Published in final edited form as:

Curr Protoc Bioinformatics. ; 60: 11.16.1–11.16.32. doi:10.1002/cpbi.39.

Using the Seven Bridges Cancer Genomics Cloud to Access and Analyze Petabytes of Cancer Data

Raunaq Malhotra^{1,2}, Isheet Sethi^{1,2}, Erik Lehnert^{1,2}, Jing Zhao¹, Gaurav Kaushik¹, Elizabeth H Williams¹, Anurag Sethi^{1,3}, and Brandi Davis-Dusenbery¹

¹Seven Bridges Genomics, Inc., 1 Main Street, Suite 500, Cambridge, MA 02142

Abstract

Next-generation sequencing has produced petabytes of data, but accessing and analyzing these data remain challenging. Traditionally, researchers investigating public datasets like The Cancer Genome Atlas (TCGA) would download the data to a high-performance cluster, which could take several weeks even with a highly optimized network connection. The National Cancer Institute (NCI) initiated the Cancer Genomics Cloud Pilots program to allow researchers without these resources to process data with cloud computational resources. We present protocols using one of these Cloud Pilots, the Seven Bridges Cancer Genomics Cloud (CGC), to find and query public datasets, bring your own data to the CGC, analyze data using standard or custom workflows, and benchmark tools for accuracy with interactive analysis features. These protocols demonstrate that the CGC is a data analysis ecosystem that fully empowers researchers with a variety of areas of expertise and interests to collaborate in the analysis of petabytes of data.

Keywords

Common Workflow Language; Scalable; Reproducible; Cloud computing; Cancer Genomics; Big Data

Introduction

The advent of next-generation sequencing has accelerated the generation of petabytes of multi-omic information, creating immense research opportunities but also immense challenges. Best-practice methods for securely organizing, sharing, and analyzing large and heterogeneous volumes of data are still being developed. The development of tools that democratize access and analysis of these datasets could have a massive impact on discovery and healthcare in the age of precision medicine.

The field of cancer genomics has been a primary driver of the biological data boom. The Cancer Genome Atlas (TCGA; (Cancer Genome Atlas Research Network et al., 2013) network alone has produced more than 2.5 petabytes of data, and the International Cancer Genome Consortium (ICGC; <http://icgc.org/>) and Pancancer Analysis of Whole Genomes

³Corresponding Author: anurag.sethi@sevenbridges.com.

²Equal Contribution

(PCAWG) study (Yung et al., 2017) also have generated large volumes of cancer data. The creation of these datasets has revealed common challenges in downstream processing and collaborative development. Downloading a complete dataset to local storage can take weeks or more; integrated analysis requires resources available only to a limited number of researchers with access to large institutional compute clusters; and reproducible, collaborative research between institutions is inhibited by difficulties associated with accessing shared infrastructure. With the aim of democratizing access to TCGA and other cancer genomic data, the National Cancer Institute launched the Cancer Genomics Cloud Pilots program in 2015 to explore colocalizing data and the computational resources necessary for its analysis in the cloud for use by researchers globally. Seven Bridges Genomics was one of three institutions that participated in the CGC Pilots program.

The Seven Bridges Cancer Genomics Cloud (CGC; www.cancer-genomics-cloud.org; Lau et al., in press) is a customizable and scalable data access and analysis platform that connects researchers to extensive public datasets, including TCGA, Therapeutically Applicable Research to Generate Effective Treatments (TARGET; <https://ocg.cancer.gov/programs/target>), the Simons Genome Diversity Project (Mallick et al., 2016), and the Cancer Cell Line Encyclopedia (Barretina et al., 2012). The CGC enables collaborative, reproducible analysis across both public and private cohorts. To reduce the unnecessary burden of installing bioinformatics software, the CGC provides a public toolkit containing more than 250 commonly used analytical and bioinformatics tools and workflows. These are described in the Common Workflow Language (Amstutz et al., 2016), an emerging standard for describing computational tools and workflows. Additional resources, including the open-source *CGC Software Development Kit* and *Data Cruncher*, facilitate custom tool prototyping and deployment. Since its launch in early 2016, more than 1800 researchers from more than 150 institutions in 38 countries have used the CGC to perform more than 100,000 analyses, representing almost 100 years of computation time.

The CGC creates secure collaborative environment in which researchers can use and reuse public multi-omic data in combination with their own private data to perform reproducible analyses utilizing scalable cloud computing resources. In addition, the CGC provides an extendable framework by giving researchers the ability to port their tools and workflows as well as write custom application programming interface (API) scripts to automate their analyses.

Here, we discuss five specific challenges that we identified in working with biomedical data at scale and then demonstrate the CGC tools that were developed to address those challenges. Our solution to these five challenges is described here in individual protocols for (i) performing metadata queries across large datasets; (ii) scaling analysis on the cloud; (iii) benchmarking of tools to promulgate community approved best practices; (iv) automation of analysis with private data using an API; and (v) creating Dockerized tools and deploying them for reproducibility on the cloud.

Basic Protocol 1: Accessing the TCGA data on the CGC platform

Datasets can be described using a structured metadata model that consists of entities and properties. Entities represent particular resources with UUIDs, such as files, cases, samples, and cell lines, and properties are values that describe a given entity. To facilitate access to publicly available datasets (including TCGA, TARGET, and CCLE), we created a visual interface called the **Data Browser** that enables interactive queries against a subset of the metadata each dataset. The original TCGA dataset contained ~700 biospecimen and ~2000 clinical metadata fields, which could differ between different sequencing centers. In addition, a large proportion of these metadata attributes were highly specific to cancer type or field. To promote dataset interrogation, we reduced the number of queryable metadata fields without compromising on usability. In particular, we found that the vast majority of the 2000 TCGA metadata fields were sparsely populated (within or between disease types) or redundant and from overall set of metadata identified the fields most often associated with values (i.e., not left blank). We excluded the sparsely populated fields, thereby reducing the set to the 106 most frequently used attributes organized into 10 entities. Based on the hierarchical metadata structure in Figure 1, we built a resource description framework (RDF) based ontology to organize each major entity in the hierarchy (e.g. 'Case', 'Sample') and enable graph-based semantic search of the public cancer datasets on the CGC. Each entity also has a set of properties that can be used to filter the dataset further (Figure 1).

This protocol details how to find and access Controlled TCGA data on the CGC using the **Data Browser**. Researchers can build their own queries from scratch using the metadata associated with each dataset. As an example, the protocol walks through how to construct a query that will return a list of files containing the raw RNA sequencing reads for Breast Invasive Carcinoma (BRCA) cases having tumor-normal paired samples. These files can be copied directly into a project on the CGC for analysis with a custom or a publicly available RNA-Seq workflow. In Alternate Protocol 1, we show how to programmatically query and copy these files to a project using the Datasets Application Programming Interface (API).

Necessary Requirements

- A computer with internet access and an up-to-date internet browser (e.g. Firefox, Chrome, Safari).
- An account on the Seven Bridges' CGC (<https://cgc.sbgenomics.com>). All researchers can query the metadata. However, you must have permission to access Controlled Data through the Database of Genotypes and Phenotypes (dbGaP) through your eRA Commons or NIH Center For Information Technology (CIT) account (Link: <https://dbgap.ncbi.nlm.nih.gov/aa/wga.cgi?page=login>). **Note:** These protocols were developed prior to the addition of Open Access raw sequencing data (e.g., CCLE data) to the Data Browser. Researchers who do not have access to Controlled TCGA Data can follow along with minor modifications using either their own data or other Open Access raw sequencing data.

1. Visit the Seven Bridges' CGC using your browser and login to query the public datasets. However, to access and copy Controlled Access Data from TCGA, you need to be registered with your eRA Commons credentials and have permission to access Controlled Data through dbGaP (Figure 2).
2. Once logged in, access the **Data Browser** from the top navigation bar on the landing page. Click on the "Data" tab, and then click **Data Browser**.
3. The first **Data Browser** page lists the multiple datasets available for querying. Select the legacy TCGA dataset (Figure 3) and click on the "Explore selected" button.
4. Selecting the legacy TCGA dataset will take you to the new query page, which lists all the available metadata entities. You can select any one of the entities to browse the dataset.
5. In order to query for tumor-normal matched samples with RNA-Seq raw reads from Breast Invasive Carcinoma patients, you need to identify the same cases (i.e., patients) with both tumor and normal RNA-seq data. For this:
 - a. Select the **Case** entity on the Data Browser, and click on the "+ Add property" button.
 - b. In the search bar below, type "Breast Invasive Carcinoma" and select the "Disease type: Breast Invasive Carcinoma" option that appears below. Alternatively, you can first add the "Disease type" sub-entity under the **Case** entity and then search for "Breast Invasive Carcinoma". After selecting the "Disease type" under the **Case** entity, the number of available cases is 1,101, as can be seen on the bottom of the screen (Figure 4).
 - c. In order to select tumor samples for Breast Invasive Carcinoma disease cases, next add the **Sample** entity from the available options connected to the current **Case** entity.
 - d. Click on "+ Add property" under the **Sample** entity and search for "Primary Tumor". Select the option **Sample type: Primary Tumor** that appears below. As an alternate, you can select the sub-entity **Sample type** and then select the "Primary Tumor" property in it.
 - e. In order to select matched normal samples for the tumor samples selected in steps c-d, click on the **Case** entity again and add another **Sample** entity to the Data Browser query.
 - f. Click on "+ Add property" under the **Sample 1** entity and then select the **Sample type** sub-entity. Select the "Solid Tissue Normal" property in it. This allows you to query for only those cases where both "Primary Tumor" and a "Solid Tissue Normal" samples are available. **Note:** Forks in queries are treated as logical AND, whereas listing multiple values in a single property is treated as logical OR.

- g. The next step is to query for the RNA-Seq files for both primary tumor and normal samples. Add the **File** entity to the **Sample** entity and search for RNA-Seq by adding a property under it. Select the **Experimental strategy: RNA-Seq** sub-entity:property combination. In order to look for raw sequencing files, add another property under the **File** entity and search for “unaligned reads”. Select the **Data subtype: Unaligned reads** entity:property pair.
 - h. Similar to step g, add the RNA-Seq unaligned read files for the “Solid Tissue Normal” samples to build the final query (Figure 5).
6. Once the query is built, click on one of the purple File boxes. This will highlight the line connecting the queries leading to it, indicating that these files have been selected. Click the green **Copy files to project** button at the top right of the screen to export the primary tumor files to a Controlled Data project. You can, at this point, add tags to the files to indicate specific sets of files in the project. For example, you may want to tag these files as **Primary Tumor**. After these files have been added to your project, select the second File box and repeat the process for the normal files.

Support Protocol 1: Creating a login with/without Controlled Data access on the CGC platform

1. Visit <https://cgc.sbgenomics.com> (Figure 6).
2. If you don't have an eRA Commons or NIH CIT (Center for Information Technology) account, click **Create a free account** and register via your email (Figure 7).
3. If you have an eRA Commons account, you also can register via that account (Figure 8).

Support Protocol 2: Create a Project and Collaborate via the Graphical User Interface (GUI)

Before you start working on the CGC, you need to create a project. We recommend that each project correspond to a specific scientific investigation and that it contain data, analysis workflows, and results for that particular investigation. There are two types of projects on the CGC: Open and Controlled Data projects. Controlled Data projects are only accessible to users with dbGaP access permissions to Controlled Data. The following steps allow you to create a project, add a collaborator, and add files and apps:

To create a Controlled Data project via the GUI

1. Click the Projects tab on the top navigation bar and select “+Create a project” from the drop-down menu.
2. Name the project.
3. Choose your project type: If you have dbGaP access permissions for TCGA Controlled Data the checkbox next to “This project will contain Controlled Data.” will be automatically checked.

4. Select the billing group. The billing group is the set of funds to which costs for storage of files and analysis for that project will be charged. Keep this in mind when setting permissions, because collaborators' tasks will be charged to the billing group set when the project was created.

Note: Public reference files are not charged, but files created during analyses are. For example, you do not accrue storage costs for a publicly available TCGA BAM file in your project. However, if you process the BAM to create a VCF, you will be charged for the analysis and storage costs for the newly created VCF file.

5. Click "Create" to complete project creation.

Add a collaborator to the project

1. Navigate to the desired project.
2. Click "Invite new members" (or "Manage project members" if there are already members in the project) on the dashboard of the chosen project.
3. Type in the collaborator's CGC username or email.
4. You can then set permissions for each collaborator with the permissions drop-down menu next to the collaborator's username. Select the checkboxes besides the desired permission(s) you want your collaborator to have (e.g., Write, Copy, Execute, and Admin). By default the permission is set to Read. In order to run a task within a project, the collaborator will need all four permissions "Read, Copy, Write, and Execute".

Figures 9 and 10 show screenshots of new projects created with and without Controlled Data access, respectively, with added collaborators each with a different permission level.

Alternate Protocol 1: Accessing Public Cancer Data Using the Datasets API

The Datasets API enables researchers to search for files in the TCGA (legacy hg19) dataset based on metadata. Similar endpoints are available for querying other public datasets such as CCLE. All researchers can perform queries to return Case, Sample, and File IDs and import level 3 files (i.e., files containing non-personally identifiable information), such as gene quantification or somatic VCF files. Files containing personally identifiable data, such as germline VCF files and raw sequencing data, can only be accessed by researchers with appropriate dbGaP access permissions. In this section, we will show how to write a script using the Python bindings for the Datasets API to search for and import into projects RNA-sequencing data from matched tumor-normal samples from patients diagnosed with BRCA.

Necessary Requirements (in addition to the requirements from Basic Protocol 1)

- Install conda and the Python bindings for the Seven Bridges API using:
 - `pip install sevenbridges-python`

The jupyter notebook corresponding to Alternate Protocol 1 is available on github at <https://github.com/sbg-cancer-cloud/Current-Protocols-in-Bioinformatics-2017>.

1. The Datasets API employs API requests written in JSON format to return entities such as Case, Sample, or File IDs. We will use two Python modules, `json` and `requests`, to write a wrapper around the API request, so the first step is to import these modules. We will then define a simple function to send and receive JSONs from the API using correctly HTTP calls. The token is your Authentication token. We show you how to get that in the next step.

```
import json
from requests import request

def api_call(path, method='GET', query=None, data=None,
token=None):
    # Base URL for datasets API
    base_url = 'https://cgc-datasets-api.sbgenomics.com/datasets/
tcga/v0/'
    # input for API call converted to json format
    data = json.dumps(data) if isinstance(data, dict) \
or isinstance(data, list) else None
    # header for API call
    headers = {
        'X-SBG-Auth-Token': token,
        'Accept': 'application/json',
        'Content-type': 'application/json',
    }
    # API call
    response = request(method, base_url + path, params=query, \
                        data=data, headers=headers)
    #Converting response from JSON to dictionary
    response_dict = response.json() if \
response.json() else {}
    if response.status_code / 100 != 2:
        print(response_dict['message'])
        print('Error Code: %i.' % (response_dict['code']))
        print(response_dict['more_info'])
        raise Exception('Server responded with status code %s.' \
                        % response.status_code)
    return response_dict
```

2. Obtain your CGC Authentication token for the API from the **Developer Dashboard** within your account. To access the Developer Dashboard, click on your username in the upper right corner and then click **Developer**. Click on **Auth token** to access the Authentication token (Figure 11).

***Note:** Your Auth token allows access to the CGC and your projects, so treat it as you would your password.*


```
auth_token = "Your Authentication Token Here"
```

3. Build a query in JSON format to find primary tumor samples that (i) are from Cases (i.e., patients) diagnosed with Breast Invasive Carcinoma that (ii) have associated RNA-Seq reads. You can query each entity's schema using an API call such as <http://docs.cancergenomicscloud.org/docs/query-via-the-datasets-api#section-step-1-get-an-entity-s-metadata-schema>.

```
tumor_samples_query = {
    "entity": "samples",
    "hasSampleType": "Primary Tumor",
    "hasCase": {
        "hasDiseaseType": "Breast Invasive Carcinoma",
    },
    "hasFile": {
        "hasExperimentalStrategy": "RNA-Seq",
        "hasDataSubtype": "Unaligned reads"
    }
}
```

4. Perform an API query using the function from step 2 and the query from step 3 to return the number of BRCA primary tumor samples with RNA-seq read files. Adding “query/total” to the base TCGA metadata path ensures that you get the number of samples that match the above metadata query.

```
total = api_call(method='POST', path='query/total', \
                 token=auth_token, data=tumor_samples_query)
print("There are {} samples matching the
query".format(total['total']))
```

5. Define a function to get all matches to the query using the corresponding HTTP calls. The previous API call gave the total number of samples related to the query (primary tumor samples from BRCA cases that have raw RNA-seq reads). If you want a list of IDs and all the metadata associated with the samples that match this query, we need to add “query” to the path and get the results for all samples matching the metadata query. In addition, the results are returned in the form of pages with a maximum of 100 entities per page. Hence, we have to loop through the results to get the final list of sample IDs and associated metadata. We can then use this new function to return all the sample IDs that match the tumor query

```
import math
def getAllMatches(auth_token, query_body):
```



```

numberFiles = api_call(method="POST", path="query/total",
                        token=auth_token,
                        data=query_body["total"])
numCalls = int(math.ceil(numberFiles/100.0))
matches = []
entity = query_body["entity"]
for i in range(0, numCalls):
    query_body["offset"] = str(i * 100)
    currSet = api_call(method="POST", path="query",
                      token=auth_token,
                      data=query_body["_embedded"][entity])
    for currMatch in currSet:
        matches.append(currMatch)
return matches
tumor_samples = getAllMatches(auth_token, tumor_samples_query)
tumor_sample_ids = [curr_sample["id"] for curr_sample in
                    tumor_samples]

```

6. Use this function to also identify all the tissue normal samples.

```

tissue_normal_samples_query = {
    "entity": "samples",
    "hasSampleType": "Solid Tissue Normal",
    "hasCase": {
        "hasDiseaseType": "Breast Invasive Carcinoma"
    },
    "hasFile": {
        "hasExperimentalStrategy": "RNA-Seq",
        "hasDataSubtype": "Unaligned reads"
    }
}
tissue_normal_samples = getAllMatches(auth_token,
                                     Tissue_normal_samples_query
                                     )
tissue_normal_sample_ids = [curr_sample["id"] for curr_sample in
                           tissue_normal_samples]
total = api_call(method='POST', path='query/total',
                 token=auth_token,
                 data=tissue_normal_samples_query)
print("There are {} samples matching the
      query".format(total['total']))

```

7. To identify the subset of patients with RNA-seq experiments for both primary tumor and tissue normal, take the intersection of the two lists.

```
tumor_cases_query = {
    "entity": "cases",
    "hasSample": tumor_sample_ids
}
tumor_cases = getAllMatches(auth_token, tumor_cases_query)
tumor_case_ids = [curr_case["id"] for curr_case in tumor_cases]
tissue_normal_cases_query = {
    "entity": "cases",
    "hasSample": tissue_normal_sample_ids
}
tissue_normal_cases = getAllMatches(auth_token,
                                     tissue_normal_cases_query)
tissue_normal_case_ids = [curr_case["id"] for curr_case in
                          tissue_normal_cases]
tumor_match_case_ids = list(set(tumor_case_ids) &
                             set(tissue_normal_case_ids))
print("There are {} cases that have both primary tumor and
      solid tissue normal samples with RNA-seq
      experiments".format(len(tumor_match_case_ids)))
```

8. Obtain the paired tumor-normal samples by querying using the Case IDs and appropriate metadata for the primary tumor and solid tissue normal samples, respectively. This query will return a list of json objects corresponding to the metadata for each file.

```
tumor_match_files_query = {
    "entity": "files",
    "hasExperimentalStrategy": "RNA-Seq",
    "hasDataSubtype" : "Unaligned reads",
    "hasSample": {
        "hasSampleType" : "Primary Tumor"
    },
    "hasCase": tumor_match_case_ids
}
tumor_match_files = getAllMatches(auth_token,
                                   tumor_match_files_query)
tissue_normal_match_files_query = {
    "entity": "files",
    "hasExperimentalStrategy": "RNA-Seq",
    "hasDataSubtype" : "Unaligned reads",
    "hasSample": {
        "hasSampleType" : "Solid Tissue Normal"
    },
    "hasCase": tumor_match_case_ids
}
```

```

        "hasCase": tumor_match_case_ids
    }
    tissue_normal_match_files = getAllMatches(auth_token,
                                              tissue_normal_match_files_query)
    print("There are {} files corresponding to raw reads for
          tumor samples in tumor-normal matched cases for
          BRCA".format(len(tumor_match_files)))
    print("There are {} files corresponding to raw reads for solid
          tissue normal samples in tumor-normal matched cases for
          BRCA".format(len(tissue_normal_match_files)))

```

9. The File objects associate information with the File ID (a unique identifier for a file on the CGC), and this ID can be used to copy the file to your project. In the case of Controlled Data, you must have appropriate dbGaP access permissions to copy these files or view their contents. Now, initialize the Seven Bridges python bindings.

```

import sevenbridges as sbg
api = sbg.Api(url='https://cgc-api.sbgenomics.com/v2',
              token=auth_token)

```

10. If you haven't already made a project for these files using the GUI, create one using the API. To create a new project, provide a project name and a billing group. This code block selects the first billing group available, which in most cases will return a user's personal billing group. If you are involved in collaborations and are in multiple billing groups, you will want to select the appropriate one for each project.

```

new_project_name = 'Protocol 1'
billing_groups = api.billing_groups.query()
print((billing_groups[0].name + \
       ' will be charged for computation and storage (if
       applicable) for your new project'))
new_project = {
    'billing_group': billing_groups[0].id,
    'name': new_project_name,
    'tags': ['tcga']
}
my_project = api.projects.create(name = new_project['name'], \
                                billing_group = new_project['billing_group'], \
                                tags = new_project['tags'])
my_project = [p for p in api.projects.query(limit=100).all() \
              if p.name == new_project_name][0]

```

11. Copy the files to the project using the ID of each file returned from the Datasets API. To do this, create a function that loops through the list of files you want to copy and copies each file to the project.

```
def copy_to_project(api, my_project, final_files):
    new_files = []
    for curr_file in final_files:
        file_object = api.files.get(id = curr_file['id'])
        my_new_file = file_object.copy(project = my_project.id,\
                                       name = file_object.name)
        new_files.append(my_new_file)
    print "Files Imported!"
copy_to_project(api, my_project, tumor_match_files)
```

Basic Protocol 2: Scaling analysis on the CGC platform with hundreds of datasets

Prior to the creation of the three CGC Pilot platforms, the analysis of petabyte-scale datasets required large computing infrastructure that was not available to many researchers. Co-localization of data and cloud computing resources enables a researcher to easily analyze hundreds of datasets without the need to download a large number of multi-GB files. The CGC makes it possible for researchers to analyze data in an efficient manner through the Graphical User Interface (GUI) and the Application Programming Interface (API). Using the API, researchers can automate their analysis from start to finish.

Users can perform an analysis using an app from the CGC Public Apps gallery or by porting their own tool(s) to the platform (Basic Protocol 5). Apps can be either individual tools or workflows (i.e., tools that have been chained together to sequentially process data). As an example of scalable analysis on the CGC platform, we show here how you can perform a gene fusion detection analysis on Breast Invasive Carcinoma samples selected with the Data Browser query performed in Basic Protocol 1. We use three different fusion callers and compare the results from these fusion callers in Figures 12–14.

Necessary Requirements

- You must have completed Basic Protocol 1 and Support Protocols 1 and 2 before you start working on this protocol.
- Python (version 2.7 or later)
- GRCh38_gencode_v24_CTAT_lib_Mar292017_prebuilt.tar.gz (available at https://data.broadinstitute.org/Trinity/CTAT_RESOURCE_LIB/_pre_July202017/) - This package consists of the GRCh38 reference human genome and corresponding Gencode annotation set for your project.

The CWL for the tools used in Basic Protocol 2 are available on github at <https://github.com/sbg-cancer-cloud/Current-Protocols-in-Bioinformatics-2017>.

1. Once the STAR-Fusion tool (Haas & Dobin, 2015) has been wrapped and the workflow is built (see Support Protocol 2), add the optimized STAR-Fusion workflow to the project. To add the workflow to your project, click on the “Add app” button, then go to “My Apps” and search for the STAR-Fusion-workflow app. Once you find the app you want, then click on the software fork symbol to copy the workflow to your project.
2. Import the sequencing reads to your project. For example, use the query from Basic Protocol 1 to import the 126 primary tumor TAR.GZ RNA-Seq files from Breast Invasive Carcinoma samples from the TCGA dataset.
3. Import the reference file(s) to your project. For example, for the STAR-Fusion workflow, upload the GRCh38 CTAT resource library file using the FTP/HTTP(S) import tool on the platform. To upload file(s) using the FTP/HTTP(S) import tool after clicking on “+Add Files”, click the FTP or HTTP(S) server, and then type in or paste the URL of the file(s) you want to import.
4. Once you have all the 126 TAR.GZ files and the STAR-Fusion workflow app in your project, use the API to automate processing of samples through the workflow within the project. For this analysis, API scripts written in Python were used to automate creation and running of 126 tasks simultaneously. The corresponding script that was used for task automation is available on github at <https://github.com/sbg-cancer-cloud/Current-Protocols-in-Bioinformatics-2017>

Note: Users can also perform Batch analysis by using the Batch Input feature in the workflow. Multiple datasets can be processed with the same workflow with identical parameters without the need to recreate the workflow for every dataset. You can provide multiple inputs and group files by a specific metadata. When you execute the workflow with batch settings, the result is a batch task that consists of several sub-tasks (one for each group of files).
5. Several output files were generated after the 126 tasks completed successfully. Obtain the final list of gene fusions from STAR-Fusion by downloading the output file with the extension “final.abridged” for each of the 126 samples.
6. To determine the total cost and time spent on processing each of the 126 samples in the project, use the API to query the compute time and cost for each task within the project.

```
my_project = api.projects.get(id = '<user-id>/<project-label>')
tasks = [t for t in api.tasks.query(project =
my_project, limit=100).all()]
cost = 0;
for t in tasks:
    if t.status == "COMPLETED" and t.name.startswith("star"):
        if t.price is not None:
            time_total = t.end_time - t.start_time
```

```

        print("Task name (%s) cost : %f, time (%s),
Instance ID(%s)" %
(t.name,t.price.amount,time_total,

t.get_execution_details().jobs[0].instance.id))
        cost = cost + t.price.amount
print("Total cost for completed tasks in this project is %f" %
(cost))

```

The time and cost of running STAR-Fusion on BRCA RNA-Seq samples are summarized in Table 1 and Figure 11.

7. Repeat the procedure described above in steps 1 through 6 twice to process the same samples using FusionCatcher (Nicorici et al., 2014) and EricScript (Benelli et al., 2012).

Note: In Table 1, we compare the three fusion callers based on the total time, cost, CPU cores and AWS instance used to process 121 paired-end RNA-Seq BRCA samples. The choice of instance type was based on the optimal combination of execution time and cost to run one sample. For example, an *r3.8xlarge* AWS instance was selected to run STAR-Fusion based on RAM and disk space usage of this workflow. All tasks were run on Linux Dedicated On-Demand Instances to ensure a fair cost comparison - prices would be lower if run on Spot instances, for example. Figure 12 shows the total time and cost spent for each of the three workflows. For our analysis of 121 paired-end RNA-Seq samples, gene fusion detection by EricScript was the least expensive, and STAR-Fusion was the fastest. FusionCatcher was the most expensive and took the maximum time to process these samples. You will notice that time and cost are not necessarily always correlated. This is because the cost to run a workflow is dependent on the AWS instance type you select.

We also compared the total number of gene fusion pairs detected by each of the three fusion callers. Figure 13 is a Venn Diagram that shows the total number of unique gene fusion pairs detected by each tool as well as the overlapping gene fusion pairs detected between each of the three tools and between each combination of two tools. All three tools were able to detect the same 270 gene fusion pairs. Out of the 270 overlapping gene fusion pairs detected, Figure 14 shows the gene fusion pairs that had the highest frequency of occurrence (top 10) detected by the any of the three tools within the population of 121 samples. Three gene fusion pairs (indicated with an asterisks) were detected with high frequency (top 10) by all three tools. There is evidence in the literature that some of these fusion pairs are known to be the recurrent gene fusions detected in Breast Invasive Carcinoma. For example: *ESR/CCDC170* and *EXTL3/SARAF* are known recurrent gene fusion pairs detected in breast invasive carcinoma (Veeraraghavan, Ma, Hu, & Wang, 2016; Yoshihara et al., 2015).

Support Protocol 3: Using the Workflow Editor to run STAR-Fusion with TCGA files

A workflow is a chain of tools connected together. Within a project, the Workflow Editor can be used to create a workflow from scratch. To create a workflow:

1. Navigate to the project dashboard, select the Apps tab, and click on the “+Add App” button.
2. Click “Create new app” and then click “Create a workflow”.
3. Provide a name for your workflow. For example, we named our workflow “star-fusion-workflow”. Once the workflow has been named, you will be taken to the Workflow Editor.
4. From the right-hand panel on the Workflow Editor, search for and select the tools that you want to include as nodes in the workflow. Once selected, drag and drop the tools to the workflow canvas.

***Note:** For Basic Protocol 2, STAR-Fusion was run using the star-fusion-workflow that consisted of three different tools: SBG Unpack FASTQs, SBG Group files by Metadata, and star-fusion-d2. Details on wrapping a tool on the CGC platform are described in detail in Basic Protocol 5. Software STAR fusion 0.8.0 was wrapped in Docker and described in the Seven Bridges Tool Editor according to the usage documented at <https://github.com/STAR-Fusion/STAR-Fusion/wiki>. The input archived paired-end fastq files (TAR.GZ files) were first extracted using SBG Unpack FASTQs and then grouped according to sample information in the metadata using SBG Group files by Metadata. Next, fusion information was called by STAR fusion 0.8.0 with the reference file in TAR.GZ format. STAR-Fusion was “scattered” so that the input files from different samples could be executed in parallel in the same instance. The workflow was run on AWS instance r3.8xlarge, a choice that took into account the RAM and disk space usage of this workflow (Figure 15).*

Basic Protocol 3: Interactive analysis of results using Data Cruncher

Gene fusions are critical markers and drivers of a variety of cancer types, and their identification can facilitate precise diagnostics, tumor classification, and patient stratification. Moreover, recent efforts have demonstrated that genomic rearrangements can act as tumor-specific targets for molecular therapies. However, detecting gene fusions from next-generation sequencing data is an ongoing challenge, and recent benchmarking efforts have shown that tools often provide discordant results, with no universal best practices available. We will use gene fusion detection as a real-life example of how to perform interactive analysis, such as benchmarking for accuracy, on the CGC. The CGC is well suited to addressing such a problem, because multiple algorithms and datasets are easily available and thus each gene fusion caller can be rapidly and directly benchmarked against others. For this example, we will demonstrate how to compare the fusion callers EricScript, FusionCatcher, and STAR-Fusion on the CGC, based on the example in (Kumar, Vo, Qin, & Li, 2016).

Necessary Requirements (in addition to the requirements of Basic Protocol 1 and Alternate Protocol 1)

- Synthetic positive dataset (**Note:** We used the FusionMap dataset (Ge et al., 2011).
 1. Upload the synthetic positive datasets to a CGC project using the CGC Uploader. The installers and instructions for use can be found at: <http://docs.cancergenomicscloud.org/docs/upload-via-the-cgc-uploader>.

Note: You can also upload files using the API or an HTTP/FTP link if it is available.
 2. Set the paired-end metadata for each file.
 - a. Once the files have been uploaded, use the filters to show only files with file extension: **FQ** and containing the string “_1” in the name.
 - b. Check the box to the left of the file name and click **Edit Metadata**.
 3. Set “Paired-end” to 1. This will update the paired-end metadata for the selected files, indicating that these are the first reads of a pair. Repeat this process by searching for “_2” and set “Paired-end” to 2.

***Note:** Metadata can also be set using the API or a manifest file provided to the CGC Uploader, which is often more convenient at scale. The exact filtration terms might differ based on your library names. You may need to filter on “_1” and “_2” or “/1” and “/2”, for example. Most aligners on the CGC require that paired-end metadata to be set in order to use paired reads properly, so always ensure that you set this metadata field. (Figure 16)*
 4. Add the apps STAR-Fusion, EricScript, and FusionCatcher to your project.
 5. Analyze the synthetic positive dataset using each App. You can create Tasks by following steps 3–6 in Basic Protocol 2.

Note: The result files generated for each sample and each fusion caller are available in the Files tab of the project. The files suffixed with “fusion_candidates.final” for STAR-Fusion, “results.filtered.tsv” for EricScript, and “candidate-fusion-genes.fusioncatcher.txt” for FusionCatcher will be used for benchmarking the positive and mixed datasets using Data Cruncher.
 6. To start a Data Cruncher instance inside a project:
 - a. Click on the “Interactive Analysis” tab inside your project and click “Open” on the Data Cruncher panel.

- b. Click the “Create new Analysis” button on the top right of the page to start a new Data Cruncher instance.
- c. After specifying the name of the analysis (for example, Benchmarking), the compute requirements (for example, the default c3.8xlarge instance with 160GB hard drive space, 8 vCPUs, and 15GB RAM), and the rules for file saving (default options of not saving files with log or zip extension), click the “Start the analysis” button to start your instance.

Note: It may take a few minutes for AWS to provision an instance.

- d. Once the instance has started, open the instance by clicking the “Open in Editor” button on the top right of the page.

Note: Data Cruncher supports Python and R kernels in which one can execute tertiary analysis on the files generated inside the project.

A Python notebook was created in Data Cruncher to benchmark the three fusion callers on the positive dataset, and the results of this benchmarking are presented in Table 2. **Note:** This is performed as an example and should not be extrapolated beyond this single synthetic dataset.

Basic Protocol 4: Analyzing private data on the CGC

An emerging challenge in bioinformatics is to make public multi-omic datasets interoperable with one another as well as with researchers’ private datasets. In this protocol, we demonstrate how CGC users can analyze their private data using the public somatic variant calling VarScan 2 workflow. This same workflow can be used to process public datasets (similar to Basic Protocol 2) on the CGC for comparison of the somatic variants called for the public datasets to the variants identified from private data.

The CGC provides multiple methods for uses to upload private data onto the platform including a desktop uploader, an uploader that access an FTP/HTTP server, a command line uploader, and a volumes API that lets users access files on their own AWS S3 bucket from within the CGC (Figure 17). In this protocol, we show how a user can import a tumor-normal matched whole-exome sequencing (WXS) dataset from the Texas Cancer Biobank (Becnel et al., 2016) to the CGC using the volumes API, run a public workflow (VarScan 2; Koboldt et al., 2012) to identify somatic variants, and export the output files to the user’s AWS S3 bucket.

Necessary Requirements (in addition to the resources necessary for Additional Protocol 1)

- Personal AWS bucket with raw sequencing data

The jupyter notebook corresponding to Basic Protocol 4 is available on github at <https://github.com/sbg-cancer-cloud/Current-Protocols-in-Bioinformatics-2017>.

- 1 Use the Seven Bridges AWS S3 policy generator that generates the IAM policy suitable for use with our volumes API. Detailed instructions are available in the CGC Knowledge Center at <http://docs.cancer-genomics-cloud.org/docs/aws-cloud-storage-tutorial>. This policy will have to be applied on the AWS console so that the platform provides you access with the correct permissions.
- 2 Obtain your CGC authentication token by entering the Developer Panel within your CGC account (Figure 11 and Step 3 of Alternate Protocol 1).

```
import sevenbridges as sbg
api = sbg.Api(url='https://cgc-api.sbgenomics.com/v2', token='MY
AUTH TOKEN')
```

- 3 The API uses volumes that behave similarly to mounted hard drives within a laptop, and volumes can be read only or read-write. To create the necessary volumes, provide your AWS authentication code once the policy generator has been applied to your AWS console.

```
volume_import = api.volumes.create_s3_volume(
    name='my_import_volume',
    bucket='BUCKET-NAME',
    access_key_id='INSERT ACCESS KEY ID',
    secret_access_key = 'INSERT SECRET ACCESS KEY',
    access_mode='RO')
```

- 4 Once the volumes have been created, import your files from the AWS bucket to the CGC. (For this, you will need to have created a project on the CGC first as shown in step 13 of Alternate Protocol 1.) In this case, import the files based on their location within the bucket.

```
#List of file names in the bucket that you want to analyze
file_list = ['TCRBOA1-T-WEX.bam',
             'TCRBOA1-N-WEX.bam']
# Loop through selected files, start one job for each.
for f_name in file_list:
    import_job = api.imports.submit_import(volume=volume_import,
                                           project=my_project,
                                           location=f_name)

    print("File {} is in state {} \n"
          .format(f_name, import_job.reload().state))
```

- 5 The VarScan 2 workflow can analyze a tumor-normal bam file for somatic variants and is available within the CGC as a Public App. Copy VarScan 2 to this project.

```

app_name = 'VarScan2 Workflow from BAM'
public_app = [a for a in
    api.apps.query(visibility='public', limit=100).all()
    if a.name == app_name][0]
new_app = public_app.copy(project = my_project.id)
my_app = [a for a in api.apps.query(project = my_project.id,
    limit=100).all()
    if a.name == app_name][0]

```

- 7 VarScan 2 requires a reference file in addition to the tumor-normal bam file. Add the hg19 reference file within the CGC Public Files collection as the reference file for this task. **Note:** The CGC public reference file collection contains a number of typically used reference files and some example files for user testing purposes.

```

f_name = 'human_glk_v37_decoy.fasta'
source_project_id = 'admin/sbg-public-data'
source_file = [f for f in api.files.query(limit = 100,
    project = source_project_id).all()
    if f.name == f_name][0]
my_new_file = source_file.copy(project = my_project.id,
    name = source_file.name)

```

- 8 Start a task with the appropriate inputs. The input ports for these files are Tumor_BAM, Normal_BAM, and input_fasta_file. These input ports have to be set to the corresponding file in this project before the task can be started.

```

inputs = {}
inputs['input_fasta_file'] = my_new_file
all_files = list(api.files.query(project=my_project.id,
    limit=100).all())
tumor_bam_file = [curr_file for curr_file in all_files
    if curr_file.name == 'TCRBOA1-T-WEX.bam'][0]
normal_bam_file = [curr_file for curr_file in all_files
    if curr_file.name == 'TCRBOA1-N-WEX.bam'][0]
inputs['Tumor_BAM'] = tumor_bam_file
inputs['Normal_BAM'] = normal_bam_file
task_name = 'VarScan2 with volumes API'
my_task = api.tasks.create(name=task_name,
    project=my_project.id,
    app=my_app.id,
    inputs=inputs,
    run=True)

```

- 9** Use the API to check the status of the task every 30 seconds. As soon as the task completes, the API will come out of the loop.

```

details = my_task.get_execution_details()
print('Your task is in %s status' % (details.status))
loop_time = 30
flag = {'taskRunning': True}
while flag['taskRunning']:
    details = my_task.get_execution_details()
    print('Task is still running.')
    if details.status == 'COMPLETED':
        flag['taskRunning'] = False
        print('Task has completed, life is beautiful')
    elif details.status == 'FAILED':
        print('Task failed, can not continue')
        raise KeyboardInterrupt
    else:
        sleep(loop_time)

```

- 10** Once the task completes, create an export volume and export the output files to the volume.

```

volume_export = api.volumes.create_s3_volume(
    name='my_volume_RW',
    bucket='BUCKET-NAME',
    access_key_id='INSERT ACCESS KEY ID',
    secret_access_key = 'INSERT SECRET ACCESS KEY',
    access_mode='RW')

exports = []
my_task = api.tasks.get(id=my_task.id)
for curr_output in my_task.outputs:
    f = my_task.outputs[curr_output]
    export = api.exports.submit_export(file=f,
                                      volume=volume_export,
                                      location=f.name)

    exports.append(export)
for j in exports:
    print('File {} export stated {}; it is {}'.format(j.destination, j.started_on, j.state))
print('\n')

```

Basic Protocol 5: Deploying reproducible and scalable containerized tools implemented in CWL

Computational reproducibility, the ability to replicate a previously-completed analysis, remains a significant challenge in biomedicine (Peng, 2011). Obstacles to reproducibility include the facts that each operating environment may have different software versions installed and that the porting process often relies on documentation and manual effort, thus creating multiple points of potential misalignment.

To address the challenge associated with reproducible bioinformatics analyses, we encourage the use of technologies such as Docker and Singularity that automate the creation and distribution of software containers. Within these software containers users can package tools in a discrete environment with its own virtual file system and all of its dependencies, making it easier to share data analysis scripts and tools with all necessary dependencies.

Even with the implementation of software container technology, however, additional barriers to reproducible analysis persist. Software containers may contain all the components or pieces for performing an analysis but not the instructions on how to use them. To address this issue, the biomedical community has developed the concept of standardized descriptions for containers to further enhance reproducibility and simplify software sharing. One such emerging standard is CWL, which supports non-containerized tools and Docker- and Singularity-based tools.

In this protocol, we demonstrate how to port the tool FunSeq2 (Fu et al., 2014) to the CGC using Docker and CWL. FunSeq2 utilizes information from inter-species conservation, intra-species conservation, sequence motifs, and functional genomics assays to annotate and prioritize somatic variants in tumor samples.

Necessary Requirements

- A computer with internet access (e.g. Firefox, Chrome, Safari).
- An account on the CGC (<https://cgc.sbgenomics.com>)
- Docker installed on a computer

The Dockerfile and CWL for the tools used in Basic Protocol 5 is available on github at <https://github.com/sbg-cancer-cloud/Current-Protocols-in-Bioinformatics-2017>.

1. To deploy a tool on the CGC, first install that tool within a Docker container and upload the tool image to the CGC Images Repository or to the Docker hub as a public image (Support Protocol 4).
2. Once you have uploaded your image to the CGC, describe the tool using the CGC Tool Editor or the Rabix Command Line Interface. To use the Tool Editor, click the Apps tab within a project on the CGC and select “Add App” followed by “New Tool”. You can give the tool a name such as “funseq2” (Figure 18).

3. Within the Tool Editor, there are five tabs: “GENERAL”, “INPUTS”, “OUTPUTS”, “ADDITIONAL INFO”, and “TEST”. In the next few steps, we will discuss each of these tabs and how they can be used to describe and test the command line output when the tool is run on AWS. The Rabix Command Line Interface will create the CWL description of the tool based on what we enter into these tabs. Within the GENERAL Tab (Figure 19):

- a. Add the Docker image address in the field “Docker Repository”.
- b. As needed, change the RAM (3 GB required) and CPU requirements based on what is known about the tool.
- c. FunSeq2 contains a script FunSeq2.sh that needs to be run from the same directory. The data context or input index files can be used as an input. To run the tool, include the base command:

```
cd /opt/funseq2-1.2/ && tar xvf
```

- d. CWL allows javascript expressions to ensure dynamic expression changes based on inputs. (A javascript cookbook with most useful javascript expressions on the CGC is available at <http://docs.cancergenomicscloud.org/docs/javascript-cookbook>.) Create a javascript function that returns the value of the complete path to the database file. The javascript for that base command should be:

```
$job.inputs.database.path
```

where the database refers to the input port (in Step 4) for the archived database file. Each input port on the CGC is available as \$job.inputs.<input-port-name>. In addition, files are javascript objects whose full path is available along with associated metadata for each input file.

- e. Later, we will run the following command within the command line and build different parameters based on javascript expressions and input parameters. For now, you can add:

```
./funseq2.sh -nc -o ./
```

to the base command to ensure that FunSeq2 is run in the mode where it only prioritizes noncoding elements and the output is written to the current directory.

4. Click on the INPUTS tab and add different ports for the input files and input parameters (Figure 20):

- a. Create an input port for the database archive file. Enter “database” for ID. The input type is “File” because we expect an input file here. In addition to files, the following input types are often used to describe parameters within CWL typically - integer, float, string, boolean, enumeration (i.e., one of several values), and array (i.e., multiple value of the previous input types).
 - b. Choose “Required input” (top of the Figure 20) to indicate that a task cannot begin unless this file is not specified.
 - c. Give a description to the input parameter such as “Prebuilt data context for FunSeq2”.
 - d. Choose the input file type to be TAR, as this is an archived file.
 - e. If the tool requires secondary files (like BAI for BAM), add them under secondary file and indicate the appropriate file extensions.
 - f. If you select “Include in command line” (bottom of Figure 20), you will get additional options. Within these additional options, you can choose command line flags, which position the command line flag appears in, and how the command line flag is separated from the input parameter value. **Note:** Because the database file is already being unarchived in the base command (Step 3d), we do not need to add it again to the command line within the input parameter tab.
 - g. Continue to add other input parameters one by one. In the case of FunSeq2, this includes maximum indel cutoff (integer), indel analysis (boolean), output format (enumeration with possible values “BED” or “VCF”), use unweighted scoring scheme (boolean), score threshold (boolean), input SNVs (can be an array of files), input format (enumeration with possible values “BED” or “VCF”), maf (float), genome type (enumeration with possible values “Somatic Genome” or “Germline or Personal Genome”), gene list (file), gene expression matrix (file), class for samples in gene expression matrix (file), gene expression format (enumeration with possible values “RPKM” or “raw”), number of threads to indicate how parallel the task is when it is run (integer), cancer type (string), and use recurrence database (boolean). Some of these variables can be built in command line (like cancer type), whereas some can be used to build in additional javascript expressions within the GENERAL tab (like maximum indel cutoff, which is only used when indel analysis is set to True).
5. Now capture outputs within the output ports (Figure 21). Add an output port:
 - a. Set ID “CandidatesSummary”.
 - b. Set Type to be “File”.
 - c. Set Glob to be “Candidates.Summary”.

- d. Set Description to be “brief output of potential candidates”.
 - e. Set Inherit Metadata from Input to be “#inputSNVs”..
 - f. Create similar output ports for detailed results (file), indel results (file), recurrence summary (file), error log (file), differentially expressed genes (file), and differential expression plot (file).
6. Within the ADDITIONAL INFORMATION tab, add information about the tools such as a brief description of the tool, version number of tool, links to the tool homepage, publication, source code, Docker image, manual, author of the tool, tool license, and tool labels.
 7. Within the TEST tab, test how the command line will look when different parameter values are set. For example, when maximum length indel cutoff is 20 with indel analysis turned to True, the -len flag should appear at the bottom command line with a value of 20.

Support Protocol 4: Creating a Docker container for FunSeq2

All tools on the CGC are deployed using Docker containers. This makes the tools portable across multiple computing environments and deployable on an AWS instance immediately before it is used to process data.

You can use Docker and CWL to port any analytical tool onto the platform. This requires Docker to be installed on your machine. You then can create a Dockerfile that contains the commands required to install the tool onto a Docker image. We will present an example with FunSeq2, a tool that prioritizes noncoding variants based on a combination of evolutionary pressure as well as functional genomics datasets. Installation of FunSeq2 on a Linux machine requires the installation of bedtools2, tabix, TFM-pvalue (which calculates P-values for transcription factor motifs based on position-specific scoring matrices), bigWigAverageOverBed (average bigwig signal files over regions present in a bed file), and FunSeq2 itself. We recommend that the tool is installed in a directory such as /opt/. The index or data files can be used as an input to the tool so that the tool can be used with different data contexts or different formats/versions of reference files from within the same species. This also keeps the Docker image lean.

1. To install FunSeq2 (if Dockerfile is on the current directory), open a unix terminal on a linux or mac OS X machine and type:

```
> docker build -t funseq2 .
```

2. Now push this Docker image to a Docker repository that is compliant with the CGC (public Docker image on Docker Hub or cgc-images). To upload to cgc-images, use commands:

```
> docker tag funseq2 cgc-images.sbgenomics.com/<cgc-user-id>/
funseq2.0.6:latest
```

```
> docker push cgc-images.sbgenomics.com/<cgc-user-id>/  
funseq2.0.6:latest
```

Here, “latest” refers to the tag of the Docker image whereas `cgc-images.sbgenomics.com/<cgc-user-id>/funseq2.0.6:latest` is the complete Docker image address. The image address should be unique for each tool.

At this stage you can port the tool to the CGC using our GUI for our SDK as explained in Basic Protocol 5. We recommend that all index files be maintained as separate archived and uploaded files on the platform to reduce the size of the Docker image as well as to retain the flexibility of using the same Docker image with different genome indices or data contexts. Index files can be added as an input port to the tool as explained in step 4 of Basic Protocol 5.

Commentary

Background Information—Due to the formidable costs associated with downloading and analyzing petabytes of TCGA data, the NCI funded the creation of the three Cancer Genomics Cloud Pilots to co-locate data with elastic cloud computing resources. The three Cloud Pilots enable a common set of functionalities: (1) secure access to the TCGA dataset and other public datasets as well as tools for uploading private data; (2) access to optimized bioinformatics workflows as well as tools to create and run custom workflows; and (3) secure collaborative workspaces. Although the underlying architecture and some features of the three Cloud Pilots differ (e.g., leveraging different commercial cloud resource providers and offering custom query and analysis tools), all three Cloud Pilots demonstrate viable solutions for overcoming several of the challenges associated with accessing petabyte-scale datasets, thereby facilitating their analysis by the research community.

The five protocols detailed above were chosen to provide an introduction to some of the basic functionalities of the Seven Bridges CGC. The CGC is continuously being developed and can be extended to perform most types of bioinformatic analysis on the cloud using public as well as private multi-omic datasets. Extensive additional documentation is available through our CGC Knowledge Center (<http://docs.cancer-genomics-cloud.org>), and training materials including presentations, workshops, and hackathons are released regularly to the public.

Building on the success of the CGC and the other Cloud Pilots, the coming years will see the creation of several cloud-based Data Commons efforts, including the NCI Cancer Research Data Commons (for which the three Cloud Pilots will continue to play a leading role); the National Human Genome Research Institute (NHGRI) Genomic Data Science Analysis, Visualization, Informatics Lab-space (AnVIL); and the NIH Data Commons. A shared goal of these efforts is to create central, interoperable workspaces so that users can easily access datasets stored in distinct Data Commons. To make these goals a reality, these Data Commons efforts will work towards the creation of common standards that make data and tools available according to FAIR (Findable, Accessible, Interoperable, and Reusable) principles. Researchers then will be able to readily perform queries and computational

analyses across datasets, which is expected to lead to insights not achievable using the same datasets in isolation. The three Cancer Genomics Cloud Pilots have proven to be an important first step towards achieving truly democratized, interoperable access to and analysis of large-scale omic datasets, and we anticipate significant further gains for the wider research community in the years to come through these Data Commons efforts.

Acknowledgments

The CGC has been funded in whole or in part with Federal funds from the National Cancer Institute, National Institutes of Health, Department of Health and Human Services, under Contract No. HHSN261201400008C. We also thank the CGC team for creating, maintaining, and continuously improving the different functionalities mentioned within the manuscript.

Literature Cited

- Amstutz, P., Crusoe, MR., Tijani, N., Chapman, B., Chilton, J., Heuer, M., Stojanovic, L. Common Workflow Language, v1.0. 2016 Jul 8. <https://doi.org/10.6084/m9.figshare.3115156.v2>
- Barretina J, Caponigro G, Stransky N, Venkatesan K, Margolin AA, Kim S, Garraway LA. The Cancer Cell Line Encyclopedia enables predictive modelling of anticancer drug sensitivity. *Nature*. 2012; 483(7391):603–607. [PubMed: 22460905]
- Becnel LB, Pereira S, Drummond JA, Gingras M-C, Covington KR, Kovar CL, Gibbs RA. An open access pilot freely sharing cancer genomic data from participants in Texas. *Scientific Data*. 2016; 3:160010. [PubMed: 26882539]
- Benelli M, Pescucci C, Marseglia G, Severgnini M, Torricelli F, Magi A. Discovering chimeric transcripts in paired-end RNA-seq data by using EricScript. *Bioinformatics*. 2012; 28(24):3232–3239. [PubMed: 23093608]
- Weinstein JN, Collisson EA, Mills GB, Shaw KRM, Ozenberger BA, Stuart JM. Cancer Genome Atlas Research Network. The Cancer Genome Atlas Pan-Cancer analysis project. *Nature Genetics*. 2013; 45(10):1113–1120. [PubMed: 24071849]
- Fu Y, Liu Z, Lou S, Bedford J, Mu XJ, Yip KY, Gerstein M. FunSeq2: a framework for prioritizing noncoding regulatory variants in cancer. *Genome Biology*. 2014; 15(10):480. [PubMed: 25273974]
- Ge H, Liu K, Juan T, Fang F, Newman M, Hoeck W. FusionMap: detecting fusion genes from next-generation sequencing data at base-pair resolution. *Bioinformatics*. 2011; 27(14):1922–1928. [PubMed: 21593131]
- Haas B, Dobin A. STAR-FUSION. STAR-Fusion. 2015
- Koboldt DC, Zhang Q, Larson DE, Shen D, McLellan MD, Lin L, Wilson RK. VarScan 2: somatic mutation and copy number alteration discovery in cancer by exome sequencing. *Genome Research*. 2012; 22(3):568–576. [PubMed: 22300766]
- Kumar S, Vo AD, Qin F, Li H. Comparative assessment of methods for the fusion transcripts detection from RNA-Seq data. *Scientific Reports*. 2016; 6:21597. [PubMed: 26862001]
- Mallick S, Li H, Lipson M, Mathieson I, Gymrek M, Racimo F, Reich D. The Simons Genome Diversity Project: 300 genomes from 142 diverse populations. *Nature*. 2016; 538(7624):201–206. [PubMed: 27654912]
- Nicorici, D., Satalan, M., Edgren, H., Kangaspeska, S., Murumagi, A., Kallioniemi, O., Kilku, O. FusionCatcher - a tool for finding somatic fusion genes in paired-end RNA-sequencing data. *bioRxiv*. 2014 Nov 19. <https://doi.org/10.1101/011650>
- Peng RD. Reproducible research in computational science. *Science*. 2011; 334(6060):1226–1227. [PubMed: 22144613]
- Veeraraghavan J, Ma J, Hu Y, Wang X-S. Recurrent and pathological gene fusions in breast cancer: current advances in genomic discovery and clinical implications. *Breast Cancer Research and Treatment*. 2016; 158(2):219–232. [PubMed: 27372070]

- Yoshihara K, Wang Q, Torres-Garcia W, Zheng S, Vegesna R, Kim H, Verhaak RGW. The landscape and therapeutic relevance of cancer-associated transcript fusions. *Oncogene*. 2015; 34(37):4845–4854. [PubMed: 25500544]
- Yung, CK., O'Connor, BD., Yakneen, S., Zhang, J., Ellrott, K., Kleinheinz, K. PCAWG Technical Working Group. Large-Scale Uniform Analysis of Cancer Whole Genomes in Multiple Computing Environments. *bioRxiv*. 2017. <https://doi.org/10.1101/161638>

Metadata Entity Structure

Figure 1.

The hierarchical metadata structure for TCGA (hg19 reference) on the CGC. The graphical structure that encapsulates the relationship between entities is shown on the left. Not shown is the entity Files, which is related to all other entities. Shown on the right are examples of the sets of related properties that describe each entity.

Log In

To access the full functionality of the CGC, log in with your eRA Commons or NIH cit account.

To access Controlled Data, you need to be approved for data access by dbGaP and must agree to use the data only in a manner consistent with your data access request. For more details click [here](#).

Having trouble? Try [resetting](#) your eRA Commons password.

[LOGIN VIA ERA COMMONS](#)

Log In without NIH authentication

Use this option if you don't have an eRA Commons or NIH cit account.

[forgot?](#)

☐ Remember me

[LOGIN](#)

New? [Create a free account](#)

Figure 2.
Login via eRA Commons.

Figure 3.
Select the legacy TCGA dataset from the Data Browser starting page.

Figure 4.
Case entity with 1101 Breast Invasive Carcinoma Cases.

Figure 5. A Data Browser query for raw RNA-Seq files from matched tumor-normal samples for breast invasive carcinoma disease cases. There are 113 cases containing 126 raw RNA-Seq files for primary tumor sample types and 114 raw RNA-Seq files for solid tissue normal sample types.

CANCER GENOMICS CLOUD
SEVEN BRIDGES

Log In

To access the full functionality of the CGC, log in with your eRA Commons or NIH cit account.

To access Controlled Data, you need to be approved for data access by dbGaP and must agree to use the data only in a manner consistent with your data access request. For more details click [here](#).

Having trouble? Try [resetting](#) your eRA Commons password.

LOGIN VIA ERA COMMONS

Log In without NIH authentication

Use this option if you don't have an eRA Commons or NIH cit account.

Username or email

Password

forgot?

☐ Remember me

LOGIN

New? [Create a free account](#)

Warning Notice

This is a U.S. Government information system, which may be accessed and used only for authorized Government business by authorized personnel. Unauthorized access or use of this system may subject violators to criminal, civil, and/or administrative action.

All information on this computer system may be intercepted, recorded, read, copied, and disclosed by and to authorized personnel for official purposes, including criminal investigations. Such information includes sensitive data encrypted to comply with confidentiality and privacy requirements. Access or use of this computer system by any person, whether authorized or unauthorized, constitutes consent to these terms. There is no right of privacy in this system.

Access to the CGC is governed by federal and state laws and regulations and by data use agreements executed by authorized users. Unauthorized access or use of the CGC may subject violators to criminal, civil, and/or administrative action. Access to the entire policy can be found on the [National Institutes of Health \(NIH\)](#) website. Furthermore, the CGC falls under TCGA [policies](#) and [data use](#) agreements governing access and/or use of the CGC.

Funding

The CGC is powered by [Seven Bridges](#) and has been funded in whole or in part with Federal funds from the National Cancer Institute, National Institutes of Health, Department of Health and Human Services, under Contract No. HHSN261201400008C.

Figure 6.

The landing page for creating a CGC login shows the two methods to register for an account on the CGC.

Register

First name Last name

Email

Username

Password

- 8 or more characters
- Uppercase and lowercase letters
- At least one number

Organization or company

Location
United States ▼

☐ I have read and accept [the CGC Terms](#)

☐ I have read and accept [all relevant data use policies](#)

☐ I'm not a robot
reCAPTCHA
[Privacy - Terms](#)

REGISTER

Existing user? [Log in](#)

Figure 7.
Register on the CGC using an email address.

8Trust
NIH SECURE IDENTITY SOLUTIONS

User Name:

Password: [Change Password](#)

Log in

OR

Insert your PIV card into your smart card reader before attempting to login.

For more information visit <http://smartcard.nih.gov>.

Log in

Warning Notice

- This warning banner provides privacy and security notices consistent with applicable federal laws, directives, and other federal guidance for accessing this Government system, which includes (1) this computer network, (2) all computers connected to this network, and (3) all devices and storage media attached to this network or to a computer on this network.
- This system is provided for Government-authorized use only.
- Unauthorized or improper use of this system is prohibited and may result in disciplinary action and/or civil and criminal penalties.
- Personal use of social media and networking sites on this system is limited as to not interfere with official work duties and is subject to monitoring.
- By using this system, you understand and consent to the following:
 - The Government may monitor, record, and audit your system usage, including usage of personal devices and email systems for official duties or to conduct HHS business. Therefore, you have no reasonable expectation of privacy regarding any communication or data transiting or stored on this system. At any time, and for any lawful Government purpose, the government may monitor, intercept, and search and seize any communication or data transiting or stored on this system.
 - Any communication or data transiting or stored on this system may be disclosed or used for any lawful Government purpose.

Figure 8.

Connecting an NIH eRA Commons account to access dbGaP-regulated Controlled Data on the CGC.

Projects ▾ **Data** ▾ **Public Apps** **Public projects** ▾ **Staff** ▾ ▾ ▾ **anurag.sethi** ▾

Dashboard Files Apps Tasks **My Project** **Copy project** Interactive Analysis Settings Notes

Description

Welcome to your new project!

Projects are the core building blocks of the CGC Platform. Each project corresponds to a distinct scientific investigation, serving as a container for its data, analysis pipelines, and results. Projects are shared only by designated project members.

Within your project, you can:

- Start [exploring TCGA dataset](#) straight away
- [Install your tools on the CGC](#) and create workflows
- [Upload your own private data](#) and analyze it along with TCGA data
- [Collaborate securely](#) with other researchers

Please record the details of your project here, such as its aims, experimental context, and any other ideas that you'd like to share with your project members. Remember that details of each pipeline execution you run on the CGC are logged on the task page. This notepad is just for your own notes.

You can also [use markdown](#) here to add formatting to your notes.

Good luck with your research! If you get stuck, take a look at the [Knowledge Center](#)

The Seven Bridges CGC Team

Members Email notifications

anurag.sethi owner
Write, Copy, Execute, Admin

isheeta.seth_cgcdemo
Write, Copy, Execute

raunaqm_cgc
Execute

elehnert
Read

Manage members

Tasks

Your executions will appear here.
Before you start, [learn more about them](#).

Figure 9.
Project dashboard for a project with no Controlled Data access on the CGC.

The screenshot displays the 'Controlled Project' dashboard on the CGC platform. The top navigation bar includes links for Projects, Data, Public Apps, Public projects, and Staff, along with a user profile for anurag.sethi. The dashboard is divided into three main sections: Description, Members, and Tasks. The Description section contains a welcome message, an explanation of projects as core building blocks, and a list of actions users can take within a project. The Members section shows a list of project members with their roles and permissions. The Tasks section is currently empty, showing a placeholder for future executions. The footer contains copyright information for Seven Bridges Genomics, Inc. and links to various resources like the Knowledge Center, Developer, Forum, Policies, Data Use, Terms, and Contact.

Projects **Data** **Public Apps** **Public projects** **Staff** **anurag.sethi**

Dashboard **Files** **Apps** **Tasks** **CONTROLLED** **Controlled Project** **Copy project** **Interactive Analysis** **Settings** **Notes**

Description

Welcome to your new project!

Projects are the core building blocks of the CGC Platform. Each project corresponds to a distinct scientific investigation, serving as a container for its data, analysis pipelines, and results. Projects are shared only by designated project members.

Within your project, you can:

- Start [exploring public datasets](#) straight away
- [Install your tools on the CGC](#) and create workflows
- [Upload your own private data](#) and analyze it along with public datasets
- [Collaborate securely](#) with other researchers

Please record the details of your project here, such as its aims, experimental context, and any other ideas that you'd like to share with your project members. Remember that details of each pipeline execution you run on the CGC are logged on the task page. This notepad is just for your own notes. You can also use [markdown](#) here to add formatting to your notes.

Good luck with your research! If you get stuck, take a look at the [Knowledge Center](#)

The Seven Bridges CGC Team

Add description

Members **Email notifications**

anurag.sethi **owner**
Write, Copy, Execute, Admin

raunaqm_cgc
Write, Copy, Execute

elehnert
Read

Manage members

Tasks

Your executions will appear here. Before you start, [learn more about them](#).

Copyright © 2017 Seven Bridges Genomics, Inc. All rights reserved. [Knowledge Center](#) [Developer](#) [Forum](#) [Policies](#) [Data Use](#) [Terms](#) [Contact](#) [Send Feedback](#)

focs.cancergenomicscloud.org/docs/sdk-overview

Figure 10.
Project dashboard for a project with Controlled Data access on the CGC.

Figure 11.

The Developer Dashboard on the CGC. The Auth token is found by clicking on **Auth Token** in the center of the screen. If you have not previously generated an Auth Token, you will need to click **Generate** to make one.

Figure 12.
Comparison of cost and time of running three fusion callers on 121 paired-end RNA-Seq datasets from Breast Invasive Carcinoma samples.

Figure 13.

Venn Diagram displaying the total number of unique gene fusion pairs detected by each of the three tools (FusionCatcher, STAR-Fusion and EricScript) and the overlapping number of gene fusion pairs detected.

Figure 14.

Overlapping gene fusion pairs with highest frequency of occurrence (top 10) detected by STAR-Fusion, FusionCatcher or EricScript. The asterisks indicate that the three gene fusion pairs were in the top 10 list for all three tools. Many of the gene fusions detected are known recurrent gene fusions in breast cancer. For example, the ESR|CCDC170 gene fusion is a known recurrent gene fusion transcript in breast invasive carcinoma (Veeraraghavan et al., 2016).

Figure 15.
The workflow editor with the custom star-fusion-workflow.

Figure 16.
Files can be filtered by file name and metadata.

Figure 17.

The volumes API can be used to connect a user's AWS bucket to the CGC.

Add apps to "My Project"

Public Apps

Projects

My Apps

Create New App

W

Workflow

Workflows are chains of interconnected tools.

Create a Workflow

Learn how to build a workflow

T

Tool

Name

funseq2

App URL

https://cgc.sbggenomics.com/u/anurag.sethi/my-project/funseq2

Cancel

Create

Figure 18.
Add a tool to the CGC.

Curr Protoc Bioinformatics. Author manuscript; available in PMC 2018 December 08.

Figure 19.
The **GENERAL** tab settings for adding FunSeq2 to the CGC.

Edit Input Property ☒ required

ID	Type ?	
<input type="text" value="database"/>	<input type="text" value="File"/>	
Allowed characters are alphanumerics and _		
Additional information		
Label	Description	
<input type="text" value="database"/>	<input type="text" value="Prebuilt data context for FunSeq2"/>	
Alternative Prefix	Category	File Types
<input type="text"/>	<input type="text"/>	<input type="text" value="TAR"/>
Stage Input		
<input type="text" value="-- none --"/>		
☐ Load Contents ?		
Secondary Files ?		
<div><input type="button" value="+"/> Click the plus button to add a file</div>		
☐ Include in command line		

Figure 20.

The database port within the INPUTS tab when adding FunSeq2 to the CGC.

Edit Output Property ☐ required

ID

candidatesSummary

Allowed characters are alphanumerics and _

Type ?

File

Glob ?

Candidates.Summar

</>

Additional information

Label

candidates summary

Description

brief output of potential candidates (codin

File Types

Metadata ?

+

Click the plus button to add a metadata key-value pair.

Inherit Metadata From Input

#inputSNVs

☐ Load Contents ?

Output Eval

Output Eval

</>

Figure 21.
The output port for adding FunSeq2 to the CGC.

Table 1
Comparison of the three fusion callers

Comparison of the three fusion callers to analyze 121 paired-end RNA-Seq tumor only Breast Invasive Carcinoma samples. All comparisons in Table 1 and Figures 11–13 are based on the 121 samples that executed successfully for all three fusion callers (for FusionCatcher tasks did not complete execution for 5 of the 126 samples). Comparison is based on total cost, time, CPU cores, and the Amazon instance type used to process the samples.

Fusion detection tool	Total Cost (in US \$)	Total run time (in hours)	CPU cores	Amazon instance type
STAR-Fusion	\$356.44	120.52	4	r3.8xlarge
EricScript	\$269.34	552.73	1	c4.2xlarge
FusionCatcher	\$793.10	619.06	8	r3.4xlarge

Table 2
Benchmarking of the three fusion callers on a positive dataset

Benchmarking of fusion callers on a positive dataset that contains 50 known gene fusions. The table shows the sensitivity, precision, F-score, and number of gene fusions detected by the fusion callers EricScript, FusionCatcher, and STAR-Fusion. The numbers in parentheses indicate the total number of true gene fusions detected by each gene fusion caller.

Fusion Caller	Sensitivity	Precision	F-score	# of predicted fusions (true positives)
EricScript	0.72	0.9	0.8	40 (36)
FusionCatcher	0.52	0.87	0.65	30 (26)
STAR-Fusion	0.82	0.85	0.84	48 (41)