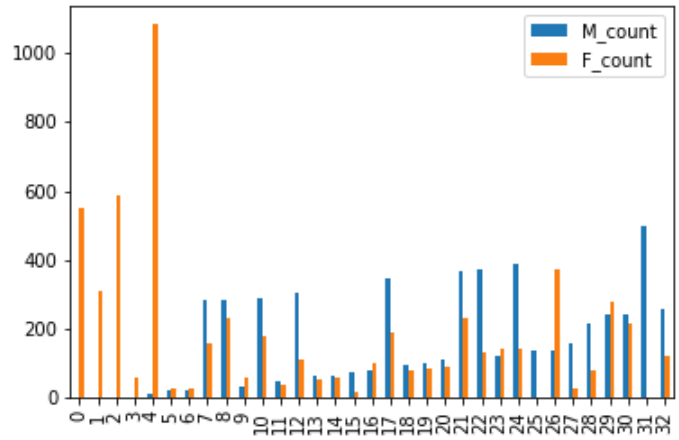
OK, to get started, I popped over to the BQ web interface and created a new dataset in my project: 'spark_job'.

Then I'm created a table that's going to be used for the spark job input.

```
SELECT
  sample_barcode AS sb,
  IF (project_short_name = 'TCGA-STAD', 1, 0) AS label,
  SUM (CASE
     WHEN (HGNC_gene_symbol = 'EGFR') THEN LOG10(normalized_count +1)
     ELSE (RAND()/1000000) END) AS'EGFR,
  SUM (CASE
     WHEN (HGNC_gene_symbol = 'TP53') THEN LOG10(normalized_count +1)
     ELSE (RAND()/1000000) END) AS TP53,
  SUM (CASE
     WHEN (HGNC_gene_symbol = 'NOTCH1') THEN LOG10(normalized_count +1)
     ELSE (RAND()/1000000) END) AS NOTCH1,
  SUM (CASE
     WHEN (HGNC_gene_symbol = 'GATA3') THEN LOG10(normalized_count +1)
     ELSE (RAND()/1000000) END) AS GATA3
FROM
  `isb-cgc.TCGA_hg19_data_v0.RNAseq_Gene_Expression_UNC_RSEM`
WHERE
  project_short_name IN ('TCGA-STAD','TCGA-BRCA')
  AND normalized_count IS NOT NULL
```

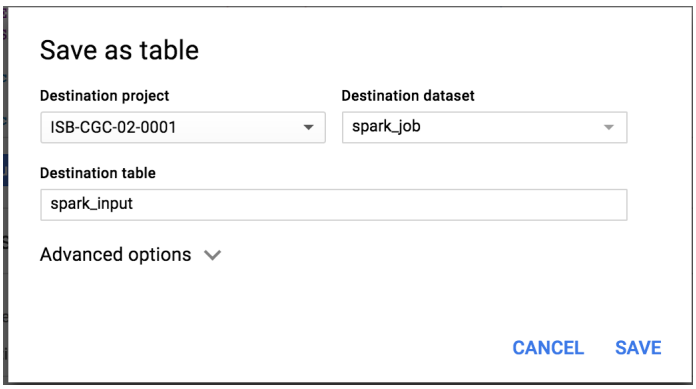
(continues on next page)

```
In [19]: import matplotlib.pyplot as plt
         plt.figure();
         df2 = pandas.DataFrame(data_frame, columns=['M_count','F_count'])
         df2.plot.bar();

         <Figure size 432x288 with 0 Axes>
```



(continued from previous page)

```
GROUP BY
  project_short_name,
  sample_barcode
```



This table is now found in my project at: isb-cgc-02-0001:spark_job.tcga_spark

PySpark

In this section, we're working with the following examples:

1. Get some data from BigQuery and perform some ML in spark .
2. Also this description of the BigQuery connector to Spark .

And here's another resource <https://github.com/jadianes/spark-py-notebooks for working with python and spark in notebook environments.

There's a special package we need to use in order to get our spark context from within a notebook. Typically, we would

submit a job to spark using the PySpark interactive environment, or submit the job through the Google cloud console. But in the notebook we can use 'findspark' to connect with pyspark, and after that, instantiate our spark context.

```
!{sys.executable} -m pip install pyspark findspark
```

Then we can:

```
import findspark
findspark.init()
```

And now we're ready to start coding our spark job. I have heavily borrowed from the examples above..

```
"""Run a logistic regression using Apache Spark ML.

In the following PySpark (Spark Python API) code, we take the following actions:

  * Load a previously created BigQuery input table
    into our Cloud Dataproc Spark cluster as an RDD (Resilient
    Distributed Dataset)
  * Transform the RDD into a Spark Dataframe
  * Vectorize the features on which the model will be trained
  * Compute a linear regression using Spark ML

"""

from datetime import datetime
from pyspark.context import SparkContext
from pyspark.ml.linalg import Vectors
from pyspark.ml.classification import LogisticRegression
from pyspark.sql.session import SparkSession

# The imports, above, allow us to access SparkML features specific to linear
# regression as well as the Vectors types.


# Use Cloud Dataprocs automatically propagated configurations to get
# the Cloud Storage bucket and Google Cloud Platform project for this
# cluster.
sc = SparkContext()
spark = SparkSession(sc)
bucket = spark._jsc.hadoopConfiguration().get("fs.gs.system.bucket")
project = spark._jsc.hadoopConfiguration().get("fs.gs.project.id")

# Set an input directory for reading data from Bigquery.
todays_date = datetime.strftime(datetime.today(), "%Y-%m-%d-%H-%M-%S")
input_directory = "gs:/input_directory = "gs://qotm_oct_2018" + todays_date

# Set the configuration for importing data from BigQuery.
# Specifically, make sure to set the project ID and bucket for Cloud Dataproc,
# and the project ID, dataset, and table names for BigQuery.

conf = {
    # Input Parameters
    "mapred.bq.project.id": project,
```

(continues on next page)

(continued from previous page)

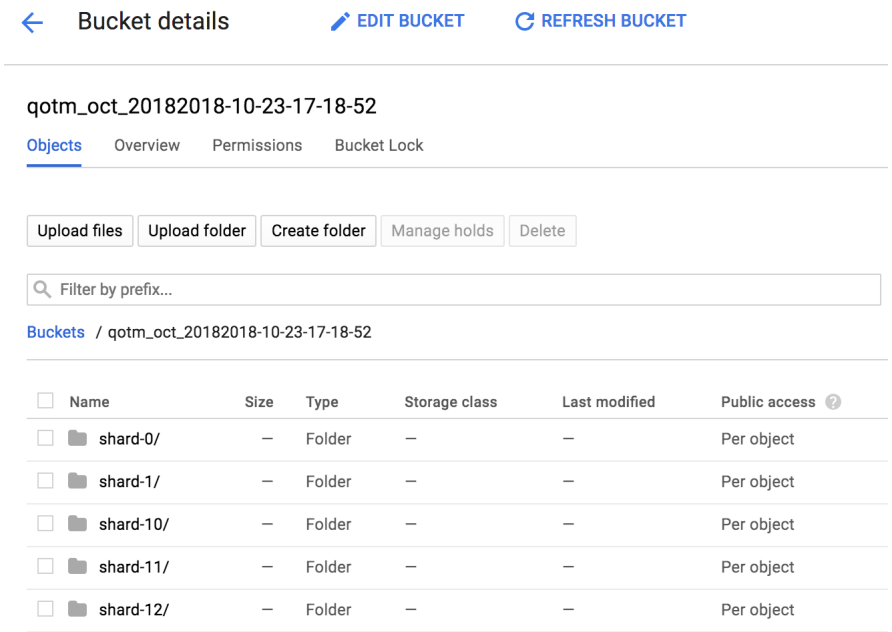
```
    "mapred.bq.gcs.bucket": bucket,
    "mapred.bq.temp.gcs.path": input_directory,
    "mapred.bq.input.project.id": project,
    "mapred.bq.input.dataset.id": "spark_job",
    "mapred.bq.input.table.id": "tcga_spark",
}

# Read the data from BigQuery into Spark as an RDD.
table_data = spark.sparkContext.newAPIHadoopRDD(
    "com.google.cloud.hadoop.io.bigquery.JsonTextBigQueryInputFormat",
    "org.apache.hadoop.io.LongWritable",
    "com.google.gson.JsonObject",
    conf=conf)

# Extract the JSON strings from the RDD.
table_json = table_data.map(lambda x: x[1])

# Load the JSON strings as a Spark Dataframe.
tcga_data = spark.read.json(table_json)

# Create a view so that Spark SQL queries can be run against the data.
tcga_data.createOrReplaceTempView("tcga_view")

# Define a function that collects the features of interest
# Package the vector in a tuple containing the label (`label`) for that
# row.
def vector_from_inputs(r):
  return (float(r["label"]), Vectors.dense(float(r['EGFR"]),
                                    float(r["TP53"]),
                                    float(r["NOTCH1"]),
                                    float(r["GATA3"])))

# As a precaution, run a query in Spark SQL against the view to ensure no NULL values␣
↪exist.
sql_query = """
SELECT *
from tcga_view
where label is not null
        and'EGFR is not null
        and TP53 is not null
        and GATA3 is not null
        and NOTCH1 is not null
"""
clean_data = spark.sql(sql_query)

# Create an input DataFrame for Spark ML using the above function.
training_data = clean_data.rdd.map(vector_from_inputs).toDF(["label",
                                                              "features"])
training_data.cache()


# Construct a new LogisticRegression object and fit the training data.
```

(continues on next page)

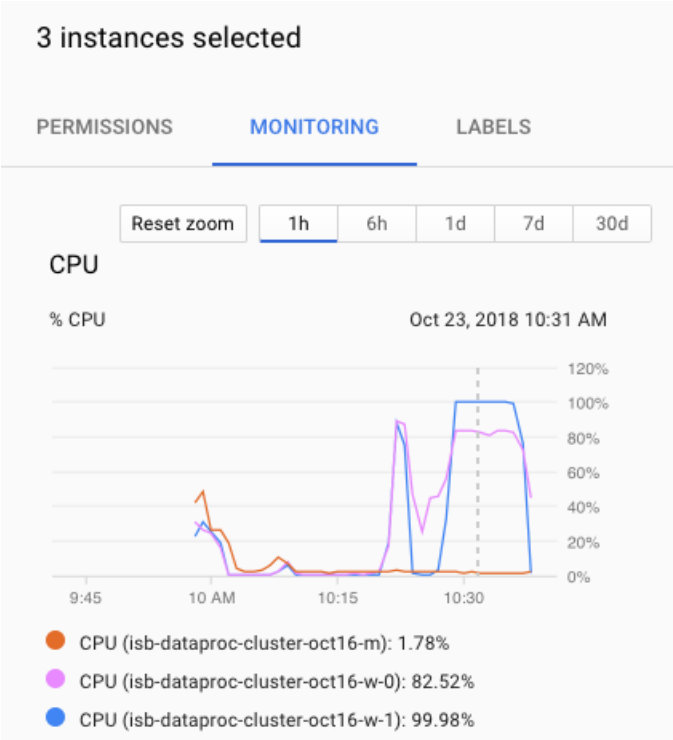
(continued from previous page)

```
# https://spark.apache.org/docs/latest/ml-classification-regression.html#binomial-
↪logistic-regression

lr = LogisticRegression(maxIter=5, regParam=0.3, elasticNetParam=0.8)
lrModel = lr.fit(training_data)
# Print the model summary.
print("Coefficients:" + str(lrModel.coefficients))
print("Intercept:" + str(lrModel.intercept))


# getting the model performance metrics
trainingSummary = lrModel.summary

# Obtain the objective per iteration
objectiveHistory = trainingSummary.objectiveHistory
print("objectiveHistory:")
for objective in objectiveHistory:
    print(objective)

# Obtain the receiver-operating characteristic as a dataframe and areaUnderROC.
trainingSummary.roc.show()
print("areaUnderROC: " + str(trainingSummary.areaUnderROC))

# we can even convert the pyspark DataFrame to a Pandas DataFrame
# and plot the ROC
import pandas
import matplotlib.pyplot as plt
plt.figure();
trainingSummary.roc.toPandas().plot.scatter('FPR','TPR')
```

If we take a look inside our named bucket, we see that data shards.



Here's what it looked like when I was running the *natality example* given by Google:



When we're all done (it actually takes just an instant in this case), we have a model with an ROC of:

```
areaUnderROC: 0.9783191586648377
```

The coefficients from the model were:

```
Coefficients:[50.29267918772197,0.0,0.16224745918590844,-0.31689142394240727]
Intercept:-0.9932429393509908
```

So, it looks like only one gene was actually needed for the classification (EGFR).

Kill -9 the cluster

As with many other cloudy things, you can kill your cluster in the web console, on on the command line:

```
gcloud dataproc clusters update cluster-name \
  --graceful-decommission-timeout="timeout-value" \
  [--num-workers and/or --num-preemptible-workers]=decreased-number-of-workers
  other args ...
```

Hope that was helpful for getting started with Jupyter notebooks! Of course you don't have to use clusters, there's a lot you can do with a single node notebook! If you have some cool examples, I would love to see them!

September, 2018

R in the cloud.

Recently, I was asked to demonstate how to simply run an R script in the google cloud, so I decided to revisit the topic and look for new, easy methods. In the past I recommended methods like using dsub (link), which uses the Google Pipelines API. It's still a good option, but it can be challenging for some users to install and use. As an alternative, I have two new (to me) methods that make running R scripts easy and straightforward.

Method 1

An RStudio server in the cloud

Super developer, Mark Edmondson (github: MarkEdmondson123), has released a number of very useful packages for working in the Google cloud. However, I would advise to work from the development versions in github to get the latest and greatest (see devtools::install_github).

These include:

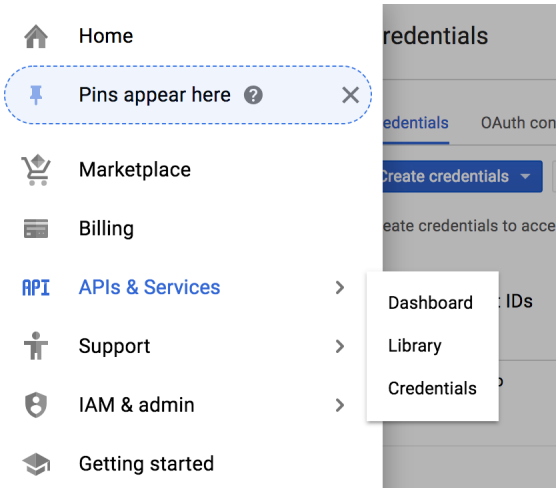
- googleAuthR
- googleComputeEngineR
- googleCloudStorageR
- bigQueryR

Also a couple cloudyr tutorial links: massively parallel and install and auth.

With these packages, it becomes super easy and fast to start up an RStudio server that can acess all the resources within a google project (i.e. read and write to buckets, execute BigQueries).

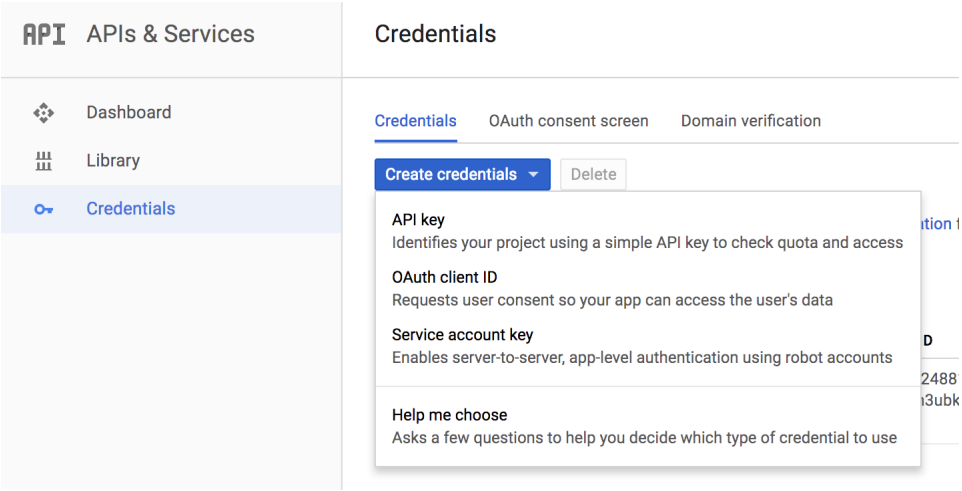
After a little setup, starting the server is accomplished with a single function call! The setup involves getting a project key. To do that, and you only need to do this once, you'll log into your google cloud console.

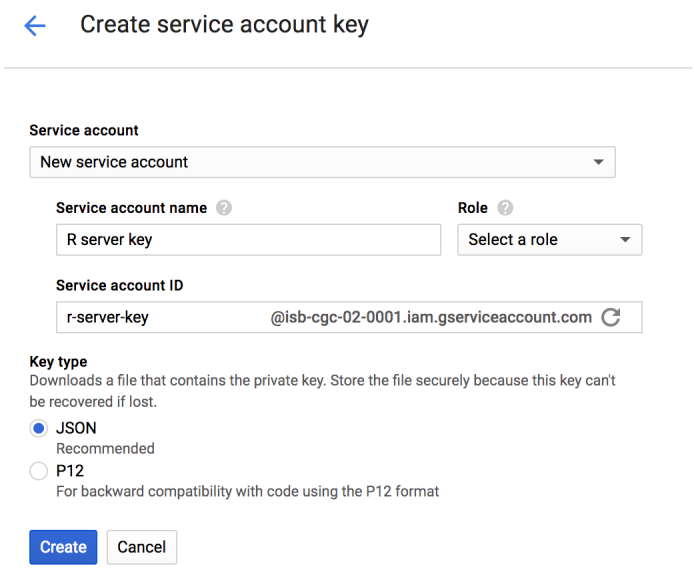
Then, use the hamburger menu (upper left corner) to navigate to the 'APIs and Credentials' page. Find the create credentials button, and select 'Service account key'. Under service account, select 'New service account' and you can give it a name and an role in the project. For simplicity you can select the editor role, knowing it has a great deal of permissions, which you may wish to scale back. When you hit the blue 'create key' button, a json file will be downloaded. Guard that key with your life!



OK.

Now, after starting up R, point your working dir to the directory holding your json key.


API APIs & Services
Credentials
Dashboard
Library
Credentials
Credentials
OAuth consent screen
Domain verification
Create credentials
Delete
API key
Identifies your project using a simple API key to check quota and access
OAuth client ID
Requests user consent so your app can access the user's data
Service account key
Enables server-to-server, app-level authentication using robot accounts
Help me choose
Asks a few questions to help you decide which type of credential to use

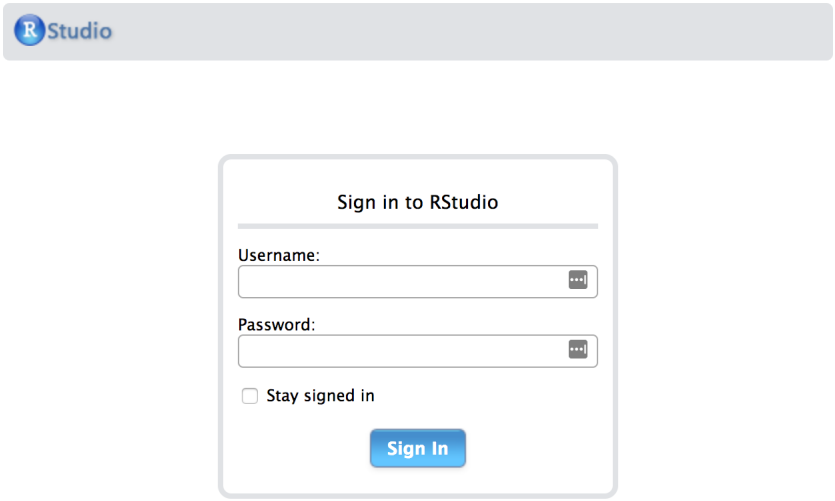

Create service account key
Service account
New service account
Service account name
R server key
Role
Select a role
Service account ID
r-server-key
@isb-cgc-02-0001.iam.gserviceaccount.com
Key type
Downloads a file that contains the private key. Store the file securely because this key can't be recovered if lost.
JSON
Recommended
P12
For backward compatibility with code using the P12 format
Create
Cancel

```
Sys.setenv("GCE_AUTH_FILE" = "~/tmp/auth.json")       # edit path to your file
Sys.setenv("GCE_DEFAULT_PROJECT_ID"="MY PROJECT ID")  # edit this
Sys.setenv("GCE_DEFAULT_ZONE"="us-west1-a")           # edit this


library(googleComputeEngineR)

vm <- gce_vm("rstudio-cron-googleauthr",
             predefined_type = "n1-standard-1",
             template = "rstudio",
             dynamic_image = "gcr.io/gcer-public/google-auth-r-cron-tidy",
             username = "myname",
             password = "secretpassword321")
```

After calling this function, a message is printed: '2018-09-28 15:48:06> VM running'. What happened? A new VM has been started using a docker image that contains Rstudio server and all of the tidyverse. How do you get to it? Well, if we exampine the vm object in R, we see:

```
> vm
==Google Compute Engine Instance==

Name:                rstudio-cron-googleauthr
Created:             2018-09-18 13:06:23
Machine Type:        n1-standard-1
Status:              RUNNING
Zone:                us-west1-b
External IP:         35.199.153.108
Disks:
                         deviceName       type       mode boot autoDelete
1 rstudio-cron-googleauthr-boot-disk PERSISTENT READ_WRITE TRUE       TRUE
```

If we copy that External IP, and paste it into our browser. Viola!



After logging in, we get a full Rstudio environment.

To get a file into your VM, the files panel in the lower right corner has an 'Upload' button that lets you select a file or dataset. To download, in the same pane, select 'More' and 'export'.

But what about reading and writing to your google bucket? To do that, we need to get the session authorized. The RStudio instance, as started up with googleComputeEngineR, contains metadata about the project, and authorization is performed using the googleAuthR package. See below for an example of working with buckets.

```
### FROM WITHIN THE RSTUDIO ENVIRONMENT ###

# first we load this library and call the authorization function
library(googleAuthR)
gar_gce_auth()
```

At this point, authorization is done, and a token has been created. The function outputs: "Token cache file: .httr-oauth".

```
# now we load up the cloud storage package and list the buckets
library(googleCloudStorageR)
googleCloudStorageR::gcs_list_buckets(projectId = 'isb-cgc-02-0001')

# we can then pick a bucket
googleCloudStorageR::gcs_global_bucket("gibbs_bucket_nov162016")

# we can also select the default bucket by setting a environment variable.
Sys.setenv("GCS_DEFAULT_BUCKET" = "gibbs_bucket_nov162016")
```

Now we're ready to start accessing our buckets!

```
## getting a list of objects in the default bucket
objects <- gcs_list_objects()

head(objects$name)  # file names

## save directly to an R object (warning, don't run out of RAM if its a big object)
## the download type is guessed into an appropriate R object

parsed_download <- gcs_get_object(objects$name[4])

# this was a .csv file, and it parsed into a tibble

# or if you already know the name
parsed_download <- gcs_get_object("catter_input.txt")

## if you want to do your own parsing, set parseObject to FALSE
## and use httr::content() to parse afterwards
raw_download <- gcs_get_object("catter_input.txt",
                               parseObject = FALSE)

## Or move from a bucket to a file in your working directory
## parseObject has no effect, it is a httr::content(req, "raw") download
gcs_get_object("catter_input.txt", saveToDisk = "catter_downloaded.csv")

## **** ##
## Here's an example of getting text from a file in GCS
## and parsing it to a data frame
```

(continues on next page)

(continued from previous page)

```
textobj <- gcs_get_object("catter_input.txt")
df <- read.delim( textConnection(textobj), header=T, sep=" ", strip.white=TRUE)
```

Great, now to move files back to the bucket.

```
dat <- read.table('catter_downloaded.csv', header=T)
# saved as cat_plot.png

## attempt upload back to the bucket
upload_try <- gcs_upload("cat_plot.png")
```

You can see how easy it is to startup a new Rstudio server (takes just a few seconds) and start reading and writing to buckets. When you're done, you can stop the VM.

```
### BACK ON YOUR LOCAL MACHINE ###

gce_vm_stop(vm)
```

However, you will still be charged for the attached disk, but this lets you resume your session anytime to start where you left off. It's also easy to just write out your files to the bucket, and delete the VM, which is what I tend towards.

As a note: it's very fast (and free as long as the VMs and buckets are in the same region) to move data around in the google cloud.

Method 2

Running R functions on a cloud-based-cluster.

Next we're going to start up a set of VMs, link them together as a cluster, and submit work to them. We're still going to use googleComputeEngineR to start up VMs, keeping them in a list, and then using the future package to create the cluster.

Here's a couple cloudyr links:

- massively parallel
- install and auth.

```
library(googleComputeEngineR) ## using the dev version from github
library(future)

## names for your cluster
vm_names <- c("vm1","vm2","vm3")

## creates jobs that are creating VMs in background

jobs <- lapply(vm_names, function(x) {
    gce_vm_template(template = "r-base",
                    predefined_type = "n1-standard-1",
                    name = x,
                    disk_size_gb = 15,
                    wait = FALSE)
                    })
```

Now, since we set wait = False, we call the function and then we get back control of the environment.

```
2018-09-28 17:23:11> Returning the startup job, not the VM instance.
2018-09-28 17:23:14> Returning the startup job, not the VM instance.
2018-09-28 17:23:16> Returning the startup job, not the VM instance.
>
> jobs
[[1]]
==Zone Operation insert :  PENDING
Started:  2018-09-28 14:23:09
[[2]]
==Zone Operation insert :  PENDING
Started:  2018-09-28 14:23:11
[[3]]
==Zone Operation insert :  PENDING
Started:  2018-09-28 14:23:14
```

The 'jobs' object is a list, which we'll convert to a list of VM objects. Then we can apply functions to that list of VMs, in order to (for example) shut them all down.

```
## wait for all the jobs to complete and VMs are ready
vms <- lapply(jobs, gce_wait)

## get the VM objects
vms <- lapply(vm_names, gce_vm)

## set up SSH for the VMs
vms <- lapply(vms, gce_ssh_setup)
```

Now for creating the cluster! This part is somewhat tricky, and at times seems to flop. If the plan function doesn't work, then just try again, the docker pulls already done will not pull again. A lot of times it takes a couple tries.

This is a cool part(!): I've created my own small docker image and pushed it to docker hub. When building the cluster, we're able to start up those docker images in each VM in the cluster. This gives us control over what software is present on each worker node.

The docker file is found at: https://hub.docker.com/r/gibbsdavidl/googlesmallr/

```
## customise as needed, this for example sets shared RAM to 13GB
my_rscript <- c("docker",
                "run", c("--net=host","--shm-size=8G"),
                "gibbsdavidl/googlesmallr:latest",
                "Rscript")

## create the cluster using custom docker image
plan(cluster,
     workers = as.cluster(vms,
                          docker_image="gibbsdavidl/googlesmallr:latest",
                          rscript=my_rscript)
     )
```

OK, now the cluster should be alive and waiting for something to do. You can go and see the VMs in your google cloud console, monitoring their workloads.

The (below) task will be to read a file from our bucket and report the size of the table.

```
## test out if it's possible to access buckets.
work_chunks <- function(chunk){

  # first we'll get the worker node authorized
  require(googleAuthR)
  require(googleCloudStorageR)
  googleAuthR::gar_gce_auth()

  # then we'll point to the bucket
  gcs_global_bucket("gibbs_bucket_nov162016")

  # and get the object, read it, and report the dimensions.
  gcs_get_object("catter_input.txt", saveToDisk = "catter_downloaded.csv")
  dat <- read.table('catter_downloaded.csv', header=T)
  return(dim(dat))
}


# We use the future_lapply to send this function to each VM.
system.time(
  result2 <- future.apply::future_lapply(vm_names, work_chunks)
)
```

```
#   user  system elapsed
#  0.035   0.004   1.155

> result2
[[1]]
[1] 21  2

[[2]]
[1] 21  2

[[3]]
[1] 21  2
```

Great! In this example, I used the vm_names to iterate across, but it could been a list of data files, or a list of parameter sets.

```
paramList <- list(
  P1=c(1,2,3), P2=c(4,5,6), P3=c(7,8,9)
)

result3 <- future.apply::future_lapply(paramList, work_chunks)

## work_chunks would need an extra parameter in the argument list ##
```

I hope these examples help get you in the cloud! Please let me know if you have trouble or have questions.

August, 2018

Using BigQuery ML in a Shiny app.

Last month, we tried out the newly-released Google BigQuery ML. This month we'll continue to build examples, learn some new things, and build a shiny web app.

One newsworthy bit of information, incase you missed it a few months ago, is that the R package used for interacting with BigQuery, bigrquery, has undergone a major revision (hitting version 1.0.0), and many of the function calls have changed significantly. The returned object from making a BigQuery call (with function 'bq_project_query') is now a "tibble" rather than a data frame.

Working with BigQuery ML is quite a bit different than what we've done before. In the past, when working with BigQuery, we've computed different statistics, and we've even used those statistics for classification, but that work was all done in the SQL – including, for example, formulating a Z-score in SQL. Now, most of our work will go into preparing the training data table to be used when fitting the model.

When fitting models, we have two important parameters to think about: the L1 and L2 regularization rates. (There are other parameters, but we'll focus on these for the moment.) Both of these parameters effectively push the weights of less useful predictors towards zero. L2 (or euclidean norm) will push weights towards zero, but L1 regularization will make variable (gene) weights exactly zero. Using these regularizers can help us get an idea of which features (*eg* genes) are most useful in separating groups (*eg* cancer types).

A good tool for getting a feel for what parameter values to use can be found here.

In this Shiny App, we will select two "cohorts" as the two groups for our classification task. Then we'll select one of the cancer hallmark gene sets from MSigDB which will provide the feature set as a list of genes. The BigQuery ML models take a number of parameters, so we'll make those available to the user as well.

In general, when storing a gene-expression matrix in BigQuery, it is most useful to store it as a tidy data table, *ie* a "long" table with just 3 columns: sample_id, gene_id, expression_value, rather than a "wide" N x M table for N samples and M genes, in which the expression values for each gene are stored in a specific column.

However, when you want to fit a model with 10 variables, you will need a table with 10 columns. To do that we'll need a new BigQuery skill! Converting long-to-wide tables. There's two keys to doing it in BigQuery. The first is to programmatically construct the query string, given the list of genes and other relevant information (*eg* the cohort name), and the second is to use an aggregate function to create each row of the result table, where each row will represent a single sample.

Here's a small example of doing that.

```
SELECT
  sample_barcode AS sb,
  project_short_name AS label,
  SUM (CASE
     WHEN (HGNC_gene_symbol = 'FRMD6') THEN LOG10(normalized_count +1)
     ELSE (RAND()/1000000) END) AS FRMD6,
  SUM (CASE
     WHEN (HGNC_gene_symbol = 'MMD') THEN LOG10(normalized_count +1)
     ELSE (RAND()/1000000) END) AS MMD,
  SUM (CASE
     WHEN (HGNC_gene_symbol = 'IMPDH2') THEN LOG10(normalized_count +1)
     ELSE (RAND()/1000000) END) AS IMPDH2,
  SUM (CASE
     WHEN (HGNC_gene_symbol = 'SNORD60') THEN LOG10(normalized_count +1)
     ELSE (RAND()/1000000) END) AS SNORD60
FROM
```

(continues on next page)

(continued from previous page)

```
  `isb-cgc.TCGA_hg19_data_v0.RNAseq_Gene_Expression_UNC_RSEM`
WHERE
  project_short_name = 'TCGA-STAD'
  AND normalized_count IS NOT NULL
GROUP BY
  project_short_name,
  sample_barcode
```

Keen readers will notice that I've included a call to RAND() when the gene symbol is not matched. The reason is that some genes in the gene lists do not map to data in TCGA. This creates a column of all zeros across samples, and causes an error in the model fitting. To get around that, I've added a very small amount of noise to the data. If the gene is not present in TCGA's annotation, it is represented as random noise and will not contribute to the model.

Now we'll get to building the shiny app. Here's the first couple functions to build up the SQL as a string. I put these SQL query-string building functions in a global.R file that gets imported in the server.R code. Then I make the query executions from an eventReactive function which is called when the user clicks on the submit button.

```
geneQuery <- function(gi) {
    # This function gets called for each gene in a list.
    # gi is the name of a gene as a string
    paste("SUM (CASE WHEN (HGNC_gene_symbol = '",gi,"') THEN normalized_count ELSE␣
↪(RAND()/1000000) END) AS ", gi, sep='')
}


buildDataSQL <- function(geneNames, cohort1, cohort2, ngenes) {

    # here we can control the number of genes going into the model
    if (length(geneNames) > ngenes) {
            geneNames <- sample(geneNames, size = ngenes, replace = F)
    }

    # create model name // format the sys.time() return
    # to put in underscores and remove colons
    # and paste in the cohort names.
    modelname <- getModelname(cohort1, cohort2)

    q <- paste("
            WITH
            C1 AS (
            SELECT
               sample_barcode AS sb,
               project_short_name AS label,\n",
               paste(sapply(geneNames, function(gi) geneQuery(gi)),collapse = ',\n'), "\n
            FROM
               `isb-cgc.TCGA_hg19_data_v0.RNAseq_Gene_Expression_UNC_RSEM`
            WHERE
               project_short_name = '",cohort1,"'
               AND normalized_count IS NOT NULL
            GROUP BY
               project_short_name,
               sample_barcode ),
```

(continues on next page)

(continued from previous page)

```
            C2 AS (
            SELECT
               sample_barcode AS sb,
               project_short_name AS label,\n",
               paste(sapply(geneNames, function(gi) geneQuery(gi)),collapse = ',\n'), "\n
            FROM
               `isb-cgc.TCGA_hg19_data_v0.RNAseq_Gene_Expression_UNC_RSEM`
            WHERE
               project_short_name = '",cohort2,"'
               AND normalized_count IS NOT NULL
            GROUP BY
               project_short_name,
            sample_barcode )

            SELECT
               0 AS label,",
               paste(geneNames,collapse = ','), "\n
            FROM
               C1
            UNION ALL
            SELECT
               1 AS label,",
               paste(geneNames,collapse = ','), "\n
            FROM
               C2
            ", sep = '')

    print(q)
    return(list(SQL=q, Dataset="tcga_model_1", Tablename=paste("isb-cgc-myproject123.
↪tcga_model_1.data", Modelname=modelname,sep="")))
    }
```

Calling this function returns the SQL query string, the Dataset where the table will be placed, the full name 'project.dataset.tablename', and the modelname, all as a list.

The R code to create the query string, authenticate, and execute the query only takes a few lines:

```
# we've saved our service account token in the data directory
service_token <- set_service_token("data/ISB-CGC-myproject-1234567.json")

# previously I made a hash keyed on gene set names, to get the list of gene members
load("data/gene_set_hash.rda")
geneNames1 <- geneSets[[setname]]

# then we build the string using the above code.
datasql <- buildDataSQL(geneNames1, input$cohortid1, input$cohortid2, input$n_genes)

# and we execute the query, explicitly naming the location where it will be saved
res0 <- bq_project_query('isb-cgc-myproject123', datasql[["SQL"]],  destination_table =␣
↪datasql[["Tablename"]])
```

At this point we've generated the dataset and saved it in a BigQuery dataset. The next step is to fit the model. We'll construct another query string and execute it.

```
buildModelSQL <- function(datasetname, tablename, modelname, input) {

    l1reg <- input$l1_reg
    l2reg <- input$l2_reg
    maxit <- input$max_iterations
    lr <- input$learn_rate
    es <- input$early_stop

    q <- paste(
    "CREATE MODEL `", datasetname ,".", modelname, "`
     OPTIONS(model_type='logistic_reg', l1_reg=",l1reg,", l2_reg=",l2reg,", max_
↪iterations=",maxit,")
     AS SELECT * FROM `", tablename ,"`
    ",sep="")

    print(q)
    return(list(SQL=q, Modelname=paste(datasetname ,".", modelname,sep='')))

}

# then build the model
# the datasql is returned from building the dataset query above
modSql <- buildModelSQL(datasql[["Dataset"]], datasql[["Tablename"]], as.list(input))
res1 <- bq_project_query('isb-cgc-myproject123', modSql[["SQL"]])
```

When the model fit is finished, we will query the *model*, rather than a table, to get information about the goodness-of-fit, and other classification metrics.

```
queryModelTrainingSQL <- function(modelname) {
  q <- paste(
    "SELECT
       *
     FROM
      ML.TRAINING_INFO(MODEL `",modelname,"`)
    ",sep="")
  print(q)
  return(list(SQL=q))
}

queryModelFeaturesSQL <- function(modelname) {
  q <- paste(
    "SELECT
       *
     FROM
      ML.FEATURE_INFO(MODEL `",modelname,"`)
    ",sep="")
  print(q)
  return(list(SQL=q))
}

queryModelWeightsSQL <- function(modelname) {
  q <- paste(
    "SELECT
```

(continues on next page)

(continued from previous page)

```
      processed_input,
      weight
     FROM
      ML.WEIGHTS(MODEL `",modelname,"`)
    ",sep="")
  print(q)
  return(list(SQL=q))
}

queryModelROCSQL <- function(modelname, tablename) {
  q <- paste(
    "SELECT
      threshold,
      false_positive_rate,
      true_positives,
      false_positives,
      true_negatives,
      false_negatives,
      recall,
      true_positives / (true_positives + false_positives) AS precision
     FROM
      ML.ROC_CURVE(MODEL `",modelname,"`, TABLE `", tablename ,"`)", sep='')
  return(list(SQL=q))
}
```

and then we call all the query contruction functions and collect the performance of the classifier.

First, using the queryModelTrainingSQL function above, we get information about the model training, which is really useful. It will show a number of training iterations, where in each iteration, there's a learning rate. When a model is fitting well, you should see a big jump in the magnitude of the learning rate. Also it needs to train for a number of iterations. In unsuccessful fittings, the model will not progress beyond just a few iterations. Here is the Google training module on this topic.

Second, we can get information about the features themselves, such as the mean and quartiles for each gene.

Finally, one of the most important calls will be to get the feature weights. Since this is a regularized regression, which is controlled using the L1 and L2 parameters, variables that are less helpful in the classification will have weights that will shrink to zero. We show the weights in an absolute value sorted order.

In supervised machine learning, each sample has a known label. In this example, the label is the tissue type. When the model is used to predict a label for a sample, we will either get it right or get it wrong. We can call the label of the sample either 'cohort 1 positive' or 'cohort 1 negative (i.e. cohort 2 positive). Then our model makes a prediction on whether the sample is 'cohort 1 positive' or not, making it a boolean value (true or false).

To determine if our model is doing well, we use classification metrics like recall and precision. Precision is the fraction of true positives over combined true and negative positives. Recall (or sensitivity) is the fraction of true positives over all positives. So when precision is very close to 100%, then there were very few false positives. When recall is close to 1, almost all of the positive cases were correctly called positive.

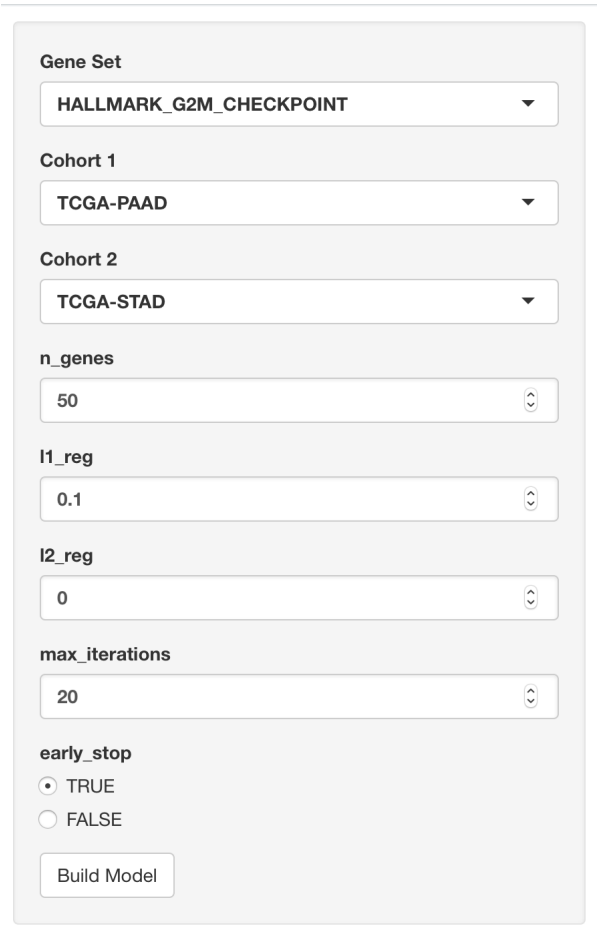
To put it another way, from Wikipedia: "In a classification task, a precision score of 1.0 for a class C means that every item labeled as belonging to class C does indeed belong to class C (but says nothing about the number of items from class C that were not labeled correctly) whereas a recall of 1.0 means that every item from class C was labeled as belonging to class C (but says nothing about how many other items were incorrectly also labeled as belonging to class C). … Often, there is an inverse relationship between precision and recall, where it is possible to increase one at the cost of reducing the other."

Below is the R code for the "server" component of our Shiny app. We want all of the bigrquery functions to be called when the user clickson the 'submit' button – to accomplish this, we wrap all of the functions together in a eventReactive.

Once it's all wrapped in a the eventReactive, we can access the results of all the queries repeatedly without having to redo any of the queries.

Let's take a look at an example. You can of course try the app here.

First we have the UI where we can pick our cohorts, the number of genes to use, the regularization rates, and a maximum number of iterations.



Then we see the name of the model (and data) which can be found in the Google BigQuery web interface. Also we see the precision-recall curve.

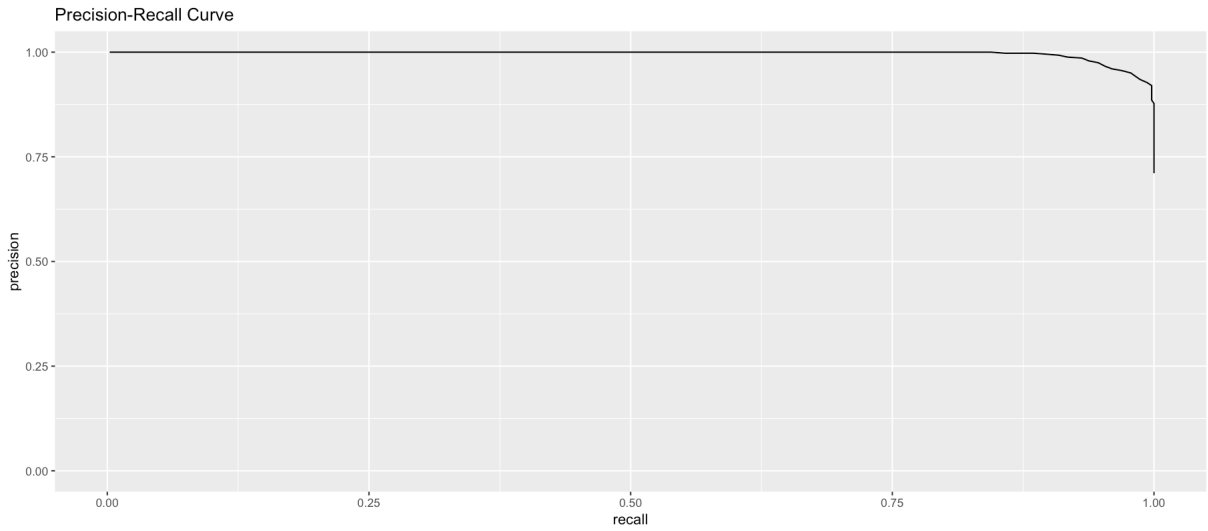
Next we have the metrics that are found when using the threshold that maximizes the F-score.

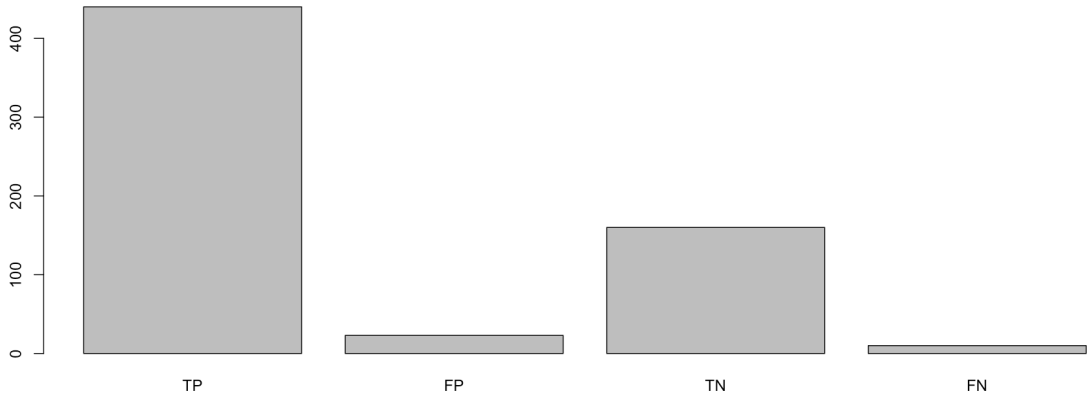
Then we have the list of features (genes) with the model weights.

Finally, we have the record of iterative model fitting.

That's it for this month. I hope you found this informative, and can see how to integrate model building and exploration within an interactive web environment. This sort of tool would allow anyone with some familiarity of gene sets and TCGA to build and reason about models. And we think that's pretty cool!

tcga_model_1.TCGA_PAAD_TCGA_STAD_2018_08_28_16_41_55





threshold	false_positive_rate	true_positives	false_positives	true_negatives	false_negatives	recall	precision	Fscore
0.36	0.13	440	23	160	10	0.98	0.95	0.96

Gene Weights

processed_input	weight
__INTERCEPT__	0.80
H2AFX	0.23
ESPL1	0.22
SMC4	0.21
MKI67	0.19
KIF11	0.17
CCNA2	0.14
PLK1	0.14
CCNF	0.14
CENPE	0.12
PLK4	0.11

training_run	iteration	loss	eval_loss	duration_ms	learning_rate
0	8	0.34	0.39	8599	0.40
0	7	0.35	0.39	9758	0.80
0	6	0.37	0.42	14785	0.40
0	5	0.39	0.42	6248	0.80
0	4	0.41	0.45	6239	0.40
0	3	0.43	0.46	2798	0.80
0	2	0.46	0.50	2765	0.80
0	1	0.51	0.54	3128	0.40
0	0	0.54	0.57	2551	0.20

July, 2018

First look: BigQuery ML.

Exciting news! Google has just released a beta feature in BigQuery: Machine Learning (ML)! There are two availble model types, linear and logistic. The first, linear regression, models a continuous variable given a selection of variables, both categorical (*eg* US postal code) and numeric (*eg* height or weight). The second, logistic regression, models a binary label given some variables. This is used for classification between two groups, which we have used extensively in this blog. An example of groups we created had features like 'does or does not have a mutation in GATA3'. Even better, the logistic regression is regularized! We have both L1 and L2 regularization available, which makes it similar to an implementation of elasticnet.

In the following examples, I'm going to be working in the BigQuery web interface, but it's also possible to train and apply these models using the command line tool (bq), the REST API, and from a scripting language (R or python).

The introductory documentation can be found here.

Something people are always concerned about: how much does it cost?! Well, from reading the docs, at this time (July 2018) pricing is still under development. But essentially it's similar to any other query. You're charged according to how much data is processed in training the model and storage fees for the model (first 10GB free). However, the actual data read for training the model is more than just the size of the table. It's not entirely clear at this point, but when I learn more, I'll report it.

Costs

This tutorial uses billable components of Cloud Platform, including:

- Google BigQuery
- BigQuery ML

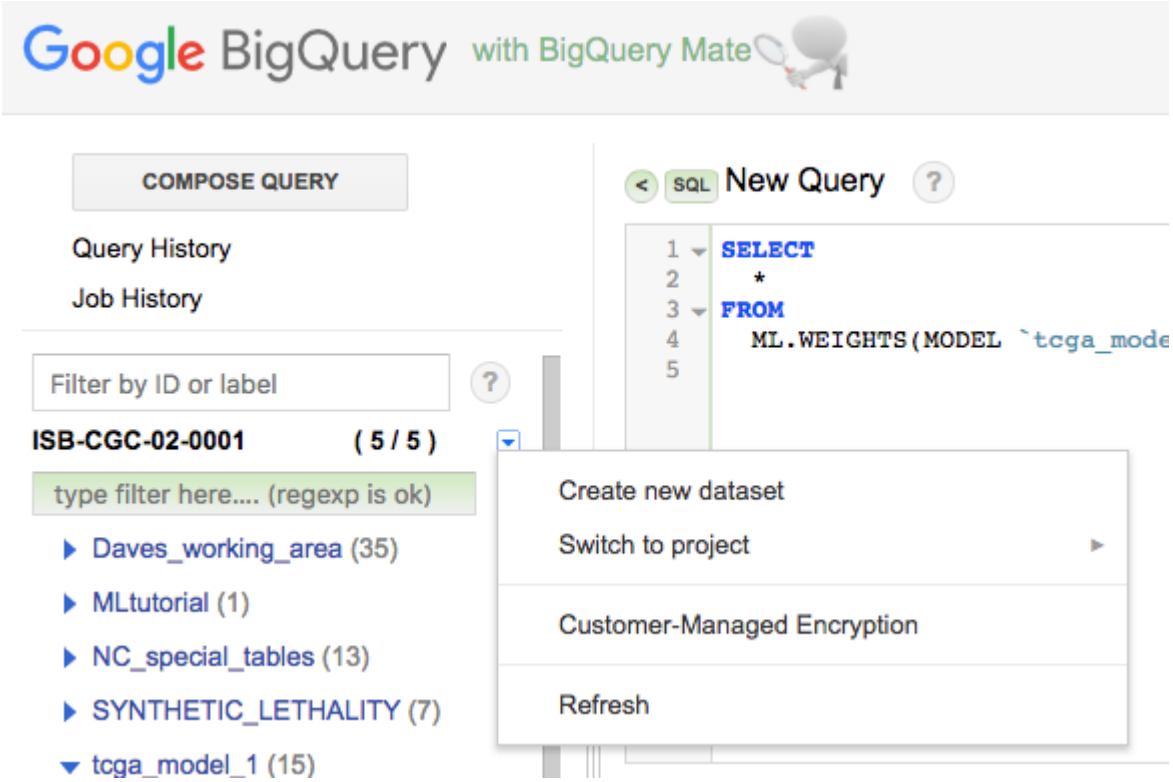
You incur charges for:

- Storing your ML model and training data in BigQuery
  - The first 10 GB of storage is free each month.
- Querying data in BigQuery
  - The first 1 TB is free each month.
  - If you are using flat-rate pricing, query costs are included in the monthly flat-rate price.
- Running BigQuery ML SQL statements

Something kind of amazing (doc link): If you're using a categorical variable, the variable is split into a number of columns, one for each category-element. This is called one hot encoding. For example, we might have two columns from our mutation status category: has-GATA3-mutation, no-GATA3-mutation. That's only 2 categories, so 2 columns of binary variables. But, if you have many, many, many more categories, those get split into columns too. From the docs: "When you use a CREATE MODEL statement, the size of the model must be 90 MB or less or the query fails. Generally, if all categorical variables are short strings, a total feature cardinality (model dimension) of 5-10 million is supported. The dimensionality is dependent on the cardinality and length of the string variables." WOW! That's a lot of columns.

Let's jump in! The first task will be to classify a couple of cancer types (by tissue) using gene expression data for 4 specific genes.

First I'm going to create a new data set to hold the training data and models. To create a new dataset, click on the blue down-arrow next to your project ID in the left side-panel of the web UI), and select "Create new dataset". I called it 'tcga_model_1'.



I've selected 'TCGA-COAD' (colon cancer) and 'TCGA-PAAD' (pancreatic cancer) as my two cancer types. They're really pretty different, so it shouldn't be a difficult classification challenge.

Below, we use a Standard SQL query to create the training data, which we then save as a table in the dataset created above.

```
-- For each gene, we'll make a subtable named C1-C4.
-- You can see where we select the gene in the WHERE section.

With

C1 AS (
SELECT
  project_short_name AS label,
  sample_barcode,
  HTSeq__FPKM_UQ CCNE1
from
  `isb-cgc.TCGA_hg38_data_v0.RNAseq_Gene_Expression`
WHERE
  project_short_name IN ('TCGA-COAD','TCGA-PAAD')
        and gene_name='CCNE1'
),

C2 AS (
SELECT
```

(continues on next page)

(continued from previous page)

```
  project_short_name AS label,
  sample_barcode,
  HTSeq__FPKM_UQ CDC6
from
  `isb-cgc.TCGA_hg38_data_v0.RNAseq_Gene_Expression`
WHERE
  project_short_name IN ('TCGA-COAD','TCGA-PAAD')
        and gene_name='CDC6'
),

C3 AS (
SELECT
  project_short_name AS label,
  sample_barcode,
  HTSeq__FPKM_UQ MDM2
from
  `isb-cgc.TCGA_hg38_data_v0.RNAseq_Gene_Expression`
WHERE
  project_short_name IN ('TCGA-COAD','TCGA-PAAD')
        and gene_name='MDM2'
),

C4 AS (
SELECT
  project_short_name AS label,
  sample_barcode,
  HTSeq__FPKM_UQ TGFA
from
  `isb-cgc.TCGA_hg38_data_v0.RNAseq_Gene_Expression`
WHERE
  project_short_name IN ('TCGA-COAD','TCGA-PAAD')
        and gene_name='TGFA'
)

-- Now we join the above gene-tables into our training data.

SELECT
C1.label AS label,
C1.sample_barcode AS sample_barcode,
  CCNE1,
  CDC6,
  MDM2,
  TGFA
FROM
C1
JOIN C2 ON C1.label=C2.label AND C1.sample_barcode=C2.sample_barcode
JOIN C3 ON C1.label=C3.label AND C1.sample_barcode=C3.sample_barcode
JOIN C4 ON C1.label=C4.label AND C1.sample_barcode=C4.sample_barcode
```

I ran the above query, and when done, clicked the 'Save to Table' button, placing it in the 'tcga_model_1' dataset. Now we're ready to train a model, which we'll do using the CREATE MODEL statement. (You will need to modify the CREATE MODEL statement below to use your project and dataset names.)

```
#standardSQL
CREATE MODEL
  `isb-cgc-02-00001.tcga_model_1.coad_vs_paad_expr_l1_l2`  -- the name of our model,␣
↪project.dataset.model_name
OPTIONS
  ( model_type='logistic_reg',            -- various options for the model
    l1_reg=1, l2_reg=1 ) AS
SELECT
  label,                                  -- here you define the training data
  CCNE1,                                  -- it's possible to give it a random subset
  CDC6,                                   -- see the next query for that.
  MDM2,
  TGFA
FROM
  `isb-cgc-02-0001.tcga_model_1.paad_coad_expr_2`
```

The model training should take a minute or two, and once complete you will now have a "Model" in your dataset (identified in the webUI using a green icon which is different from the blue one we are used to seeing next to tables). You can click on this model to see information about it, along with new buttons such as "Query Model" and "Training Stats":

Looking at the "Training Stats" will give you a sense of how the training process went. Each row in the Training Stats table represents a single iteration, with the following four pieces of information:

- training data loss: loss metric on the training data, calculated after this training iteration
- evaluation data loss: loss metric computed on the held-out data
- learn rate: hyper-parameter which controls how fast the model weights are adjusted (how this value is determined will depend on the learn_rate_strategy specified in the CREATE MODEL statement)
- completion time (sec)

When the model's fit is not improving, the training will end early (you can turn this feature off). I found that models that were not doing well, tended to end after just about four rounds, with high training data loss (~0.45). Also, when models are doing well, you should see the learning rate really ramp up, otherwise the model is 'not getting any traction'.

Once we have created a model, we have a few options of what to do with it:

- evaluation functions: ML.EVALUATE and ML.ROC_CURVE (which only applies to logistic regression models)
- prediction function: ML.PREDICT
- inspection functions: ML.TRAINING_INFO, ML.FEATURE_INFO, and ML.WEIGHTS

Below is an example "query" showing how to use the ML.EVALUATE function: here I am evaluating the model on a random subset (half) of the data that I used to train the model. (In a more rigorous scenario, you would of course either do cross-fold validation or evaluate on a subset of the data that was not used at all during training.)

```
#standardSQL
SELECT
  *
FROM
  ML.EVALUATE(MODEL `isb-cgc-02-0001.tcga_model_1.coad_vs_paad_expr_l1_l2`, (
SELECT
  label,
  CCNE1,
  CDC6,
```

(continues on next page)

Model Info

Model ID	isb-cgc-02-0001:tcga_model_1.coad_vs_paad_expr_l1_l2
Table Size	40 B
Creation Time	Jul 30, 2018, 10:00:18 AM
Last Modified	Jul 30, 2018, 10:01:17 AM
Expiration Time	Never Edit
Data Location	US
Model Type	Logistic regression
Loss Type	Mean log loss
Labels	None Edit

Training Options

Max Allowed Iterations	20
Actual Iterations	9
L1 Regularization	1
L2 Regularization	1
Early Stop	true
Minimum Relative Progress	0.01
Learn Rate Strategy	Line search
Line Search Initial Learn Rate	0.1

Model Details: coad_vs_paad_expr_l1_l2

Training Run: 1

Details | Training Stats | Schema

Row	Iteration	Training Data Loss	Evaluation Data Loss	Learn Rate	Completion Time (seconds)
1	8	0.27	0.26	6.40	6.45
2	7	0.27	0.28	25.60	4.99
3	6	0.29	0.29	12.80	4.89
4	5	0.32	0.32	6.40	5.83
5	4	0.37	0.36	3.20	6.12
6	3	0.44	0.44	1.60	4.91
7	2	0.53	0.53	0.80	6.12
8	1	0.61	0.61	0.40	5.44
9	0	0.66	0.66	0.20	7.05

(continued from previous page)

```
  MDM2,
  TGFA
FROM
  `isb-cgc-02-0001.tcga_model_1.paad_coad_expr_2`
WHERE
  RAND() < 0.5
  )
 )
```

Results | Details

Row	precision	recall	accuracy	f1_score	log_loss	roc_auc
1	0.7636363636363637	0.6923076923076923	0.8854041013268998	0.7262247838616714	0.26973732767854014	0.942319

Table | JSON

One last thing, we can get the weights (or model coefficients) by again querying the model.

```
SELECT
  *
FROM
  ML.WEIGHTS(MODEL `isb-cgc-02-0001.tcga_model_1.coad_vs_paad_expr_2`)
```

From the magnitude of the weights, we can see that CDC6 was very useful, but MDM2 wasn't. It's a great way of testing the use of each variable for the particular problem at hand.

Neat! Let's do one more example, this time using the somatic mutation data for a couple of well-known genes (above we used gene expression data). For this training data, I'm going to do a little feature engineering (see also this). Our "engineering" will simply consist of combining two columns to create a new "feature".

Results Details

Row	processed_input	weight	category_weights.category	category_weights.weight
1	CCNE1	-0.7842583407600104		
2	CDC6	-2.2759546912451274		
3	MDM2	-0.015306280787606256		
4	TGFA	0.6158875549267074		
5	__INTERCEPT__	-2.4112478267421085		

Table JSON

One tricky aspect of working with the somatic mutation data table is that the table only contains rows describing observed somatic mutations. The *lack* of a mutation at a particular position in a gene, for a particular sample, is only implied by the lack of such a row in the table. It is also possible that a mutation occurred but was missed by the sequencing or the mutation-calling pipeline. We will assume, however, that all samples that are mentioned at least once in the table can be assumed to have been "tested" for all of the mutations that are present in the table. We will therefore start with a list of *all* of the samples we are interested in, and then join in the mutation data using a LEFT JOIN, which will retain all of the same barcodes. At the end, after selecting for a number of different genes, we join subtables for each gene of interest.

```
WITH

-- First we make a table with all barcodes for the two cancer types.
-- (This table will have 579 rows, one per tumor sample.)
all_barcodes AS (
SELECT
  project_short_name AS label,
  sample_barcode_tumor AS barcode
FROM
  `isb-cgc.TCGA_hg38_data_v0.Somatic_Mutation_DR10`
WHERE
  project_short_name IN ('TCGA-COAD', 'TCGA-PAAD')
GROUP BY 1, 2 ),

-- Then we make a table of mutations, concatenating strings into new features.
-- For now, we'll do this only for the APC gene, but we could easily
-- add more genes or change the gene being tested for.
--
-- This table will have 444 rows, describing APC mutations in 313 tumor samples.
-- For example, sample TCGA-AA-A010-01A has 3 mutations, TCGA-AA-AA017-01A has one,
-- and TCGA-AA-A01P-01A has none and is not present in this table.
apc_mut AS (
SELECT
  project_short_name AS label,
  sample_barcode_tumor AS barcode,
  CONCAT(Hugo_Symbol, ' Mut') AS Variant,
  CONCAT(Hugo_Symbol, ' ', Variant_Classification) AS Variant_Classification,
  CONCAT(Hugo_Symbol, ' ', Variant_Type) AS Variant_Type
from
```

(continues on next page)

(continued from previous page)

```
  `isb-cgc.TCGA_hg38_data_v0.Somatic_Mutation_DR10`
WHERE
  project_short_name IN ('TCGA-COAD','TCGA-PAAD') AND Hugo_Symbol='APC'
GROUP BY 1, 2, 3, 4, 5),

-- Left Join all the barcodes, with barcodes that have mutation data.
apc_join AS (
SELECT
  b.label,
  b.barcode,
  m.Variant AS apc,
  m.Variant_Classification AS apcvarclass,
  m.Variant_Type AS apcvartype
FROM
  all_barcodes b
LEFT JOIN
  apc_mut m
ON
  b.barcode=m.barcode AND b.label=m.label
GROUP BY 1, 2, 3, 4, 5 ),

-- Finaly, we do need to replace the NULL values since the ML functions
-- cannot be trained with NULL values
apc_final AS (
SELECT
  label, barcode, apc, apcvarclass, apcvartype
FROM
  apc_join
WHERE
  apc IS NOT NULL
UNION ALL
SELECT
  label, barcode,
  "APC WT" AS apc,
  "APC WT" AS apcvarclass,
  "APC WT" AS apcvartype
FROM
  apc_join
WHERE
  apc IS NULL )
```

So, if a sample doesn't have a mutation in APC, the nulls are replaced with the "APC WT" strings, and otherwise, the features specify the mutation class and type, *eg*:

```
Row label      barcode            apc       apcvarclass             apcvartype
1   TCGA-COAD  TCGA-AD-6965-01A   APC WT    APC WT                  APC WT

-- But if you select a tumor with a mutation in APC you get:

1   TCGA-COAD  TCGA-AA-A010-01A   APC Mut   APC In_Frame_Del        APC DEL
2   TCGA-COAD  TCGA-AA-A010-01A   APC Mut   APC Nonsense_Mutation   APC SNP
3   TCGA-COAD  TCGA-AA-A010-01A   APC Mut   APC Intron              APC SNP
```

Let's repeat the above, but this time with KRAS, and then join the two results so that we have data for both KRAS and APC. We will save this table in our new dataset with the name *apc_kras_data*. (The apc_join table should have 710 rows containing information for 579 barcodes, and the kras_join table should have 589 rows containing information for the same 579 barcodes. Joining these two will produce a table with 721 rows. 105 of these rows represent tumor samples with no mutations in either APC or KRAS; 167 rows represent tumor samples with mutations only in KRAS; 223 rows represent tumor samples with mutations only in APC; and finally 226 rows represent samples with mutations in *both* APC and KRAS.)

```
SELECT
  k.label, k.barcode,
  apc, apcvarclass, apcvartype,
  kras, krasvarclass, krasvartype
FROM
  kras_final k
JOIN
  apc_final a
ON
  k.label=a.label AND k.barcode=a.barcode
```

Then we create our model:

```
#standardSQL
CREATE MODEL `isb-cgc-02-0001.tcga_model_1.apc_kras_model`
OPTIONS(
model_type='logistic_reg', l1_reg=1, l2_reg=1
) AS
SELECT
  label,
  apc, apcvarclass, apcvartype,
  kras, krasvarclass, krasvartype
FROM
  `isb-cgc-02-0001.tcga_model_1.apc_kras_data`
```

and we can evaluate it:

```
#standardSQL
SELECT
  *
FROM
  ML.EVALUATE(MODEL `isb-cgc-02-0001.tcga_model_1.apc_kras_model`, (
SELECT
  label,
  apc, apcvarclass, apcvartype,
  kras, krasvarclass, krasvartype
FROM
  `isb-cgc-02-0001.tcga_model_1.apc_kras_data`
WHERE
  RAND() < 0.5
  )
 )
```

Row	precision	recall	accuracy	f1_score	log_loss	roc_auc
1	0.8068181818181818	0.7717391304347826	0.8986666666666666	0.7888888888888889	0.24625310129724287	0.916799

Let's take a look at the model weights. Our model wound up having 20 weights: 11 for APC-related features and 9 for KRAS-related features, and it is clear that the APC-related features are the most informative for this particular task.

```
SELECT
  w.category, w.weight
FROM
  ML.WEIGHTS(MODEL `isb-cgc-02-0001.tcga_model_1.apc_kras_model`),
  UNNEST(category_weights) AS w
GROUP BY 1, 2
HAVING
  category LIKE "APC%"
ORDER BY
  ABS(weight) DESC
```

category	weight
APC Mut	-0.8397549393679742
APC Nonsense_Mutation	-0.8347451580229922
APC INS	-0.8118321576319595
APC Frame_Shift_Ins	-0.8048143387523083
APC DEL	-0.8006457455945754
APC Frame_Shift_Del	-0.8006457455945754
APC SNP	-0.7599227764016531
APC WT	0.5010719567029974
APC Splice_Site	-0.18532465379406166
APC Missense_Mutation	-0.06318992845640045
APC Silent	0.0

category	weight
KRAS WT	-0.5796828516555209
KRAS Silent	-0.14751711019807756
KRAS Mut	-0.03267322767394467
KRAS SNP	-0.012844676082355292
KRAS DEL	0.0
KRAS Nonsense_Mutation	0.0
KRAS INS	0.0
KRAS 3'UTR	0.0
KRAS Missense_Mutation	0.0

OK, pretty cool! We see some of the features have very little information and have weights of zero (or close to zero) like 'APC Silent' or most of the KRAS-related features, while others seem very important. You could test this by removing a specific feature from your input data, and then checking to see if the model statistics change.

Overall, that seems pretty useful! Of course, BigQuery ML is in beta, and our experiance with Google products: expect things to change! Have you found any good tricks? If you have, please share them via email or on twitter (@isb-cgc)!

June, 2018

Processing bam files using WDL, scatter, and Cromwell

In the last two editions, we've described a multi-step workflow for generating statistics from bam files (from ENCODE) using the common workflow language (CWL). This month, we've translated the example to WDL (workflow description language) and moved to executing the workflow using Cromwell, a 'workflow management system' that can operate in the Google cloud.

So again, starting with a collection of bam files, we're going to bin sequence reads by GC content, and produce a single output file summarizing all the input files.

Using the same dockerized tools as last time (tool reuse!) , we're going to be using Cromwell to run the workflows. You can find installation instructions here. Also install 'womtool', availble in the same location as cromwell.

The plan:

- compute some statistics over a list of bam files with samtools
- use grep to parse out a portion of the stats output

- use cut to process columns from the output
- use cat to gather the outputs into a single file

Each of these steps is now defined as a WDL task, and together they make a workflow.

Let's look at the first task:

```
task samtools_stats_tool {
    File filein
    String filename

    runtime {
      docker: "biocontainers/samtools"
    }
    command {
        samtools stats ${filein} > ${filename}_gc_stats.txt
    }
    output {
         File statsout = "${filename}_gc_stats.txt"
    }
}
```

In this task, we have two input parameters, a file and a string. Then, we define the runtime environment (AKA the docker image). This is followed by the actual command we would use in running the job. If the command needs to be split across multiple lines, just use the '' to end each line (just like in a terminal). Notice we reference the parameters with a ${}. We are reading ${filein} and writing to ${filename}. Last, we have the output of the task, setting an output variable to be referenced in other tasks.

The next three tasks follow this same form: input parameters (usually a file), runtime definition, command, and output. Here's what they look like:

```
task grep_tool {
    File grepin

    runtime {
      docker: "biocontainers/samtools"
    }

    command {
        grep --with-filename '^GCF' ${grepin} > grep_out.txt
    }

    output {
        File grepout = "grep_out.txt"
    }
}


task cut_tool {
    File cutin

    runtime {
      docker: "biocontainers/samtools"
    }
```

(continues on next page)

(continued from previous page)

```
    command {
      cut -d '/' -f 9- ${cutin} > cut_out.txt
    }

    output {
        File cuttoolout = "cut_out.txt"
    }

}


task cat_tool {
    Array[File] filesin

    runtime {
      docker: "biocontainers/samtools"
    }

    command {
        cat ${sep=" " filesin} > final_gc_stats_out.txt
    }

    output {
        File finalfile = "final_gc_stats_out.txt"
    }
}
```

The stats tool, grep tool, and cut tool all take a single file, while the cat tool (as you might expect) takes an array of files. Additionally, in this case, the output of grep was different compared to running the workflow in CWL, so the cut tool command changed to account for that.

The next 'task' for *us* is to take these task-definitions and connect them together into a workflow. In this example, I've got a tab separated file with two columns. The first column has google bucket paths to bam files, and the second column is a file label (see below). That said, the input parameter to the workflow, is just telling the workflow where the list of bam files is.

```
workflow gcStats {

    File inputSamplesFile

    Array[Array[String]] inputSamples = read_tsv(inputSamplesFile)

    scatter (sample in inputSamples) {
     call samtools_stats_tool {
       input:
         filein=sample[0],
         filename=sample[1]
       }
     }

     scatter (statout in samtools_stats_tool.statsout) {
      call grep_tool {
        input:
```

(continues on next page)

(continued from previous page)

```
          grepin = statout
      }
     }

     scatter (grepped in grep_tool.grepout) {
       call cut_tool {
            input:
              cutin = grepped
       }
     }

     call cat_tool {
       input:
         filesin = cut_tool.cuttoolout
     }

} # end workflow
```

In calling the first tool, we perform a scatter operation over input files. For the next tools, we perform scatter operations over the previously called tool outputs. The last tool (the cat tool) gets an array of files as an input, and concatenates them.

The task definitions and the workflow are placed into the same file, here named 'gcstats.wdl'. We can validate the workflow by calling:

```
java -jar womtool-32.jar validate gcstats.wdl
```

If it's valid, there's no errors reported.

The list of bam files is stored in a file named 'bamfiles.txt'. Below is the file listing.

```
gs://daves-cromwell-bucket/bamfiles/wgEncodeUwRepliSeqBg02esG1bAlnRep1.bam      bam1
gs://daves-cromwell-bucket/bamfiles/wgEncodeUwRepliSeqBjG1bAlnRep1.bam  bam2
gs://daves-cromwell-bucket/bamfiles/wgEncodeUwRepliSeqBjG2AlnRep1.bam   bam3
gs://daves-cromwell-bucket/bamfiles/wgEncodeUwRepliSeqBg02esG1bAlnRep1.bam      bam4
```

The workflow input then refers to the list of bam files. This file is named 'gcstats.input'. A template for the json input can be generated, and filled in using this command:

```
java -jar womtool-32.jar inputs gcstats.wdl  > gcstats.inputs
```

After it's filled in, the file looks like:

```
{
        "gcStats.inputSamplesFile": "gs://daves-cromwell-bucket/bamfiles.txt"
}
```

Then, a new bucket was created which serves as the root for the cromwell execution. In that bucket, 'daves-cromwell-bucket', I created another folder called 'bamfiles' and placed the data. In the root I placed the bamfile list 'bamfiles.txt'.

We're almost ready to run! But first we need to deal with authorization. So, to do that, all the instructions for 'Configuring a Google Project' need to be followed here. That configuration is saved in a file named 'google.conf'. Make sure you can run the 'hello.wdl' example.

FINALLY, now we're ready to run with this command:

Buckets / daves-cromwell-bucket

Name	Size	Type
bamfiles.txt	312 B	text/plain
bamfiles/	–	Folder

```
java -Dconfig.file=google.conf -jar cromwell-32.jar run gcstats.wdl -i gcstats.inputs
```

This command starts up a VM in the Google cloud, runs the tasks in parallel, and writes the output to your bucket. The resulting directory 'cromwell-execution' looks like this:

Buckets / daves-cromwell-bucket / cromwell-execution

Name	Size	Type
gcStats/	–	Folder
wf_hello/	–	Folder

The take outputs are organized (under an unreadable folder name) by name:

Buckets / daves-cromwell-bucket / cromwell-execution / gcStats / 1ba9e249-2f57-4fb9-b5df-899e91e437ac

Name	Size	Type	Storage class	Last modified
call-cat_tool/	–	Folder	–	–
call-cut_tool/	–	Folder	–	–
call-grep_tool/	–	Folder	–	–
call-samtools_stats_tool/	–	Folder	–	–

And if we look inside our cat_tool folder, we see the final output file. Cool.

That's it! We've run a multi-step workflow, with steps in parallel, written in WDL, and run with Cromwell in the google cloud. Not too bad, right? Were you able to run your own workflow? Let us know!

May, 2018

Processing bam files using CWL 'scatter and gather'

In this edition, we're going to continue our exploration of using CWL to run workflows on the Google cloud. Last time, we performed a 'scatter' operation, where a tool is applied to a list of files. This time, we'll complete the paradigm by performing a 'gather' to collate the results of a scatter. Additionally, we will propagate the scatter through a series of steps.

Specifically, for a list of files, we're going to bin sequence reads by GC content, producing a single output file that we can use to make a plot.

The tools are found in this docker image

The plan:

- compute some statistics over a list of bam files with samtools

Buckets / daves-cromwell-bucket / cromwell-execution / gcStats / 1ba9e249-2f57-4fb9-b5df-899e91e437ac / call-cat_tool

Name	Size	Type	Storage class	Last modified
cat_tool-rc.txt	2 B	text/plain	Regional	6/28/18, 5:39 PM
cat_tool-stderr.log	0 B	text/plain	Regional	6/28/18, 5:39 PM
cat_tool-stdout.log	0 B	text/plain	Regional	6/28/18, 5:39 PM
cat_tool.log	7.28 KB	text/plain	Regional	6/28/18, 5:39 PM
exec.sh	1.26 KB	text/plain	Regional	6/28/18, 5:37 PM
final_gc_stats_out.txt	126 B	text/plain	Regional	6/28/18, 5:39 PM

- use grep to parse out a portion of the stats output
- use cut to select some columns from the output
- use cat to gather the outputs into a single file

Each of these steps is defined as a CWL tool, and together they make a workflow.

The first three steps of the workflow are considered scatter operations, and the last is the gather, where outputs are combined.

Let's look at the first tool:

```
#samtools_stats_tool.cwl

cwlVersion: v1.0
class: CommandLineTool

baseCommand: [samtools, stats]

requirements:
  - class: InlineJavascriptRequirement

inputs:
  filein:
    type: File
    inputBinding:
      position: 1

outputs:
  statsout:
    type: File
    outputBinding:
      glob: "*.stats"

stdout: $(inputs.filein.path.split('/').pop() + '.stats')
```

This tool definition is going to compute several different statistics for each bam file. The statistic-type is delineated by a column label. An interesting thing here, is that the standard output, usually printed to the screen, is captured and saved using the input file name with '.stats' added on, and the output looks for that '.stats' with the file glob. We want to hold onto the file name to use as a label in the final results.

The next tool is going to parse out our statistic of interest, the GC content.

```
#grep_tool.cwl

cwlVersion: v1.0

class: CommandLineTool

baseCommand: grep

arguments:
  - "--with-filename"
  - "^GCF"

inputs:
  input_file:
    type: File
    inputBinding:
      position: 1

outputs:
  grepout:
    type: stdout
```

This is a very general tool that could be applied in many settings… it's just grep! Grep is a pattern matching tool, so each line of text that starts with 'GCF' is printed. Also we're going to add the input file name to each line (–with-filename). Interesting thing here: since we don't explicitly define the file name for the standard output (stdout), the file name is random. We can let the workflow runner worry about it.

The next tool wraps the cut command.

```
#cut_tool.cwl

cwlVersion: v1.0

class: CommandLineTool

baseCommand: cut
arguments:
  - "-d "
  - "-f"
  - "1,5-"

inputs:
  input_file:
    type: File
    inputBinding:
      position: 1

outputs:
  cutout:
    type: stdout
```

Here, we define some 'arguments' to the baseCommand. The '-d ' sets the delimiter to white space, '-f 1,5-' says we want fields (-f) 1, and 5+. Column 1 is the file name, and the stats of interest appear in columns 5 and beyond. Again we don't define any file names for the output.

Lastly, we are going to gather all the results.

```
#cat_tool.cwl

cwlVersion: v1.0

class: CommandLineTool

baseCommand: cat

inputs:
  filein:
    type: File[]
    inputBinding:
      position: 1

outputs:
  catout:
    type: stdout

stdout: final_output.txt
```

It's important to note that in the cat tool, we have defined the input to be an array of files. The command becomes 'cat file.a file.b file.c'. And here we *do* define the output file name.

OK! Let's work these tools into a flow.

Important note: I developed all these tools on my local machine with small test cases. When I was ready to test on the google cloud, the only change I made was adding the docker hint (DockerRequirement, see below), and moving tool definitions (the cwl files) data into a bucket. It was surprisingly easy to move from the local environment to the cloud.

The main workflow:

```
#!/usr/bin/env cwl-runner

cwlVersion: v1.0
class: Workflow

requirements:
  ScatterFeatureRequirement: {}

hints:
  DockerRequirement:
    dockerPull: biocontainers/samtools

inputs:
  filein: File[]

outputs:
  pipeline_result:
    type: File
    outputSource: step4/catout

steps:
```

(continues on next page)

(continued from previous page)

```
  step1:
    run: data/samtools_stats_tool.cwl
    scatter: [filein]
    scatterMethod: dotproduct
    in:
      filein: filein
    out:
      [statsout]

  step2:
    run: data/grep_tool.cwl
    scatter: [input_file]
    scatterMethod: dotproduct
    in:
      input_file: step1/statsout
    out:
      [grepout]

  step3:
    run: data/cut_tool.cwl
    scatter: [input_file]
    scatterMethod: dotproduct
    in:
      input_file: step2/grepout
    out:
      [cutout]

  step4:
    run: data/cat_tool.cwl
    in:
      filein: step3/cutout
    out:
      [catout]
```

You will notice, that we only define an array of bam files as inputs to the workflow (File[]), the intermediates are carried through without explicitly naming them. Inputs to the middle steps point to the outputs of previous steps (step1/statsout -> step2).

And we need to define our input files (scatter_gather_pipeline.yml):

```
filein:
  - {class: File, path: data/wgEncodeUwRepliSeqBg02esG1bAlnRep1.bam}
  - {class: File, path: data/wgEncodeUwRepliSeqBjG1bAlnRep1.bam}
  - {class: File, path: data/wgEncodeUwRepliSeqBjG2AlnRep1.bam}
```

To run this, we use the google_cwl_runner found here.

I made a working folder in my google bucket called 'workflow-1', and two additional folders within, 'data' and 'output'. I put the bam files and the cwl tool definitions into 'data' and I put the workflow file in the top level of my working folder.

```
./examples-Compute/google_cwl_runner/cwl_runner.sh \
     --workflow-file gs://my-bucket/workflow-1/scatter_gather_pipeline.cwl \
```

(continues on next page)

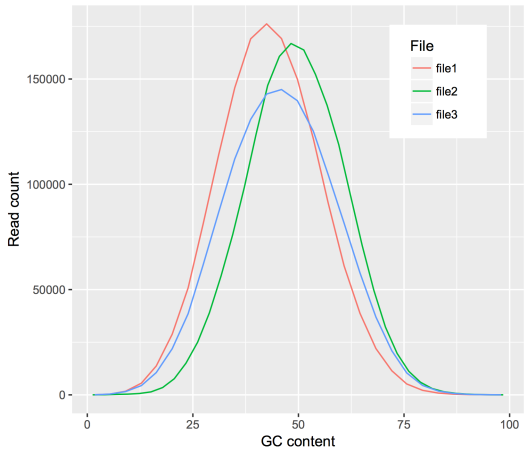
(continued from previous page)

```
    --settings-file gs://my-bucket/workflow-1/scatter_gather_pipeline.yml \
    --input-recursive gs://my-bucket/workflow-1/data \
    --output gs://my-bucket/workflow-1/output \
    --machine-type n1-standard-4 \
    --zone us-west1-a \
    --preemptible
```

Running this command prints out some google cloud commands to enable us to check on the status of the job, and when it's finished we have logs for stderr, stdout, and status. Additionally we get the cwl_startup.sh, cwl_runner.sh, and cwl_shutdown.sh scripts for your perusal.

But most importantly(!), we get our single output file containing the binned GC content (shown below). The first column is the file name that produced the result, the second column is the percent GC, and the last column is the number of reads.

```
/long.tmp.path/wgEncodeUwRepliSeqBg02esG1bAlnRep1.bam.stats:GCF 1.26    22
/long.tmp.path/wgEncodeUwRepliSeqBg02esG1bAlnRep1.bam.stats:GCF 4.02    66
/long.tmp.path/wgEncodeUwRepliSeqBg02esG1bAlnRep1.bam.stats:GCF 6.78    232
/long.tmp.path/wgEncodeUwRepliSeqBg02esG1bAlnRep1.bam.stats:GCF 9.55    349
/long.tmp.path/wgEncodeUwRepliSeqBg02esG1bAlnRep1.bam.stats:GCF 12.31   647
/long.tmp.path/wgEncodeUwRepliSeqBg02esG1bAlnRep1.bam.stats:GCF 15.08   1413
/long.tmp.path/wgEncodeUwRepliSeqBg02esG1bAlnRep1.bam.stats:GCF 17.84   3473
/long.tmp.path/wgEncodeUwRepliSeqBg02esG1bAlnRep1.bam.stats:GCF 20.60   7676
/long.tmp.path/wgEncodeUwRepliSeqBg02esG1bAlnRep1.bam.stats:GCF 23.37   15064
...
```



Thanks for reading! Let us know if you have questions or comments!

April, 2018

For the next few months, we're going to be focusing on running workflows in the Google cloud, starting with workflows defined with CWL. There's a lot to explore in this area, and it's hopefully going to be useful for people.

We appear to be moving towards a future where bioinformatics tools are routinely bundled into runtime containers (like docker), which are somehow 'approved' by the research community as being safe and effective, launched in the cloud, process data in the cloud, and write results to the cloud. Sounds great, but how does one do that?

We previously documented some very low level details on running CWL workflows, but to make things easier, we started with some code from Google, made some updates to the scripts, and with them you can (fairly) easily run CWL workflows in the cloud. The current working google CWL runner is found in our examples-Compute repo.

In this example, we'll be using a workflow (transform.cwl) from the GDC that was used to 'harmonize' TCGA data.

The CWL workflow has a few sections: requirements, inputs, outputs, and steps. The purpose of the workflow is to execute the steps, where each step is a tool (also described by a CWL) which has a set of inputs and outputs.

In transform.cwl, the first step is:

```
steps:
  - id: samtools_bamtobam
    run: ../../tools/samtools_bamtobam.cwl
    in:
      - id: INPUT
        source: bam_path
    out:
      - id: OUTPUT
```

Let's look more closely at this first step (samtools_bamtobam.cwl) and see if we can run it; first on one file, and then on a set of files.

single file

To make the example a little more accessible, I've rewritten the CWL, as seen below.

```
#!/usr/bin/env cwl-runner

cwlVersion: v1.0

class: CommandLineTool

hints:
  DockerRequirement:
    dockerPull: biocontainers/samtools

baseCommand: [samtools, view, -Shb]

inputs:
  filein:
    type: File
    inputBinding:
      position: 1
  fileout:
    type: string
    inputBinding:
      prefix: -o
```

(continues on next page)

(continued from previous page)

```
      position: 2

outputs:
  bamsout:
    type: File
    outputBinding:
      glob: "*.bam"
```

OK, let's start at the top. This is going to run samtools, so we've defined this CWL as a CommandLineTool.

Then, since this is running in the cloud, we need the tool to be dockerized. I did a little searching, and found a biocontainer with samtools. We let CWL know this is a dockerized tool with our DockerRequirement hint.

Next we have the baseCommand, which is a command line command that's broken up by commas. What's samtools view doing? Well, if we look in Aaron Quinlan's github repo: "The samtools view command is the most versatile tool in the samtools package. It's main function, not surprisingly, is to allow you to convert the binary (i.e., easy for the computer to read and process) alignments in the BAM file view to text-based SAM alignments that are easy for humans to read and process." The 'view' command, with the -Shb flags will (-S is depreciated) -h includes the header, -b outputs a bam. This is simply a bam-to-bam workflow.

We will 'view' an input file, named 'filein', and output a file, named 'fileout' (creative huh?). But those are the inputs, or arguments, that I'm giving to samtools. The order of the samtools parameters is controlled by the position, and in the case of 'fileout', I'm attaching a prefix '-o' (the output flag).

We still have the CWL outputs section. This is actually what gets connected up to a workflow (a series of steps). Here I've called it 'bamsout', and declare the file will be named 'something dot bam' (*.bam).

OK then, still with me? Next I set up a google bucket with data and output folders. This is also where the CWL files go.

```
my-bucket/
    bamtobam/
        samtools_bamtobam_single_file.cwl
        data/
            bam1.bam
            bam2.bam (etc)
        outputs/
```

To get some data to work on, I moved some bams from the ENCODE project to my bucket's data folder.

The last thing we need is a settings file. This maps actual file names in our bucket and parameter values to variable names found in the CWL. The paths are relative to the location to the CWL file. Here's what I used (in the yaml format):

```
filein:
  class: File
  path: data/wgEncodeUwRepliSeqBjG1bAlnRep1.bam
fileout: samoutput.bam
```

You can see the filein and fileout names match up with what's in the CWL. This is important.

Then after cloning our repo (git clone https://github.com/isb-cgc/examples-Compute), we can run it! The cwl_runner.sh script sets up a new VM, runs the cwl_startup.sh script, attaches a disk, copies over the data, runs the CWL, copies out the data, and shuts everything down by running the cwl_shutdown.sh script. Here's the command:

```
./examples-Compute/google_cwl_runner/cwl_runner.sh \
  --workflow-file gs://my-bucket/bamtobam/samtools_bamtobam_single_file.cwl \
```

(continues on next page)

(continued from previous page)

```
  --settings-file gs://my-bucket/bamtobam/bamtobam_params.yml \
  --input-recursive gs://my-bucket/bamtobam/data \
  --output gs://isb-cgc-02-0001-workflows/bamtobam/output \
  --machine-type n1-standard-4 \
  --zone us-central1-f \
  --keep-alive \
  --preemptible
```

Notice we can use preemptible machines to save money. So running this command reads the bam and writes out a new bam, which is found in our bucket after a few minutes.

debugging

If something geos wrong (it probably will), there's a few things we can do. First, check the logs! The stderr and stdout is copied back to our bucket, and within those files we can find errors. Commonly (for me) paths can be wrong.

Second, you can try modifying the cwl_startup.sh script to include some print statements. I've put in some statements to print out directory listings, which gives you a view into the runtime environment.

Third, towards the bottom of the stderr, you can see the actual command that's run. Then, if you use the –keep-alive flag (given to cwl_runner.sh), the VM will not shutdown, and you can get in there and try the command and see what happens.

multiple files

It's much more common to run a tool over a set of files. To do that, we're going to take our tool definition above, and write a new workflow that uses it. In CWL terms, this is called a 'scatter'. Later we'll perform the 'gather', but for now we'll process the files, and write them back to our bucket.

```
#!/usr/bin/env cwl-runner

cwlVersion: v1.0

class: Workflow

requirements:
  ScatterFeatureRequirement: {}

inputs:
  filein: File[]
  fileout: string[]

outputs:
  bamsout:
    type: File[]
    outputSource: step1/bamsout

steps:
  step1:
    run: data/samtools_bamtobam_single_file.cwl
    scatter: [filein, fileout]
    scatterMethod: dotproduct
    in:
      filein: filein
      fileout: fileout
```

(continues on next page)

(continued from previous page)

```
    out:
      [bamsout]
```

There's some big differences between this definition and our tool definition. For starters we have a new requirements: ScatterFeatureRequirement which let's us perform a scatter (process multiple files in parallel).

Then in our input section, we have defined an array of files and strings. In the output section, an array of output files, noting that the source of these outputs is from step1 (outputSource: step1/bamsout).

The steps section has only a single step. We'll run our previously defined tool, as a scatter over files and parameters (the output file name), with scatterMethod dotproduct (match up the input and output file names), and we have in: and out: names that match up with the tool definition. Very important that names connect across the CWL definitions. It has to all fit together like legos.

Our settings file, the yaml, is also different because now we're going to be giving it a list of files and output file names. Note the use of 'dictionaries' with the curly brackets.

```
filein:
  - {class: File, path: data/wgEncodeUwRepliSeqBjG1bAlnRep1.bam}
  - {class: File, path: data/wgEncodeUwRepliSeqGm12801G1bAlnRep1.bam}
  - {class: File, path: data/wgEncodeUwRepliSeqGm12878S4AlnRep1.bam}
fileout:
    - wgEncodeUwRepliSeqBjG1bAlnRep1_OUT.bam
    - wgEncodeUwRepliSeqGm12801G1bAlnRep1_OUT.bam
    - wgEncodeUwRepliSeqGm12878S4AlnRep1_OUT.bam
```

And again we run it the same way, except, note that we put our tool definition **into** the data folder so it gets copied over with the data.

```
./pipelines-api-examples/cwl_runner/cwl_runner.sh \
  --workflow-file gs://isb-cgc-02-0001-workflows/bamtobam/samtools_bamtobam_scatter.cwl \
  --settings-file gs://isb-cgc-02-0001-workflows/bamtobam/bamtobam_params_scatter.yml \
  --input-recursive gs://isb-cgc-02-0001-workflows/bamtobam/data \
  --output gs://isb-cgc-02-0001-workflows/bamtobam/output \
  --machine-type n1-standard-4 \
  --zone us-central1-f \
  --preemptible
```

If all goes well, you should see:

Whew! It's a really steep learning curve, but the payoff is that in two years, when you wonder 'how did I do this?', you can look back and easily figure that out and probably (maybe) even run it again.

Next month we'll continue our exploration of workflows and workflow runners. Let me know how it goes!

March, 2018

This month we demonstrate an implementation of a machine learning classifier using BigQuery.

The 'Top Scoring Pairs' method, finds a pair of genes that show the maximum difference in ranking between two user specified groups. Given two genes, and two groups of samples, the ranking of the genes flip-flops between the two groups. If gene_i < gene_j in group 1, then gene_i > gene_j in group 2.

To describe this more formally, let R_i be a vector of ranks denoting the rank of the i-th gene in a given sample.

Buckets / isb-cgc-02-0001-workflows / bamtobam / output

Name	Size	Type	Storage class
cwl_runner-38290.sh	913 B	application/x-sh	Regional
cwl_shutdown-38290.sh	1.38 KB	application/x-sh	Regional
cwl_startup-38290.sh	5.82 KB	application/x-sh	Regional
status-38290.txt	10 B	text/plain	Regional
stderr-38290.txt	13.72 KB	text/plain	Regional
stdout-38290.txt	52.11 KB	text/plain	Regional
wgEncodeUwRepliSeqBjG1bAlnRep1_OUT.bam	62.14 MB	application/octet-stream	Regional
wgEncodeUwRepliSeqGm12801G1bAlnRep1_OUT.bam	62.12 MB	application/octet-stream	Regional
wgEncodeUwRepliSeqGm12878S4AlnRep1_OUT.bam	58.67 MB	application/octet-stream	Regional

Genes are evaluated in pairs, and scored by their differences in the probabilities, P(R_i < R_j), between class C1 and class C2, formally defined as the difference in the following conditional probabilities:

_ij = P(Ri < Rj C1) P(Ri < Rj C2)

Then _ij is used as a criterion to produce a ordering on gene pairs.

For reference see this.

First let's produce some simulated data for testing.

```
# phenotype
ys <- c(rep(1, 10), rep(0,10))

# matrix of features
xs <- matrix(data=rnorm(n=200), nrow=20)

# the IDs
idstr <- as.character(randomStrings(n=nrow(df), len=5, digits=FALSE,
                      upperalpha=TRUE,loweralpha=FALSE, unique=TRUE, check=TRUE))

# the two best genes
i <- 5
j <- 6

# create the gene pair
xs[ys == 1, i] <- rnorm(n=10, mean=-2)
xs[ys == 1, j] <- rnorm(n=10, mean=+2)
xs[ys == 0, i] <- rnorm(n=10, mean=+2)
xs[ys == 0, j] <- rnorm(n=10, mean=-2)
df <- data.frame(IDs=idstr, Y=ys, xs)

tidydf <- df %>% tidyr::gather(key="Gene", value="Expr", X1,X2,X3,X4,X5,X6,X7,X8,X9,X10)

write.table(tidydf, file="sim_for_tsp.tsv", sep='\t', row.names=F, col.names=F, quote=F)
```

The results should show genes X5 & X6 as the best pair for separating groups (y==0 vs y==1).

OK, let's walk through this Top Scoring Pairs (TSP) query.

```
WITH
  #
  # First we'll rank the gene expression data by sample
  # using the simulated data.
  #
  GeneRanks AS (
  SELECT
    ID,
    Phenotype,
    Gene,
    Expr,
    RANK() OVER (PARTITION BY ID ORDER BY Expr ) AS ERank
  FROM
    `isb-cgc.QotM.tsp_sim_data`
  GROUP BY
    ID,
    Phenotype,
    Gene,
    Expr ),
  #
  # Then let's prepare to
  # generate the conditional probability for each
  # pair of genes within Class 1.
  # Calculating the upper triangle here.
  #
  Class1GenePairs AS (
  SELECT
    a.Gene AS Genei,
    b.Gene AS Genej,
    a.ID AS IDi,
    b.ID AS IDj,
    a.ERank AS Eranki,
    b.ERank AS Erankj
  FROM
    GeneRanks a
  JOIN
    GeneRanks b
  ON
    a.Gene < b.Gene
    AND a.ID = b.ID
  WHERE
    a.Phenotype = 1
    AND b.Phenotype = 1
  GROUP BY
    a.Gene,
    b.Gene,
    a.ID,
    b.ID,
    a.ERank,
    b.ERank ),
```

(continues on next page)

(continued from previous page)

```
#
# Then, for each pair of genes,
# how many times does gene_i have lower rank
# than gene_j? That's where the probability comes from.
#
Class1Probs AS (
SELECT
  Genei,
  Genej,
  SUM(CAST(Eranki > -1000 AS INT64)) AS N,  # number of pairs
  SUM(CAST(Eranki < Erankj AS INT64)) AS P  # pairs with i < j
FROM
  Class1GenePairs
WHERE
  (Genei != Genej)
GROUP BY
  Genei,
  Genej ),
#
# Then repeat the process for Class 2.
#
Class2GenePairs AS (
SELECT
  a.Gene AS Genei,
  b.Gene AS Genej,
  a.ID AS IDi,
  b.ID AS IDj,
  a.ERank AS Eranki,
  b.ERank AS Erankj
FROM
  GeneRanks a
JOIN
  GeneRanks b
ON
  a.Gene < b.Gene
  AND a.ID = b.ID
WHERE
  a.Phenotype = 1
  AND b.Phenotype = 1
GROUP BY
  a.Gene,
  b.Gene,
  a.ID,
  b.ID,
  a.ERank,
  b.ERank ),
#
# and get our conditional probabilities
#
Class2Probs AS (
SELECT
  Genei,
```

(continues on next page)

(continued from previous page)

```
    Genej,
    SUM(CAST(Eranki > -1000 AS INT64)) AS N,
    SUM(CAST(Eranki < Erankj AS INT64)) AS P
  FROM
    Class2GenePairs
  WHERE
    (Genei != Genej)
  GROUP BY
    Genei,
    Genej )
  #
  # and compute differences in conditional probs
  #
SELECT
  a.Genei AS ai,
  a.Genej AS aj,
  b.Genei AS bi,
  b.Genej AS bj,
  a.P AS Pa,
  a.N AS Na,
  b.P AS Pb,
  b.N AS Nb,
  ABS((a.P / a.N) - (b.P / b.N)) AS PDiff
FROM
  Class1Probs a
JOIN
  Class2Probs b
ON
  a.Genei = b.Genei
  AND a.Genej = b.Genej
ORDER BY
  PDiff DESC
```

Running this query returns a table that is ordered by the Probability difference (the pair of genes that best separates the classes in 'rank-space'). As a note, I left in the gene names from both tables after the join, as a sanity check.

Results Details Show Pivot Download as CSV

Row	ai	aj	bi	bj	Pa	Na	Pb	Nb	PDiff
1	X5	X6	X5	X6	0	10	10	10	1.0
2	X4	X5	X4	X5	10	10	0	10	1.0
3	X1	X6	X1	X6	0	10	10	10	1.0
4	X5	X7	X5	X7	0	10	10	10	1.0
5	X6	X7	X6	X7	9	10	0	10	0.9
6	X2	X5	X2	X5	9	10	0	10	0.9

Table JSON First < Prev Rows 1 - 6 of 45 Next > Last

And we see that genes 'X5' and 'X6' are indeed the 'top pair'. Trailing the leading spot, we see other pairs that are composed of one of the two 'top pair'.

But, let's suppose that we want to 'train' the model using a subset of samples. In that case we want to pull out a sample, train on the remainder, and then apply to the 'test' case.

```
WITH
  #
  # Now we pull out a sample (ID: DRRTF)
  # and use the remaining samples to train the model.
  #
  GeneRanks AS (
  SELECT
    ID,
    Phenotype,
    Gene,
    Expr,
    RANK() OVER (PARTITION BY ID ORDER BY Expr ) AS ERank
  FROM
    `isb-cgc.QotM.tsp_sim_data`
  WHERE
    ID != 'DRRTF'
  GROUP BY
    ID,
    Phenotype,
    Gene,
    Expr ),
```

Then, last, we will query using that sample to determine if it's been classified correctly.

```
#
# first we'll join the TSP result table, that contains the best pair of genes,
# with the expression data for our held out ID.
#
callTbl AS (
  SELECT
    ai AS gene_i,
    aj AS gene_j,
    Pa,
    Pb,
    b.ID AS ID,
    b.Phenotype AS Phenotype,
    Expr
  FROM
    PairScore a
  JOIN
    `isb-cgc.QotM.tsp_sim_data` b
  ON
    b.ID = 'DRRTF'
    AND (a.ai = b.Gene
      OR a.aj = b.Gene) )
#
# Then, depending on the gene_a & gene_b comparison,
# we make a prediction using the expr. values.
#
SELECT
  ID,
  Phenotype,
  IF(Pa < Pb,
```

(continues on next page)

(continued from previous page)

```
      0,
      1) AS Prediction
FROM
  callTbl
GROUP BY
  ID,
  Phenotype,
  Prediction
```

Results | Details

Row	ID	Phenotype	Prediction
1	DRRTF	0	0

Table | JSON

Seems to be working. Let's make a few small changes, and apply it to TCGA expression data! First we'll create our data set, then we'll apply TSP on it.

```
WITH
#
# To reduce the number of genes we're working with,
# first we'll rank the gene expression data by coefficient of variation.
# Then we'll be able to take a subset with high variance.
# Also we'll filter out some long RNAs, etc.
#
GeneSelection AS (
SELECT
  gene_name,
  STDDEV(HTSeq__FPKM_UQ) / AVG(HTSeq__FPKM_UQ) AS CVExpr
FROM
  `isb-cgc.TCGA_hg38_data_v0.RNAseq_Gene_Expression`
WHERE
  HTSeq__FPKM_UQ > 0
  AND REGEXP_CONTAINS(sample_barcode, '-01A')
  AND (project_short_name = 'TCGA-PAAD'
   OR project_short_name = 'TCGA-KIRP')
  AND (NOT (REGEXP_CONTAINS(gene_name, 'MT-')
      OR REGEXP_CONTAINS(gene_name, 'RN7')
      OR REGEXP_CONTAINS(gene_name, 'RNU')
      OR REGEXP_CONTAINS(gene_name, 'SNOR') ) )
GROUP BY
  gene_name
ORDER BY
  CVExpr DESC
LIMIT
  50 ),
#
#
# Then we'll pick a set of random samples from
# the biospecimen table. Making sure we get only
# primary tumors by filtering barcodes that don't
```

(continues on next page)

(continued from previous page)

```
# end with '-01A'.
#
SampleSelection AS (
SELECT
  project_short_name,
  sample_barcode
FROM
  `isb-cgc.TCGA_bioclin_v0.Biospecimen`
WHERE
  REGEXP_CONTAINS(sample_barcode, '-01A')
  AND (project_short_name = 'TCGA-PAAD'
    OR project_short_name = 'TCGA-KIRP')
GROUP BY
  sample_barcode,
  project_short_name
ORDER BY
  rand()
LIMIT
  200 ),
#
#
# With genes and samples, we can subset our expression data.
#
ExprTable AS (
SELECT
  sample_barcode,
  project_short_name,
  gene_name AS Gene,
  HTSeq__FPKM_UQ AS Expr
FROM
  `isb-cgc.TCGA_hg38_data_v0.RNAseq_Gene_Expression`
WHERE
  gene_name IN (
  SELECT
    gene_name
  FROM
    GeneSelection)
  AND sample_barcode IN (
  SELECT
    sample_barcode
  FROM
    SampleSelection)
GROUP BY
  sample_barcode,
  project_short_name,
  Gene,
  Expr )
#
# And we rank the gene expression and create a phenotype variable.
#
SELECT
  sample_barcode AS ID,
```

(continues on next page)

(continued from previous page)

```
  project_short_name,
  IF(project_short_name = 'TCGA-PAAD',
    0,
    1) AS Phenotype,
  Gene,
  Expr,
  RANK() OVER (PARTITION BY sample_barcode ORDER BY Expr ) AS ERank
FROM
  ExprTable
GROUP BY
  ID,
  project_short_name,
  Phenotype,
  Gene,
  Expr
```

And I'll save that ranked table in the query of the month dataset as isb-cgc.QotM.paad_kirp_random_sample_1002.

```
WITH
  #
  # First let's create the table of gene pairs.
  #
  Class1GenePairs AS (
  SELECT
    a.Gene AS Genei,
    b.Gene AS Genej,
    a.ID AS IDa,
    b.ID AS IDb,
    a.ERank AS Eranka,
    b.ERank AS Erankb
  FROM
    `isb-cgc.QotM.paad_kirp_random_sample_1002` a
  JOIN
    `isb-cgc.QotM.paad_kirp_random_sample_1002` b
  ON
    a.Gene < b.Gene
    AND a.ID = b.ID
  WHERE
    a.Phenotype = 0
    AND b.Phenotype = 0
  GROUP BY
    a.Gene,
    b.Gene,
    a.ID,
    b.ID,
    a.ERank,
    b.ERank ),
  #
  # Then, for each pair of genes,
  # how many times does gene_i have lower rank
  # than gene_j? This is how the conditional
  # probability is calculated.
```

(continues on next page)

(continued from previous page)

```
#
Class1Probs AS (
SELECT
  Genei,
  Genej,
  SUM(CAST(Eranka > -1000 AS INT64)) AS N,
  SUM(CAST(Eranka < Erankb AS INT64)) AS P
FROM
  Class1GenePairs
WHERE
  (Genei != Genej)
GROUP BY
  Genei,
  Genej ),
#
# Then pair up the genes in class2
#
Class2GenePairs AS (
SELECT
  a.Gene AS Genei,
  b.Gene AS Genej,
  a.ID AS IDa,
  b.ID AS IDb,
  a.ERank AS Eranka,
  b.ERank AS Erankb
FROM
  `isb-cgc.QotM.paad_kirp_random_sample_1002` a
JOIN
  `isb-cgc.QotM.paad_kirp_random_sample_1002` b
ON
  a.Gene < b.Gene
  AND a.ID = b.ID
WHERE
  a.Phenotype = 1
  AND b.Phenotype = 1
GROUP BY
  a.Gene,
  b.Gene,
  a.ID,
  b.ID,
  a.ERank,
  b.ERank ),
#
# and get our conditional probabilities
#
Class2Probs AS (
SELECT
  Genei,
  Genej,
  SUM(CAST(Eranka > -1000 AS INT64)) AS N,
  SUM(CAST(Eranka < Erankb AS INT64)) AS P
FROM
```

(continues on next page)

(continued from previous page)

```
    Class2GenePairs
  WHERE
    (Genei != Genej)
  GROUP BY
    Genei,
    Genej ),
  #
  # and compute differences in conditional probs
  #
  Results AS (
  SELECT
    a.Genei AS gene_i,  # gene pair #1
    a.Genej AS gene_j,  # gene pair #2
    a.P AS Pa,          # number of pairs where gene #1 < gene #2 in group A
    a.N AS Na,          # total number of pairs
    b.P AS Pb,          # numbers for group B.
    b.N AS Nb,
    ABS((a.P / a.N) - (b.P / b.N)) AS PDiff  # difference in conditional probabilities
  FROM
    Class1Probs a
  JOIN
    Class2Probs b
  ON
    a.Genei = b.Genei
    AND a.Genej = b.Genej
  ORDER BY
    PDiff DESC )
SELECT
  *
FROM
  Results
```

Results | Details | Show Pivot | Download as CSV

Row	gene_i	gene_j	Pa	Na	Pb	Nb	PDiff
1	NKX6-3	SLC12A3	9	79	114	117	0.8604349237260629
2	ALDH3B2	RIPPLY1	8	79	104	117	0.7876230661040787
3	ALDH3B2	RP11-557H15.3	7	79	102	117	0.7831872768581629
4	ALDH3B2	PRSS57	3	79	96	117	0.7825381369685167
5	NKX6-3	PRSS57	9	79	103	117	0.766417829708969

Table | JSON

First < Prev Rows 1 - 5 of 1225 Next > Last

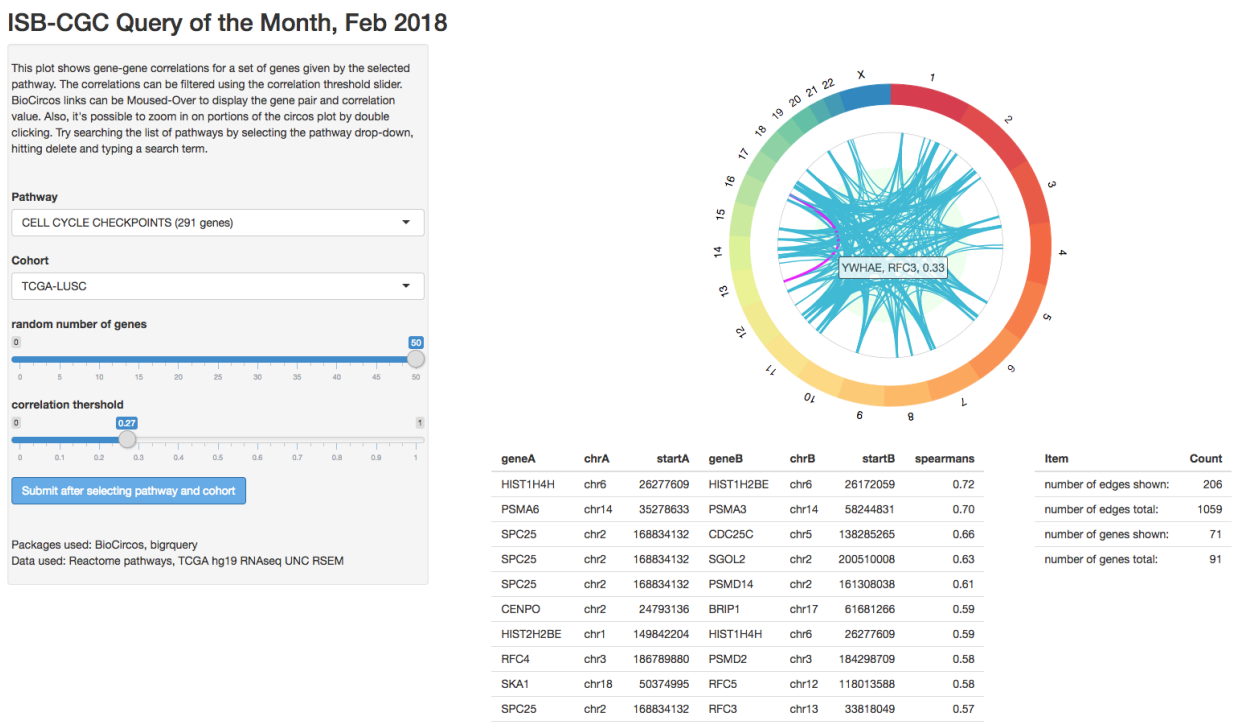
So we see that the best pair of genes for separating these two studies are NKX6-3 and SLC12A3 and 9/79 PAAD samples had expression ranking where (NKX6-3 < SLC12A3) and for KIRP samples 114/117 had a ranking where (NKX6-3 < SLC12A3). So it's doing a pretty good job!

Let's check this gene pair with another random set of samples. To do that, I'll make another data table, similar to the above method, but pull out our top scoring pair of genes. Then check if the rank ordering is consistent within the two groups. This is a good exercise for the reader. You can use the following tables: isb-cgc.QotM.results_1002 and isb-cgc.QotM.paad_kirp_result_check

Phenotype	FALSE	TRUE
TCGA-KIRP	5	119
TCGA-PAAD	54	18

When we see how we did (see table above, 88% accuracy), it's very similar to what we previously found, but clearly there's room for improvement. Probably changing the way genes are selected would make a difference, and perhaps using more samples. Let me know if you give it a try!

February, 2018



ISB-CGC-BioCircos-Shiny

For this entry, we're going to get back to Shiny programming and work on the skeleton of a BioCircos app. These types of plots show data and relations across the genome (typically), using a circular layout.

The Circos plot was first described in Circos: An information aesthetic for comparative genomics by Krzywinski et al., Genome Research, 2009 (cited 3,295 times!), and originally aimed to display relations between genomic loci. From the paper: "Circos uses a circular ideogram layout to facilitate the display of relationships between pairs of positions by the use of ribbons". Visualizing genomic rearrangements was one of the key uses of the circos plot, but it's also a good way to visualize gene-gene interactions which are often separated by large genomic distances.

In this example, we will select a group of samples, and a pathway (a set of genes), and look at the correlations between pairs of genes. If a pathway is active, we might expect strong correlations between the participating genes. The pathways come from Reactome which was supplied by the Bader Lab and can be found in our Query of the Month table, isb-cgc:QotM.Reactome_a1 (a1 indicates annotation).

For the remainder of this month's episode, we'll go through the app, and see how it works.

In this app, there are a few objects that must stay in sync. First we have the big table of correlations, which is generated by constructing and executing a BigQuery based on user inputs. That table is filtered based on the user-adjustable slider

bar, and summary statistics are shown in the small table on the right. The filtered table is also shown in a table directly under the BioCircos plot.

First we'll define the UI.

```
library(shiny)
library(BioCircos)
library(bigrquery)
library(stringr)
source("global.R")

ui <- fluidPage(

   # Application title
   titlePanel("ISB-CGC Query of the Month, Feb 2018"),

   sidebarLayout(
      sidebarPanel(
        fluidRow(
          textOutput('pageInfo'),
          HTML("<br><br>")
        ),
        fluidRow(
          selectInput("pathway", "Pathway", pathwayNames(), selected ="SIGNALLING TO RAS␣
↪(18 genes)"),
          getTCGAProjs(),
          sliderInput('numGenes', 'random number of genes',min=0, max=50, value=10),
          actionButton(inputId="submit",label = "Submit after selecting pathway and␣
↪cohort",
                       style="color: #ffffff; background-color: #67abe5; border-color:
↪#2e6da4"),
          HTML("<br><br>"),
          sliderInput('corrThershold', 'abs value correlation threshold',min=0, max=1,␣
↪value=0.33)
        ),
        fluidRow(
          HTML("<br><br>"),
          textOutput('pageOutro')
        )
      ),

      mainPanel(
         BioCircosOutput("circosPlot", width = "100%", height = "500px"),  # MAIN PLOT!
         fluidRow(
           column(8, align="center", div(tableOutput('table'), style = "font-size:80%")␣
↪),
           tableOutput('textboxinfo')
         )
      )
   )
)
```

It's a pretty simple layout. We have some user-interface objects on the left, and a plot and tables on the right. The code to populate the interface is shown below.

```
server <- function(input, output, session) {

    # some variables used in the table summary
    rvar <- reactiveValues(geneCount = 0, edgeCount = 0, edgesTotal = 0, genesTotal = 0)

    # info text
    output$pageInfo <- renderText(infoTextBlock())
    # and some text citing the packages we used
    output$pageOutro <- renderText("Packages used: BioCircos, bigrquery")

    # calling BigQuery
    bq_data <- eventReactive(input$submit, {
      withProgress(message = 'Contacting the cloud...', value = 0, {
        incProgress(1/2, "BigQuerying...")
        pathname <- input$pathway
        cohort   <- input$cohort
        n        <- input$numGenes
        sql      <- buildQuery(pathname, cohort, n)
        service_token <- set_service_token("data/our_saved_token.json")
        data <- query_exec(sql, project='our-project-id', use_legacy_sql = F)
        data
      })
    })

    # filter out the edges by correlation strength
    filterData <- reactive({
      bqdf <- bq_data()
      bqdfFilt <- bqdf[abs(as.numeric(bqdf$spearmans)) > input$corrThershold,]
      # potentially there are no gene-gene edges after filtering
      bqdfFilt
    })

    # update the table summary after filtering the data.
    checkTable <- observe(
      {
      # grab our handles
      bqdf <- bq_data()
      bqdfFilt <- filterData()
      # collect some counts on this data
      rvar$edgeCount <- nrow(bqdfFilt)
      rvar$edgeTotal <- nrow(bqdf)
      rvar$geneTotal <- length( c(unique(bqdf$geneA), unique(bqdf$geneB)) )
      rvar$geneCount <- length( c(unique(bqdfFilt$geneA), unique(bqdfFilt$geneB)) )
      if (nrow(bqdf) == 0) {
        rvar$edgeTotal <- 0
        rvar$geneCount <- 0
      }
      if (nrow(bqdfFilt) == 0) {
        bqdfFilt <- data.frame(geneA="NA",chrA="NA",startA=0,geneB="NA",chrB="NA",
↪startB=0,spearmans=0)
        rvar$edgeCount <- 0
        rvar$geneCount <- 0
      }
```

(continues on next page)

(continued from previous page)

```
    })

    # the main plot
    output$circosPlot <- renderBioCircos({

        # edge positions, From and To
        bqdfFiltered <- filterData()
        links_chromosomes_1 <- str_replace_all(bqdfFiltered$chrA, pattern="chr",␣
↪replacement="")
        links_chromosomes_2 <- str_replace_all(bqdfFiltered$chrB, pattern="chr",␣
↪replacement="")
        links_pos_1 <- bqdfFiltered$startA
        links_pos_2 <- bqdfFiltered$startB
        links_corr <- round(bqdfFiltered[,7],2)
        links_labels = sapply(1:length(links_pos_1), function(i) {
            paste(bqdfFiltered[i,1],bqdfFiltered[i,4], round(bqdfFiltered[i,7],2), sep=',
↪  ')
        })

      # start with the base circos-track
      tracklist = BioCircosBackgroundTrack("myBackgroundTrack",
                                           minRadius = 0,
                                           maxRadius = 0.55,
                                           borderSize = 0,
                                           fillColors = "#EEFFEE")

      # if there's links to display, add in a link-track to the track-list
      if (nrow(bqdfFiltered) > 0) {
        tracklist = tracklist + BioCircosLinkTrack('myLinkTrack',
                                                   links_chromosomes_1, links_pos_1,␣
↪links_pos_1 + 50000,
                                                   links_chromosomes_2, links_pos_2,␣
↪links_pos_2 + 50000,
                                                   maxRadius = 0.8, labels=links_labels,␣
↪displayLabel=FALSE)
      }

      # and call the main function with our track-list
      expr2 <- BioCircos(tracklist, genome = "hg19", yChr = FALSE, chrPad = 0,
                         displayGenomeBorder = FALSE, genomeTicksDisplay = FALSE,␣
↪genomeLabelDy = 0,
                         LINKMouseOverTooltipsHtml01 = "",
                         LINKMouseOutStrokeWidth=3)

      expr2
      }, quoted = F)

      # display the table of correlations
      output$table <- renderTable({
        df <- filterData();
        df$lociA <- str_c(df$chrA, ':', df$startA)
        df$lociB <- str_c(df$chrB, ':', df$startB)
```

(continues on next page)

(continued from previous page)

```
        df$corr <- df$spearmans
        df <- df[, c("geneA","lociA","geneB", "lociB", "corr")]
        df
      })


      # and the table of table summaries
      output$textboxinfo <- renderTable({
        data.frame(Item=c("number of edges shown:", "number of edges total:",
                          "number of genes shown:", "number of genes total:"),
                   Count=c(rvar$edgeCount, rvar$edgeTotal,
                           rvar$geneCount, rvar$geneTotal)
        )})
    }

# Run the application
shinyApp(ui = ui, server = server)
```

That's it! The BigQuery tables are only executed when we make a change to one of the drop-down selectors and click the 'submit' button. But, in using the observer function, the tables are updated whenever the correlation filter is used.

The BigQuery data-getting function is very similar to the one used in our October example you can find it here . The user inputs are pasted into the query-string, and bigrquery::query_exec() is used to retrieve the data. We've had a lot of examples of this in the past.

One fun BigQuery trick is to use RAND() to shuffle the order of rows, followed by a LIMIT to get a random sub-sample.

```
ORDER BY
  rand()
LIMIT
  N
```

Let's try an looking at an example in some detail.

From Cai et al.,

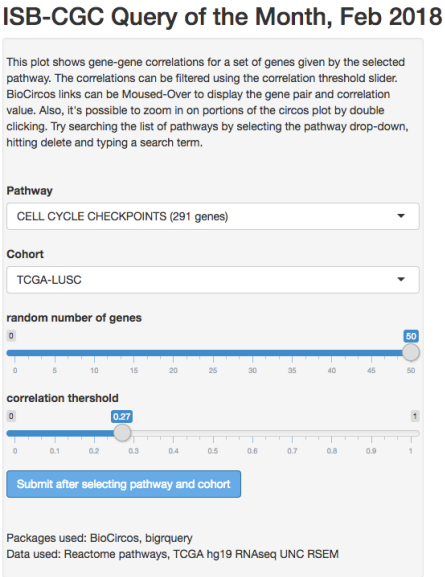
"Analyzing biological system abnormalities in cancer patients based on measures of biological entities, such as gene expression levels, is an important and challenging problem. This paper applies existing methods, Gene Set Enrichment Analysis and Signaling Pathway Impact Analysis, to pathway abnormality analysis in lung cancer using microarray gene expression data."

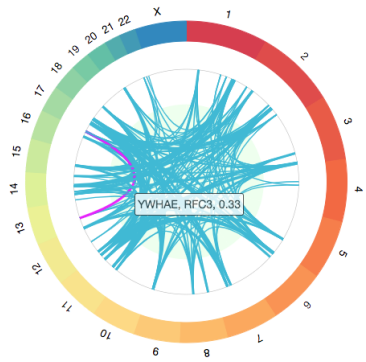
This study uses the Lung Squamous Cell Carcinoma (LUSC) data from the TCGA. In this work, they've performed a gene set enrichment analysis, reporting some high scoring pathways. Table 4 has a comparison of two gene set scoring methods, each gene set is ranked by method, so we can get a sense of where the agreement lies. Let's look at the correlation structure of a couple high ranking gene sets: cell cycle and p53 signaling pathway.

First, the cell cycle. We have the Reactome Cell Cycle pathway, but it's made up from 617 genes, which produces way more correlations that BioCircos can display. This is an interesting, but unsolved, problem regarding having too many gene-pairs. Alternately, by selecting the pathway, hitting delete, and searching for cycle, I found the 'TP53 regulates transcription of the cell cycle' pathway, perfect!

The top correlations include the gene pairs (CCNB1, AURKA, 0.68). It's an interesting pair since Cyclin B1 is strongly related to oncogenesis and AURKA, another kinase related to cell cycle, is recently getting some attention for new theraputics. Looking at the table of correlations, it's safe to say that the correlated pairs make sense in this context.

An extension to this work would involve producing a distribution of correlations through permuations which would give a good background for comparison.





geneA	chrA	startA	geneB	chrB	startB	spearmans
HIST1H4H	chr6	26277609	HIST1H2BE	chr6	26172059	0.72
PSMA6	chr14	35278633	PSMA3	chr14	58244831	0.70
SPC25	chr2	168834132	CDC25C	chr5	138285265	0.66
SPC25	chr2	168834132	SGOL2	chr2	200510008	0.63
SPC25	chr2	168834132	PSMD14	chr2	161308038	0.61
CENPO	chr2	24793136	BRIP1	chr17	61681266	0.59
HIST2H2BE	chr1	149842204	HIST1H4H	chr6	26277609	0.59
RFC4	chr3	186789880	PSMD2	chr3	184298709	0.58
SKA1	chr18	50374995	RFC5	chr12	118013588	0.58
SPC25	chr2	168834132	RFC3	chr13	33818049	0.57

Item	Count
number of edges shown:	206
number of edges total:	1059
number of genes shown:	71
number of genes total:	91

That's it for this month. As always, if you have any special requests for queries you would like to see or would like to submit a query of the month, please get a hold of us! Thanks!

January, 2018

This month, we're going to implement a common bioinformatics task: gene set scoring. In this procedure, we will compare the <joint> expression of a set of genes between two groups.

Gene sets frequently result from experiments in which, for example, expression is compared between two groups (*e.g.* control and treatment), and the genes that are significantly differentially expressed between the two groups form the "gene set". Given a gene set, numerous approaches have been proposed to assign functional interpretations to a particular list of genes.

An related approach is to consider pathways as gene sets: each gene set will therefore be a canonical representation of a biological process compiled by domain experts. We will use the WikiPathways gene sets that were assembled for a previous Query of the Month (*May2017*). In total, 381 pathways were downloaded from WikiPathways. In the BQ table, each row contains a pathway and a gene associated with that pathway.

Our task will be to determine which genesets are differentially expressed when comparing two groups of samples. We will define groups of samples using the new hg38 Somatic Mutations table based on Data Release 10 (DR10) from the NCI Genomic Data Commons.

The BigQuery table at ISB-CGC is found at: isb-cgc:TCGA_hg38_data_v0.Somatic_Mutation_DR10

The BigQuery pathway table is found at: isb-cgc:QotM.WikiPathways_20170425_Annotated

We will implement a method found in a paper titled "Gene set enrichment analysis made simple" (Rafael A Irrazary *et al*, PMID 20048385). They propose a simple solution which outperforms a popular and more complex method known as GSEA.

In short, (I'll explain more later), the gene set score comes from an average of T-tests, where the T-statistics come from testing each gene for differential expression between the two groups. The statistic is then weighted by the square root

of the sample size (number of genes in the set), so that with larger gene sets, the 'signficant' effect size can get pretty small.

At the end of it, the full query processes 29.8GB in 10.1s and cost ~$0.15.

Let's get started. First, which tissue type should we focus on? Let's choose PARP1 as an interesting gene – it encodes an enzyme involved in DNA damage repair and is also the target of some therapeutic drugs – and use the Somatic Mutation table to choose a cancer type:

```
SELECT
  project_short_name,
  COUNT(DISTINCT(sample_barcode_tumor)) AS n
FROM
  `isb-cgc.TCGA_hg38_data_v0.Somatic_Mutation_DR10`
WHERE
  Hugo_Symbol = 'PARP1'
GROUP BY
  project_short_name
ORDER BY
  n DESC
```

The result of that query shows that 46 tumor samples have PARP1 mutations in UCEC, followed by COAD and STAD with 22 each. That's a big lead by UCEC, so let's focus our work there.

Here's where our main query will begin. Since this is standard SQL, we'll be naming each subtable, and the full query can be constructed by concatenating each of the following sub-queries.

```
WITH
s1 AS (
  SELECT
    sample_barcode_tumor AS sample_barcode
  FROM
    `isb-cgc.TCGA_hg38_data_v0.Somatic_Mutation_DR10`
  WHERE
    project_short_name = 'TCGA-UCEC'
  GROUP BY
    1
)
SELECT * FROM s1
```

This query returns 530 tumor sample barcodes with at least one known somatic mutation. (The TCGA Biospecimen table includes information for a total of 553 UCEC tumor samples, but some may have not been sequenced or may have no somatic mutations – the former being more likely than the latter.) Recall that somatic mutations are variants in the DNA that are found when comparing the tumor sequence to the 'matched normal' sequence (typically from a blood sample, but sometimes from adjacent tissue). Next, for all these samples, we'll want to restrict our analysis to samples for which we also have mRNA expression data:

```
sampleGroup AS (
SELECT
  sample_barcode
FROM
  `isb-cgc.TCGA_hg38_data_v0.RNAseq_Gene_Expression`
WHERE
  project_short_name = 'TCGA-UCEC'
  AND sample_barcode IN
```

(continues on next page)

(continued from previous page)

```
  (select sample_barcode from s1)
GROUP BY
  1 )
```

Now we have 526 samples for which we have gene expression and somatic mutation calls. We are interested in partitioning this group into two parts: one with a mutation of interest, and one without. So let's gather barcodes for tumors with non-silent mutations in PARP1.

```
--
-- The first group has non-synonymous mutations in PARP1
--
grp1 AS (
SELECT
  sample_barcode_tumor AS sample_barcode
FROM
  `isb-cgc.TCGA_hg38_data_v0.Somatic_Mutation_DR10`
WHERE
  Hugo_Symbol = 'PARP1'
  AND One_Consequence <> 'synonymous_variant'
  AND sample_barcode_tumor IN (
    SELECT
      sample_barcode
    FROM
      sampleGroup )
  GROUP BY sample_barcode
),
--
-- group 2 is the rest of the samples
--
grp2 AS (
SELECT
  sample_barcode
FROM
  sampleGroup
WHERE
  sample_barcode NOT IN (
    SELECT
      sample_barcode
    FROM
      grp1)
),
```

This results in 41 tumor samples with non-synonymous PARP1 variants and 485 samples without.

Next we're going to summarize the gene expression within each of these groups. This will be used for calulating T-statistics in the following portion of the query we are constructing. For each gene, we'll take the mean, variance, and count of samples.

```
-- Summaries for Group 1 (with mutation)
--
summaryGrp1 AS (
  select
    gene_name as symbol,
```

(continues on next page)

(continued from previous page)

```
    AVG(LOG10( HTSeq__FPKM_UQ +1)) as genemean,
    VAR_SAMP(LOG10( HTSeq__FPKM_UQ +1)) as genevar,
    count(sample_barcode) as genen
  FROM
    `isb-cgc.TCGA_hg38_data_v0.RNAseq_Gene_Expression`
  WHERE
    sample_barcode IN (select sample_barcode FROM grp1)
    AND gene_name IN (
      SELECT
        Symbol as gene_name
      FROM
        `isb-cgc.QotM.WikiPathways_20170425_Annotated`
    )
  GROUP BY
    gene_name
),
--
-- Summaries for Group 2 (without mutation)
--
summaryGrp2 AS (
  select
    gene_name as symbol,
    AVG(LOG10( HTSeq__FPKM_UQ +1)) as genemean,
    VAR_SAMP(LOG10( HTSeq__FPKM_UQ +1)) as genevar,
    count(sample_barcode) as genen
  FROM
    `isb-cgc.TCGA_hg38_data_v0.RNAseq_Gene_Expression`
  WHERE
    sample_barcode IN (select sample_barcode FROM grp2)
    AND gene_name IN (
      SELECT
        Symbol as gene_name
      FROM
        `isb-cgc.QotM.WikiPathways_20170425_Annotated`
    )
  GROUP BY
    gene_name
),
--
```

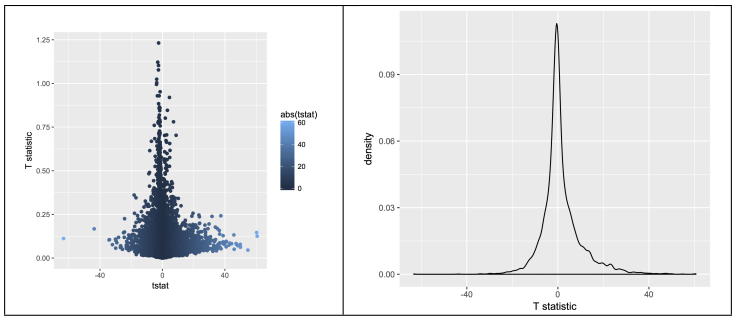
This results in two sets of summaries for 4,822 genes. (There are 4,962 unique gene symbols in the WikiPathways table, but 140 of them do not match any of the symbols in the TCGA hg38 expression table.) With this, we are ready to calculate T-statistics. Here we're going to use a two sample T-test assuming independent variance (and that we have enough samples to assume that). The T-statistic is found by taking the difference in means (of gene expression between our two groups), and normalizing it by measures of variance and sample size. Here, we want to keep T-statistics that are zero, which might come from having zero variance, because having a T-stat for each gene is important in the gene set score, even if it's a zero. To do that, you'll see the use of an IF statement below.

```
tStatsPerGene AS (
SELECT
  grp1.symbol as symbol,
  grp1.genen as grp1_n,
```

(continues on next page)

(continued from previous page)

```
  grp2.genen AS grp2_n,
  grp1.genemean AS grp1_mean,
  grp2.genemean AS grp2_mean,
  grp1.genemean - grp2.genemean as meandiff,
  IF ((grp1.genevar > 0
       AND grp2.genevar > 0
       AND grp1.genen > 0
       AND grp2.genen > 0),
    (grp1.genemean - grp2.genemean) / SQRT( (POW(grp1.genevar,2)/grp1.genen)+ (POW(grp2.
↪genevar,2)/grp2.genen) ),
    0.0) AS tstat
FROM
  summaryGrp1 as grp1
  JOIN
  summaryGrp2 AS grp2
  ON
  grp1.symbol = grp2.symbol
GROUP BY
  grp1.symbol,
  grp1.genemean,
  grp2.genemean,
  grp1.genevar,
  grp2.genevar,
  grp1.genen,
  grp2.genen
),
```



OK! We have a distribution of T statistics. The question is whether there's some hidden structure to these values. Are there gene sets with unusually high T-statistics? And do these gene sets make any sort of biological sense? Let's find out!

Now we are going to integrate our gene set table. This is as easy as doing a table join.

```
geneSetTable AS (
SELECT
  gs.pathway,
  gs.wikiID,
  gs.Symbol,
  st.grp1_n,
  st.grp2_n,
  st.grp1_mean,
```

(continues on next page)

(continued from previous page)

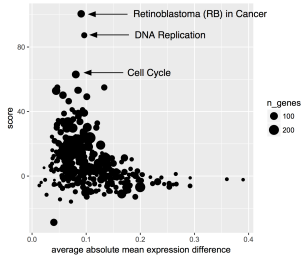
```
  st.grp2_mean,
  st.meandiff,
  st.tstat
FROM
  `isb-cgc.QotM.WikiPathways_20170425_Annotated` as gs
JOIN
  tStatsPerGene as st
ON
  st.symbol = gs.symbol
GROUP BY
  gs.pathway,
  gs.wikiID,
  gs.Symbol,
  st.grp1_n,
  st.grp2_n,
  st.grp1_mean,
  st.grp2_mean,
  st.meandiff,
  st.tstat
)
```

That's it! For each gene in the pathways (gene sets) table, we have joined in the T-statistic comparing our two groups. Now for the gene set score! To get this, we're going to simply average over the T's within each pathway, and scale the result by the square root of the number of genes. When the number of genes gets large (reasonably so), the value approximates a Z-score. In this way, using R for example, we could get a p-value and perform multiple testing correction in order to control the false discovery rate.

```
geneSetScores AS (
SELECT
  pathway,
  wikiID,
  COUNT(symbol) AS n_genes,
  AVG(ABS(meandiff)) AS avgAbsDiff,
  (SQRT(COUNT(symbol))/COUNT(symbol)) * SUM(tstat) AS score
FROM
  geneSetTable
GROUP BY
  pathway,
  wikiID )
--
--
SELECT
  *
FROM
  geneSetScores
ORDER BY
  score DESC
```

So, we see that 'Retinoblastoma (RB) in Cancer' is in the top spot with a score way above the #2 position. Why might that be? Well, PARP1 is involved in DNA damage repair, specifically through the non-homologous endjoining (NHEJ) mechanism. Samples that are deficient in PARP1 are going to have a hard time repairing DNA breaks, which makes cancer more likely. So, RB1 might need to take up the slack, and indeed it's known as a 'tumor suppressor protein': when DNA is damaged, the cell cycle needs to freeze, which happens to be one of RB1's special tricks, and also

Row	pathway	wikiID	n_genes	avgAbsDiff	score
1	Retinoblastoma (RB) in Cancer	WikiPathways_20170410	85	0.09074099343631607	100.5866566713379
2	DNA Replication	WikiPathways_20170410	41	0.09638722991485915	87.29008922117913
3	Cell Cycle	WikiPathways_20170410	98	0.08078487275431012	62.98593883820528
4	Eukaryotic Transcription Initiation	WikiPathways_20170410	37	0.046661555931630155	55.004553246277034
5	Glycolysis and Gluconeogenesis	WikiPathways_20170410	47	0.13334847972415492	54.94265001824355
6	miRNA Regulation of DNA Damage Response	WikiPathways_20170410	70	0.0845097431877511	53.20074037522477
7	DNA Damage Response	WikiPathways_20170410	68	0.0855182772416452	53.194501760511152



probably why we see the next two top ranked pathways 'DNA Replication' and 'Cell Cycle'.

For more on this topic see:

- Retinoblastoma (RB) in Cancer (Homo sapiens) (Wiki pathway WP2446)
- RB1 gene (Wikipedia entry)
- Direct involvement of retinoblastoma family proteins in DNA repair by non-homologous end-joining. (Cook et al, 2015)

Thanks, and feel free to ask about a particular topic! We're happy to take requests!

December, 2017

For December we're getting back to BigQuery. And, we've got a good one this month. Perhaps you've heard of The Genotype-Tissue Expression (GTEx) project ? It's a fantastic collection of data; donors provided a wide range of healthy tissue samples that have been used to produce a range of data types, including expression and genomics data. You can see the documentation for a complete description.

Well… for this month's query we've put some of this data in the *cloud*!

As a demonstration use-case, we thought it would be interesting to compare gene expression signatures between TCGA and GTEx, and look at the correlation between each TCGA sample (of which there are over 10,000) and each GTEx tissue type (of which there are 53, with the expression data averaged from multiple samples from the same tissue type). See below for R code and visualizations.

The bigquery below contains the following steps:

- select the 5K most variable genes from each data source
- find the intersection of these two lists: this will be the list of ~3200 genes that we'll use in the correlations
- build sub-tables of the expression data
- rank the expression data within each sample
- perform an all-by-all correlation of the ranks, between TCGA samples and GTEx tissue types (this is Spearman's correlation)

- return 545,317 rows(!) where each row represents the correation between one TCGA sample and one GTEx tissue type

Amazingly, this query processes 12.7 GB, takes about 10-20 seconds, and only cost 6 cents!

```
WITH
--
-- # First we select the 5,000 most variable genes from GTEx.
-- # This is across all tissues.
--
  GTEx_top5K AS (
  SELECT
    gene_id,
    gene_description,
    STDDEV(gene_exp) AS sigmaExp
  FROM
    `isb-cgc.GTEx_v7.gene_median_tpm`
  GROUP BY
    1,
    2
  ORDER BY
    sigmaExp DESC
  LIMIT
    5000 ),
    --
    -- # Then we select the most variable 5K genes from TCGA.
    --
  TCGA_top5K AS (
  SELECT
    HGNC_gene_symbol,
    STDDEV(normalized_count) AS sigmaExp
  FROM
    `isb-cgc.TCGA_hg19_data_v0.RNAseq_Gene_Expression_UNC_RSEM`
  WHERE
    platform='IlluminaHiSeq'
    AND HGNC_gene_symbol IS NOT NULL
  GROUP BY
    1
  ORDER BY
    sigmaExp DESC
  LIMIT
    5000 ),
    --
    -- # next we join the gene symbol tables to get the gene lists.
    --
  geneList AS (
  SELECT
    gene_id,
    HGNC_gene_symbol AS gene_symbol
  FROM
    GTEx_top5K
  JOIN
    TCGA_top5K
  ON
```

(continues on next page)

(continued from previous page)

```
    gene_description=HGNC_gene_symbol ),
    --
    -- # then we rank the gene expression within a sample_barcode.
    --
  tcgaData AS (
  SELECT
    sample_barcode,
    project_short_name AS project,
    HGNC_gene_symbol AS gene_symbol,
    normalized_count AS expr,
    DENSE_RANK() OVER (PARTITION BY sample_barcode ORDER BY normalized_count ASC) AS␣
↪rankExpr
  FROM
    `isb-cgc.TCGA_hg19_data_v0.RNAseq_Gene_Expression_UNC_RSEM`
  WHERE
    platform='IlluminaHiSeq'
    AND HGNC_gene_symbol IN (
    SELECT
      gene_symbol
    FROM
      geneList) ),
      --
      -- # and also rank the GTEx data
      --
  gtexData AS (
  SELECT
    SMTSD AS tissueType,
    gene_description AS gene_symbol,
    gene_exp AS expr,
    DENSE_RANK() OVER (PARTITION BY SMTSD ORDER BY gene_exp ASC) AS rankExpr
  FROM
    `isb-cgc.GTEx_v7.gene_median_tpm`
  WHERE
    gene_description IN (
    SELECT
      gene_symbol
    FROM
      geneList ) ),
      --
      -- # last table join on TCGA and GTEx ranked gene expression data
      --
  j1 AS (
  SELECT
    g.tissueType AS GTEx_tissueType,
    g.gene_symbol,
    g.rankExpr AS gRank,
    t.sample_barcode,
    t.project AS TCGA_project,
    t.rankExpr AS tRank
  FROM
    gtexData g
  JOIN
```

(continues on next page)

(continued from previous page)

```
    tcgaData t
  ON
    g.gene_symbol=t.gene_symbol ),
    --
    -- # and last, we correate on the ranks (Spearman's correlation).
    --
  gtCorr AS (
  SELECT
    GTEx_tissueType,
    sample_barcode,
    TCGA_project,
    CORR(gRank,tRank) AS corr
  FROM
    j1
  GROUP BY
    1,
    2,
    3 )
    --
    --
    --
SELECT
  *
FROM
  gtCorr
ORDER BY
  corr DESC
```

So now we'll run the query and create some visualizations.

```
library(bigrquery)

q <- as.character(_the_query_above_)

res0 <- query_exec(q, project=__my_project__, use_legacy_sql=F)

dim(res0)
#[1] 545317      4

head(res0)
  GTEx_tissueType   sample_barcode TCGA_project      corr
1           Liver TCGA-DD-A39V-11A    TCGA-LIHC 0.9213024
2           Liver TCGA-DD-A39Z-11A    TCGA-LIHC 0.9189148
3           Liver TCGA-DD-A3A1-11A    TCGA-LIHC 0.9176827
4           Liver TCGA-FV-A3R2-11A    TCGA-LIHC 0.9153921
5           Liver TCGA-DD-A3A5-11A    TCGA-LIHC 0.9149076
6           Ovary TCGA-BG-A3PP-11A    TCGA-UCEC 0.9139462
```

OK, now we have our table of results, where each TCGA sample is paired with a GTEx tissue type. Let's take a look at how the tissues correspond.

Just a note here: it would be a good idea to save the results from this query into a new BigQuery table, and continue to query the new table … but we'll just bring it down and process it locally.

First question: what is the top scoring correlation for each TCGA type (and for each GTEx tissue type)? We'll group by TCGA tissue type, and within those groups, pull out the row containing the maximum correlation. Then, we'll group by GTEx tissue-type, and then for each tissue-type pull out row containing the maximum correlation.

```
library(dplyr)
byTCGA <- res0 %>%
  select(GTEx_tissueType, TCGA_project, corr) %>%
    group_by(TCGA_project) %>%
      filter(corr == max(corr))

  byGTEx <- res0 %>%
    select(GTEx_tissueType, TCGA_project, corr) %>%
      group_by(GTEx_tissueType) %>%
        filter(corr == max(corr))

> data.frame(byTCGA)
                       GTEx_tissueType TCGA_project      corr
1                                Liver    TCGA-LIHC 0.9213024
2                                Ovary    TCGA-UCEC 0.9139462
3                         Adrenal Gland    TCGA-PCPG 0.9100819
4          Brain - Frontal Cortex (BA9)     TCGA-LGG 0.9100485
5                                Liver    TCGA-CHOL 0.9089752
6          Brain - Frontal Cortex (BA9)     TCGA-GBM 0.9021079
7                               Uterus    TCGA-CESC 0.8998127
8                              Thyroid    TCGA-THCA 0.8927743
9                         Adrenal Gland     TCGA-ACC 0.8871793
10                   Esophagus - Mucosa    TCGA-HNSC 0.8843076

> data.frame(byGTEx)
                            GTEx_tissueType TCGA_project      corr
1                                     Liver    TCGA-LIHC 0.9213024
2                                     Ovary    TCGA-UCEC 0.9139462
3                              Adrenal Gland    TCGA-PCPG 0.9100819
4                 Brain - Frontal Cortex (BA9)     TCGA-LGG 0.9100485
5                               Brain - Cortex     TCGA-LGG 0.9049930
6                                     Uterus    TCGA-CESC 0.8998127
7   Brain - Anterior cingulate cortex (BA24)     TCGA-LGG 0.8929013
8                                    Thyroid    TCGA-THCA 0.8927743
9                         Esophagus - Mucosa    TCGA-HNSC 0.8843076
10                          Brain - Amygdala     TCGA-LGG 0.8832739
```

The tissue signatures match up very well across projects! We see TCGA tissue types correlating most strongly with the most similar GTEx tissue types. There are some differences depending on whether we look in blocks by TCGA tumor-type or by GTEx tissue-type, but 15 match exactly from the two tables.

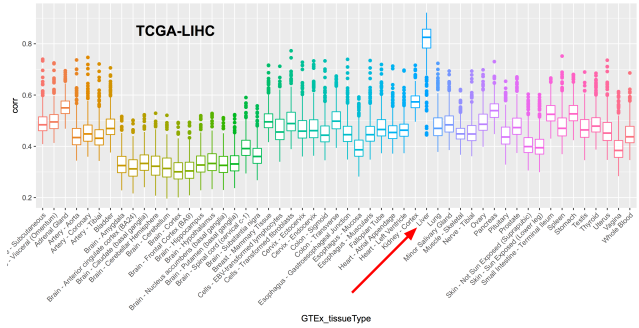
The "Liver" expression profile in particular seems to be very specific. Let's see how TCGA liver samples correlate with GTEx.

```
library(ggplot2)

lihcCorrs <- res0 %>%
  filter(TCGA_project == 'TCGA-LIHC') %>%
    select(GTEx_tissueType, TCGA_project, corr) %>%
      group_by(TCGA_project)
```

(continues on next page)

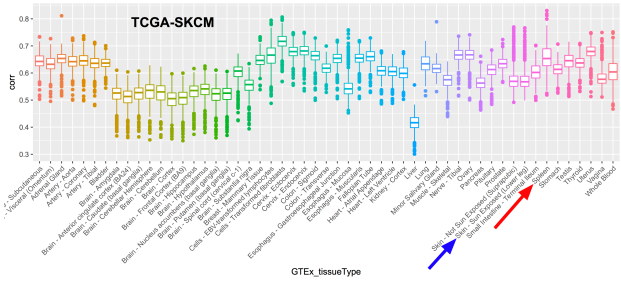
(continued from previous page)

```
qplot(data=lihcCorrs, x=GTEx_tissueType, y=corr, geom="boxplot", col=as.factor(GTEx_
↪tissueType)) +
  theme(legend.position="none") +
  theme(axis.text.x=element_text(angle=45, hjust=1)) +
  ggtitle("TCGA-LIHC")
```



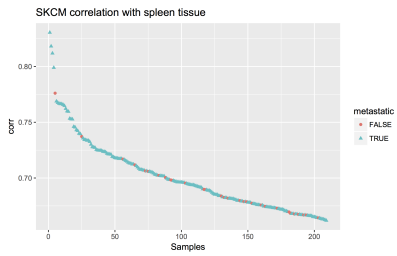
Where do we find unexpected correlations? When grouping by TCGA tissue, TCGA-SKCM actually has the highest correlation with Spleen (0.798). SKCM is the abbreviation for *melanoma* (SKin Cancer Melanoma) "a cancer in the type of skin cells called melanocytes" (from the GDC's site). To me that was a little unexpected, so let's unpack that a bit.

```
library(ggplot2)

skcmCorrs <- res0 %>%
  filter(TCGA_project == 'TCGA-SKCM') %>%
    select(GTEx_tissueType, TCGA_project, corr) %>%
      group_by(TCGA_project)

qplot(data=skcmCorrs, x=GTEx_tissueType, y=corr, geom="boxplot", col=as.factor(GTEx_
↪tissueType)) +
  theme(legend.position="none") +
  theme(axis.text.x=element_text(angle=45, hjust=1))
```



After mentioning this to Sheila, she remembered that many of the melanoma samples are metastatic samples taken from lymph nodes, and, the spleen, like the lymph nodes, is a secondary or peripheral lymphoid organ. Primary tumor sample barcodes are of the form 'TCGA-XY-1234-01' (with the final two digits indicating the sample type), while metastatic sample barcodes end in '-06'. Let's label the points according to the sample type and see what that looks like.

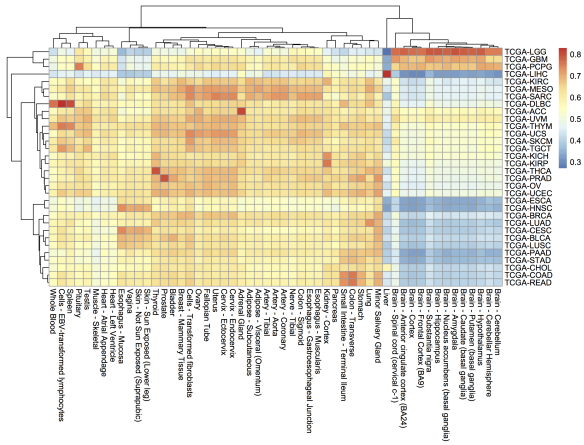
```
skcmSpleenRows <- res0 %>% filter(GTEx_tissueType == 'Spleen' & TCGA_project == 'TCGA-
↪SKCM')
qplot(data=skcmSpleenRows, x=1:nrow(skcmSpleenRows), y=corr, col=metastatic,␣
↪pch=metastatic) +
  xlab("Samples") +
  ggtitle("SKCM correlation with spleen tissue")
```



So from the plot we see that, indeed, most of the SKCM samples are metastatic samples taken from lymph nodes, explaining the high correlation that a few of them have with the Spleen tissue-type.

Lastly, let's just look at the median correlations between each TCGA and GTEx tissue type.

```
library(dplyr)
library(tidyr)
library(pheatmap)
res1 <- res0 %>% group_by(GTEx_tissueType, TCGA_project) %>% summarize(MeanCorr =␣
↪median(corr, na.rm=T))
res2 <- spread(res1, key=GTEx_tissueType, value=MeanCorr)
resdf <- as.data.frame(res2)
rownames(resdf) <- resdf$TCGA_project
pheatmap(resdf[,-1])
```



Thanks everyone! Hope you learned something this year. We sure did. See you in 2018!

Sincerely, the ISB-CGC team.

November, 2017

This month, we're going to shift topics and talk about running scripts in the cloud. In this example we're going to use R, but it would be just as easy to run a python script.

The code can be found here

In this example, I'm going to be fitting Bayesian logistic regression models using Stan (a statistical modeling and computation platform). Each job will process a different file, but we could also have each job represent a different set of parameters, all processing the same data.

We are going to use dsub to run the script, which is similar to qsub, the common job scheduler found on many clusters and grids. To run each job in parallel, dsub spins up a GCE VM, starts up a named docker on the VM, copies input data from a GCS bucket, runs the specified script, copies the output data back out to a (potentially different) GCS bucket, and finally shuts down the VM. (** See below for help on installation on Macs! **)

'Batch mode' is one of the most important dsub features. It allows one to launch many jobs with a single command. Batch mode takes a "tasks file" or as I call it, the "task matrix", and reads each row as command line parameters for a job. So for example, one column can name the location (in a google bucket) of the input data, another column can have parameters related to the script.

As part of this demonstratation, we will:

1. Define a custom R script to process user data. (stan_logistic_regression.R, data/*)
2. Generate a 'task matrix', each row describing a job in the google cloud. (cmd_generator.R, task_matrix.txt)
3. Use Google dsub to automatically start up a VM, run a script, and shutdown. Please see the how_to_dsub.txt file for instructions.

OK, so first we will take a look at the R code. The way dsub works is that a docker image is started up on a VM, and at that time, variables defined in the 'tasks file' become available as environment variables. So you can think about these like command line arguments, but to get them in the script, we use 'Sys.getenv()'. The variables enter the script as strings and will need to be typecast, depending on the need. The environment variable names are set in the tasks file as column names using *–env*. We'll look at that next.

```
# get the file name from the env variable.
dat <- read.csv(Sys.getenv('DATA_FILE'))

# stan models take a list of data
data_list <- list(y = dat$y, x = dat$x, N = length(dat$y))

# compiling and producing posterior samples from the model.
stan_samples <- stan(model_code = model, data = data_list)

# use the environment variable as a file name for a plot.
png(Sys.getenv('OUTPUT_PLOT'))
plot(stan_samples)
dev.off()

# and finally writing out a table
write.table(as.data.frame(stan_samples), file=Sys.getenv('OUTPUT_TABLE'), quote=F, row.
↪names=F)
```

It's pretty easy to programmatically construct a tasks file. You can find an example of that in cmd_generator.R, which writes out a tab-delimited table with the variables needed in each row. There are essentially three types of parameters: inputs, outputs, and environment variables. Most importantly, the inputs and outputs need to be GCS urls to objects in a bucket. So I've put my data in my bucket, and I use that link in the script.

Please see these examples.

```
--env SAMPLE_ID     --input DATA_FILE                    --output OUTPUT_TABLE          ␣
↪--output OUTPUT_PLOT
1                   gs://my_bucket/data/data_file_1.csv  gs://my_bucket/stan_table1.txt ␣
↪gs://my_bucket/stan_plot1.png
2                   gs://my_bucket/data/data_file_2.csv  gs://my_bucket/stan_table2.txt ␣
↪gs://my_bucket/stan_plot2.png
3                   gs://my_bucket/data/data_file_3.csv  gs://my_bucket/stan_table3.txt ␣
↪gs://my_bucket/stan_plot3.png
```

Now, if you're on a Mac, it can be pretty hard to get dsub installed. It's due to a conflict with the apple version of the 'six' python library, see https://github.com/pypa/pip/issues/3165 for more info.

To get around this, we can install dsub in a virtual environment . Make sure you're using the 'bash shell'.

```
virtualenv dsub_libs
source dsub_libs/bin/activate
pip install dsub
```

Now we're pretty close at this point. We need to put our data in a google bucket and find a suitable docker image. If you can't find a docker image with everything you need, it's fairly easy to build one. But for this purpose, I searched for 'docker and RStan' and found some docker images.

- https://github.com/jburos/rstan-docker
- https://hub.docker.com/r/jackinovik/rstan-complete/

To run dsub, the command looks like:

```
dsub \
  --project my-google-project-0001 \
  --zones "us-west-*" \
  --logging gs://my_google_bucket/logs/ \
  --image jackinovik/rstan-complete \
  --script ./stan_logistic_regression.R \
  --tasks task_matrix.txt \
  --preemptible \
  --wait
```

We simply run that and we get a response...:

```
Job: stan-logis--davidlgibbs--171107-193915-44
Launched job-id: stan-logis--davidlgibbs--171107-193915-44
3 task(s)
To check the status, run:
  dstat --project my-google-project-0001 --jobs 'stan-logis--davidlgibbs--171107-193915-
↪44' --status '*'
To cancel the job, run:
  ddel --project my-google-project-0001 --jobs 'stan-logis--davidlgibbs--171107-193915-44
↪'
Waiting for job to complete...
Waiting for: stan-logis--davidlgibbs--171107-193915-44.
```

Now, we can check if our job's finished using the *dstat* command or simply look in our output bucket. If there's a problem, it's mandatory to read the logs!

The exact same procedure could be used to run python or bash scripts.

October, 2017

For October, we're going to dive into using Plotly for visualziation in Shiny apps. In particular, we're going to implement an interatictive heatmap using heatmaply. To start, here's some important links.

ISB-CGC Gene Set Correlation Heatmaply!

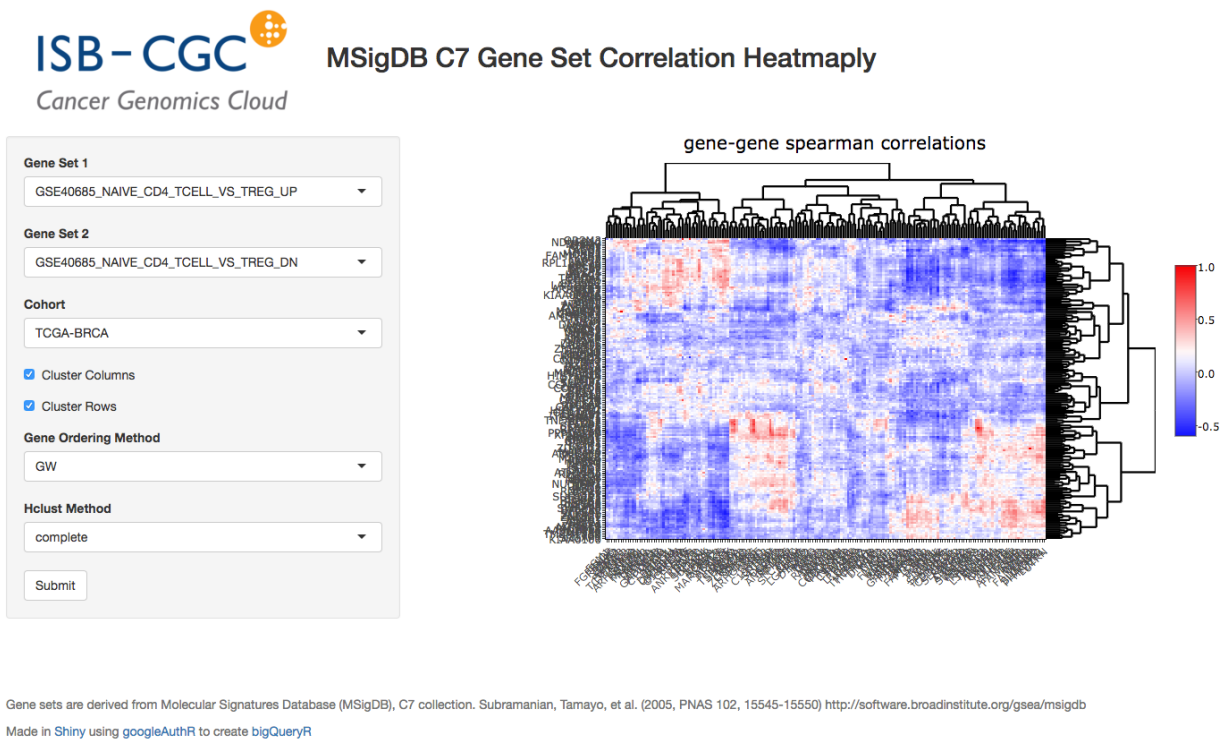
Heatmaply

The heatmaply paper

Shiny-Plotly

bigrquery

Exciting highlights include using BigRQuery to make queries from *inside* Shiny! We do that by using service account authorization. And of course, heatmaply, an interactive heatmap that lets you zoom and scroll around.



So we'll just jump right into it!

First I'll list out the ui.R code.

```
library(shiny)
library(plotly)
library(heatmaply)


source("global.R")

ui <- fluidPage(
  titlePanel(title=div(img(src="half_isb_logo.png"), "Immune-related Gene Set␣
↪Correlation-Heatmaply (MSigDB C7)")),
  helpText(HTML("<strong>Hit submit to call Google BigQuery. In the heatmap, select an␣
↪area to zoom.<strong>")),
  sidebarLayout(
    sidebarPanel(
      selectInput("var1", "Gene Set 1", getGeneSets(), selected ="GSE40685_NAIVE_CD4_
↪TCELL_VS_TREG_UP"),
      selectInput("var2", "Gene Set 2", getGeneSets(), selected = "GSE40685_NAIVE_CD4_
↪TCELL_VS_TREG_DN"),
      getTCGAProjs(), # returns a selectInput obj
      checkboxInput("showlabels", "Show Labels", value=T),
      checkboxInput("clustercols", label = "Cluster Columns", value = T),
      checkboxInput("clusterrows", label = "Cluster Rows", value = T),
      selectInput("seriate", "Gene Ordering Method", c("OLO", "GW", "mean", "none"),␣
↪selected = "GW"),
      selectInput("hclust_method", "Hclust Method", c("ward.D", "ward.D2", "single",
                                                      "complete", "average", "mcquitty",
                                                      "median", "centroid"), selected=
↪"ward.D2"),
      actionButton(inputId="submit",label = "Submit")
    ),

    mainPanel(
      tags$head(
        tags$style(# thanks BigDataScientist @ stackoverflow!
          HTML(".shiny-notification {
               height: 50;
               width: 400px;
               position:fixed;
               top: calc(50% - 50px);;
               left: calc(50% - 200px);;
               }
               "
            )
          )
        ),
      plotlyOutput("plot", height = "600px")
    )
  ),
  br(),
  helpText("What's going on here? The genes belonging to two immune-related gene sets␣
↪are used to compute Spearman correlation on RNA-seq data from a given type of cancer.␣
↪It's a visualization of the relationship between two gene sets."),
```

(continues on next page)

(continued from previous page)

```
  helpText("Heatmaply: Tal Galili, Alan O'Callaghan, Jonathan Sidi, Carson Sievert;␣
↪heatmaply: an R package for creating interactive cluster heatmaps for online␣
↪publishing, Bioinformatics, , btx657, https://doi.org/10.1093/bioinformatics/btx657"),
  helpText("Gene sets: Molecular Signatures Database (MSigDB), C7 collection.␣
↪Subramanian, Tamayo, et al. (2005, PNAS 102, 15545-15550) http://software.
↪broadinstitute.org/gsea/msigdb"),
  helpText("Made in", a("Shiny", href="http://shiny.rstudio.com/"), " using ", a("google␣
↪bigquery, bigrquery, heatmaply, and plotly"))
)
```

So we start up a fluidPage layout, and define a number of controls in the sidebar. The gene set selectors get a long list of gene set names that I've captured from MSigDB and dumped in a file (around 4000 gene sets!).

Another interesting technique is to define a function, like getTCGAProjs(), that builds and returns a selectInput object, using the long list of TCGA projects. Works great and keeps the code easy to read.

Also notice the use of CSS to change the default progress bar.

OK, now we can jump over to the server.R file.

```
server <- function(input, output, session) {

  # calling BigQuery
  bq_data <- eventReactive(input$submit, {
    load("data/gene_set_hash.rda")
    geneNames1 <- getGenes(sethash, input$var1)
    geneNames2 <- getGenes(sethash, input$var2)
    cohort <- input$cohortid
    sql <- buildQuery(geneNames1, geneNames2, cohort)
    service_token <- set_service_token("data/my_private_key.json")
    data <- query_exec(sql, project='our_bq_project', useLegacySql = F)
    data
  })

  output$plot <- renderPlotly({

    withProgress(message = 'Working...', value = 0, {
      incProgress()

      # first make the bigquery
      bqdf <- bq_data()
      incProgress()

      # then build the correlation matrix
      df <- buildCorMat(bqdf)
      incProgress()

      # then get the heatmap options
      cluster_cols <- as.logical(input$clustercols)
      cluster_rows <- as.logical(input$clusterrows)

      # color scheme
```

(continues on next page)

(continued from previous page)

```
      incProgress()
      rwb <- colorRampPalette(colors = c("blue", "white", "red"))
      heatmaply(df,
              main = 'gene-gene spearman correlations',
              Colv=cluster_cols, Rowv=cluster_rows,
              colors = rwb, seriate=input$seriate,
              hclust_method = input$hclust_method,
              showticklabels = as.logical(input$showlabels),
              margins = c(150,200,NA,0))
    })
  })


  output$event <- renderPrint({
    d <- event_data("plotly_hover")
    if (is.null(d)) "Hover on a point!" else d
  })
}
```

So, the first function that's listed is the bq_data function. This function executes after the user makes their selection and hits the 'submit' button. I have previously taken the ~4000 gene sets, and built a hash (using the awesome hash library) that takes gene set names and returns the genes. The getGenes functions performs that task. Then the BigQuery SQL is then contructed:

```
buildQuery <- function(geneNames1, geneNames2, cohort) {
  q <- paste(
    "
    WITH
    --
    -- first we create a subtable for gene set 1
    --
    cohortExpr1 AS (
      SELECT
        sample_barcode,
        HGNC_gene_symbol,
        LOG10( normalized_count +1) AS logexpr,
        RANK() OVER (PARTITION BY HGNC_gene_symbol ORDER BY normalized_count ASC) AS␣
↪expr_rank
      FROM
        `isb-cgc.TCGA_hg19_data_v0.RNAseq_Gene_Expression_UNC_RSEM`
      WHERE
        project_short_name = '",cohort ,"'
        AND HGNC_gene_symbol IN ",geneNames1 ,"
        AND normalized_count IS NOT NULL
        AND normalized_count > 0),
    --
    -- then we create a subtable for gene set 2
    --
    cohortExpr2 AS (
      SELECT
        sample_barcode,
        HGNC_gene_symbol,
```

(continues on next page)

(continued from previous page)

```
        LOG10( normalized_count +1) AS logexpr,
        RANK() OVER (PARTITION BY HGNC_gene_symbol ORDER BY normalized_count ASC) AS␣
↪expr_rank
      FROM
        `isb-cgc.TCGA_hg19_data_v0.RNAseq_Gene_Expression_UNC_RSEM`
      WHERE
        project_short_name = '",cohort ,"'
        AND HGNC_gene_symbol IN ",geneNames2 ,"
        AND normalized_count IS NOT NULL
        AND normalized_count > 0),
    --
    -- then we join the two gene expression tables,
    -- we could get rid of the logexpr fields, but maybe
    -- they'd prove useful in some other query.
    --
    jtab AS (
      SELECT
        cohortExpr1.sample_barcode,
        cohortExpr2.sample_barcode,
        cohortExpr1.HGNC_gene_symbol as g1,
        cohortExpr2.HGNC_gene_symbol as g2,
        cohortExpr1.expr_rank as e1,
        cohortExpr2.expr_rank as e2
      FROM
        cohortExpr1
      JOIN
        cohortExpr2
      ON
        cohortExpr1.sample_barcode = cohortExpr2.sample_barcode
      GROUP BY
        cohortExpr1.sample_barcode,
        cohortExpr2.sample_barcode,
        cohortExpr1.HGNC_gene_symbol,
        cohortExpr2.HGNC_gene_symbol,
        cohortExpr1.logexpr,
        cohortExpr1.expr_rank,
        cohortExpr2.logexpr,
        cohortExpr2.expr_rank )
    --
    -- last, we correlate the RANKs, to get a Spearman correlation.
    --
    SELECT
      g1,
      g2,
      corr(e1, e2) as spearmans
    FROM
      jtab
    GROUP BY
      g1,g2
    ",
    sep="")
}
```

OK, so with that query-string constructed, we can make the call to Google BigQuery. In order to get authorized, here we're using a service account. To get that set up, you will need to log into your Google Cloud Console, and follow these instructions (Service account credentials). This will generate a small .json file containing your private key (protect it, and don't lose it). Then we can use the bigrquery function set_service_token, providing the path to the json file. Very easy! After that we simply make the call using query_exec.

Then, to make the heatmap, we use the renderPlotly function. In that function, first we get the summarized data from BigQuery, which is returned as a data.frame (bqdf), and we transform that into a matrix using the reshape2 library.

```
library(reshape2)

buildCorMat <- function(bqdf) {
  # bqdf is a data.frame returned from query_exec
  # g1 and g2 are gene names
  # and we have a Spearman correlation
  meltdf <- melt(bqdf)
  sqdf <- dcast(meltdf, g1~g2, value.var = "value")
  # then a bit of tidying up.
  sqdf[is.na(sqdf)] <- 0
  rownames(sqdf) <- sqdf[,'g1']
  sqdf <- sqdf[,-1]
  return(sqdf)
}
```

With that done (building the correlation matrix), we can simply make the call to heatmaply, which is linked back to our plotlyOutput in the mainPanel (in the ui.R code).

So we've just tied together Shiny, Plotly, Heatmaply, and BigQuery into one interactive web tool. Got a question about it? Let me know! dgibbs *at* systemsbiology *dot* org.

September, 2017

Greetings! For September we've implemented a new statistical test: the one-way ANOVA. This statistical test can be used to determine whether there is a statistically significant difference between the means of two or more independent groups. Although in this example, I'm only looking at two groups, it would not be difficult to extend this to any number of groups, assuming there is a reasonable number of samples within each group.

Consider the model $y_{ij} = m + a_i + e_{ij}$, where y_{ij} is a continuous variable over samples *j*, in groups *i*, and a_i is a constant for each group *i*, and e_{ij} is a gaussian error term with mean 0.

Using this model, we are describing the data as being sampled from groups, with each group having a mean value equal to $m + a_i$. The null hypothesis is that each of the group means is the same (*ie* that the a_i terms are zero), while the alternative hypothesis is that at least one of the a_i terms is *not* zero.

We use the F-test to compare these two hypotheses. To compute the test statistic, we compute the within-group variation and the between-group variation. Recall that sample variance is defined as the sum of squared differences between observations and the mean, divided by the number of samples (normalized).

Before we get into the query, please note that you can find a specialized version of the below query that compares the expression between individuals with a SNP and without a SNP, using the same SQL as the August query. I've put that query in this github gist.

And you can find the associated Shiny app, using the same layout from August, where we plot the F distribution and show a comparison of means. You can find that here.

Let's look at the query:

```
WITH
  -- using standard SQL,
  -- we'll select our cohort and gene expression
  --
  cohortExpr AS (
  SELECT
    sample_barcode,
    LOG10(normalized_count) AS expr
  FROM
    `isb-cgc.TCGA_hg19_data_v0.RNAseq_Gene_Expression_UNC_RSEM`
  WHERE
    project_short_name = 'TCGA-BRCA'
    AND HGNC_gene_symbol = 'TP53'
    AND normalized_count IS NOT NULL
    AND normalized_count > 0),
  --
  -- And we'll select the variant data for our cohort,
  -- we're going to be comparing variant types (SNP, DEL, etc)
  --
  cohortVar AS (
  SELECT
    Variant_Type,
    sample_barcode_tumor AS sample_barcode
  FROM
    `isb-cgc.TCGA_hg19_data_v0.Somatic_Mutation_MC3`
  WHERE
    SYMBOL = 'TP53' ),
  --
  -- then we join the expression and variant data using sample barcodes
  --
  cohort AS (
  SELECT
    cohortExpr.sample_barcode AS sample_barcode,
    Variant_Type AS group_name,
    expr
  FROM
    cohortExpr
  JOIN
    cohortVar
  ON
    cohortExpr.sample_barcode = cohortVar.sample_barcode ),
  --
  -- First part of the calculation, the grand mean (over everything)
  --
  grandMeanTable AS (
  SELECT
    AVG(expr) AS grand_mean
  FROM
    cohort ),
  --
  -- Then we need a mean per group, and we can get a count of samples
  -- per group.
  --
```

(continues on next page)

(continued from previous page)

```
groupMeansTable AS (
SELECT
  AVG(expr) AS group_mean,
  group_name,
  COUNT(sample_barcode) AS n
FROM
  cohort
GROUP BY
  group_name),
--
-- To get the between-group variance
-- we take the difference between the grand mean
-- and the means for each group and sum over all samples
-- ... a short cut being taking the product with n.
-- Later we'll sum over the n_sq_diff
--
ssBetween AS (
SELECT
  group_name,
  group_mean,
  grand_mean,
  n,
  n*POW(group_mean - grand_mean,2) AS n_diff_sq
FROM
  groupMeansTable
CROSS JOIN
  grandMeanTable ),
--
-- Then, to get the variance within each group
-- we have to build a table matching up the group mean
-- with the values for each group. So we join the group
-- means to the values on group name. We are going to
-- sum over this table just like ssBetween
--
ssWithin AS (
SELECT
  a.group_name AS group_name,
  expr,
  group_mean,
  b.n AS n,
  POW(expr - group_mean, 2) AS s2
FROM
  cohort a
JOIN
  ssBetween b
ON
  a.group_name = b.group_name ),
--
-- The F stat comes from a ratio, the numerator is
-- calculated using the between group variance, and
-- dividing by the number of groups (k) minus 1.
--
```

(continues on next page)

(continued from previous page)

```
  numerator AS (
  SELECT
    'dummy' AS dummy,
    SUM(n_diff_sq) / (count(group_name) - 1) AS mean_sq_between
  FROM
    ssBetween ),
  --
  -- The denominator of the F stat ratio is found using the
  -- variance within groups. We divide the sum of the within
  -- group variance and divide it by (n-k).
  --
  denominator AS (
  SELECT
    'dummy' AS dummy,
    COUNT(distinct(group_name)) AS k,
    COUNT(group_name) AS n,
    SUM(s2)/(COUNT(group_name)-COUNT(distinct(group_name))) AS mean_sq_within
  FROM
    ssWithin),
  --
  -- Now we're ready to calculate F!
  --
  Ftable AS (
  SELECT
    n,
    k,
    mean_sq_between,
    mean_sq_within,
    mean_sq_between / mean_sq_within AS F
  FROM
    numerator
  JOIN
    denominator
  ON
    numerator.dummy = denominator.dummy)

SELECT
  *
FROM
  Ftable
```

OK, so let's check our work. Using the BRCA cohort and TP53 as our gene, we have 375 samples with a variant in this gene. We're going to look at whether the type of variant is related to the gene expression we observe. If we just pull down the data using the 'cohort' subtable (as above), we can get a small data frame, which let's us do the standard F stat table in R.

```
> # dat is the data.frame created by running the above query
>
> head(dat)
    sample_barcode group_name     expr
1 TCGA-A2-A0T3-01A        DEL 2.623283
2 TCGA-A8-A07B-01A        DEL 2.450762
```

(continues on next page)

(continued from previous page)

```
3 TCGA-AR-A5QQ-01A        DEL 2.579250
4 TCGA-A2-A0YE-01A        DEL 2.298823
5 TCGA-C8-A135-01A        DEL 2.744527
6 TCGA-A7-A13E-01A        DEL 3.246725
>
> dat %>% group_by(group_name) %>% summarize(mean=mean(expr), sd=sd(expr))
# A tibble: 3 × 3
  group_name     mean        sd
      <fctr>    <dbl>     <dbl>
1        DEL 2.791941 0.3220669
2        INS 2.642215 0.1158877
3        SNP 3.218580 0.3129593
>
>
> anova(lm(data=dat, expr~group_name))
Analysis of Variance Table

Response: expr
            Df Sum Sq Mean Sq F value    Pr(>F)
group_name   2 12.460  6.2302  65.147 < 2.2e-16 ***
Residuals  372 35.576  0.0956
---
```

OK, if you run the above BigQuery, you'll see the same results. We see that the F statistic is really high, which makes sense looking at the difference in mean expression values across the groups (these are log10 expression values).

I have created a specialized version of the above test (as opposed to generalized, ha ha) that compares the expression between individuals with a SNP and without a SNP, using the same SQL to create groups as last month. I've put that query in this github gist.

And additionally, I've put that query into a Shiny app, that uses the same layout from August. You can find that here.

August, 2017

This month we have been working on a small demo application using BigQuery, with a graphical front-end built with R Shiny . You can try it out yourself here and even watch the video.

Using the R programming language, Shiny is an easy way to produce interactive visualizations that can be hosted on the web.

Shiny sites are hosted by a Shiny server, which you can set up locally or use the free shinyapps.io service, which is provided by the same company that produces the RStudio (which has a builtin Shiny server for dev work).

In the past, we've shown how queries can be programmatically built up; here we're going to provide a user interface to collect variables that are inserted into the BigQuery (like gene names).

The query is going to look at patient survival, and how survival rates change with gene mutations. Therefore we'll be using two tables and a small set of variables:

- isb-cgc:TCGA_bioclin_v0.Clinical for survival data
  - **days_to_last_known_alive**: This field indicates the number of days to the last follow up appointment (still alive) or until death, relative to "time zero" (typically the day of diagnosis).

  - **vital_status**: This field is filled in for all but 4 cases and is correct as of the last available follow up for that individual. Over all TCGA, 7534 cases were known to still be "Alive", while 3622 were "Dead", and 4 were of unknown vital status.

- isb-cgc:TCGA_hg38_data_v0.Somatic_Mutation for mutation status
  - **Variant_Classification**: eg Missense_Mutation, Silent, 3'UTR, Intron, etc (18 different values occur in this table)
  - **Variant_Type**: one of 3 possible values: SNP, DEL, INS
  - **IMPACT**: one of 4 values: LOW, MODERATE, HIGH, or MODIFIER

What we want the query to do, is to collect a cohort of patients into two groups, those that have a SNP with some potential effect in a particular gene, and those that do not. Then we can compare the survival rates between the two groups to assess whether the mutation has some potential effect.

Let's take a look at an example query, then we'll see how to build it up in the code.

```
-- using Standard SQL --
--
-- First we build our table of survival times
--
WITH clin_table AS (
SELECT
  case_barcode,
  days_to_last_known_alive,
  vital_status
FROM
  `isb-cgc.TCGA_bioclin_v0.Clinical`
WHERE
  project_short_name = 'TCGA-GBM' ),
--
-- Then we can build our table of mutation status.
-- We do that by using an If statement with a sub-query.
--
mut_table AS (
SELECT
  case_barcode,
  IF ( case_barcode IN (
    SELECT
      case_barcode
    FROM
      `isb-cgc.TCGA_hg38_data_v0.Somatic_Mutation`
    WHERE
      SYMBOL = 'IDH1'
      AND Variant_Classification <> 'Silent'
      AND Variant_Type = 'SNP'
      AND IMPACT <> 'LOW'), 'Mutant', 'WT') AS mutation_status
FROM
  `isb-cgc.TCGA_hg38_data_v0.Somatic_Mutation` )
--
-- Finally, we can join those tables.
--
SELECT
  mut_table.case_barcode,
```

(continues on next page)

(continued from previous page)

```
  days_to_last_known_alive,
  vital_status,
  mutation_status
FROM
  clin_table
JOIN
  mut_table
ON
  clin_table.case_barcode = mut_table.case_barcode
GROUP BY
  mut_table.case_barcode,
  days_to_last_known_alive,
  vital_status,
  mutation_status
```

The Shiny App

Now we'll move on to the description of the app. When creating a new Shiny project in RStudio, two main files are created: 'ui.R' and 'server.R'. Additionally, I created one more called 'global.R'. The 'ui.R' file contains the code needed to build the html interface, 'server.R' contains the code that responds to the interface, and 'global.R' contains the functions that build the query, call the BigQuery API, and plot the results.

Starting with the interface found in 'ui.R', the googleAuthR package was used to perform authorization, using the googleAuthUI("loginButton"). Next, the GCP project ID is collected using the textInput widget, we need this because even after logging in, we still need to tell BigQuery which GCP project is going to be *billed* for the query. (You will need to be a member of at least one GCP project, with permissions to run BigQuery queries. To find out the ID(s) for GCP project(s) you are a member of, you can go to the Google Cloud Console.) Then, patient cohorts are selected using the selectInput widget, which is like a drop down menu of TCGA studies. And lastly, we have a textInput widget to specify the gene symbol. At the bottom of the interface is an actionButton called submit that kicks off the work.

```
ui <- fluidPage(

  #!! excluding layout code like sidebarLayout and panels. !!#

  googleAuthUI("loginButton"),

  textInput("projectid", "Project ID", value = "your project id", placeholder = "your␣
↪project id"),

  selectInput("cohortid", label = "Cohort",
                  choices = list(
                    "TCGA-ACC"="TCGA-ACC",
                    "TCGA-BLCA"="TCGA-BLCA",
                    "TCGA-BRCA"="TCGA-BRCA",
                    "TCGA-CESC"="TCGA-CESC",
                    "TCGA-CHOL"="TCGA-CHOL",
                    ## etc ##
            ),selected = "TCGA-GBM") ,

  textInput("varname",  "Gene Symbol", value = "IDH1", placeholder = "IDH1"),

  actionButton(inputId="submit",label = "Submit")
```

(continues on next page)

(continued from previous page)

```
)
```

In the 'server.R' file, there's one main function called 'server'. Inside that function, we get our accessToken by calling the googleAuth module, which is linked to the 'loginButton'. Also we have a function linked to the submit button called outputPlot, which wraps our plot function in the googleAuthR::with_shiny function, in order to make our API calls while properly logged in.

See this help on authorization.

```
options("googleAuthR.scopes.selected" = c("https://www.googleapis.com/auth/bigquery"))
options("googleAuthR.webapp.client_id" = "get_this_from_your_cloud_console_under_
↪Credentials_OAuth 2.0 client IDs.apps.googleusercontent.com")
options("googleAuthR.webapp.client_secret" = "get_from_cloud_console")


server <- function(input, output, session){

  ## Create access token and render login button
  access_token <- callModule(googleAuth, "loginButton", approval_prompt = "force")

  outputPlot <- eventReactive(input$submit,{
    ## wrap existing function using googleAuthR::with_shiny
    ## pass the reactive token using shiny_access_token param
    project  <- as.character(input$projectid)
    cohort   <- as.character(input$cohortid)
    varname  <- input$varname

    if(is.null(access_token())) {
      errorPlot()
    } else {
      with_shiny(f = drawPlot, shiny_access_token = access_token(), project, cohort,␣
↪varname)
    }
  })

  output$plot <- renderPlot({outputPlot()}, width=600, height=500)
}
```

The plot is drawn using a model from the survival package and a ggplots style package called ggsurvminer

```
drawPlot <- function(project, cohort, varname) {
  #
  # first make a call to BigQuery, and build the data frame
  dat <- buildAndRunQuery(varname, project, cohort)
  #
  # then we fit our survival model
  fit <- survfit(Surv(days_to_last_known_alive, vital_status) ~ mutation_status,␣
↪data=dat)
  #
  # finally visualize the survival model using ggsurvplot.
  survminer::ggsurvplot(fit=fit, data=dat, pval=T, risk.table=T, conf.int=T)
}
```

The last portion we'll look at, and maybe the most important, involves the call to big query! In the 'buildAndRunQuery'

function, we build up the query as a long string, then contruct an API function using googleAuthR functions, and finally make the API call, and get the results. There are helper functions found in the bigQueryR, but I think it's instructional to see how the backend works. In future QotMs, we will explore using bigQueryR.

```
buildAndRunQuery <- function(varName, aproject, cohort) {
  #
  # First we're going to build the string representing the BigQuery #
  #
  q <- paste(
    "
    WITH
    clin_table AS (
      select
      case_barcode,
      days_to_last_known_alive,
      vital_status
    from
      `isb-cgc.TCGA_bioclin_v0.Clinical`
    WHERE
      project_short_name = '", cohort,"' ),
    mut_table AS (
    SELECT
    case_barcode,
    IF ( case_barcode IN (
      SELECT
      case_barcode
      FROM
      `isb-cgc.TCGA_hg38_data_v0.Somatic_Mutation`
      WHERE
      SYMBOL = '", varName, "'
      AND Variant_Classification <> 'Silent'
      AND Variant_Type = 'SNP'
      AND IMPACT <> 'LOW'), 'Mutant', 'WT') as mutation_status
    FROM
    `isb-cgc.TCGA_hg38_data_v0.Somatic_Mutation` )
  SELECT
    mut_table.case_barcode,
    days_to_last_known_alive,
    vital_status,
    mutation_status
  FROM
    clin_table
  JOIN
    mut_table
  ON
    clin_table.case_barcode = mut_table.case_barcode
  GROUP BY
    mut_table.case_barcode,
    days_to_last_known_alive,
    vital_status,
    mutation_status
    ",
    sep="")
```

(continues on next page)

(continued from previous page)

```
  # define body for the POST request to the Google BigQuery API
  body = list(
    query=q,
    defaultDataset.projectId=aproject,
    useLegacySql = F
  )

  #create a function to make the POST call to Google BigQuery
  f = gar_api_generator("https://www.googleapis.com/bigquery/v2",
                        "POST",
                        path_args = list(projects=aproject, queries=""))

  # call function with body as input argument
  response = f(the_body=body)

  dat <- data.frame()
  if(!is.null(response))
  {
    # have to construct the data.frame from a list of results.
    dat = as.data.frame(do.call("rbind",lapply(response$content$rows$f,FUN = t)))
    colnames(dat) <- c("ID", "days_to_last_known_alive", "vital_status", "mutation_status
↪")

    # then we need to do a little data-cleaning to get ready for our survival model
    dat$days_to_last_known_alive <- as.numeric(as.character(dat$days_to_last_known_
↪alive))
    dat$vital_status <- ifelse(dat$vital_status == 'Alive', 0, 1)
    dat$mutation_status <- as.factor(dat$mutation_status)
  }
  return(dat)
}
```

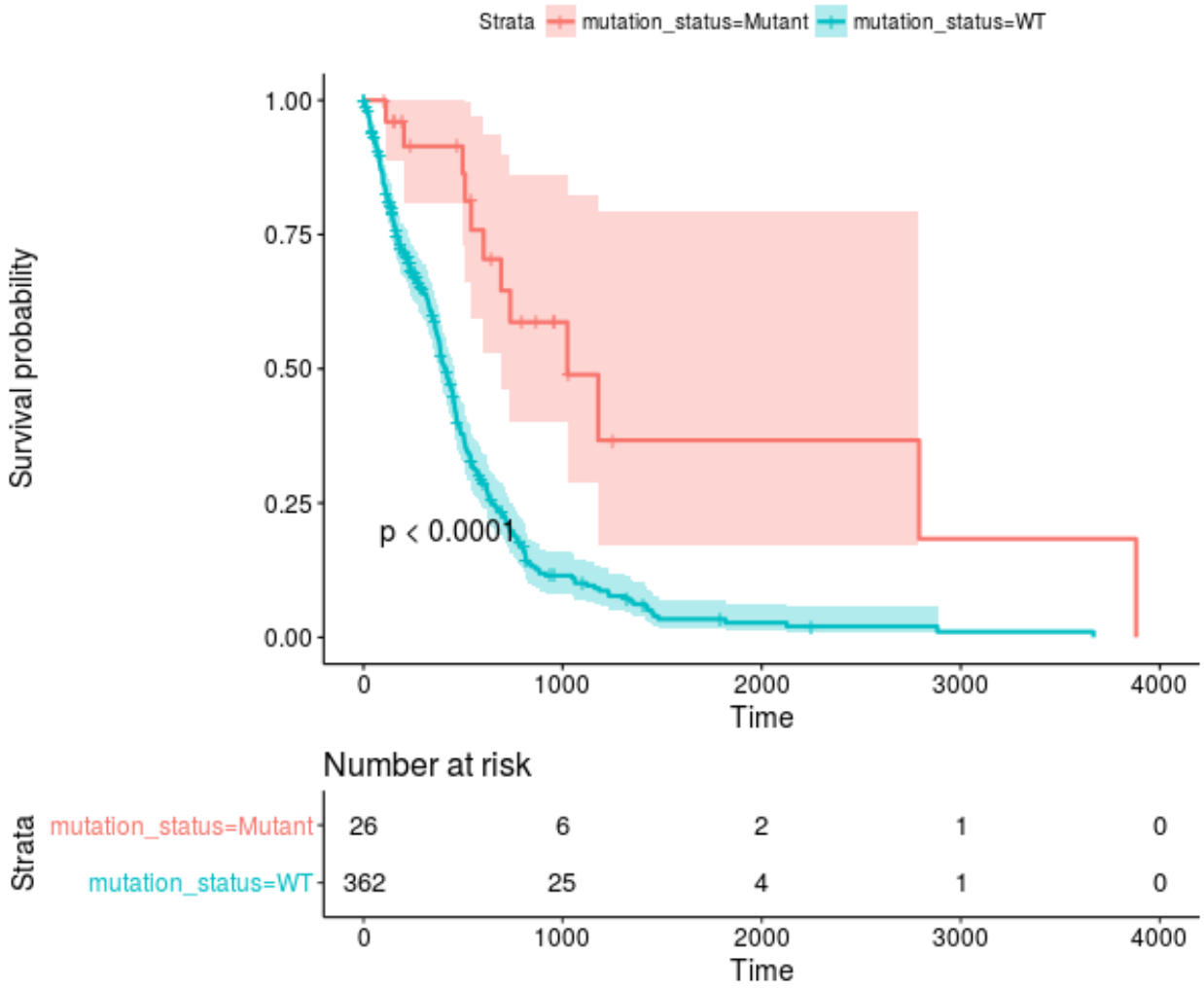
The resulting plot will show if the two groups, defined by SNP mutation status, have significantly different survival rates. In our example, contrary to intuition, a mutation in the IDH1 gene, in GBM, actually has a positive effect on survival. (See this 2014 paper by Cohen *et al* about *IDH1 and IDH2 Mutations in Gliomas* for more information about this.)

The results for the TCGA-LGG cohort are also quite striking – go have a look!

July, 2017

Way back in December we started talking about the new NCI-GDC Data Portal which includes both hg19 and hg38 alignments. At that time, those were part of isb-cgc:hg19_data_previews and isb-cgc:hg38_data_previews. Now, at this point they've matured into three data sets:

- isb-cgc:TCGA_bioclin_v0
- isb-cgc:TCGA_hg19_data_v0
- isb-cgc:TCGA_hg38_data_v0


Strata
mutation_status=Mutant
mutation_status=WT
Survival probability
1.00
0.75
0.50
0.25
0.00
p < 0.0001
0
1000
2000
3000
4000
Time
Number at risk
Strata
mutation_status=Mutant
26
6
2
1
0
mutation_status=WT
362
25
4
1
0
0
1000
2000
3000
4000
Time

And, as you'll discover, there's been some changes to the "standard" column names that we used previously. This was done to better align with the GDC, and make things more universal across data sources.

For one, barcode column names (and most column names) have become all lower case and underscore_separated. So 'AliquotBarcode' has become 'aliquot_barcode'. Same with SampleBarcode (sample_barcode). However, ParticipantBarcode has become case_barcode. Also 'Study' is now refered to as 'project_short_name'. So if you're having trouble getting an 'old' query to work, make sure the column names haven't changed, and check whether it's in Legacy SQL or Standard SQL.

As we transition to standard SQL and the new GDC datasets, one question that's come up around here relates to records. Overall, in the isb-cgc datasets, there's very few data types other than STRINGs, INTEGERs, and FLOATs. But occasionally you'll bump into something that needs a different query structure, and the RECORD type is one of those. One place to find this rare beast is in the methylation probe annotation (isb-cgc:platform_reference.methylation_annotation).

Each methylation probe has a specified genomic location, given as a chromosome and the base position on that chromosome. An analytical question that often arises is something like "does methylation in this region of the genome affect RNA transcription?". It's a good question, and can actually be pretty hard to determine. But here, we'll focus on one of the first steps in the analysis, mapping probes to genes.

In the **isb-cgc:platform_reference.methylation_annotation** table schema, we find a RECORD called 'UCSC'. If we look at the details of the table (via the web interface) we see that the table has 485,577 rows and has the following description:

```
This table is based on the Illumina DNA methylation platform annotation information
found in the file HumanMethylation450_15017482_v.1.2.csv which can be obtained
from Illumina.  This information has been loaded into a BigQuery table and made
available to the public with permission from Illumina.
```

Sounds good! A couple important columns are going to be the IlmnID, which is the probe ID (example: cg10232580), and the UCSC RECORD, where we'll find the gene symbol, RefGene accession ID, and the portion of the gene the probe is closest to (approximately).

Let's start with an easy one:

```
SELECT
  Infinium_Design_Type,
  COUNT(Infinium_Design_Type) as type_count
FROM
  `isb-cgc.platform_reference.methylation_annotation`
GROUP BY
  Infinium_Design_Type
```

Row	Infinium_Design_Type	type_count
1	I	135501
2	II	350076

Why does that matter? Well, the array was actually a blend of two different technologies. This paper and this paper show that the performance of the two probes is very different, and that type II probes appear to be less useful than the type I probes.

Now, let's suppose we are interested in a particular pathway, and we'd like to know the distribution of probe types across the pathway genes. Using our previous 'Query of the Month' data set (isb-cgc:QotM.WikiPathways_20170425_Annotated), we can get a list of functionally related genes.

```
SELECT
  DISTINCT(Symbol) as gene_symbol
```

(continues on next page)

(continued from previous page)

```
FROM
  `isb-cgc.QotM.WikiPathways_20170425_Annotated`
WHERE
  pathway = 'Oxidative Damage'
```

and we get 40 genes. So now we're going to join the annotation table, to our table of pathway related genes, and get the probe types.

```
WITH
pathway AS (
SELECT
  DISTINCT(Symbol) as gene_symbol
FROM
  `isb-cgc.QotM.WikiPathways_20170425_Annotated`
WHERE
  pathway = 'Oxidative Damage'
  )
SELECT
    Infinium_Design_Type,
    COUNT(Infinium_Design_Type)
FROM
    `isb-cgc.platform_reference.methylation_annotation` as m
JOIN
    pathway
ON
    pathway.gene_symbol = m.UCSC.RefGene_Name
GROUP BY
  Infinium_Design_Type
```

```
Query Failed
Error: Cannot access field RefGene_Name on a value with type
ARRAY<STRUCT<RefGene_Group STRING, RefGene_Accession STRING, RefGene_Name STRING>> at␣
↪[18:34]
```

What happened? It's that darned RECORD, which in our error, actually looks to be an ARRAY of STRUCTS! We have previously used arrays in our queries in past months where we took a list of values and created an array to be passed to a JavaScript function. The result of the function gave us back an array, and we had to UNNEST it, to get back one row per entry. It's similar in this instance. Some probes are mapped to multiple RefGene_Accession IDs. For example, cg10241718 maps to NM_033302, NM_033303, NM_033304, and NM_000680. Interestingly, you see this same set as part of the HG-U133A probe annotations (Thanks, genecard-geneannot webservice). These are representing four different transcripts of the same gene ADRA1A, the methylation probe has the same relationship to three of the isoforms (the gene body), but for one isoform, NM_000680, the probe is positioned 3'-UTR, which could change its effect. In light of that, we might want to group by gene symbol (mostly the same) and the refgene_group, which tells us the relative position of the probe to the gene.

To (finally!) address the problem of RECORDS, we need to check the BigQuery docs There, we see that the RECORD in Legacy SQL has becomes a STRUCT in Standard SQL. In order to flatten the table, in Legacy SQL we would use FLATTEN, but now, in Standard SQL we are going to use UNNEST.

So what's the difference between an ARRAY and a STRUCT? Well an ARRAY is "an ordered list of zero or more elements of non-ARRAY values," and a STRUCT is a "container of ordered fields each with a type." Hmmm, sounds pretty similar, the difference being that a STRUCT can be a collection of different data types (STRINGS and INTs for example), while ARRAYs have to be a single data type.

To get around that, we are going to flatten the table using UNNEST.

```
SELECT
  RefGene_Name,
  RefGene_Group
FROM
  `isb-cgc.platform_reference.methylation_annotation`,
  UNNEST(UCSC)
LIMIT
  1000
```

Row	RefGene_Name	RefGene_Group
1	null	null
2	null	null
3	IQCE	Body
4	IQCE	Body
5	CRYGN	3'UTR
6	IQCE	Body
7	IQCE	Body
8	ELFN1	5'UTR

That's more like it! Now we can write our final query.

```
WITH
  -- first we'll UNNEST our probes
  --
  probes AS (
  SELECT
    RefGene_Name,
    RefGene_Group,
    Infinium_Design_Type
  FROM
    `isb-cgc.platform_reference.methylation_annotation`,
    UNNEST(UCSC) ),
  --
  -- Then we'll select genes taking part in our pathway of interest
  --
  genes AS (
  SELECT
    DISTINCT(Symbol) AS gene_symbol
  FROM
    `isb-cgc.QotM.WikiPathways_20170425_Annotated`
  WHERE
    pathway = 'Oxidative Damage' ),
  --
  -- We can join the unnested table to our table of genes.
  --
  join_table AS (
  SELECT
    genes.gene_symbol,
    probes.RefGene_Group,
    probes.Infinium_Design_Type
  FROM
```

(continues on next page)

(continued from previous page)

```
    probes
  JOIN
    genes
  ON
    genes.gene_symbol = probes.RefGene_Name
  GROUP BY
    genes.gene_symbol,
    probes.RefGene_Group,
    probes.Infinium_Design_Type )
--
-- And summarize on the probe type.
--
SELECT
  Infinium_Design_Type,
  COUNT(Infinium_Design_Type),
  RefGene_Group
FROM
  join_table
GROUP BY
  RefGene_Group,
  Infinium_Design_Type
```

Row	Infinium_Design_Type	type_count	RefGene_Group
1	I	3	3'UTR
2	I	17	TSS1500
3	I	20	TSS200
4	I	20	Body
5	I	20	1stExon
6	I	23	5'UTR
7	II	25	5'UTR
8	II	25	1stExon
9	II	29	TSS200
10	II	31	3'UTR
11	II	38	TSS1500
12	II	39	Body

OK! So, this pathway is covered by probes of both types, and we do see more of the type II probes (which lack in performance), but there's also a good number of type I probes that should be useful.

So, in summary, when using the ISB-CGC tables, you probably won't run into too many RECORD data types, but if you do, you'll be prepared.

As an exercise for the reader, you might want to try and join the information explored above with the information in the one of the GENCODE tables – try using the methylation probe coordinates and the GENCODE gene coordinates to see if the information in the UCSC record in the methylation table is completely accurate, or check to see if there are important differences between hg19/GRCh37 and hg38/Grch38. If you come up with some useful queries, feel free to email us and we'll feature you on this page!

May, 2017

This month we are going to extend the query from April and focus on estimating the distance between samples based on shared mutations in pathways. To clarify, we want to know, given a particular pathway, such as the WNT signaling pathway, whether two samples share deleterious mutations within that pathway. In April, we were comparing samples based on shared mutations, but in considering all genes simultaneously, we had some pretty low Jaccard indices.

A second goal will be to create a set of pathways, for each sample, where pathways contain at least one potentially harmful mutation. Then we will again estimate the distance between samples based on the set of (potentially) altered pathways.

New for this month, we also have a whole host of new BigQuery tables from COSMIC.

For our query this month, we downloaded 381 pathways from WikiPathways. In the BQ table, each row contains a pathway and a gene associated with that pathway.

For this portion of the work, I wrote a small python script to parse .gmt files to output a 'tidy' (format), which is required for uploading to BigQuery. Then with this file, I used the BQ web interface uploader. To upload a table, clicking the '+' symbol next to a dataset reveals the 'Create Table' interface. For smaller files, we can upload it directly, whereas with larger files, we need to move it to cloud storage first. After giving it a table name, and with some luck, we can just click the 'Automatically detect' schema check box. I've been having good luck with it, but you might run into trouble if the 'top' of a column looks like an integer, but the actual type is a 'string'.

I've created a table with a column listing the pathway name, and a second column listing the genes associated with the pathway. I used the org.Hs.eg.db human database of gene identifiers found in Bioconductor to map the gene IDs to a few often used variants.

For this analysis, first I will select a few pathways that are well known and often important in cancer processes, then we'll move to using all pathways. And towards the end, we'll look at all pathways and all studies!

```
SELECT
  Symbol
FROM
  `isb-cgc:QotM.WikiPathways_20170425_Annotated`
WHERE
  pathway = 'Notch Signaling Pathway'
GROUP BY
  Symbol
```

The above query returns 79 gene symbols. Let's see how many variants are found in this pathway.

```
WITH
  pathGenes AS (
    SELECT
      Symbol
    FROM
      `isb-cgc.QotM.WikiPathways_20170425_Annotated`
    WHERE
      pathway = 'Notch Signaling Pathway'
    GROUP BY
      Symbol
  ),
  varsMC3 AS (
    SELECT
      project_short_name,
      case_barcode,
```

(continues on next page)

(continued from previous page)

```
      Hugo_Symbol
    FROM
      `isb-cgc.TCGA_hg19_data_v0.Somatic_Mutation_MC3`
    WHERE
      Variant_Type = 'SNP'
      AND Consequence = 'missense_variant'
      AND biotype = 'protein_coding'
      AND SWISSPROT IS NOT NULL
      AND REGEXP_CONTAINS(PolyPhen, 'damaging')
      AND REGEXP_CONTAINS(SIFT, 'deleterious')
      AND Hugo_Symbol IN (select Symbol as Hugo_Symbol from pathGenes)
    GROUP BY
      project_short_name,
      case_barcode,
      Hugo_Symbol
  )
--
--
SELECT
  project_short_name,
  COUNT(*) AS N_vars
FROM
  varsMC3
GROUP BY
  project_short_name
ORDER BY
  N_vars DESC
```

Row	project_short_name	N_genes
1	TCGA-UCEC	805
2	TCGA-SKCM	466
3	TCGA-COAD	261
4	TCGA-STAD	246
5	TCGA-LUSC	219
6	TCGA-LUAD	204
7	TCGA-HNSC	179

So there are quite a few variants found in this pathway. Let's find out a little more information about them. I'm going to replace the last 'select' statement of the above query to look at the returned rows. Also, similar to last month, we're going to look at a small subset of cancer types to ensure the queries come back quickly.

```
SELECT
  Hugo_Symbol,
  count(Hugo_Symbol) as gene_count
FROM
  varsMC3
WHERE
  project_short_name = 'TCGA-COAD'
group by
  Hugo_Symbol
order by
  gene_count dESC
```

These counts show that some genes are mutated more often than others. In COAD, FBXW7 is mutated more than twice as often as the next most mutated gene, NOTCH1. Both of these genes are well known among cancer researchers.

Row	Hugo_Symbol	gene_count
1	FBXW7	38
2	NOTCH1	15
3	NOTCH3	13
4	JAG1	9
5	MAPT	9
6	NOTCH2	8

OK, let's compute a Jaccard index based on this pathway!

```
WITH
  --
  -- First we define our pathway of interest.
  --
  pathGenes AS (
    SELECT
      Symbol as Hugo_Symbol
    FROM
      `isb-cgc.QotM.WikiPathways_20170425_Annotated`
    WHERE
      pathway = 'Notch Signaling Pathway'
    GROUP BY
      Symbol
  ),
  --
  -- Then we're going to extract just the project names, cases, and gene symbols,
  -- using the "GROUP BY" to make sure we only count one mutation per gene per case
  -- and we'll just take genes that are in the pathway.
  --
  firstVars AS (
  SELECT
    project_short_name,
    case_barcode,
    Hugo_Symbol
  FROM
    `isb-cgc.TCGA_hg19_data_v0.Somatic_Mutation_MC3`
  WHERE
    Variant_Type = 'SNP'
    AND Hugo_Symbol IN (select Hugo_Symbol from pathGenes)
    AND Consequence = 'missense_variant'
    AND biotype = 'protein_coding'
    AND ( REGEXP_CONTAINS(PolyPhen, 'damaging')
      OR REGEXP_CONTAINS(SIFT, 'deleterious') )
    AND project_short_name IN ('TCGA-PAAD', 'TCGA-GBM', 'TCGA-LGG')
    -- We could remove the above line to compute using all samples,
    -- but to speed things up, let's just look at 3 studies.
  GROUP BY
    project_short_name,
    case_barcode,
    Hugo_Symbol ),
  --
```

(continues on next page)

(continued from previous page)

```
  -- Next we transform resulting table using the ARRAY_AGG function
  -- to create a list of mutated genes for each case
  --
  arrayMC3 AS (
  SELECT
    project_short_name,
    case_barcode,
    ARRAY_AGG(DISTINCT Hugo_Symbol) AS geneArray
  FROM
    firstVars
  GROUP BY
    project_short_name,
    case_barcode ),
  --
  -- Now we can do some "set operations" on these gene-lists:  a self-join
  -- of the previously created table with itself will allow for a pairwise
  -- pairwise comparison (notice the inequality in the JOIN ... ON clause)
  --
  setOpsTable AS (
  SELECT
    a.case_barcode AS case1,
    a.project_short_name AS study1,
    ARRAY_LENGTH(a.geneArray) AS length1,
    b.case_barcode AS case2,
    b.project_short_name AS study2,
    ARRAY_LENGTH(b.geneArray) AS length2,
    --
    -- here's the intersection
    (SELECT
      COUNT(1) FROM UNNEST(a.geneArray) AS ga JOIN UNNEST(b.geneArray) AS gb ON ga = gb)
        AS gene_intersection,
    --
    -- and here's the union
    (SELECT
      COUNT(DISTINCT gx) FROM UNNEST(ARRAY_CONCAT(a.geneArray,b.geneArray)) AS gx)
        AS gene_union
  FROM
    arrayMC3 AS a
  JOIN
    arrayMC3 AS b
  ON
    a.case_barcode < b.case_barcode )
  --
  -- and finally, we can compute the Jaccard index, and
  -- do a little bit of filtering and then output a list of
  -- pairs, sorted based on the Jaccard index:
SELECT
  case1,
  study1,
  length1 AS geneCount1,
  case2,
  study2,
```

(continues on next page)

(continued from previous page)

```
  length2 AS geneCount2,
  gene_intersection,
  gene_union,
  (gene_intersection / gene_union) AS jaccard_index
FROM
  setOpsTable
WHERE
  (gene_intersection / gene_union) > 0.1
  AND gene_intersection > 5
ORDER BY
  jaccard_index DESC
```

Row	case1	study1	geneCount1	case2	study2	geneCount2	gene_intersection	gene_union	jaccard_index
1	TCGA-06-5416	TCGA-GBM	30	TCGA-IB-7651	TCGA-PAAD	34	20	44	0.45454545454545453
2	TCGA-DU-6392	TCGA-LGG	21	TCGA-IB-7651	TCGA-PAAD	34	13	42	0.30952380952380953
3	TCGA-06-5416	TCGA-GBM	30	TCGA-19-5956	TCGA-GBM	16	10	36	0.2777777777777778
4	TCGA-06-5416	TCGA-GBM	30	TCGA-DU-6392	TCGA-LGG	21	11	40	0.275
5	TCGA-19-5956	TCGA-GBM	16	TCGA-DU-6392	TCGA-LGG	21	7	30	0.23333333333333334
6	TCGA-19-5956	TCGA-GBM	16	TCGA-IB-7651	TCGA-PAAD	34	8	42	0.19047619047619047

So, it's very interesting that we are getting samples from GBM (brain) and PAAD (pancreas) with high overlaps in the gene sets. But it makes sense since the Notch signaling pathway was been implciated in both of these cancer types.

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4283135/

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4621772/

Now we'll move on to the COSMIC data, where we will compare a GBM sample to various non-TCGA samples in COSMIC.

```
WITH
  --
  -- First we define our pathway of interest.
  --
  pathGenes AS (
    SELECT
      Symbol as Hugo_Symbol
    FROM
      `isb-cgc.QotM.WikiPathways_20170425_Annotated`
    WHERE
      pathway = 'Notch Signaling Pathway'
    GROUP BY
      Symbol
  ),
  --
  -- Then we'll select a single TCGA sample, with filters similar to the above.
  --
  tcgaSample AS (
  SELECT
    sample_barcode_tumor,
    Hugo_Symbol
  FROM
    `isb-cgc.TCGA_hg19_data_v0.Somatic_Mutation_MC3`
```

(continues on next page)

(continued from previous page)

```
WHERE
  sample_barcode_tumor = 'TCGA-06-5416-01A'
  AND Hugo_Symbol IN (select Hugo_Symbol from pathGenes)
  AND Variant_Type = 'SNP'
  AND Consequence = 'missense_variant'
  AND biotype = 'protein_coding'
  AND ( REGEXP_CONTAINS(PolyPhen, 'damaging')
    OR REGEXP_CONTAINS(SIFT, 'deleterious') )
GROUP BY
  sample_barcode_tumor,
  Hugo_Symbol),
--
-- Then we'll create a sub-table of COSMIC samples, sans TCGA.
--
cosmicSample AS (
SELECT
  Sample_name,
  Primary_site,
  Primary_histology,
  Sample_source,
  Gene_name
FROM
  `isb-cgc.COSMIC.grch37_v80`
WHERE
  STARTS_WITH(Sample_name, "TCGA") = FALSE
  AND Mutation_Description = 'Substitution - Missense'
  AND FATHMM_prediction = "PATHOGENIC"
  AND Gene_name IN (select Hugo_symbol from pathGenes)
GROUP BY
  Sample_name,
  Primary_site,
  Primary_histology,
  Sample_source,
  Gene_name ),
--
-- Then we make the array of genes for the TCGA sample.
--
tcgaSampleArray AS (
SELECT
  sample_barcode_tumor,
  ARRAY_AGG(DISTINCT Hugo_Symbol) AS geneArray
FROM
  tcgaSample
GROUP BY
  sample_barcode_tumor ),
--
-- Then we make the array of genes for each cosmic sample.
--
cosmicSampleArray AS (
SELECT
  Sample_name,
  Primary_site,
```

(continues on next page)

(continued from previous page)

```
    Primary_histology,
    Sample_source,
    ARRAY_AGG(DISTINCT Gene_name) AS geneArray
  FROM
    cosmicSample
  GROUP BY
    Sample_name,
    Primary_site,
    Primary_histology,
    Sample_source ),
  --
  -- Next we can perform our set operations on the arrays.
  --
  setOpsTable AS (
  SELECT
    a.sample_barcode_tumor AS tcgaSample,
    b.Sample_name AS cosmicSample,
    b.Primary_site,
    b.Primary_histology,
    b.Sample_source,
    ARRAY_LENGTH(a.geneArray) AS length1,
    ARRAY_LENGTH(b.geneArray) AS length2,
    (SELECT COUNT(1) FROM UNNEST(a.geneArray) AS ga JOIN UNNEST(b.geneArray) AS gb ON ga␣
↪= gb) AS gene_intersection,
    (SELECT COUNT(DISTINCT gx) FROM UNNEST(ARRAY_CONCAT(a.geneArray,b.geneArray)) AS gx)␣
↪AS gene_union
  FROM
    tcgaSampleArray AS a
  JOIN
    cosmicSampleArray AS b
  ON
    a.sample_barcode_tumor < b.Sample_name )
  --
  -- And build our final results.
  --
SELECT
  tcgaSample,
  length1 AS geneCount1,
  cosmicSample,
  Primary_site,
  Primary_histology,
  length2 AS geneCount2,
  gene_intersection AS intersection,
  gene_union,
  (gene_intersection / gene_union) AS jaccard_index
FROM
  setOpsTable
WHERE
  (gene_intersection / gene_union) > 0.00
  AND gene_intersection > 1
  AND gene_union > 1
ORDER BY
```

(continues on next page)

(continued from previous page)

```
  jaccard_index DESC
```

tcgaSample	geneCount1	cosmicSample	Primary_site	Primary_histology	geneCount2	intersection	gene_union	jaccard_index
TCGA-06-5416-01A	30	YUKLAB	skin	malignant_melanoma	9	7	32	0.21875
TCGA-06-5416-01A	30	YUKAT	skin	malignant_melanoma	11	7	34	0.20588235294117646
TCGA-06-5416-01A	30	cSCCP6	skin	carcinoma	8	6	32	0.1875
TCGA-06-5416-01A	30	YULAN	skin	malignant_melanoma	9	5	34	0.14705882352941177
TCGA-06-5416-01A	30	WSU-HN8	upper_aerodigestive_tract	carcinoma	12	5	37	0.13513513513513514
TCGA-06-5416-01A	30	sysucc-880T	large_intestine	carcinoma	5	4	31	0.12903225806451613

So, for this particular pathway, the Jaccard indices are not spectacular. But(!), what we really want is to look at *all* pathways simultaneously. Then for any given pair of samples, we could rank the mutation overlap by pathway. To do that, instead of selecting a pathway in the first subtable… we build a table containing all pathways, and join on that further down in the query.

Just note, this is a longer running query (takes about 2 minutes).

```
WITH
  --
  -- First we make a table with pathways and genes.
  --
  pathGenes AS (
    SELECT
      pathway,
      Symbol as Hugo_Symbol
    FROM
      `isb-cgc.QotM.WikiPathways_20170425_Annotated`
    GROUP BY
      Symbol,
      pathway
  ),
  --
  -- Then we're going to extract just the project names, cases, and gene symbols,
  -- using the "GROUP BY" to make sure we only count one mutation per gene per case
  -- and we'll join to the above pathway table.
  --
  firstVars AS (
  SELECT
    a.project_short_name,
    a.case_barcode,
    a.Hugo_Symbol,
    b.pathway
  FROM
    `isb-cgc.TCGA_hg19_data_v0.Somatic_Mutation_MC3` as a
  join
    pathGenes as b
  on
    a.Hugo_Symbol = b.Hugo_Symbol
  WHERE
    Variant_Type = 'SNP'
    AND Consequence = 'missense_variant'
    AND biotype = 'protein_coding'
```

(continues on next page)

(continued from previous page)

```
    AND ( REGEXP_CONTAINS(PolyPhen, 'damaging')
      OR REGEXP_CONTAINS(SIFT, 'deleterious') )
    AND project_short_name IN ('TCGA-PAAD', 'TCGA-GBM', 'TCGA-LGG', 'TCGA-BRCA', 'TCGA-
↪KIRC')
    -- We could remove the above line to compute using all samples,
    -- but to speed things up, let's just look at 3 studies.
  GROUP BY
    project_short_name,
    case_barcode,
    Hugo_Symbol,
    pathway
    ),
  --
  -- Next we transform resulting table using the ARRAY_AGG function
  -- to create a list of mutated genes for each case
  --
  arrayMC3 AS (
  SELECT
    project_short_name,
    case_barcode,
    pathway,
    ARRAY_AGG(DISTINCT Hugo_Symbol) AS geneArray
  FROM
    firstVars
  GROUP BY
    project_short_name,
    case_barcode,
    pathway
    ),
  --
  -- Now we can do some "set operations" on these gene-lists:  a self-join
  -- of the previously created table with itself will allow for a pairwise
  -- pairwise comparison (notice the inequality in the JOIN ... ON clause)
  --
  setOpsTable AS (
  SELECT
    a.case_barcode AS case1,
    a.project_short_name AS study1,
    a.pathway as pathway,
    ARRAY_LENGTH(a.geneArray) AS length1,
    b.case_barcode AS case2,
    b.project_short_name AS study2,
    ARRAY_LENGTH(b.geneArray) AS length2,
    --
    -- here's the intersection
    (SELECT
      COUNT(1) FROM UNNEST(a.geneArray) AS ga JOIN UNNEST(b.geneArray) AS gb ON ga = gb)
        AS gene_intersection,
    --
    -- and here's the union
    (SELECT
      COUNT(DISTINCT gx) FROM UNNEST(ARRAY_CONCAT(a.geneArray,b.geneArray)) AS gx)
```

(continues on next page)

(continued from previous page)

```
        AS gene_union
  FROM
    arrayMC3 AS a
  JOIN
    arrayMC3 AS b
  ON
    a.case_barcode < b.case_barcode AND
    a.pathway = b.pathway
    )
  --
  -- and finally, we can compute the Jaccard index, and
  -- do a little bit of filtering and then output a list of
  -- pairs, sorted based on the Jaccard index:
SELECT
  pathway,
  case1,
  study1,
  length1 AS geneCount1,
  case2,
  study2,
  length2 AS geneCount2,
  gene_intersection,
  gene_union,
  (gene_intersection / gene_union) AS jaccard_index
FROM
  setOpsTable
WHERE
  (gene_intersection / gene_union) > 0.3
  AND gene_intersection > 10
ORDER BY
  jaccard_index DESC
```

Row	pathway	case1	study1	geneCount1	case2	study2	geneCount2	gene_intersection	gene_union	jaccard_index
1	Histone Modifications	TCGA-19-5956	TCGA-GBM	17	TCGA-DU-6392	TCGA-LGG	20	13	24	0.5416666666666666
2	Alpha 6 Beta 4 signaling pathway	TCGA-06-5416	TCGA-GBM	17	TCGA-DU-6392	TCGA-LGG	15	11	21	0.5238095238095238
3	Histone Modifications	TCGA-06-5416	TCGA-GBM	15	TCGA-19-5956	TCGA-GBM	17	11	21	0.5238095238095238
4	Histone Modifications	TCGA-06-5416	TCGA-GBM	15	TCGA-DU-6392	TCGA-LGG	20	12	23	0.5217391304347826
5	Interferon type I signaling pathways	TCGA-06-5416	TCGA-GBM	22	TCGA-IB-7651	TCGA-PAAD	20	14	28	0.5
6	Kit receptor signaling pathway	TCGA-19-5956	TCGA-GBM	15	TCGA-DU-6392	TCGA-LGG	20	11	24	0.4583333333333333

OK! Now we've got some pretty decent overlaps. We now have a way to search for similarities among groups of samples based on functionally based shared mutation profiles.

Turning this question around a little bit, what if we looked at the overlap on the pathway level instead?

```
WITH
  --
  -- First we'll join the filtered somatic mutation table to the
  -- table of pathways.
  --
  vars AS (
  SELECT
    mc3.project_short_name,
```

(continues on next page)

(continued from previous page)

```
  mc3.case_barcode,
  mc3.Hugo_Symbol,
  wikip.pathway
FROM
  `isb-cgc.TCGA_hg19_data_v0.Somatic_Mutation_MC3` as mc3
join
  `isb-cgc.QotM.WikiPathways_20170425_Annotated` as wikip
ON
  mc3.Hugo_Symbol = wikip.Symbol
WHERE
  mc3.Variant_Type = 'SNP'
  AND mc3.Consequence = 'missense_variant'
  AND mc3.biotype = 'protein_coding'
  AND ( REGEXP_CONTAINS(mc3.PolyPhen, 'damaging')
    OR REGEXP_CONTAINS(mc3.SIFT, 'deleterious') )
  AND mc3.project_short_name IN ('TCGA-BRCA', 'TCGA-GBM')
  -- We could remove the above line to compute using all samples,
  -- but to speed things up, let's just look at 3 studies.
GROUP BY
  mc3.project_short_name,
  mc3.case_barcode,
  mc3.Hugo_Symbol,
  wikip.pathway),
--
-- Next we transform resulting table using the ARRAY_AGG function
-- to create a list of pathways for each case, where each pathway
-- contains at least one mutated gene.
--
arrayPath AS (
SELECT
  project_short_name,
  case_barcode,
  ARRAY_AGG(DISTINCT pathway) AS pathArray
FROM
  vars
GROUP BY
  project_short_name,
  case_barcode ),

--
-- Now we can do some "set operations" on these pathway-lists:  a self-join
-- of the previously created table with itself will allow for a pairwise
-- pairwise comparison (notice the inequality in the JOIN ... ON clause)
--
setOpsTable AS (
SELECT
  a.case_barcode AS case1,
  a.project_short_name AS study1,
  ARRAY_LENGTH(a.pathArray) AS length1,
  b.case_barcode AS case2,
  b.project_short_name AS study2,
  ARRAY_LENGTH(b.pathArray) AS length2,
```

(continues on next page)

(continued from previous page)

```
    --
    -- here's the intersection
    (SELECT
      COUNT(1) FROM
        UNNEST(a.pathArray) AS ga JOIN UNNEST(b.pathArray) AS gb ON ga = gb)
          AS path_intersection,
    --
    -- and here's the union
    (SELECT
      COUNT(DISTINCT gx) FROM
        UNNEST(ARRAY_CONCAT(a.pathArray,b.pathArray)) AS gx)
          AS path_union
  FROM
    arrayPath AS a
  JOIN
    arrayPath AS b
  ON
    a.case_barcode < b.case_barcode )
  --

  --
  -- and finally, we can compute the Jaccard index, and
  -- do a little bit of filtering and then output a list of
  -- pairs, sorted based on the Jaccard index:
SELECT
  case1,
  study1,
  length1 AS pathCount1,
  case2,
  study2,
  length2 AS pathCount2,
  path_intersection,
  path_union,
  (path_intersection / path_union) AS jaccard_index
FROM
  setOpsTable
WHERE
  (path_intersection / path_union) > 0.8
  AND path_intersection > 10
ORDER BY
  jaccard_index DESC
```

Row	case1	study1	pathCount1	case2	study2	pathCount2	path_intersection	path_union	jaccard_index
1	TCGA-AR-A1AX	TCGA-BRCA	38	TCGA-EW-A1P1	TCGA-BRCA	38	38	38	1
2	TCGA-AR-A0TR	TCGA-BRCA	33	TCGA-E2-A153	TCGA-BRCA	33	33	33	1
3	TCGA-A2-A04V	TCGA-BRCA	31	TCGA-AC-A3YJ	TCGA-BRCA	31	31	31	1
4	TCGA-A2-A0YI	TCGA-BRCA	31	TCGA-D8-A3Z6	TCGA-BRCA	31	31	31	1
5	TCGA-A2-A04V	TCGA-BRCA	31	TCGA-A2-A0YI	TCGA-BRCA	31	31	31	1
6	TCGA-A2-A0YI	TCGA-BRCA	31	TCGA-E2-A1L6	TCGA-BRCA	31	31	31	1
7	TCGA-AN-A0FF	TCGA-BRCA	31	TCGA-D8-A3Z6	TCGA-BRCA	31	31	31	1
8	TCGA-AC-A3YJ	TCGA-BRCA	31	TCGA-E2-A1L6	TCGA-BRCA	31	31	31	1

Row	case1	study1	pathCount1	case2	study2	pathCount2	path_intersection	path_union	jaccard_index
35	TCGA-B6-A0IC	TCGA-BRCA	33	TCGA-E9-A1R3	TCGA-BRCA	32	32	33	0.96969696969696972
36	TCGA-AO-A0JE	TCGA-BRCA	63	TCGA-BH-A0BF	TCGA-BRCA	65	63	65	0.96923076923076923
37	TCGA-A2-A0T7	TCGA-BRCA	32	TCGA-AC-A3YJ	TCGA-BRCA	31	31	32	0.96875
38	TCGA-AC-A3YJ	TCGA-BRCA	31	TCGA-LL-A50Y	TCGA-BRCA	32	31	32	0.96875
39	TCGA-A2-A04V	TCGA-BRCA	31	TCGA-BH-A0BV	TCGA-BRCA	32	31	32	0.96875
40	TCGA-A2-A0YI	TCGA-BRCA	31	TCGA-BH-A0W3	TCGA-BRCA	32	31	32	0.96875
41	TCGA-A2-A04V	TCGA-BRCA	31	TCGA-BH-A0BM	TCGA-BRCA	32	31	32	0.96875
42	TCGA-AC-A3YJ	TCGA-BRCA	31	TCGA-E9-A1R3	TCGA-BRCA	32	31	32	0.96875

Wow, those are some excellent jaccard indices! Considering that we started with just over 300 pathways, we have samples perfectly or nearly perfectly in agreement.

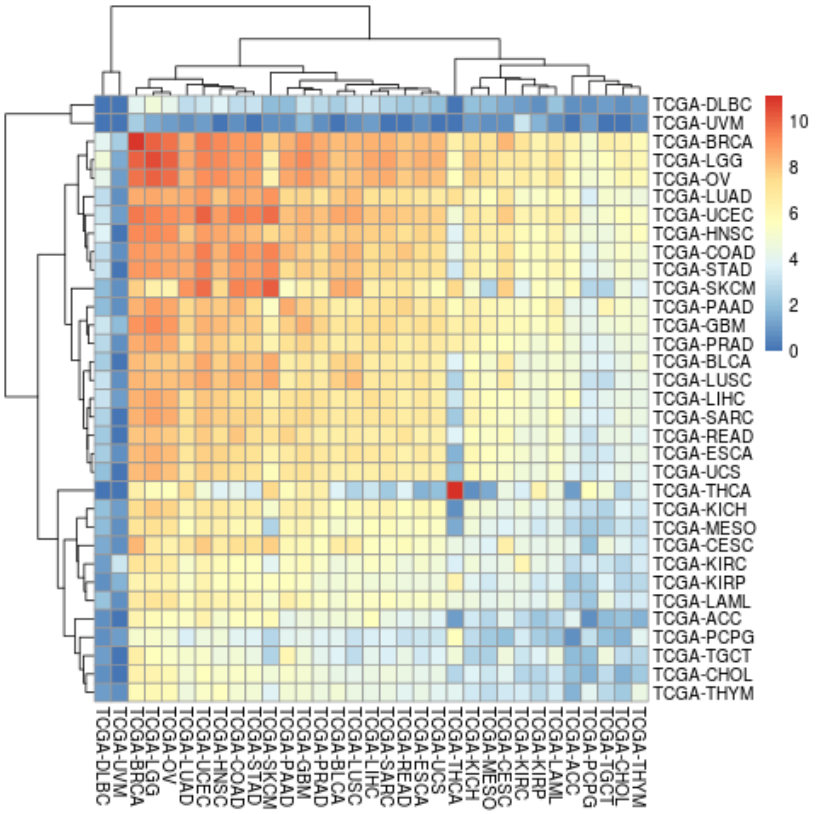
Next, let's see how the samples are associating on a tissue level. I'm going to add this query to the end of the above query, to tabulate how often study1 and study2 agree regarding tissue of origin (between BRCA and GBM).

```
SELECT
  table_cell,
  COUNT(*) AS n
FROM (
  SELECT (
    CASE
      WHEN study1 = 'TCGA-BRCA' AND study2 = 'TCGA-BRCA' THEN 'BRCA-BRCA'
      WHEN study1 = 'TCGA-BRCA' AND study2 = 'TCGA-GBM' THEN 'BRCA-GBM'
      WHEN study1 = 'TCGA-GBM' AND study2 = 'TCGA-BRCA' THEN 'GBM-BRCA'
      WHEN study1 = 'TCGA-GBM' AND study2 = 'TCGA-GBM' THEN 'GBM-GBM'
      ELSE 'None'
    END ) AS table_cell
  FROM
    jtable )
GROUP BY
  table_cell
ORDER BY
  n DESC
```

Row	table_cell	n
1	BRCA-BRCA	2246
2	GBM-BRCA	112
3	GBM-GBM	19

So, we see that the really high Jaccard indices are coming (mostly) from BRCA-BRCA sample comparisons.

With one additional change to the above query we can query across all TCGA studies rather than just a few. The change involves removing the `mc3.project_short_name IN ('TCGA-BRCA', 'TCGA-GBM')` conditional. This is a good trick that works in many cases. By removing the conditional, instead of querying on just a few, we query across all studies, letting each sample be paired with every other. Doing this, and setting the Jaccard index threshold to 0.5, we get > 800K rows of results back, where each sample pair is compared on the similarity of their potentially disrupted pathways. Bringing the results into R, I created a heatmap showing how many TCGA study pairs were among the results. We see some tissue types are most similar to only that type of tissue, whereas other tissue types share patterns of disrupted pathways.



April, 2017

In this month's query, we are going to look at two new data sources. The first is the MC3 somatic mutation table, and the second is the COSMIC mutation database. The objective is to compute a similarity metric based on overlapping mutations between samples. First we'll look at pairwise similarity among TCGA samples, and then we'll pick a single TCGA sample and search for a matching COSMIC sample.

The MC3 table comes from the TCGA Pan-Cancer effort, a multi-center project aiming to analyze all 33 TCGA tumor-types together. This somatic mutation calls table is based on the unified call set recently published by the TCGA Network. (For more details or the original source file, please check Synapse.)

The COSMIC (Catalogue Of Somatic Mutations In Cancer) data comes from the Wellcome Trust Sanger Institue and represents the *"the world's largest and most comprehensive resource for exploring the impact of somatic mutations in human cancer"*.

To compute a similarity score between any two samples, we'll use the Jaccard index, in which the intersection is divided by the union, so that samples with no overlap in mutations will have a Jaccard index of 0, while samples with some overlap will have a Jaccard index between 0 and 1.

We'll start with the MC3 table – which includes the predicted effect of each mutation call. The mutation might result in a change in the amino acid sequence (non-synonomous), or introduce a new stop codon (stop insert), or no amino-acid change (synonomous). In this work we're going to focus on single nucleotide polymorphisms (SNPs).

First, lets see what kind of "consequences" are present in this table:

```
SELECT
  Consequence,
  count (1) AS n
```

(continues on next page)

(continued from previous page)

```
FROM
  `isb-cgc.TCGA_hg19_data_v0.Somatic_Mutation_MC3`
WHERE
  Variant_Type = 'SNP'
GROUP BY
  Consequence
ORDER BY
  n DESC
```

Row	Consequence	n
1	missense_variant	1921705
2	synonymous_variant	781564
3	3_prime_UTR_variant	253582
4	stop_gained	156768
5	intron_variant	86347
6	5_prime_UTR_variant	77069
7	non_coding_transcript_exon_variant	46761
8	splice_acceptor_variant	29658
9	downstream_gene_variant	19048
10	splice_donor_variant	18239
11	splice_region_variant	15231
12	upstream_gene_variant	14990
13	start_lost	2718
14	stop_lost	2038
15	stop_retained_variant	1077

For the sake of simplicity, we're going to focus on the most common type of variant, the missense_variant which is more likely to have a functional impact through an alteration of the amino acid sequence.

Another question we might ask is: how are variants distributed across the tumor types (aka "studies" or "projects" within TCGA).

```
WITH
  firstMC3 AS (
  SELECT
    project_short_name,
    case_barcode,
    Hugo_Symbol
  FROM
    `isb-cgc.TCGA_hg19_data_v0.Somatic_Mutation_MC3`
  WHERE
    Variant_Type = 'SNP'
    AND Consequence = 'missense_variant'
    AND biotype = 'protein_coding'
    AND SWISSPROT IS NOT NULL
    AND REGEXP_CONTAINS(PolyPhen, 'damaging')
    AND REGEXP_CONTAINS(SIFT, 'deleterious')
  GROUP BY
    project_short_name,
    case_barcode,
    Hugo_Symbol )
```

(continues on next page)

(continued from previous page)

```
--
--
SELECT
  project_short_name,
  COUNT(*) AS N_genes
FROM
  firstMC3
GROUP BY
  project_short_name
ORDER BY
  N_genes DESC
```

Wow! The very high mutation counts for SKCM (melanoma) and LUAD (lung adenocarcinoma) may not be surprising, but the high mutation rate in endometial cancer (UCEC) may be less well known.

Row	project_short_name	N_genes
1	TCGA-UCEC	156877
2	TCGA-SKCM	112324
3	TCGA-LUAD	53119
4	TCGA-COAD	51072
5	TCGA-LUSC	44260
6	TCGA-STAD	44229
.		

OK, let's compute a Jaccard index across all samples in a few selected tumor-specific projects. Look for how the 'array' gets used.

```
WITH
  --
  -- first we're going to extract just the project names, cases, and gene symbols,
  -- using the "GROUP BY" to make sure we only count one mutation per gene per case
  firstMC3 AS (
  SELECT
    project_short_name,
    case_barcode,
    Hugo_Symbol
  FROM
    `isb-cgc.TCGA_hg19_data_v0.Somatic_Mutation_MC3`
  WHERE
    Variant_Type = 'SNP'
    AND Consequence = 'missense_variant'
    AND biotype = 'protein_coding'
    AND ( REGEXP_CONTAINS(PolyPhen, 'damaging')
      OR REGEXP_CONTAINS(SIFT, 'deleterious') )
    AND project_short_name IN ('TCGA-PAAD', 'TCGA-GBM', 'TCGA-LGG')
    -- We could remove the above line to compute using all samples,
    -- but to speed things up, let's just look at 3 studies.
  GROUP BY
    project_short_name,
    case_barcode,
    Hugo_Symbol ),
```

(continues on next page)

(continued from previous page)

```
--
-- next we transform resulting table using the ARRAY_AGG function
-- to create a list of mutated genes for each case
arrayMC3 AS (
SELECT
  project_short_name,
  case_barcode,
  ARRAY_AGG(DISTINCT Hugo_Symbol) AS geneArray
FROM
  firstMC3
GROUP BY
  project_short_name,
  case_barcode ),
--
-- now we can do some "set operations" on these gene-lists:  a self-join
-- of the previously created table with itself will allow for a pairwise
-- pairwise comparison (notice the inequality in the JOIN ... ON clause)
setOpsTable AS (
SELECT
  a.case_barcode AS case1,
  a.project_short_name AS study1,
  ARRAY_LENGTH(a.geneArray) AS length1,
  b.case_barcode AS case2,
  b.project_short_name AS study2,
  ARRAY_LENGTH(b.geneArray) AS length2,
  --
  -- here's the intersection
  (
  SELECT
    COUNT(1)
  FROM
    UNNEST(a.geneArray) AS ga
  JOIN
    UNNEST(b.geneArray) AS gb
  ON
    ga = gb) AS gene_intersection,
  --
  -- and here's the union
  (
  SELECT
    COUNT(DISTINCT gx)
  FROM
    UNNEST(ARRAY_CONCAT(a.geneArray,b.geneArray)) AS gx) AS gene_union
FROM
  arrayMC3 AS a
JOIN
  arrayMC3 AS b
ON
  a.case_barcode < b.case_barcode )
--
-- and finally, we can compute the Jaccard index, and
-- do a little bit of filtering and then output a list of
```

(continues on next page)

(continued from previous page)

```
  -- pairs, sorted based on the Jaccard index:
SELECT
  case1,
  study1,
  length1 AS geneCount1,
  case2,
  study2,
  length2 AS geneCount2,
  gene_intersection,
  gene_union,
  (gene_intersection / gene_union) AS jaccard_index
FROM
  setOpsTable
WHERE
  (gene_intersection / gene_union) > 0.1
  AND gene_intersection > 10
ORDER BY
  jaccard_index DESC
```

The top 5 results from the above query surprisingly find the highest similarity between a GBM (glioblastoma) sample and PAAD (pancreatic adenocarcinoma) sample. The net highest similarity is between a LGG (lower-grade glioma) sample and the same PAAD sample. (Recall that our query above had, somewhat randomly, chosen only GBM, LGG, and PAAD tumor-specific projects.)

Row	case1	study1	geneCount1	case2	study2	geneCount2	gene_intersection	gene_union	jaccard_index
1	TCGA-06-5416	TCGA-GBM	5221	TCGA-IB-7651	TCGA-PAAD	5877	2277	8821	0.2581339984128784
2	TCGA-DU-6392	TCGA-LGG	4781	TCGA-IB-7651	TCGA-PAAD	5877	2141	8517	0.25137959375366914
3	TCGA-06-5416	TCGA-GBM	5221	TCGA-DU-6392	TCGA-LGG	4781	1840	8162	0.22543494241607448
4	TCGA-06-5416	TCGA-GBM	5221	TCGA-19-5956	TCGA-GBM	3266	1390	7097	0.19585740453712835
5	TCGA-19-5956	TCGA-GBM	3266	TCGA-IB-7651	TCGA-PAAD	5877	1478	7665	0.19282452707110243
6	TCGA-19-5956	TCGA-GBM	3266	TCGA-DU-6392	TCGA-LGG	4781	1259	6788	0.18547436652916913

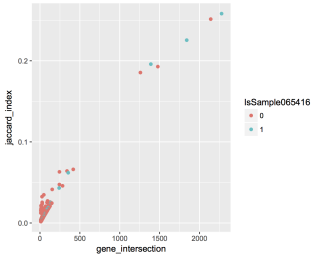


Fig. 1: Fig1. Each dot represents a pair of cases and the associated Jaccard index. The blue points show the pairs that involve the GBM case TCGA-06-5416.

Those unions look high to me. Let's double check them.

```
--
--
WITH
  g1 AS (
  SELECT
```

(continues on next page)

(continued from previous page)

```
    Hugo_Symbol
  FROM
    `isb-cgc.TCGA_hg19_data_v0.Somatic_Mutation_MC3`
  WHERE
    Variant_Type = 'SNP'
    AND Consequence = 'missense_variant'
    AND biotype = 'protein_coding'
    AND (REGEXP_CONTAINS(PolyPhen, 'damaging')
      OR REGEXP_CONTAINS(SIFT, 'deleterious'))
    AND case_barcode = 'TCGA-06-5416'
  GROUP BY
    Hugo_Symbol),
--
--
--
  g2 AS (
  SELECT
    Hugo_Symbol
  FROM
    `isb-cgc.TCGA_hg19_data_v0.Somatic_Mutation_MC3`
  WHERE
    Variant_Type = 'SNP'
    AND Consequence = 'missense_variant'
    AND biotype = 'protein_coding'
    AND (REGEXP_CONTAINS(PolyPhen, 'damaging')
       OR REGEXP_CONTAINS(SIFT, 'deleterious'))
    AND case_barcode = 'TCGA-IB-7651'
  GROUP BY
    Hugo_Symbol)

--
-- First the intersection,
--
SELECT
  count ( distinct a.Hugo_Symbol ) AS n
FROM
  g1 AS a
JOIN
  g2 AS b
ON
  a.Hugo_Symbol = b.Hugo_Symbol

UNION ALL -- to bring the intersection and union queries together

--
-- then the union.
--
SELECT
  count ( distinct Hugo_Symbol ) AS n
FROM
  (select * from  g1
     union all
```

(continues on next page)

(continued from previous page)

```
  select * from g2)

ORDER BY n
```

The above query returns 2277 (intersection) and 8821 (union), which is what we were expecting given the first row in the previous set of results.

Next we'll turn our attention to the COSMIC catalog. We will select a single sample, and perform the same Jaccard index across all samples in COSMIC (removing TCGA samples in COSMIC), and see what comes up. The sample we've selected for this next analysis is from the COAD project (Colon Adenocarcinoma).

Similar to the MC3 table, variants in COSMIC have been annotated. Let's take a look at what types of variants are present.

```
--
-- What kind of mutations are found in COSMIC?
--
SELECT
  Mutation_Description,
  count(1) AS n
FROM
  `isb-cgc.COSMIC.grch37_v80`
GROUP BY
  Mutation_Description
ORDER BY n DESC
```

Row	Mutation_Description	n
1	Substitution - Missense	3115431
2	Substitution - coding silent	1017162
3	Substitution - Nonsense	204293
4	Unknown	167135
5	Deletion - Frameshift	113237
6	Insertion - Frameshift	51345
7	Deletion - In frame	37833
8	Insertion - In frame	24870
9	Complex - deletion inframe	3212
10	Nonstop extension	2751
...	...	...

So, like above, we will focus on the most common type of variant, the Missense.

```
  --
  -- First we'll select a single TCGA sample, with filters similar to the above.
  --
WITH
  --
  tcgaSample AS (
  SELECT
    sample_barcode_tumor,
    Hugo_Symbol
  FROM
    `isb-cgc.TCGA_hg19_data_v0.Somatic_Mutation_MC3`
```

(continues on next page)

(continued from previous page)

```
WHERE
  sample_barcode_tumor = 'TCGA-CA-6718-01A'
  AND Variant_Type = 'SNP'
  AND Consequence = 'missense_variant'
  AND biotype = 'protein_coding'
  AND ( REGEXP_CONTAINS(PolyPhen, 'damaging')
    OR REGEXP_CONTAINS(SIFT, 'deleterious') )
GROUP BY
  sample_barcode_tumor,
  Hugo_Symbol),
--
-- Then we'll create a sub-table of COSMIC samples, sans TCGA.
--
cosmicSample AS (
SELECT
  Sample_name,
  Primary_site,
  Primary_histology,
  Sample_source,
  Gene_name
FROM
  `isb-cgc.COSMIC.grch37_v80`
WHERE
  STARTS_WITH(Sample_name, "TCGA") = FALSE
  AND Mutation_Description = 'Substitution - Missense'
  AND FATHMM_prediction = "PATHOGENIC"
GROUP BY
  Sample_name,
  Primary_site,
  Primary_histology,
  Sample_source,
  Gene_name ),
--
-- Then we make the array of genes for the TCGA sample.
--
tcgaSampleArray AS (
SELECT
  sample_barcode_tumor,
  ARRAY_AGG(DISTINCT Hugo_Symbol) AS geneArray
FROM
  tcgaSample
GROUP BY
  sample_barcode_tumor ),
--
-- Then we make the array of genes for each cosmic sample.
--
cosmicSampleArray AS (
SELECT
  Sample_name,
  Primary_site,
  Primary_histology,
  Sample_source,
```

(continues on next page)

(continued from previous page)

```
    ARRAY_AGG(DISTINCT Gene_name) AS geneArray
  FROM
    cosmicSample
  GROUP BY
    Sample_name,
    Primary_site,
    Primary_histology,
    Sample_source ),
  --
  -- Next we can perform our set operations on the arrays.
  --
  setOpsTable AS (
  SELECT
    a.sample_barcode_tumor AS tcgaSample,
    b.Sample_name AS cosmicSample,
    b.Primary_site,
    b.Primary_histology,
    b.Sample_source,
    ARRAY_LENGTH(a.geneArray) AS length1,
    ARRAY_LENGTH(b.geneArray) AS length2,
    (SELECT COUNT(1) FROM UNNEST(a.geneArray) AS ga JOIN UNNEST(b.geneArray) AS gb ON ga␣
↪= gb) AS gene_intersection,
    (SELECT COUNT(DISTINCT gx) FROM UNNEST(ARRAY_CONCAT(a.geneArray,b.geneArray)) AS gx)␣
↪AS gene_union
  FROM
    tcgaSampleArray AS a
  JOIN
    cosmicSampleArray AS b
  ON
    a.sample_barcode_tumor < b.Sample_name )
  --
  -- And build our final results.
  --
SELECT
  tcgaSample,
  length1 AS geneCount1,
  cosmicSample,
  Primary_site,
  Primary_histology,
  Sample_source,
  length2 AS geneCount2,
  gene_intersection AS intersection,
  gene_union,
  (gene_intersection / gene_union) AS jaccard_index
FROM
  setOpsTable
WHERE
  (gene_intersection / gene_union) > 0.00
  AND gene_intersection > 5
  AND gene_union > 5
ORDER BY
  jaccard_index DESC
```

Recall that the TCGA-CA-6718-01A sample is from the COAD (colon adenocarcinoma) TCGA project.

Row	tcgaSample	geneCount1	cosmicSample	Primary_site	Primary_histology	geneCount2	intersection	gene_union	jaccard_index
1	TCGA-CA-6718-01A	1560	sysucc-311T	large_intestine	carcinoma	2942	307	4195	0.0731823599523242
2	TCGA-CA-6718-01A	1560	YUKLAB	skin	malignant_melanoma	2175	240	3495	0.06866952789699571
3	TCGA-CA-6718-01A	1560	YUKAT	skin	malignant_melanoma	3706	333	4933	0.06750456111899453
4	TCGA-CA-6718-01A	1560	YUWAND	skin	malignant_melanoma	1069	140	2489	0.0562474889513861
5	TCGA-CA-6718-01A	1560	YURAY	skin	malignant_melanoma	1335	149	2746	0.054260742898761836
6	TCGA-CA-6718-01A	1560	YULAN	skin	malignant_melanoma	1342	149	2753	0.054122775154377044
7	TCGA-CA-6718-01A	1560	cSCCP7	skin	carcinoma	647	104	2103	0.04945316214931051

Cool – the top COSMIC (non-TCGA) hit is from a very similar tumor type! The other close matches are all from melanoma, a cancer with a very high mutation rate which might result in "noisy" associations at this level.

Since the two mutation tables that we are comparing here (TCGA MC3 and COSMIC) use different scores for assessing the implications of each somatic mutation, another idea would be to integrate the Tute Genomics hg19 annotation table so that a single score can be used in assessing the intersection of mutated genes between two cancer samples.

Thanks for joining us this month!

March, 2017

This month we're going to compute a pairwise distance matrix and visualize it using a heatmap in R. Many methods, such as clustering, depend on having a distance matrix, and although I would not recommend using BigQuery to download large tables, this works well for smaller feature sets (10s-100s).

In this example, we will be selecting primary tumor samples from both BRCA and STAD cohorts, along with a list of the top 50 most variable miRNAs. Then we'll compute a pairwise distance metric on samples. The distance will be based on Spearman's correlation.

As usual, we are going to be using standard SQL, so make sure to select that option.

```
WITH
  --
  -- *sample_lists*
  -- First, start by defining the list of BRCA TP samples.
  -- (TP == tumor, primary)
  --
  brca_sample_list AS (
  SELECT
    SampleBarcode
  FROM
    `isb-cgc.tcga_201607_beta.Biospecimen_data`
  WHERE
    SampleTypeLetterCode='TP'
    AND Study='BRCA'
  LIMIT
    50),
  --
```

(continues on next page)

(continued from previous page)

```
-- Then let's create a list of STAD samples.
--
stad_sample_list AS (
SELECT
  SampleBarcode
FROM
  `isb-cgc.tcga_201607_beta.Biospecimen_data`
WHERE
  SampleTypeLetterCode='TP'
  AND Study='STAD'
LIMIT
  50),
--
-- Now, we are going to merge the two sample tables using a UNION ALL.
--
sample_list AS (
select * from stad_sample_list
UNION ALL
select * from brca_sample_list
),
--
-- *miRNA_list*
-- Next, we define the miRNAs of interest. We order the miRNAs by standard
-- deviation, then take the top 50. Notice we select value from the
-- subset defined above.
--
miRNA_list AS (
SELECT
  mirna_accession,
  STDDEV(normalized_count) AS sigma_count
FROM
  `isb-cgc.tcga_201607_beta.miRNA_Expression`
WHERE
  SampleBarcode IN (
  SELECT
    SampleBarcode
  FROM
    sample_list )
GROUP BY
  mirna_accession
ORDER BY
  sigma_count DESC
LIMIT
  50 ),
--
-- *miRNA_data*
-- Now that we have the sample_list and the mirna_list, we can select our
-- data of interest from the larger miRNA_Expression table.
--
miRNA_data AS (
SELECT
  SampleBarcode,
```

(continues on next page)

(continued from previous page)

```
    Study,
    mirna_id,
    mirna_accession,
    LOG10(normalized_count+1) AS log10_count
  FROM
    `isb-cgc.tcga_201607_beta.miRNA_Expression`
  WHERE
    SampleBarcode IN (
    SELECT
      SampleBarcode
    FROM
      sample_list )
    AND mirna_accession IN (
    SELECT
      mirna_accession
    FROM
      miRNA_list ) ),
  --
  -- *pairs*
  -- Now, we JOIN the miRNA_data matrix with *itself*, creating all possible pairs of␣
↪samples
  -- (excluding self-comparisons which are unnecessary) combined with a dense-ranking of
  -- the miRNA expression values. By computing the Pearson correlation on ranks, we
  -- end up with Spearman's correlation!
  --
  pairs AS (
  SELECT
    lhs.mirna_id,
    lhs.mirna_accession,
    lhs.SampleBarcode AS SampleA,
    rhs.SampleBarcode AS SampleB,
    lhs.Study AS StudyA,
    rhs.Study as StudyB,
    DENSE_RANK() OVER (PARTITION BY lhs.mirna_accession ORDER BY lhs.log10_count ASC) AS␣
↪ExpA,
    DENSE_RANK() OVER (PARTITION BY rhs.mirna_accession ORDER BY rhs.log10_count ASC) AS␣
↪ExpB
  FROM
    miRNA_data AS lhs
  JOIN
    miRNA_data AS rhs
  ON
    lhs.mirna_accession=rhs.mirna_accession
    AND lhs.SampleBarcode < rhs.SampleBarcode )
  --
  -- **Finally**, we compute the pairwise distance between each pair of samples.
  --
SELECT
  SampleA,
  SampleB,
  StudyA,
  StudyB,
```

(continues on next page)

(continued from previous page)

```
  COUNT(mirna_accession) AS numObs,
  (1.-CORR(ExpA, ExpB))  AS sampleDistance
FROM
  pairs
GROUP BY
  SampleA,StudyA,
  SampleB,StudyB
ORDER BY
  sampleDistance ASC
```

Now, let's see that distance matrix in R!

```
library(bigrquery) # make sure it's a recent version with the useLegacySql param!#

q <- "The Query From Above"

corrs <- query_exec(q, project="YOUR PROJECT ID", useLegacySql=F)

# Use bigrquery to get the results or export the results to cloud storage and
# download them like so.
# gcs_url <- "gs://MY-BUCKET/MY-FILE.csv"
# corrs <- read.csv(pipe(sprintf("gsutil cat %s", gcs_url)))

library(dplyr)
library(ggplot2)
library(pheatmap)

mat <- xtabs(sampleDistance~SampleA+SampleB, data=corrs)
# or tidyr::spread(data=corrs, key=SampleA, value=sampleDistance, fill=0)

dim(mat) # 99 x 99

# Make the matrix symmetric.
mat2 <- mat + t(mat)

# Let's make an annotation matrix for cancer type
studyMat <- unique(corrs[,c("StudyA", "SampleA")])
studyMat$color <- ifelse(studyMat$Study == "BRCA", "blue", "red")
rownames(studyMat) <- studyMat$SampleA

# We can show the distances between samples as a dendrogram
# install.packages("dendextend")
library(dendextend)
hc <- hclust(as.dist(mat2), method="ward.D2")
dend <- as.dendrogram(hc)
labels_colors(dend) <- studyMat[labels(dend), "color"]
dend <- set(dend, "labels_cex", 0.5)

## Fig1 ##
plot(dend, main="BRCA in clue and STAD in red")

# If we want to make two groups, then we cut the dendrogram
```

(continues on next page)

(continued from previous page)

```
# leaving two branches.
cas <- cutree(tree=hc, k=2)

# Then we can use our cluster labels to annotate the heatmap.
annotMat <- data.frame(cluster=cas)
annotMat$SampleID <- names(cas)
rownames(annotMat) <- names(cas)
annotMat <- merge(x=annotMat, by.x="SampleID", y=studyMat, by.y="SampleA")
rownames(annotMat) <- annotMat$SampleID

# And we can plot cluster assignments on a heatmap
# to see how hclust-default and pheatmap-defaults compare.

## Fig2 ##
pheatmap(mat2, fontsize=6, annotation=annotMat[,-1])

# Or we can use the dendsort library (from pheatmap examples)
library(dendsort)
callback = function(hc, ...){dendsort(hc)}

## Fig3 ##
pheatmap(mat2, fontsize=6, annotation=annotMat[,-1], clustering_callback = callback)

# Modify ordering of the clusters using clustering callback option
callback = function(hc, mat){
    sv = svd(t(mat))$v[,1]
    dend = reorder(as.dendrogram(hc), wts = sv)
    as.hclust(dend)
}
## Fig4 ##
pheatmap(mat2, clustering_callback = callback, annotation=annotMat[,-1],
    fontsize_row=4, fontsize_col=4, border_color=NA)
```

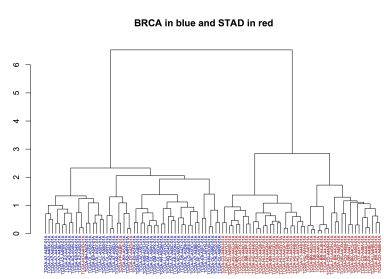


Fig. 2: Fig1. Dendrogram showing the sample-wise relationships based on miRNA expression.

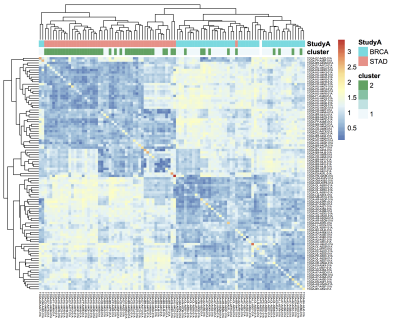


Fig. 3: Fig2. Heatmap of pairwise distances, using pheatmap default clustering.

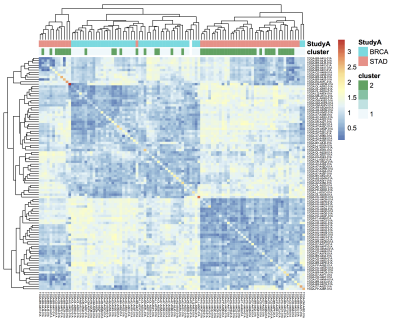


Fig. 4: Fig3. Heatmap of pairwise distances, using the dendsort library.

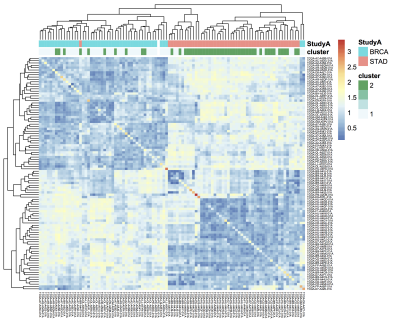


Fig. 5: Fig4. Ordering the samples after singular value decomposition.

February, 2017

This month, we explore user defined functions or UDFs. BigQuery allows you to define custom functions, things that you can't easily do in standard SQL, using JavaScript. These functions are defined as part of the SQL and then called within the query.

UDFs take a set of parameters, and return a value. They are strongly typed functions, which means that we need to define the types of inputs and outputs. For example, we might have FLOAT64 and BOOL input types and return a STRING. See the official Google documentation for the complete list of available types. (Note in particular that there is no INT64 type, so you will need to use either FLOAT64 or STRING when working with integers, depending on your needs.)

In our first example, we'll define two new functions. The first classifies a sample as having a higher expression value than a given input level. And second, a function that glues three strings together. Then, in the SQL query we call both functions. These initial queries will be starting points for a more complicated example below.

These queries use *Standard* SQL, so if you're in the web interface, remember to open the options panel and uncheck the 'Use Legacy SQL' button.

```
-- this next line tells BigQuery that a UDF is coming
CREATE TEMPORARY FUNCTION
  -- followed by the function name and parameter names/types:
  BiggerThan (x FLOAT64, y FLOAT64)
  -- and then the return type
  RETURNS BOOL
  -- and the language
  LANGUAGE js AS """
    // careful to use this delimiter for comments inside the function
    return (x > y);
  """;

-- now let's create another function that takes 3 input strings
-- combines them, using underscores and returns a single string:
CREATE TEMPORARY FUNCTION
  Combiner (x STRING, y STRING, z STRING)
  RETURNS STRING
  LANGUAGE js AS """
    return (x + "_" + y + "_" + z);
  """;

--
-- Now that we've defined our two UDFs, we can use them.
-- But first we're going to create an intermediate table
-- with the expression of ESR1 in the BRCA samples:
--
WITH
  gene1 AS (
  SELECT
    AliquotBarcode AS barcode1,
    Study AS study1,
    SampleTypeLetterCode AS code1,
    HGNC_gene_symbol AS gene_id1,
    AVG(LOG(normalized_count+1, 2)) AS count1
  FROM
```

(continues on next page)

(continued from previous page)

```
    `isb-cgc.tcga_201607_beta.mRNA_UNC_RSEM`
  WHERE
    Study = 'BRCA'
    AND SampleTypeLetterCode = 'TP'
    AND HGNC_gene_symbol = 'ESR1'
    AND normalized_count >= 0
  GROUP BY
    AliquotBarcode,
    gene_id1,
    study1,
    code1)

--
--
-- Now we can call our functions,
-- processing the subtable.
--
SELECT
  Combiner(barcode1, study1, code1) AS cString,
  BiggerThan(count1, 5.1) AS overExp
FROM
  gene1
```

OK, so that was just warm-up, and obviously what was being done with the UDFs could have been done in SQL as well. But now we're going to do something a bit more complicated(!), and estimate cluster assignments using a K-means algorithm (wikipedia), implemented in JavaScript, as a UDF!

We're going to try to cluster each BRCA sample based on the expression of two genes: ESR1 and EGFR. This type of clustering is implemented as an iterative process that starts with two random cluster centers. In each iteration, each sample is labeled according to the nearest cluster-center, and then we recompute the locations of the cluster centers.

```
CREATE TEMPORARY FUNCTION

  -- In this function, we're going to be working on arrays of values.
  -- we're also going to define a set of functions 'inside' the kMeans.

  -- *heavily borrowing from https://github.com/NathanEpstein/clusters* --

  kMeans(x ARRAY<FLOAT64>,  -- ESR1 gene expression
         y ARRAY<FLOAT64>,  -- EGFR gene expression
         iterations FLOAT64,  -- the number of iterations
         k FLOAT64)           -- the number of clusters

  RETURNS ARRAY<FLOAT64>  -- returns the cluster assignments

  LANGUAGE js AS """
  'use strict'


  function sumOfSquareDiffs(oneVector, anotherVector) {
    // the sum of squares error //
    var squareDiffs = oneVector.map(function(component, i) {
      return Math.pow(component - anotherVector[i], 2);
```

(continues on next page)

(continued from previous page)

```
    });
    return squareDiffs.reduce(function(a, b) { return a + b }, 0);
  };

function mindex(array) {
  // returns the index to the minimum value in the array
  var min = array.reduce(function(a, b) {
    return Math.min(a, b);
  });
  return array.indexOf(min);
};

function sumVectors(a, b) {
  // The map function gets used frequently in JavaScript
  return a.map(function(val, i) { return val + b[i] });
};

function averageLocation(points) {
  // Take all the points assigned to a cluster
  // and find the averge center point.
  // This gets used to update the cluster centroids.
  var zeroVector = points[0].location.map(function() { return 0 });
  var locations = points.map(function(point) { return point.location });
  var vectorSum = locations.reduce(function(a, b) { return sumVectors(a, b) },␣
↪zeroVector);
  return vectorSum.map(function(val) { return val / points.length });
};

function Point(location) {
  // A point object, each sample is represented as a point //
  var self = this;
  this.location = location;
  this.label = 1;
  this.updateLabel = function(centroids) {
    var distancesSquared = centroids.map(function(centroid) {
      return sumOfSquareDiffs(self.location, centroid.location);
    });
    self.label = mindex(distancesSquared);
  };
};


function Centroid(initialLocation, label) {
  // The cluster centroids //
  var self = this;
  this.location = initialLocation;
  this.label = label;
  this.updateLocation = function(points) {
    var pointsWithThisCentroid = points.filter(function(point) { return point.label ==␣
↪self.label });
    if (pointsWithThisCentroid.length > 0) {
      self.location = averageLocation(pointsWithThisCentroid);
```

(continues on next page)

(continued from previous page)

```
    }
  };
};


var data = [];

// Our data list is list of lists. The small list being each (x,y) point
for (var i = 0; i < x.length; i++) {
  data.push([x[i],y[i]])
}

// initialize point objects with data
var points = data.map(function(vector) { return new Point(vector) });


// intialize centroids
var centroids = [];
for (var i = 0; i < k; i++) {
  centroids.push(new Centroid(points[i % points.length].location, i));
};


// update labels and centroid locations until convergence
for (var iter = 0; iter < iterations; iter++) {
  points.forEach(function(point) { point.updateLabel(centroids) });
  centroids.forEach(function(centroid) { centroid.updateLocation(points) });
};

// return the cluster labels.
var labels = []
for (var i = 0; i < points.length; i++) {
  labels.push(points[i].label)
}

return labels;

""";
--
-- *** In this query, we create two subtables, one for each gene of
--     interest, then create a set of arrays in joining the two tables.
--     We call the UDF using the arrays, and get a result back
--     made of arrays.
--
--     Due to a technical issue we save the table of arrays to
--     to a personal dataset, then unpack it. ***
--
WITH
  -- gene1, the first subtable
  --
  gene1 AS (
  SELECT
```

(continues on next page)

(continued from previous page)

```
  ROW_NUMBER() OVER() row_number,
  AliquotBarcode AS barcode1,
  HGNC_gene_symbol AS gene_id1,
  AVG(LOG(normalized_count+1, 2)) AS count1
FROM
  `isb-cgc.tcga_201607_beta.mRNA_UNC_RSEM`
WHERE
  Study = 'BRCA'
  AND SampleTypeLetterCode = 'TP'
  AND HGNC_gene_symbol = 'ESR1'
  AND normalized_count >= 0
GROUP BY
  AliquotBarcode,
  gene_id1),
--
-- gene2, the second subtable
--
gene2 AS (
SELECT
  AliquotBarcode AS barcode2,
  HGNC_gene_symbol AS gene_id2,
  AVG(LOG(normalized_count+1, 2)) AS count2
FROM
  `isb-cgc.tcga_201607_beta.mRNA_UNC_RSEM`
WHERE
  Study = 'BRCA'
  AND SampleTypeLetterCode = 'TP'
  AND HGNC_gene_symbol = 'EGFR'
  AND normalized_count >= 0
GROUP BY
  AliquotBarcode,
  HGNC_gene_symbol),
--
-- Then we create a table of arrays
-- and join the two gene tables.
-- ** We need to make sure all the arrays are constructed using the same index. **
--
arrayTable AS (
SELECT
  ARRAY_AGG(m1.row_number ORDER BY m1.barcode1) AS arrayn,
  ARRAY_AGG(m1.barcode1 ORDER BY m1.barcode1) AS barcode,
  ARRAY_AGG(count1 ORDER BY m1.barcode1) AS esr1,
  ARRAY_AGG(count2 ORDER BY m1.barcode1) AS egfr
FROM
  gene1 AS m1
JOIN
  gene2 AS m2
ON
  m1.barcode1 = m2.barcode2 )
--
-- Now we call the k-means UDF.
--
```

(continues on next page)

(continued from previous page)

```
SELECT
  arrayn,
  barcode,
  esr1,
  egfr,
  kMeans(esr1, egfr, 200.0, 2.0) AS cluster
FROM
  arrayTable
  --
  -- save the resulting table to a personal dataset
```

We need to save the above results to an intermediate table. You will need to have a dataset that you have write-access to in BigQuery to do this. For your convenience, we've saved the query above as a public gist and also created a bitly link to it. Rather than pasting the entire script into the BigQuery web UI, you can us the **bq** command line (part of the cloud SDK) and run the query and automatically save the outputs as shown below.

```
#!/bin/bash

qFile="kMeans_in_BQ.sql"
## get the lengthy query from the bitly link, and rename
wget -O $qFile http://bit.ly/2mn1R5D

## before you can run this you will need to modify
## the destination table to be in a project and dataset
## that you have write-access to,
## eg:  dTable="isb-cgc:scratch_dataset.kMeans_out"
dTable="YOUR_PROJECT:DATASET_NAME.TMP_TABLE"

## run the query using the 'bq' command line tool
## not all of the options are strictly necessary -- with
## the exception of "nouse_legacy_sql"
bq query --allow_large_results \
         --destination_table=$dTable \
         --replace \
         --nouse_legacy_sql \
         --nodry_run \
         "$(cat $qFile)" > /dev/null
```

The results of the kMeans query is *one* row of arrays. It's a little tricky to unpack the arrays into rows, which is what the next query does. (Again you'll need to edit it to select from the intermediate table you created in the previous step. Remember that in Standard SQL, the delimiter between the project name and the dataset name is just a '.' whereas the bq command-line requres a ':' as a separator.)

```
WITH
  resultTable AS (
  SELECT
    *
  FROM
    `YOUR_PROJECT.DATASET_NAME.TMP_TABLE` )
SELECT
  index row_idx,
  barcode[OFFSET(index_offset)] AS AliquotBarcode,
```

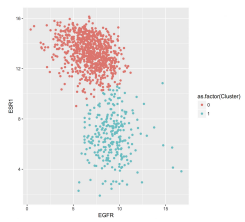
(continues on next page)

(continued from previous page)

```
  esr1[OFFSET(index_offset)] AS ESR1,
  egfr[OFFSET(index_offset)] AS EGFR,
  cluster[OFFSET(index_offset)] AS Cluster
FROM
  resultTable,
  UNNEST(resultTable.arrayn) AS index
WITH
OFFSET
  index_offset
ORDER BY
  index
```

Finall let's visualize the resulting clusters! Save the cluster assignments to a csv file, and read it into R.

```
library(ggplot2)
res0 <- read.table("results-from-the-k-means.csv", sep=",", header=T, stringsAsFactors=F)
qplot(data=res0, x=EGFR, y=ESR1, col=as.factor(Cluster))
```



January, 2017

This month we'll be comparing Standard SQL and Legacy SQL. It's possible to write queries using either form, but as we'll see, using standard SQL can be easier to write and improves readability.

In order to 'activate' standard SQL in the web browser, just under the 'New Query' text window, click the 'Show Options' button, and towards the bottom of the options you'll find the 'Use Legacy SQL' check box – uncheck that, and then you can 'Hide Options' to collapse the options away again.

To use R and bigrquery to execute standard SQL, you'll need to make sure you're using the most up-to-date version of the R package. I would recommend installing it from the github page using devtools. Please see bigrquery for more information on installation. The important bit: there's now support for sending a parameter called 'useLegacySql' in the REST calls.

Our task this month will be to compute correlations between copy number variants and gene expression, over all genes, using only BRCA samples. The copy number data is expressed in a series of segments, each with a chromosome, start-point, end-point, and value indicating whether a duplication or deletion event (or neither) has taken place.

Note that the mean copy-number values are not discrete because these data represent heterogeneous mixtures of cells – only a fraction of the cells might include an amplification or a deletion, so the 'mean copy-number' value represents the effect of the discrete amplifications or deletions of a specific genomic segment, averaged over the heterogenous population of cells.

One might hypothesize that a copy number *amplified* gene would have higher expression levels than when *not* amplified. However, our gene expression data has no location information, making it necessary to join the genomic locations from

an appropriate reference. The resulting annotated expression table can then be joined to the copy number segments. But computing the overlap of DNA segments and genes locations can get tricky! Below we show two different ways of accomplishing the task.

Data Tables

You can get familiar with the data sources by opening the BigQuery web interface and taking a preview of the tables.

- isb-cgc.tcga_cohorts.BRCA … Curated cohort table for TCGA BRCA study: 1087 unique patients and 2236 unique samples.
- isb-cgc.genome_reference.GENCODE_v19 … This table is based on release 19 of the GENCODE reference gene set. Note that these annotations are based on the hg19/GRCh37 reference genome.
- isb-cgc.tcga_201607_beta.mRNA_UNC_HiSeq_RSEM … This table contains all mRNA expression data produced by the UNC-LCCC (Lineberger Comprehensive Cancer Center) using the Illumina HiSeq platform and processed through their RNASeqV2 / RSEM pipeline.
- isb-cgc.tcga_201607_beta.Copy_Number_segments … This table contains one row for each copy-number segment identified for each TCGA aliquot. Affymetrix SNP6 data is used in making the calls.

Legacy SQL

```
# This query makes use of a legacy UDF or 'user defined function'.
# To define UDFs in R, we need to define it 'inline'.
# For another example of inline definitions, see:
# https://github.com/googlegenomics/bigquery-examples/blob/master/pgp/sql/schema-
↪comparisons/missingness-udf.sql

# Big legacy SQL queries grow like onions, they start in the center,
# and grow in layers, until the outer-most select statement returns the result.
# And like onions, they will make you cry.

SELECT
  # Here's the final select statement, computing Pearson's correlation
  # on the avgCNsegMean, the copy number mean for a particular gene
  # and avglogExp, the average expression for the same gene.
  gene,
  chr,
  CORR(avgCNsegMean,avglogExp) AS corr,
  COUNT(*) AS n
FROM (

  SELECT
    # This is the select statement on the joined CN and expr tables,
    # where averages are computed on copy number and expression.
    annotCN.gene AS gene,
    annotCN.chr AS chr,
    annotCN.SampleBarcode AS SampleBarcode,
    AVG(annotCN.CNsegMean) AS avgCNsegMean,
    AVG(exp.logExp) AS avgLogExp
  FROM (
```

(continues on next page)

(continued from previous page)

```
    SELECT
      # This is the select statement that annotates the CN segments via binning.
      # To annotate the segments, the CN segment start and end positions are binned,
      # as well as the gene reference information.
      # The bins provide a sort of grid that can be used for aligning the segments
      # to gene locations.
      geneInfo.gene AS gene,
      geneInfo.chr AS chr,
      geneInfo.region_start AS gene_start,
      geneInfo.region_end AS gene_end,
      geneInfo.bin AS bin,
      cnInfo.SampleBarcode AS SampleBarcode,
      cnInfo.Segment_Mean AS CNsegMean,
      cnInfo.region_start AS cn_start,
      cnInfo.region_end AS cn_end
    FROM (
      SELECT
        label AS gene,
        chr,
        region_start,
        region_end,
        bin
      FROM js (  # This User-defined function bins the genome making the join possible.

          (SELECT
            gene_name AS label,
            FLOAT(start) AS value,
            LTRIM(seq_name,\"chr\") AS chr,
            start AS region_start,
            END AS region_end
          FROM
            [isb-cgc:genome_reference.GENCODE_v19]
          WHERE
            feature=\"gene\"
            AND gene_status=\"KNOWN\"
            AND source=\"HAVANA\"),

            label, value, chr, region_start, region_end,

            \"[{'name': 'label', 'type': 'string'},   // Output schema
             {'name': 'value', 'type': 'float'},
             {'name': 'chr',   'type': 'string'},
             {'name': 'region_start', 'type': 'integer'},
             {'name': 'region_end',   'type': 'integer'},
             {'name': 'bin',   'type': 'integer'}]\",

             \"function binIntervals(row, emit) {
               // This is JavaScript ... here we use '//' for comments
               // Legacy UDFs take a single row as input.
               // and return a row.. can be a different number of columns.
               var binSize = 10000;  // Make sure this matches the value in the SQL (if␣
↪necessary)
```

(continues on next page)

(continued from previous page)

```
            var startBin = Math.floor(row.region_start / binSize);
            var endBin = Math.floor(row.region_end / binSize);

            // Since an interval can span multiple bins, emit
            // a record for each bin it spans.
            for(var bin = startBin; bin <= endBin; bin++) {
              emit({label: row.label,
                    value: row.value,
                    chr: row.chr,
                    region_start: row.region_start,
                    region_end: row.region_end,
                    bin: bin,
                   });
            }
        }\")) AS geneInfo

JOIN EACH (
  # This is the join between the binned CNs, and the binned gene reference. #

  SELECT
    # This is the select statement that bins the gene reference information
    label AS SampleBarcode,
    value AS Segment_Mean,
    chr,
    region_start,
    region_end,
    bin
  FROM ( js (
      (SELECT
        SampleBarcode AS label,
        Segment_Mean AS value,
        Chromosome AS chr,
        start AS region_start,
        END AS region_end
      FROM
        [isb-cgc:tcga_201607_beta.Copy_Number_segments]
      WHERE
        SampleBarcode IN (
        SELECT
          SampleBarcode
        FROM
          [isb-cgc:tcga_cohorts.BRCA] ) ),

      label,value,chr,region_start,region_end,

      \"[{'name': 'label', 'type': 'string'},
        {'name': 'value', 'type': 'float'},
        {'name': 'chr',   'type': 'string'},
        {'name': 'region_start', 'type': 'integer'},
        {'name': 'region_end',   'type': 'integer'},
        {'name': 'bin',   'type': 'integer'}]\",
```

(continues on next page)

(continued from previous page)

```
          \"function binIntervals(row, emit) {
            // This is JavaScript ... here we use '//' for comments
            // Legacy UDFs take a single row as input.
            var binSize = 10000;  // Make sure this matches the value in the SQL (if␣
↪necessary)
            var startBin = Math.floor(row.region_start / binSize);
            var endBin = Math.floor(row.region_end / binSize);
            // Since an interval can span multiple bins, emit
            // a record for each bin it spans.
            for(var bin = startBin; bin <= endBin; bin++) {
              emit({label: row.label,
                    value: row.value,
                    chr: row.chr,
                    region_start: row.region_start,
                    region_end: row.region_end,
                    bin: bin,
                   });
            }
          }\"
        ) ) ) AS cnInfo
    ON
      ( geneInfo.chr = cnInfo.chr )
      AND ( geneInfo.bin = cnInfo.bin ) ) AS annotCN

  JOIN EACH (
    # Here's the join between annotated copy number table and the gene expression table.

    SELECT
      # Here we get the gene expression data, and barcodes.
      # We join on the SampleBarcodes in each table.
      SampleBarcode,
      HGNC_gene_symbol,
      LOG2(normalized_count+1) AS logExp
    FROM
      [isb-cgc:tcga_201607_beta.mRNA_UNC_HiSeq_RSEM]
    WHERE
      SampleBarcode IN (
      SELECT
        SampleBarcode
      FROM
        [isb-cgc:tcga_cohorts.BRCA] ) ) AS exp
  ON
    ( exp.HGNC_gene_symbol = annotCN.gene )
    AND ( exp.SampleBarcode = annotCN.SampleBarcode )
  GROUP BY
    gene,
    chr,
    SampleBarcode )
GROUP BY
  gene,
  chr
HAVING
```

(continues on next page)

(continued from previous page)

```
  corr IS NOT NULL
ORDER BY
  corr DESC
```

Standard SQL

```
# In standard SQL, we define a list of tables, that can build
# off earlier definitions, so it's a little more linear and modular.

WITH
# This says: "we're going to define a list of tables WITH which we will perform␣
↪subsequent SELECTs..."

  geneInfo AS (
    # First table: the gene reference information
    SELECT
      gene_name AS gene,
      LTRIM(seq_name,'chr') AS chr,
      `start` as region_start,
      `end`   as region_end
    FROM
      `isb-cgc.genome_reference.GENCODE_v19`
    WHERE
      feature='gene'
      AND gene_status='KNOWN'
      AND source = 'HAVANA'),

cnInfo AS(
  # Second: the copy number data, but only for the BRCA samples (note the sub-query).
  SELECT
    SampleBarcode,
    Segment_Mean,
    Chromosome AS chr,
    `start` AS region_start,
    `end`   AS region_end
  FROM
    `isb-cgc.tcga_201607_beta.Copy_Number_segments`
  WHERE
    SampleBarcode IN (
    SELECT
      SampleBarcode
    FROM
      `isb-cgc.tcga_cohorts.BRCA` )),

gexp AS (
  # Third: we get the gene expression data, again only for the BRCA samples
  # included is a LOG() transform as well as an AVG() aggregation function
  # which will only be relevant if there are multiple expression values for
  # a single (gene,sample) pair.
  SELECT
```

(continues on next page)

(continued from previous page)

```
    SampleBarcode,
    HGNC_gene_symbol,
    AVG(LOG(normalized_count+1,2)) AS logExp
  FROM
    `isb-cgc.tcga_201607_beta.mRNA_UNC_HiSeq_RSEM`
  WHERE
    SampleBarcode IN (
    SELECT
      SampleBarcode
    FROM
      `isb-cgc.tcga_cohorts.BRCA` )
  GROUP BY
    SampleBarcode,
    HGNC_gene_symbol),

cnAnnot AS (
  # Now, we start to re-use previously defined tables.  Here, we annotate
  # the copy-number segments by JOINing on matching chromosomes and
  # looking for overlapping regions between the copy-number segments and
  # the gene regions previously obtained from the GENCODE_v19 table.
  SELECT
    geneInfo.gene AS gene,
    geneInfo.chr AS chr,
    geneInfo.region_start AS gene_start,
    geneInfo.region_end AS gene_end,
    cnInfo.SampleBarcode AS SampleBarcode,
    AVG(cnInfo.Segment_Mean) AS Avg_CNsegMean
  FROM
  cnInfo JOIN geneInfo
  ON
    (geneInfo.chr = cnInfo.chr)
  WHERE
  (cnInfo.region_start BETWEEN geneInfo.region_start AND geneInfo.region_end) OR
  (cnInfo.region_end   BETWEEN geneInfo.region_start AND geneInfo.region_end) OR
  (cnInfo.region_start < geneInfo.region_start AND cnInfo.region_end > geneInfo.region_
↪end)
  GROUP BY
    gene,
    chr,
    gene_start,
    gene_end,
    SampleBarcode
),

bigJoin AS (
  # This is essentially the final step: in this last table definition, we make
  # a big join between the annotated copy-number table with the gene-expression
  # table and use the built-in CORR() function to compute a Pearson correlation.
  SELECT
    cnAnnot.gene AS gene,
    cnAnnot.chr AS chr,
    CORR(cnAnnot.Avg_CNsegMean,gexp.logExp) AS corr_cn_gexp,
```

(continues on next page)

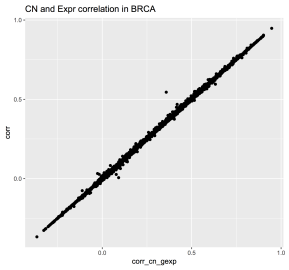
(continued from previous page)

```
    count(*) as n
  FROM
    cnAnnot join gexp
  ON
    ( gexp.HGNC_gene_symbol = cnAnnot.gene )
    AND ( gexp.SampleBarcode = cnAnnot.SampleBarcode )
  GROUP BY
    gene,
    chr
)

# Finally, let's pull down all the rows!
select *
from bigJoin
```

R script

```
# Here, we're going to execute the two above queries, and see how
# the correlations compare.

library(bigrquery)
library(ggplot2)

my_project_id <- "xyz"

q_legacy <- " ... first query above"

q_std <- " ... second query from above ..."

legacy_res <- query_exec(q_legacy, project=my_project_id, useLegacySql=T)

std_res <- query_exec(q_std, project=my_project_id, useLegacySql=F)

res0 <- merge(legacy_res, std_res, by="gene")

dim(std_res)
#[1] 18447     4

dim(legacy_res)
#[1] 18424     4

dim(res0)
#[1] 18424     7

qplot(data=res0, x=corr_cn_gexp, y=corr, main="CN and Expr correlation in BRCA",
      xlab="Standard SQL", ylab="Legacy SQL")
```

This plot shows the correlations found using the Legacy SQL solution (y-axis) compared to the correlations found using the Standard SQL solution (x-axis). Note that an exact match between the two methods was not expected because the



implementation is not identical. The "legacy" solution bins the copy-number segment values into uniform length genomic segments, while the "standard" solution takes a simpler approah.

December, 2016

The ISB-CGC team is starting to add the new hg38-based TCGA data available from the NCI-GDC Data Portal and one of the first obvious questions might be: how does the new hg38 expression data compare to the hg19 data?

Description

This is exactly the type of question that the ISB-CGC resources and the BigQuery engine were made to answer. In a single SQL query, we will compare two sets of gene-level expression estimates based on RNA-Seq data. The first set consists of the hg19-based RSEM normalized gene-level expression values previously available from the TCGA DCC and now available in an easy-to-use table in BigQuery (and also from the NCI-GDC Legacy Archive). The second set was produced by the NCI-GDC mRNA Analysis Pipeline which includes a STAR alignment to hg38, and gene expression quantification using HTSeq (with annotation based on GENCODE v22).

Rather than look at one gene at a time, it's easy (and fast!) to compute correlations for all genes simultaneously. Note that this is done in a *single* query. You do **not** want to *loop* over all of the genes, computing one correlation at a time because the *cost* of a BigQuery query depends primarily on the amount of data scanned during the query, and since the data for *all* genes across *all* TCGA samples are in a *single* table, if you were to loop over 10,000 genes running one query per gene, your costs would go *up* by a factor of 10,000! As we show below, BigQuery computes across all the genes at once (one of its benefits), and thus keeps the costs low.

In addition to using the two gene-expression data tables, our SQL query also uses the GENCODE_v22 table (one of many tables in the **isb-cgc.genome_reference** dataset) to map from the HGNC gene symbol (used in the older hg19 expression table) to the Ensembl gene identifier (used in the new hg38 expression table).

The query below performs both Pearson and Spearman correlations for each gene. The result is a table with 20,021 rows – one for each gene, with the Ensembl gene identifier, the gene symbol, the Pearson and Spearman correlation coefficients, and the difference between the two. The table has also been sorted by the Spearman coefficient, in descending order. This query executes in less than one minute and processes a total of 34 GB of data. This is a great example of how cloud computing can significantly enhance analytic capabilities beyond running large analytic bioinformatics pipelines quickly!

Back in June, Google announced full support for Standard SQL in BigQuery. The query below makes use of Standard SQL, so if you want to try running this query yourself by cutting-and-pasting it into the BigQuery web UI you'll need to go into the **Show Options** section and uncheck the "Use Legacy SQL" box. If you're used to using Legacy SQL, one small change you'll need to make right away is in how you refer to tables: rather than `[isb-cgc:genome_reference.GENCODE_v22]` for example, you will instead write `isb-cgc.genome_reference.GENCODE_v22` inside single-quotes.

As a concrete example of what these data look like, we created plots of the expression data for EGFR in R (see below for the SQL and R code).

BigQuery SQL

```
WITH
--
-- *GdcGene*
-- We start by extracting gene-expression data from the new NCI-GDC/hg38-based
-- table in the isb-cgc:hg38_data_previews dataset.  For each row, we
-- extract simply the SamplesSubmitterID (aka the TCGA sample barcode),
-- the Ensembl gene ID (eg ENSG00000182253), and the FPKM value.  The input
-- table has ~671M rows and many more fields, but we just need these 3.
GdcGene AS (
SELECT
  SamplesSubmitterID AS sampleID,
  Ensembl_gene_ID AS geneID,
  HTSeq__FPKM AS HTSeq_FPKM
FROM
  `isb-cgc.hg38_data_previews.TCGA_GeneExpressionQuantification` ),
--
-- *GeneRef*
-- Next, we're going to get the gene-id to gene-symbol mapping from the GENCODE
-- reference table because the NCI-GDC table reference above contains only the gene-id
-- while the expression data we want to compare that to contains gene symbols.
GeneRef AS (
SELECT
  gene_id,
  gene_name
FROM
  `isb-cgc.genome_reference.GENCODE_v22`
WHERE
  feature='gene' ),
--
-- *Hg38*
-- Now we'll join the two tables above to annotate the NCI-GDC expression data with gene-
↪symbols,
-- and we'll call it Hg38.  We're also going to create a ranking of the expression values
-- so that we can compute a Spearman correlation later on.
Hg38 AS (
SELECT
  GdcGene.sampleID,
  GdcGene.geneID,
  GeneRef.gene_name,
  GdcGene.HTSeq_FPKM,
  DENSE_RANK() OVER (PARTITION BY GdcGene.geneID ORDER BY GdcGene.HTSeq_FPKM ASC) AS␣
↪rankHTSeq
FROM
  GdcGene
JOIN
  GeneRef
ON
```

(continues on next page)

(continued from previous page)

```
  GdcGene.geneID = GeneRef.gene_id ),
--
-- *Hg19*
-- Now, we'll get the older hg19-based TCGA gene expression data that was generated
-- by UNC using RSEM.  This table has ~228M rows and we're just going to extract
-- the sample-barcode, the gene-symbol, the normalized-count, and the platform (since
-- this data ws produced on two different platforms and this might be relevant later).
-- As above, we will also create ranking of the expression values.
Hg19 AS (
SELECT
  SampleBarcode,
  HGNC_gene_symbol,
  normalized_count as RSEM_FPKM,
  DENSE_RANK() OVER (PARTITION BY HGNC_gene_symbol ORDER BY normalized_count ASC) AS␣
↪rankRSEM,
  Platform
FROM
  `isb-cgc.tcga_201607_beta.mRNA_UNC_RSEM`
WHERE
  HGNC_gene_symbol IS NOT NULL ),
--
-- *JoinAndCorr*
-- Finally, we join the two tables and compute correlations
JoinAndCorr AS (
SELECT
  hg38.geneID AS gene_id,
  hg38.gene_name AS gene_name,
  CORR(LOG10(hg38.HTSeq_FPKM+1),
    LOG10(hg19.RSEM_FPKM+1)) AS gexpPearsonCorr,
  CORR(hg38.rankHTSeq,
    hg19.rankRSEM) AS gexpSpearmanCorr
FROM
  Hg19
JOIN
  Hg38
ON
  hg38.sampleID=hg19.SampleBarcode
  AND hg38.gene_name=hg19.HGNC_gene_symbol
GROUP BY
  hg38.geneID,
  hg38.gene_name )
--
-- Lastly, we make one last select
-- to get a difference between Pearson and Spearman correlations.
SELECT
  gene_id,
  gene_name,
  gexpPearsonCorr,
  gexpSpearmanCorr,
  (gexpSpearmanCorr-gexpPearsonCorr) AS deltaCorr
FROM
  JoinAndCorr
```

(continues on next page)

(continued from previous page)

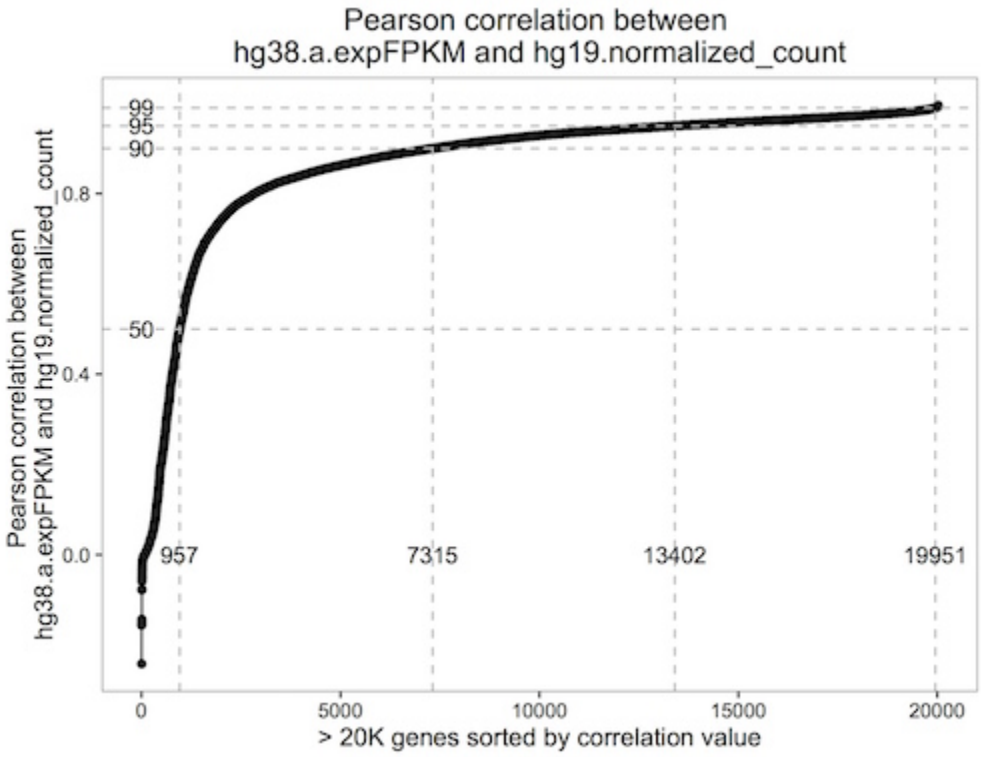
```
WHERE
  IS_NAN(gexpSpearmanCorr) = FALSE
ORDER BY
  gexpSpearmanCorr DESC
```

The results of any BigQuery query executed in the BigQuery web UI can easily be saved to a table in case you want to perform follow-up queries on the result. For example we might want to ask what the distribution of the correlation coefficients produced by the preceding query look like. We can ask BigQuery to compute the deciles on the saved results like this:

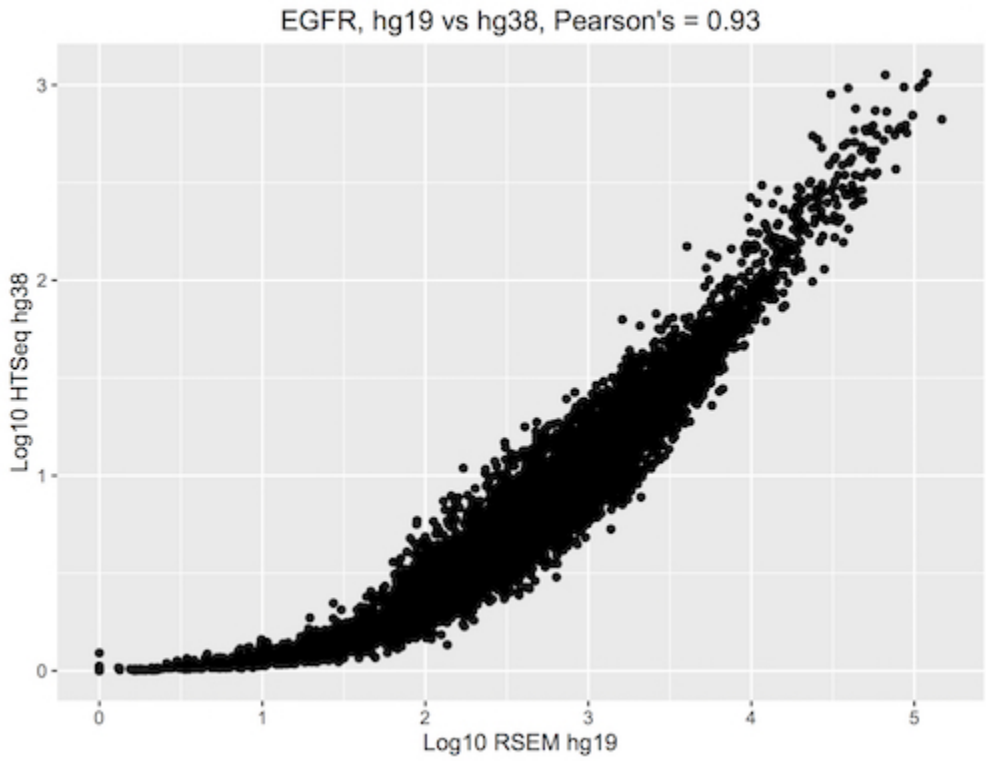
```
SELECT
  APPROX_QUANTILES ( gexpPearsonCorr, 10 ) AS PearsonQ,
  APPROX_QUANTILES ( gexpSpearmanCorr, 10 ) AS SpearmanQ,
  APPROX_QUANTILES ( deltaCorr, 10 ) AS deltaQ
FROM
  `<<insert your results table name here>>`
```

The result of the above query shows that 80% of genes have a Pearson correlation >= 0.84 and a Spearman correlation >= 0.88, and that 80% of the time the difference between these two correlations is between -0.012 and +0.098. The median Pearson correlation is nearly 0.93 and the median Spearman correlation is nearly 0.96.

Visualizations



This plot shows the cumulative distribution of the Pearson correlation between the hg19 RSEM expression and the hg38 HTSeq expression data. Each point represents one gene.



This plot shows the EGFR log10 expression, with the hg19 RSEM values on the x-axis and the hg38 HTSeq values on the y-axis. Note that overall, the difference between the RSEM and HTSeq methods may have a more significant impact on the expression values than the change in the genome build.

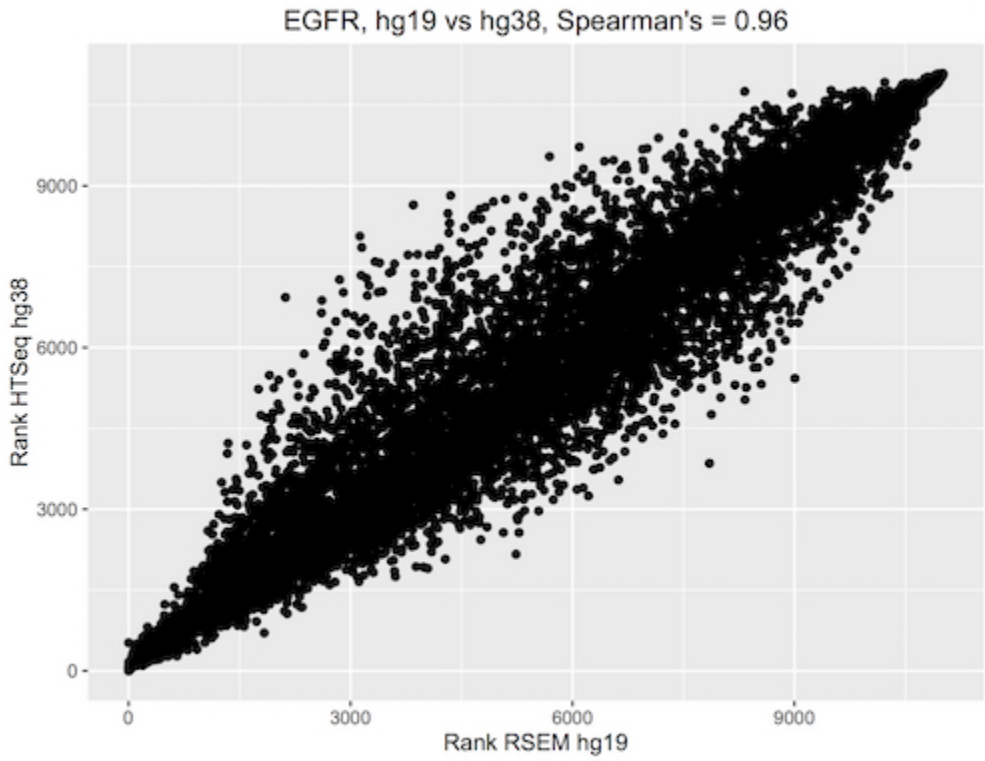
This plot shows the ranked EGFR expression, with the hg19 RSEM values on the x-axis and the hg38 HTSeq values on the y-axis.

R Script

Note that the latest version of the bigrquery package supports standard SQL, so make sure you're up to date.

```
library(devtools)
devtools::install_github("rstats-db/bigrquery")

library(bigrquery)
library(ggplot2)
library(stringr)

# saving the above query as a string variable named 'q'

res1 <- query_exec(q, project='isb-cgc-02-abcd', useLegacySql = FALSE)

n <-  dim(res1)[1]
ys <- c(0.5, 0.9, 0.95, 0.99)
ls <- sapply(1:4, function(i) sum(res1$gexpPearsonCorr < ys[i]))
```

(continues on next page)



(continued from previous page)

```
qplot(x=1:n, y=sort(res1$gexpPearsonCorr)) + geom_line() +
geom_hline(yintercept = ys, col='grey', lty=2) +
geom_vline(xintercept = ls, col='grey', lty=2) +
annotate(geom="text", label=ls[1], x=ls[1], y=0) +
annotate(geom="text", label=ls[2], x=ls[2], y=0) +
annotate(geom="text", label=ls[3], x=ls[3], y=0) +
annotate(geom="text", label=ls[4], x=ls[4], y=0) +
annotate(geom="text", label="50", y=ys[1], x=0) +
annotate(geom="text", label="90", y=ys[2], x=0) +
annotate(geom="text", label="95", y=ys[3], x=0) +
annotate(geom="text", label="99", y=ys[4], x=0) +
xlab("> 20K genes sorted by correlation value") +
ylab("Pearson correlation between \nhg38.a.expFPKM and hg19.normalized_count") +
ggtitle("Pearson correlation between \nhg38.a.expFPKM and hg19.normalized_count") +
theme_bw() +
theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
      panel.background = element_blank(), axis.line = element_line(colour = "black"))


# As an exercise, you could make the above plot with Spearman's correlations.


# Then let's take a look at one of our favorite genes, EGFR.

q <- "
  WITH
```

(continues on next page)

(continued from previous page)

```
  --
  Hg38 AS (
  SELECT
    SamplesSubmitterID AS sampleID,
    Ensembl_gene_ID AS geneID,
    DENSE_RANK() OVER (PARTITION BY Ensembl_gene_ID ORDER BY HTSeq__FPKM ASC) AS␣
↪rankHTSeq,
    HTSeq__FPKM AS HTseq_FPKM
  FROM
    `isb-cgc.hg38_data_previews.TCGA_GeneExpressionQuantification`
  WHERE
    Ensembl_gene_ID = 'ENSG00000146648'),
  --
  Hg19 AS (
  SELECT
    SampleBarcode,
    HGNC_gene_symbol,
    normalized_count as RSEM_FPKM,
    DENSE_RANK() OVER (PARTITION BY HGNC_gene_symbol ORDER BY normalized_count ASC) AS␣
↪rankRSEM,
    Platform
  FROM
    `isb-cgc.tcga_201607_beta.mRNA_UNC_RSEM`
  WHERE
    HGNC_gene_symbol = 'EGFR' )
  --
  -- *Join and Get Expr*
  SELECT
    hg38.geneID AS gene_id,
    hg19.HGNC_gene_symbol AS gene_name,
    LOG10(hg38.HTseq_FPKM+1) as Log10_hg38_HTSeq,
    LOG10(hg19.RSEM_FPKM+1) AS Log10_hg19_RSEM,
    rankRSEM,
    rankHTSeq
  FROM
    Hg19
  JOIN
    Hg38
  ON
    hg38.sampleID=hg19.SampleBarcode
  GROUP BY
    gene_id,
    gene_name,
    Log10_hg38_HTSeq,
    Log10_hg19_RSEM,
    rankRSEM,
    rankHTSeq"

result <- query_exec(q, project="isb-cgc-02-abcd", useLegacySql=F)

qplot(data=result, x=Log10_hg19_RSEM, y=Log10_hg38_HTSeq, main="EGFR, hg19 vs hg38,␣
↪Pearson's = 0.93", xlab="Log10 RSEM hg19", ylab="Log10 HTSeq hg38")
```

(continues on next page)

(continued from previous page)

```
qplot(data=result, x=rankRSEM, y=rankHTSeq, main="EGFR, hg19 vs hg38, Spearman's = 0.96",
↪ xlab="Rank RSEM hg19", ylab="Rank HTSeq hg38")

# As an exercise, try plotting some other genes. Maybe genes
# with both high and low correlations. What do you notice?
```

Let us know if you're having trouble! We're here to help.

Additional Resources:

- ISB-CGC examples-R github repo
- ISB-CGC R-workshop workshop material
- BigQuery web UI quickstart
- BigQuery 101 video
- Fun with a Petabyte: Pushing the limits of Google BigQuery video

---

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

26.3 Importing a GDC File Manifest into ISB-CGC

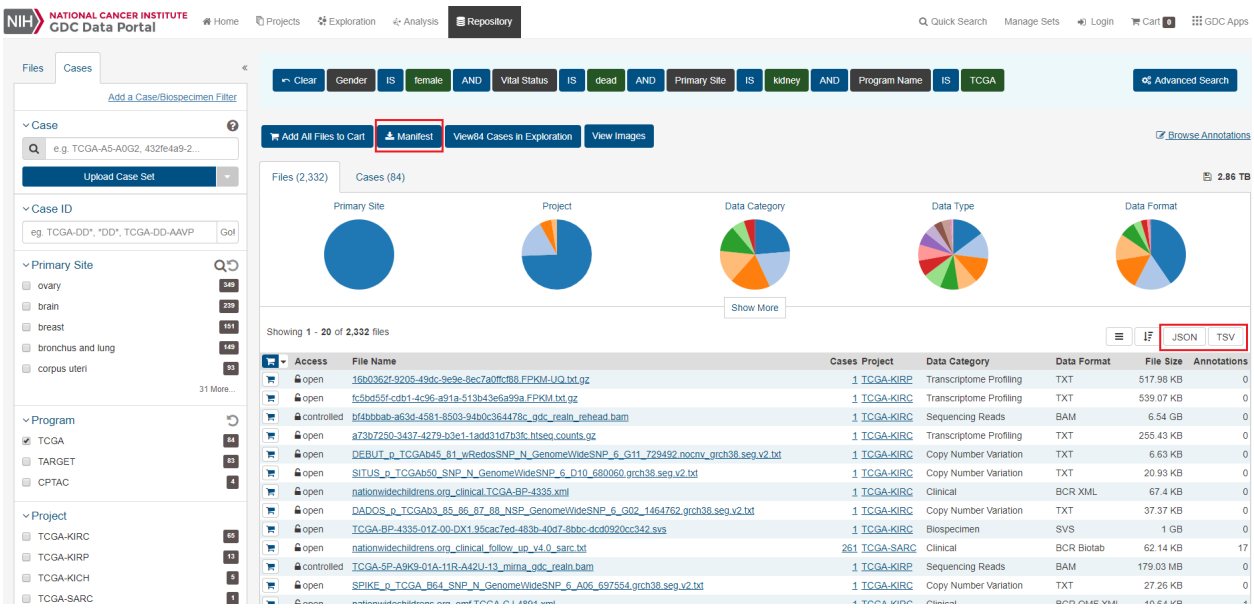
If you've been using the National Cancer Institute's Genomic Data Commons Portal, you know that while you can identify interesting cases and files, you need to download files to your own system in order to perform unique analysis.

Since the ISB-CGC stores Google Cloud file references for the GDC data, you can do your analysis on the cloud without having to move data. This tutorial will show you how to take a downloaded file manifest from the GDC, and use ISB-CGC to find the file locations on the cloud, providing a useful analysis starting point.

Download the File Manifest from GDC

On the GDC Data Portal, first use the selection filters to create your cohort. In the example shown below, the filters of Program: TCGA, Primary Site: kidney, Vital Status: dead and Gender: female were set to produce a cohort of 84 cases with 2332 files.

To download a File Manifest, which we'll use to find the file locations in ISB-CGC, on the **Repository** screen, click on the **Manifest** button.



Import the File Manifest into Google BigQuery

Importing a GDC file manifest into its own BigQuery table will enable you to join that table with an ISB-CGC BigQuery table containing the file locations on the Google Cloud. Here's how to do it.

If you don't already have a Google Cloud Project, please see the following ISB-CGC documentation pages for guidance:

- How to create a Google Cloud Platform (GCP) project
- How to link ISB-CGC BigQuery tables to your Google Cloud Platform (GCP) project

One way of keeping your file manifests organized is to create a data set specifically for those tables. New data sets can be created by clicking on the **Create Dataset** button within your project in BigQuery.

Creating a table from a GDC file manifest is remarkably easy:

- Click on the **Create Table** button while you are within your new data set.
- In the resulting screen, for **Create table from**, select **Upload**. Select your manifest file and set the **File format** to **CSV**. (Tab delimited will work with this setting.)
- Have BigQuery automatically create the schema by checking the **Auto detect** box for Schema.
- Click on **Advanced options**. Select **Tab** for **Field delimiter**; enter **1** for **Header rows to skip**.
- Click on the **Create Table** button.

Source

Create table from: Upload

Select file: gdc_manifest.2020-03-26.txt Browse

File format: CSV

Destination

Project name: kids-first-drc

Dataset name: GDC_Import

Table type: Native table

Table name: GDC_Kidney_File_manifest

Schema

Auto detect

☑ Schema and input parameters

Schema will be automatically generated.

Partition and cluster settings

Partitioning: No partitioning

Clustering order (optional):

Clustering order determines the sort order of the data. Clustering can only be used on a partitioned table, and works with tables partitioned either by column or ingestion time.

Comma-separated list of fields to define clustering order (up to 4)

Advanced options

Write preference: Write if empty

Number of errors allowed: 0

Unknown values: ☐ Ignore unknown values

Field delimiter: Tab

Header rows to skip: 1

Quoted newlines ☐ Allow quoted newlines

Jagged rows ☐ Allow jagged rows

Create table Cancel

Find the file locations on the Google Cloud

Now that you have a table containing the GDC file identifiers, the next step is to find the locations for the Level 1 files on the Google Cloud. To help with that task, ISB-CGC maintains BigQuery tables that contain the GDC file identifier and the Google bucket location for the file in data set GDC_metadata. Adding the Google bucket location to our GDC information can be done via a simple SQL query:

```
SELECT gdc.*, isb.file_gdc_url
FROM `Your-project.GDC_Import.GDC_Kidney_File_manifest` as gdc,
     `isb-cgc.GDC_metadata.rel22_GDCfileID_to_GCSurl` as isb
```

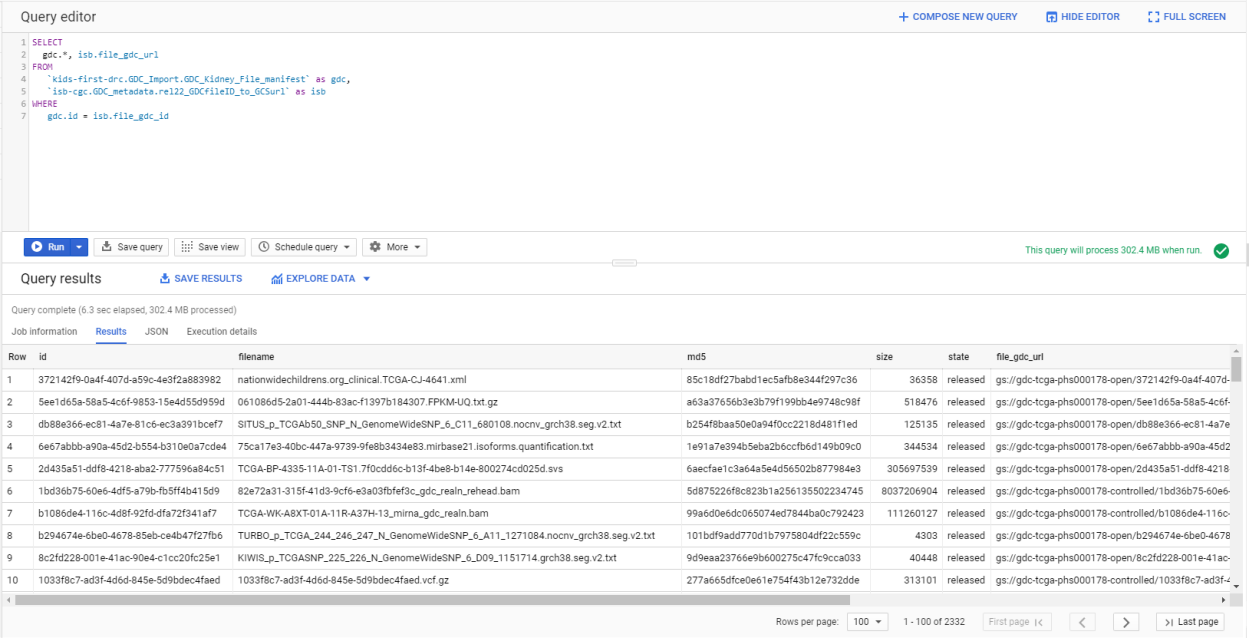
(continues on next page)

(continued from previous page)

```
WHERE gdc.id = isb.file_gdc_id
```

Note that you'll need to replace "Your-project.GDC_Import.GDC_Kidney_File_manifest" with your project and the data set and table that you created above.

This query will return the results shown below and, as with any BigQuery result, you can either export it as a file or save it as a new table in BigQuery.



Have feedback or corrections? Please email us at feedback@isb-cgc.org.

26.4 Analysis Using BigQuery, R & Bioconductor

In this tutorial, we are interested in analyzing gene expression and protein abundance differences between two types of TCGA kidney cancers, Kidney Renal Clear Cell Carcinoma (KIRC) and Kidney Renal Papillary Carcinoma (KIRP). We will build a cohort of patients with these cancer types and extract their respective gene expression and protein abundance data using Google BigQuery.

Note: An interactive step-by-step version of this tutorial can also be found on our homepage here.

This tutorial demonstrates how to:

- Identify tables of interest using the ISB-CGC BigQuery Table Search UI
- Navigate to tables in the Google BigQuery Console directly from the ISB-CGC BigQuery Table Search
- Build and run queries in the Google BigQuery Console
- Link to R notebooks in the Google AI Platform for data interrogation and plot visualization
- Use Bioconductor packages designed for TCGA data on ISB-CGC BigQuery tables

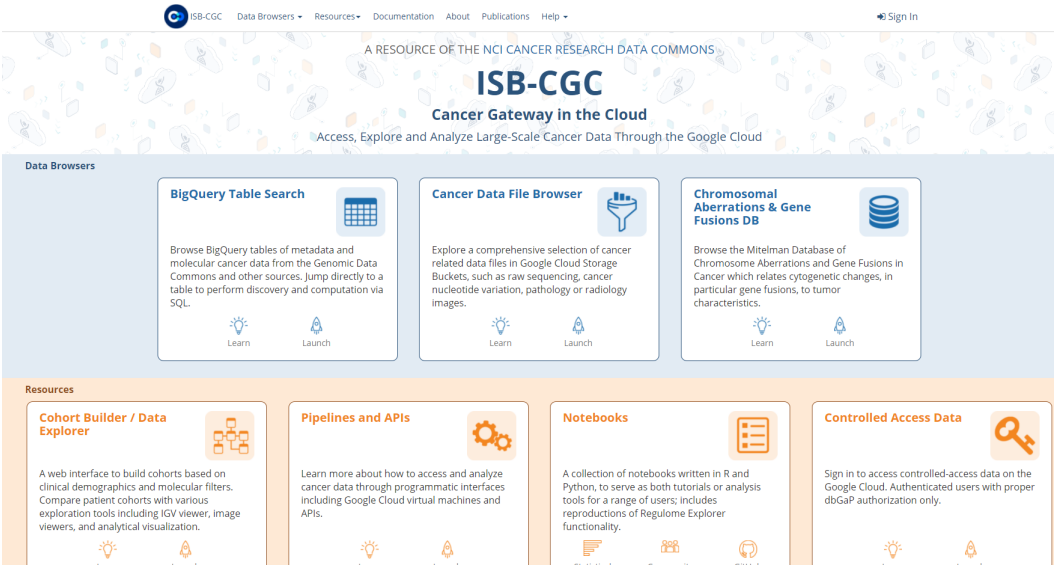
There are no prerequisites for using the ISB-CGC BigQuery Table Search, but in order to use the ISB-CGC tables in the Google Cloud BigQuery Console and within an R program, you'll need to have a Google Cloud Platform project and have linked it to the ISB-CGC BigQuery tables. Please see these sections of the ISB-CGC documentation for guidance:

- How to create a Google Cloud Platform (GCP) project
- How to link ISB-CGC BigQuery tables to your Google Cloud Platform (GCP) project

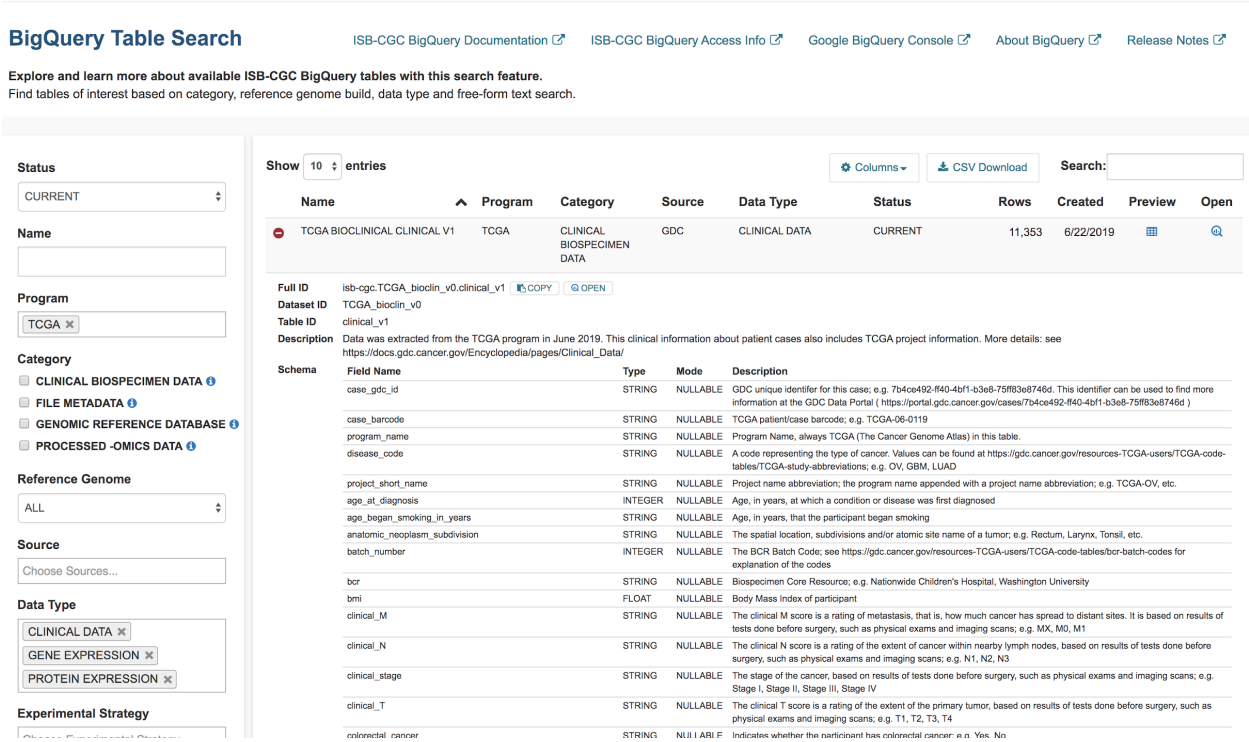
Click on the screenshots below to enlarge them.

26.4.1 Using the ISB-CGC BigQuery Table Search

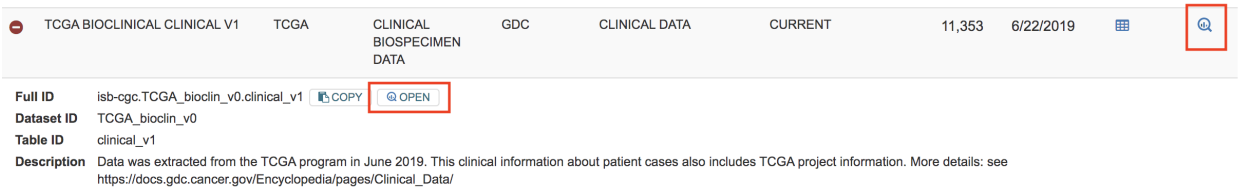
Navigate to the ISB-CGC homepage: https://isb-cgc.org and click on the **Launch** icon in the **BigQuery Table Search** box.



Let's search for ISB-CGC hosted BigQuery tables that contain information for TCGA gene expression, protein expression and clinical data. We want to build a cohort of TCGA patients for which both gene expression and protein abundance data exists. Enter **TCGA** in the **Program** filter and **Clinical Data**, **Gene Expression**, and **Protein Expression** in the **Data Type** filter. To see the table schema of the clinical table, click on the (+) icon.

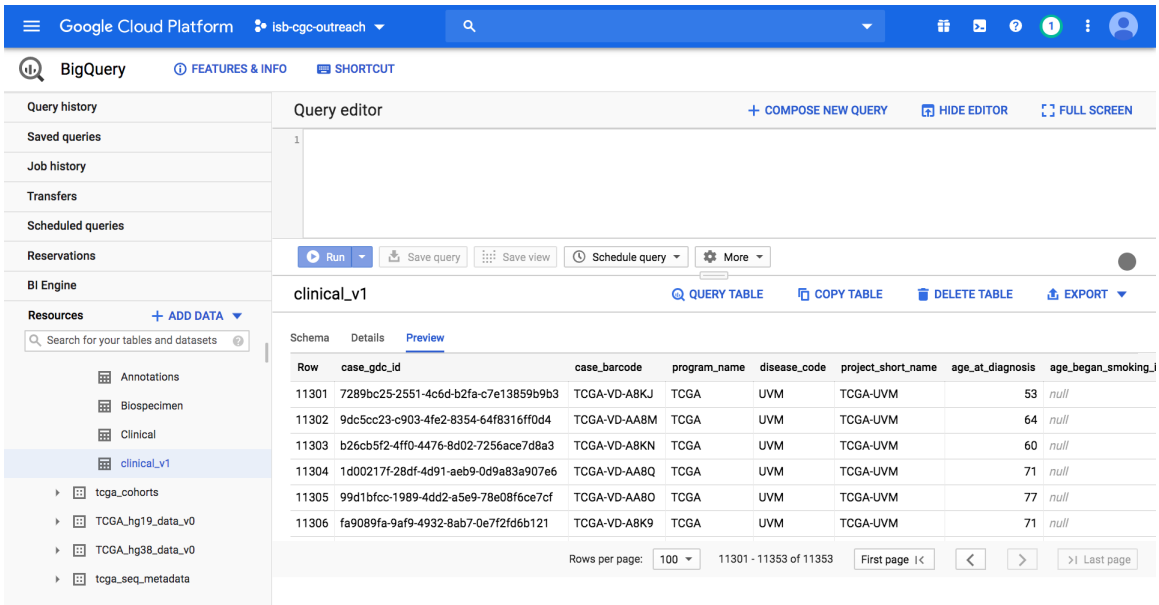


Navigate to the Google Cloud Platform (GCP) BigQuery Console by clicking on the "open" button under the table preview or on the "magnifying glass" icon on the right hand side of the Table Search row.



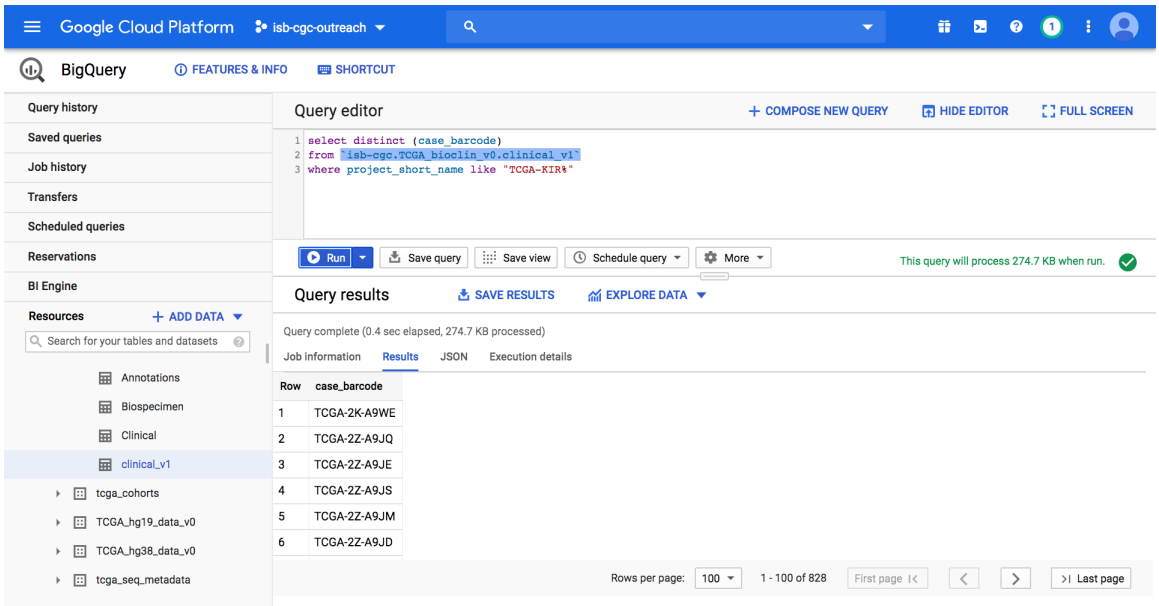
26.4.2 Using the Google Cloud BigQuery Console

On the GCP BigQuery Console we can preview the table, look at the schema, and perform queries. The image below shows the preview of the contents of the TCGA Clinical BigQuery table.



Here's a short SQL query (that completes in 0.3 seconds) which identifies how many patients there are with TCGA kidney cancers. Enter this SQL query in the BigQuery Console and click **Run**:

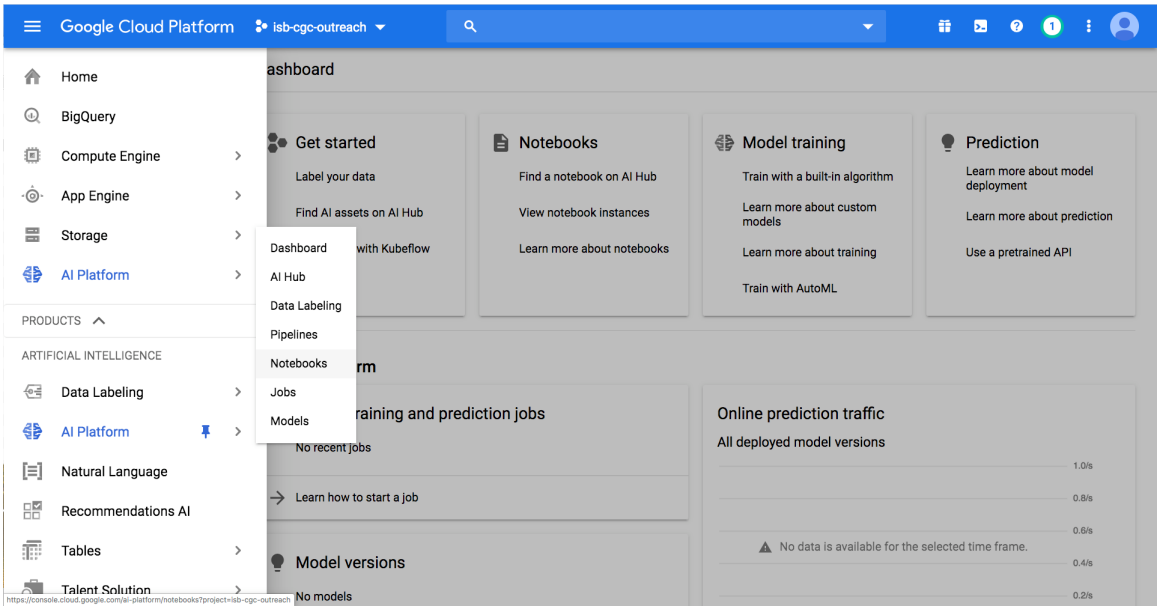
```
SELECT distinct (case_barcode)
FROM `isb-cgc.TCGA_bioclin_v0.clinical_v1`
WHERE project_short_name LIKE "TCGA-KIR%"
```



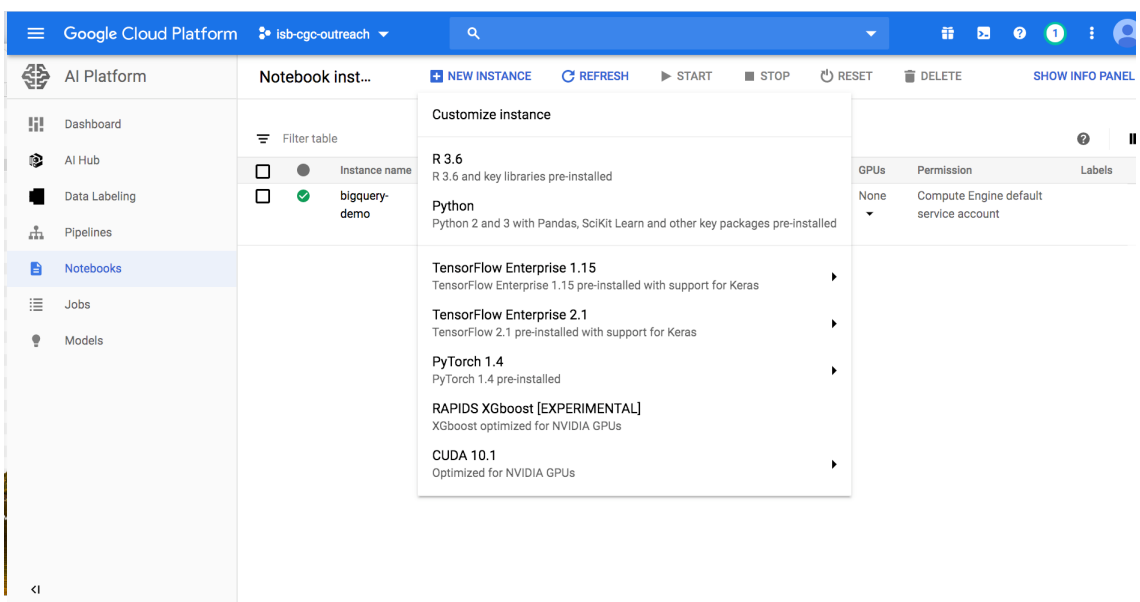
26.4.3 Using a Google Cloud AI Platform R Notebook and Bioconductor

From here, we can use either R or Python to perform higher level analyses. In this example, we will be running an R notebook in the Google Cloud AI Platform Notebooks environment. If you prefer, you can run this example in a local R environment instead.

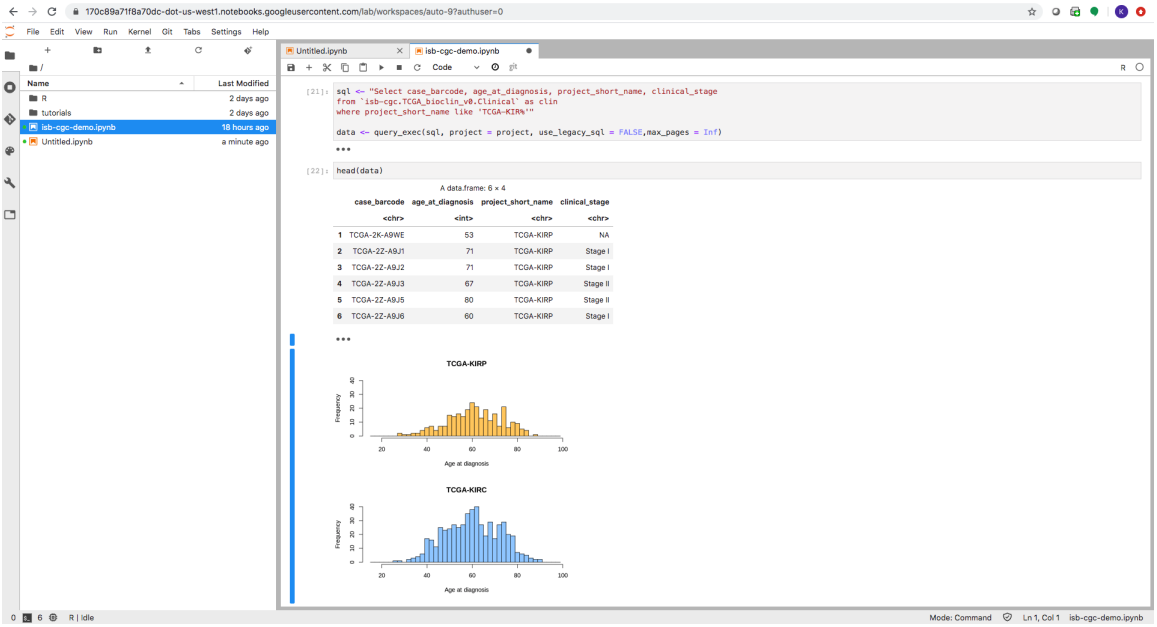
To use Google Cloud AI Platform Notebooks, from the Google Cloud Platform Navigation menu (on the left), select AI Platform -> Notebooks under the Artificial Intelligence section.



Notebooks can be created in both R or Python. We'll create our notebook in R.



The Google Cloud AI platform R notebook environment looks very similar to other Jupyter notebook environments. Users can create interactive R notebooks or simpler R console notebooks.



Enter or copy each block into the R terminal. Click **Run** after each block to see the results.

```
install.packages("bigrquery")
library(bigrquery)
project <- "your project" #Replace with your project name
```

```
# Query the clinical table for our cohort.
# Retrieve Age at Diagnosis and Clinical Stage for Kidney Cancer data.
sql <- "Select case_barcode, age_at_diagnosis, project_short_name, clinical_stage
        from `isb-cgc.TCGA_bioclin_v0.Clinical` as clin
        where project_short_name like 'TCGA-KIR%'"

clinical_tbl <- bq_project_query (project, query = sql) #Put data in temporary BQ table
clinical_data <- bq_table_download(clinical_tbl) #Put data into a dataframe
head(clinical_data)
```

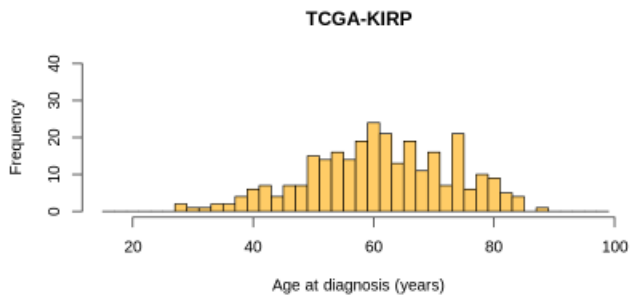
case_barcode	age_at_diagnosis	project_short_name	clinical_stage
<chr>	<int>	<chr>	<chr>
TCGA-2K-A9WE	53	TCGA-KIRP	NA
TCGA-2Z-A9J1	71	TCGA-KIRP	Stage I
TCGA-2Z-A9J2	71	TCGA-KIRP	Stage I
TCGA-2Z-A9J3	67	TCGA-KIRP	Stage II
TCGA-2Z-A9J5	80	TCGA-KIRP	Stage II
TCGA-2Z-A9J6	60	TCGA-KIRP	Stage I

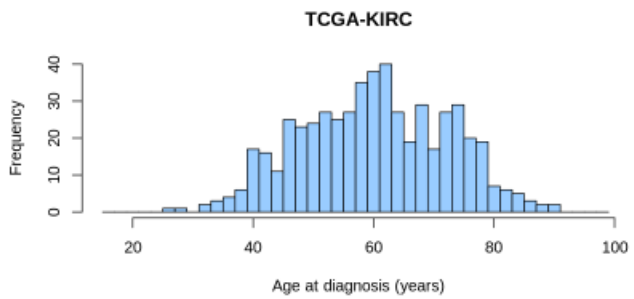
```
# Plot two histograms of age of diagnosis data of our cohort.
layout(matrix(1:2, 2, 1))
hist(clinical_data[clinical_data$project_short_name == "TCGA-KIRP",]$age_at_diagnosis,
    xlim=c(15,100), ylim=c(0,40), breaks=seq(15,100,2),
```

(continues on next page)

(continued from previous page)

```
    col="#FFCC66", main='TCGA-KIRP', xlab='Age at diagnosis (years)')

hist(clinical_data[clinical_data$project_short_name == "TCGA-KIRC",]$age_at_diagnosis,
    xlim=c(15,100), ylim=c(0,40), breaks=seq(15,100,2),
    col="#99CCFF", main='TCGA-KIRC', xlab='Age at diagnosis (years)')
```






```
# Create SQL query to retrieve the mean gene expression and mean protein expression per
↪project/case.
# Load it into a dataframe.
sql_expression <- "with gexp as (
    select project_short_name, case_barcode, gene_name, avg(HTSeq__FPKM) as mean_gexp
    from `isb-cgc.TCGA_hg38_data_v0.RNAseq_Gene_Expression`
    where project_short_name like 'TCGA-KIR%' and gene_type = 'protein_coding'
    group by project_short_name, case_barcode, gene_name
), pexp as (
    select project_short_name, case_barcode, gene_name, avg(protein_expression) as mean_
↪pexp
    from `isb-cgc.TCGA_hg38_data_v0.Protein_Expression`
    where project_short_name like 'TCGA-KIR%'
    group by project_short_name, case_barcode, gene_name
)
select gexp.project_short_name, gexp.case_barcode, gexp.gene_name, gexp.mean_gexp, pexp.
↪mean_pexp
from gexp inner join pexp
on gexp.project_short_name = pexp.project_short_name
  and gexp.case_barcode = pexp.case_barcode
  and gexp.gene_name = pexp.gene_name"

expression_data <- bq_table_download(bq_project_query (project, query = sql_expression))
↪#Put data into a dataframe
head(expression_data)
```

project_short_name	case_barcode	gene_name	mean_gexp	mean_pexp
<chr>	<chr>	<chr>	<dbl>	<dbl>
TCGA-KIRP	TCGA-UZ-A9PS	CTNNA1	93.41217	-0.09747864
TCGA-KIRC	TCGA-CZ-5467	PYGM	0.49243	0.24809379
TCGA-KIRC	TCGA-CJ-4886	SDHB	27.18014	-0.07058236
TCGA-KIRC	TCGA-CW-5583	CA9	162.15654	-0.20347070
TCGA-KIRC	TCGA-AK-3436	PKM	305.69375	1.02044110
TCGA-KIRC	TCGA-CW-6090	SDHB	42.79914	-0.55778735

```
# Determine the number of cases from each project.
length(unique(expression_data$case_barcode[expression_data$project_short_name == "TCGA-
↪KIRP"]))
length(unique(expression_data$case_barcode[expression_data$project_short_name == "TCGA-
↪KIRC"]))
```

```
214
472
```

```
#Create a dataframe that lists all the cases.
expression_data$id <- paste(expression_data$project_short_name, expression_data$case_
↪barcode, sep='.')
cases <- unique(expression_data$id)

# Transform the expression_data data frame, so that columns are samples, rows are genes.
list_exp <- lapply(cases, function(case){
    temp <- expression_data[expression_data$id == case, c('gene_name', 'mean_gexp')]
    names(temp) <- c('gene_name', case)
    return(temp)
})

gene_exps <- Reduce(function(x, y) merge(x, y, all=T, by="gene_name"), list_exp)
head(gene_exps)
dim(gene_exps)
```

A data.frame: 6 × 687

	gene_name	TCGA-KIRP.TCGA-UZ-A9PS	TCGA-KIRC.TCGA-CZ-5467	TCGA-KIRC.TCGA-CJ-4886	TCGA-KIRC.TCGA-CW-5583	TCGA-KIRC.TCGA-AK-3436	TCGA-KIRC.TCGA-CW-6090	TCGA-KIRC.TCGA-B0-5080	TCGA-KIRC.TCGA-BP-4338	TCGA-KIRC.TCGA-BP-5184	...	TCGA-KIRP.TCGA-IA-A83W
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	...	<dbl>
1	ABL1	18.839194	13.204290	15.770270	19.038732	6.013492	12.905264	21.834588	9.709215	15.731076	...	14.339578
2	ACACA	3.824316	3.569681	2.204905	2.643183	1.610779	2.372528	3.218430	3.012679	1.831968	...	2.524890
3	ACACB	2.993119	4.807056	2.442744	4.811248	2.442785	3.457209	1.489348	4.628481	2.138174	...	4.915340
4	ACVRL1	2.060649	15.442982	26.626688	19.232095	11.531087	10.714139	44.209905	6.758663	21.674077	...	7.113441
5	ADAR	29.146091	35.793838	41.887275	44.842195	22.106074	38.817011	34.015421	28.103323	30.840014	...	25.879441
6	AKT1	8.925272	10.079546	9.679650	11.041838	5.322968	6.572360	11.208015	8.713414	11.347751	...	9.271672

196 · 687

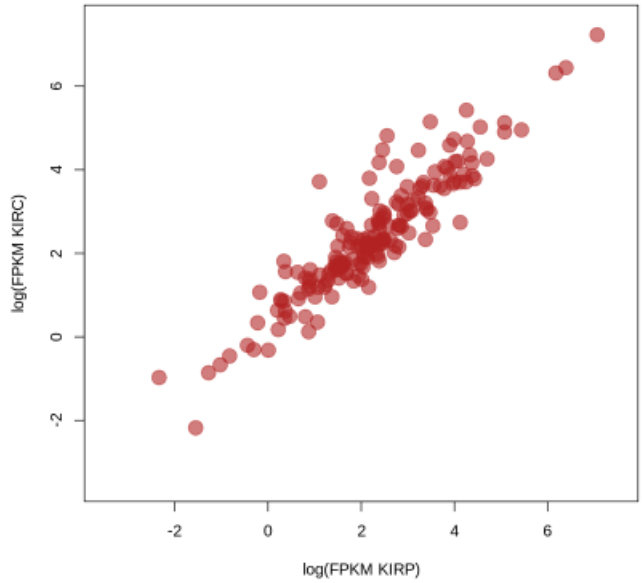
```
# Perform the same transform for protein abundance.
  list_abun <- lapply(cases, function(case){
      temp <- expression_data[expression_data$id == case, c('gene_name', 'mean_pexp')]
      names(temp) <- c('gene_name', case)
      return(temp)
  })
  pep_abun <- Reduce(function(x, y) merge(x, y, all=T, by="gene_name"), list_abun)
  head(pep_abun)
  dim(pep_abun)
```

A data.frame: 6 × 687

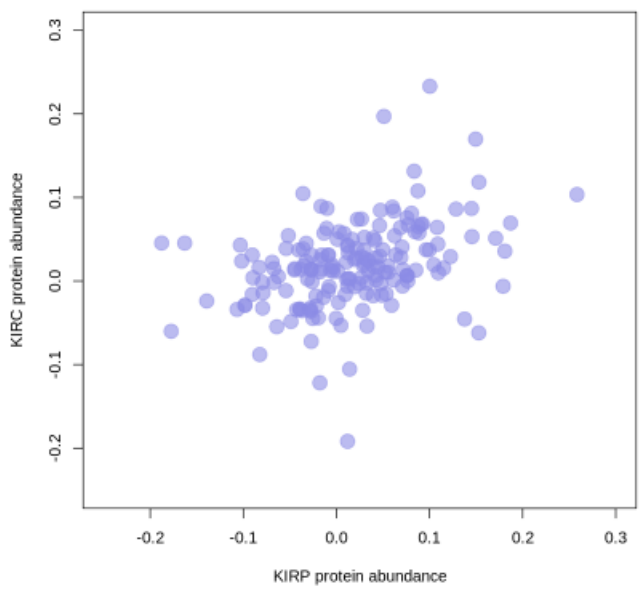
	gene_name	TCGA-KIRP.TCGA-UZ-A9PS	TCGA-KIRC.TCGA-CZ-5467	TCGA-KIRC.TCGA-CJ-4886	TCGA-KIRC.TCGA-CW-5583	TCGA-KIRC.TCGA-AK-3436	TCGA-KIRC.TCGA-CW-6090	TCGA-KIRC.TCGA-B0-5080	TCGA-KIRC.TCGA-BP-4338	TCGA-KIRC.TCGA-BP-5184	...	TCGA-KIRP.TCGA-IA-A83W
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	...	<dbl>
1	ABL1	-0.09664002	0.21942807	-0.007042168	-0.03515470	-0.2331283	-0.10948885	0.08025133	0.03148955	-0.12268847	...	-0.02385270
2	ACACA	-0.27065025	-0.23535413	-0.240316365	-0.39147438	0.1332921	-0.30670630	0.06110671	0.75291898	-0.41855657	...	-0.46794541
3	ACACB	-0.27961784	-0.14651645	-0.157676117	-0.30705621	0.3346727	-0.33232132	0.12746179	0.76752227	-0.46973348	...	-0.41733040
4	ACVRL1	0.14179396	-0.21096608	0.274044723	-0.05665730	0.1138674	-0.46641370	-0.17392663	0.12770249	-0.05799289	...	0.08964699
5	ADAR	-0.10918400	0.08549571	-0.197059759	-0.03109023	0.0179514	0.16353338	-0.04067161	0.08761717	-0.38553785	...	-0.04609027
6	AKT1	0.23840072	-0.24432873	0.071690661	0.76875954	-0.1018915	-0.07049873	-0.12648343	-0.84745613	0.74019487	...	0.50708815

196 · 687

```
# Separate the cohorts (types of kidney cancer) into two dataframes and
# generate a scatterplot of gene expression and protein abundance.
# Gene expression first.
exp_p <- gene_exps[,grep('KIRP', names(gene_exps))]
exp_c <- gene_exps[,grep('KIRC', names(gene_exps))]
plot(log(rowMeans(exp_p)), log(rowMeans(exp_c)),
    xlab='log(FPKM KIRP)', ylab='log(FPKM KIRC)',
    xlim=c(-3.5,7.5), ylim=c(-3.5,7.5), pch=19, cex=2,
    col=rgb(178,34,34,max=255,alpha=150))
```




```
# Peptide expression second.
abun_p <- pep_abun[,grep('KIRP', names(pep_abun))]
abun_c <- pep_abun[,grep('KIRC', names(pep_abun))]
plot(rowMeans(abun_p), rowMeans(abun_c),
   xlab='KIRP protein abundance', ylab="KIRC protein abundance",
   xlim=c(-0.25,0.3), ylim=c(-0.25,0.3), pch=19, cex=2,
   col=rgb(140,140,230,max=255,alpha=150))
```




```
# Load the Bioconductor package maftools, which has capabilities to summarize,
# analyze and visualize Mutation Annotation Format (MAF) data.
install.packages("maftools")
library("maftools")
```

```
# Use BigQuery to load TCGA somatic mutation data for our cancers of interest.
sql_kirc<-"SELECT Hugo_Symbol, Chromosome, Start_Position, End_Position, Reference_
↪Allele,
Tumor_Seq_Allele2, Variant_Classification, Variant_Type, sample_barcode_tumor FROM
`isb-cgc.TCGA_hg38_data_v0.Somatic_Mutation` WHERE project_short_name = 'TCGA-KIRC'"

sql_kirp<-"SELECT Hugo_Symbol, Chromosome, Start_Position, End_Position, Reference_
↪Allele,
Tumor_Seq_Allele2, Variant_Classification, Variant_Type, sample_barcode_tumor FROM
`isb-cgc.TCGA_hg38_data_v0.Somatic_Mutation` WHERE project_short_name = 'TCGA-KIRP'"

maf_kirc <- bq_table_download(bq_project_query (project, query = sql_kirc)) #Put data␣
↪into a dataframe
maf_kirp <- bq_table_download(bq_project_query (project, query = sql_kirp)) #Put data␣
↪into a dataframe

#Rename column 9 to the field name required by maftools.
colnames(maf_kirc)[9] <- "Tumor_Sample_Barcode"
colnames(maf_kirp)[9] <- "Tumor_Sample_Barcode"

head(maf_kirc)
head(maf_kirp)
```

A tibble: 6 × 9

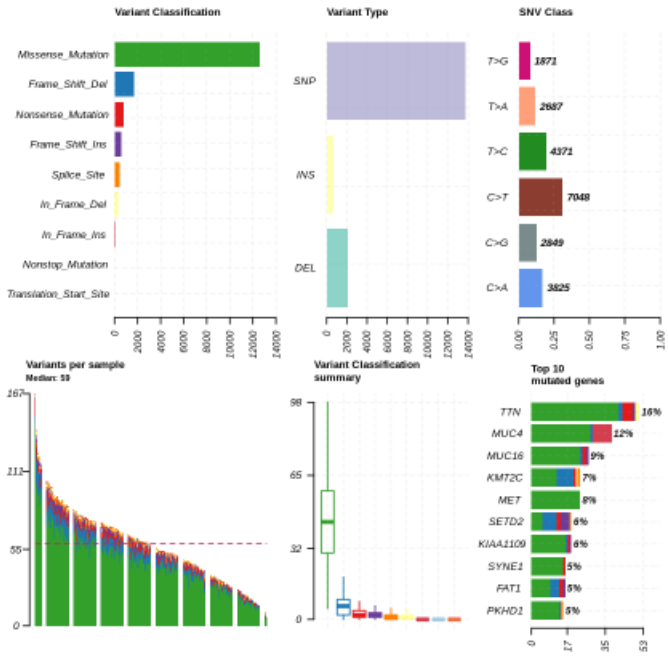
Hugo_Symbol	Chromosome	Start_Position	End_Position	Reference_Allele	Tumor_Seq_Allele2	Variant_Classification	Variant_Type	Tumor_Sample_Barcode
<chr>	<chr>	<int>	<int>	<chr>	<chr>	<chr>	<chr>	<chr>
RP11-160C18.4	chr15	78753126	78753126	A	G	RNA	SNP	TCGA-A3-3380-01A
SUPT20HL2	chrX	24312504	24312504	T	A	RNA	SNP	TCGA-A3-A8OU-01A
TUBBP5	chr9	138175461	138175461	C	T	RNA	SNP	TCGA-AK-3465-01A
MSL3P1	chr2	233866793	233866793	T	A	RNA	SNP	TCGA-B0-4827-01A
CTD-2303H24.2	chr17	18514649	18514649	G	A	RNA	SNP	TCGA-B0-4842-01A
BAGE2	chr21	10413724	10413724	C	T	RNA	SNP	TCGA-B0-5075-01A

A tibble: 6 × 9

Hugo_Symbol	Chromosome	Start_Position	End_Position	Reference_Allele	Tumor_Seq_Allele2	Variant_Classification	Variant_Type	Tumor_Sample_Barcode
<chr>	<chr>	<int>	<int>	<chr>	<chr>	<chr>	<chr>	<chr>
MIR99AHG	chr21	16231065	16231065	A	C	RNA	SNP	TCGA-5P-A9JU-01A
TRIM51HP	chr11	55297929	55297929	A	T	RNA	SNP	TCGA-5P-A9JV-01A
CEP170P1	chr4	118513757	118513761	GAAAG	-	RNA	DEL	TCGA-5P-A9K0-01A
DPY19L1P1	chr7	32632560	32632560	T	A	RNA	SNP	TCGA-5P-A9KH-01A
MIR517C	chr19	53741314	53741314	A	C	RNA	SNP	TCGA-A4-A48D-01A
AOC4P	chr17	42868538	42868538	C	T	RNA	SNP	TCGA-B1-A656-01A

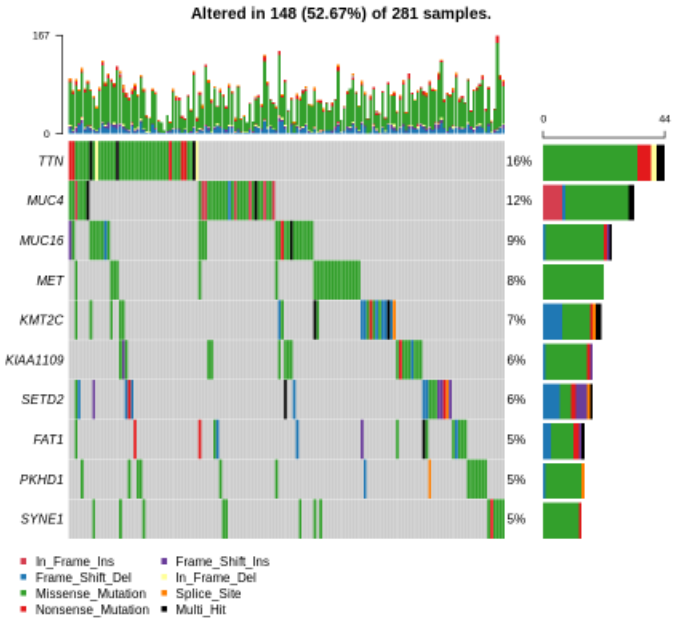
```
# Convert data frames to maftools objects.
kirc <- read.maf(maf_kirc)
kirp <- read.maf(maf_kirp)
# Leverage maftools plotting functionality.
plotmafSummary(maf = kirp, rmOutlier = TRUE, addStat = 'median', dashboard = TRUE,␣
↪titvRaw = FALSE)
plotmafSummary(maf = kirc, rmOutlier = TRUE, addStat = 'median', dashboard = TRUE,␣
↪titvRaw = FALSE)
```

Here is the MAF Plot Summary for Kidney Renal Papillary Carcinoma.




```
oncoplot(maf = kirp, top = 10)
oncoplot(maf = kirc, top = 10)
```

Here is the oncoplot for Kidney Renal Papillary Carcinoma.



Have feedback or corrections? Please email us at feedback@isb-cgc.org.

26.5 Teaching Sessions and Workshops

26.5.1 NIH Library Session - October 14th, 2021

We offered a half-day online bioinformatics workshop in collaboration with the NIH Library on October 14th, 2021. This workshop included a two hour interactive data science and bioinformatics component using the R statistical language and Google Cloud (BigQuery) to explore NCI genomic and proteomics (TCGA) datasets. The following outline of the interactive workshop links to Jupyter notebooks used during the training. These notebooks can be executed in Google Colab or other Jupyter environments.

- Exploration of BigQuery Datasets
  - Select age of cases in a project
  - Characterize expression of a single gene in normal and tumor samples
  - Find annotation information for a gene
  - Find and plot all mutations in and around a gene
  - Join mutation data with survival data
- Large-Scale Statistical Analysis in BigQuery
- Survival Analysis in BigQuery
- Introduction to BigQuery Machine Learning
  - Train a classifier that uses expression profiles to predict tumor status
  - Evaluate the classifier
  - Use the classifier to predict tumor status of future samples

---

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CHAPTER

TWENTYSEVEN

RELEASE NOTES

The ISB-CGC has created documentation to inform researchers about major changes between the ISB-CGC Data Releases, ISB-CGC Table Search, and the ISB-CGC WebApp. For more information, please select one of the options below.

27.1 ISB-CGC Data Release Notes

June 23, 2022

HTAN data added

BigQuery tables created

- isb-cgc-bq.HTAN_versioned.scRNAseq_CHOP_seurat_regrCycleHeatShockGenes_pool_18Infants_scRNA_VEG3000_updated_
- isb-cgc-bq.HTAN_versioned.scRNAseq_CHOP_seurat_pool_logNorm_gini_FiveHD_10Xv3_downsample10000HSPC_r2
- isb-cgc-bq.HTAN_versioned.scRNAseq_CHOP_seurat_integrated_18MLLr_normal_final_rename_r2
- isb-cgc-bq.HTAN_versioned.schema_r2
- isb-cgc-bq.HTAN_versioned.scRNAseq_VUMC_HTAN_VAL_EPI_V2_r2
- isb-cgc-bq.HTAN_versioned.scRNAseq_VUMC_HTAN_VAL_DIS_NONEPI_V2_r2
- isb-cgc-bq.HTAN_versioned.scRNAseq_VUMC_HTAN_DIS_EPI_V2_r2
- isb-cgc-bq.HTAN_versioned.scRNAseq_VUMC_ABNORMALS_EPI_V2_r2
- isb-cgc-bq.HTAN_versioned.clinical_tier1_therapy_r2
- isb-cgc-bq.HTAN_versioned.scRNAseq_level4_metadata_r2
- isb-cgc-bq.HTAN_versioned.scRNAseq_level3_metadata_r2
- isb-cgc-bq.HTAN_versioned.scRNAseq_level2_metadata_r2
- isb-cgc-bq.HTAN_versioned.scRNAseq_level1_metadata_r2
- isb-cgc-bq.HTAN_versioned.scATACseq_level4_metadata_r2
- isb-cgc-bq.HTAN_versioned.scATACseq_level3_metadata_r2
- isb-cgc-bq.HTAN_versioned.scATACseq_level1_metadata_r2
- isb-cgc-bq.HTAN_versioned.srrs_imaging_level2_metadata_r2
- isb-cgc-bq.HTAN_versioned.srrs_clinical_tier2_r2
- isb-cgc-bq.HTAN_versioned.srrs_biospecimen_r2

- isb-cgc-bq.HTAN_versioned.proteomics_metadata_r2
- isb-cgc-bq.HTAN_versioned.clinical_tier1_moleculartest_r2`
- isb-cgc-bq.HTAN_versioned.metabolomics_metadata_r2
- isb-cgc-bq.HTAN_versioned.clinical_tier3_lung_r2
- isb-cgc-bq.HTAN_versioned.lipidomics_metadata_r2
- isb-cgc-bq.HTAN_versioned.imaging_level2_metadata_r2
- isb-cgc-bq.HTAN_versioned.clinical_tier1_followup_r2
- isb-cgc-bq.HTAN_versioned.clinical_tier1_familyhistory_r2
- isb-cgc-bq.HTAN_versioned.clinical_tier1_exposure_r2
- isb-cgc-bq.HTAN_versioned.clinical_tier1_diagnosis_r2
- isb-cgc-bq.HTAN_versioned.clinical_tier1_demographics_r2
- isb-cgc-bq.HTAN_versioned.clinical_tier2_r2
- isb-cgc-bq.HTAN_versioned.bulkWES_level2_metadata_r2
- isb-cgc-bq.HTAN_versioned.bulkWES_level1_metadata_r2
- isb-cgc-bq.HTAN_versioned.bulkRNAseq_level3_metadata_r2
- isb-cgc-bq.HTAN_versioned.bulkRNAseq_level2_metadata_r2
- isb-cgc-bq.HTAN_versioned.bulkRNAseq_level1_metadata_r2
- isb-cgc-bq.HTAN_versioned.clinical_tier3_breast_r2
- isb-cgc-bq.HTAN_versioned.biospecimen_r2
- isb-cgc-bq.HTAN.scRNAseq_CHOP_seurat_regrCycleHeatShockGenes_pool_18Infants_scRNA_VEG3000_updated_rename_cu
- isb-cgc-bq.HTAN.scRNAseq_CHOP_seurat_pool_logNorm_gini_FiveHD_10Xv3_downsample10000HSPC_current
- isb-cgc-bq.HTAN.scRNAseq_CHOP_seurat_integrated_18MLLr_normal_final_rename_current
- isb-cgc-bq.HTAN.schema_current
- isb-cgc-bq.HTAN.scRNAseq_VUMC_HTAN_VAL_EPI_V2_current
- isb-cgc-bq.HTAN.scRNAseq_VUMC_HTAN_VAL_DIS_NONEPI_V2_current
- isb-cgc-bq.HTAN.scRNAseq_VUMC_HTAN_DIS_EPI_V2_current
- isb-cgc-bq.HTAN.scRNAseq_VUMC_ABNORMALS_EPI_V2_current
- isb-cgc-bq.HTAN.clinical_tier1_therapy_current
- isb-cgc-bq.HTAN.scRNAseq_level4_metadata_current
- isb-cgc-bq.HTAN.scRNAseq_level3_metadata_current
- isb-cgc-bq.HTAN.scRNAseq_level2_metadata_current
- isb-cgc-bq.HTAN.scRNAseq_level1_metadata_current
- isb-cgc-bq.HTAN.scATACseq_level4_metadata_current
- isb-cgc-bq.HTAN.scATACseq_level3_metadata_current
- isb-cgc-bq.HTAN.scATACseq_level1_metadata_current
- isb-cgc-bq.HTAN.srrs_imaging_level2_metadata_current

- isb-cgc-bq.HTAN.srrs_clinical_tiecurrent_current
- isb-cgc-bq.HTAN.srrs_biospecimen_current
- isb-cgc-bq.HTAN.proteomics_metadata_current
- isb-cgc-bq.HTAN.clinical_tier1_moleculartest_current
- isb-cgc-bq.HTAN.metabolomics_metadata_current
- isb-cgc-bq.HTAN.clinical_tier3_lung_current
- isb-cgc-bq.HTAN.lipidomics_metadata_current
- isb-cgc-bq.HTAN.imaging_level2_metadata_current
- isb-cgc-bq.HTAN.clinical_tier1_followup_current
- isb-cgc-bq.HTAN.clinical_tier1_familyhistory_current
- isb-cgc-bq.HTAN.clinical_tier1_exposure_current
- isb-cgc-bq.HTAN.clinical_tier1_diagnosis_current
- isb-cgc-bq.HTAN.clinical_tier1_demographics_current
- isb-cgc-bq.HTAN.clinical_tier2_current
- isb-cgc-bq.HTAN.bulkWES_level2_metadata_current
- isb-cgc-bq.HTAN.bulkWES_level1_metadata_current
- isb-cgc-bq.HTAN.bulkRNAseq_level3_metadata_current
- isb-cgc-bq.HTAN.bulkRNAseq_level2_metadata_current
- isb-cgc-bq.HTAN.bulkRNAseq_level1_metadata_current
- isb-cgc-bq.HTAN.clinical_tier3_breast_current
- isb-cgc-bq.HTAN.biospecimen_current

June 15, 2022

New clinical tables added to isb-cgc-bq for GDC release 33.

BigQuery tables created

- isb-cgc-bq.TRIO.clinical_gdc_current
- isb-cgc-bq.TRIO_versioned.clinical_gdc_r33
- isb-cgc-bq.TARGET_versioned.clinical_gdc_r33
- isb-cgc-bq.HCMI_versioned.clinical_follow_ups_molecular_tests_gdc_r33
- isb-cgc-bq.HCMI_versioned.clinical_follow_ups_gdc_r33
- isb-cgc-bq.HCMI_versioned.clinical_diagnoses_treatments_gdc_r33
- isb-cgc-bq.HCMI_versioned.clinical_diagnoses_gdc_r33
- isb-cgc-bq.CTSP_versioned.clinical_gdc_r33
- isb-cgc-bq.CMI_versioned.clinical_gdc_r33
- isb-cgc-bq.CGCI_versioned.clinical_follow_ups_molecular_tests_gdc_r33
- isb-cgc-bq.CGCI_versioned.clinical_follow_ups_gdc_r33
- isb-cgc-bq.CGCI_versioned.clinical_diagnoses_treatments_gdc_r33

- isb-cgc-bq.CGCI_versioned.clinical_diagnoses_gdc_r33
- isb-cgc-bq.CGCI_versioned.clinical_gdc_r33
- isb-cgc-bq.MP2PRT.clinical_gdc_current
- isb-cgc-bq.MP2PRT_versioned.clinical_gdc_r33
- isb-cgc-bq.EXC_RESPONDERS.clinical_diagnoses_treatments_gdc_current
- isb-cgc-bq.EXC_RESPONDERS_versioned.clinical_diagnoses_treatments_gdc_r33
- isb-cgc-bq.EXC_RESPONDERS.clinical_diagnoses_gdc_current
- isb-cgc-bq.EXC_RESPONDERS_versioned.clinical_diagnoses_gdc_r33
- isb-cgc-bq.EXC_RESPONDERS.clinical_gdc_current
- isb-cgc-bq.EXC_RESPONDERS_versioned.clinical_gdc_r33

BigQuery tables updated

- isb-cgc-bq.TARGET.clinical_gdc_current
- isb-cgc-bq.HCMI.clinical_follow_ups_molecular_tests_gdc_current
- isb-cgc-bq.HCMI.clinical_follow_ups_gdc_current
- isb-cgc-bq.HCMI.clinical_diagnoses_treatments_gdc_current
- isb-cgc-bq.HCMI.clinical_diagnoses_gdc_current
- isb-cgc-bq.CTSP.clinical_gdc_current
- isb-cgc-bq.CMI.clinical_gdc_current
- isb-cgc-bq.CGCI.clinical_follow_ups_molecular_tests_gdc_current
- isb-cgc-bq.CGCI.clinical_follow_ups_gdc_current
- isb-cgc-bq.CGCI.clinical_diagnoses_treatments_gdc_current
- isb-cgc-bq.CGCI.clinical_diagnoses_gdc_current
- isb-cgc-bq.CGCI.clinical_gdc_current

May 5, 2022

New file metadata tables added to isb-cgc-bq for GDC release 32.

BigQuery tables created

- isb-cgc-bq.GDC_case_file_metadata_versioned.fileData_active_r32
- isb-cgc-bq.GDC_case_file_metadata_versioned.fileData_legacy_r32
- isb-cgc-bq.GDC_case_file_metadata_versioned.caseData_r32
- isb-cgc-bq.GDC_case_file_metadata_versioned.aliquot2caseIDmap_r32
- isb-cgc-bq.GDC_case_file_metadata_versioned.slide2caseIDmap_r32
- isb-cgc-bq.GDC_case_file_metadata.fileData_active_current
- isb-cgc-bq.GDC_case_file_metadata.fileData_legacy_current
- isb-cgc-bq.GDC_case_file_metadata.caseData_current
- isb-cgc-bq.GDC_case_file_metadata.aliquot2caseIDmap_current
- isb-cgc-bq.GDC_case_file_metadata.slide2caseIDmap_current

April 28, 2022

Cluster these TCGA DNA methylation and TCGA RNAseq tables to improve query performance.

BigQuery tables created

- isb-cgc-bq.TCGA.DNA_methylation_hg19_gdc_current
- isb-cgc-bq.TCGA.DNA_methylation_hg38_gdc_current
- isb-cgc-bq.TCGA.RNAseq_hg19_gdc_current
- isb-cgc-bq.TCGA.RNAseq_hg38_gdc_current

BigQuery tables updated

- isb-cgc-bq.TCGA.per_sample_file_metadata_hg38_gdc_current

February 10, 2022

New clinical tables added to isb-cgc-bq for GDC release 31.

BigQuery tables created

- isb-cgc-bq.REBC_versioned.clinical_gdc_r31
- isb-cgc-bq.REBC.clinical_gdc_current
- isb-cgc-bq.REBC_versioned.clinical_diagnoses_treatments_gdc_r31
- isb-cgc-bq.REBC.clinical_diagnoses_treatments_gdc_current
- isb-cgc-bq.TRIO_versioned.clinical_gdc_r31
- isb-cgc-bq.TRIO.clinical_gdc_current
- isb-cgc-bq.BEATAML1_0_versioned.clinical_gdc_r31
- isb-cgc-bq.CGCI_versioned.clinical_gdc_r31
- isb-cgc-bq.CGCI_versioned.clinical_diagnoses_gdc_r31
- isb-cgc-bq.CGCI_versioned.clinical_diagnoses_treatments_gdc_r31
- isb-cgc-bq.CGCI_versioned.clinical_follow_ups_gdc_r31
- isb-cgc-bq.CPTAC_versioned.clinical_gdc_r31
- isb-cgc-bq.CTSP_versioned.clinical_gdc_r31
- isb-cgc-bq.FM_versioned.clinical_gdc_r31
- isb-cgc-bq.GENIE_versioned.clinical_gdc_r31
- isb-cgc-bq.HCMI_versioned.clinical_gdc_r31
- isb-cgc-bq.HCMI_versioned.clinical_diagnoses_gdc_r31
- isb-cgc-bq.HCMI_versioned.clinical_diagnoses_treatments_gdc_r31
- isb-cgc-bq.HCMI_versioned.clinical_follow_ups_gdc_r31
- isb-cgc-bq.HCMI_versioned.clinical_follow_ups_molecular_tests_gdc_r31
- isb-cgc-bq.MMRF_versioned.clinical_gdc_r31
- isb-cgc-bq.NCICCR_versioned.clinical_gdc_r31
- isb-cgc-bq.OHSU_versioned.clinical_gdc_r31
- isb-cgc-bq.ORGANOID_versioned.clinical_gdc_r31

- isb-cgc-bq.TARGET_versioned.clinical_gdc_r31
- isb-cgc-bq.TCGA_versioned.clinical_gdc_r31
- isb-cgc-bq.VAREPOP_versioned.clinical_gdc_r31
- isb-cgc-bq.WCDT_versioned.clinical_gdc_r31

Current clinical tables updated to GDC release 31.

BigQuery tables updated

- isb-cgc-bq.BEATAML1_0.clinical_gdc_current
- isb-cgc-bq.CGCI.clinical_gdc_current
- isb-cgc-bq.CGCI.clinical_diagnoses_gdc_current
- isb-cgc-bq.CGCI.clinical_diagnoses_treatments_gdc_current
- isb-cgc-bq.CGCI.clinical_follow_ups_gdc_current
- isb-cgc-bq.CPTAC.clinical_gdc_current
- isb-cgc-bq.CTSP.clinical_gdc_current
- isb-cgc-bq.FM.clinical_gdc_current
- isb-cgc-bq.GENIE.clinical_gdc_current
- isb-cgc-bq.HCMI.clinical_gdc_current
- isb-cgc-bq.HCMI.clinical_diagnoses_gdc_current
- isb-cgc-bq.HCMI.clinical_diagnoses_treatments_gdc_current
- isb-cgc-bq.HCMI.clinical_follow_ups_gdc_current
- isb-cgc-bq.HCMI.clinical_follow_ups_molecular_tests_gdc_current
- isb-cgc-bq.MMRF.clinical_gdc_current
- isb-cgc-bq.NCICCR.clinical_gdc_current
- isb-cgc-bq.OHSU.clinical_gdc_current
- isb-cgc-bq.ORGANOID.clinical_gdc_current
- isb-cgc-bq.TARGET.clinical_gdc_current
- isb-cgc-bq.TCGA.clinical_gdc_current
- isb-cgc-bq.VAREPOP.clinical_gdc_current
- isb-cgc-bq.WCDT.clinical_gdc_current

February 2, 2022

New tables for Synthetic Lethality.

BigQuery tables created

- isb-cgc-bq.annotations.gene_info_human_NCBI_current
- isb-cgc-bq.annotations.gene2ensembl_human_NCBI_current
- isb-cgc-bq.annotations.gene2refseq_human_NCBI_current
- isb-cgc-bq.annotations.Human2Yeast_mapping_Alliance_for_Genome_Resources_current
- isb-cgc-bq.annotations.Yeast2Human_mapping_Alliance_for_Genome_Resources_current

- isb-cgc-bq.annotations_versioned.gene_info_human_NCBI_2020_07
- isb-cgc-bq.annotations_versioned.gene2ensembl_human_NCBI_2020_07
- isb-cgc-bq.annotations_versioned.gene2refseq_human_NCBI_2020_07
- isb-cgc-bq.annotations_versioned.Human2Yeast_mapping_Alliance_for_Genome_Resources_R3_0_1
- isb-cgc-bq.annotations_versioned.Yeast2Human_mapping_Alliance_for_Genome_Resources_R3_0_1
- isb-cgc-bq.DEPMAP.Achilles_gene_effect_DepMapPublic_current
- isb-cgc-bq.DEPMAP.CCLE_gene_cn_DepMapPublic_current
- isb-cgc-bq.DEPMAP.CCLE_gene_expression_DepMapPublic_current
- isb-cgc-bq.DEPMAP.CCLE_mutation_DepMapPublic_current
- isb-cgc-bq.DEPMAP.CCLE_SomaticMutation_DEMETER2_current
- isb-cgc-bq.DEPMAP.Combined_gene_dep_score_DEMETER2_current
- isb-cgc-bq.DEPMAP.RNAseq_IRPKM_DEMETER2_current
- isb-cgc-bq.DEPMAP.Sample_Info_DEMETER2_current
- isb-cgc-bq.DEPMAP.sample_info_DepMapPublic_current
- isb-cgc-bq.DEPMAP.WES_SNP_CN_DEMETER2_current
- isb-cgc-bq.DEPMAP_versioned.Achilles_gene_effect_DepMapPublic_20Q3
- isb-cgc-bq.DEPMAP_versioned.CCLE_gene_cn_DepMapPublic_20Q3
- isb-cgc-bq.DEPMAP_versioned.CCLE_gene_expression_DepMapPublic_20Q3
- isb-cgc-bq.DEPMAP_versioned.CCLE_mutation_DepMapPublic_20Q3
- isb-cgc-bq.DEPMAP_versioned.CCLE_SomaticMutation_DEMETER2_v6
- isb-cgc-bq.DEPMAP_versioned.Combined_gene_dep_score_DEMETER2_v6
- isb-cgc-bq.DEPMAP_versioned.RNAseq_IRPKM_DEMETER2_v6
- isb-cgc-bq.DEPMAP_versioned.Sample_Info_DEMETER2_v6
- isb-cgc-bq.DEPMAP_versioned.sample_info_DepMapPublic_20Q3
- isb-cgc-bq.DEPMAP_versioned.WES_SNP_CN_DEMETER2_v6
- isb-cgc-bq.supplementary_tables.Bailey_etal_Cell_2018_cancer_driver_genes
- isb-cgc-bq.supplementary_tables.Constanzo_etal_Science_2016_SGA_Genetic_Interactions
- isb-cgc-bq.synthetic_lethality.gene_info_human_HGNC_NCBI_2020_07
- isb-cgc-bq.synthetic_lethality.sample_info_TCGAlabels_DepMapPublic_20Q3

January 26, 2022

New GENCODE annotation tables.

BigQuery tables created

- isb-cgc-bq.GENCODE_versioned.annotation_gtf_hg38_v39

BigQuery tables updated

- isb-cgc-bq.GENCODE.annotation_gtf_hg38_current

January 13, 2022

New TCGA Radiology Images tables.

BigQuery tables created

- isb-cgc-bq.TCGA_versioned.radiology_images_tcia_2022_01

BigQuery tables updated

- isb-cgc-bq.TCGA.radiology_images_tcia_current

December 7, 2021

New per sample file metadata added to isb-cgc-bq for GDC release 30.

BigQuery tables created

- isb-cgc-bq.GENIE_versioned.per_sample_file_metadata_hg38_gdc_r30
- isb-cgc-bq.CTSP_versioned.per_sample_file_metadata_hg38_gdc_r30
- isb-cgc-bq.CGCI_versioned.per_sample_file_metadata_hg38_gdc_r30
- isb-cgc-bq.HCMI_versioned.per_sample_file_metadata_hg38_gdc_r30
- isb-cgc-bq.CPTAC_versioned.per_sample_file_metadata_hg38_gdc_r30
- isb-cgc-bq.TCGA_versioned.per_sample_file_metadata_hg38_gdc_r30
- isb-cgc-bq.TARGET_versioned.per_sample_file_metadata_hg38_gdc_r30
- isb-cgc-bq.REBC.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.REBC_versioned.per_sample_file_metadata_hg38_gdc_r30
- isb-cgc-bq.TRIO.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.TRIO_versioned.per_sample_file_metadata_hg38_gdc_r30

Current per sample file metadata tables updated to GDC release 30.

BigQuery tables updated

- isb-cgc-bq.GENIE_versioned.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.CTSP_versioned.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.CGCI_versioned.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.HCMI_versioned.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.CPTAC_versioned.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.TARGET_versioned.per_sample_file_metadata_hg38_gdc_current

New Datasets REBC, REBC_versioned, TRIO, and TRIO_versioned were created.

November 3, 2021 and December 3, 2021

New file metadata tables added to isb-cgc-bq for GDC release 30.

BigQuery tables created

- isb-cgc-bq.GDC_case_file_metadata_versioned.GDCfileID_to_GCSurl_r30
- isb-cgc-bq.GDC_case_file_metadata_versioned.fileData_legacy_r30
- isb-cgc-bq.GDC_case_file_metadata_versioned.fileData_active_r30

- isb-cgc-bq.GDC_case_file_metadata_versioned.caseData_r30
- isb-cgc-bq.GDC_case_file_metadata_versioned.aliquot2caseIDmap_r30
- isb-cgc-bq.GDC_case_file_metadata_versioned.slide2caseIDmap_r30

BigQuery tables updated

- isb-cgc-bq.GDC_case_file_metadata.fileData_legacy_current
- isb-cgc-bq.GDC_case_file_metadata.fileData_active_current
- isb-cgc-bq.GDC_case_file_metadata.caseData_current
- isb-cgc-bq.GDC_case_file_metadata.aliquot2caseIDmap_current
- isb-cgc-bq.GDC_case_file_metadata.slide2caseIDmap_current

October 19, 2021

New Pan-Cancer Atlas Clinical and Survival Data

BigQuery table created

- isb-cgc-bq.pancancer_atlas.TCGA_Clinical_Data_Resource_Extra

October 1, 2021

New Targetome datasets and tables added to isb-cgc-bq.

BigQuery tables created

- isb-cgc-bq.targetome.drug_synonyms_current
- isb-cgc-bq.targetome.experiments_current
- isb-cgc-bq.targetome.interactions_current
- isb-cgc-bq.targetome.sources_current
- isb-cgc-bq.targetome.target_synonyms_current
- isb-cgc-bq.targetome_versioned.drug_synonyms_v1
- isb-cgc-bq.targetome_versioned.experiments_v1
- isb-cgc-bq.targetome_versioned.interactions_v1
- isb-cgc-bq.targetome_versioned.sources_v1
- isb-cgc-bq.targetome_versioned.target_synonyms_v1

September 22, 2021

BigQuery tables created

New Copy Number Segment tables added to isb-cgc-bq.

- isb-cgc-bq.CGCI.copy_number_segment_hg38_gdc_current
- isb-cgc-bq.CGCI_versioned.copy_number_segment_hg38_gdc_r27
- isb-cgc-bq.CPTAC.copy_number_segment_hg38_gdc_current
- isb-cgc-bq.CPTAC_versioned.copy_number_segment_hg38_gdc_r28
- isb-cgc-bq.HCMI.copy_number_segment_hg38_gdc_current
- isb-cgc-bq.HCMI_versioned.copy_number_segment_hg38_gdc_r29
- isb-cgc-bq.TARGET.copy_number_segment_allelic_hg38_gdc_current

- isb-cgc-bq.TARGET_versioned.copy_number_segment_allelic_hg38_gdc_r23
- isb-cgc-bq.TCGA.copy_number_segment_allelic_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.copy_number_segment_allelic_hg38_gdc_r23

September 3, 2021

BigQuery tables created

- isb-cgc-bq.CPTAC_versioned.masked_somatic_mutation_hg38_gdc_r28

BigQuery tables updated

- isb-cgc-bq.CPTAC.masked_somatic_mutation_hg38_gdc_current

September 1, 2021

New Reactome datasets and tables added to isb-cgc-bq.

BigQuery tables created

- isb-cgc-bq.reactome.pathway_current
- isb-cgc-bq.reactome.physical_entity_current
- isb-cgc-bq.reactome.pe_to_pathway_current
- isb-cgc-bq.reactome.pathway_hierarchy_current
- isb-cgc-bq.reactome_versioned.pathway_v77
- isb-cgc-bq.reactome_versioned.physical_entity_v77
- isb-cgc-bq.reactome_versioned.pe_to_pathway_v77
- isb-cgc-bq.reactome_versioned.pathway_hierarchy_v77

Added release 28 miRNAseq isoform table and RNAseq for TCGA

BigQuery tables created

- isb-cgc-bq.TCGA_versioned.miRNAseq_isoform_hg38_gdc_r28
- isb-cgc-bq.TCGA_versioned.miRNAseq_hg38_gdc_r28
- isb-cgc-bq.TCGA_versioned.RNAseq_hg38_gdc_r28

BigQuery tables updated

- isb-cgc-bq.TCGA.miRNAseq_isoform_hg38_gdc_current
- isb-cgc-bq.TCGA.miRNAseq_hg38_gdc_current
- isb-cgc-bq.TCGA.RNAseq_hg38_gdc_current

August 2, 2021

New study, case metadata, file metadata, clinical, project-level per-sample file, and protein abundance log2ratio (quant) tables added to isb-cgc-bq for PDC V1.21.

BigQuery tables created

- isb-cgc-bq.CBTTC_versioned.quant_phosphoproteome_pediatric_brain_cancer_pilot_study_pdc_V1_21
- isb-cgc-bq.CBTTC_versioned.quant_proteome_pediatric_brain_cancer_pilot_study_pdc_V1_21
- isb-cgc-bq.CPTAC_versioned.clinical_CPTAC3_other_pdc_V1_21
- isb-cgc-bq.CPTAC_versioned.clinical_proteogenomic_translational_research_centers_pdc_V1_21

- isb-cgc-bq.CPTAC_versioned.per_sample_file_metadata_CPTAC2_other_pdc_V1_21
- isb-cgc-bq.CPTAC_versioned.per_sample_file_metadata_CPTAC3_other_pdc_V1_21
- isb-cgc-bq.CPTAC_versioned.per_sample_file_metadata_proteogenomic_translational_research_centers_pdc_V1_21
- isb-cgc-bq.CPTAC_versioned.quant_acetylome_CPTAC_GBM_discovery_study_pdc_V1_21
- isb-cgc-bq.CPTAC_versioned.quant_acetylome_CPTAC_LUAD_discovery_study_pdc_V1_21
- isb-cgc-bq.CPTAC_versioned.quant_acetylome_CPTAC_UCEC_discovery_study_pdc_V1_21
- isb-cgc-bq.CPTAC_versioned.quant_acetylome_prospective_breast_BI_pdc_V1_21
- isb-cgc-bq.CPTAC_versioned.quant_glycoproteome_prospective_ovarian_JHU_N_linked_glycosite_containing_peptide_pdc_V1
- isb-cgc-bq.CPTAC_versioned.quant_phosphoproteome_CPTAC_CCRCC_discovery_study_pdc_V1_21
- isb-cgc-bq.CPTAC_versioned.quant_phosphoproteome_CPTAC_GBM_discovery_study_pdc_V1_21
- isb-cgc-bq.CPTAC_versioned.quant_phosphoproteome_CPTAC_HNSCC_discovery_study_pdc_V1_21
- isb-cgc-bq.CPTAC_versioned.quant_phosphoproteome_CPTAC_LUAD_discovery_study_pdc_V1_21
- isb-cgc-bq.CPTAC_versioned.quant_phosphoproteome_CPTAC_UCEC_discovery_study_pdc_V1_21
- isb-cgc-bq.CPTAC_versioned.quant_phosphoproteome_prospective_breast_BI_pdc_V1_21
- isb-cgc-bq.CPTAC_versioned.quant_phosphoproteome_prospective_colon_PNNL_lumos_pdc_V1_21
- isb-cgc-bq.CPTAC_versioned.quant_phosphoproteome_prospective_ovarian_PNNL_lumos_pdc_V1_21
- isb-cgc-bq.CPTAC_versioned.quant_proteome_CPTAC_CCRCC_discovery_study_pdc_V1_21
- isb-cgc-bq.CPTAC_versioned.quant_proteome_CPTAC_GBM_discovery_study_pdc_V1_21
- isb-cgc-bq.CPTAC_versioned.quant_proteome_CPTAC_HNSCC_discovery_study_pdc_V1_21
- isb-cgc-bq.CPTAC_versioned.quant_proteome_CPTAC_LUAD_discovery_study_pdc_V1_21
- isb-cgc-bq.CPTAC_versioned.quant_proteome_CPTAC_UCEC_discovery_study_pdc_V1_21
- isb-cgc-bq.CPTAC_versioned.quant_proteome_prospective_breast_BI_pdc_V1_21
- isb-cgc-bq.CPTAC_versioned.quant_proteome_prospective_colon_PNNL_qeplus_pdc_V1_21
- isb-cgc-bq.CPTAC_versioned.quant_proteome_prospective_ovarian_JHU_pdc_V1_21
- isb-cgc-bq.CPTAC_versioned.quant_proteome_prospective_ovarian_PNNL_qeplus_pdc_V1_21
- isb-cgc-bq.ICPC_versioned.quant_phosphoproteome_HBV_related_hepatocellular_carcinoma_pdc_V1_21
- isb-cgc-bq.ICPC_versioned.quant_phosphoproteome_proteogenomics_of_gastric_cancer_pdc_V1_21
- isb-cgc-bq.ICPC_versioned.quant_proteome_HBV_related_hepatocellular_carcinoma_pdc_V1_21
- isb-cgc-bq.ICPC_versioned.quant_proteome_proteogenomics_of_gastric_cancer_pdc_V1_21
- isb-cgc-bq.PDC_metadata_versioned.aliquot_to_case_mapping_V1_21
- isb-cgc-bq.PDC_metadata_versioned.case_metadata_V1_21
- isb-cgc-bq.PDC_metadata_versioned.file_associated_entity_mapping_V1_21
- isb-cgc-bq.PDC_metadata_versioned.file_metadata_V1_21
- isb-cgc-bq.PDC_metadata_versioned.gene_info_V1_21
- isb-cgc-bq.PDC_metadata_versioned.studies_V1_21
- isb-cgc-bq.TCGA_versioned.clinical_CPTAC_TCGA_pdc_V1_21

- isb-cgc-bq.TCGA_versioned.quant_phosphoproteome_TCGA_breast_cancer_pdc_V1_21
- isb-cgc-bq.TCGA_versioned.quant_phosphoproteome_TCGA_ovarian_PNNL_velos_qexactive_pdc_V1_21
- isb-cgc-bq.TCGA_versioned.quant_proteome_TCGA_breast_cancer_pdc_V1_21
- isb-cgc-bq.TCGA_versioned.quant_proteome_TCGA_ovarian_JHU_pdc_V1_21
- isb-cgc-bq.TCGA_versioned.quant_proteome_TCGA_ovarian_PNNL_pdc_V1_21

BigQuery tables updated

- isb-cgc-bq.CBTTC.quant_phosphoproteome_pediatric_brain_cancer_pilot_study_pdc_current
- isb-cgc-bq.CBTTC.quant_proteome_pediatric_brain_cancer_pilot_study_pdc_current
- isb-cgc-bq.CPTAC.clinical_CPTAC3_other_pdc_current
- isb-cgc-bq.CPTAC.clinical_proteogenomic_translational_research_centers_pdc_current
- isb-cgc-bq.CPTAC.per_sample_file_metadata_CPTAC2_other_pdc_current
- isb-cgc-bq.CPTAC.per_sample_file_metadata_CPTAC3_other_pdc_current
- isb-cgc-bq.CPTAC.per_sample_file_metadata_proteogenomic_translational_research_centers_pdc_current
- isb-cgc-bq.CPTAC.quant_acetylome_CPTAC_GBM_discovery_study_pdc_current
- isb-cgc-bq.CPTAC.quant_acetylome_CPTAC_LUAD_discovery_study_pdc_current
- isb-cgc-bq.CPTAC.quant_acetylome_CPTAC_UCEC_discovery_study_pdc_current
- isb-cgc-bq.CPTAC.quant_acetylome_prospective_breast_BI_pdc_current
- isb-cgc-bq.CPTAC.quant_glycoproteome_prospective_ovarian_JHU_N_linked_glycosite_containing_peptide_pdc_current
- isb-cgc-bq.CPTAC.quant_phosphoproteome_CPTAC_CCRCC_discovery_study_pdc_current
- isb-cgc-bq.CPTAC.quant_phosphoproteome_CPTAC_GBM_discovery_study_pdc_current
- isb-cgc-bq.CPTAC.quant_phosphoproteome_CPTAC_HNSCC_discovery_study_pdc_current
- isb-cgc-bq.CPTAC.quant_phosphoproteome_CPTAC_LUAD_discovery_study_pdc_current
- isb-cgc-bq.CPTAC.quant_phosphoproteome_CPTAC_UCEC_discovery_study_pdc_current
- isb-cgc-bq.CPTAC.quant_phosphoproteome_prospective_breast_BI_pdc_current
- isb-cgc-bq.CPTAC.quant_phosphoproteome_prospective_colon_PNNL_lumos_pdc_current
- isb-cgc-bq.CPTAC.quant_phosphoproteome_prospective_ovarian_PNNL_lumos_pdc_current
- isb-cgc-bq.CPTAC.quant_proteome_CPTAC_CCRCC_discovery_study_pdc_current
- isb-cgc-bq.CPTAC.quant_proteome_CPTAC_GBM_discovery_study_pdc_current
- isb-cgc-bq.CPTAC.quant_proteome_CPTAC_HNSCC_discovery_study_pdc_current
- isb-cgc-bq.CPTAC.quant_proteome_CPTAC_LUAD_discovery_study_pdc_current
- isb-cgc-bq.CPTAC.quant_proteome_CPTAC_UCEC_discovery_study_pdc_current
- isb-cgc-bq.CPTAC.quant_proteome_prospective_breast_BI_pdc_current
- isb-cgc-bq.CPTAC.quant_proteome_prospective_colon_PNNL_qeplus_pdc_current
- isb-cgc-bq.CPTAC.quant_proteome_prospective_ovarian_JHU_pdc_current
- isb-cgc-bq.CPTAC.quant_proteome_prospective_ovarian_PNNL_qeplus_pdc_current
- isb-cgc-bq.ICPC.quant_phosphoproteome_HBV_related_hepatocellular_carcinoma_pdc_current

- isb-cgc-bq.ICPC.quant_phosphoproteome_proteogenomics_of_gastric_cancer_pdc_current
- isb-cgc-bq.ICPC.quant_proteome_HBV_related_hepatocellular_carcinoma_pdc_current
- isb-cgc-bq.ICPC.quant_proteome_proteogenomics_of_gastric_cancer_pdc_current
- isb-cgc-bq.PDC_metadata.aliquot_to_case_mapping_current
- isb-cgc-bq.PDC_metadata.case_metadata_current
- isb-cgc-bq.PDC_metadata.file_associated_entity_mapping_current
- isb-cgc-bq.PDC_metadata.file_metadata_current
- isb-cgc-bq.PDC_metadata.gene_info_current
- isb-cgc-bq.PDC_metadata.studies_current
- isb-cgc-bq.TCGA.clinical_CPTAC_TCGA_pdc_current
- isb-cgc-bq.TCGA.quant_phosphoproteome_TCGA_breast_cancer_pdc_current
- isb-cgc-bq.TCGA.quant_phosphoproteome_TCGA_ovarian_PNNL_velos_qexactive_pdc_current
- isb-cgc-bq.TCGA.quant_proteome_TCGA_breast_cancer_pdc_current
- isb-cgc-bq.TCGA.quant_proteome_TCGA_ovarian_JHU_pdc_current
- isb-cgc-bq.TCGA.quant_proteome_TCGA_ovarian_PNNL_pdc_current

July 14, 2021

Added release 28 miRNAseq isoform table for CPTAC

BigQuery tables created

- isb-cgc-bq.CPTAC_versioned.miRNAseq_isoform_hg38_gdc_r28
- isb-cgc-bq.CPTAC.miRNAseq_isoform_hg38_gdc_current

June 21, 2021

Updated the release 28 CPTAC miRNAseq tables to include the sample_type_name field

BigQuery tables created

- isb-cgc-bq.CPTAC_versioned.miRNAseq_hg38_gdc_r28_v2

BigQuery tables updated

- isb-cgc-bq.CPTAC.miRNAseq_hg38_gdc_current

June 18, 2021

New study, case metadata, file metadata, clinical, project-level per-sample file, and protein abundance log2ratio (quant) tables added to isb-cgc-bq for PDC V1.19.

BigQuery tables created

- isb-cgc-bq.CBTTC_versioned.quant_proteome_pediatric_brain_cancer_pilot_study_pdc_V1_19
- isb-cgc-bq.CPTAC_versioned.clinical_CPTAC2_other_pdc_V1_19
- isb-cgc-bq.CPTAC_versioned.clinical_CPTAC3_other_pdc_V1_19
- isb-cgc-bq.CPTAC_versioned.quant_glycoproteome_prospective_ovarian_JHU_N_linked_glycosite_containing_peptide_pdc_V1
- isb-cgc-bq.CPTAC_versioned.quant_proteome_CPTAC_CCRCC_discovery_study_pdc_V1_19
- isb-cgc-bq.CPTAC_versioned.quant_proteome_CPTAC_GBM_discovery_study_pdc_V1_19

- isb-cgc-bq.CPTAC_versioned.quant_proteome_CPTAC_HNSCC_discovery_study_pdc_V1_19
- isb-cgc-bq.CPTAC_versioned.quant_proteome_CPTAC_LUAD_discovery_study_pdc_V1_19
- isb-cgc-bq.CPTAC_versioned.quant_proteome_CPTAC_UCEC_discovery_study_pdc_V1_19
- isb-cgc-bq.CPTAC_versioned.quant_proteome_prospective_breast_BI_pdc_V1_19
- isb-cgc-bq.CPTAC_versioned.quant_proteome_prospective_colon_PNNL_qeplus_pdc_V1_19
- isb-cgc-bq.CPTAC_versioned.quant_proteome_prospective_ovarian_JHU_pdc_V1_19
- isb-cgc-bq.CPTAC_versioned.quant_proteome_prospective_ovarian_PNNL_qeplus_pdc_V1_19
- isb-cgc-bq.ICPC_versioned.quant_proteome_HBV_related_hepatocellular_carcinoma_pdc_V1_19
- isb-cgc-bq.ICPC_versioned.quant_proteome_proteogenomics_of_gastric_cancer_pdc_V1_19
- isb-cgc-bq.PDC_metadata_versioned.aliquot_to_case_mapping_V1_19
- isb-cgc-bq.PDC_metadata_versioned.case_metadata_V1_19
- isb-cgc-bq.PDC_metadata_versioned.file_associated_entity_mapping_V1_19
- isb-cgc-bq.PDC_metadata_versioned.file_metadata_V1_19
- isb-cgc-bq.PDC_metadata_versioned.gene_info_V1_19
- isb-cgc-bq.PDC_metadata_versioned.refseq_mapping_2021_03
- isb-cgc-bq.PDC_metadata_versioned.studies_V1_19
- isb-cgc-bq.TCGA_versioned.quant_proteome_TCGA_breast_cancer_pdc_V1_19
- isb-cgc-bq.TCGA_versioned.quant_proteome_TCGA_ovarian_JHU_pdc_V1_19
- isb-cgc-bq.TCGA_versioned.quant_proteome_TCGA_ovarian_PNNL_pdc_V1_19

BigQuery tables updated

- isb-cgc-bq.CBTTC.quant_proteome_pediatric_brain_cancer_pilot_study_pdc_current
- isb-cgc-bq.CPTAC.clinical_CPTAC2_other_pdc_current
- isb-cgc-bq.CPTAC.clinical_CPTAC3_other_pdc_current
- isb-cgc-bq.CPTAC.quant_glycoproteome_prospective_ovarian_JHU_N_linked_glycosite_containing_peptide_pdc_current
- isb-cgc-bq.CPTAC.quant_proteome_CPTAC_CCRCC_discovery_study_pdc_current
- isb-cgc-bq.CPTAC.quant_proteome_CPTAC_GBM_discovery_study_pdc_current
- isb-cgc-bq.CPTAC.quant_proteome_CPTAC_HNSCC_discovery_study_pdc_current
- isb-cgc-bq.CPTAC.quant_proteome_CPTAC_LUAD_discovery_study_pdc_current
- isb-cgc-bq.CPTAC.quant_proteome_CPTAC_UCEC_discovery_study_pdc_current
- isb-cgc-bq.CPTAC.quant_proteome_prospective_breast_BI_pdc_current
- isb-cgc-bq.CPTAC.quant_proteome_prospective_colon_PNNL_qeplus_pdc_current
- isb-cgc-bq.CPTAC.quant_proteome_prospective_ovarian_JHU_pdc_current
- isb-cgc-bq.CPTAC.quant_proteome_prospective_ovarian_PNNL_qeplus_pdc_current
- isb-cgc-bq.ICPC.quant_proteome_HBV_related_hepatocellular_carcinoma_pdc_current
- isb-cgc-bq.ICPC.quant_proteome_proteogenomics_of_gastric_cancer_pdc_current
- isb-cgc-bq.PDC_metadata.aliquot_to_case_mapping_current

- isb-cgc-bq.PDC_metadata.case_metadata_current
- isb-cgc-bq.PDC_metadata.file_associated_entity_mapping_current
- isb-cgc-bq.PDC_metadata.file_metadata_current
- isb-cgc-bq.PDC_metadata.gene_info_current
- isb-cgc-bq.PDC_metadata.refseq_mapping_current
- isb-cgc-bq.PDC_metadata.studies_current
- isb-cgc-bq.TCGA.quant_proteome_TCGA_breast_cancer_pdc_current
- isb-cgc-bq.TCGA.quant_proteome_TCGA_ovarian_JHU_pdc_current
- isb-cgc-bq.TCGA.quant_proteome_TCGA_ovarian_PNNL_pdc_current

June 10, 2021

New study and project-level per sample file metadata tables added to isb-cgc-bq for PDC V1.17.

BigQuery tables created

- isb-cgc-bq.PDC_metadata_versioned.studies_V1_17
- isb-cgc-bq.CBTTC_versioned.per_sample_file_metadata_pediatric_brain_cancer_pdc_V1_17
- isb-cgc-bq.CPTAC_versioned.per_sample_file_metadata_CPTAC_2_pdc_V1_17
- isb-cgc-bq.CPTAC_versioned.per_sample_file_metadata_CPTAC2_other_pdc_V1_17
- isb-cgc-bq.CPTAC_versioned.per_sample_file_metadata_CPTAC3_discovery_pdc_V1_17
- isb-cgc-bq.CPTAC_versioned.per_sample_file_metadata_CPTAC3_other_pdc_V1_17
- isb-cgc-bq.GPRP_versioned.per_sample_file_metadata_georgetown_lung_cancer_pdc_V1_17
- isb-cgc-bq.ICPC_versioned.per_sample_file_metadata_academia_sinica_LUAD_100_pdc_V1_17
- isb-cgc-bq.ICPC_versioned.per_sample_file_metadata_HBV_related_hepatocellular_carcinoma_pdc_V1_17
- isb-cgc-bq.ICPC_versioned.per_sample_file_metadata_human_early_onset_gastric_cancer_pdc_V1_17
- isb-cgc-bq.ICPC_versioned.per_sample_file_metadata_oral_squamous_cell_carcinoma_pdc_V1_17
- isb-cgc-bq.Quant_Maps_Tissue_Biopsies_versioned.per_sample_file_metadata_pct_swath_kidney_pdc_V1_17
- isb-cgc-bq.TCGA_versioned.per_sample_file_metadata_CPTAC_TCGA_pdc_V1_17
- isb-cgc-bq.PDC_metadata.studies_current
- isb-cgc-bq.CBTTC.per_sample_file_metadata_pediatric_brain_cancer_pdc_current
- isb-cgc-bq.CPTAC.per_sample_file_metadata_CPTAC_2_pdc_current
- isb-cgc-bq.CPTAC.per_sample_file_metadata_CPTAC2_other_pdc_current
- isb-cgc-bq.CPTAC.per_sample_file_metadata_CPTAC3_discovery_pdc_current
- isb-cgc-bq.CPTAC.per_sample_file_metadata_CPTAC3_other_pdc_current
- isb-cgc-bq.GPRP.per_sample_file_metadata_georgetown_lung_cancer_pdc_current
- isb-cgc-bq.ICPC.per_sample_file_metadata_academia_sinica_LUAD_100_pdc_current
- isb-cgc-bq.ICPC.per_sample_file_metadata_HBV_related_hepatocellular_carcinoma_pdc_current
- isb-cgc-bq.ICPC.per_sample_file_metadata_human_early_onset_gastric_cancer_pdc_current
- isb-cgc-bq.ICPC.per_sample_file_metadata_oral_squamous_cell_carcinoma_pdc_current

- isb-cgc-bq.Quant_Maps_Tissue_Biopsies.per_sample_file_metadata_pct_swath_kidney_pdc_current
- isb-cgc-bq.TCGA.per_sample_file_metadata_CPTAC_TCGA_pdc_current

May 28, 2021

New per sample file metadata added to isb-cgc-bq for GDC release 29.

BigQuery tables created

- isb-cgc-bq.CMI_versioned.per_sample_file_metadata_hg38_gdc_r29
- isb-cgc-bq.CGCI_versioned.per_sample_file_metadata_hg38_gdc_r29
- isb-cgc-bq.HCMI_versioned.per_sample_file_metadata_hg38_gdc_r29
- isb-cgc-bq.CPTAC_versioned.per_sample_file_metadata_hg38_gdc_r29
- isb-cgc-bq.TCGA_versioned.per_sample_file_metadata_hg38_gdc_r29
- isb-cgc-bq.TCGA_versioned.per_sample_file_metadata_hg19_gdc_r29

Current per sample file metadata tables updated to GDC release 29.

BigQuery tables updated

- isb-cgc-bq.CMI.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.CGCI.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.HCMI.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.CPTAC.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.TCGA.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.TCGA.per_sample_file_metadata_hg19_gdc_current

May 27, 2021

New controlled-access VCF tables.

BigQuery tables created

- isb-cgc-cbq.VAREPOP_versioned.vcf_hg38_gdc_r24
- isb-cgc-cbq.VAREPOP.vcf_hg38_gdc_current
- isb-cgc-cbq.TCGA_versioned.vcf_hg38_gdc_r24
- isb-cgc-cbq.TCGA.vcf_hg38_gdc_current
- isb-cgc-cbq.ORGANOID_versioned.vcf_hg38_gdc_r24
- isb-cgc-cbq.ORGANOID.vcf_hg38_gdc_current
- isb-cgc-cbq.MMRF_versioned.vcf_hg38_gdc_r24
- isb-cgc-cbq.MMRF.vcf_hg38_gdc_current
- isb-cgc-cbq.HCMI_versioned.vcf_hg38_gdc_r24
- isb-cgc-cbq.HCMI.vcf_hg38_gdc_current
- isb-cgc-cbq.FM_versioned.vcf_hg38_gdc_r24
- isb-cgc-cbq.FM.vcf_hg38_gdc_current

May 26, 2021

New case metadata, file metadata, clinical, and quant data (for actylome, glycoproteome, phosphoproteome, and proteome) added to isb-cgc-bq from PDC V1.17.

BigQuery tables created

- isb-cgc-bq.CBTTC_versioned.clinical_diagnoses_pediatric_brain_cancer_pdc_V1_17
- isb-cgc-bq.CBTTC_versioned.clinical_pediatric_brain_cancer_pdc_V1_17
- isb-cgc-bq.CBTTC_versioned.quant_phosphoproteome_pediatric_brain_cancer_pilot_study_pdc_V1_17
- isb-cgc-bq.CBTTC_versioned.quant_proteome_pediatric_brain_cancer_pilot_study_pdc_V1_17
- isb-cgc-bq.CPTAC_versioned.clinical_CPTAC_2_pdc_V1_17
- isb-cgc-bq.CPTAC_versioned.clinical_CPTAC2_other_pdc_V1_17
- isb-cgc-bq.CPTAC_versioned.clinical_CPTAC3_discovery_pdc_V1_17
- isb-cgc-bq.CPTAC_versioned.clinical_CPTAC3_other_pdc_V1_17
- isb-cgc-bq.CPTAC_versioned.quant_acetylome_CPTAC_GBM_discovery_study_pdc_V1_17
- isb-cgc-bq.CPTAC_versioned.quant_acetylome_CPTAC_LUAD_discovery_study_pdc_V1_17
- isb-cgc-bq.CPTAC_versioned.quant_acetylome_CPTAC_UCEC_discovery_study_pdc_V1_17
- isb-cgc-bq.CPTAC_versioned.quant_acetylome_prospective_breast_BI_pdc_V1_17
- isb-cgc-bq.CPTAC_versioned.quant_glycoproteome_prospective_ovarian_JHU_n_linked_glycosite_containing_peptide_pdc_V1
- isb-cgc-bq.CPTAC_versioned.quant_phosphoproteome_CPTAC_CCRCC_discovery_study_pdc_V1_17
- isb-cgc-bq.CPTAC_versioned.quant_phosphoproteome_CPTAC_GBM_discovery_study_pdc_V1_17
- isb-cgc-bq.CPTAC_versioned.quant_phosphoproteome_CPTAC_HNSCC_discovery_study_pdc_V1_17
- isb-cgc-bq.CPTAC_versioned.quant_phosphoproteome_CPTAC_LUAD_discovery_study_pdc_V1_17
- isb-cgc-bq.CPTAC_versioned.quant_phosphoproteome_CPTAC_UCEC_discovery_study_pdc_V1_17
- isb-cgc-bq.CPTAC_versioned.quant_phosphoproteome_prospective_breast_BI_pdc_V1_17
- isb-cgc-bq.CPTAC_versioned.quant_phosphoproteome_prospective_colon_PNNL_lumos_pdc_V1_17
- isb-cgc-bq.CPTAC_versioned.quant_phosphoproteome_prospective_ovarian_PNNL_lumos_pdc_V1_17
- isb-cgc-bq.CPTAC_versioned.quant_proteome_CPTAC_CCRCC_discovery_study_pdc_V1_17
- isb-cgc-bq.CPTAC_versioned.quant_proteome_CPTAC_GBM_discovery_study_pdc_V1_17
- isb-cgc-bq.CPTAC_versioned.quant_proteome_CPTAC_HNSCC_discovery_study_pdc_V1_17
- isb-cgc-bq.CPTAC_versioned.quant_proteome_CPTAC_LUAD_discovery_study_pdc_V1_17
- isb-cgc-bq.CPTAC_versioned.quant_proteome_CPTAC_UCEC_discovery_study_pdc_V1_17
- isb-cgc-bq.CPTAC_versioned.quant_proteome_prospective_breast_BI_pdc_V1_17
- isb-cgc-bq.CPTAC_versioned.quant_proteome_prospective_colon_PNNL_qeplus_pdc_V1_17
- isb-cgc-bq.CPTAC_versioned.quant_proteome_prospective_ovarian_JHU_pdc_V1_17
- isb-cgc-bq.CPTAC_versioned.quant_proteome_prospective_ovarian_PNNL_qeplus_pdc_V1_17
- isb-cgc-bq.GPRP_versioned.clinical_georgetown_lung_cancer_pdc_V1_17
- isb-cgc-bq.ICPC_versioned.clinical_academia_sinica_LUAD_100_pdc_V1_17

- isb-cgc-bq.ICPC_versioned.clinical_HBV_related_hepatocellular_carcinoma_pdc_V1_17
- isb-cgc-bq.ICPC_versioned.clinical_human_early_onset_gastric_cancer_pdc_V1_17
- isb-cgc-bq.ICPC_versioned.clinical_oral_squamous_cell_carcinoma_pdc_V1_17
- isb-cgc-bq.ICPC_versioned.quant_phosphoproteome_HBV_related_hepatocellular_carcinoma_pdc_V1_17
- isb-cgc-bq.ICPC_versioned.quant_phosphoproteome_proteogenomics_of_gastric_cancer_pdc_V1_17
- isb-cgc-bq.ICPC_versioned.quant_proteome_HBV_related_hepatocellular_carcinoma_pdc_V1_17
- isb-cgc-bq.ICPC_versioned.quant_proteome_proteogenomics_of_gastric_cancer_pdc_V1_17
- isb-cgc-bq.PDC_metadata_versioned.aliquot_to_case_mapping_V1_17
- isb-cgc-bq.PDC_metadata_versioned.case_metadata_V1_17
- isb-cgc-bq.PDC_metadata_versioned.file_associated_entity_mapping_V1_17
- isb-cgc-bq.PDC_metadata_versioned.file_metadata_V1_17
- isb-cgc-bq.PDC_metadata_versioned.gene_info_V1_17
- isb-cgc-bq.PDC_metadata_versioned.refseq_mapping_2021_02
- isb-cgc-bq.Quant_Maps_Tissue_Biopsies_versioned.clinical_pct_swath_kidney_pdc_V1_17
- isb-cgc-bq.TCGA_versioned.clinical_CPTAC_TCGA_pdc_V1_17
- isb-cgc-bq.TCGA_versioned.quant_phosphoproteome_TCGA_breast_cancer_pdc_V1_17
- isb-cgc-bq.TCGA_versioned.quant_phosphoproteome_TCGA_ovarian_PNNL_velos_qexactive_pdc_V1_17
- isb-cgc-bq.TCGA_versioned.quant_proteome_TCGA_breast_cancer_pdc_V1_17
- isb-cgc-bq.TCGA_versioned.quant_proteome_TCGA_ovarian_JHU_pdc_V1_17
- isb-cgc-bq.TCGA_versioned.quant_proteome_TCGA_ovarian_PNNL_pdc_V1_17
- isb-cgc-bq.CBTTC.quant_phosphoproteome_pediatric_brain_cancer_pilot_study_pdc_current
- isb-cgc-bq.CPTAC.clinical_CPTAC2_other_pdc_current
- isb-cgc-bq.CPTAC.clinical_CPTAC3_other_pdc_current
- isb-cgc-bq.CPTAC.quant_acetylome_CPTAC_GBM_discovery_study_pdc_current
- isb-cgc-bq.CPTAC.quant_acetylome_CPTAC_LUAD_discovery_study_pdc_current
- isb-cgc-bq.CPTAC.quant_acetylome_CPTAC_UCEC_discovery_study_pdc_current
- isb-cgc-bq.CPTAC.quant_acetylome_prospective_breast_BI_pdc_current
- isb-cgc-bq.CPTAC.quant_glycoproteome_prospective_ovarian_JHU_n_linked_glycosite_containing_peptide_pdc_current
- isb-cgc-bq.CPTAC.quant_phosphoproteome_CPTAC_CCRCC_discovery_study_pdc_current
- isb-cgc-bq.CPTAC.quant_phosphoproteome_CPTAC_GBM_discovery_study_pdc_current
- isb-cgc-bq.CPTAC.quant_phosphoproteome_CPTAC_HNSCC_discovery_study_pdc_current
- isb-cgc-bq.CPTAC.quant_phosphoproteome_CPTAC_LUAD_discovery_study_pdc_current
- isb-cgc-bq.CPTAC.quant_phosphoproteome_CPTAC_UCEC_discovery_study_pdc_current
- isb-cgc-bq.CPTAC.quant_phosphoproteome_prospective_breast_BI_pdc_current
- isb-cgc-bq.CPTAC.quant_phosphoproteome_prospective_colon_PNNL_lumos_pdc_current
- isb-cgc-bq.CPTAC.quant_phosphoproteome_prospective_ovarian_PNNL_lumos_pdc_current

- isb-cgc-bq.CPTAC.quant_proteome_CPTAC_GBM_discovery_study_pdc_current
- isb-cgc-bq.CPTAC.quant_proteome_CPTAC_HNSCC_discovery_study_pdc_current
- isb-cgc-bq.ICPC.quant_phosphoproteome_HBV_related_hepatocellular_carcinoma_pdc_current
- isb-cgc-bq.ICPC.quant_phosphoproteome_proteogenomics_of_gastric_cancer_pdc_current
- isb-cgc-bq.PDC_metadata.gene_info_current
- isb-cgc-bq.PDC_metadata.refseq_mapping_current
- isb-cgc-bq.TCGA.quant_phosphoproteome_TCGA_breast_cancer_pdc_current
- isb-cgc-bq.TCGA.quant_phosphoproteome_TCGA_ovarian_PNNL_velos_qexactive_pdc_current

BigQuery tables updated

- isb-cgc-bq.CBTTC.clinical_diagnoses_pediatric_brain_cancer_pdc_current
- isb-cgc-bq.CBTTC.clinical_pediatric_brain_cancer_pdc_current
- isb-cgc-bq.CBTTC.quant_proteome_pediatric_brain_cancer_pilot_study_pdc_current
- isb-cgc-bq.CPTAC.clinical_CPTAC_2_pdc_current
- isb-cgc-bq.CPTAC.clinical_CPTAC3_discovery_pdc_current
- isb-cgc-bq.CPTAC.quant_proteome_CPTAC_CCRCC_discovery_study_pdc_current
- isb-cgc-bq.CPTAC.quant_proteome_CPTAC_LUAD_discovery_study_pdc_current
- isb-cgc-bq.CPTAC.quant_proteome_CPTAC_UCEC_discovery_study_pdc_current
- isb-cgc-bq.CPTAC.quant_proteome_prospective_breast_BI_pdc_current
- isb-cgc-bq.CPTAC.quant_proteome_prospective_colon_PNNL_qeplus_pdc_current
- isb-cgc-bq.CPTAC.quant_proteome_prospective_ovarian_JHU_pdc_current
- isb-cgc-bq.CPTAC.quant_proteome_prospective_ovarian_PNNL_qeplus_pdc_current
- isb-cgc-bq.GPRP.clinical_georgetown_lung_cancer_pdc_current
- isb-cgc-bq.ICPC.clinical_academia_sinica_LUAD_100_pdc_current
- isb-cgc-bq.ICPC.clinical_HBV_related_hepatocellular_carcinoma_pdc_current
- isb-cgc-bq.ICPC.clinical_human_early_onset_gastric_cancer_pdc_current
- isb-cgc-bq.ICPC.clinical_oral_squamous_cell_carcinoma_pdc_current
- isb-cgc-bq.ICPC.quant_proteome_HBV_related_hepatocellular_carcinoma_pdc_current
- isb-cgc-bq.ICPC.quant_proteome_proteogenomics_of_gastric_cancer_pdc_current
- isb-cgc-bq.PDC_metadata.aliquot_to_case_mapping_current
- isb-cgc-bq.PDC_metadata.case_metadata_current
- isb-cgc-bq.PDC_metadata.file_associated_entity_mapping_current
- isb-cgc-bq.PDC_metadata.file_metadata_current
- isb-cgc-bq.Quant_Maps_Tissue_Biopsies.clinical_pct_swath_kidney_pdc_current
- isb-cgc-bq.TCGA.clinical_CPTAC_TCGA_pdc_current
- isb-cgc-bq.TCGA.quant_proteome_TCGA_breast_cancer_pdc_current
- isb-cgc-bq.TCGA.quant_proteome_TCGA_ovarian_JHU_pdc_current

- isb-cgc-bq.TCGA.quant_proteome_TCGA_ovarian_PNNL_pdc_current

New CPTAC controlled-access VCF tables.

BigQuery tables created

- isb-cgc-cbq.CPTAC3_versioned.vcf_hg38_gdc_r24
- isb-cgc-cbq.CPTAC3.vcf_hg38_gdc_current
- isb-cgc-cbq.CPTAC2_versioned.vcf_hg38_gdc_r24
- isb-cgc-cbq.CPTAC2.vcf_hg38_gdc_current

May 24, 2021

New CPTAC RNA Seq table added to isb-cgc-bq for GDC release 28.

BigQuery tables created

- isb-cgc-bq.CPTAC_versioned.RNAseq_hg38_gdc_r28

BigQuery tables updated

- isb-cgc-bq.CPTAC.RNAseq_hg38_gdc_current

May 21, 2021

New clinical tables added to isb-cgc-bq for GDC release 29.

BigQuery tables created

- isb-cgc-bq.BEATAML1_0_versioned.clinical_gdc_r29
- isb-cgc-bq.CGCI_versioned.clinical_gdc_r29
- isb-cgc-bq.CGCI_versioned.clinical_diagnoses_gdc_r29
- isb-cgc-bq.CGCI_versioned.clinical_diagnoses_treatments_gdc_r29
- isb-cgc-bq.CGCI_versioned.clinical_follow_ups_gdc_r29
- isb-cgc-bq.CGCI_versioned.clinical_follow_ups_molecular_tests_gdc_r29
- isb-cgc-bq.CMI_versioned.clinical_gdc_r29
- isb-cgc-bq.CPTAC_versioned.clinical_gdc_r29
- isb-cgc-bq.CTSP_versioned.clinical_gdc_r29
- isb-cgc-bq.FM_versioned.clinical_gdc_r29
- isb-cgc-bq.GENIE_versioned.clinical_gdc_r29
- isb-cgc-bq.HCMI_versioned.clinical_gdc_r29
- isb-cgc-bq.HCMI_versioned.clinical_diagnoses_gdc_r29
- isb-cgc-bq.HCMI_versioned.clinical_diagnoses_treatments_gdc_r29
- isb-cgc-bq.HCMI_versioned.clinical_follow_ups_gdc_r29
- isb-cgc-bq.HCMI_versioned.clinical_follow_ups_molecular_tests_gdc_r29
- isb-cgc-bq.MMRF_versioned.clinical_gdc_r29
- isb-cgc-bq.MMRF_versioned.clinical_diagnoses_treatments_gdc_r29
- isb-cgc-bq.MMRF_versioned.clinical_family_histories_gdc_r29
- isb-cgc-bq.MMRF_versioned.clinical_follow_ups_gdc_r29

- isb-cgc-bq.MMRF_versioned.clinical_follow_ups_molecular_tests_gdc_r29
- isb-cgc-bq.NCICCR_versioned.clinical_gdc_r29
- isb-cgc-bq.OHSU_versioned.clinical_gdc_r29
- isb-cgc-bq.ORGANOID_versioned.clinical_gdc_r29
- isb-cgc-bq.TARGET_versioned.clinical_gdc_r29
- isb-cgc-bq.TCGA_versioned.clinical_gdc_r29
- isb-cgc-bq.TCGA_versioned.clinical_diagnoses_treatments_gdc_r29
- isb-cgc-bq.VAREPOP_versioned.clinical_gdc_r29
- isb-cgc-bq.VAREPOP_versioned.clinical_diagnoses_treatments_gdc_r29
- isb-cgc-bq.VAREPOP_versioned.clinical_family_histories_gdc_r29
- isb-cgc-bq.WCDT_versioned.clinical_gdc_r29

Current clinical tables updated to GDC release 29.

BigQuery tables updated

- isb-cgc-bq.BEATAML1_0.clinical_gdc_current
- isb-cgc-bq.CGCI.clinical_gdc_current
- isb-cgc-bq.CGCI.clinical_diagnoses_gdc_current
- isb-cgc-bq.CGCI.clinical_diagnoses_treatments_gdc_current
- isb-cgc-bq.CGCI.clinical_follow_ups_gdc_current
- isb-cgc-bq.CGCI.clinical_follow_ups_molecular_tests_gdc_current
- isb-cgc-bq.CMI.clinical_gdc_current
- isb-cgc-bq.CPTAC.clinical_gdc_current
- isb-cgc-bq.CTSP.clinical_gdc_current
- isb-cgc-bq.FM.clinical_gdc_current
- isb-cgc-bq.GENIE.clinical_gdc_current
- isb-cgc-bq.HCMI.clinical_gdc_current
- isb-cgc-bq.HCMI.clinical_diagnoses_gdc_current
- isb-cgc-bq.HCMI.clinical_diagnoses_treatments_gdc_current
- isb-cgc-bq.HCMI.clinical_follow_ups_gdc_current
- isb-cgc-bq.HCMI.clinical_follow_ups_molecular_tests_gdc_current
- isb-cgc-bq.MMRF.clinical_gdc_current
- isb-cgc-bq.MMRF.clinical_diagnoses_treatments_gdc_current
- isb-cgc-bq.MMRF.clinical_family_histories_gdc_current
- isb-cgc-bq.MMRF.clinical_follow_ups_gdc_current
- isb-cgc-bq.MMRF.clinical_follow_ups_molecular_tests_gdc_current
- isb-cgc-bq.NCICCR.clinical_gdc_current
- isb-cgc-bq.OHSU.clinical_gdc_current

- isb-cgc-bq.ORGANOID.clinical_gdc_current
- isb-cgc-bq.TARGET.clinical_gdc_current
- isb-cgc-bq.TCGA.clinical_gdc_current
- isb-cgc-bq.TCGA.clinical_diagnoses_treatments_gdc_current
- isb-cgc-bq.VAREPOP.clinical_gdc_current
- isb-cgc-bq.VAREPOP.clinical_diagnoses_treatments_gdc_current
- isb-cgc-bq.VAREPOP.clinical_family_histories_gdc_current
- isb-cgc-bq.WCDT.clinical_gdc_current

May 18, 2021

New file metadata tables added to isb-cgc-bq for GDC release 29 and New GENCODE annotation tables.

BigQuery tables created

- isb-cgc-bq.GDC_case_file_metadata_versioned.GDCfileID_to_GCSurl_r29
- isb-cgc-bq.GDC_case_file_metadata_versioned.fileData_legacy_r29
- isb-cgc-bq.GDC_case_file_metadata_versioned.fileData_active_r29
- isb-cgc-bq.GDC_case_file_metadata_versioned.caseData_r29
- isb-cgc-bq.GDC_case_file_metadata_versioned.aliquot2caseIDmap_r29
- isb-cgc-bq.GDC_case_file_metadata_versioned.slide2caseIDmap_r29
- isb-cgc-bq.GENCODE_versioned.annotation_gtf_hg38_v38

BigQuery tables updated

- isb-cgc-bq.GDC_case_file_metadata.GDCfileID_to_GCSurl_current
- isb-cgc-bq.GDC_case_file_metadata.fileData_legacy_current
- isb-cgc-bq.GDC_case_file_metadata.fileData_active_current
- isb-cgc-bq.GDC_case_file_metadata.caseData_current
- isb-cgc-bq.GDC_case_file_metadata.aliquot2caseIDmap_current
- isb-cgc-bq.GDC_case_file_metadata.slide2caseIDmap_current
- isb-cgc-bq.GENCODE.annotation_gtf_hg38_current

April 14, 2021

New PDC Aliquot and Case Metadata tables.

BigQuery tables created

- isb-cgc-bq.PDC_metadata.aliquot_to_case_mapping_pdc_current
- isb-cgc-bq.PDC_metadata_versioned.aliquot_to_case_mapping_pdc_V1_11
- isb-cgc-bq.PDC_metadata.case_metadata_pdc_current
- isb-cgc-bq.PDC_metadata_versioned.case_metadata_pdc_V1_11

April 2, 2021

New GENCODE annotation tables.

BigQuery tables created

- isb-cgc-bq.GENCODE_versioned.annotation_gtf_hg38_v36
- isb-cgc-bq.GENCODE_versioned.annotation_gtf_hg38_v37

BigQuery tables updated

- isb-cgc-bq.GENCODE.annotation_gtf_hg38_current

March 30, 2021

New CPTAC miRNA expression tables.

BigQuery tables created

- isb-cgc-bq.CPTAC.miRNAseq_hg38_gdc_current
- isb-cgc-bq.CPTAC_versioned.miRNAseq_hg38_gdc_r28

March 22, 2021

New TARGET miRNA isoform expression tables.

BigQuery tables created

- isb-cgc-bq.TARGET_versioned.miRNAseq_isoform_hg38_gdc_r25

BigQuery tables updated

- isb-cgc-bq.TARGET.miRNAseq_isoform_hg38_gdc_current

March 17, 2021

New HCMI RNA Seq table

BigQuery tables created

- isb-cgc-bq.HCMI_versioned.RNAseq_hg38_gdc_r28

BigQuery tables updated

- isb-cgc-bq.HCMI.RNAseq_hg38_gdc_current

March 11, 2021

New HCMI Masked Somatic Mutation table

BigQuery tables created

- isb-cgc-bq.HCMI_versioned.masked_somatic_mutation_hg38_gdc_r28

BigQuery tables updated

- isb-cgc-bq.HCMI.masked_somatic_mutation_hg38_gdc_current

March 5, 2021

New file metadata, per sample file metadata, and clinical tables added to isb-cgc-bq for GDC release 28.

BigQuery tables created

- isb-cgc-bq.CMI_versioned.per_sample_file_metadata_hg38_gdc_r28
- isb-cgc-bq.WCDT_versioned.per_sample_file_metadata_hg38_gdc_r28
- isb-cgc-bq.GENIE_versioned.per_sample_file_metadata_hg38_gdc_r28
- isb-cgc-bq.OHSU_versioned.per_sample_file_metadata_hg38_gdc_r28
- isb-cgc-bq.FM_versioned.per_sample_file_metadata_hg38_gdc_r28
- isb-cgc-bq.VAREPOP_versioned.per_sample_file_metadata_hg38_gdc_r28

- isb-cgc-bq.CTSP_versioned.per_sample_file_metadata_hg38_gdc_r28
- isb-cgc-bq.NCICCR_versioned.per_sample_file_metadata_hg38_gdc_r28
- isb-cgc-bq.ORGANOID_versioned.per_sample_file_metadata_hg38_gdc_r28
- isb-cgc-bq.MMRF_versioned.per_sample_file_metadata_hg38_gdc_r28
- isb-cgc-bq.CGCI_versioned.per_sample_file_metadata_hg38_gdc_r28
- isb-cgc-bq.HCMI_versioned.per_sample_file_metadata_hg38_gdc_r28
- isb-cgc-bq.BEATAML1_0_versioned.per_sample_file_metadata_hg38_gdc_r28
- isb-cgc-bq.CPTAC_versioned.per_sample_file_metadata_hg38_gdc_r28
- isb-cgc-bq.TARGET_versioned.per_sample_file_metadata_hg38_gdc_r28
- isb-cgc-bq.TCGA_versioned.per_sample_file_metadata_hg38_gdc_r28
- isb-cgc-bq.CCLE_versioned.per_sample_file_metadata_hg19_gdc_r28
- isb-cgc-bq.TARGET_versioned.per_sample_file_metadata_hg19_gdc_r28
- isb-cgc-bq.TCGA_versioned.per_sample_file_metadata_hg19_gdc_r28
- isb-cgc-bq.GDC_case_file_metadata_versioned.GDCfileID_to_GCSurl_r28
- isb-cgc-bq.GDC_case_file_metadata_versioned.fileData_legacy_r28
- isb-cgc-bq.GDC_case_file_metadata_versioned.fileData_active_r28
- isb-cgc-bq.GDC_case_file_metadata_versioned.caseData_r28
- isb-cgc-bq.GDC_case_file_metadata_versioned.aliquot2caseIDmap_r28
- isb-cgc-bq.GDC_case_file_metadata_versioned.slide2caseIDmap_r28
- isb-cgc-bq.HCMI_versioned.clinical_follow_ups_molecular_tests_gdc_r28
- isb-cgc-bq.HCMI_versioned.clinical_diagnoses_treatments_gdc_r28
- isb-cgc-bq.HCMI_versioned.clinical_diagnoses_gdc_r28
- isb-cgc-bq.CPTAC_versioned.clinical_gdc_r28
- isb-cgc-bq.HCMI_versioned.clinical_gdc_r28
- isb-cgc-bq.CMI_versioned.clinical_gdc_r28
- isb-cgc-bq.HCMI_versioned.clinical_follow_ups_gdc_r28

Current file metadata, per sample file metadata, and clinical tables updated to GDC release 28.

BigQuery tables updated

- isb-cgc-bq.CMI.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.WCDT.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.GENIE.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.OHSU.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.FM.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.VAREPOP.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.CTSP.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.NCICCR.per_sample_file_metadata_hg38_gdc_current

- isb-cgc-bq.ORGANOID.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.MMRF.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.CGCI.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.BEATAML1_0.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.CPTAC.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.TARGET.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.TCGA.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.CCLE.per_sample_file_metadata_hg19_gdc_current
- isb-cgc-bq.TARGET.per_sample_file_metadata_hg19_gdc_current
- isb-cgc-bq.TCGA.per_sample_file_metadata_hg19_gdc_current
- isb-cgc-bq.GDC_case_file_metadata.GDCfileID_to_GCSurl_current
- isb-cgc-bq.GDC_case_file_metadata.fileData_legacy_current
- isb-cgc-bq.GDC_case_file_metadata.fileData_active_current
- isb-cgc-bq.GDC_case_file_metadata.caseData_current
- isb-cgc-bq.GDC_case_file_metadata.aliquot2caseIDmap_current
- isb-cgc-bq.GDC_case_file_metadata.slide2caseIDmap_current
- isb-cgc-bq.HCMI.clinical_follow_ups_molecular_tests_gdc_current
- isb-cgc-bq.HCMI.clinical_diagnoses_treatments_gdc_current
- isb-cgc-bq.HCMI.clinical_diagnoses_gdc_current
- isb-cgc-bq.CPTAC.clinical_gdc_current
- isb-cgc-bq.HCMI.clinical_gdc_current
- isb-cgc-bq.CMI.clinical_gdc_current
- isb-cgc-bq.HCMI.clinical_follow_ups_gdc_current

March 3, 2021

PDC metadata

- isb-cgc-bq.PDC_metadata.file_associated_entity_mapping_current
- isb-cgc-bq.PDC_metadata_versioned.file_associated_entity_mapping_V1_9
- isb-cgc-bq.PDC_metadata.file_metadata_current
- isb-cgc-bq.PDC_metadata_versioned.file_metadata_V1_9

February 25, 2021

New TARGET miRNA-seq table

BigQuery tables created

- isb-cgc-bq.TARGET_versioned.miRNAseq_hg38_gdc_r25

BigQuery tables updated

- isb-cgc-bq.TARGET.miRNAseq_hg38_gdc_current

February 18, 2021

Pediatric Brain Cancer Pilot Study clinical data from PDC

- isb-cgc-bq.CBTTC.clinical_pediatric_brain_cancer_pdc_current
- isb-cgc-bq.CBTTC_versioned.clinical_pediatric_brain_cancer_pdc_V1_9
- isb-cgc-bq.CBTTC.clinical_diagnoses_pediatric_brain_cancer_pdc_current
- isb-cgc-bq.CBTTC_versioned.clinical_diagnoses_pediatric_brain_cancer_pdc_V1_9

Hepatitis B Virus (HBV) Related Hepatocellular Carcinoma clinical data from PDC

- isb-cgc-bq.ICPC.clinical_HBV_related_hepatocellular_carcinoma_pdc_current
- isb-cgc-bq.ICPC_versioned.clinical_HBV_related_hepatocellular_carcinoma_pdc_V1_9

Proteogenomics of Gastric Cancer Proteome clinical data from PDC

- isb-cgc-bq.ICPC.clinical_human_early_onset_gastric_cancer_pdc_current
- isb-cgc-bq.ICPC_versioned.clinical_human_early_onset_gastric_cancer_pdc_V1_9

Oral Squamous Cell Carcinoma clinical data from PDC

- isb-cgc-bq.ICPC.clinical_oral_squamous_cell_carcinoma_pdc_current
- isb-cgc-bq.ICPC_versioned.clinical_oral_squamous_cell_carcinoma_pdc_V1_9

Academia Sinica LUAD-100 clinical data from PDC

- isb-cgc-bq.ICPC.clinical_academia_sinica_LUAD_100_pdc_current
- isb-cgc-bq.ICPC_versioned.clinical_academia_sinica_LUAD_100_pdc_V1_9

Georgetown Lung Cancer Proteomics Study clinical data from PDC

- clinical_georgetown_lung_cancer_pdc_current
- clinical_georgetown_lung_cancer_pdc_V1_9

Quantitative digital maps of tissue biopsies clinical data from PDC

- isb-cgc-bq.Quant_Maps_Tissue_Biopsies.clinical_pct_swath_kidney_pdc_current
- isb-cgc-bq.Quant_Maps_Tissue_Biopsies_versioned.clinical_pct_swath_kidney_pdc_V1_9

CPTAC clincal data from PDC

- isb-cgc-bq.TCGA.clinical_CPTAC_TCGA_pdc_current
- isb-cgc-bq.TCGA_versioned.clinical_CPTAC_TCGA_pdc_V1_9
- isb-cgc-bq.CPTAC.clinical_CPTAC_2_pdc_current
- isb-cgc-bq.CPTAC_versioned.clinical_CPTAC_2_pdc_V1_9
- isb-cgc-bq.CPTAC.clinical_CPTAC3_discovery_pdc_current
- isb-cgc-bq.CPTAC_versioned.clinical_CPTAC3_discovery_pdc_V1_9

New CGCI and HCMI Masked Somatic Mutation tables

BigQuery tables created

- isb-cgc-bq.CGCI.masked_somatic_mutation_hg38_gdc_current
- isb-cgc-bq.CGCI_versioned.masked_somatic_mutation_hg38_gdc_r27
- isb-cgc-bq.HCMI_versioned.masked_somatic_mutation_hg38_gdc_r27

BigQuery tables updated

- isb-cgc-bq.HCMI.masked_somatic_mutation_hg38_gdc_current

February 1, 2021

New CTSP RNA Seq tables

BigQuery tables created

- isb-cgc-bq.CTSP.RNAseq_hg38_gdc_current
- isb-cgc-bq.CTSP_versioned.RNAseq_hg38_gdc_r23

January 12, 2021

New HCMI RNA Seq table

BigQuery tables created

- isb-cgc-bq.HCMI.RNAseq_hg38_gdc_r27

BigQuery tables updated

- isb-cgc-bq.HCMI.RNAseq_hg38_gdc_current

January 4, 2021

New TARGET RNA Seq tables

BigQuery tables created

- isb-cgc-bq.TARGET.RNAseq_hg38_gdc_current
- isb-cgc-bq.TARGET_versioned.RNAseq_hg38_gdc_r25
- isb-cgc-bq.TARGET_versioned.RNAseq_hg38_gdc_r26

December 17, 2020

New CPTAC Masked Somatic Mutation (MAF) tables.

BigQuery tables created

- isb-cgc-bq:CPTAC.masked_somatic_mutation_hg38_gdc_current
- isb-cgc-bq:CPTAC_versioned.masked_somatic_mutation_hg38_gdc_r25

December 16, 2020

New per sample file metadata tables added to isb-cgc-bq for GDC release 27.

BigQuery tables created

- isb-cgc-bq.CMI_versioned.per_sample_file_metadata_hg38_gdc_r27
- isb-cgc-bq.WCDT_versioned.per_sample_file_metadata_hg38_gdc_r27
- isb-cgc-bq.GENIE_versioned.per_sample_file_metadata_hg38_gdc_r27
- isb-cgc-bq.OHSU_versioned.per_sample_file_metadata_hg38_gdc_r27
- isb-cgc-bq.FM_versioned.per_sample_file_metadata_hg38_gdc_r27
- isb-cgc-bq.VAREPOP_versioned.per_sample_file_metadata_hg38_gdc_r27
- isb-cgc-bq.CTSP_versioned.per_sample_file_metadata_hg38_gdc_r27
- isb-cgc-bq.NCICCR_versioned.per_sample_file_metadata_hg38_gdc_r27
- isb-cgc-bq.ORGANOID_versioned.per_sample_file_metadata_hg38_gdc_r27

- isb-cgc-bq.MMRF_versioned.per_sample_file_metadata_hg38_gdc_r27
- isb-cgc-bq.CGCI_versioned.per_sample_file_metadata_hg38_gdc_r27
- isb-cgc-bq.HCMI_versioned.per_sample_file_metadata_hg38_gdc_r27
- isb-cgc-bq.BEATAML1_0_versioned.per_sample_file_metadata_hg38_gdc_r27
- isb-cgc-bq.CPTAC_versioned.per_sample_file_metadata_hg38_gdc_r27
- isb-cgc-bq.TARGET_versioned.per_sample_file_metadata_hg38_gdc_r27
- isb-cgc-bq.TCGA_versioned.per_sample_file_metadata_hg38_gdc_r27
- isb-cgc-bq.CCLE_versioned.per_sample_file_metadata_hg19_gdc_r27
- isb-cgc-bq.TARGET_versioned.per_sample_file_metadata_hg19_gdc_r27
- isb-cgc-bq.TCGA_versioned.per_sample_file_metadata_hg19_gdc_r27

Current per sample file metadata tables updated to GDC release 27.

BigQuery tables updated

- isb-cgc-bq.CMI.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.WCDT.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.GENIE.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.OHSU.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.FM.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.VAREPOP.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.CTSP.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.NCICCR.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.ORGANOID.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.MMRF.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.CGCI.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.BEATAML1_0.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.CPTAC.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.TARGET.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.TCGA.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.CCLE.per_sample_file_metadata_hg19_gdc_current
- isb-cgc-bq.TARGET.per_sample_file_metadata_hg19_gdc_current
- isb-cgc-bq.TCGA.per_sample_file_metadata_hg19_gdc_current

December 14, 2020

New GDC release 27 file metadata tables.

BigQuery tables created

- isb-cgc-bq.GDC_case_file_metadata_versioned.GDCfileID_to_GCSurl_r27
- isb-cgc-bq.GDC_case_file_metadata_versioned.fileData_legacy_r27
- isb-cgc-bq.GDC_case_file_metadata_versioned.fileData_active_r27

- isb-cgc-bq.GDC_case_file_metadata_versioned.caseData_r27
- isb-cgc-bq.GDC_case_file_metadata_versioned.aliquot2caseIDmap_r27
- isb-cgc-bq.GDC_case_file_metadata_versioned.slide2caseIDmap_r27

Current file metadata tables updated to GDC release 27.

BigQuery tables updated

- isb-cgc-bq.GDC_case_file_metadata.GDCfileID_to_GCSurl_current
- isb-cgc-bq.GDC_case_file_metadata.fileData_legacy_current
- isb-cgc-bq.GDC_case_file_metadata.fileData_active_current
- isb-cgc-bq.GDC_case_file_metadata.caseData_current
- isb-cgc-bq.GDC_case_file_metadata.aliquot2caseIDmap_current
- isb-cgc-bq.GDC_case_file_metadata.slide2caseIDmap_current

December 9, 2020

New CPTAC RNA Seq tables

BigQuery tables created

- isb-cgc-bq.CPTAC.RNAseq_hg38_gdc_current
- isb-cgc-bq.CPTAC_versioned.RNAseq_hg38_gdc_r25

December 8, 2020

CPTAC2, CPTAC3, TCGA quant proteome data from PDC, released Sept. 2020.

BigQuery tables created

- isb-cgc-bq.TCGA.quant_proteome_TCGA_ovarian_PNNL_pdc_current
- isb-cgc-bq.TCGA_versioned.quant_proteome_TCGA_ovarian_PNNL_pdc_2020_09
- isb-cgc-bq.TCGA.quant_proteome_TCGA_ovarian_JHU_pdc_current
- isb-cgc-bq.TCGA_versioned.quant_proteome_TCGA_ovarian_JHU_pdc_2020_09
- isb-cgc-bq.TCGA.quant_proteome_TCGA_breast_cancer_pdc_current
- isb-cgc-bq.TCGA_versioned.quant_proteome_TCGA_breast_cancer_pdc_2020_09
- isb-cgc-bq.CPTAC.quant_proteome_prospective_ovarian_PNNL_qeplus_pdc_current
- isb-cgc-bq.CPTAC_versioned.quant_proteome_prospective_ovarian_PNNL_qeplus_pdc_2020_09
- isb-cgc-bq.CPTAC.quant_proteome_prospective_ovarian_JHU_pdc_current
- isb-cgc-bq.CPTAC_versioned.quant_proteome_prospective_ovarian_JHU_pdc_2020_09
- isb-cgc-bq.CPTAC.quant_proteome_prospective_colon_PNNL_qeplus_pdc_current
- isb-cgc-bq.CPTAC_versioned.quant_proteome_prospective_colon_PNNL_qeplus_pdc_2020_09
- isb-cgc-bq.CPTAC.quant_proteome_prospective_breast_BI_pdc_current
- isb-cgc-bq.CPTAC_versioned.quant_proteome_prospective_breast_BI_pdc_2020_09
- isb-cgc-bq.CPTAC.quant_proteome_CPTAC_UCEC_discovery_study_pdc_current
- isb-cgc-bq.CPTAC_versioned.quant_proteome_CPTAC_UCEC_discovery_study_pdc_2020_09
- isb-cgc-bq.CPTAC.quant_proteome_CPTAC_LUAD_discovery_study_pdc_current

- isb-cgc-bq.CPTAC_versioned.quant_proteome_CPTAC_LUAD_discovery_study_pdc_2020_09
- isb-cgc-bq.CPTAC.quant_proteome_CPTAC_CCRCC_discovery_study_pdc_current
- isb-cgc-bq.CPTAC_versioned.quant_proteome_CPTAC_CCRCC_discovery_study_pdc_2020_09

Pediatric Brain Cancer Pilot proteome study from PDC, released Sept. 2020.

- isb-cgc-bq.CBTTC.quant_proteome_pediatric_brain_cancer_pilot_study_pdc_current
- isb-cgc-bq.CBTTC_versioned.quant_proteome_pediatric_brain_cancer_pilot_study_pdc_2020_09

Hepatitis B Virus (HBV) Related Hepatocellular Carcinoma Proteome study, released Sept. 2020.

- isb-cgc-bq.ICPC.quant_proteome_HBV_related_hepatocellular_carcinoma_pdc_current
- isb-cgc-bq.ICPC_versioned.quant_proteome_HBV_related_hepatocellular_carcinoma_pdc_2020_09

Proteogenomics of Gastric Cancer Proteome study, released Sept. 2020.

- isb-cgc-bq.ICPC.quant_proteome_proteogenomics_of_gastric_cancer_pdc_current
- isb-cgc-bq.ICPC_versioned.quant_proteome_proteogenomics_of_gastric_cancer_pdc_2020_09

December 2, 2020

Clinical data tables released for GDC release 27. Current clinical tables were updated to GDC release 27.

BigQuery tables created and updated

- isb-cgc-bq.MMRF.clinical_gdc_current
- isb-cgc-bq.MMRF_versioned.clinical_gdc_r27
- isb-cgc-bq.NCICCR.clinical_gdc_current
- isb-cgc-bq.NCICCR_versioned.clinical_gdc_r27
- isb-cgc-bq.OHSU.clinical_gdc_current
- isb-cgc-bq.OHSU_versioned.clinical_gdc_r27
- isb-cgc-bq.HCMI.clinical_follow_ups_molecular_tests_gdc_current
- isb-cgc-bq.HCMI_versioned.clinical_follow_ups_molecular_tests_gdc_r27
- isb-cgc-bq.HCMI.clinical_diagnoses_treatments_gdc_current
- isb-cgc-bq.HCMI_versioned.clinical_diagnoses_treatments_gdc_r27
- isb-cgc-bq.ORGANOID.clinical_gdc_current
- isb-cgc-bq.ORGANOID_versioned.clinical_gdc_r27
- isb-cgc-bq.CGCI.clinical_diagnoses_treatments_gdc_current
- isb-cgc-bq.CGCI_versioned.clinical_diagnoses_treatments_gdc_r27
- isb-cgc-bq.MMRF.clinical_diagnoses_treatments_gdc_current
- isb-cgc-bq.MMRF_versioned.clinical_diagnoses_treatments_gdc_r27
- isb-cgc-bq.MMRF.clinical_follow_ups_gdc_current
- isb-cgc-bq.MMRF_versioned.clinical_follow_ups_gdc_r27
- isb-cgc-bq.TCGA.clinical_gdc_current
- isb-cgc-bq.TCGA_versioned.clinical_gdc_r27
- isb-cgc-bq.TARGET.clinical_gdc_current

- isb-cgc-bq.TARGET_versioned.clinical_gdc_r27
- isb-cgc-bq.MMRF.clinical_follow_ups_molecular_tests_gdc_current
- isb-cgc-bq.MMRF_versioned.clinical_follow_ups_molecular_tests_gdc_r27
- isb-cgc-bq.GENIE.clinical_gdc_current
- isb-cgc-bq.GENIE_versioned.clinical_gdc_r27
- isb-cgc-bq.VAREPOP.clinical_gdc_current
- isb-cgc-bq.VAREPOP_versioned.clinical_gdc_r27
- isb-cgc-bq.CTSP.clinical_gdc_current
- isb-cgc-bq.CTSP_versioned.clinical_gdc_r27
- isb-cgc-bq.CGCI.clinical_follow_ups_molecular_tests_gdc_current
- isb-cgc-bq.CGCI_versioned.clinical_follow_ups_molecular_tests_gdc_r27
- isb-cgc-bq.VAREPOP.clinical_family_histories_gdc_current
- isb-cgc-bq.VAREPOP_versioned.clinical_family_histories_gdc_r27
- isb-cgc-bq.BEATAML1_0.clinical_gdc_current
- isb-cgc-bq.BEATAML1_0_versioned.clinical_gdc_r27
- isb-cgc-bq.MMRF.clinical_family_histories_gdc_current
- isb-cgc-bq.MMRF_versioned.clinical_family_histories_gdc_r27
- isb-cgc-bq.WCDT.clinical_gdc_current
- isb-cgc-bq.WCDT_versioned.clinical_gdc_r27
- isb-cgc-bq.VAREPOP.clinical_diagnoses_treatments_gdc_current
- isb-cgc-bq.VAREPOP_versioned.clinical_diagnoses_treatments_gdc_r27
- isb-cgc-bq.HCMI.clinical_diagnoses_gdc_current
- isb-cgc-bq.HCMI_versioned.clinical_diagnoses_gdc_r27
- isb-cgc-bq.CGCI.clinical_diagnoses_gdc_current
- isb-cgc-bq.CGCI_versioned.clinical_diagnoses_gdc_r27
- isb-cgc-bq.CGCI.clinical_gdc_current
- isb-cgc-bq.CGCI_versioned.clinical_gdc_r27
- isb-cgc-bq.CGCI.clinical_follow_ups_gdc_current
- isb-cgc-bq.CGCI_versioned.clinical_follow_ups_gdc_r27
- isb-cgc-bq.TCGA.clinical_diagnoses_treatments_gdc_current
- isb-cgc-bq.TCGA_versioned.clinical_diagnoses_treatments_gdc_r27
- isb-cgc-bq.CPTAC.clinical_gdc_current
- isb-cgc-bq.CPTAC_versioned.clinical_gdc_r27
- isb-cgc-bq.HCMI.clinical_gdc_current
- isb-cgc-bq.HCMI_versioned.clinical_gdc_r27
- isb-cgc-bq.CMI.clinical_gdc_current

- isb-cgc-bq.CMI_versioned.clinical_gdc_r27
- isb-cgc-bq.FM.clinical_gdc_current
- isb-cgc-bq.FM_versioned.clinical_gdc_r27
- isb-cgc-bq.HCMI.clinical_follow_ups_gdc_current
- isb-cgc-bq.HCMI_versioned.clinical_follow_ups_gdc_r27

November 16, 2020

New TARGET controlled-access VCF tables.

BigQuery tables created

- isb-cgc-cbq.TARGET.vcf_hg38_gdc_current
- isb-cgc-cbq.TARGET_versioned.vcf_hg38_gdc_r22

October 30, 2020

RNA Seq data tables released for the WCDT program.

BigQuery tables created

- isb-cgc-bq:WCDT.RNAseq_hg38_gdc_current
- isb-cgc-bq:WCDT_versioned.RNAseq_hg38_gdc_r22

October 23, 2020

Clinical data tables released for GDC release 25 and 26.

BigQuery tables created

- isb-cgc-bq:BEATAML1_0_versioned.clinical_gdc_r25
- isb-cgc-bq:CGCI_versioned.clinical_gdc_r25
- isb-cgc-bq:CGCI_versioned.clinical_diagnoses_gdc_r25
- isb-cgc-bq:CGCI_versioned.clinical_diagnoses_treatments_gdc_r25
- isb-cgc-bq:CGCI_versioned.clinical_follow_ups_gdc_r25
- isb-cgc-bq:CGCI_versioned.clinical_follow_ups_molecular_tests_gdc_r25
- isb-cgc-bq:CPTAC_versioned.clinical_gdc_r25
- isb-cgc-bq:CTSP_versioned.clinical_gdc_r25
- isb-cgc-bq:FM_versioned.clinical_gdc_r25
- isb-cgc-bq:GENIE_versioned.clinical_gdc_r25
- isb-cgc-bq:HCMI_versioned.clinical_gdc_r25
- isb-cgc-bq:HCMI_versioned.clinical_diagnoses_gdc_r25
- isb-cgc-bq:HCMI_versioned.clinical_diagnoses_treatments_gdc_r25
- isb-cgc-bq:HCMI_versioned.clinical_follow_ups_gdc_r25
- isb-cgc-bq:HCMI_versioned.clinical_follow_ups_molecular_tests_gdc_r25
- isb-cgc-bq:MMRF_versioned.clinical_gdc_r25
- isb-cgc-bq:MMRF_versioned.clinical_diagnoses_treatments_gdc_r25
- isb-cgc-bq:MMRF_versioned.clinical_family_histories_gdc_r25

- isb-cgc-bq:MMRF_versioned.clinical_follow_ups_gdc_r25
- isb-cgc-bq:MMRF_versioned.clinical_follow_ups_molecular_tests_gdc_r25
- isb-cgc-bq:NCICCR_versioned.clinical_gdc_r25
- isb-cgc-bq:OHSU_versioned.clinical_gdc_r25
- isb-cgc-bq:ORGANOID_versioned.clinical_gdc_r25
- isb-cgc-bq:TARGET_versioned.clinical_gdc_r25
- isb-cgc-bq:TCGA_versioned.clinical_gdc_r25
- isb-cgc-bq:TCGA_versioned.clinical_diagnoses_treatments_gdc_r25
- isb-cgc-bq:VAREPOP_versioned.clinical_gdc_r25
- isb-cgc-bq:VAREPOP_versioned.clinical_diagnoses_treatments_gdc_r25
- isb-cgc-bq:VAREPOP_versioned.clinical_family_histories_gdc_r25
- isb-cgc-bq:WCDT_versioned.clinical_gdc_r25
- isb-cgc-bq:BEATAML1_0_versioned.clinical_gdc_r26
- isb-cgc-bq:CGCI_versioned.clinical_gdc_r26
- isb-cgc-bq:CGCI_versioned.clinical_diagnoses_gdc_r26
- isb-cgc-bq:CGCI_versioned.clinical_diagnoses_treatments_gdc_r26
- isb-cgc-bq:CGCI_versioned.clinical_follow_ups_gdc_r26
- isb-cgc-bq:CGCI_versioned.clinical_follow_ups_molecular_tests_gdc_r26
- isb-cgc-bq:CMI_versioned.clinical_gdc_r26
- isb-cgc-bq:CPTAC_versioned.clinical_gdc_r26
- isb-cgc-bq:CTSP_versioned.clinical_gdc_r26
- isb-cgc-bq:FM_versioned.clinical_gdc_r26
- isb-cgc-bq:GENIE_versioned.clinical_gdc_r26
- isb-cgc-bq:HCMI_versioned.clinical_gdc_r26
- isb-cgc-bq:HCMI_versioned.clinical_diagnoses_gdc_r26
- isb-cgc-bq:HCMI_versioned.clinical_diagnoses_treatments_gdc_r26
- isb-cgc-bq:HCMI_versioned.clinical_follow_ups_gdc_r26
- isb-cgc-bq:HCMI_versioned.clinical_follow_ups_molecular_tests_gdc_r26
- isb-cgc-bq:MMRF_versioned.clinical_gdc_r26
- isb-cgc-bq:MMRF_versioned.clinical_diagnoses_treatments_gdc_r26
- isb-cgc-bq:MMRF_versioned.clinical_family_histories_gdc_r26
- isb-cgc-bq:MMRF_versioned.clinical_follow_ups_gdc_r26
- isb-cgc-bq:MMRF_versioned.clinical_follow_ups_molecular_tests_gdc_r26
- isb-cgc-bq:NCICCR_versioned.clinical_gdc_r26
- isb-cgc-bq:MMRF_versioned.clinical_gdc_r26
- isb-cgc-bq:NCICCR_versioned.clinical_gdc_r26

- isb-cgc-bq:MMRF_versioned.clinical_gdc_r26
- isb-cgc-bq:NCICCR_versioned.clinical_gdc_r26
- isb-cgc-bq:MMRF_versioned.clinical_diagnoses_treatments_gdc_r26
- isb-cgc-bq:NCICCR_versioned.clinical_gdc_r26
- isb-cgc-bq:MMRF_versioned.clinical_diagnoses_treatments_gdc_r26
- isb-cgc-bq:NCICCR_versioned.clinical_family_histories_gdc_r26
- isb-cgc-bq:MMRF_versioned.clinical_gdc_r26
- isb-cgc-bq:CMI.clinical_gdc_current

Current clinical tables were updated to GDC release 26.

BigQuery tables updated

- isb-cgc-bq:BEATAML1_0.clinical_gdc_current
- isb-cgc-bq:CGCI.clinical_gdc_current
- isb-cgc-bq:CGCI.clinical_diagnoses_gdc_current
- isb-cgc-bq:CGCI.clinical_diagnoses_treatments_gdc_current
- isb-cgc-bq:CGCI.clinical_follow_ups_gdc_current
- isb-cgc-bq:CGCI.clinical_follow_ups_molecular_tests_gdc_current
- isb-cgc-bq:CPTAC.clinical_gdc_current
- isb-cgc-bq:CTSP.clinical_gdc_current
- isb-cgc-bq:FM.clinical_gdc_current
- isb-cgc-bq:GENIE.clinical_gdc_current
- isb-cgc-bq:HCMI.clinical_gdc_current
- isb-cgc-bq:HCMI.clinical_diagnoses_gdc_current
- isb-cgc-bq:HCMI.clinical_diagnoses_treatments_gdc_current
- isb-cgc-bq:HCMI.clinical_follow_ups_gdc_current
- isb-cgc-bq:HCMI.clinical_follow_ups_molecular_tests_gdc_current
- isb-cgc-bq:MMRF.clinical_gdc_current
- isb-cgc-bq:MMRF.clinical_diagnoses_treatments_gdc_current
- isb-cgc-bq:MMRF.clinical_family_histories_gdc_current
- isb-cgc-bq:MMRF.clinical_follow_ups_gdc_current
- isb-cgc-bq:MMRF.clinical_follow_ups_molecular_tests_gdc_current
- isb-cgc-bq:NCICCR.clinical_gdc_current
- isb-cgc-bq:MMRF.clinical_gdc_current
- isb-cgc-bq:NCICCR.clinical_gdc_current
- isb-cgc-bq:MMRF.clinical_gdc_current
- isb-cgc-bq:NCICCR.clinical_gdc_current
- isb-cgc-bq:MMRF.clinical_diagnoses_treatments_gdc_current

- isb-cgc-bq:NCICCR.clinical_gdc_current
- isb-cgc-bq:MMRF.clinical_diagnoses_treatments_gdc_current
- isb-cgc-bq:NCICCR.clinical_family_histories_gdc_current
- isb-cgc-bq:MMRF.clinical_gdc_current

RNA Seq data tables released for the CMI program.

BigQuery tables created

- isb-cgc-bq:CMI.RNAseq_hg38_gdc_current
- isb-cgc-bq:CMI_versioned.RNAseq_hg38_gdc_r26

October 21, 2020

RNA Seq data tables released for the CGCI program.

BigQuery tables created

- isb-cgc-bq:CGCI.RNAseq_hg38_gdc_current
- isb-cgc-bq:CGCI_versioned.RNAseq_hg38_gdc_r24

October 15, 2020

Current file metadata tables updated to GDC release 26.

BigQuery tables updated

- isb-cgc-bq.GDC_case_file_metadata.GDCfileID_to_GCSurl_current
- isb-cgc-bq.GDC_case_file_metadata.fileData_legacy_current
- isb-cgc-bq.GDC_case_file_metadata.fileData_active_current
- isb-cgc-bq.GDC_case_file_metadata.caseData_current
- isb-cgc-bq.GDC_case_file_metadata.aliquot2caseIDmap_current
- isb-cgc-bq.GDC_case_file_metadata.slide2caseIDmap_current

October 14, 2020

New GDC release 26 file metadata tables.

BigQuery tables created

- isb-cgc-bq.GDC_case_file_metadata_versioned.GDCfileID_to_GCSurl_r26
- isb-cgc-bq.GDC_case_file_metadata_versioned.fileData_legacy_r26
- isb-cgc-bq.GDC_case_file_metadata_versioned.fileData_active_r26
- isb-cgc-bq.GDC_case_file_metadata_versioned.caseData_r26
- isb-cgc-bq.GDC_case_file_metadata_versioned.aliquot2caseIDmap_r26
- isb-cgc-bq.GDC_case_file_metadata_versioned.slide2caseIDmap_r26

New per sample file metadata tables added to isb-cgc-bq for GDC release 26.

BigQuery tables created

- isb-cgc-bq.WCDT_versioned.per_sample_file_metadata_hg38_gdc_r26
- isb-cgc-bq.GENIE_versioned.per_sample_file_metadata_hg38_gdc_r26
- isb-cgc-bq.OHSU_versioned.per_sample_file_metadata_hg38_gdc_r26

- isb-cgc-bq.FM_versioned.per_sample_file_metadata_hg38_gdc_r26
- isb-cgc-bq.VAREPOP_versioned.per_sample_file_metadata_hg38_gdc_r26
- isb-cgc-bq.CTSP_versioned.per_sample_file_metadata_hg38_gdc_r26
- isb-cgc-bq.NCICCR_versioned.per_sample_file_metadata_hg38_gdc_r26
- isb-cgc-bq.ORGANOID_versioned.per_sample_file_metadata_hg38_gdc_r26
- isb-cgc-bq.MMRF_versioned.per_sample_file_metadata_hg38_gdc_r26
- isb-cgc-bq.CGCI_versioned.per_sample_file_metadata_hg38_gdc_r26
- isb-cgc-bq.HCMI_versioned.per_sample_file_metadata_hg38_gdc_r26
- isb-cgc-bq.BEATAML1_0_versioned.per_sample_file_metadata_hg38_gdc_r26
- isb-cgc-bq.CPTAC_versioned.per_sample_file_metadata_hg38_gdc_r26
- isb-cgc-bq.TARGET_versioned.per_sample_file_metadata_hg38_gdc_r26
- isb-cgc-bq.TCGA_versioned.per_sample_file_metadata_hg38_gdc_r26
- isb-cgc-bq.CCLE_versioned.per_sample_file_metadata_hg19_gdc_r26
- isb-cgc-bq.TARGET_versioned.per_sample_file_metadata_hg19_gdc_r26
- isb-cgc-bq.TCGA_versioned.per_sample_file_metadata_hg19_gdc_r26
- isb-cgc-bq.CMI_versioned.per_sample_file_metadata_hg38_gdc_r26
- isb-cgc-bq.CMI.per_sample_file_metadata_hg38_gdc_current

Current per sample file metadata tables updated to GDC release 26.

BigQuery tables updated

- isb-cgc-bq.WCDT.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.GENIE.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.OHSU.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.FM.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.VAREPOP.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.CTSP.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.NCICCR.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.ORGANOID.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.MMRF.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.CGCI.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.HCMI.per_sample_file_metadata_hg38_gdc_r26
- isb-cgc-bq.BEATAML1_0.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.CPTAC.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.TARGET.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.TCGA.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq.CCLE.per_sample_file_metadata_hg19_gdc_current
- isb-cgc-bq.TARGET.per_sample_file_metadata_hg19_gdc_current

- isb-cgc-bq.TCGA.per_sample_file_metadata_hg19_gdc_current

October 06, 2020

New per sample file metadata tables added to isb-cgc-bq for GDC release 25.

BigQuery tables created

- isb-cgc-bq.WCDT_versioned.per_sample_file_metadata_hg38_gdc_r25
- isb-cgc-bq.GENIE_versioned.per_sample_file_metadata_hg38_gdc_r25
- isb-cgc-bq.OHSU_versioned.per_sample_file_metadata_hg38_gdc_r25
- isb-cgc-bq.FM_versioned.per_sample_file_metadata_hg38_gdc_r25
- isb-cgc-bq.VAREPOP_versioned.per_sample_file_metadata_hg38_gdc_r25
- isb-cgc-bq.CTSP_versioned.per_sample_file_metadata_hg38_gdc_r25
- isb-cgc-bq.NCICCR_versioned.per_sample_file_metadata_hg38_gdc_r25
- isb-cgc-bq.ORGANOID_versioned.per_sample_file_metadata_hg38_gdc_r25
- isb-cgc-bq.MMRF_versioned.per_sample_file_metadata_hg38_gdc_r25
- isb-cgc-bq.CGCI_versioned.per_sample_file_metadata_hg38_gdc_r25
- isb-cgc-bq.HCMI_versioned.per_sample_file_metadata_hg38_gdc_r25
- isb-cgc-bq.BEATAML1_0_versioned.per_sample_file_metadata_hg38_gdc_r25
- isb-cgc-bq.CPTAC_versioned.per_sample_file_metadata_hg38_gdc_r25
- isb-cgc-bq.TARGET_versioned.per_sample_file_metadata_hg38_gdc_r25
- isb-cgc-bq.TCGA_versioned.per_sample_file_metadata_hg38_gdc_r25
- isb-cgc-bq.CCLE_versioned.per_sample_file_metadata_hg19_gdc_r25
- isb-cgc-bq.TARGET_versioned.per_sample_file_metadata_hg19_gdc_r25
- isb-cgc-bq.TCGA_versioned.per_sample_file_metadata_hg19_gdc_r25

October 02, 2020

Open Somatic Mutation data tables released for the HCMI program.

BigQuery tables created

- isb-cgc-bq.HCMI.masked_somatic_mutation_hg38_gdc_current
- isb-cgc-bq.HCMI_versioned.masked_somatic_mutation_hg38_gdc_r23

The new COSMIC release v92 data is available in BigQuery.

BigQuery tables created

- isb-cgc-bq.COSMIC.ASCAT_purity_ploidy_grch37_current
- isb-cgc-bq.COSMIC.ASCAT_purity_ploidy_grch38_current
- isb-cgc-bq.COSMIC.breakpoints_grch37_current
- isb-cgc-bq.COSMIC.breakpoints_grch38_current
- isb-cgc-bq.COSMIC.cancer_gene_census_grch37_current
- isb-cgc-bq.COSMIC.cancer_gene_census_grch38_current
- isb-cgc-bq.COSMIC.cancer_gene_census_hallmarks_of_cancer_grch37_current

- isb-cgc-bq.COSMIC.cancer_gene_census_hallmarks_of_cancer_grch38_current
- isb-cgc-bq.COSMIC.classification_grch37_current
- isb-cgc-bq.COSMIC.classification_grch38_current
- isb-cgc-bq.COSMIC.complete_CNA_grch37_current
- isb-cgc-bq.COSMIC.complete_CNA_grch38_current
- isb-cgc-bq.COSMIC.complete_differential_methylation_grch37_current
- isb-cgc-bq.COSMIC.complete_differential_methylation_grch38_current
- isb-cgc-bq.COSMIC.complete_gene_expression_grch37_current
- isb-cgc-bq.COSMIC.complete_gene_expression_grch38_current
- isb-cgc-bq.COSMIC.complete_targeted_screens_mutant_grch37_current
- isb-cgc-bq.COSMIC.complete_targeted_screens_mutant_grch38_current
- isb-cgc-bq.COSMIC.fusion_grch37_current
- isb-cgc-bq.COSMIC.fusion_grch38_current
- isb-cgc-bq.COSMIC.genome_screens_mutant_grch37_current
- isb-cgc-bq.COSMIC.genome_screens_mutant_grch38_current
- isb-cgc-bq.COSMIC.HGNC_grch37_current
- isb-cgc-bq.COSMIC.HGNC_grch38_current
- isb-cgc-bq.COSMIC.mutant_census_grch37_current
- isb-cgc-bq.COSMIC.mutant_census_grch38_current
- isb-cgc-bq.COSMIC.mutant_grch37_current
- isb-cgc-bq.COSMIC.mutant_grch38_current
- isb-cgc-bq.COSMIC.mutation_tracking_grch37_current
- isb-cgc-bq.COSMIC.mutation_tracking_grch38_current
- isb-cgc-bq.COSMIC.NCV_grch37_current
- isb-cgc-bq.COSMIC.NCV_grch38_current
- isb-cgc-bq.COSMIC.resistance_mutations_grch37_current
- isb-cgc-bq.COSMIC.resistance_mutations_grch38_current
- isb-cgc-bq.COSMIC.sample_grch37_current
- isb-cgc-bq.COSMIC.sample_grch38_current
- isb-cgc-bq.COSMIC.structural_variants_grch37_current
- isb-cgc-bq.COSMIC.structural_variants_grch38_current
- isb-cgc-bq.COSMIC.transcripts_grch37_current
- isb-cgc-bq.COSMIC.transcripts_grch38_current
- isb-cgc-bq.COSMIC_versioned.ASCAT_purity_ploidy_grch37_v92
- isb-cgc-bq.COSMIC_versioned.ASCAT_purity_ploidy_grch38_v92
- isb-cgc-bq.COSMIC_versioned.breakpoints_grch37_v92

- isb-cgc-bq.COSMIC_versioned.breakpoints_grch38_v92
- isb-cgc-bq.COSMIC_versioned.cancer_gene_census_grch37_v92
- isb-cgc-bq.COSMIC_versioned.cancer_gene_census_grch38_v92
- isb-cgc-bq.COSMIC_versioned.cancer_gene_census_hallmarks_of_cancer_grch37_v92
- isb-cgc-bq.COSMIC_versioned.cancer_gene_census_hallmarks_of_cancer_grch38_v92
- isb-cgc-bq.COSMIC_versioned.classification_grch37_v92
- isb-cgc-bq.COSMIC_versioned.classification_grch38_v92
- isb-cgc-bq.COSMIC_versioned.complete_CNA_grch37_v92
- isb-cgc-bq.COSMIC_versioned.complete_CNA_grch38_v92
- isb-cgc-bq.COSMIC_versioned.complete_differential_methylation_grch37_v92
- isb-cgc-bq.COSMIC_versioned.complete_differential_methylation_grch38_v92
- isb-cgc-bq.COSMIC_versioned.complete_gene_expression_grch37_v92
- isb-cgc-bq.COSMIC_versioned.complete_gene_expression_grch38_v92
- isb-cgc-bq.COSMIC_versioned.complete_targeted_screens_mutant_grch37_v92
- isb-cgc-bq.COSMIC_versioned.complete_targeted_screens_mutant_grch38_v92
- isb-cgc-bq.COSMIC_versioned.fusion_grch37_v92
- isb-cgc-bq.COSMIC_versioned.fusion_grch38_v92
- isb-cgc-bq.COSMIC_versioned.genome_screens_mutant_grch37_v92
- isb-cgc-bq.COSMIC_versioned.genome_screens_mutant_grch38_v92
- isb-cgc-bq.COSMIC_versioned.HGNC_grch37_v92
- isb-cgc-bq.COSMIC_versioned.HGNC_grch38_v92
- isb-cgc-bq.COSMIC_versioned.mutant_census_grch37_v92
- isb-cgc-bq.COSMIC_versioned.mutant_census_grch38_v92
- isb-cgc-bq.COSMIC_versioned.mutant_grch37_v92
- isb-cgc-bq.COSMIC_versioned.mutant_grch38_v92
- isb-cgc-bq.COSMIC_versioned.mutation_tracking_grch37_v92
- isb-cgc-bq.COSMIC_versioned.mutation_tracking_grch38_v92
- isb-cgc-bq.COSMIC_versioned.NCV_grch37_v92
- isb-cgc-bq.COSMIC_versioned.NCV_grch38_v92
- isb-cgc-bq.COSMIC_versioned.resistance_mutations_grch37_v92
- isb-cgc-bq.COSMIC_versioned.resistance_mutations_grch38_v92
- isb-cgc-bq.COSMIC_versioned.sample_grch37_v92
- isb-cgc-bq.COSMIC_versioned.sample_grch38_v92
- isb-cgc-bq.COSMIC_versioned.structural_variants_grch37_v92
- isb-cgc-bq.COSMIC_versioned.structural_variants_grch38_v92
- isb-cgc-bq.COSMIC_versioned.transcripts_grch37_v92

- isb-cgc-bq.COSMIC_versioned.transcripts_grch38_v92

September 21, 2020

Current file metadata tables updated to GDC release 25.

BigQuery tables updated

- isb-cgc-bq.GDC_case_file_metadata.GDCfileID_to_GCSurl_current
- isb-cgc-bq.GDC_case_file_metadata.fileData_legacy_current
- isb-cgc-bq.GDC_case_file_metadata.fileData_active_current
- isb-cgc-bq.GDC_case_file_metadata.caseData_current
- isb-cgc-bq.GDC_case_file_metadata.aliquot2caseIDmap_current
- isb-cgc-bq.GDC_case_file_metadata.slide2caseIDmap_current

September 18, 2020

New GDC release 25 file metadata tables.

BigQuery tables created

- isb-cgc-bq.GDC_case_file_metadata_versioned.GDCfileID_to_GCSurl_r25
- isb-cgc-bq.GDC_case_file_metadata_versioned.fileData_legacy_r25
- isb-cgc-bq.GDC_case_file_metadata_versioned.fileData_active_r25
- isb-cgc-bq.GDC_case_file_metadata_versioned.caseData_r25
- isb-cgc-bq.GDC_case_file_metadata_versioned.aliquot2caseIDmap_r25
- isb-cgc-bq.GDC_case_file_metadata_versioned.slide2caseIDmap_r25

September 8, 2020

Table generated as part of an analysis for a poster submitted to the ACM-BCB2020 conference.

BigQuery tables created

- isb-cgc-bq.supplementary_tables.Abdilleh_etal_ACM_BCB_2020_TCGA_bioclin_v0_Clinical_UNPIVOT

September 2, 2020

New GENCODE data, version 34 and 35.

BigQuery tables created

- isb-cgc-bq.GENCODE_versioned.annotation_gtf_hg38_v34
- isb-cgc-bq.GENCODE_versioned.annotation_gtf_hg38_v35
- isb-cgc-bq.GENCODE.annotation_gtf_hg38_current

August 28, 2020

New GDC release 24 clinical tables.

BigQuery tables created

- isb-cgc-bq:BEATAML1_0.clinical_gdc_current
- isb-cgc-bq:BEATAML1_0_versioned.clinical_gdc_r24
- isb-cgc-bq:CGCI.clinical_diagnoses_gdc_current
- isb-cgc-bq:CGCI.clinical_diagnoses_treatments_gdc_current

- isb-cgc-bq:CGCI.clinical_follow_ups_gdc_current
- isb-cgc-bq:CGCI.clinical_follow_ups_molecular_tests_gdc_current
- isb-cgc-bq:CGCI.clinical_gdc_current
- isb-cgc-bq:CGCI_versioned.clinical_diagnoses_gdc_r24
- isb-cgc-bq:CGCI_versioned.clinical_diagnoses_treatments_gdc_r24
- isb-cgc-bq:CGCI_versioned.clinical_follow_ups_gdc_r24
- isb-cgc-bq:CGCI_versioned.clinical_follow_ups_molecular_tests_gdc_r24
- isb-cgc-bq:CGCI_versioned.clinical_gdc_r24
- isb-cgc-bq:CPTAC.clinical_gdc_current
- isb-cgc-bq:CPTAC_versioned.clinical_gdc_r24
- isb-cgc-bq:CTSP.clinical_gdc_current
- isb-cgc-bq:CTSP_versioned.clinical_gdc_r24
- isb-cgc-bq:FM.clinical_gdc_current
- isb-cgc-bq:FM_versioned.clinical_gdc_r24
- isb-cgc-bq:GENIE.clinical_gdc_current
- isb-cgc-bq:GENIE_versioned.clinical_gdc_r24
- isb-cgc-bq:HCMI.clinical_diagnoses_gdc_current
- isb-cgc-bq:HCMI.clinical_diagnoses_treatments_gdc_current
- isb-cgc-bq:HCMI.clinical_follow_ups_gdc_current
- isb-cgc-bq:HCMI.clinical_follow_ups_molecular_tests_gdc_current
- isb-cgc-bq:HCMI.clinical_gdc_current
- isb-cgc-bq:HCMI_versioned.clinical_diagnoses_gdc_r24
- isb-cgc-bq:HCMI_versioned.clinical_diagnoses_treatments_gdc_r24
- isb-cgc-bq:HCMI_versioned.clinical_follow_ups_gdc_r24
- isb-cgc-bq:HCMI_versioned.clinical_follow_ups_molecular_tests_gdc_r24
- isb-cgc-bq:HCMI_versioned.clinical_gdc_r24
- isb-cgc-bq:MMRF.clinical_diagnoses_treatments_gdc_current
- isb-cgc-bq:MMRF.clinical_family_histories_gdc_current
- isb-cgc-bq:MMRF.clinical_follow_ups_gdc_current
- isb-cgc-bq:MMRF.clinical_follow_ups_molecular_tests_gdc_current
- isb-cgc-bq:MMRF.clinical_gdc_current
- isb-cgc-bq:MMRF_versioned.clinical_diagnoses_treatments_gdc_r24
- isb-cgc-bq:MMRF_versioned.clinical_family_histories_gdc_r24
- isb-cgc-bq:MMRF_versioned.clinical_follow_ups_gdc_r24
- isb-cgc-bq:MMRF_versioned.clinical_follow_ups_molecular_tests_gdc_r24
- isb-cgc-bq:MMRF_versioned.clinical_gdc_r24

- isb-cgc-bq:NCICCR.clinical_gdc_current
- isb-cgc-bq:NCICCR_versioned.clinical_gdc_r24
- isb-cgc-bq:OHSU.clinical_gdc_current
- isb-cgc-bq:OHSU_versioned.clinical_gdc_r24
- isb-cgc-bq:ORGANOID.clinical_gdc_current
- isb-cgc-bq:ORGANOID_versioned.clinical_gdc_r24
- isb-cgc-bq:TARGET.clinical_gdc_current
- isb-cgc-bq:TARGET_versioned.clinical_gdc_r24
- isb-cgc-bq:TCGA.clinical_diagnoses_treatments_gdc_current
- isb-cgc-bq:TCGA.clinical_gdc_current
- isb-cgc-bq:TCGA_versioned.clinical_diagnoses_treatments_gdc_r24
- isb-cgc-bq:TCGA_versioned.clinical_gdc_r24
- isb-cgc-bq:VAREPOP.clinical_diagnoses_treatments_gdc_current
- isb-cgc-bq:VAREPOP.clinical_family_histories_gdc_current
- isb-cgc-bq:VAREPOP.clinical_gdc_current
- isb-cgc-bq:VAREPOP_versioned.clinical_diagnoses_treatments_gdc_r24
- isb-cgc-bq:VAREPOP_versioned.clinical_family_histories_gdc_r24
- isb-cgc-bq:VAREPOP_versioned.clinical_gdc_r24
- isb-cgc-bq:WCDT.clinical_gdc_current
- isb-cgc-bq:WCDT_versioned.clinical_gdc_r24

July 23, 2020

New TCGA controlled-access MAF tables. New TARGET GDC release 22 RNAseq and miRNAseq tables.

BigQuery tables created

- isb-cgc-cbq:TCGA.maf_hg38_gdc_current
- isb-cgc-cbq:TCGA_versioned.maf_hg38_gdc_r14
- isb-cgc-bq:TARGET_versioned.miRNAseq_hg38_gdc_r22
- isb-cgc-bq:TARGET_versioned.RNAseq_hg38_gdc_r22
- isb-cgc-bq:TARGET.miRNAseq_hg38_gdc_current
- isb-cgc-bq:TARGET.RNAseq_hg38_gdc_current

July 21, 2020

New HCMI RNA seq table.

BigQuery tables created

- isb-cgc.HCMI.RNAseq_hg38_gdc_r23

July 9, 2020

New per sample file metadata tables added to isb-cgc-bq for GDC release 24.

BigQuery tables created

- isb-cgc-bq:BEATAML1_0.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq:BEATAML1_0_versioned.per_sample_file_metadata_hg38_gdc_r24
- isb-cgc-bq:TCGA.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq:TCGA_versioned.per_sample_file_metadata_hg38_gdc_r24
- isb-cgc-bq:TARGET.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq:TARGET_versioned.per_sample_file_metadata_hg38_gdc_r24
- isb-cgc-bq:GENIE.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq:GENIE_versioned.per_sample_file_metadata_hg38_gdc_r24
- isb-cgc-bq:CGCI.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq:CGCI_versioned.per_sample_file_metadata_hg38_gdc_r24
- isb-cgc-bq:CPTAC.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq:CPTAC_versioned.per_sample_file_metadata_hg38_gdc_r24
- isb-cgc-bq:CTSP.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq:CTSP_versioned.per_sample_file_metadata_hg38_gdc_r24
- isb-cgc-bq:FM.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq:FM_versioned.per_sample_file_metadata_hg38_gdc_r24
- isb-cgc-bq:HCMI.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq:HCMI_versioned.per_sample_file_metadata_hg38_gdc_r24
- isb-cgc-bq:MMRF.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq:MMRF_versioned.per_sample_file_metadata_hg38_gdc_r24
- isb-cgc-bq:NCICCR.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq:NCICCR_versioned.per_sample_file_metadata_hg38_gdc_r24
- isb-cgc-bq:OHSU.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq:OHSU_versioned.per_sample_file_metadata_hg38_gdc_r24
- isb-cgc-bq:ORGANOID.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq:ORGANOID_versioned.per_sample_file_metadata_hg38_gdc_r24
- isb-cgc-bq:VAREPOP.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq:VAREPOP_versioned.per_sample_file_metadata_hg38_gdc_r24
- isb-cgc-bq:WCDT.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq:WCDT_versioned.per_sample_file_metadata_hg38_gdc_r24
- isb-cgc-bq:CCLE.per_sample_file_metadata_hg38_gdc_current
- isb-cgc-bq:CCLE_versioned.per_sample_file_metadata_hg38_gdc_r24

Existing GDC Release 24 file metadata tables in the isb-cgc project were copied to the isb-cgc-bq project.

BigQuery tables created

- isb-cgc-bq.GDC_case_file_metadata_versioned.slide2caseIDmap_r24
- isb-cgc-bq.GDC_case_file_metadata_versioned.GDCfileID_to_GCSurl_r24

- isb-cgc-bq.GDC_case_file_metadata_versioned.fileData_legacy_r24
- isb-cgc-bq.GDC_case_file_metadata_versioned.fileData_active_r24
- isb-cgc-bq.GDC_case_file_metadata_versioned.caseData_r24
- isb-cgc-bq.GDC_case_file_metadata_versioned.aliquot2caseIDmap_r24
- isb-cgc-bq.GDC_case_file_metadata.slide2caseIDmap_current
- isb-cgc-bq.GDC_case_file_metadata.GDCfileID_to_GCSurl_current
- isb-cgc-bq.GDC_case_file_metadata.fileData_legacy_current
- isb-cgc-bq.GDC_case_file_metadata.fileData_active_current
- isb-cgc-bq.GDC_case_file_metadata.caseData_current
- isb-cgc-bq.GDC_case_file_metadata.aliquot2caseIDmap_current

June 16, 2020

The new COSMIC release v91 data is available in BigQuery.

BigQuery tables created

- isb-cgc:COSMIC_v91_grch37.ASCAT_Purity_Ploidy
- isb-cgc:COSMIC_v91_grch37.Breakpoints
- isb-cgc:COSMIC_v91_grch37.Cancer_Gene_Census
- isb-cgc:COSMIC_v91_grch37.Complete_CNA
- isb-cgc:COSMIC_v91_grch37.Complete_Differential_Methylation
- isb-cgc:COSMIC_v91_grch37.Complete_Gene_Expression
- isb-cgc:COSMIC_v91_grch37.Complete_Targeted_Screens_Mutant
- isb-cgc:COSMIC_v91_grch37.Fusion
- isb-cgc:COSMIC_v91_grch37.Genome_Screens_Mutant
- isb-cgc:COSMIC_v91_grch37.HGNC
- isb-cgc:COSMIC_v91_grch37.Mutant
- isb-cgc:COSMIC_v91_grch37.Mutant_Census
- isb-cgc:COSMIC_v91_grch37.Mutation_Tracking
- isb-cgc:COSMIC_v91_grch37.NCV
- isb-cgc:COSMIC_v91_grch37.Resistance_Mutations
- isb-cgc:COSMIC_v91_grch37.Sample
- isb-cgc:COSMIC_v91_grch37.Structural_Variants
- isb-cgc:COSMIC_v91_grch37.Transcripts
- isb-cgc:COSMIC_v91_grch38.ASCAT_Purity_Ploidy
- isb-cgc:COSMIC_v91_grch38.Breakpoints
- isb-cgc:COSMIC_v91_grch38.Cancer_Gene_Census
- isb-cgc:COSMIC_v91_grch38.Classification
- isb-cgc:COSMIC_v91_grch38.Complete_CNA

- isb-cgc:COSMIC_v91_grch38.Complete_Differential_Methylation
- isb-cgc:COSMIC_v91_grch38.Complete_Gene_Expression
- isb-cgc:COSMIC_v91_grch38.Complete_Targeted_Screens_Mutant
- isb-cgc:COSMIC_v91_grch38.Fusion
- isb-cgc:COSMIC_v91_grch38.Genome_Screens_Mutant
- isb-cgc:COSMIC_v91_grch38.HGNC
- isb-cgc:COSMIC_v91_grch38.Mutant
- isb-cgc:COSMIC_v91_grch38.Mutant_Census
- isb-cgc:COSMIC_v91_grch38.Mutation_Tracking
- isb-cgc:COSMIC_v91_grch38.NCV
- isb-cgc:COSMIC_v91_grch38.Resistance_Mutations
- isb-cgc:COSMIC_v91_grch38.Sample
- isb-cgc:COSMIC_v91_grch38.Structural_Variants
- isb-cgc:COSMIC_v91_grch38.Transcripts

June 09, 2020

New GDC file ID to GCS url tables added to isb-cgc for GDC release 24.

BigQuery tables created

- isb-cgc:GDC_metadata.rel24_GDCfileID_to_GCSurl

May 28, 2020

New data set and RNA Sequence table derived data tables added to isb-cgc.

BigQuery tables created

- isb-cgc:TARGET.RNAseq_hg38_r22

May 27, 2020

PanCancer tables were added to the isb-cgc project. The Pan-Cancer Atlas tables include clinical, methylation, RPPA and copy number data.

BigQuery tables created

The following tables were created under the isb-cgc:pancer-altas data set:

- BarcodeMap
- clinical_PANCAN_patient_with_followup
- EBpp_AdjustPANCAN_IlluminaHiSeq_RNASeqV2_genExp
- Filtered_all_CNVR_data_by_gene
- Filtered_clinical_PANCAN_patient_with_followup
- Filtered_EBpp_AdjustPANCAN_IlluminaHiSeq_RNASeqV2_genExp
- Filtered_jhu_usc_edu_PANCAN_HumanMethylation27_betaValue_whitelisted
- Filtered_jhu_usc_edu_PANCAN_HumanMethylation450_betaValue_whitelisted
- Filtered_jhu_usc_edu_PANCAN_merged_HumanMethylation27_HumanMethylation450_betaValue_whitelisted

- Filtered_MC3_MAF_V5_one_per_tumor_sample
- Filtered_pancanMiRs_EBadjOnProtocolPlatformWithoutRepsWithUnCorrectMiRs_08_04_16
- Filtered_TCGA_RPPA_pancan_clean
- jhu_usc_edu_PANCAN_HumanMethylation27_betaValue_whitelisted
- jhu_usc_edu_PANCAN_HumanMethylation450_betaValue_whitelisted
- jhu_usc_edu_PANCAN_merged_HumanMethylation27_HumanMethylation450_betaValue_whitelisted
- merged_sample_quality_annotations
- pancanMiRs_EBadjOnProtocolPlatformWithoutRepsWithUnCorrectMiRs_08_04_16
- TCGA_CDR
- TCGA_RPPA_pancan_clean
- Whitelist_ParticipantBarcodes

GDC data release 24.0 was released on May 7, 2020.

Updates to existing programs and projects

- 110 new cases were released from the HNSCC cohort of CPTAC-3. This includes WXS, WGS, RNA-Seq and miRNA-Seq data.
- Aliquot-level WXS MAFs are now available from the following projects: CPTAC-2 and CPTAC-3

BigQuery tables created

- isb-cgc:GDC_metadata.rel24_aliquot2caseIDmap
- isb-cgc:GDC_metadata.rel24_caseData
- isb-cgc:GDC_metadata.rel24_fileData_active
- isb-cgc:GDC_metadata.rel24_fileData_legacy
- isb-cgc:GDC_metadata.rel24_slide2caseIDmap

New programs and projects available in Google Cloud Storage

- New project released: CGCI-HTMCP-CC - HIV+ Tumor Molecular Characterization Project - Cervical Cancer
- RNA-Seq: Alignments and gene expression levels
- miRNA-Seq: Alignments and miRNA expression levels
- WGS: Alignments
- Targeted Sequencing: Alignments

New data sets and RNA Sequence tables derived data tables added to isb-cgc.

BigQuery tables created

- isb-cgc:BEATAML1_0.RNA_hg38_r19
- isb-cgc:ORGANOID.RNA_hg38_r18

May 8, 2020

GDC data release 23.0 was posted on April 7, 2020.

Updates to existing programs and projects

- HCMI-CMDC Aliquot-level MAFs were released

- TARGET-ALL-P2 Aliquot-level MAFs were released
- TARGET-ALL-P3 Aliquot-level MAFs were released
- TARGET-AML Aliquot-level MAFs were released
- TARGET-NBL Aliquot-level MAFs were released
- TARGET-OS Aliquot-level MAFs were released
- TARGET-WT Aliquot-level MAFs were released
- All TCGA Projects Copy number segment and estimate files from SNP6 ASCAT were released
- TARGET-ALL-P2 Copy number segment and estimate files from SNP6 ASCAT were released
- TARGET-AML Copy number segment and estimate files from SNP6 ASCAT were released
- HCMI-CMDC RNA-seq data was released
- CGCI-BLGSP clinical data was updated
- HCMI-CMDC clinical data was updated
- WCDT-MCRPC clinical data was updated

BigQuery tables created

- isb-cgc:GDC_metadata.rel23_aliquot2caseIDmap
- isb-cgc:GDC_metadata.rel23_caseData
- isb-cgc:GDC_metadata.rel23_fileData_active
- isb-cgc:GDC_metadata.rel23_fileData_legacy
- isb-cgc:GDC_metadata.rel23_slide2caseIDmap
- isb-cgc:GDC_metadata.rel23_GDCfileID_to_GCSurl

March 16, 2020

GDC data release 22.0 was posted on January 16, 2020.

New programs and projects available in Google Cloud Storage

- WCDT-MCRPC (Genomic Characterization of Metastatic Castration Resistant Prostate Cancer), RNA-Seq and WGS Data included

Updates to existing programs and projects

- HCMI-CMDC new RNA-Seq, WXS, WGS data was released.
- CPTAC-3 new WXS, WGS, and RNA-Seq data and miRNA-Seq data for currently released cases was released

BigQuery tables created

- isb-cgc:GDC_metadata.rel22_aliquot2caseIDmap
- isb-cgc:GDC_metadata.rel22_caseData
- isb-cgc:GDC_metadata.rel22_fileData_active
- isb-cgc:GDC_metadata.rel22_fileData_legacy
- isb-cgc:GDC_metadata.rel22_slide2caseIDmap
- isb-cgc:GDC_metadata.rel22_GDCfileID_to_GCSurl

January 11, 2020

GDC data release 21.0 was posted on December 10, 2019.

New programs and projects available in Google Cloud Storage

- GENIE-MDA
- GENIE-VICC
- GENIE-DFCI
- GENIE-MSK
- GENIE-UHN
- GENIE-JHU
- GENIE-GRCC
- GENIE-NKI

BigQuery tables created

- isb-cgc:GDC_metadata.rel21_aliquot2caseIDmap
- isb-cgc:GDC_metadata.rel21_caseData
- isb-cgc:GDC_metadata.rel21_fileData_active
- isb-cgc:GDC_metadata.rel21_fileData_legacy
- isb-cgc:GDC_metadata.rel21_slide2caseIDmap

December 20, 2019

GDC data release 19.0 was posted on September 17, 2019.

GDC data release 19.1 was posted on November 6, 2019.

New programs and projects available in Google Cloud Storage

- BEATAML1.0-COHORT (Functional Genomic Landscape of Acute Myeloid Leukemia) WXS and RNA-Seq data was included.

Updates to existing programs and projects

- TARGET-ALL-P1 new RNA-Seq data was released.
- TARGET-ALL-P2 new RNA-Seq, WXS, and miRNA-Seq data was released.
- TARGET-ALL-P3 new miRNA-Seq data was released.
- TARGET-AML new WXS and WGS data was released.
- TARGET-NBL new WXS and RNA-Seq data was released.
- TARGET-RT new WGS and RNA-Seq data was released.
- TARGET-WT new WGS, WXS, and RNA-Seq data was released.
- CGCI-BLGSP new WGS data was released.
- TARGET-ALL-P3 new Pindel VCFs was released.
- MMRF new Pindel VCFs was released.
- HCMI new Pindel VCFs was released.
- CPTAC-3 new Pindel VCFs was released.

- Disease-specific staging properties for many projects released.

BigQuery tables created

- isb-cgc:GDC_metadata.rel19_aliquot2caseIDmap
- isb-cgc:GDC_metadata.rel19_caseData
- isb-cgc:GDC_metadata.rel19_fileData_active
- isb-cgc:GDC_metadata.rel19_fileData_legacy
- isb-cgc:GDC_metadata.rel19_slide2caseIDmap

GDC data release 18 was posted on July 8, 2019.

New programs and projects available in Google Cloud Storage

- ORGANOID-PANCREATIC (Pancreas Cancer Organoid Profiling)
- MMRF-COMMPASS (Multiple Myeloma CoMMpass Study)
- CGCI-BLGSP (Burkitt Lymphoma Genome Sequencing Project)
- TARGET-ALL-P1 (Acute Lymphoblastic Leukemia - Phase I)
- TARGET-ALL-P2 (Acute Lymphoblastic Leukemia - Phase II)

Updates to existing programs and projects

- TARGET-ALL-P3 new RNA-Seq data was released.
- TARGET-CCSK new RNA-Seq data was released.
- TARGET-OS new RNA-Seq data was released.

BigQuery tables created

- isb-cgc:GDC_metadata.rel18_aliquot2caseIDmap
- isb-cgc:GDC_metadata.rel18_caseData
- isb-cgc:GDC_metadata.rel18_fileData_active
- isb-cgc:GDC_metadata.rel18_fileData_legacy
- isb-cgc:GDC_metadata.rel18_slide2caseIDmap

September 29, 2019

GDC data release 17.0 was posted on June 5, 2019.

GDC data release 17.1 was posted on June 12, 2019.

New programs and projects available in Google Cloud Storage

- HCMI-CMDC 500 files, 2.8TB
- BEATAML1.0-CRENOLANIB 700 files, 3.6TB

Updates to existing programs and projects

- CPTAC-3 RNA-Seq - 7400 files, 16.6 TB
- TCGA ATAC-Seq - 820 files, 9.2 TB
- NCICCR-DLBCL RNA-Seq - 2900 files, 11.9 TB
- CTSP-DLBCL1 RNA-Seq - 250 files, .96TB
- Updates to TCGA clinical data

- Migrations of three properties across all projects from diagnosis to demographic (vital_status, days_to_birth, days_to_death)

BigQuery tables created

- isb-cgc:GDC_metadata.rel17_aliquot2caseIDmap
- isb-cgc:GDC_metadata.rel17_caseData
- isb-cgc:GDC_metadata.rel17_fileData_active
- isb-cgc:GDC_metadata.rel17_fileData_legacy
- isb-cgc:GDC_metadata.rel17_slide2caseIDmap

April 4, 2019

GDC data release 16 was posted on March 26, 2019.

New programs and projects available in Google Cloud Storage

- CPTAC-3

BigQuery tables created

- isb-cgc:GDC_metadata.rel16_aliquot2caseIDmap
- isb-cgc:GDC_metadata.rel16_caseData
- isb-cgc:GDC_metadata.rel16_fileData_active
- isb-cgc:GDC_metadata.rel16_fileData_legacy
- isb-cgc:GDC_metadata.rel16_slide2caseIDmap

March 6, 2019

GDC data release 15 was posted on February 20, 2019.

New programs and projects available in Google Cloud Storage

- TARGET-ALL-P3

BigQuery tables created

- isb-cgc:GDC_metadata.rel15_aliquot2caseIDmap
- isb-cgc:GDC_metadata.rel15_caseData
- isb-cgc:GDC_metadata.rel15_fileData_current
- isb-cgc:GDC_metadata.rel15_fileData_legacy
- isb-cgc:GDC_metadata.rel15_slide2caseIDmap

January 4, 2019

GDC data release 14 was posted on December 18, 2018.

New programs and projects available in Google Cloud Storage

- FM-AD

BigQuery tables created

- isb-cgc:GDC_metadata.rel14_aliquot2caseIDmap
- isb-cgc:GDC_metadata.rel14_caseData
- isb-cgc:GDC_metadata.rel14_fileData_current

- isb-cgc:GDC_metadata.rel14_fileData_legacy
- isb-cgc:GDC_metadata.rel14_GDCfileID_to_GCSurl
- isb-cgc:GDC_metadata.rel14_GDCfileID_to_GCSurl_NEW
- isb-cgc:GDC_metadata.rel14_slide2caseIDmap
- isb-cgc:TCGA_hg38_data_v0.miRNAseq_Expression
- isb-cgc:TCGA_hg38_data_v0.miRNAseq_Isoform_Expression

October 2, 2018

GDC data release 13 was posted on September 27, 2018.

New programs and projects available in Google Cloud Storage

- VAREPOP-APOLLO
- CTSP-DLBCL1
- NCICCR-DLBCL

DR13, active archive contains 428,543 files (DR12 contained 356,381 files)

- 116 files were removed: 88 VCF files, 24 BAM files, and 2 miRNA "mirnas.quantification" files and (corresponding) 2 miRNA "isoforms.quantification" files.
- 72278 files were added: 47248 BAI files, 23203 TBI files, 1287 BAM files, 504 SEG files, 36 SVS files.

June 25, 2018:

GDC data release 12 was posted on Wednesday, June 13, 2018.

- There is absolutely no change in the legacy archive data between DR11 and DR12
- There is also no change in the total number of cases in either archive
- The number of files in the current archive has increased from 329,165 to 356,381:
- 67,220 files were removed
- 94,436 files were added

More details about the changes to the current archive of **TCGA** data:

Copy Number Variation | Genotyping Array | TXT files:

- 22376 Copy Number Segment files replaced (ie removed and added)
- 22376 Masked Copy Number Segment files replaced

Biospecimen | BCR XML files:

- 11294 files replaced

Clinical | BCR XML files:

- 11160 files removed / 11167 files added (ie 7 extra files)

Biospecimen | Diagnostic Slide | SVS files:

- 11730 Slide Image files added

Biospecimen | BCR SSF XML files:

- 10557 Biospecimen Supplement files added

Biospecimen | BCR Auxiliary XML files:

- 2884 Biospecimen Supplement files added

Clinical | BCR OMF XML files:

- 1051 Clinical Supplement files added

Biospecimen | BCR Biotab files:

- 340 Biospecimen Supplement files added

Clinical | BCR Biotab files:

- 226 Clinical Supplement files added

Simple Nucleotide Variation | WXS | VCF | Varscan2 files :

- 1 Raw Simple Somatic Mutation file removed (2017-03-04)
- 1 Annotated Somatic Mutation file removed (2017-06-17)

Both for ESCA samples:

TCGA-VR-A8ET-01A-11D-A403-09;TCGA-VR-A8ET-10B-01D-A403-09

For **TARGET** data:

RNA-Seq data:

- 3 BAM files and 9 Gene Expression Quantification files removed
- Sample barcodes: TARGET-30-PAKYZS-01A-01R, TARGET-30-PAMEZH-01A-01R, TARGET-30-PANRRW-01A-01R
- Raw CGI Variant | WGS | Combined Nucleotide Variation | VCF files:435 files added

June 4, 2018:

The metadata tables for GDC data release 11 are now available in BigQuery.

May 8, 2018:

The gnomAD database (release 2.0.2, dated October 2017) is now available in BigQuery! **isb-cgc:genome_reference.gnomAD_20171003_GRCh37**.

April 30, 2018:

Recently released (2018-04-01) ClinVar VCFs are now available in BigQuery! Two new tables (**ClinVar_20180401_GRCh37** and **ClinVar_20180401_GRCh38**) can be found in our genome_reference dataset; also available is dbSNP build 151 (announced 2018-04-24): **isb-cgc:genome_reference.dbSNP_b151_GRCh37p13_All**.

February 22, 2018:

A genenames_mapping table has been added to our numerous reference sources in BigQuery to simplify mapping between HGNC IDs, HGNC symbols, Entrez Gene IDs, Ensembl Gene IDs, Pubmed IDs, and RefSeq IDs!

June 9, 2018:

The metadata tables for GDC data release 10 are now available in BigQuery.

May 8, 2018:

The release 85 of the **COSMIC** database is now available in BigQuery.

February 13, 2018:

The release 84 of the **COSMIC** database is now available in BigQuery.

December 19, 2017:

The ISB-CGC cohort metadata has been update to reflect the new and update TARGET gene expression data provided by the GDC in their data release 9.

December 6, 2017:

The GDC release 9 included some updated and new TARGET gene expression data. The BigQuery table **isb-cgc:TARGET_hg38_data_v0.RNAseq_Gene_Expression** has been updated to reflect this.

November 7, 2017:

The release 83 of the **COSMIC** database is now available in BigQuery.

November 3, 2017:

The metadata tables for GDC data release 9 are now available in BigQuery.

October 30, 2017:

The 'harmonized' hg38 TCGA VCF files (raw and annotated) are now available in the ISB-CGC controlled-data repository in Google Cloud Storage.

August 30, 2017:

The hg38 TARGET VCF files (raw and annotated) are now available in the ISB-CGC controlled-data repository in Google Cloud Storage.

August 3, 2017:

Release 82 of the **COSMIC** database is now available in BigQuery.

June 30, 2017:

The genome sequence hg19 and hg38 TARGET WXS, RNA-Seq, and miRNA-Seq BAM files are now available in the ISB-CGC controlled-data repository in Google Cloud Storage.

May 9, 2017:

Release 81 of the COSMIC database is now available in BigQuery.

May 5, 2017:

A table mapping between UniProtKB accessions and identifiers has been added to our reference dataset: **isb-cgc:genome_reference.UniProtKB_idmapping**.

April 10, 2017:

We have re-organized our TCGA clinical, biospecimen, and molecular data into new datasets in BigQuery.

Please find them below:

- isb-cgc:TCGA_bioclin_v0
- isb-cgc:TCGA_hg19_data_v0
- isb-cgc:TCGA_hg38_data_v0

The hg19 data can also be found in the GDC's legacy archive, while the hg38 data is available at the GDC data portal.

March 30, 2017:

The 'harmonized' hg38 TCGA miRNA-Seq BAM files from the initial GDC data release are now available in the ISB-CGC controlled-data repository in Google Cloud Storage.

February 20, 2017:

In collaboration with the Sanger Institute, the COSMIC database is now available in BigQuery (registered users only).

February 5, 2017:

Genomic coordinates (in GFF3 format) for human microRNAs added for miRBase v20 and v21 to the **isb-cgc:genome_reference** BigQuery dataset.

January 30, 2017:

The final, unified "MC3" TCGA somatic mutations call set is available in the BigQuery. **isb-cgc:hg19_data_previews** dataset (also available on Synapse).

January 10, 2017:

miRBase_v20 table added to the **isb-cgc:genome_reference** BigQuery dataset.

January 4, 2017:

Ensembl gene-set releases 75 (GRCh37) and 87 (GRCh38) are now also available in the **isb-cgc:genome_reference** BigQuery dataset.

December 30, 2016:

The 'harmonized' hg38 TCGA WXS BAM files and RNA-Seq BAM files from the initial GDC data release (1.0), as well as the legacy hg19. TCGA 'Level 2' Genome-Wide SNP6 array genotype files ('birdseed') files are now available in the ISB-CGC controlled-data repository in Google Cloud Storage.

November 14, 2016:

TCGA radiology and tissue slide images are now available in Google Cloud Storage! This includes radiology images (DICOM files) from the Cancer Imaging Archive (TCIA) and tissue slide images from the NCI-GDC data portal (SVS files).

November 16, 2016:

TCGA proteomics data from the CPTAC (Phase II) is now available in Google Cloud Storage.

September 10, 2016:

GENCODE versions 19, 22, 23, and 24 are all now available in the **isb-cgc:genome_reference** BigQuery dataset, with an updated and more complete schema. – Note also that the naming convention is now **GENCODE_v19** rather than GENCODE_r19; also that v19 is the *last* version based on hg19/GRCh37, and all subsequent versions are based on hg38/GRCh38.

August 31, 2016:

A table based on the latest liftOver hg19-to-hg38 chain files is available in the **isb-cgc:tcga_genome_reference** BigQuery dataset.

August 26, 2016:

A set of tables based on running Picard over ~67,000 TCGA bam files in GCS have been added to the **isb-cgc:tcga_seq_metadata** BigQuery dataset: information contained in these tables includes bam-index stats, insert-size metrics, quality-distribution metrics, and quality-yield metrics – these tables can be used in conjunction with the FastQC-based tables to look for bam and/or fastq data files that meet your analysis criteria.

August 21, 2016:

New **miRBase_v21** table added to the **isb-cgc:genome_reference** BigQuery dataset.

August 20, 2016:

Updated **hg19** and **hg38** Kaviar tables added to the **isb-cgc:genome_reference** BigQuery dataset.

August 17, 2016:

New **isb-cgc:GDC_metadata** BigQuery dataset containing metadata for both *legacy* and *current* files hosted at the NCI-GDC.

July 28, 2016:

New **isb-cgc:tcga_201607_beta** BigQuery dataset based on the *final* TCGA data upload from the DCC. This dataset largely mirrors the previous **isb-cgc:tcga_20510_alpha** dataset and is now also supporting the ISB-CGC Web-App. The curated TCGA cohort tables in the **isb-cgc:tcga_cohorts** BigQuery dataset have also been updated.

June 24, 2016:

An updated listing of all ISB-CGC hosted data in Google Cloud Storage (GCS) is now available in the **GCS_listing_24jun2016** table in the **isb-cgc:tcga_seq_metadata** dataset in BigQuery, in addition the **CGHub_Manifest_24jun2016** table contains the final CGHub Manifest prior to the transition of all data to the Genomic Data Commons.

June 18, 2016:

New **GENCODE_r24** table added to the **isb-cgc:genome_reference** BigQuery dataset.

May 13, 2016:

New **NCBI_Viral_Annotations_Taxid10239** table added to the **isb-cgc:genome_reference** BigQuery dataset.

May 9, 2016:

New **Ensembl2Reactome** and **miRBase2Reactome** tables added to the **isb-cgc:genome_reference** BigQuery dataset.

May 3, 2016:

New **isb-cgc:tcga_seq_metadata** BigQuery dataset contains metadata and FastQC metrics for thousands of TCGA DNA-seq and RNA-seq data files: - **CGHub_Manifest** table contains metadata for all TCGA files at CGHub as of April 27th, 2016 - **GCS_listing_27apr2016** table contains metadata for all TCGA files hosted by ISB-CGC in GCS - **RNAseq_FastQC** table contains metrics derived from FastQC runs on the RNAseq data files, including urls to the FastQC html reports that you can cut and paste directly into your browser - **WXS_FastQC** table contains metrics derived from FastQC runs on the exome DNAseq data files

April 28, 2016

GO_Ontology and **GO_Annotations** tables added to the **isb-cgc:genome_reference** BigQuery dataset.

March 14, 2016

With the release of our **Web-App**, controlled-data is now accessible (programmatically) to users who have previously obtained dbGaP approval for TCGA data and go through the NIH authentication process built-in to the Web-App.

February 26, 2016

New CCLE dataset in BigQuery **isb-cgc:ccle_201602_alpha** includes sample metadata, mutation calls, copy-number segments, and expression data (metadata includes full cloud-storage-path for world-readable BAM and SNP CEL files, and Genomics dataset- and readgroupset-ids for sequence data imported into Google Genomics).

February 22, 2016

Kaviar database now available in the **isb-cgc:genome_reference** BigQuery dataset.

February 19, 2016

CCLE RNAseq and DNAseq bam files imported into **Google Genomics**.

January 10, 2016

GENCODE_r19 and **miRBase_v20** tables added to the **isb-cgc:genome_reference** BigQuery dataset.

December 26, 2015

Public release of new **isb-cgc:genome_reference** BigQuery dataset: the first table is based on the just-published **miRTarBase** release 6.1.

December, 12, 2015

Curated TCGA cohort lists available in **isb-cgc:tcga_cohorts** BigQuery dataset.

December 3, 2015

Version v0.1.

First tagged release of the web-app.

November 16, 2015

Initial upload of data from CGHub into **Google Cloud Storage** (GCS) complete (not publicly released).

November 2, 2015

First public release of TCGA open-access data in BigQuery tables.

- **isb-cgc:tcga_201510_alpha** dataset contains updated set of BigQuery tables, based on data available at the TCGA DCC as of October 2015
- Includes **Annotations** table with information about redacted samples, etc
- **isb-cgc:platform_reference** contains annotation information for the Illumina DNA Methylation platform

October 4, 2015

Complete data upload from TCGA DCC, including controlled-access data

September 21, 2015

Draft set of BigQuery tables (not publicly released)

- **isb-cgc:tcga_201507_alpha** dataset containing clinical, biospecimen, somatic mutation calls and Level-3 TCGA data available at the TCGA DCC as of July 2015

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

27.2 ISB-CGC Migration to Project isb-cgc-bq Release Notes

December 15, 2020

Existing TCGA hg19 DNA Methylation tables in the isb-cgc project were copied to the isb-cgc-bq project, TCGA and TCGA_versioned data sets. Corresponding TCGA tables in the isb-cgc project were deprecated.

BigQuery tables created

- isb-cgc-bq.TCGA.DNA_methylation_chrY_hg19_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chrY_hg19_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chrX_hg19_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chrX_hg19_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr22_hg19_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr22_hg19_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr21_hg19_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr21_hg19_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr20_hg19_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr20_hg19_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr19_hg19_gdc_current

- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr19_hg19_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr18_hg19_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr18_hg19_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr17_hg19_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr17_hg19_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr16_hg19_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr16_hg19_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr15_hg19_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr15_hg19_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr14_hg19_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr14_hg19_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr13_hg19_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr13_hg19_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr12_hg19_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr12_hg19_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr11_hg19_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr11_hg19_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr10_hg19_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr10_hg19_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr9_hg19_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr9_hg19_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr8_hg19_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr8_hg19_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr7_hg19_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr7_hg19_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr6_hg19_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr6_hg19_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr5_hg19_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr5_hg19_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr4_hg19_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr4_hg19_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr3_hg19_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr3_hg19_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr2_hg19_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr2_hg19_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr1_hg19_gdc_current

- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr1_hg19_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_hg19_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_hg19_gdc_2017_01

December 11, 2020

Existing TCGA hg38 DNA Methylation tables in the isb-cgc project were copied to the isb-cgc-bq project, TCGA and TCGA_versioned data sets. Corresponding TCGA tables in the isb-cgc project were deprecated.

BigQuery tables created

- isb-cgc-bq.TCGA.DNA_methylation_chrY_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chrY_hg38_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chrX_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chrX_hg38_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr22_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr22_hg38_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr21_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr21_hg38_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr20_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr20_hg38_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr19_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr19_hg38_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr18_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr18_hg38_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr17_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr17_hg38_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr16_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr16_hg38_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr15_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr15_hg38_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr14_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr14_hg38_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr13_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr13_hg38_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr12_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr12_hg38_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr11_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr11_hg38_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr10_hg38_gdc_current

- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr10_hg38_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr9_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr9_hg38_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr8_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr8_hg38_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr7_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr7_hg38_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr6_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr6_hg38_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr5_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr5_hg38_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr4_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr4_hg38_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr3_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr3_hg38_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr2_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr2_hg38_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_chr1_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_chr1_hg38_gdc_2017_01
- isb-cgc-bq.TCGA.DNA_methylation_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.DNA_methylation_hg38_gdc_2017_01

December 4, 2020

Existing pancancer_atlas tables in the isb-cgc project were copied to the isb-cgc-bq project. Corresponding tables in the isb-cgc project were deprecated.

BigQuery tables created

The following tables were created under the isb-cgc-bq:pancer-altas data set:

- Original_TCGA_RPPA_pancan_clean
- Original_pancanMiRs_EBadjOnProtocolPlatformWithoutRepsWithUnCorrectMiRs_08_04_16
- Original_jhu_usc_edu_PANCAN_merged_HumanMethylation27_HumanMethylation450_betaValue_whitelisted
- Original_jhu_usc_edu_PANCAN_HumanMethylation450_betaValue_whitelisted
- Original_jhu_usc_edu_PANCAN_HumanMethylation27_betaValue_whitelisted
- Original_EBpp_AdjustPANCAN_IlluminaHiSeq_RNASeqV2_genExp
- Original_clinical_PANCAN_patient_with_followup
- Filtered_TCGA_RPPA_pancan_clean
- Filtered_pancanMiRs_EBadjOnProtocolPlatformWithoutRepsWithUnCorrectMiRs_08_04_16
- Filtered_MC3_MAF_V5_one_per_tumor_sample

- Filtered_jhu_usc_edu_PANCAN_merged_HumanMethylation27_HumanMethylation450_betaValue_whitelisted
- Filtered_jhu_usc_edu_PANCAN_HumanMethylation450_betaValue_whitelisted
- Filtered_jhu_usc_edu_PANCAN_HumanMethylation27_betaValue_whitelisted
- Filtered_EBpp_AdjustPANCAN_IlluminaHiSeq_RNASeqV2_genExp
- Filtered_clinical_PANCAN_patient_with_followup
- Filtered_all_CNVR_data_by_gene
- merged_sample_quality_annotations
- TCGA_CDR
- Whitelist_ParticipantBarcodes
- BarcodeMap

November 13, 2020

Existing methylation annotation and liftover tables in the isb-cgc project were copied to the isb-cgc-bq project. Corresponding tables in the isb-cgc project were deprecated.

BigQuery tables created

- isb-cgc-bq.annotations.methylation_annotation_hg19_illumina_current
- isb-cgc-bq.annotations_versioned.methylation_annotation_hg19_illumina_2015_06
- isb-cgc-bq.annotations.methylation_annotation_hg38_gdc_current
- isb-cgc-bq.annotations_versioned.methylation_annotation_hg38_gdc_2016_11
- isb-cgc-bq.annotations.liftover_hg19_to_hg38_current
- isb-cgc-bq.annotations_versioned.liftover_hg19_to_hg38_2016_08
- isb-cgc-bq.annotations.methylation_liftover_hg19_illumina_to_hg38_current
- isb-cgc-bq.annotations_versioned.methylation_liftover_hg19_illumina_to_hg38_2016_08

November 9, 2020

Existing GENCODE tables in the isb-cgc project were copied to the isb-cgc-bq project. Corresponding GENCODE tables in the isb-cgc project were deprecated.

BigQuery tables created

- isb-cgc-bq.GENCODE_versioned.annotation_gtf_hg19_v19
- isb-cgc-bq.GENCODE_versioned.annotation_gtf_hg38_v22
- isb-cgc-bq.GENCODE_versioned.annotation_gtf_hg38_v23
- isb-cgc-bq.GENCODE_versioned.annotation_gtf_hg38_v24

October 22, 2020

Existing TARGET tables in the isb-cgc project (data sets TARGET_bioclin_v0 and TARGET_bioclin_v0) were copied to the isb-cgc-bq project, TARGET and TARGET_versioned data sets. Corresponding TARGET tables in the isb-cgc project were deprecated.

BigQuery tables created

- isb-cgc-bq.TARGET_versioned.per_sample_file_metadata_hg38_gdc_r14
- isb-cgc-bq.TARGET_versioned.miRNAseq_isoform_hg38_gdc_r11

- isb-cgc-bq.TARGET.miRNAseq_isoform_hg38_gdc_current
- isb-cgc-bq.TARGET_versioned.miRNAseq_isoform_hg38_gdc_r14
- isb-cgc-bq.TARGET_versioned.miRNAseq_hg38_gdc_r11
- isb-cgc-bq.TARGET_versioned.miRNAseq_hg38_gdc_r14
- isb-cgc-bq.TARGET_versioned.RNAseq_hg38_gdc_2017_12
- isb-cgc-bq.TARGET.biospecimen_gdc_current
- isb-cgc-bq.TARGET_versioned.biospecimen_gdc_2017_04
- isb-cgc-bq.TARGET_versioned.clinical_gdc_2019_06
- isb-cgc-bq.TARGET_versioned.clinical_gdc_2017_04

October 07, 2020

Existing TCGA tables (except for DNA Methylation) in the isb-cgc project were copied to the isb-cgc-bq project, TCGA and TCGA_versioned data sets. Corresponding TCGA tables in the isb-cgc project were deprecated.

BigQuery tables created

- isb-cgc-bq.TCGA.annotations_gdc_current
- isb-cgc-bq.TCGA_versioned.annotations_gdc_2017_04
- isb-cgc-bq.TCGA.biospecimen_gdc_current
- isb-cgc-bq.TCGA_versioned.biospecimen_gdc_2017_02
- isb-cgc-bq.TCGA_versioned.clinical_gdc_2018_06
- isb-cgc-bq.TCGA_versioned.clinical_gdc_2019_06
- isb-cgc-bq.TCGA.slide_images_gdc_current
- isb-cgc-bq.TCGA_versioned.slide_images_gdc_r17
- isb-cgc-bq.TCGA.radiology_images_tcia_current
- isb-cgc-bq.TCGA_versioned.radiology_images_tcia_2018_06
- isb-cgc-bq.TCGA_versioned.per_sample_file_metadata_hg19_gdc_r14
- isb-cgc-bq.TCGA_versioned.per_sample_file_metadata_hg38_gdc_r14
- isb-cgc-bq.TCGA_versioned.somatic_mutation_hg19_MC3_2017_02
- isb-cgc-bq.TCGA_versioned.somatic_mutation_hg19_DCC_2017_02
- isb-cgc-bq.TCGA_versioned.somatic_mutation_hg38_gdc_r6
- isb-cgc-bq.TCGA_versioned.somatic_mutation_hg38_gdc_r7
- isb-cgc-bq.TCGA.somatic_mutation_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.somatic_mutation_hg38_gdc_r10
- isb-cgc-bq.TCGA.miRNAseq_hg19_gdc_current
- isb-cgc-bq.TCGA_versioned.miRNAseq_hg19_gdc_2017_03
- isb-cgc-bq.TCGA.miRNAseq_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.miRNAseq_hg38_gdc_r14
- isb-cgc-bq.TCGA.protein_expression_hg19_gdc_current

- isb-cgc-bq.TCGA_versioned.protein_expression_hg19_gdc_2017_02
- isb-cgc-bq.TCGA.protein_expression_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.protein_expression_hg38_gdc_2017_02
- isb-cgc-bq.TCGA.miRNAseq_isoform_hg19_gdc_current
- isb-cgc-bq.TCGA_versioned.miRNAseq_isoform_hg19_gdc_2017_02
- isb-cgc-bq.TCGA.miRNAseq_isoform_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.miRNAseq_isoform_hg38_gdc_r14
- isb-cgc-bq.TCGA.RNAseq_hg19_gdc_current
- isb-cgc-bq.TCGA_versioned.RNAseq_hg19_gdc_2017_02
- isb-cgc-bq.TCGA.RNAseq_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.RNAseq_hg38_gdc_2017_12
- isb-cgc-bq.TCGA_versioned.copy_number_segment_masked_hg38_gdc_2017_02
- isb-cgc-bq.TCGA.copy_number_segment_masked_hg19_gdc_current
- isb-cgc-bq.TCGA_versioned.copy_number_segment_masked_hg19_gdc_2017_02
- isb-cgc-bq.TCGA.copy_number_segment_masked_hg38_gdc_current
- isb-cgc-bq.TCGA_versioned.copy_number_segment_masked_hg38_gdc_r14

September 3, 2020

Existing CCLE tables in the isb-cgc project were copied to the isb-cgc-bq project, CCLE and CCLE_versioned data sets. Corresponding CCLE tables in the isb-cgc project were deprecated.

BigQuery tables created

- isb-cgc-bq.CCLE_versioned.clinical_2019_06
- isb-cgc-bq.CCLE.clinical_current
- isb-cgc-bq.CCLE_versioned.biospecimen_2019_04
- isb-cgc-bq.CCLE.biospecimen_current
- isb-cgc-bq.CCLE_versioned.sample_information_hg19_2016_02
- isb-cgc-bq.CCLE.sample_information_hg19_current
- isb-cgc-bq.CCLE_versioned.RMA_expression_hg19_2016_02
- isb-cgc-bq.CCLE.RMA_expression_hg19_current
- isb-cgc-bq.CCLE_versioned.copy_number_segment_hg19_2016_02
- isb-cgc-bq.CCLE.copy_number_segment_hg19_current
- isb-cgc-bq.CCLE_versioned.somatic_mutation_hg19_2016_02
- isb-cgc-bq.CCLE.somatic_mutation_hg19_current
- isb-cgc-bq.CCLE_versioned.file_metadata_hg19_2016_03
- isb-cgc-bq.CCLE_versioned.fastqc_metrics_hg19_2016_03
- isb-cgc-bq.CCLE.fastqc_metrics_hg19_current
- isb-cgc-bq.CCLE_versioned.per_sample_file_metadata_hg19_gdc_r14

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

27.3 ISB-CGC BigQuery Table Search Release Notes

To learn about this discovery tool created by the ISB-CGC, please visit ISB-CGC BigQuery Table Search.

For more detailed information about the data stored in ISB-CGC BigQuery tables please visit ISB-CGC BigQuery Tables.

September 8, 2021

New Features

On the Search results, added an "Example Joins" column. This column specifies the number of example join queries, for the table on that row, which are provided by the BigQuery Table Search.

Functionality includes:

- Click the number in the "Example Joins" column to see a list of examples.
- From there, click on View Details for a particular example to see the SQL Query and a longer description.
- On the View Details screen, click on COPY to copy the query to your clipboard.

December 8, 2020 v1.04

New Features

- On the Filter panel, under Show More Filters, a filter BQ Project has been added. It has also been added to the Column selection dropdown list.

Bug Fixes

- On multi-select filters (Program, Source, Data Type, Experimental Strategy), the X button to delete the selected value did not completely display. It has been fixed so that the entire X button displays.

July 23, 2020 v1.03

New Features

- The Access filter has been added, which has options of All, Open Access and Controlled Access. Controlled Access data cannot be previewed, but can be opened in the Google BigQuery Console, if the user has the required permissions.

March 11, 2020 v1.02

New Features

- Users now have the ability to access, query and inspect in detail BigQuery tables in Google Cloud Platform's BigQuery console directly from the Table Search UI. Every table has a "Open" option which when clicked will send the user to the table in the BigQuery console on the Google Cloud.
- An Open button, with the same functionality as described above, has been added to the Schema Description section.
- Program and Experimental Strategy filters were added.
- Values for the Data Type and Source filters have been modified in order to align more closely with GDC naming conventions.

January 30,2020 v1.01

New Features

- A "Name" column consisting of user-friendly descriptive names for the BigQuery tables has been introduced.
- The Name filter, a free-form text search field is now available allowing users to search for all or a portion of the user-friendly descriptive names.
- Columns can be now added or removed from the display by using the Columns selector option.
- By default, Dataset ID and Table ID are no longer initially displayed in the full column view, but can be added to the display using the columns selector.
- The Full ID, which is denoted [projectID.datasetID.TableID] (concatenation of the project ID, dataset ID and the Table ID, each separated by a period symbol) is listed under the detailed table information section found after clicking on the blue plus sign.
- A Copy button, found adjacent to the Full ID has been added. The Full ID adheres to BigQuery Standard SQL format and contains the necessary grave accents (`) required for executing SQL queries in BigQuery. When copied to the clipboard, the Full ID can be directly used to run queries in BigQuery Query Editor without any further manual modifications.

Enhancements

- Individual table schemas captured by the "Fields" column in the CSV download now contain field information in comma-separated format.

November 26, 2019 v1.0

Initial Release

The ISB-CGC BigQuery Table Search UI is a discovery tool that allows users to explore and search for ISB-CGC hosted BigQuery tables. It can be accessed directly from the ISB-CGC homepage.

Major features in the initial release include:

- The ability to search for BigQuery tables by multiple filters:
- Status
- Categories
- Reference Genome Build
- Source
- Data Type
- Dataset ID
- Table ID
- Table Description
- Labels
- Field Name
- Display of search results in a tabular format, with the following information about BigQuery tables:
- Dataset ID
- Table ID
- Status
- Source

- Data Type
- Num Rows
- Created Date
- Detailed schema information for each table, including full table ID, table description, and field descriptions.
- The ability to preview the first eight rows in the BigQuery table of choice.
- The ability to download a CSV format file of search results.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

27.4 Mitelman Database Release Notes

To search this database hosted by the ISB-CGC, please visit Mitelman Database of Chromosome Aberrations and Gene Fusions in Cancer.

For more detailed information about the Mitelman Database of Chromosome Aberrations and Gene Fusions in Cancer, see Mitelman Database.

July 27, 2022

Mitelman Database of Chromosome Aberrations and Gene Fusions in Cancer quarterly update.

Updated totals

- Total number of cases 73,930
- Total number of unique gene fusions 33,457
- Total number of genes involved 14,061

June 6, 2022

Enhancements and New Features

Mitelman Database Now Includes Genomic Coordinates

Until June 2022, the resulting genetic location information retrieved from the database was only displayed in karyotypes. Now, genomic coordinates are also displayed. Thanks to procedures incorporated from the web-based tool CytoConverter, karyotypes are converted to genomic coordinates and can be optionally viewed by the Mitelman Database user.

The user has the option of viewing the genomic coordinate information for either individual karyotypes or for multiple karyotypes in a search result. For individual karyotypes, the corresponding chromosome and its start and end position are given. In addition, the type of imbalance (gain or loss) is noted. For multiple karyotypes in the search results, net imbalances across the selected group are displayed in chart, ideogram or tabular format; information includes the chromosome affected, start and end positions, and whether the segment has been lost or gained.

April 18, 2022

Mitelman Database of Chromosome Aberrations and Gene Fusions in Cancer quarterly update.

Updated totals

- Total number of cases 72,718
- Total number of unique gene fusions 32,962
- Total number of genes involved 14,016

January 18, 2022

Mitelman Database of Chromosome Aberrations and Gene Fusions in Cancer quarterly update.

Updated totals

- Total number of cases 72,421
- Total number of unique gene fusions 32,855
- Total number of genes involved 14,022

Enhancements and New Features

According to the recent recommendations of The Hugo Genome Nomenclature Committee (HGNC), the designations of all fusion genes have been changed from forward slash (/) to double colon (::). This affects the searches: "Gene Fusions", Clinical Associations", and "Recurrent Chromosome Aberrations".

October 15, 2021

Mitelman Database of Chromosome Aberrations and Gene Fusions in Cancer quarterly update.

Updated totals

- Total number of cases 72,105
- Total number of unique gene fusions 32,795
- Total number of genes involved 14,023

Enhancements and New Features

- Removed the size limit on the search results: User can perform blank searches to retrieve the full data.
- View SQL Statements: User can view and utilize the new SQL statement that was used to perform the search.

July 15, 2021

Mitelman Database of Chromosome Aberrations and Gene Fusions in Cancer quarterly update.

Updated totals

- Total number of cases 71,734
- Total number of unique gene fusions 32,721
- Total number of genes involved 14,019

Enhancements

Security enhancement (including Data Tables package version update)

April 15, 2021

Mitelman Database of Chromosome Aberrations and Gene Fusions in Cancer quarterly update.

Updated totals

- Total number of cases 71,298
- Total number of unique gene fusions 32,677
- Total number of genes involved 14,020

Bug Fixes

Gene Fusion Search failed to return gene fusion results if searched by gene names with a hyphen ('-') in it (e.g. ARPC4-TTLL3). This has been fixed.

January 15, 2021

Mitelman Database of Chromosome Aberrations and Gene Fusions in Cancer quarterly update.

Updated totals

- Total number of cases 71,149
- Total number of unique gene fusions 32,618
- Total number of genes involved 14,016

October 26, 2020

Bug Fixes

Cases Cytogenetics Searcher: Using 'Sole Abnormality' flag with a 'Breakpoint' entry will now search cases with karyotypes of sole abnormality with the specified breakpoint.

October 15, 2020

Mitelman Database of Chromosome Aberrations and Gene Fusions in Cancer quarterly update.

Updated totals

- Total number of cases 70,818
- Total number of unique gene fusions 32,578
- Total number of genes involved 14,014

July 15, 2020

Mitelman Database of Chromosome Aberrations and Gene Fusions in Cancer quarterly update.

Updated totals

- Total number of cases 70,469
- Total number of unique gene fusions 32,551
- Total number of genes involved 14,014

April 15, 2020

Mitelman Database of Chromosome Aberrations and Gene Fusions in Cancer quarterly update.

Updated totals

- Total number of cases 70,236
- Total number of unique gene fusions 31,626
- Total number of genes involved 13,913

Other changes

- New Mitelman Database Logo

August 27, 2019

Initial Release

- Mitelman Database of Chromosome Aberrations and Gene Fusions in Cancer released on the ISB-CGC platform.

The following searches are available:

- Cases Cytogenetics Searcher
- Gene Fusions Searcher
- Clinical Associations Searcher

- Recurrent Chromosome Aberrations Searcher
- References Searcher

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

27.5 The *TP53* Database Release Notes

The *TP53* Database compiles various types of data and information from the literature and generalist databases on human *TP53* gene variations related to cancer. ISB-CGC started hosting this database on October 25, 2021.

- **Database Release Notes** are on The *TP53* Database application About page.
- **Application Release Notes** are on The *TP53* Database application Release Notes page.

To search this database hosted by the ISB-CGC, please visit The *TP53* Database at https://tp53.isb-cgc.org.

For more detailed information about The *TP53* Database, see the documentation.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

27.6 ISB-CGC Web App Release Notes

September 8, 2021

- On the ISB-CGC home page, the banner with the link to the survey has been removed.
- On the Tutorials for Workflow on Google Cloud page, a GeneFlow RNA-seq workflow has been added. (From the ISB-CGC home page, on the Pipelines and APIs box, click Launch. Then on the displayed Pipelines and APIs page, click the Tutorials for Workflow on Google Cloud box.)
- From the Web App Programs page, remove the ability for the user to upload their own program.
- On the Create Cohort – Filters page:
  - Add a Clear All option to the Cohort Filters panel.
  - Previously, if all the filters under a program name were removed, the program name remained in the Cohort Filters box. It has been changed so that if the last filter for the program is removed, the program name is also removed from the Cohort Filters panel.
- On the ISB-CGC BigQuery Table Search, on the search results, added an "Example Joins" column. This column specifies the number of example join queries, for the table on that row, which are provided by the BigQuery Table Search. Functionality includes:
  - Click the number in the "Example Joins" column to see a list of examples.
  - From there, click on View Details for a particular example to see the SQL Query and a longer description.
  - On the View Details screen, click on COPY to copy the query to your clipboard.

July 19, 2021

- The Warning Notice about accessing a government website that should pop up when ISB-CGC is accessed was missing. It has been reinstated.
- There was an issue loading the Variable selection page. This has been fixed.

- On the Create and Edit Variable screens, the data set/programs tabs have been changed to a drop down.
- In some cases when creating a cohort with multiple programs, the Data Set Clinical Features panel was not displaying the appropriate information. This has been corrected.
- The system now logs when users refresh their 24 hour access at DCF so that this can be monitored.

June 24, 2021

New Features

- The Citations link on the top-level menu has been replaced with a Publications link. The page now includes ISB-CGC publications and posters, as well as citations.
- Enhance the Cohort Builder UI and Cohort Details UI so that additional node based GDC and PDC Programs can be selected and edited. The Programs now available are:
  - GDC
    - TCGA
    - TARGET
    - CCLE
    - BEATAML1.0
    - FM
    - MMRF
    - OHSU
  - PDC
    - Georgetown Proteomics (GPRP)
- On the Cohort Builder, replace the Selected Filters panel with the Cohort Filters panel. This panel displays all selected filters for all selected Programs.

April 13, 2021

New Features

- On the home page, the title in each box is now clickable and will bring the user to that function.
- A Citations link has been added to the top level menu. Papers which reference ISB-CGC are listed.
- In the Cancer Data File Browser, when viewing a cohort that only has HG19 data, the Build filter will automatically be set to HG19.

Bug Fixes

- Fixed the following issues in the Cancer Data File Browser:
  - When selecting Build of HG19 and Data Format of Zip, incorrect results were displayed.
  - When selecting Build of HG19 and Program of TCGA, the incorrect number of entries was displayed under the File Listing.
  - When selecting Build of HG19 and Data Format of Raw sequencing data, the number of entries displayed under the File Listing was off by one.

February 22, 2021

New Features

- In the Cohort Builder (via filters), there is a new option to hide filter attributes with zero counts.

- In the Cancer Data File Browser - Pathology Image Viewer, Pathology Report Viewer and Radiology Image Viewer, login is no longer required.
- A survey link has been added to the ISB-CGC home page.
- On the home page, the icons next to the title in each box are now clickable and will bring the user to that function.
- Security update.

Bug Fixes

- In the Cancer Data File Browser - Radiology Image Viewer, error handing for records with missing disease code or project short name has been added.
- In the Cancer Data File Browser, the display of the 'Next' button for paginations has been fixed.
- In the Cohort Builder (via filters), inaccurate filter and sample counts for numeric range type filters (e.g. Age at diagnosis) were corrected.

December 8, 2020

New Features

- In the Cancer Data File Browser, BEATAML1.0 has been added as a choice under filter Program Name. It has also been added to the Cohort Builder/Data Explorer.
- On the ISB-CGC home page, a Contact Us page has been added under the Help dropdown menu item.
- On the ISB-CGC home page, a step by step guide called **How to Discover Cancer Data through ISB-CGC** has been added.
- There is a registration maximum of six programs/datasets to a service account implementation from the DCF.
- Bootstrap library upgraded from version 3.3.1 to 3.4.1.

Known Issues

- Work is underway to rework our cohort creation page to better display images associated with samples.
- The user data upload feature will return an error message stating, "Error submitting response : Could not connect to data upload server."
- Analysis Type: Seq peek Formatting is Elongated on occasion.
- If the user shares a Cohort, neither the owner nor the person who was granted access to the Cohort will receive a confirmation email when sharing a Cohort.
- CCLE data cannot be plotted when working with workbooks. ISB-CGC will resolve this functionality after the GDC formally releases CCLE data.
- When a user duplicates a Worksheet, then tries to implement the log scale, it will not function properly.
- The set operation for existing Cohorts complement is behaving exceptionally slow.
- The mouse-over feature is currently disabled for program TARGET with disease code ALL.
- When uploading TARGET files using the cohort barcode creation feature from the GDC, you may get an invalid barcode error message and unable to upload all the barcodes.
- On the File Browser page for Diagnostic images, there is no GDC file UUID associated with them.
- Sharing a workbook with someone else will cause the analysis to reset.

August 19, 2020

New Features

- ISB-CGC has a new home page, which prominently features ISB-CGC Data Browsers and Resources.

- A Cancer Data File Browser is now available directly from the ISB-CGC home page. It is similar to the existing File Browser within the Web App, except:
  - Sign in is not needed.
  - It is not dependent on cohorts built through the Web App.
  - Output can be downloaded to CSV. To download to Google Storage Buckets or Google BigQuery tables, the user must sign in.
  - Program filter has been added.
- The ISB-CGC home page includes a Programmatic API section. The Launch functionality includes links to:
  - ISB-CGC API;
  - Tutorials for Workflow on Google Cloud;
  - Comparison of Workflow Languages.
- All NA filters have been renamed to None in both the File Browser page and the Cohort Builder page.
- User details page has been modified to include more icons and buttons. The essential functionality and contents are not modified.

Known Issues

- Work is underway to rework our cohort creation page to better display images associated with samples.
- The user data upload feature will return an error message stating, "Error submitting response : Could not connect to data upload server."
- Analysis Type: Seq peek Formatting is Elongated on occasion.
- If the user shares a Cohort, neither the owner nor the person who was granted access to the Cohort will receive a confirmation email when sharing a Cohort.
- CCLE data cannot be plotted when working with workbooks. ISB-CGC will resolve this functionality after the GDC formally releases CCLE data.
- When a user duplicates a Worksheet, then tries to implement the log scale, it will not function properly.
- The set operation for existing Cohorts complement is behaving exceptionally slow.
- The mouse-over feature is currently disabled for program TARGET with disease code ALL.
- When uploading TARGET files using the cohort barcode creation feature from the GDC, you may get an invalid barcode error message and unable to upload all the barcodes.
- On the File Browser page for Diagnostic images, there is no GDC file UUID associated with them.
- Sharing a workbook with someone else will cause the analysis to reset.

July 23, 2020

New Features

- To increase system speed when filtering cohorts, switched metadata counting to use Apache Solr (instead of MySQL).
- The WebApp is now performing its data retrieval and counts on ISB-CGC Google BigQuery tables which are based on the latest GDC data release. This means that you will see current data, but that the same queries in the WebApp could produce different results if they were run during different time periods, when the WebApp was based on different GDC data releases.
- On the Create Cohorts – Filters page, on the left-hand filter panel, display the number of cases available for each filter, instead of the number of samples.

- Within the Cohort Details page, on the Current Filters panel, when there are more filters than what fits on the initial screen, display the selected cohort filters in a gradient (fade-away) overlay instead of a clipped design.
- The video tutorials have been moved to the ISB-CGC YouTube channel.

Bug Fixes

- Clicking on the X on an existing cohort filter token in the Selected Filter panel did not delete the existing cohort filter token. (This issue was caused by a jQuery update.) It has now been fixed.
- When a user tried to register for controlled access for 12 or more programs, this caused an error from Data Commons Framework (DCF) to occur. This was fixed by limiting the number of controlled access programs that a user could register for at one time to six.

Known Issues

- The Program filter is listing 'NA' as an option.
- Work is underway to rework our cohort creation page to better display images associated with samples.
- The user data upload feature will return an error message stating, "Error submitting response : Could not connect to data upload server."
- Analysis Type: Seq peek Formatting is Elongated on occasion.
- If the user shares a Cohort, neither the owner nor the person who was granted access to the Cohort will receive a confirmation email when sharing a Cohort.
- CCLE data cannot be plotted when working with workbooks. ISB-CGC will resolve this functionality after the GDC formally releases CCLE data.
- When a user duplicates a Worksheet, then tries to implement the log scale, it will not function properly.
- The set operation for existing Cohorts complement is behaving exceptionally slow.
- The mouse-over feature is currently disabled for program TARGET with disease code ALL.
- When uploading TARGET files using the cohort barcode creation feature from the GDC, you may get an invalid barcode error message and unable to upload all the barcodes.
- On the File Browser page for Diagnostic images, there is no GDC file UUID associated with them.
- Sharing a workbook with someone else will cause the analysis to reset.

May 27, 2020

New Features

- Modify the opt-in (subscription) form to have an "Ask me later" option.
- Provide a link (https://isb-cgc.appspot.com/opt_in/form_reg_user/) to the opt-in page. The link will first prompt the user to login with their google ID (if they are not already logged in). After the login, the feedback page will open.

Bug Fixes

- When writing and saving a comment in the cohort details or worksheet sections, the system displayed underlying code (such as escape characters) along with the text entered in the Comments panel. This has been corrected.
- Some data results were not displaying when working with OncoGrid due to it being unable to handle the amount of data being processed. This has been fixed.
- All plotting components under the Plot settings should be disabled when user views a shared workbook; however, 'Plot by' and 'Plot as Log' were not. This has been fixed.
- On analysis plots for workbooks, sometimes the y-axis tick marks would overlap the y-axis label when using the zoom out feature. This has been fixed.

- On the Create Cohorts - Filters page, when using the program TARGET with the filter Days to Birth, the Total Number of Cases and Total Number of Samples were not displaying. Also, the Save As New Cohort button was disabled. This has been corrected.

Known Issues

- Work is underway to rework our cohort creation page to better display images associated with samples.
- The user data upload feature will return an error message stating, "Error submitting response : Could not connect to data upload server."
- Analysis Type: Seq peek Formatting is Elongated on occasion.
- If the user shares a Cohort, neither the owner nor the person who was granted access to the Cohort will receive a confirmation email when sharing a Cohort.
- CCLE data cannot be plotted when working with workbooks. ISB-CGC will resolve this functionality after the GDC formally releases CCLE data.
- When a user duplicates a Worksheet, then tries to implement the log scale, it will not function properly.
- The set operation for existing Cohorts complement is behaving exceptionally slow.
- The mouse-over feature is currently disabled for program TARGET with disease code ALL.
- When uploading TARGET files using the cohort barcode creation feature from the GDC, you may get an invalid barcode error message and unable to upload all the barcodes.
- On the File Browser page for Diagnostic images, there is no GDC file UUID associated with them.
- Sharing a workbook with someone else will cause the analysis to reset.

April 16, 2020

New Features

- The Cohort Creation by Filter builder is now accessible without having to log in to ISB-CGC.
- The Cohort, Workbooks, and Gene and Variable Favorites lists are now paginated to display 10 to 15 records at a time.
- The 'To complete this analysis' section on the workbook creation page has changed from a checklist to an interactive tool. After each step is completed, its icon changes from an orange arrow to a green checkmark.
- A link 'Learn more about our available Analyses' was added next to the Analysis Type selection field. Clicking on this link opens up a screen with a detailed explanation of all the analysis options.

Bug Fixes

- On the File Browser, the search by CASE filter on the Radiology Images tab has been fixed.
- 'How to Cite Us' text on the Home page has been updated to reflect the entire ISB-CGC platform.
- When using a workbook, if you completely zoomed out of a plot, the chart was being reduced to half of the screen. This has been corrected.

Known Issues

- Work is underway to rework our cohort creation page to better display images associated with samples.
- The workbook zoom-out feature will cause text overlap in the y-axis panel of analysis.
- The user data upload feature will return an error message stating, "Error submitting response : Could not connect to data upload server."
- Analysis Type: Seq peek Formatting is Elongated on occasion.

- If the user shares a Cohort, neither the owner nor the person who was granted access to the Cohort will receive a confirmation email when sharing a Cohort.
- CCLE data cannot be plotted when working with workbooks. ISB-CGC will resolve this functionality after the GDC formally releases CCLE data.
- When a user duplicates a Worksheet, then tries to implement the log scale, it will not function properly.
- The set operation for existing Cohorts complement is behaving exceptionally slow.
- The mouse-over feature is currently disabled for program TARGET with disease code ALL.
- When uploading TARGET files using the cohort barcode creation feature from the GDC, you may get an invalid barcode error message and unable to upload all the barcodes.
- On the File Browser page for Diagnostic images, there is no GDC file UUID associated with them.
- Sharing a workbook with someone else will cause the analysis to reset.

March 11, 2020

New Features

- An Opt-in page was created for the user to sign up for ISB-CGC announcements.

Bug Fixes

- When working with the ISB-CGC API DELETE/cohorts/{cohort_id}, only able to delete cohorts owned by authenticated user.

Known Issues

- Analysis Type: Seq peek Formatting is Elongated on occasion.
- If the user shares a Cohort, neither the owner nor the person who was granted access to the Cohort will receive a confirmation email when sharing a Cohort.
- CCLE data cannot be plotted when working with workbooks. ISB-CGC will resolve this functionality after the GDC formally releases CCLE data.
- When a user duplicates a Worksheet, then tries to implement the log scale, it will not function properly.
- The set operation for existing Cohorts complement is behaving exceptionally slow.
- The mouse-over feature is currently disabled for program TARGET with disease code ALL.
- When uploading TARGET files using the cohort barcode creation feature from the GDC, you may get an invalid barcode error message and unable to upload all the barcodes.
- On the File Browser page for Diagnostic images, there is no GDC file UUID associated with them.
- Sharing a workbook with someone else will cause the analysis to reset.
- Work is underway to rework our cohort creation page to better display images associated with samples.

January 30, 2020

The following datasets (open and controlled access) have been added to the ISB-CGC for service account registration:

1. Genomics Evidence Neoplasia Information Exchange (GENIE)
2. The Pancreas Cancer Organoid Profiling (ORGANOID)
3. The Multiple Myeloma CoMMpass Study (MMRF)
4. Burkitt Lymphoma Genome Sequencing Project (CGCI)
5. Acute Lymphoblastic Leukemia - Phase I (TARGET-ALL-P1)

6. Acute Lymphoblastic Leukemia - Phase II (TARGET-ALL-P2)
7. Functional Genomic Landscape of Acute Myeloid Leukemia (BEATAML1.0-COHORT)

New Features

- The File Browser is enabled to define cancer names under the Disease Code filter in the left panel.

Bug Fixes

- The Cohorts share button is now enabled from the cohorts list page.
- The Cohort builder - filters, when using Pathologic Stage filter, the filters display in the correct format.
- Add a gene & miRNA variable favorite list from menu bar selection is now enabled.

November 26, 2019 v1.21

New Features

APIs

- Endpoint GET/data/available/registration lists all possible open and controlled programs available for registration with a service account.
- Endpoint GET/data/available/cohorts list all possible programs and projects available to use to make a cohort of the data available.

Known Issues

- Analysis Type: Seq peek Formatting is Elongated on occasion.
- If the user shares a Cohort, neither the owner nor the person who was granted access to the Cohort will receive a confirmation email when sharing a Cohort.
- CCLE data cannot be plotted when working with workbooks. ISB-CGC will resolve this functionality after the GDC formally releases CCLE data.
- When a user duplicates a Worksheet, then tries to implement the log scale, it will not function properly.
- The set operation for existing Cohorts complement is behaving exceptionally slow.
- The mouse-over feature is currently disabled for program TARGET with disease code ALL.
- When uploading TARGET files using the cohort barcode creation feature from the GDC, you may get an invalid barcode error message and unable to upload all the barcodes.
- On the File Browser page for Diagnostic images, there is no GDC file UUID associated with them.
- Sharing a workbook with someone else will cause the analysis to reset.
- Work is underway to rework our cohort creation page to better display images associated with samples.

August 27, 2019 v1.20

The following datasets (open and controlled access) have been added to the ISB-CGC for service account registration:

1. The Human Cancer Models Initiative (HCMI)
2. The Functional Genomic Landscape of Acute Myeloid Leukemia (BEATAML1.0)

New Features

- ISB-CGC APIs have been updated to a Swagger user interface as well as Google Endpoints OpenAPI, now known as APIsv4.

Known Issues

- Analysis Type: Seq peek Formatting is Elongated on occasion

- If the user shares a Cohort, neither the owner nor the person who was granted access to the Cohort will receive a confirmation email when sharing a Cohort.
- CCLE data cannot be plotted when working with workbooks. ISB-CGC will resolve this functionality after the GDC formally releases CCLE data.
- When a user duplicates a Worksheet, then tries to implement the log scale, it will not function properly.
- The set operation for existing Cohorts complement is behaving exceptionally slow.
- The mouse-over feature is currently disabled for program TARGET with disease code ALL.
- When uploading TARGET files using the cohort barcode creation feature from the GDC, you may get an invalid barcode error message and unable to upload all the barcodes.
- On the File Browser page for Diagnostic images, there is no GDC file UUID associated with them.
- Sharing a workbook with someone else will cause the analysis to reset.
- Work is underway to rework our cohort creation page to better display images associated with samples.

July 18, 2019 v3.19

The following datasets (open and controlled access) have been added to the ISB-CGC for service account registration:

1. The Clinical Proteomic Tumor Analysis Consortium (CPTAC)

New Features

Workbooks

- Edit plot settings feature provides the ability to plot by either cases or samples barcode count for a bar chart, histogram, scatter plot, violin plot, and cubby hole plot analyses.
- Detailed information provided by dbGaP for every program available when registering a Google service account.

Known Issues

- Analysis Type: Seq peek Formatting is Elongated on occasion
- If the user shares a Cohort, neither the owner nor the person who was granted access to the Cohort will receive a confirmation email when sharing a Cohort.
- CCLE data cannot be plotted when working with workbooks. ISB-CGC will resolve this functionality after the GDC formally releases CCLE data.
- When a user duplicates a Worksheet, then tries to implement the log scale, it will not function properly.
- The set operation for existing Cohorts complement is behaving exceptionally slow.
- The mouse-over feature is currently disabled for program TARGET with disease code ALL.
- When uploading TARGET files using the cohort barcode creation feature from the GDC, you may get an invalid barcode error message and unable to upload all the barcodes.
- On the File Browser page for Diagnostic images, there is no GDC file UUID associated with them.
- Sharing a workbook with someone else will cause the analysis to reset.
- Work is underway to rework our cohort creation page to better display images associated with samples.

April 25, 2019 v3.18

The following datasets (open and controlled access) have been added to the ISB-CGC for service account registration:

1. The National Cancer Institute Center for Cancer Research (NCICCR)
2. Foundation Medicine (FM)

3. Clinical Trial Sequencing Project (CTSP)
4. Veterans Research for Precision Oncology Program (VAREPOP)
5. Acute Lymphoblastic Leukemia - Phase III (TARGET-ALL-P3)

Enhancements

- When working with Oncogrid, OncoPrint, or a SeqPeek plot on a workbook, you will receive an automated list of genes ready for analysis.
- When on an additional workbook, text has been added to guide the user to select edit plot settings to choose a gene/miRNA/variable filter and cohort to used in the selected analysis.
- The Workbook comments section has been reformatted to better align with analysis displayed.
- On the cohort creation - filter page, the filters have been updated in the left filter panel to specify the count type displayed (samples).

Bug Fixes

- Clicking on a legend entry to toggle the display of the data points on a scatter or violin plot will now work correctly, even if the legend text has a space.
- Plotting with sample type filter on a workbook will now display counts correctly.
- When working with the color by feature on either a Scatter plot or a Violin plot, the numerical values are now displayed as a color-gradient legend.
- When using a workbook with OncoGrid analysis you are now able to plot using genomic build hg19.
- When using a workbook with a Cubby Hole plot analysis text is no longer cut off when using sample type or residual tumor as a filter.

Known Issues

- Analysis Type: Seq peek Formatting is Elongated on occasion
- If the user shares a Cohort, neither the owner nor the person who was granted access to the Cohort will receive a confirmation email when sharing a Cohort.
- CCLE data cannot be plotted when working with workbooks. ISB-CGC will resolve this functionality after the GDC formally releases CCLE data.
- When a user duplicates a Worksheet, then tries to implement the log scale, it will not function properly.
- The set operation for existing Cohorts complement is behaving exceptionally slow.
- The mouse-over feature is currently disabled for program TARGET with disease code ALL.
- When uploading TARGET files using the cohort barcode creation feature from the GDC, you may get an invalid barcode error message and unable to upload all the barcodes.
- On the File Browser page for Diagnostic images, there is no GDC file UUID associated with them.
- Sharing a workbook with someone else will cause the analysis to reset.
- Work is underway to rework our cohort creation page to better display images associated with samples.

March 8, 2019 v3.17

Enhancements

- When working with a workbook many overall enhancements of user functionality have been improved.
- Cubby hole plot analysis has been reformatted to better suit the end user by now allowing resizing and scrolling through the cubby hole plot analysis.

- You are now able to work on a workbook via fullscreen for added comfort.
- You are also now able to download plot data for Bar charts, Histogram charts, Scatter plots, Violin plot charts, and Cubby hole plots as a CSV file.
- OncoGrid has been added as an analysis option when working with a workbook.
- On the File Browser section you are now able to use full screen on all image viewers.
- On the register/adjust a service account page, we've clarified the notification message if a key or role is found associated to a service account.

Bug Fixes

- When using a workbook you will no longer see text overlap when working on a violin/scatter plot with the color by feature sample type as filter option.
- When working on the Pathology images viewer you will no longer see text overlap on the top right hand side of viewer.

Known Issues

- Analysis Type: Seq peek Formatting is Elongated on occasion
- If the user shares a Cohort, neither the owner nor the person who was granted access to the Cohort will receive a confirmation email when sharing a Cohort.
- CCLE data cannot be plotted when working with workbooks. ISB-CGC will resolve this functionality after the GDC formally releases CCLE data.
- When a user duplicates a Worksheet, then tries to implement the log scale, it will not function properly.
- The set operation for existing Cohorts complement is behaving exceptionally slow.
- The mouse-over feature is currently disabled for program TARGET with disease code ALL.
- When uploading TARGET files using the cohort barcode creation feature from the GDC, you may get an invalid barcode error message and unable to upload all the barcodes.
- On the File Browser page for Diagnostic images, there is no GDC file UUID associated with them.
- Sharing a workbook with someone else will cause the analysis to reset.
- Work is underway to rework our cohort creation page to better display images associated with samples.

January 22, 2019 v3.16

Enhancements

- On the Gene list creation page, you can now upload line separated and tab separated gene lists to be used for analysis.
- We have made some updates to the workbooks plotting section.
- You are now able to redraw to the original plot after any changes.
- Plots are now able to be saved as a .SVG, .PNG, or .JSON file.

Bug Fixes

- On the cohort creation using the barcode upload feature, the table page list feature now is now displayed properly.
- If you have not linked to the Data Commons Framework at all you are able to unregister a Google Cloud Project. If you are not linked to the Data Commons Framework, but others in the Google Cloud project are, only they will be able to unregister the GCP.

Known Issues

- Analysis Type: Seq peek Formatting is Elongated on occasion
- If the user shares a Cohort, neither the owner nor the person who was granted access to the Cohort will receive a confirmation email when sharing a Cohort.
- CCLE data cannot be plotted when working with workbooks. ISB-CGC will resolve this functionality after the GDC formally releases CCLE data.
- When a user duplicates a Worksheet, then tries to implement the log scale, it will not function properly.
- The set operation for existing Cohorts complement is behaving exceptionally slow.
- The mouse-over feature is currently disabled for program TARGET with disease code ALL.
- When uploading TARGET files using the cohort barcode creation feature from the GDC, you may get an invalid barcode error message and unable to upload all the barcodes.
- On the File Browser page for Diagnostic images, there is no GDC file UUID associated with them.
- Sharing a workbook with someone else will cause the analysis to reset.
- Work is underway to rework our cohort creation page to better display images associated with samples.

December 5, 2018 v3.15

Enhancements

- The ISB-CGC homepage has been updated to provide Funding and Partnership information, and the About Us section is now hidden by default.
- An introduction video has been added to the videos tutorials section. This video covers the user interface, BigQuery and using the API endpoints.
- Funding information has been updated on the ISB-CGC homepage.
- On the Register/Adjust a service account page all spacing issues have been addressed.
- On the Register/Adjust a service account pages you are now returned more detailed information. You will be returned verification results for all users on the Google Cloud Project, datasets permissions verification, registered service account verification results, and all service accounts verification results.
- On the File Browser page, when working with on a cohort with CCLE data included for genomic build hg38 you are displayed a notification message for CSV export button.
- On the File Browser a new column has been added for File Size for all tabs.
- When exporting a large cohort on the File Browser page you are returned a notification message stating cohort export is underway to check BigQuery in a few minutes.
- On the File Browser you are now able to view/download/print Pathology Reports in pdf format.
- On the Pathology Images viewer, the GDC has released multiple versions of slide barcodes. To handle this we now sort the pathology image files by UUID.
- On the the File Browser for Radiology Images, ISB-CGC has upgraded the viewer to run OHIF for better performance times and views.

Bug Fixes

- When working on the File Browser export to BigQuery/Google Cloud Storage entering an invalid name will disable the export feature, even after toggling between datasets.
- When on a Workbook, using an OncoPrint analysis using certain genes with no gene positions will return correct error message stating no internal feature ID was found.
- Certain gene names which symbol '_' included will now return data points when working with a Workbook.

Known Issues

- Analysis Type: Seq peek Formatting is Elongated on occasion
- If the user shares a Cohort, neither the owner nor the person who was granted access to the Cohort will receive a confirmation email when sharing a Cohort.
- CCLE data cannot be plotted when working with workbooks. ISB-CGC will resolve this functionality after the GDC formally releases CCLE data.
- When a user duplicates a Worksheet, then tries to implement the log scale, it will not function properly.
- The set operation for existing Cohorts complement is behaving exceptionally slow.
- The mouse-over feature is currently disabled for program TARGET with disease code ALL.
- When uploading TARGET files using the cohort barcode creation feature from the GDC, you may get an invalid barcode error message and unable to upload all the barcodes.
- On the File Browser page for Diagnostic images, there is no GDC file UUID associated with them.
- Sharing a workbook with someone else will cause the analysis to reset.
- Work is underway to rework our cohort creation page to better display images associated with samples.

September 20, 2018 v3.14

Enhancements

- When on the File browser page, the case barcode column is included when downloading the file manifest CSV format option.
- You will now need to log into the Data Commons Framework to be able to access controlled data.

Bug Fixes

- API endpoint cohort.creation will no longer include NULL values in sample counts when cohort is created.
- On the File Browser tab using filter option NA will now return all entries associated to it.
- Program TCGA and TARGET have new miRNA based on the GDC release 11 is now available in Google BigQuery and for plotting.

Known Issues

- Analysis Type: Seq peek Formatting is Elongated on occasion
- If the user shares a Cohort, neither the owner nor the person who was granted access to Cohort will receive a confirmation email when sharing a Cohort.
- CCLE data cannot be plotted when working with workbooks. ISB-CGC will resolve this functionality after the GDC formally releases CCLE data.
- When a user duplicates a Worksheet, then tries to implement the log scale, it will not function properly.
- The set operation for existing Cohorts complement is behaving exceptionally slow.
- The mouse-over feature is currently disabled for program TARGET with disease code ALL.
- When uploading TARGET files using the cohort barcode creation feature from the GDC, you may get an invalid barcodes error message and unable to upload all the barcodes.
- On the File Browser page for Diagnostic images there is no GDC file UUID associated to them.
- Sharing a workbook with someone else will cause the analysis to reset.
- When using a workbook, a gene with symbol "_" will produce a error message saying, "There was an error retrieving plot data. Please try again."

- Work is underway to rework our cohort creation page to better differentiate between samples which are from image data vs. those which are not.

July 31, 2018 v3.13

Enhancements

- When working on the File Browser you now have the ability to search by case barcode all on tabs(Pathology Images, Radiology Images, IGV Browser, All Files).
- On the File Browser page for the Pathology Images tab, you can now also filter by Disease Code, Data Format, and Data Type. For the Radiology Images, a disease code was added.
- On the File Browser page, you now have the ability to hide the filters and expand the file list to full width.
- On the File Browser page, if you download the file manifest using the export CSV feature, you will see newly updated file paths. The older paths are still in existence but will be deleted within the next month.
- On the File Browser page if you use a cohort with CCLE data present, switch to build hg38 and attempt to export you will return a notification no CCLE data will be present for build hg38.
- On the homepage, we have added a carousel scrolling feature for all how-to videos for easy access.
- A description has been added to all video tutorials.
- The menu bar text variable favorites have been updated to be undifferentiated.

Bug Fixes

- When creating a cohort using the filter selection option, if the filter options selected add up to zero the save cohort button will be disabled.
- A workbook with user upload data and public data e.g TCGA data will plot any analyses.
- For the export to GCS and BigQuery feature the export button will now disable when an invalid name is given.
- On a registered Google Cloud Project detail page, datasets can no longer be duplicated within a project, and bucket names are globally unique (across all projects).

Known Issues

- Analysis Type: Seq peek Formatting is Elongated on occasion
- If the user shares a Cohort, neither the owner nor the person who was granted access to Cohort will receive a confirmation email when sharing a Cohort.
- CCLE data cannot be plotted when working with workbooks. ISB-CGC will resolve this functionality after the GDC formally releases CCLE data.
- When a user duplicates a Worksheet, then tries to implement the log scale, it will not function properly.
- The set operation for existing Cohorts complement is behaving exceptionally slow.
- The mouse-over feature is currently disabled for program TARGET with disease code ALL.
- When uploading TARGET files using the cohort barcode creation feature from the GDC, you may get an invalid barcodes error message and unable to upload all the barcodes.
- API endpoint cohort.creation will include NULL values in sample counts when the cohort is created.
- On the File Browser page for Diagnostic images, there is no GDC file UUID associated to them.
- Sharing a workbook with someone else will cause the analysis to reset.
- When downloading the CSV file for Radiology Images tab on the File Browser page you will noticed there are no samples barcodes associated to Radiology Images. ISB-CGC will add a case barocde to the CSV file export table in the next release.

- Work is underway to rework our cohort creation page to better differentiate between samples which are from image data vs. those which are not.

June 18, 2018 v3.12

Enhancements

- The ISB-CGC has enabled OncoPrint visualization tool for germline mutations (codebase obtained with permission from cBioPortal) as another Workbook analysis tool. For more information please go here.
- You are now able to view Radiology Images from TCIA data through the File Browser using the Osimis viewer. For more information please go here here.
- Two new videos have been added to our video tutorials section. You can now learn how to sign up with a Google account and how to make a gene list easily. For more information please go here. here
- The Dashboard has been upgraded to include a collapse feature for all panels (workbooks and cohorts are opened by default) and a direct link to the File Browser has been added to the Cohorts panel.
- Under cohort creation by filters, the Molecular tab for TCGA data has been upgraded to combine multiple gene mutation filters. Filters can be combined using AND (requires all filters to be met for the data to be filtered) or OR (at least one criteria needs to be met for the data to be displayed).
- The CSV download, Export to BigQuery, and Export to GCS feature has been added to the IGV Browser, Pathology Images, and the Radiology Images tab on the File Browser.
- On the File Browser All files tab the clinical filter now displays the accurate count available for analysis.
- The File Browser has been upgraded to now include the option of which columns to display and the ability to jump to any page.
- The site menu has been improved to allow faster load times and better overall performance. Please Note that Workbooks must now be created from a data source (Cohorts, Variable lists, Gene & miRNA lists) or from the Workbook list page.

Bug Fixes

- When working on Firefox browser a violin plot will display the data plotted correctly when working on a Worksheet.
- A cohort with user uploaded data present and public data present in our system e.g TCGA data, the cohort details page for the selected filters panel will sort the filters by their appropriate program.
- On the cohort creation - barcode upload page the 'Samples' and 'Cases' column headers were sometimes swapped. This has been corrected.
- When trying to reload a stored Seq-Peek plot from a Workbook the previous gene selection is stored and the plot will automatically be loaded.
- On the File Browser IGV Browser tab when switching genomic builds the view column selection option will be disabled.

Known Issues

- Analysis Type: Seq peek Formatting is Elongated on occasion
- If the user shares a Cohort, neither the owner nor the person who was granted access to Cohort will receive a confirmation email when sharing a Cohort.
- CCLE data cannot be plotted when working with workbooks. ISB-CGC will resolve this functionality after the GDC formally releases CCLE data.
- When a user duplicates a Worksheet, then tries to implement the log scale, it will not function properly.
- The set operation for existing Cohorts complement is behaving exceptionally slow.

- The mouse-over feature is currently disabled for program TARGET with disease code ALL.
- When uploading TARGET files using the cohort barcode creation feature from the GDC, you may get an invalid barcodes error message and unable to upload all the barcodes.
- API endpoint cohort.creation will include NULL values in sample counts when cohort is created.
- Duplicate entries can be entered for the register a dataset and the register a bucket on the Google cloud project details page.
- On the File Browser page for Diagnostic images there is no GDC file UUID associated to them.
- Sharing a workbook with someone else will cause the analysis to reset.
- A Workbook using a cohort that has user uploaded data and public TCGA data present will not return data for any analysis.
- Work is underway to rework our cohort creation page to better differentiate between samples which are from image data vs. those which are not.

May 3, 2018 v3.11

Enhancements

- The export to BigQuery feature has been enhanced to include faster processing time for larger cohorts with e.g 30,000 > samples and 65,000 > file records.
- You are now able to export cohort and cohort file manifests to a Google Cloud Storage using either .JSON or .CSV format from the cohort details page and from the File Browser page.
- We have enhanced our instructions associated with buttons to further provide directions to the end-users.
- On the File Browser page it is now possible to change how many entries are displayed at a time, as well as sort columns by clicking on the column header.
- Google Cloud Project membership is now automatically updated every six hours. If you are adding someone new to the project they will be able to use the project after six hours maximum without someone having to log in and manually refresh the project.

Bug Fixes

- You can no longer share a cohort with yourself (email currently logged into) and cause the file browser page to disable.
- DNA methylation has been re-enabled to be used with hg38 and hg19 data when working with workbooks and plotting.
- Sharing inputs have had their security restrictions tightened. This also includes the registering a service account page.
- On the File Browser page when downloading the file manifest via the CSV button you are no longer able to re-select the CSV button while the file is building.
- On the File Browser tab if you toggle between entries pages on the All Files tab it will not affect the IGV tab or Pathology Images tab entries counts display.
- On the File Browser page you can now freely toggle between entries pages with no errors displayed.
- On the File Browser page selecting filters from the left hand side while exploring pages will no longer crash and require you to back or refresh the page to fix.

Known Issues

- Analysis Type: Seq peek Formatting is Elongated on occasion

- If the user shares a Cohort, neither the owner nor the person who was granted access to Cohort will receive a confirmation email when sharing a Cohort.
- CCLE data cannot be plotted when working with workbooks. ISB-CGC will resolve this functionality after the GDC formally releases CCLE data.
- When a user duplicates a Worksheet, then tries to implement the log scale, it will not function properly.
- The set operation for existing Cohorts complement is behaving exceptionally slow.
- The mouse-over feature is currently disabled for program TARGET with disease code ALL.
- When working on Firefox browser a violin plot does not display the data plotted correctly when working on a Worksheet.
- When uploading TARGET files using the cohort barcode creation feature from the GDC, you may get an invalid barcodes error message and unable to upload all the barcodes.
- API endpoint cohort.creation will include NULL values in sample counts when cohort is created.
- Duplicate entries can be entered for the register a dataset and the register a bucket on the Google cloud project details page.
- A cohort with user uploaded data present and public data present in our system e.g TCGA data, the cohort details page for the selected filters panel does not properly display the filters selected.
- On the File Browser page for Diagnostic images there is no GDC file UUID associated to them.
- Work is underway to rework our cohort creation page to better differentiate between samples which are from image data vs. those which are not.

April 2, 2018 v3.10

Enhancements

- When working with the File List table you can now Export the cohort file list to BigQuery for later analysis.
- When registering or adjusting a service account to use controlled data, the page will no longer briefly appear as if no datasets had been selected. This should reduce confusion.
- Selecting the refresh project button from a registered Google Cloud Project details page will leave you on the details page rather than redirecting you to the registered Google cloud project list table page.
- On the cohort creation page, using the barcode upload page, the valid/invalid entries table can now be sorted by on any column with either ascending/descending order.
- Removing someone from the IAM and Admin list does not remove them from the web-app automatically. If the removed user still has the GCP present in their webapp interface attempting to register or refresh a service account will remove the GCP from the web app, and a display message informing them they are no longer a member of the project will be seen.
- When working with any tables that can be sorted on smaller screens, there is no longer any text overlap in the table columns.
- Character restrictions has been relaxed, you can now use characters such as []{}(); for entity names and descriptions.

Bug Fixes

- SeqPeek and CNVR can only be plotted with TCGA data, but if a cohort contains no TCGA samples the SeqPeek analysis will now return an error message saying, "The chosen cohorts do not contain samples from programs with Gene Mutation data."
- API endpoint samples.get can now be used to return data for all three programs.

- On the adjust service account page, when attempting to remove the service account from being able to access controlled data, and then immediately trying to add the service account back to controlled data, the system will require you to verify the service account's users again.

Known Issues

- Analysis Type: Seq peek Formatting is Elongated on occasion
- If the user shares a Cohort, neither the owner nor the person who was granted access to Cohort will receive a confirmation email when sharing a Cohort.
- CCLE data cannot be plotted when working with workbooks. ISB-CGC will resolve this functionality after the GDC formally releases CCLE data.
- When a user duplicates a Worksheet, then tries to implement the log scale, it will not function properly.
- The set operation for existing Cohorts complement is behaving exceptionally slow.
- The mouse-over feature is currently disabled for program TARGET with disease code ALL.
- When working on Firefox browser a violin plot does not display the data plotted correctly when working on a Worksheet.
- When uploading TARGET files using the cohort barcode creation feature from the GDC, you may get an invalid barcodes error message and unable to upload all the barcodes.
- On the cohort File List Browser page, while you are downloading CSV files, other filters can be selected.
- Work is underway to rework our cohort creation page to better differentiate between samples which are from image data vs. those which are not.

February 28, 2018 v3.9

Enhancements

- On the register a Google Cloud Project you now can only register the project ID. Registering the project name or project number will now result in an error message. Additionally, the GCP Project Name and ID will now both display on the GCP detail and list pages, and refreshing a GCP Project in the Web Application will update the Name if it was changed in the GCP console.
- For cohort creation via sets of barcodes, the barcode set (pasted in the text box or uploaded as a file) can now be a simple list of sample or case barcodes separated by newlines, commas, or tabs; the program listing is no longer needed, and you don't need to supply the barcodes in a distinct columnar format.. The previous 3-column format will continue to work as well.
- On a worksheet, if no table is being searched the BQ table(s) used panel becomes inactive.

Bug Fixes

- When editing the name of a cohort the cancel feature is now working properly.
- When working on a worksheet the SeqPeek feature will now work with all genes.
- All genes can be plotted on a worksheet when working with a histogram.
- When registered Service Accounts for controlled data, the Adjust/Register can only be clicked once.
- When working with SeqPeek, the BQ table(s) used panel will now refresh every time even if no new data is plotted.
- When a user is removed from their Google project the user interface doesn't remove the project from their list. Instead, the individual removed will receive error messages saying they are no longer on the project if they try to refresh the project or register the service account.
- On a registered Google Cloud Project page, the refresh button will now properly add and remove users from the project if they are added or removed from the IAM and Admin list on the Google console.

- When working on the Internet Explorer you can again create a cohort using the filter creation page.
- When using the dbGaP eRA authentication you will now be logged out at 24 hours instead of 16 hours.
- For cohort creation when uploading a large set of barcodes you will no longer return a 400 bad request error.

Known Issues

- Analysis Type: Seq peek Formatting is Elongated on occasion
- If the user shares a Cohort, neither the owner nor the person who was granted access to Cohort will receive a confirmation email when sharing a Cohort.
- CCLE data cannot be plotted when working with workbooks. ISB-CGC will resolve this functionality after the GDC formally releases CCLE data.
- When a user duplicates a Worksheet, then tries to implement the log scale, it will not function properly.
- The set operation for existing Cohorts complement is behaving exceptionally slow.
- The mouse-over feature is currently disabled for program TARGET with disease code ALL.
- When working on Firefox browser a violin plot does not display the data plotted correctly when working on a Worksheet.
- When uploading TARGET files using the cohort barcode creation feature from the GDC, you may get an invalid barcodes error message and unable to upload all the barcodes.
- SeqPeek and CNVR can only be plotted with TCGA data, but if a cohort contains no TCGA samples the SeqPeek analysis will still search the TCGA BigQuery tables
- API endpoint samples.get currently down and will return a 503 error for all three programs.
- On the File Browser page, while you are downloading CSV files, other filters can be selected.
- Work is underway to rework our cohort creation page to better differentiate between samples which are from image data vs. those which are not.

February 1, 2018 v3.8

Enhancements

- We have enabled DNA methylation data to be used when plotting with genomic build hg38.
- The cohort view files page has been updated to File Browser. The File Browser page also now has new filters data level, data type, disease code, data format, and experimental strategy. A time stamp has also been added to the CSV file that can be downloaded.
- The IGV browser and caMicroscope are now more clearly defined and separated on the File Browser page.
- When uploading a set of barcodes to create a cohort the error message has been redefined to direct someone to the instructions.

Bug Fixes

- You can now plot DNA methylation data using genomic build hg19 when working on a worksheet.
- When registering a service account to controlled data you will no longer receive an error message when certain Google managed service accounts are also on the IAM and Admin page.
- On a worksheet, if you add new cohorts to a worksheet with pre-existing cohorts. Now the older and newly added cohorts are present on the worksheet for analysis.
- When working with a worksheet you are now able to plot gene names that contain periods.

Known Issues

- You cannot make a cohort using the cohort creation filter option on an Internet Explorer browser.

- Analysis Type: Seq peek Formatting Elongated on occasion.
- If the user shares a Cohort neither the owner nor the person who was granted access to Cohort will receive a confirmation email when sharing a Cohort.
- CCLE data cannot be plotted when working with workbooks. ISB-CGC will resolve this functionality after the GDC formally releases CCLE data.
- When a user duplicates a Worksheet, then tries to implement the log scale it will not function properly.
- The set operation for existing Cohorts complement is behaving exceptionally slow.
- The mouse-over feature is currently disabled for program TARGET with disease code ALL.
- When working on Firefox browser a violin plot does not display the data plotted correctly when working on a Worksheet.
- When uploading TARGET files using the cohort barcode creation feature from the GDC you may get an invalid barcodes error message and unable to upload all the barcodes.
- SeqPeek can only be plotted with TCGA data, but if a cohort contains no TCGA samples the SeqPeek analysis will still search the TCGA BigQuery tables.
- API endpoint samples.get currently down and will return a 503 error for all three programs.
- Currently unable to use TARGET data with the IGV browser to view .bam files.
- When editing the name of a cohort the cancel feature is not working properly.
- When working on a worksheet the SeqPeek feature is currently not working with certain genes.
- Certain genes will produce a blank chart with no data on a worksheet when working with a histogram.
- Work is underway to rework our cohort creation page to better differentiate between samples which are from image data vs. those which are not.

December 20, 2017 v3.7

Enhancements

- Using the 'View Files' page you can now view TCGA pathology images using caMicroscope!
- After logging into dbGaP you are now redirected to the user details page.
- Due to recent updates with Google, we have implemented new security requirements when working with the service accounts and attempting the access the controlled data. For more information about new requirements please go here.

Bug Fixes

- You will no longer experience a 502 error when trying to create a new variable favorite list if you have uploaded a lot of your own data using the user data upload feature.

Known Issues

- Analysis Type: Seq Peek formatting elongated on occasion
- If the user shares a Cohort neither the owner nor the person who was granted access to Cohort will receive a confirmation email when sharing a Cohort.
- CCLE data cannot be plotted when working with workbooks. ISB-CGC will resolve this functionality after the GDC formally releases CCLE data.
- When a user duplicates a Worksheet, then tries to implement the log scale it will not function properly.
- The set operation for existing Cohorts complement is behaving exceptionally slow.
- The mouse-over feature is currently disabled for program TARGET with disease code ALL.

- When working on Firefox browser a violin plot does not display the data plotted correctly when working on a Worksheet.
- When working on a workbook if you add new cohorts to the worksheet the pre-existing cohorts will be de-selected from the worksheet.
- If you have uploaded a lot of data using the User Data Upload feature, it is likely you will experience 502 error page when attempting to create a new variable favorite list.
- When uploading TARGET files using the cohort barcode creation feature from the GDC you may get an invalid barcodes error message and unable to upload all the barcodes.
- Work is underway to rework our cohort creation page to better differentiate between samples which are from image data vs. those which are not.

November 20, 2017 v3.6

Enhancements

- You can now send a cohort you have created in the web application to a new BigQuery dataset or append an existing table.
- The cohort creation by uploading barcodes feature has been extended to include .JSON and .TSV files from the Genomic Data Commons data portal.
- Created a new API endpoint to be used to return a GCS object URL given a GDC file identifier also known as a UUID.
- Updated the registered Google Cloud Project to clearly state if the project's service accounts are active or not.
- You can now enter special characters into the comments section for workbooks and cohorts e.g URL
- On the register a service account page the Compute Engine default service account is automatically added to the enter service ID text box.
- When creating a new cohort we have implemented a text saying, "Creating cohort…" for instances when creating a new cohort takes a little longer than usual.
- We have significantly sped up loading times for the cohorts detail and cohorts table list page for users who have 50 + cohorts which caused slow loading time.

Bug Fixes

- A duplication of the exact cohort will no longer happen when you select the confirmation multiple times while the page is loading working with Set Operations.
- On the cohort details, you can no longer select the clinical feature panel and edit filters without selecting the edit button first.
- On the cohort creation page, you can use the clinical feature panel to select filters when working with the User data upload tab.

Known Issues

- Analysis Type: Seq peek Formatting Elongated on occasion
- If the user shares a Cohort neither the owner nor the person who was granted access to Cohort will receive a confirmation email when sharing a Cohort.
- CCLE data cannot be plotted when working with workbooks. ISB-CGC will resolve this functionality after the GDC formally releases CCLE data.
- When a user duplicates a Worksheet, then tries to implement the log scale it will not function properly.
- The set operation for existing Cohorts complement is behaving exceptionally slow.

- The mouse-over feature is currently disabled for program TARGET with disease code ALL.
- When working on Firefox browser a violin plot does not display the data plotted correctly when working on a Worksheet.
- When working on a workbook if you add new cohorts to the worksheet the pre-existing cohorts will be de-selected from the worksheet.
- If you have uploaded a lot of data using the User Data Upload feature, it is likely you will experience 502 error page when attempting to create a new variable favorite list.
- When working with the API endpoints the sample.get for all three programs will return a 503 internal server error.

October 13, 2017 v3.5

Enhancements

- You can now upload sample and case identifiers from programs TCGA, CCLE and TARGET to create a cohort.
- We have begun to allow the addition/removal of a service account with a new button instead of the user having to re-register the service account every time.
- For the Set Operations feature when working with cohorts has been enhanced and has become easier to work with.
- For the Set Operation Complement feature you will now create a cohort faster than before.
- You will now be displayed mouse over text when working with the New Workbook, Delete, Set Operations, and Share button on the Cohorts list details page.
- The About Us link in the top left of the page has been re-named to Homepage.

Bug Fixes

- All bam files for the TARGET program are available to be used with the IGV browser.
- On the Cohort creation page, you can now select a filter for your Cohort by selecting an option from the Clinical Feature graphs using Histological Type for program CCLE.

Known Issues

- Analysis Type: Seq peek Formatting Elongated on occasion
- If the user shares a Cohort neither the owner nor the person who was granted access to Cohort will receive a confirmation email when sharing a Cohort.
- CCLE data cannot be plotted when working with workbooks. ISB-CGC will resolve this functionality after the GDC formally releases CCLE data.
- When a user duplicates a Worksheet, then tries to implement the log scale it will not function properly.
- The set operation for existing Cohorts complement is behaving exceptionally slow.
- A duplication of the exact cohort happens when you select the confirmation multiple times while the page is loading working with Set Operations.
- The mouse-over feature is currently disabled for program TARGET with disease code ALL.
- When working on Firefox browser a violin plot does not display the data plotted correctly when working on a Worksheet.
- We need to rework our cohort creation page to better differentiate between samples which are from image data vs. those which are not.

September 21,2017 v3.4

Enhancements

- When plotting, certain values will now be displayed as categorical when before it was displayed as a numerical value e.g Tobacco Smoking History.
- The Homepage has been updated to incorporate links for TARGET and CCLE programs.
- The extended list of programs and projects on the new User Uploaded Data creation page is now displayed in alphabetical order.
- On the user details page you are now shown a confirmation box when you attempt to unlink the NIH identity account associated to the Google Identity you originally logged in with.
- When working with Workbooks you are now shown a table on the top right hand side of Worksheet which shows what BigQuery tables the information being displayed is from.
- On the Cohort creation page you can now select a filter for your Cohort by selecting an option from the Clinical Features graphs.
- On the user details page, if you attempt to associate you Google Identity to an NIH Identity that is already registered in the system to another Google Account you are given a yellow error message stating which email the NIH Identity is already associated to.

Bug Fixes

- When working with Workbooks the log scale graphing option will be saved when a user comes back to the Worksheet at another time.
- On the existing Cohorts table list page, the confirmation delete 'blue x' button will now remove a selected Cohort if you select another option e.g Set Operation.
- The Google Cloud Project details page refresh wheel and delete icon are now working properly for service accounts.
- The Cloud Project details page now lists the authorized datasets active with an associated service account.
- When deleting a User Uploaded program you are now sent to the existing programs list page if you delete the program. If you delete the project you stay on the program details page.
- The ownership of a Variable list, Gene and miRNa list, and User Uploaded Programs are now verified. This means you can no longer view any existing in system if you are not the original creator.
- A confirmation on the Register a Service Account page has been implemented for service accounts when the user attempts to register.
- On the Cohort creation when toggling between the tabs for the different programs, you now cannot switch tabs until the tab on display is loaded.
- We need to rework our cohort creation page to better differentiate between samples which are from image data vs. those which are not.

Known Issues

- Analysis Type : Seq peek Formatting Elongated on occasion
- If the user shares a Cohort neither the owner nor the person who was granted access to Cohort will receive a confirmation email when sharing a Cohort.
- CCLE data cannot be plotted when working with workbooks. ISB-CGC will resolve this functionality after the GDC formally releases CCLE data.
- When a user duplicates a Worksheet, then tries to implement the log scale it will not function properly.
- The set operation for existing Cohorts complement is behaving exceptionally slow.

- A duplication of the exact cohort happens when you select the confirmation multiple times while the page is loading working with Set Operations.
- The mouse over feature is currently disabled for program TARGET with disease code ALL.
- A very small amount of bam files for program TARGET currently have the wrong file name and cannot be used with the IGV browser.
- When working on Firefox browser a violin plot does not display the data plotted correctly when working on a Worksheet.

August 23, 2017 v3.3

Enhancements

- Users with NIH-approved access can now view and analyze TARGET (Therapeutically Applicable Research To Generate Effective Treatments) controlled data using service accounts and also on the IGV browser.
- You will be returned a more detailed error message when invalid characters are used with user data uploading titles.
- On the File list page you will be allowed to select only one genomic build at a time for clarity on which build will be used by the IGV browser.
- When attempting to duplicate the registration of your Google Cloud Project you are given an error message saying, "A Google Cloud Project with the id xxx-xxx-xxxx already exists."
- If you attempt to register a service account with the same datasets it already has activated, you will be given an error message saying, "Service account xxxxxxxxxxxx-compute@developer.gserviceaccount.com already exists with these datasets, and so does not need to be registered."
- The Data Use Certification and Agreement covering your access to all controlled data has been added to the user details page in the interface.
- The CCLE user.get API endpoint has been removed from the system due to the fact we do not currently host any controlled CCLE data.
- The format of CSV file downloaded with Download IDs button from the cohort details page has been changed to display the case and sample barcodes as two separate columns.
- From the User uploaded program detail page, you can now edit the project name and description by selecting the gear option.

Bug Fixes

- When creating a large cohort you are no longer returned a red error message.
- The sharing feature for Workbooks, Cohorts, and User Uploaded Programs has been re-activated. You must enter a valid email address that is present in the system to share the workbook, cohort, or user uploaded program. If they are not present in our system please feel free to invite them to the ISB-CGC website.
- When working with a new worksheet or a duplicate worksheet with workbooks for categorical features e.g bar chart, you can no longer select the log option. The log option only applies to numerical options.
- When working with workbooks, selecting the Delete button multiple times will no longer result in an error, and instead return you to the Workbooks list page after successful deletion of the Workbook.
- Users can plot user uploaded data when working with workbooks when using variables and cohorts from the same files that were uploaded.
- The cohort.list API endpoint will display the correct cases count for cohorts listed.
- The Download File List as CSV on the File List page will download the correct information when genomic build hg38 is selected.

- You are no longer able to add XSS-vulnerable characters to the edit section for user uploaded data.
- An improved error message is displayed when attempting to register a Google Project you are not associated with.
- Making a new Gene and miRNA set from a Workbook will no longer result in lowercase gene and miRNA names.
- The TCGA Sample.get API endpoint will no longer return a response with sample ID duplicates.

Known Issues

- Analysis Type : Seq peek Formatting Elongated on occasion
- If the user shares a cohort neither the owner nor the person who was granted access to cohort will receive a confirmation email when sharing a cohort.
- CCLE data cannot be plotted when working with workbooks. ISB-CGC will resolve this functionality after the GDC formally releases CCLE data.
- When a user duplicates a worksheet, then tries to implement the log scale it will not function properly.
- On the existing cohorts table list page, the confirmation delete 'blue x' button does not remove selected cohort if you select another option e.g Set Operation. The same issue can be found in reverse if you select the 'blue x' on the confirmation page for set operation you can then select the delete button and see the cohort on the confirmation panel.
- When working with working with workbooks the log option is not working properly for the plot settings.
- The set operation for existing cohorts complement is behaving exceptionally slow.
- A duplication of the exact cohort happens when you select the confirmation multiple times while the page is loading working with Set Operations.
- When plotting, certain values will be displayed as numerical when it should be a categorical value e.g Tobacco Smoking History.
- The mouse over feature is currently disabled for program TARGET with disease code ALL.

July 31, 2017 v3.2

Enhancements

- You will be returned a more detailed error message when using invalid characters when working with user data uploading titles.
- On the File list page you will are allowed to select only one genomic build at a time for better clarification of which build you will view on the IGV browser.

Bug Fixes

- When working with Swap Values button on a worksheet, the log option selected for either axis is now carried over as well when the swap values button is selected.
- On the IGV browser when working with TCGA data build hg38 the interface will no longer return a No feature found with name "efgr" at the bottom of the IGV browser page.
- When working with the cohort.create API endpoint you have the ability to create a large cohort with the barcode filter without a timeout error.
- When creating a cohort with the cohort.create API endpoint you can view the list of barcodes from the cohort details page in the ISB-CGC user interface irrelevant of size.
- When working with the create a new variable favorites list page, you can now create a variable list using the USER DATA tab.

Known Issues

- The sharing feature for Workbooks, Cohorts, and User Uploaded Programs is currently disabled
- Analysis Type : Seq peek Formatting Elongated on occasion
- The CCLE data in GUI is not parallel to the CCLE data in BigQuery.
- Cannot plot any data if you use a CCLE data cohort on a worksheet.
- On the existing cohorts table list page, the confirmation delete 'blue x' button does not remove selected cohort if you select another option e.g Set Operation. The same issue can be found in reverse if you select the 'blue x' on the confirmation page for set operation you can then select the delete button and see the cohort on the confirmation panel.
- The set operation for existing cohorts complement is behaving exceptionally slow.
- A duplication of the exact cohort happens when you select the confirmation multiple times while the page is loading working with Set Operations.
- When working with a new worksheet or a duplicate worksheet with workbooks for categorical features e.g bar chart you can select the log option. The log option only applies to numerical options.
- When working with workbooks, if you select the delete confirmation button multiple times while the page is loading you will be sent to an error page.
- You currently cannot plot user uploaded data when working with workbooks.
- When plotting, certain values will be displayed as numerical when it should be a categorical value e.g Tobacco Smoking History.
- The mouse over feature is currently disabled for program TARGET with disease code ALL.
- The cohort.list API endpoint will display the incorrect cases count for cohort listed.
- The Download File List as CSV on the File List page downloads the wrong information when genomic build hg38 is selected.
- You are currently able to add non-whitelist characters to edit section for user uploaded data.
- You are returned a vague error message on the register a Google Cloud Project page when attempting to register a Google Project you are not associated to.
- The samples and cases filters have not been removed from the cohort.list API endpoint and are visible as a possible filter.
- The user.get CCLE program API endpoint will return a 503 internal server error.
- When creating large cohort you will be given a red error message saying, "There was an error saving your cohort; it may not have been saved correctly."

June 14, 2017 v3.1

Known Issues

- Analysis Type : Seq peek Formatting Elongated on occasion
- The CCLE data in the Webapp is not exactly the same as the CCLE data in BigQuery.
- Users cannot plot any data from a CCLE cohort on a worksheet.
- In the Webapp, the log scale on graphs does not function properly for duplicated worksheets.
- On the existing cohorts table list page, the confirmation delete 'blue x' button does not remove selected cohort if you select another option e.g Set Operation. The same issue can be found in reverse if you select the 'blue x' on the confirmation page for set operation you can then select the delete button and see the cohort on the confirmation panel.
- Swap values is not working properly for the plot settings.

- The set operation for existing cohorts complement is behaving exceptionally slow.
- A duplication of the exact cohort happens when you select the confirmation multiple times while the page is loading working with Set Operations.
- When working with a new worksheet or a duplicate worksheet with workbooks for categorical features e.g bar chart you can select the log option. The log option only applies to numerical options.
- When working with workbooks, if you select the delete confirmation button multiple times while the page is loading you will be sent to an error page.
- You currently cannot plot user uploaded data when working with workbooks.
- When plotting, certain values will be displayed as numerical when it should be a categorical value e.g Tobacco Smoking History.
- On the IGV browser when working with TCGA data build hg38 you get a No feature found with name "efgr" at the bottom of the iGV browser page.
- On the cohort creation page for TCGA data the filters disease code and project short name NA is an option which is not a valid disease.
- The mouse over feature is currently disabled for program TARGET with disease code ALL.
- The sharing feature for Workbooks, Cohorts, and User Uploaded Programs is currently disabled.
- A number of TCGA and CCLE case IDs shown below will have been removed from all cohorts since they are no longer available from NCI's Genomics Data Commons, and ISB-CGC is trying to mirror that data as closely as possible.
- TCGA cases:

TCGA-33-4579, TCGA-35-3621, TCGA-66-2746, TCGA-66-2747, TCGA-66-2750, TCGA-66-2751, TCGA-66-2752, TCGA-AN-A0FE, TCGA-AN-A0FG, TCGA-BH-A0B2, TCGA-BR-4186, TCGA-BR-4190, TCGA-BR-4194, TCGA-BR-4195, TCGA-BR-4196, TCGA-BR-4197, TCGA-BR-4199, TCGA-BR-4200, TCGA-BR-4205, TCGA-BR-4259, TCGA-BR-4260, TCGA-BR-4261, TCGA-BR-4263, TCGA-BR-4264, TCGA-BR-4265, TCGA-BR-4266, TCGA-BR-4270, TCGA-BR-4271, TCGA-BR-4272, TCGA-BR-4273, TCGA-BR-4274, TCGA-BR-4276, TCGA-BR-4277, TCGA-BR-4278, TCGA-BR-4281, TCGA-BR-4282, TCGA-BR-4283, TCGA-BR-4284, TCGA-BR-4285, TCGA-BR-4286, TCGA-BR-4288, TCGA-BR-4291, TCGA-BR-4298, TCGA-BR-4375, TCGA-BR-4376, TCGA-DM-A286, TCGA-E2-A1IP, TCGA-F4-6857, TCGA-GN-A261, TCGA-O2-A5IC, TCGA-PN-A8M9

- CCLE cases:

LS123, LS1034

- The number of cases and samples when viewed in the User Interface as compared to the BigQuery tables vary across all three projects (TCGA, TARGET, and CCLE). This is because the user interface reflects the data available at the Genomic Data Commons, whereas data in BigQuery reflects either data at the original TCGA data coordinating center supplemented with Genomic Data Commons Data (for TCGA and CCLE), or for TARGET, data received from the TARGET data coordinating center, not the Genomic Data Commons.
- We have removed Google Genomics functionality from the user interface. You will still be able to access CCLE open access data in Google Genomics from the command line. We are open to adding Google Genomics controlled data back into the user interface if you have a use case for it. Also we are restructuring the handling of multiple Programs of data. Please feel free to provide feedback.
- For TARGET data the clinical and Gene Expression files themselves are available in the system.

Enhancements

- You will be returned a more detailed error message when uploading your own user data.
- On the Data Availability section on the cohort details page now displays the HG38 somatic mutation information for program TCGA.

Bug Fixes

- There is now a 2000 character limit for the workbook title section.
- When selecting the cohort link to complete analysis section on a worksheet will send you to the existing cohort list table page.
- Latency issues when working with the cohort creation page have been resolved.
- When working with TCGA data the IGV browser will not give you a 401 or a 404 error.
- The mouse over feature will display the long name for disease code and project short name for all programs.
- On the cohort creation page you can now filter with the HG38 somatic mutation data by gene for program TCGA using the Molecular tab.
- On the IGV Browser when working with TCGA genomic build hg38 you will no longer get a 404 error.
- On the cohort creation page when working with User Data tab, the left filter panel sorts the other filter.
- Cohorts created with API specific filters are now accessible to access by their cohort details page.
- You are now able to plot miRNA data with genomic build hg38 for TARGET data.

May 25, 2017 v3.0

In collaboration with the GDC we now have TARGET pediatric cancer data available for analysis in the user interface. You are now able to create cohorts and plot analysis with information from TARGET, TCGA, and CCLE data.

In addition, we have replaced the previous APIs with a new version that supports the new user interface.

We have also released the analyzed data types that are based on genome build GRCh38 for TCGA and TARGET data. GRCh37 (HG19) is also still available for TCGA, TARGET, and CCLE datasets.

Workbooks, cohorts, and variables favorites list created before the data structure migration will still be available for analysis and have been labeled as legacy and version 1. If you have difficulty using version 1 workbooks, please contact us

Known Issues

- Analysis Type : Seq peek Formatting Elongated on occasion
- The CCLE data in GUI is not parallel to the CCLE data in BigQuery.
- If the user shares a cohort neither the owner nor the person who was granted access to cohort will receive a confirmation email.
- Cannot plot any data if you use a CCLE data cohort on a worksheet.
- When a user duplicates a worksheet, then tries to implement the log scale it will not function properly.
- On the existing cohorts table list page, the confirmation delete 'blue x' button does not remove selected cohort if you select another option e.g Set Operation. The same issue can be found in reverse if you select the 'blue x' on the confirmation page for set operation you can then select the delete button and see the cohort on the confirmation panel.
- On the cohort view files page there are capitalization bugs on the Platform filter.
- Swap values is not working properly for the plot settings.
- The set operation for existing cohorts complement is behaving exceptionally slow.
- A duplication of the exact cohort happens when you select the confirmation multiple times while the page is loading working with Set Operations.
- When working with a new worksheet or a duplicate worksheet with workbooks for categorical features e.g bar chart you can select the log option. The log option only applies to numerical options.

- When working with workbooks, if you select the delete confirmation button multiple times while the page is loading you will be sent to an error page.
- When working on a scatter plot the Tobacco Smoking being used as the Legend is displayed in numerical values when it should be displayed as categorical values.
- The character limit for a workbook title name is currently inactive, if you exceed the possible limit you will be sent to an error page.
- You currently cannot plot user uploaded data when working with workbooks.
- Selecting cohort from worksheet "To Complete Analysis" section will send you to a 400 Bad Request error.
- You will experience latency issues when working with the create a new cohort page.
- When plotting, certain values will be displayed as numerical when it should be a categorical value e.g Tobacco Smoking History.
- The Data File Availability Panel for program CCLE in currently inactive when on the cohort details page and also editing a cohort with CCLE data.
- On the File List page you currently unable to access the bam files for the IGV Browser associated to build hg38 when working with TCGA data.
- A number of TCGA and CCLE case IDs shown below will have been removed from all cohorts since they are no longer available from NCI's Genomics Data Commons, and ISB-CGC is trying to mirror that data as much as possible.
- TCGA cases:

TCGA-33-4579, TCGA-35-3621, TCGA-66-2746, TCGA-66-2747, TCGA-66-2750, TCGA-66-2751, TCGA-66-2752, TCGA-AN-A

- CCLE cases:

LS123, LS1034 - The number of cases and samples when viewed in the User Interface as compared to the BigQuery tables vary across all three projects (TCGA, TARGET, and CCLE). This is because the user interface reflects the data available at the Genomic Data Commons, whereas data in BigQuery reflects either (for TCGA and CCLE) data at the original TCGA data coordinating center supplemented with Genomic Data Commons Data, or for TARGET, data received from the TARGET data coordinating center, not the Genomic Data Commons. - We have removed Google Genomics functionality from the user interface. You will still be able to access CCLE open access data in Google Genomics from the command line. We are open to adding Google Genomics controlled data back into the user interface if you have a use case for it. Also we are restructuring the handling of multiple Programs of data. Please feel free to provide feedback. - For TARGET data the clinical and Gene Expression files themselves are available in the system. The bam files will be available soon!

Enhancements

- You will be returned a more detailed error message when uploading your own user data.
- The user interface now displays the same nomenclature as the Genomic Data Commons (GDC).

Bug Fixes

- The user data upload is enabled and users can now upload their own datasets and create cohorts using existing programs and newly uploaded data by the user.
- You can now have multiple Google Cloud Projects associated to your account and use only one bucket and dataset on one project with no interference.

April 12, 2017 v1.15

Known Issues

- We are currently having issues viewing bam files using the IGV browser for TCGA and CCLE data. We are working to fix the issue and it should be resolved as soon as possible.

February 26, 2017 v1.14

Known Issues

- Analysis Type : Seq peek Formatting Elongated
- The CCLE data in GUI is not parallel to the CCLE data in BigQuery.
- If the user shares a cohort neither the owner nor the person who was granted access to cohort will receive a confirmation email.
- Cannot plot any data if you use a CCLE data cohort on a worksheet.
- When a user duplicates a worksheet, then tries to implement the log scale it will not function properly.
- On the existing cohorts table list page, the confirmation delete 'blue x' button does not remove selected cohort if you select another option e.g Set Operation. The same issue can be found in reverse if you select the 'blue x' on the confirmation page for set operation you can then select the delete button and see the cohort on the confirmation panel.
- On the cohort view files page there are capitalization bugs on the Platform filter.
- Swap values is not working properly for the plot settings.
- The set operation for existing cohorts complement is behaving exceptionally slow.
- A duplication of the exact cohort happens when you select the confirmation multiple times while the page is loading working with Set Operations.
- When working with a new worksheet or a duplicate worksheet with workbooks for categorical features e.g bar chart you can select the log option. The log option only applies to numerical options.
- If multiple Google Cloud Projects are registered through the user interface, it is advised to to add Google buckets and BigQuery datasets to both projects currently.
- When working with workbooks, if you select the delete confirmation button multiple times while the page is loading you will be sent to an error page.
- When working on a scatter plot the Tobacco Smoking being used as the Legend is displayed in numerical values when it should be displayed as categorical values.
- The character limit for a workbook title name is currently inactive, if you exceed the possible limit you will be sent to an error page.
- We have removed Google Genomics functionality from the user interface. You will still be able to access CCLE open access data in Google Genomics from the command line. We are open to adding Google Genomics controlled data back into the user interface if you have a use case for it. Also we are restructuring the handling of multiple Programs of data. Please feel free to provide feedback.
- There will be a reduced number of releases and features over the next month (or so) while we do some rework required for enabling the distribution of additional data sets and types copied from the NCI-GDC. The new data type is TARGET data, and different analyzed data types are based on the hg38 genome builds. Stay tuned in likely the early part of 2017.
- User data uploads are currently disabled. Any projects you have previously uploaded will continue to be available in your Saved Projects list, and you can continue to work with them, but new data cannot be added at this time. We are working on bringing this function up again, please stay tuned.

Bug Fixes

- User will no longer be sent to the Social Network Login page when trying to login. If this occurs, please feel free to send ISB-CGC feedback using this link feedback.

November 30, 2016 v1.13

Known Issues

- Analysis Type : Seq peek Formatting Elongated
- The CCLE data in GUI is not parallel to the CCLE data in BigQuery.
- User will occasionally be sent to the Social Network Login page when trying to login. If this occurs, please go the the home page of the Web Application and try again.
- If the user shares a cohort they do not receive a confirmation email.
- Cannot plot any data if you use CCLE data cohort on a worksheet.
- When a user duplicates a worksheet, then tries to implement the log scale it will not function properly.
- If a researcher leaves the workbooks inactive the page freezes.
- On the existing cohort list page for the delete button, select the blue x does nothing. It should be disabled.
- On the cohort view files page there are capitalization bugs on the Platform filter.
- Swap values is not working properly for the plot settings.
- Some plot setting are saved or retrieved when working with worksheets.
- The set operation for existing cohorts intersection is behaving exceptionally slow.
- We have removed Google Genomics functionality from the user interface. You will still be able to access CCLE open access data in Google Genomics from the command line. We are open to adding Google Genomics controlled data back into the user interface if you have a use case for it. Also we are restructuring the handling of multiple Programs of data. Please feel free to provide here.
- There will be a reduced number of releases and features over the next month (or so) while we do some rework required for enabling the distribution of additional data sets and types copied from the NCI-GDC. The new data type is TARGET data, and different analyzed data types are based on the hg38 genome builds. Stay tuned in likely the early part of 2017.

Bug Fixes

- The user can no longer see BCGSC expression as an option when plotting genes if user does not select center filter on worksheet.
- Worksheets added to an existing workbook now behave the same as the original worksheet.
- Cohort set operations no longer performing exceptionally slow.

November 16, 2016 v1.12

Known Issues

- Analysis Type : Seq peek Formatting is Elongated
- The CCLE data in GUI is not parallel to the CCLE data in BigQuery.
- User will occasionally be sent to the Social Network Login page when trying to login. If this occurs, please go the the home page of the Web Application and try again.
- If the user shares a cohort they do not receive a confirmation email.
- Cannot plot any data if you use CCLE data cohort on a worksheet.
- When a user duplicates a worksheet, then tries to implement the log scale it will not function properly.
- If a researcher leaves the workbooks inactive the page freezes.

- On the existing cohort list page for the delete button, selecting the blue x does nothing. It will be be disabled in a future release.
- On the cohort view files page there are capitalization bugs on the Platform filter.
- Swap values is not working properly for the plot settings.
- Some plot setting are saved or retrieved when working with worksheets.
- Worksheets added to an existing workbook behave differently than the original worksheet.
- The user can see BCGSC expression as an option when plotting genes if user does not select center filter on worksheet.
- The set operation for existing cohorts intersection is behaving exceptionally slow.
- We are removing Google Genomics from the user interface. You will still be able to access CCLE open access data in Google Genomics from the command line. We are open to adding Google Genomics controlled data back into the user interface if you have a use case for it. Please feel free to provide feedback.

Enhancements

- A warning will be displayed if the user is trying to plot with required data missing e.g. must select an analysis, gene or variable, and a cohort to create a plot.
- On the project details page user will be sent to upload new study in existing project tab when they select upload data.
- When the user plots a graph with NA values, you will be returned a notification stating no valid data was found.
- There is no longer text overlapping on the Cloud Hosted Datasets readthedocs page in the documentation.

Bug Fixes

- The user can no longer add the same gene symbol twice if list to the same worksheet even if they have given their list different names.
- When the user selects multiple cohorts for color by feature for scatter plot all cohorts selected display on the graph.
- On the existing cohorts table for public cohorts, the new workbook and set operations buttons are now active.
- For all analysis types the x-axis and y-axis with certain variables text will no longer overlap and is displayed clearly.
- The upload data button is disabled on the review files page when no buckets or datasets are associated.
- Someone with multiple eRA accounts will be no longer have issues when trying to access controlled data.

November 2, 2016 v1.11

Known Issues

- The user can add same gene twice if list to the same worksheet it they have different names.
- Analysis Type : Seq peek Formatting Elongated
- The CCLE data in GUI is not parallel to the CCLE data in BigQuery.
- If a user creates a cohort with sample type filter Cell Lines and CCLE the total number of samples count off by one.
- User will occasionally be sent to the Social Network Login page when trying to login. If this occurs, please go the the home page of the Web Application and try again.
- If the user shares a cohort they do not receive a confirmation email.
- When the user selects multiple cohorts for color by feature for scatter plot they do not display in chart.

- Cannot plot any data if you use CCLE data cohort on a worksheet.
- When the user plots a graph with NA values the UI returns a blank graph.
- When a user duplicates a worksheet, then tries to implement the log scale it will not function properly.
- If a researcher leaves the workbooks inactive the page freezes.
- On the existing cohort list page for the delete button, selecting the blue x does nothing. It should be disabled.
- On the cohort view files page capitalization bugs on the Platform filter.
- Swap values is not working properly for the plot settings.
- Some plot settings are saved or retrieved when working with worksheets.
- On the existing cohorts table for public cohorts, the new workbook and set operations buttons are currently inactive.
- Worksheets added to an existing workbook behave differently than the original worksheet.

Enhancements

- Introduce user data upload functionality see documentation here.
- More fluid zoom feature when working with analysis worksheets.
- Case Sensitivity is now maintained in creating and displaying Workbook names throughout the entire User Interface.
- You can now create a new cohort from the menu bar.
- Variables menu bar is displayed similar to the rest of the favorites variables.
- On the dashboard, all create new buttons/links are identical.
- Owner of what is shared either a workbook or a cohort is able to remove multiple viewers. Viewers are also able to remove themselves.
- Removed BCGSC gene expression from the UI gene specification selection for plot analysis.

Bug Fixes

- X or Y- Axis for text no longer overlaps on worksheet for any analysis type, except for violin plot.
- The Legend is no longer displayed elongated when you use multiple cohort for color by feature for violin plot.
- miRNA_expression_values_fixed table in dataset 2016_07_09_tcga_data_open reflect only hg19.mirbase20 files.
- You are now able to duplicate a workbook that has been shared with you by someone else.
- Added pseudo-counts to the mosaic plots on the create new cohort page. This allows you to be sure of always being able to see (and select) the smallest contributors in these mosaics.
- Removing the filter from the filter confirmation from the create new cohort page, this will remove it from the rest of filter selections.
- Select the "check-all" feature on the create new cohort page will no longer cause duplicates on the selected filters panel.
- Create cohort from plot selection now works with all analysis types.
- Data inconsistencies between the create new cohort histogram filter and the most recent BigQuery datasets has been addressed and resolved.

September 21, 2016 v1.10

Enhancements

- Text in confirmation box of a duplication of a workbook has been enhanced.
- On the registered Google Cloud Projects page, icon has been added for the user to go directly to the Google Cloud Console page if desired.
- When the a Service Account is removed from the Access Control List, the project owner is sent an email with an explanation as to why the account was removed.
- IGV File List page displays of which page user is browsing.

Bug Fixes

- For a Cubby hole plot the x - axis name can be seen clearly.
- On a duplicate worksheet when working with gene specifications, user is able to select between all options multiple times.
- Page becomes elongated when the user builds a Cubby Hole plot.
- The selected variables for the plot setting on a worksheet are saved after the user leaves the workbook.
- When registering a Google Cloud Project the user is displayed the list of emails associated to the GCP only once.

Known Issues

- The user can add same gene twice if list to the same worksheet it they have different names.
- The Bar chart on the worksheet panel renders overlapping text.
- Analysis Type : Seq peek Formatting Elongated
- The CCLE data in GUI is not parallel to the CCLE data in BigQuery.
- If a user creates a cohort with sample type filter Cell Lines and CCLE the total number of samples count off by one.
- User will occasionally be sent to the Social Network Login page when trying to login. If this occurs, please go the the home page of the Web Application and try again.
- If the user shares a cohort they do not receive a confirmation email.
- The Legend is displayed elongated when you use multiple cohort for color by feature for violin plot.
- When the user selects multiple cohorts for color by feature for scatter plot they do not display in chart.
- Cannot plot any data if you use CCLE data cohort on a worksheet.
- When the user plots a graph with NA values the UI returns a blank graph.
- When a user duplicates a worksheet, then tries to implement the log scale it will not function properly.
- There are duplicate rows in the molecular data table in BigQuery.

September 7, 2016 v1.9

Enhancements

- Dictionary mapping feature types to units for use in plot displays added to worksheets.
- The user now has the option to make the axis logarithmic if the plot can display continuous numerical data for eg. mRNA expression levels.
- The NIH username entry is now case insensitive for dbGaP authorization.
- The mouse over feature works when the user has created a long workbook name on the existing workbooks table page.
- The mouse over functionality was added to the worksheet name within a workbook.

Bug Fixes

- The order by ascending or descending feature is now working properly for the existing workbooks table page.
- Tobacco Smoking History filter in the create cohort page displays the filters in descriptive values.
- The user can now select all existing cohorts when on the add cohort(s) to worksheet page.
- The gene specification selection on the worksheet page is now working properly.
- When a user shares a workbook with someone the person who received viewer access to the workbook is sent a confirmation email. If the person who shared the workbook then deletes the workbook before it's opened, then the person clicks the invitation link the person is sent to the unknown invitation page. The button to go back to the Dashboard page appears like this, "Your Dashboard"
- The user is sent an email when the Service Account is removed the Access controlled list for having a user associated to the project who is not dbGaP authorized.

Known Issues

- The user can add same gene twice if list to the same worksheet it they have different names.
- The Bar chart on the worksheet panel renders overlapping text.
- Analysis Type : Seq peek Formatting Elongated
- The CCLE data in GUI is not parallel to the CCLE data in BigQuery.
- If a user creates a cohort with sample type filter Cell Lines and CCLE the total number of samples count off by one.
- User will occasionally be sent to the Social Network Login page when trying to login. If this occurs, please go the the home page of the Web Application and try again.
- Page becomes elongated when the user builds a Cubby Hole plot.
- X-axis name cut off for cubby hole plot when x-axis has only 3 criteria.
- If the user shares a cohort they do not receive a confirmation email.
- The Legend is displayed elongated when you use multiple cohort for color by feature for violin plot.
- When the user selects multiple cohorts for color by feature for scatter plot they do not display in chart.
- When the user creates a duplicate worksheet,the bar chart with a gene with specification protein can freeze when selecting an option for the Select Feature.
- Cannot plot any data if you use CCLE data cohort on a worksheet.
- When the user plots a graph with NA values the UI returns a blank graph.
- When a user duplicates a worksheet, some functionality related to plotting will not function properly on the duplicate worksheet.

August 24, 2016 v1.8

Known Issues

- The user can add same gene twice if list to the same worksheet it they have different names.
- The Bar chart on the worksheet panel renders overlapping text.
- Analysis Type : Seq peek Formatting Elongated.
- The CCLE data in GUI is not parallel to the CCLE data in BigQuery.
- If a user creates a cohort with sample type filter Cell Lines and CCLE the total number of samples count off by one.

- User will occasionally be sent to the Social Network Login page when trying to login. If this occurs, please go the the home page of the Web Application and try again.
- Page becomes elongated when the user builds a Cubby Hole plot.
- X-axis name cut off for cubby hole plot when x-axis has only 3 criteria.
- When the user shares a cohort they do not receive a confirmation email.
- User will be spammed with email every one minute when their service account is removed from the ACL control list. To stop this, please either delete your service account from the ISB-CGC interface, or remove the GCP project member(s) who is (are) not authorized to access the controlled data set. (see documentation here). We are planning to reduce the frequency of the notification emails to once per day.
- The Legend is displayed elongated when you use multiple cohort for color by feature for violin plot.
- When the user selects multiple cohorts for color by feature for scatter plot they do not display in chart.
- When the user creates a duplicate worksheet,the bar chart with a gene with specification protein can freeze when selecting an option for the Select Feature.
- When a user shares a workbook with someone the person who received viewer access to the workbook is sent a confirmation email. If the person who shared the workbook then deletes the workbook before it's opened, then the person clicks the invitation link the person is sent to the unknown invitation page. The button to go back to the Dashboard page appears like this, "Your Dashboard{"
- Cannot plot any data if you use CCLE data cohort on a worksheet.

Enhancements

- When the researcher is on the Register Service Account page, after they have submitted the Service Account associated to their Google Cloud Project a table that shows who is authorized will be prompted.
- There is now a column that says "Has NIH Identity", before it said, "Has eRA Commons".
- When the researcher creates a new cohort with more than 20 filters chosen the URL exceeds the limit of 2K characters and this affects the count for the Details panel. Therefore the user is now prompted with an alert box that will say, "You have selected too many filters. The current counts shown will not be accurate until one or more filter options are removed." if this is ever the case.
- In the user details page, if the researcher has not registered a Google Cloud Project it will say, "Register a Google Cloud Project" on the link.

Bug Fixes

- The researcher can now delete whom they share cohort with from existing cohorts table.
- After 24-hours of use, a dbGaP authorized user can re-authenticate through the link provided in the user details page.
- The variable favorites list table page can now support a long title for the variable list.
- The filter name will appear aligned in the verification panel when the filter is name too long for the create in cohort filter confirmation selection on the create new cohort page.
- Grouped Data Type filter counts (Methylation, RNA Seq, miRNA Seq) now behave like the other count groups. The counts will behave as grouped values.
- The user can no longer select a categorical variable for selection for Histogram plot.
- The Filter token displays are now shown in 'readable' names when working with cohort filters.
- Controlled access BAM files are now viewable viewable in the IGV browser after the user has authorized their credentials.
- The user can now unlink an eRA commons account from their Google Identity in the user detail page.

- The violin plot was inconsistently failing. We have updated the JavaScript, therefore the Violin plot no longer fail.

August 10, 2016 v1.7

New Features

- The researcher can now create a cohort of participants and samples based on the presence of a gene mutation in a specified gene. Look for the new "Molecular" tab when you are creating a cohort.
- The bioinformatics programmer now has the ability to associate their Google Cloud Project's Service Account. This allows the researcher to run computational pipelines from Google Virtual Machines using TCGA Controlled data (e.g. BAM files) for seven days before they have to reauthorize. For more information please select here.

Known Issues

- The user can add same gene twice if list to the same worksheet it they have different names.
- The Bar chart on the worksheet panel renders overlapping text.
- Cannot delete whom you share cohort with from existing cohorts table.
- Analysis Type : Seq peek Formatting Elongated
- The CCLE data in GUI is not exactly coordinated the CCLE data in BigQuery.
- If a user creates a cohort with sample type filter Cell Lines and CCLE the total number of samples count is off by one.
- After 24-hours of use, a dbGaP authorized user has to logout and then log back in to be prompted with NIH login link to re-access controlled data.
- User will occasionally be sent to the Social Network Login page when trying to login. If this occurs, please go the the home page of the Web Application and try again.
- Page becomes elongated when the user builds a Cubby Hole plot.
- X-axis name cut off for Cubby Hole plot when x-axis has only 3 criteria.
- When the user shares a cohort they do not receive a confirmation email.
- When a name is too long for variable favorites list table, the Last Updated" column will appear cut off.
- Filter name will appear off the verification panel when the filter is name too long for the create in cohort filter selection.
- Grouped Data Type filter counts (Methylation, RNA Seq, miRNA Seq) don't behave like other count groups. The counts behave as though the values were for distinct categories.
- User will be spammed with email every one minute when their service account is removed from the ACL control list. To stop this, please either delete your service account from the ISB-CGC interface, or remove the GCP project member(s) who is (are) not authorized to access the controlled data set. (see documentation here). We are planning to reduce the frequency of the notification emails to once per day.
- The user can select a categorical variable for selection for Histogram plot, and will return a graph with no data.
- The Legend is displayed elongated when you use multiple cohort for color by feature for violin plot.
- When the user selects multiple cohorts for color by feature for scatter plot they do not display in chart.
- When the user creates a duplicate worksheet,the bar chart with a gene with specification protein can freeze when selecting an option for the Select Feature.

Enhancements

- The user now has the option to select all or deselect all possible filters for any tab that has more than 10 possible options in the create new cohort page.

- The user can now set all existing tables by either ascending or descending order.
- The cohort_id has been added to the detail cohort page. This allows the user to reference a desired cohort with ease in the API endpoints.
- When creating a new cohort, the user is given the full description for sample type in the selected filters panel.

Bug Fixes

- Histological Type entries in create new cohort page on the user interface now match the Google BigQuery entries in terms of capitalization.
- Filters for data type counts in left panel currently is now working properly.
- When a user sets a cohort as Color by feature for violin plot legend will be set to cohort. Then when the user sets another color by feature it will update the legend.
- The user can no longer make a gene list without selecting a gene first.
- The user can now list the Last Modified section for the existing cohort table by either ascending or descending order.
- In the create new cohort page for the data type tab, the user can now select either True or False for DNA Sequencing, Protein, and SNP Copy Number filters.
- When the user edits a new cohort and sets the edited cohort to return zero samples, the user will be prompted to select different set of filters.

July 20, 2016 v1.6

Known Issues

- The user can add same gene twice if two identical worksheets with different names are uploaded.
- The Bar chart on the worksheet panel renders overlapping text.
- User cannot delete whom you share cohort with from existing cohorts table.
- Analysis Type : Seq peek Formatting Elongated.
- The CCLE data in GUI is not parallel to the CCLE data in BigQuery.
- If a user creates a cohort with sample type filter Cell Lines and CCLE the total number of samples count off by one.
- Histological Type entries in create new cohort page on the user interface should match the Google BigQuery entries in terms of capitalization.
- When a user sets a cohort as Color by feature for violin plot legend will remain cohort.
- After 24 hour dbGaP authorization runs out the user is unable to re authenticate. (If you have this issue, please log out and log back in to be prompted with login link for dbGaP authorization.)

Enhancements

- Created ability in GUI to make cohorts based on presence of an HPV status.
- Created ability in GUI to make cohorts based on BMI value.
- In the details panel for existing cohort have a section that shows the ISB-CGC cohort_id.
- Enhancements of GUI to view submenu item in different screen sizes and resolutions.
- New version of IGV javascript installed.

Bug Fixes

- User can no longer add same filter to existing cohorts.

- Optimized Security in the user interface.
- If a user opens a shared cohort it will appear once on the dashboard.
- Pathologic State Filter in create cohort Stage is displayed capitalized.
- Filter counts with 0 value do list when editing a pre-existing cohort.
- Filters for data type counting in left panel is working properly.
- After 24 hour dbGaP authorization runs out the user is able to re authenticate.
- User can not create new gene list without giving the gene list a name.

July 6, 2016 v1.5

Known Issues

- The user can add same gene twice if list to the same worksheet it they have different names.
- The user can add same filter to existing cohorts.
- The Bar chart on the worksheet panel renders overlapping text.
- Cannot delete whom you share cohort with from existing cohorts table.
- Analysis Type : Seqpeek Formatting Elongated.
- The CCLE data in GUI is not parallel to the CCLE data in BigQuery.
- If a user opens a shared cohort it will appear twice on the dashboard.
- If a user creates a cohort with sample type filter Cell Lines and CCLE the total number of samples count are off by one.
- Pathologic State Filter in create cohort Stage should be displayed capitalized.
- Histological Type entries in create new cohort page on the user interface should match the Google BigQuery entries in terms of capitalization.
- Filter counts with 0 value don't list when editing a pre-existing cohort.
- Filters for data type counting in left panel currently is not working properly.

Enhancements

- A user can only select the cloud storage checkbox if he or she has been authenticated and authorized through the user details page. Otherwise the user can view the cloud storage checkbox but there will be a disabled cursor icon when the user hovers over in an attempt to select the checkbox.
- The counts for the queries were refactored to match what was done for the APIs .
- The Download File List as CSV was refactored to a maximum of 65,000 files at once.
- Date formats on Workbooks, Cohort, Gene, and Variables list pages all reflect the same format.
- The Last Updated columns to variable and gene lists were added to the user Dashboard

Bug Fixes

- The user can now select a cohort in the color by feature section for the violin and the scatter plots in the worksheet section.
- The Gene list variable used for analysis in the worksheet plot settings section is the exact gene as compared to a gene that contains the string.
- The Comments button for both the workbook and the cohort section, when the user clicks the request multiple times within one second the user interface will not post duplicate comments in the comments section.

- The user can now select gene HP in Create Gene list favorite page to be used for analysis. For worksheet analysis the user now has ability to select different genes once one already selected and utilized for analysis.
- In the variable favorites table, the menu for a specific variable will no longer be cut off once a certain set of variables list are exceeded.
- A 400 Error pop up window will no longer appear as the user transitions from the File List page to IGV browser page.
- The Public Data Availability section will no longer display any cut off if the user drags data type to the left of the page away from the panel itself, in detail page of existing cohort or the create new cohort page.
- When the user edits a cohort, details section will display which filter(s) were applied for each update.
- Cloud storage path in CSV file download for GA/BCGSC and GA/UNC V2 platforms can now be viewed.
- The menu bar will display existing list for variable favorites list, gene favorites list, cohorts, and workbooks with no cut off.
- When the user has selected a variable for the y-axis, the chart will display the selected variable in the charts.
- When the user clicks Save Changes when modifying an existing cohort the user can will no longer be spammed with multiple cohorts created at once when clicking the button multiple times within one second.
- The Save cohort Endpoint default example for v1 now works properly.
- For the cohort_list API endpoint v1 will now pull only the cohort_id you specified.

June 8, 2016 v1.4

Known Issues

- The user can add same gene twice if list has different names.
- The user can add same filter to existing cohorts.
- In the Create new Cohort page, the left filters (#) does not re-populate as you select filters to match the sample number in clinical feature panel.
- The bar chart renders overlapping text in the x-axis and y-axis for certain variables.
- A user cannot delete whom you share a cohort with from the existing cohorts table.
- On a worksheet with the Analysis Type : Seq peek, the formatting will display Elongated when the user selects a certain gene.
- CCLE data in GUI is currently not parallel the CCLE data in BigQuery.
- User currently cannot select a cohort in the color by feature section in a worksheet.
- The Gene list used for analysis currently uses genes similar as to original gene and well as the specific gene added to list, in the plot settings menu.
- The comments button for both workbooks/cohorts, if user clicks the comment button multiple times within one second will post duplicate comment.
- User currently cannot select gene HP or gene's with only two letters in the Create Gene list favorite page.
- In Violin plot - the user has no ability to select a different gene once one is already selected.
- In the variable favorites table, the menu for a specific variable will be cut off once a certain set of variables list are exceeded.
- A 400 Error pop up window will appear as the user transitions from the File List page to IGV browser page.
- Public Data Availability section will be cut is user drags data type title to the left of the page away from the panel itself,in detail page of existing cohort.

Enhancements

- Upgraded system from using Django 1.8 to Django 1.9.
- A link to the google cloud platform has been added to the user details page.
- The TCGA filter is selected as the default project when creating a new cohort.
- When the user clicks on the browser back button, the user will remain on the same worksheet that they were previously on.
- When the user goes adds a new gene list, variable favorites list, and/or cohort from the worksheet data type panel, the button will display "Apply to Worksheet".
- The feedback/help section has been moved to the top of the page to provide the user a more convenient way to send us feedback.

Bug Fixes

- User can no longer add a duplicate gene to same gene favorites list.
- To edit a gene name the user must now delete and re-type the desired gene name.
- The functionality of a duplicate worksheet drop down menu reflects the same functionality of the original worksheet.
- The Last Updated section reflects any changes made to the variable list, cohort list, and gene list in their corresponding tables.
- The File list page now allows the user to add a maximum of five files to use in the IGV browser between all the pages in the file list table.
- When a user hovers over clinical feature panel for Sample Type and Tumor Tissue Type the top row when hovered over the name is displayed clearly.
- Order by Ascending/Descending is working properly for Existing Cohorts table page.
- The user is now able to plot gene's with a hyphen(-) in the gene name itself.
- The user is now able to download a maximum of 85,000 files at a time, in the File List page for a selected cohort.

May 10, 2016 v1.3

Known Issues

- A user can add same gene twice if identical gene list have different names.
- The user can add same filter already selected to an existing cohort.
- The create new Cohort left filters number count does not re-populate as you select filters to match sample number count in clinical feature panel.
- When a Bar chart renders overlapping text is displayed on the x-axis of the plot.
- Cannot delete whom you share a cohort with from the existing cohorts table only from the details page of a cohort.
- Analysis Type : Seq peek formatting is elongated when a user selects certain gene for analysis. Using the gene TP53 can reproduce this issue.
- The CCLE data in GUI currently does not parallel the CCLE data in BigQuery.
- A user can add a duplicate gene to same gene favorites list in the create new gene list page.
- By double clicking a gene name in the create new gene list page, the gene will expand but display a blank space.
- A duplicate worksheet will display the color by feature variables twice in the drop down list.

- A user currently cannot select a cohort in the color by feature section.
- The Gene list drop down list used for analysis should be exact gene only.
- The comments button for both workbook and cohort comments section, if the user is to click comment button multiple time within one second, this action will post a duplicate comment.
- The last Update section should reflect any changes made to variable list, cohort, and gene list for their corresponding tables.
- The user cannot select the gene HP in the Create Gene list favorite page.

Enhancements

- Data Use Certification Agreement link updated and the help link was removed. -
- The Data Type section in the Create new Cohort page name change from MIRNA Sequencing to miRNA Sequencing and SNP CN to SNP Copy-Number.
- The number of patients is now dynamically displayed in the create new cohort page when selecting filters in the details panel.
- The number of samples is now dynamically displayed in the create new cohort page when selecting filters in the details panel.
- By default in the create new cohort page, you will have the TCGA data filter selected.
- When creating a cohort, checking feature boxes will be throttled so as to avoid miss-represented data.
- Tooltips were added to the Sample Type section in the clinical features panel.
- Minor changes were made in personal details page.

Bug Fixes

- The Clinical Features Panel in the create new cohort page will no longer display BRCA even if unselected.
- The last updated section in existing workbooks panel does update when changes are made to existing workbook.
- Set operation Union patient number is working correctly.
- Upon duplicating a cohort it will duplicate the selected filter(s) as well.
- User is able to download file list as csv for any cohort with any filter selected.
- There is no legend cut off for violin plot or any other analysis type when the color by feature is set to Prior Diagnosis or any other variable.
- When user switches gene in plot settings the feature choices for that specification will refresh.
- The variable clinical search feature works properly when the user searches for clinical variables and then are used for analysis.

April 27, 2016 v1.2

Known Issues

- Can add same gene twice if list has different names.
- User can add same filter to existing cohorts.
- Create new Cohort left filters (#) does not re-populate as you select filters to match sample # in clinical feature panel.
- Clinical Features Panel in create new cohort page will still display BRCA even if unselected.
- Last updated section in existing workbooks panel does not update when changes are made to existing workbook.
- Bar chart renders overlapping text.

- Set operation Union patient # off by one.
- Legend Name cut off when name is too long.
- Upon duplicating a cohort it duplicates the selected filter as well.
- Cannot delete whom you share cohort with from existing cohorts table.
- Unable to down file list as csv for any other cohort only selected filter CCLE.
- Legend Cut Off for violin plot when color by feature set to Prior Diagnosis.
- When user switches gene in plot settings the feature choices for that specification disappears.

Enhancements

- The comments section now has a max number of characters 1000 limit.
- Link created to Extend controlled access period to 24-hours from the moment the link is clicked.

Bug Fixes

- A user can now click new worksheet multiple times within a few seconds and only produce one sheet.
- The user must now add a new filter in an existing cohort to edit it the cohort.
- The duplicate button for an existing cohort will only make one duplicate at a time.
- Clicking 150+ selected filters will not create an error page.
- Cancel button on Create new gene list page will send you to Gene list favorites table menu.
- Violin plot : User can not add categorial value to y-axis.
- If user edits an existing cohort, the old filter(s) will not be removed.
- If a new worksheet is generated, the worksheet functionality is working properly.
- User will get the '500: There was an error while handling your request. If you are trying to access a cohort please log out - and log back in. Sorry for the inconvenience.' if the user is inactive for more in 15 minutes when trying to create/use existing cohort.
- Clinical Feature Panel is displayed properly and reacts to filters being added/removed quickly.
- The user must have text to add a comment.
- All columns in file list table will be transferred/displayed when exported as csv file.

April 14, 2016 v1.1

Known Issues

- If user clicks create in new worksheet too many times within a few seconds will create duplicate worksheets
- Can add same gene twice if list has different names
- Apply filters button work when no filter is selected in edit cohorts page
- If user clicks create in new cohorts too many times within a few seconds will create duplicate cohorts
- User can add same filter to existing cohorts
- Clicking 150+ selected filters will create error page
- Create new Cohort left filters (#) does not re-populate as you select filters to match sample # in clinical feature panel
- Clinical Features Panel in create new cohort page will still display BRCA even if unselected
- Cancel button on Create new gene list page will send you to Data Source | Gene Favorites page

- Violin plot : User can add categorial value to y-axis
- Last updated section in existing workbooks panel does not update when changes are made to existing workbook
- If user edits an existing cohort the old filter(s) will be removed

Enhancements

- Tool tips added for disease code in create new cohort page
- Disease in longname in tool tips the first letter is capitalized

Bug Fixes

- The user detail page will now display the correct date
- The plot settings for a new worksheet are now working properly
- Plot settings for duplicate worksheets are now working properly
- The plot settings will now match the analysis type for existing worksheet plot
- The user can now edit existing cohort name
- Set Operations : Intersection working properly
- Set Operations : Union working properly
- Set Operations : Complement is now working properly
- User is now able to delete selected filters from selected filter panel in new cohort page using the blue X
- Editing an existing variable favorites list will display previously selected variables
- (Already in documentation) Green checkmark will appear for IGV link
- Update plot button will now work on a duplicate worksheet(can be added with 3)
- User can now delete all cohorts with the select all feature
- Fixed bugs with Data Type Create new cohort generating errors
- The user can now search for variable favorite with the miRNA feature
- The user can now search for a variable favorite through the clinical search feature

March 14, 2016 v1.0

- When working with a worksheet two plots will be generated occasionally.
- Axis labels and tick values sometimes overlap and get cutoff.
- Page elongated when Cubby Hole plot generated and there are lots of values in the y axis.

December 23, 2015 v0.2

- Treemap graphs in cohort details and cohort creation pages will not apply its own filters to itself. For example, if you select a study, the study treemap graph will not update.
- Cohort file list download not working.

December 3, 2015 v0.1

- First tagged release of the web-app

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CHAPTER

TWENTYEIGHT

QUICK LINKS

The table below describes the types of functions that can be performed in ISB-CGC and provides links to documentation on how to do those functions.

Function	In web application	With Command Line
Work on the Google Cloud Platform	• Getting a trial account on the GCP sponsored by Google • Getting a trial account on the GCP sponsored by ISB-CGC • Specific information on how to set up your GCP project for use with ISB-CGC	• Google Cloud SDK • Google Cloud Shell • Authenticating with Google
Understand details of the data	• Access controlled data • View File List in GUI • View Sequences with IGV • Google BigQuery Walkthrough • Google BigQuery Console	• For data in BigQuery • BigQuery Command Line Tool • Visualize results from BigQuery • For data in Google Cloud Storage • Google Cloud Storage JSON API • Google Cloud Storage gsutil • How to work with cloud storage notebook
Analyze ISB-CGC data	• Analyses with Bar Charts, Histograms, Scatter Plots, Violin Plots, Cubby Hole Plots and SeqPeek • Google BigQuery Web UI and BigQuery UI Walkthrough	• Community Notebooks • ISB-CGC Endpoints
Create cohorts of patients	• Create cohorts • Cohorts set operations • Create a cohort from visualization (background and tool image) • Develop queries with Google BigQuery Web UI and/or ISB-CGC API v4 before creating cohorts with command line tools	• Use GUI saved cohorts in ISB-CGC Endpoints • BigQuery Command Line Tool, and/or BigQuery REST API
Add your data to the cloud	• Uploading your low level data (e.g. Bam and VCF files) to Google Cloud Storage • Upload high level summary data (such as Methylation, Gene Expression, microRNA, Protein Expression & Customized data) for plotting in the UI.	• Uploading your data to Google Cloud Storage • Uploading your data to Google BigQuery
Analyze your data with ISB-CGC data	• Plot your high level experimental data (such as Methylation, Gene Expression, microRNA, Protein Expression and Customized data) with other ISB-CGC	• Community Notebooks

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CHAPTER

TWENTYNINE

FREQUENTLY ASKED QUESTIONS (FAQ)

29.1 ISB-CGC Accounts and Cloud Projects

29.1.1 Do I have to request an ISB-CGC account before I can try the web app?

No, you can just "sign in" to the Web App using your Google identity.

29.1.2 I want to be able to run big jobs using Google Compute Engine on the TCGA data hosted by the ISB-CGC. What should I do?

You will need to request a Google Cloud Platform (GCP) project. Please see How to Request Cloud Credits for more details about requesting a project.

29.1.3 Can I use any email address as a Google identity?

Yes, you can. If your email address is not already linked to a Google account, you can create a Google account with your current email address. Please note, however that although these two accounts will then share the same *name*, they will still be two separate accounts, with two separate passwords, *etc*. (It is also possible that your institutional email address is *already* a Google account, if your institution uses Google Apps. This is how to find out).

29.1.4 How do I connect my Google Cloud Project to the ISB-CGC?

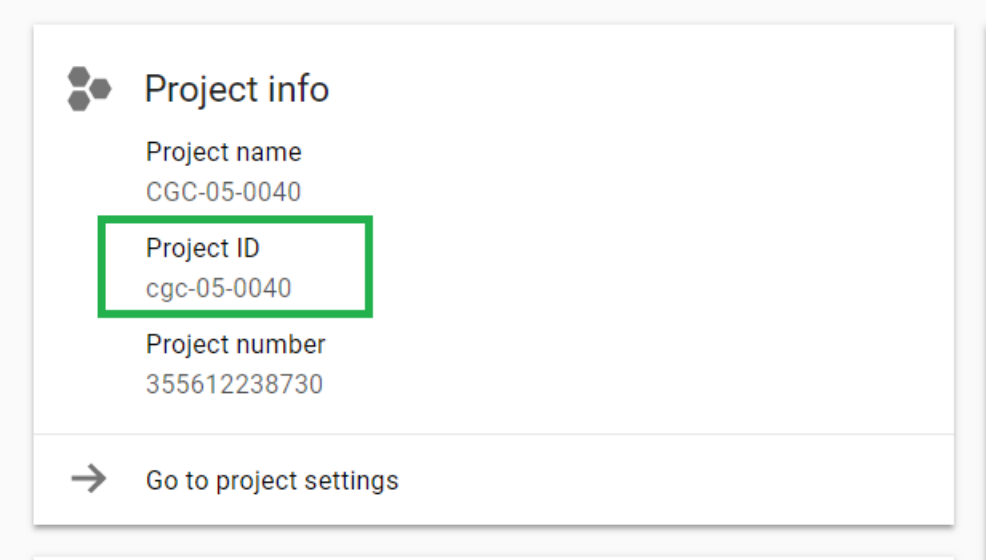
Your Google Cloud Project gives you access to all of the technologies that make up the Google Cloud Platform. These technologies include BigQuery, Cloud Storage, Compute Engine, etc. The ISB-CGC makes use of a variety of these technologies to provide access to the TCGA data, as well as many other data sets. Please see the Google Cloud Project Setup and Data Access section in the Quick Start Guide.

The connection between your Google Cloud Project (whether it is an ISB-CGC sponsored and funded project or your own personal project) and the ISB-CGC is your Google identity (also referred to as your "user credentials").

Access to all ISB-CGC hosted data is controlled using the Data Commons Framework Gen3 which defines the permissions attached to each data set, bucket, or object.

29.1.5 What project information do I input on the Register a Google Cloud Project page?

You will need to input the Google Cloud Project ID which can be found on the Dashboard page of the Google Console under Project info.



29.1.6 Why do I add the service account 907668440978-oskt05du3ao083cke14641u35deokgjj@developer.gserviceaccount.com to my Google Cloud Project?

This service account is needed in your Google Cloud Project IAM page for the ISB-CGC project to be able to automatically verify that all users of your Google Cloud Project have the same appropriate access rights to the protected data that has been requested for the project.

29.1.7 What service account do I use on the Register a Service Account page to be able to gain access to protected data?

On the Register a Service account page you are asked to input a service account ID. You need to go to the IAM and Admin page which can be found in your console for your Google Cloud Project to find the correct service account. The service account you would like to use is named, "Compute Engine default service account". This service account is the default option on the Register A Service Account page. *Please DO NOT use the service account 144657163696-utjumdn9c03fof16ig7bjak44hfj53o6@developer.gserviceaccount.com (you will be prevented from using this account by our software and an error message will be sent indicating this).*

29.1.8 Why can't I reauthorize my Service Account on my Google Cloud Project?

Your service account may have had its permissions revoked (because, for example, the 7-day limit has been reached, or you have added a member to the GCP who is not authorized to use controlled data the service account is linked with or has not logged into the ISB-CGC UI and authenticated using their dbGaP credentials). If permissions were revoked because an unauthorized user was added to the project, the Google Cloud Project owner will be sent an email specifying the Service Account, and Google Cloud Project which resulted in the access being revoked. If the user has not logged into the ISB-CGC Web App and/or has not authenticated, you will be given a red error message saying, "There was an error in processing your service account. Please try again." when attempting to refresh using the refresh wheel. To see which new user hasn't logged in or authenticated, please go to either the Register a Service Account page or the Adjust a Service Account page and see which user it is within the table for which the data set is not selected and there are X's in the Registered and Has NIH Identity.

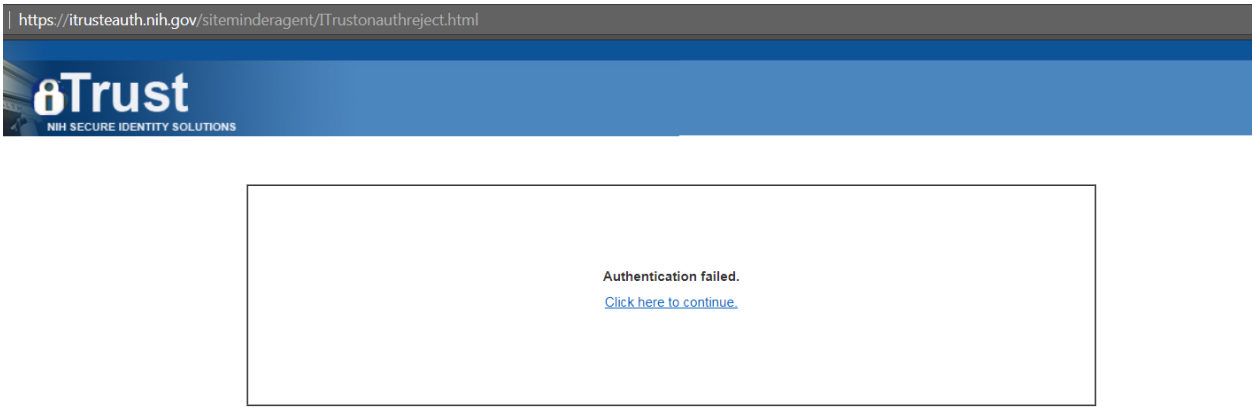
User Email	Registered	Has NIH Identity	Authorized Datasets
madelynestr@gmail.com	✖	✖	✔All Open Datasets

Ensure that the user has 1) Logged into the ISB-CGC web app and 2) Has registered their NIH Identity with their user interface identity.

To reauthorize the service account 1) Remedy the problem that resulted in access being denied, and 2) Select the "Adjust A Service Account" icon(plus sign) next to Current Access Expires.

Another reason could be if some users are marked as unable to access datasets they should have access to, make sure they have logged into the system and linked their eRA Commons/NIH Identity to their Google Identity.

29.1.9 Why would I get an Authorization Failed page on NIH iTrust when attempting to link my Google Account with my NIH identity?



You can get this page for two reasons: First, if you may have typed in your password incorrectly, please select the Click Here to continue link and try to log in again. Second, if you have typed your password correctly, it could be time to refresh your NIH identity password. Please reset your password by using this link here and try again. This should allow you to link your NIH Identity to the ISB-CGC web app.

29.1.10 What happens if I accidently delete the default service account from a Google Cloud Project?

If you accidently delete the default service account associated to the Google Cloud Project you are working in, you can no longer authorize the service account during instance creation, associate the service account to controlled access data, and many other functionalities will no longer work.

If you then try to add the service account back to the Google Cloud Project, this error occurs:

> *ERROR: (gcloud.compute.instances.create) Some requests did not succeed: - The resource 'xx… @project.gserviceaccount.com' of type 'serviceAccount' was not found.*

Unfortunately at this time, there is no direct way to recover the default service account.

One workaround to recreate the Google Compute Engine default service account is to disable and reenable Google Compute Engine API in your project. This will only work if you have no Google Compute Engine resources (e.g VMs, Disks, Snapshots etc) in your project; otherwise, you will get "Backend Provisioning Error" when you try to disable Compute Engine API.

Another solution would be creating a new project and redeploying your instances there.

Google has an internal feature request to prevent accidental deletion of default service accounts.

There is a Google forum discussion that can be found here with more details and explanation.

29.2 ISB-CGC Web Interface

29.2.1 I ran the same query in the Web App that I've run before, but the results were different. Why is that?

The Web App performs its data retrieval and counts on ISB-CGC Google BigQuery tables which are based on the latest GDC data release. So, it's possible that a new GDC release occurred since you last performed that query.

29.2.2 Why do I sometimes get a "Do you want to leave this site?" pop-up box when leaving a page or canceling a workflow edit?

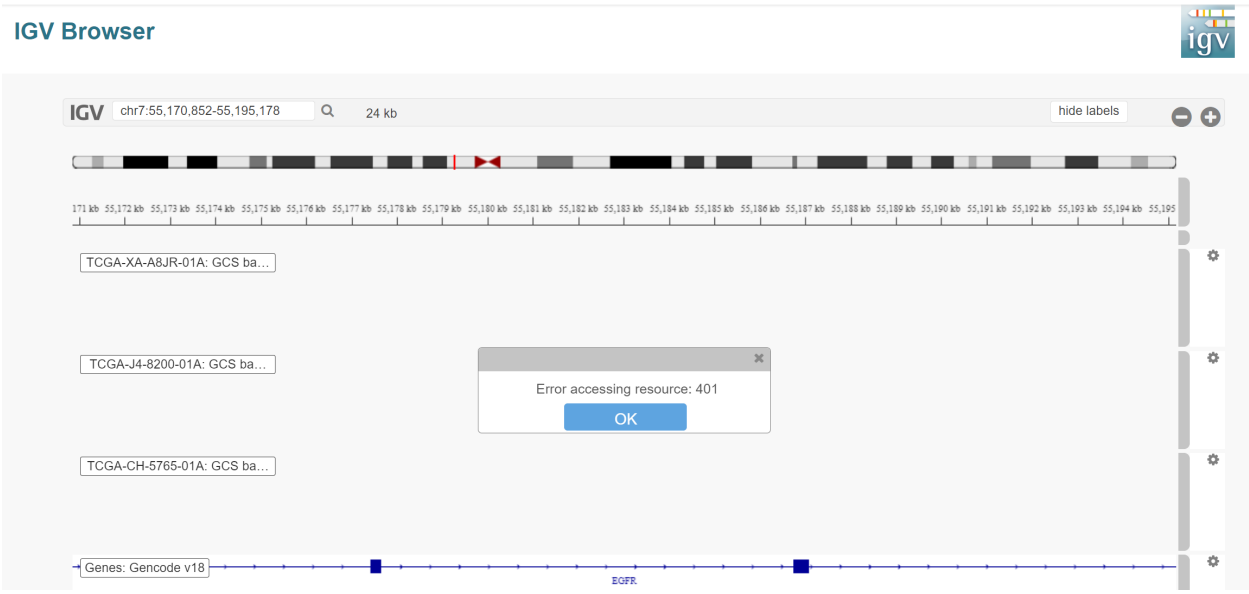
This is a security feature when working with forms found in most web browsers; it lets you know that you may have made some changes which will be lost when you navigate away from the page. If you intend to cancel what you were doing, you can safely ignore it.

29.2.3 Which web browser is recommended when working with the site?

We recommend using Google Chrome browser. Currently a chart will display slightly off when working with workbooks on a FireFox browser.

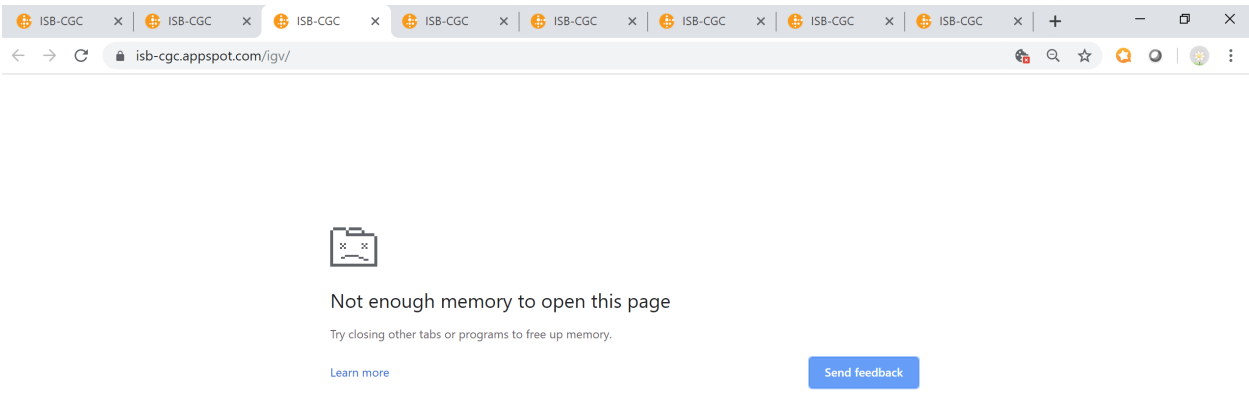
29.2.4 Why did I get a 401 error on the IGV Browser?

You will see the 401 error only if your pop-up blocker is enabled for the ISB-CGC website. Please disable the pop-up blocker on the top right-hand side of the screen by selecting to always allow pop-ups from ISB-CGC.



29.2.5 Why does the web browser crash if too many IGV Browser tabs are opened at once?

The web browser may crash when too many IGV Browser tabs are open due to the memory intensive nature of loading bam files. When working with the IGV Browser, please be mindful of having multiple tabs of the IGV Browser open.



29.2.6 Does SeqPeek and CNVR plotting only work with TCGA data?

We currently have no data associated with CNVR or Seqpeek for TARGET or CCLE. Therefore, SeqPeek and CNVR will only work with TCGA data.

29.3 Data Access

29.3.1 Does all TCGA data require dbGaP authorization prior to access?

No, generally only the low-level sequence (DNA and RNA) and SNP-array data (CEL files) require dbGaP authorization. All of the "high-level" molecular data, as well as the clinical data are open-access and much of this has been made available in a convenient set of BigQuery tables.

29.3.2 Where can I find the TCGA data that ISB-CGC has made publicly available in BigQuery tables?

The BigQuery web interface can be accessed at https://console.cloud.google.com/bigquery. If you have not already added the ISB-CGC data sets to your BigQuery "view", click on the blue arrow next to your project name at the top of the left side-bar, select "Switch to Project", then "Display Project…", and enter "isb-cgc-bq" (without quotes) in the text box labeled "Project ID". For older ISB-CGC data sets, repeat and enter "isb-cgc". All ISB-CGC public BigQuery data sets and tables will now be visible in the left side-bar of the BigQuery web interface. *Note that in order to use BigQuery, you need to be a member of a Google Cloud Project.*

29.3.3 How can I apply for access to low-level DNA and RNA sequence data?

In order to access the TCGA or All other controlled-access data available, you will need to apply to dbGaP. Please also review our section on **Understanding Data Security**.

29.3.4 I have dbGaP authorization. How do I provide this information to the ISB-CGC platform?

In order for us to verify your dbGaP authorization, you first need to associate your Google Identity (used to sign-in to the Web App) with a valid NIH login (*eg* your eRA Commons ID). After you have signed in, click on your avatar (next to your name in the upper-right corner) and you will be taken to your account details page where you can verify your dbGaP authorization. You will be redirected to the NIH iTrust login page and after you successfully authenticate, you will be brought back to the ISB-CGC Web App. After you successfully authenticate, we will verify that you also have dbGaP authorization for the TCGA controlled-access data and other programs you have dbGaP access to.

We also ask that you review our section on Understanding Data Security.

29.3.5 My professor has dbGaP authorization. Do I have to have my own authorization too?

Yes, your professor will need to add you as a "data downloader" to his/her dbGaP application so that you have your own dbGaP authorization associated with your own eRA Commons ID. (This video explains how an authorized user of controlled-access data can assign a downloader role to someone in his/her institution.)

I already authenticated using my eRA Commons ID but now I want to use a different Google identity to access the ISB-CGC Web App. Can I reauthenticate using the same eRA Commons ID?

Yes, but you will first need to sign in using your previous Google identity and "unlink" your eRA Commons ID from that one before you can link it with your new Google Identity. An eRA Commons ID cannot be associated with more than one Google Identity within the ISB-CGC platform at any one time.

29.3.6 Can I authenticate to NIH programmatically?

No, the current NIH authentication flow requires web-based authentication and must therefore be done from within the ISB-CGC Web App. Once you have authenticated to NIH via the Web App, and your dbGaP authorization has been verified, the Google identity associated with your account will have access to the controlled-data for 24 hours.

29.4 Data Content

29.4.1 I get a different number of samples in BigQuery than I do with the same query in the Web App. Why?

Older programs like TCGA have both legacy data (data from the original program) and harmonized data (data run through the Genomics Data Commons). The Web App primarily uses harmonized data whereas BigQuery contains both legacy and harmonized data. In addition, some cases and samples have been removed from the Web App if annotation suggests the data from those cases or samples are incorrect, misleading or from cases of uncertain origin. Most of these cases and samples are still in BigQuery and users are encouraged to check the annotations tables.

29.5 Python Users

29.5.1 I want to write Python scripts that access the TCGA data hosted by the ISB-CGC. Do you have some examples that can get me started?

Yes, of course! The best place to start is with our Community Notebooks or our repository in GitHub. You can run any of these examples yourself. It includes an introduction explaining what Notebooks are, how to get started as a novice user, and how to run more advanced analyses once you are comfortable.

29.6 R Users

29.6.1 I want to use R and Bioconductor packages to work with the TCGA data. How can I do that?

You can run RStudio locally or deploy a dockerized version on a Google Compute Engine VM. You can find some great examples to get you started in our Community Notebooks or our repository in Community Notebooks GitHub.

For an example on how to use Bioconductor packages with TCGA data in BigQuery, please check out our interactive tutorial found here.

29.7 Regulome Explorer Users

29.7.1 Can I run Regulome Explorer Analyses using TCGA tables of heterogeneous data in BigQuery?

Yes, of course! A series of Python Notebooks have been created to replicate Regulome Explorer and includes detailed information on the statistical methods implemented. To get started, please visit our Regulome Explorer page in readthedocs or our Repository in Regulome Explorer GitHub.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

CHAPTER

THIRTY

CONTACT US

For general information about the ISB-CGC please contact us at feedback@isb-cgc.org. We are especially keen on learning about your particular use-cases, and how we can help you take advantage of the latest in cloud-computing technologies to answer your research questions.

For feature requests or bug reports, please send e-mail to feedback@isb-cgc.org.

We have virtual Office Hours on Tuesdays and Thursdays for any questions on ISB-CGC functionality or data that you may have. We look forward to speaking with you.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.

Have feedback or corrections? Please email us at feedback@isb-cgc.org.