Use GitHub for effective version control and collaborative work

Serghei Mangul, Ph.D

Assistant Professor of Clinical Pharmacy and Biological Sciences, University of Southern California

Learning Objectives

By the end of this modules you should:

- Understand the importance of version control and public repositories
- Be able to use basic git commands to:
  - Create a new repository
  - Pull updates to a repository
  - Push your changes to a repository
- Create easy-to-navigate repository with the analysis accompanying your paper

Why share your data publicly?

Sharing data and code is essential for reproducibility of published research

- Published results can be reproduced by others with access to the data and code
- New results can be generated from existing data by secondary analysis of your data

The proper sharing of research material is required by many journals

- Use of online version control repository to store the files
- Public access for anyone allowing the reuse of the shared material

Where to store your data and code

GitHub

Data availability

Genomic and transcriptomic sequence datasets, including metadata with library construction and sequencing approaches have been deposited at the European Genome–phenome Archive (EGA, http://www.ebi.ac.uk/ega/) as part of the study EGAS00001001159 with accession numbers as listed in Supplementary Table 1. Data on mutations, copy changes and expression from tumor samples in the POG program organized by OncoTree classification (http://oncotree.mskcc.org) are also accessible from https://www.personalizedoncogenomics.org/cbioportal/. The complete small mutation catalog and gene expression TPMs are available for download from http://bcgsc.ca/downloads/POG570/. Previously published TCGA and PCAWG data that were re-analyzed here are available from data portals (https://portal.gdc.cancer.gov/ and https://dcc.icgc.org/) with sample barcodes as listed in Supplementary Table 9. All other data supporting the findings of this study are available from the corresponding author on reasonable request.

Code availability

The bioinformatics analyses were performed using open-source software, including Burrows–Wheeler alignment tool (v.0.5.7 for up to 125 bp reads and v.0.7.6a for 150 bp reads), CNAseq[91] (v.0.0.6), APOLLOH[92] (v.0.1.1), SAMtools[93] (v.0.1.17), MutationSeq[94] (v.1.0.2 and v.4.3.5), Strelka[95] (v.1.0.6), SNPEff[96] (v.3.2 for somatic and v.4.1 for germline), ABySS[97] (v.1.3.4), TransABySS[97,98] (v.1.4.10), Chimerascan[99] (v.0.4.5), DeFuse[100] (v.0.6.2), Manta[101] (v.1.0.0), Delly[102] (v.0.7.3), MAVIS[71] (v.2.1.1), STAR[73] (v.2.5.2b), RSEM[74] (v.1.3.0), MSIsensor[76] (v.0.2), HRDtools[16] (v.0.0.0.9), BioBloomTools[78] (v.2.0.11b), EXPANDS[84] (v.2.1.1), SignIT (https://github.com/eyzhao/SignIT), samtools[93] (v.0.1.17), ClinVar[103] (v.20180905), InterVar[88], ControlFREEC[104] (v.5), CIBERSORT[89] (v.1.04), Jaguar[105] (v.2.0.3), MiXCR[90] (Java, v.2.1.2) and VDJtools[106] (v.1.1.9). Additional packages used for meta-analyses include R packages ClusteredMutations (v.1.0.1), vegan (v.2.5.3), ConsensusClusterPlus (v.1.44.0), ComplexHeatmap (v.1.18.1), survival (v.2.42.3), survminer (v.0.4.2) and Python package scikit-learn[86] (Python, v.0.20). Additional processing involved in-house scripts that are available upon request.

Why use GitHub?

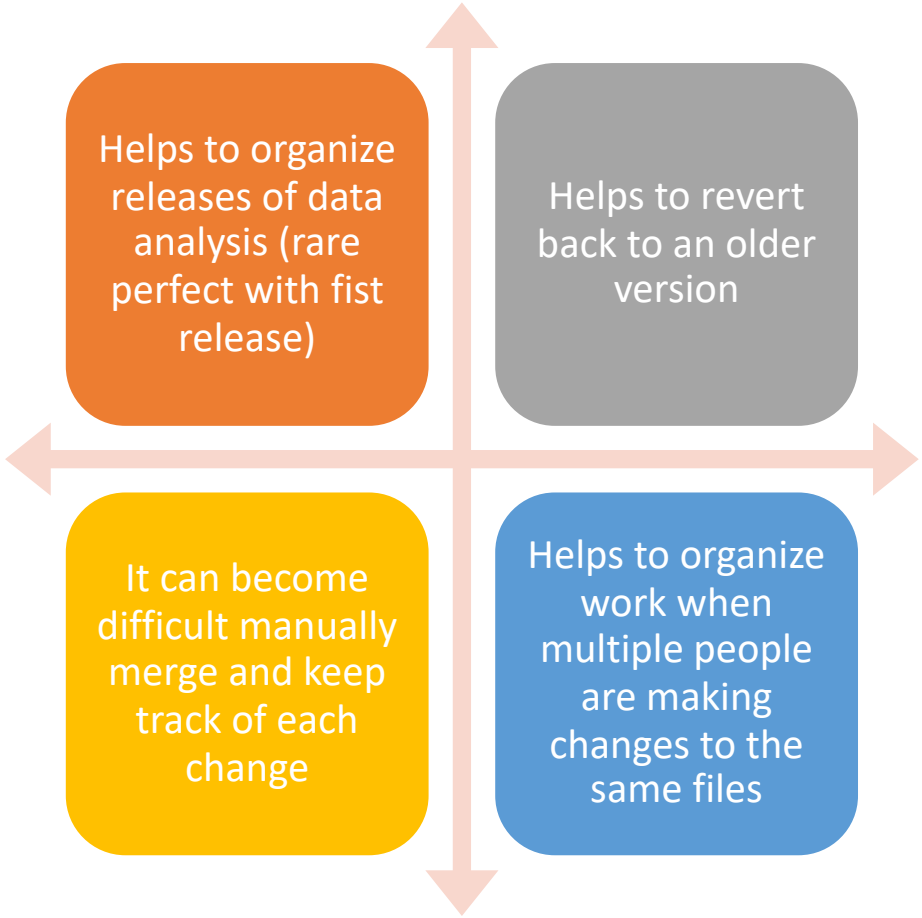
- GitHub is repository hosting service that can be used to publicly share data and code
  - Widely adopted at the academia community
  - Allows the sharing of data and code by providing a URL to the repository
  - Uses version control (a process of keeping track of file changes)

What is version control

- The term version control refers to a system that records changes to a file or set of files over time called the ‘versions’.
- In other words, these versions will help you in tracking the changes in your data/code and undo those changes as well

https://www.edureka.co/blog/git-vs-github/

Why use version control



Git

Git is a most widely and popularly used modern version control software

It is an open source project. It was developed back in 2005, by Linus Torvalds who is also the creator of Linux.

Git Concepts

SNAPSHOTS

METHOD USED BY GIT TO KEEP TRACK OF CHANGES

A SNAPSHOT REFLECTS THE STATE OF THE FILES AT A CERTAIN INSTANT IN TIME

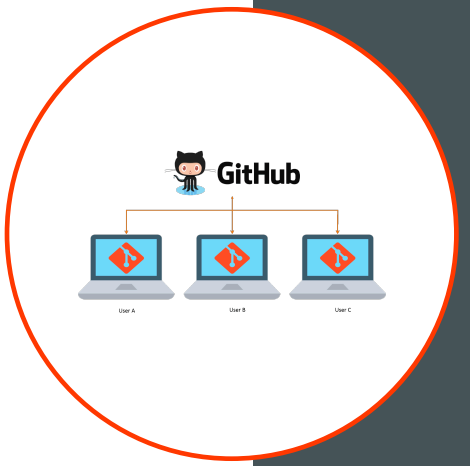
REPOSITORIES

COLLECTION OF ALL FILES/FOLDERS AND THEIR HISTORY (ALL SNAPSHOTS)

OFTEN SHORTENED TO ‘REPO’

GitHub is a web-based git version control repository hosting service

- It is a place to store your **repositories**
- GitHub with an abundant storage space where you can store your repositories and build a proper profile which holds a **great value**
- By default the **repositories** are public i.e., everyone can see your code and data but you can make it private as well



Git vs GitHub

1. It is a software
2. It is installed locally on the system
3. It is a command line tool
4. It is a tool to manage different versions of edits, made to files
5. It provides functionalities like version control and data management

1. It is a service
2. It is hosted on web
3. It provides a graphical interface
4. It is a space to upload a copy of the **git** repository
5. It provides functionalities of git as well as adding few of its own features

https://www.edureka.co/blog/git-vs-github/

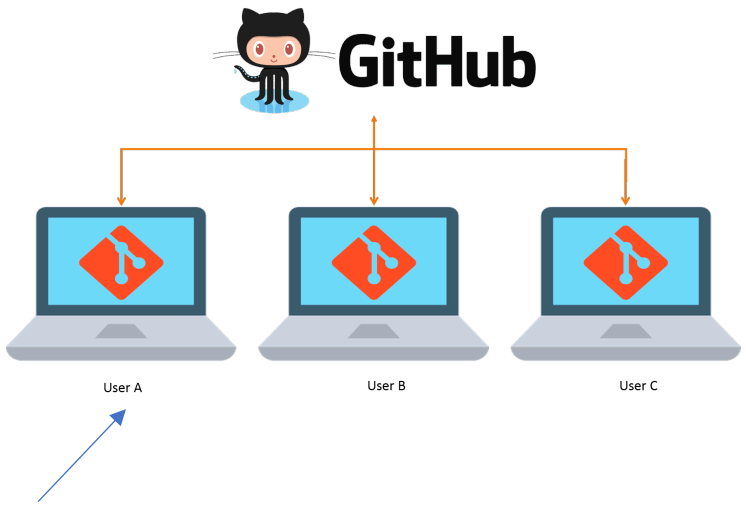
Useful features of version control

- **Backup and Restore.** Files are saved as they are edited, and you can restore files from any version
- **Effective collaboration.** Synchronize files with other member of the team. Allows people to share files and stay up-to-date with the latest version
- **Short-term undo.** Throw away your changes and go back to the “last known good” version in the analysis
- **Track Changes**. As files are updated, you can leave messages explaining why the change happened. This makes it easy to see how a file is evolving over time, and why.
- **Sandboxing**, or insurance against yourself. Making a big change? You can make temporary changes in an isolated area, test and work out the kinks before “checking in” your changes online

https://betterexplained.com/articles/a-visual-guide-to-version-control/

GitHub

GitHub stores the repo (https://github.com/)



Working Copy: Your local directory of files, where you make changes

Essential git commands

- git clone
- git add
- git commit
- git push
- git pull

Essential git commands

Git clone creates a copy of an existing remote repository

Git add puts a file into the repo for the first time, i.e. begin tracking it with version control

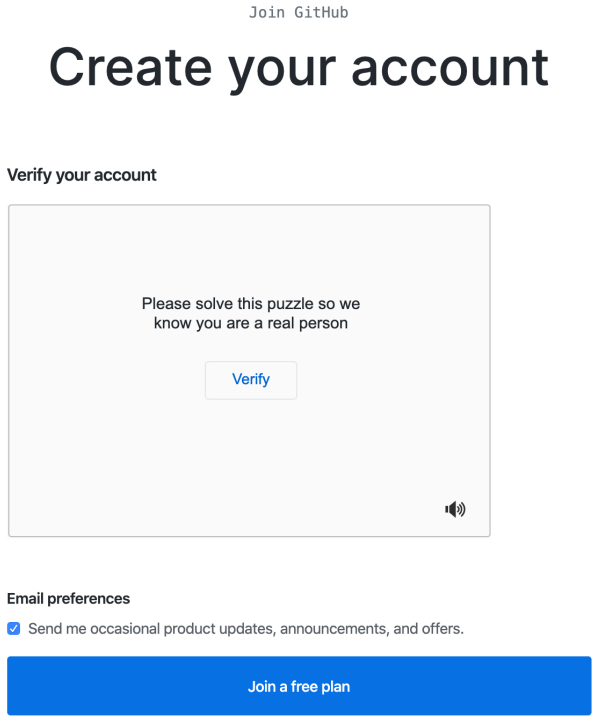
Git commit captures a snapshot of the project

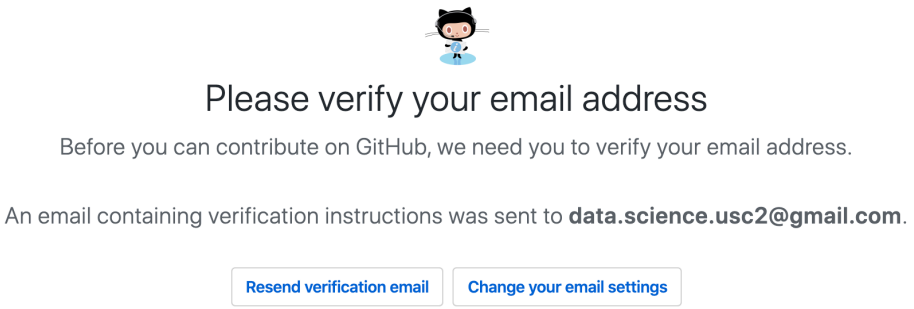
Git push pushes any changes in your local repo to the repo on GitHub

Git pull obtains all updates to the repository on GitHub, and merge them with your local repo

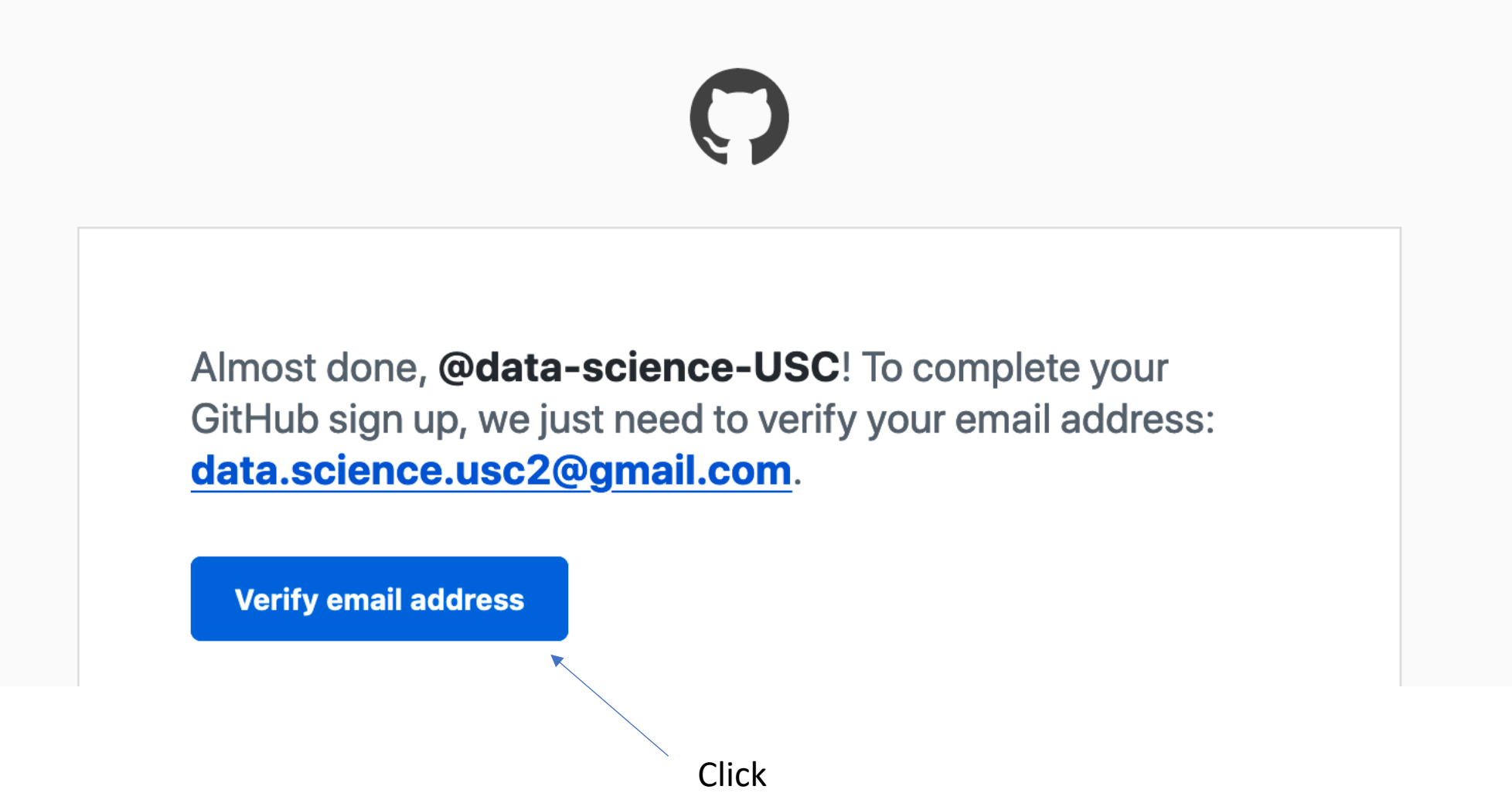
Open GitHub account

Go to
https://github.com/



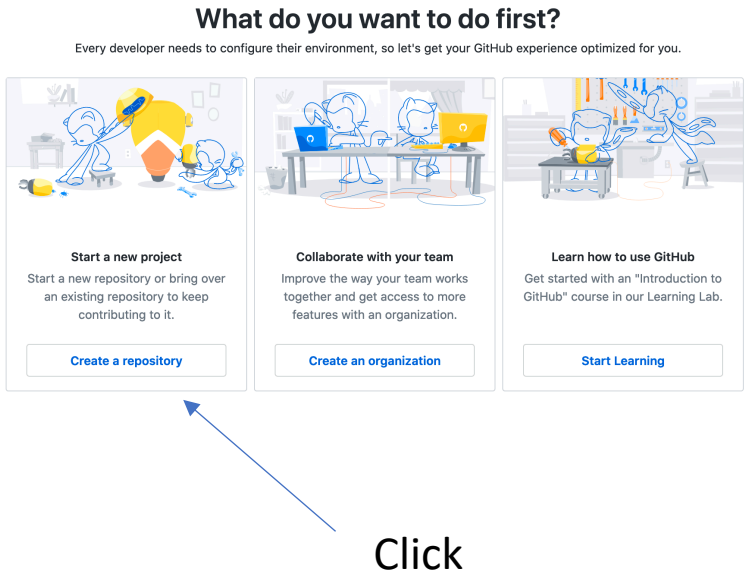


[GitHub] Please verify your email address.

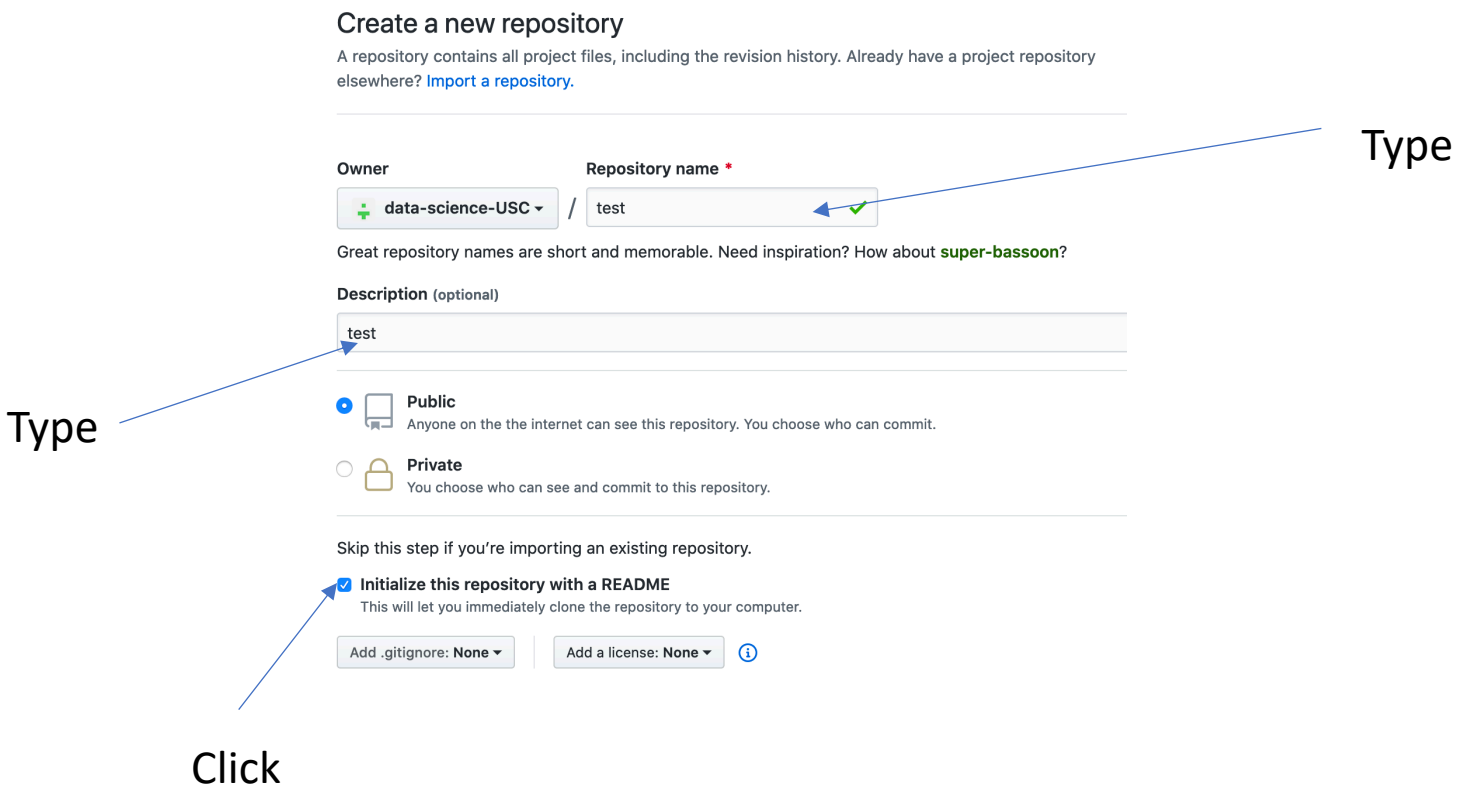


Creating a GitHub repo

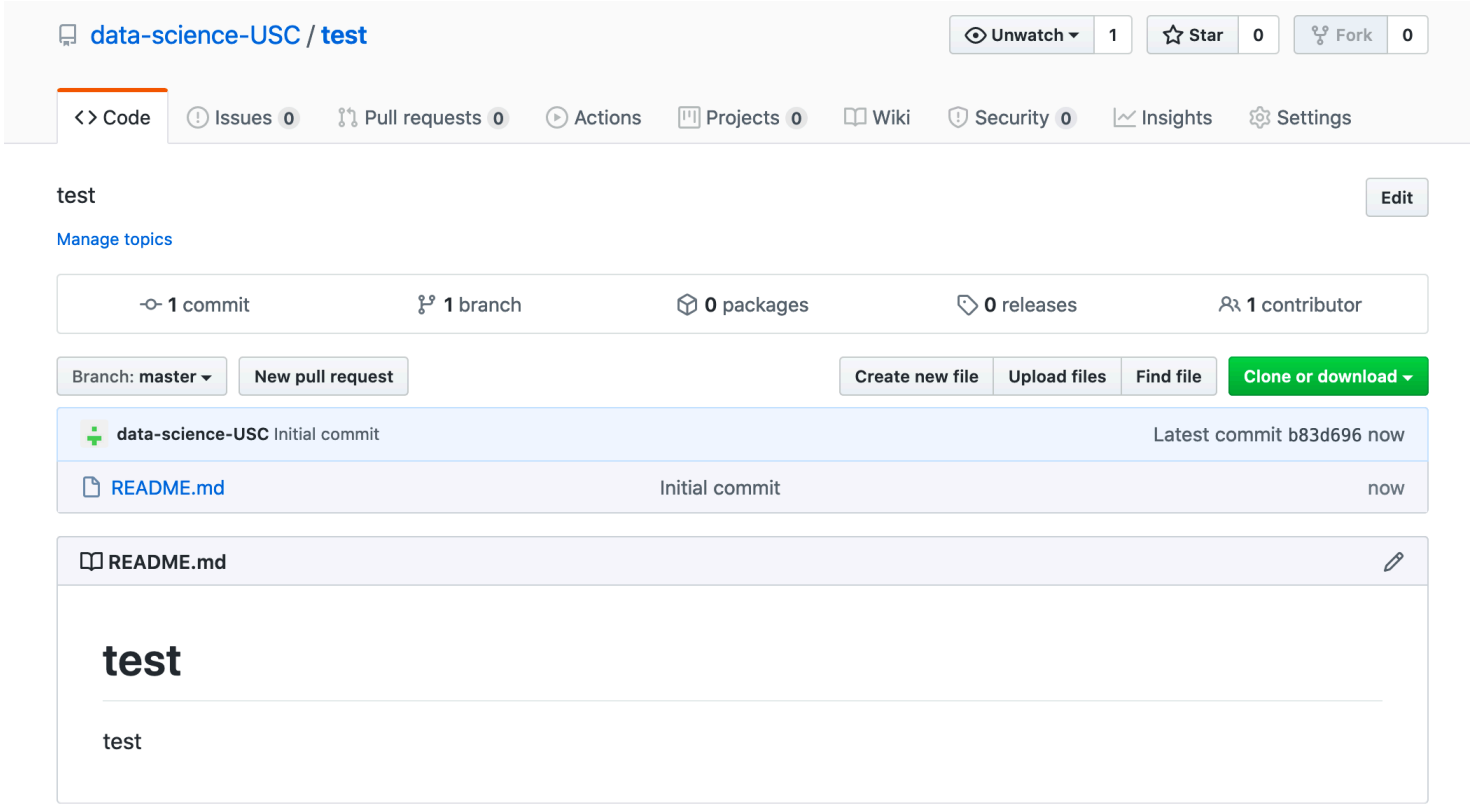
- Let’s create a new repository on GitHub
  - Go to your GitHub page
  - Navigate to the “Repositories” tab
  - Hit the “New” button in the upper right corner
  - Fill out a name, and check the box that says “Initialize this repository with a README”
  - Then hit “Create Repository”
  - This will take you to your new repo!



Creating a GitHub repo



Your first GitHub repo

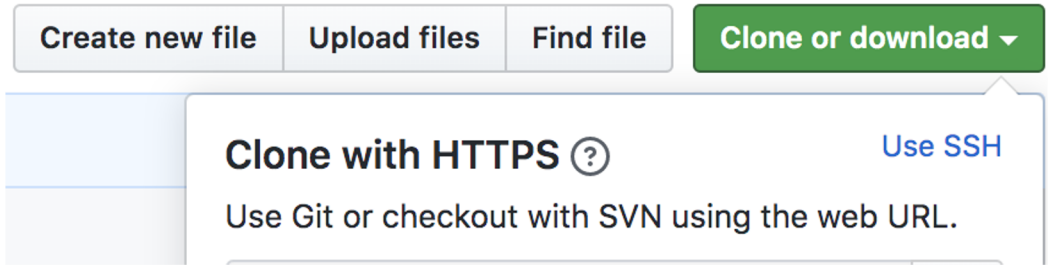


Git clone

Create a copy of an existing remote repository

Clone GitHub repos

- Now, how do we interact with this repo?
- We will want to clone a local copy of this repo using command line
  - To do this go to the upper right corner of your repo page
  - Select “Clone or Download”
  - This creates a dropdown menu
  - There should be a link, copy the link
- Now, open up a terminal
  - Navigate to where you want to locate this repo, then use the clone command
  - This will copy the repo to your computer




```
git clone <the-https-link-you-copied>
```

Interacting with GitHub via Google Colab

Git and bash commands can be run on Google Colab notebooks

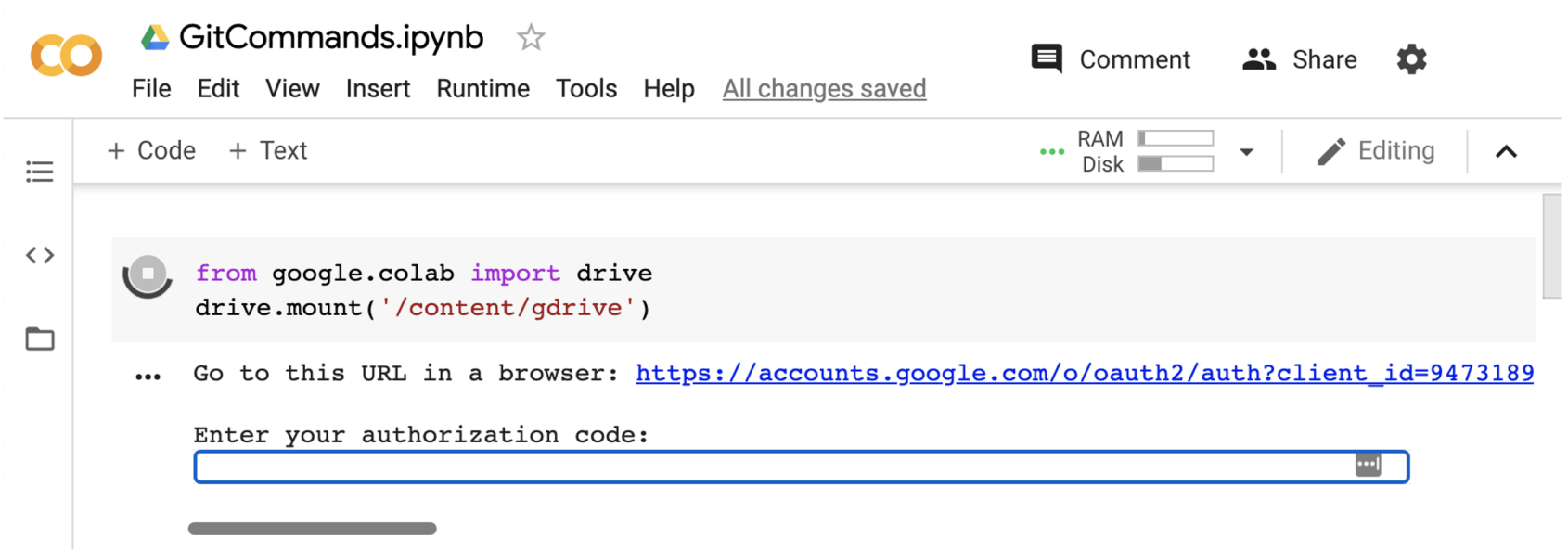
- Use ‘!’ before git and bash commands
  - ! git add .
- Use ‘%’ before bash commands to navigate through directories
  - % cd test

Google Drive can be used for permanent storage

- Colab storage is non-persistent
- Files created on notebooks can be saved on your Drive (See Module 1)

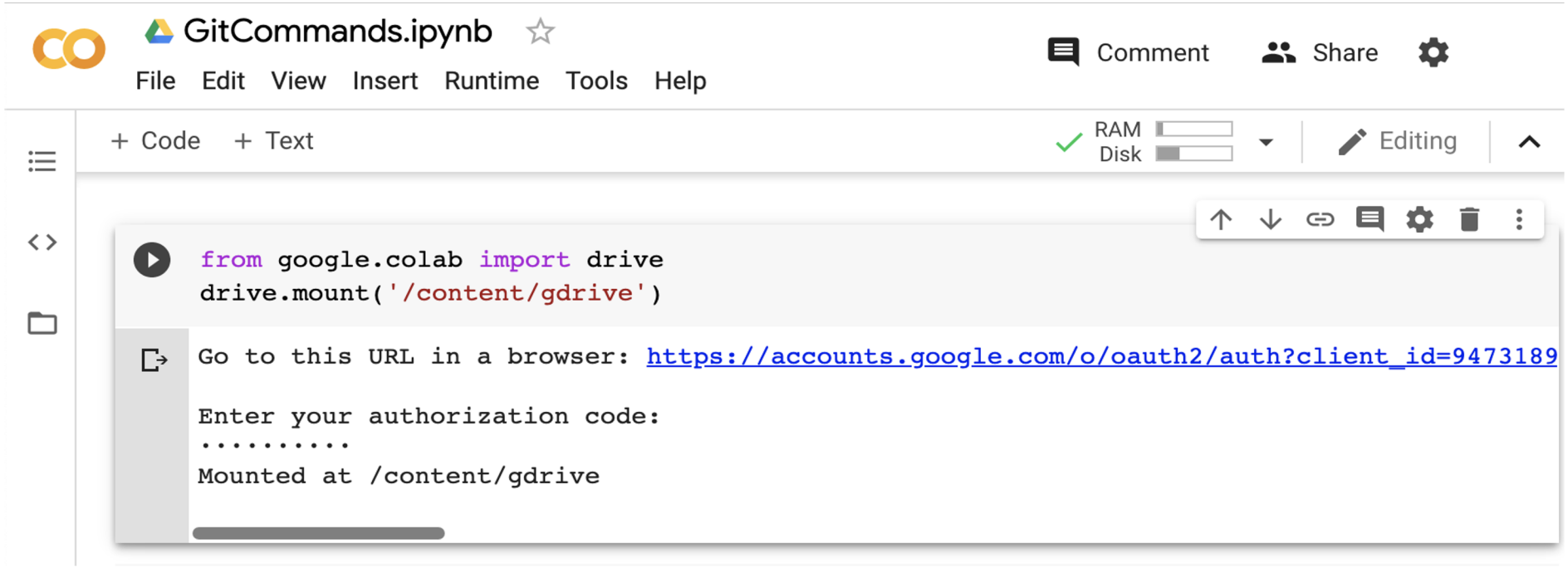
Mounting your Google Drive

- Open the url to authorize Colab access your Google Drive
- Copy and paste the authorization code



Mounting your Google Drive

- Google Drive is now mounted at /content/gdrive



Interacting with GitHub via Google Colab

- Mounting your Google Drive
  - Access ‘/content/drive/My Drive’ to clone the repo

```
[1] from google.colab import drive
    drive.mount('/content/drive')
```

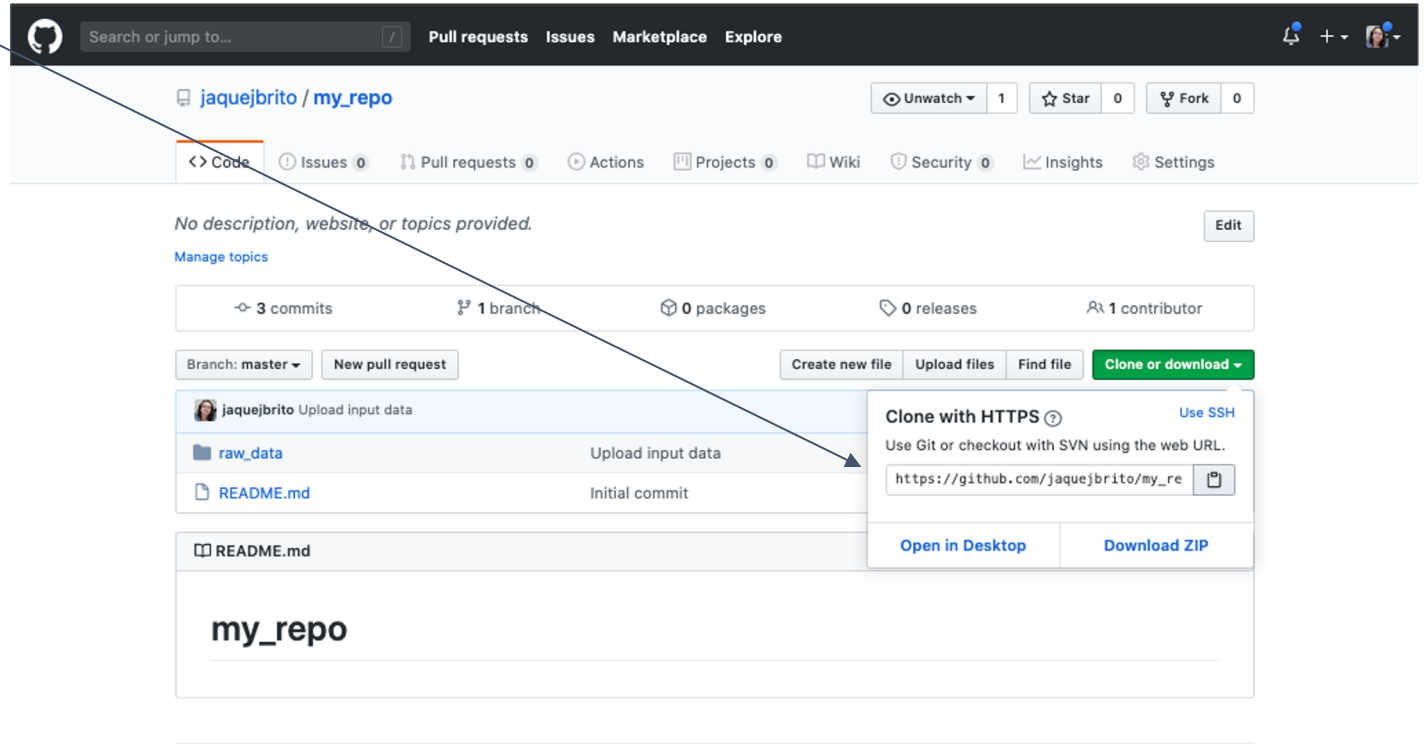
```
Go to this URL in a browser: https://acc

Enter your authorization code:
··········
Mounted at /content/drive
```

```
%cd /content/drive/My\ Drive/
!ls
```

```
/content/drive/My Drive
data_science_workshop
```

Interacting with GitHub via Google Colab

- Copy the repo URL for cloning it in Google Colab



Interacting with GitHub via Google Colab

- Cloning the repository into Google Drive
  - Use the copied URL to clone the repo at ‘/content/gdrive/My Drive’

Create a directory in GitHub repository

```
%cd notebooks
!mv /content/drive/My\ Drive/data_science_workshop/module1.ipynb ./
```

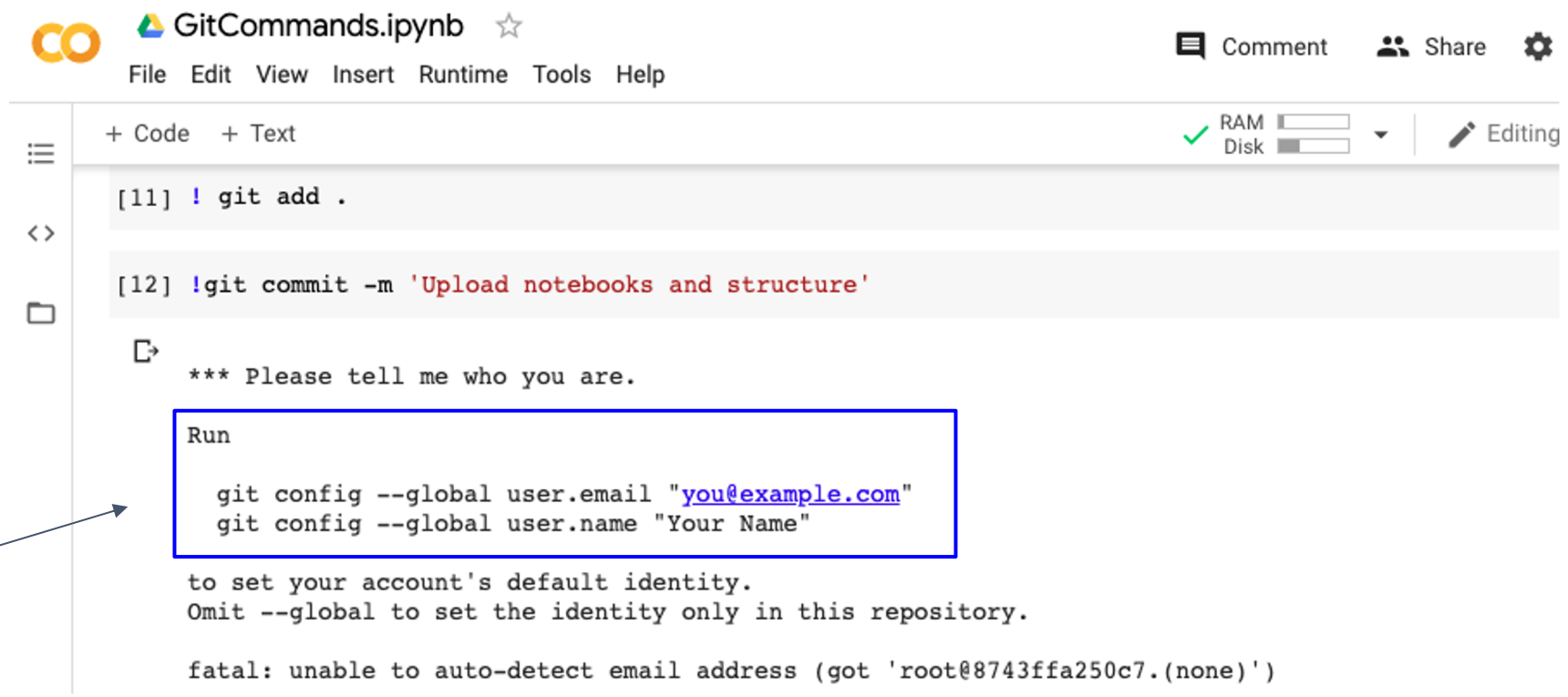
```
!ls
module1.ipynb
```

```
% cd ..
```

Interacting with GitHub via Google Colab

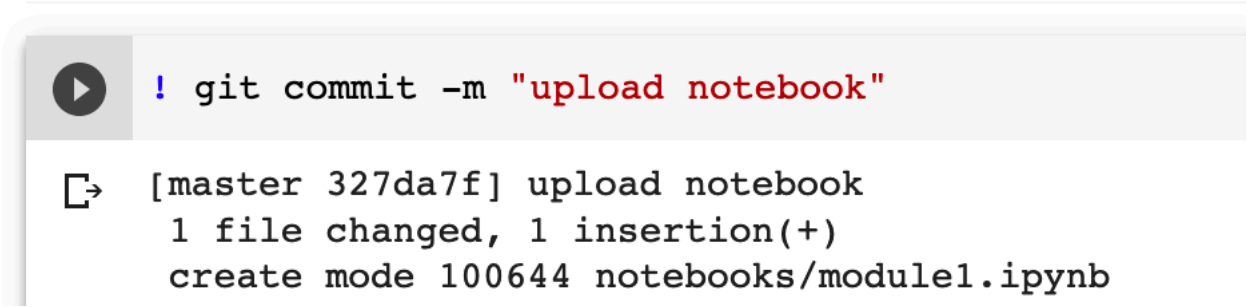
- Push all modifications to GitHub
  - Any other files generated can also be pushed

To commit, you may need to setup your GitHub username and email



Authorize the changes from local copy to your GitHub

```
!git config --global user.email "data.science.usc2@gmail.com"
```



Git commit

- git commit captures a snapshot of the project

A SNAPSHOT REFLECTS THE STATE OF THE FILES AT A CERTAIN INSTANT IN TIME

Git push

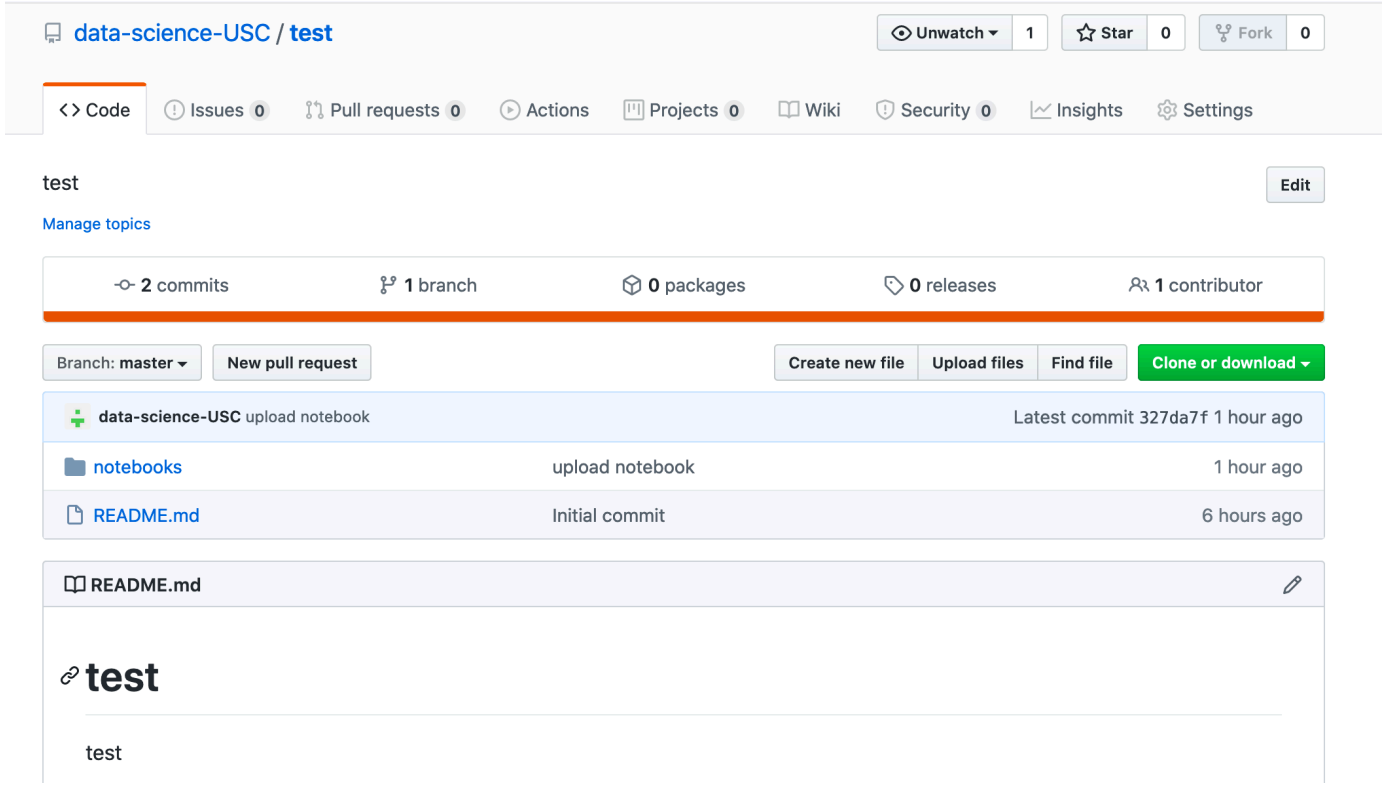
push any changes in your local repo to the repo on GitHub

If all local changes are based on the most up-to-date version of the GitHub repo, this will go smoothly

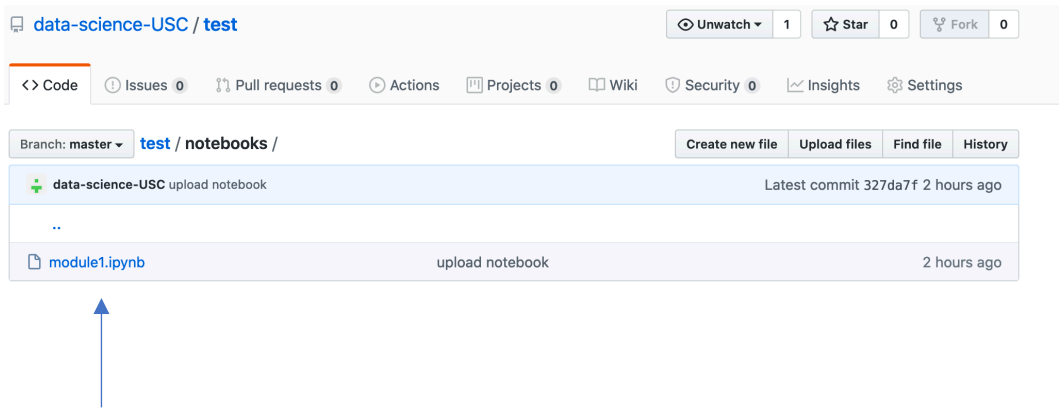
Push all modifications to GitHub


```
! git push 'https://data-science-USC:pasword123!@github.com/data-science-USC/test.git' --all

Counting objects: 4, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 4.38 KiB | 1.09 MiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://github.com/data-science-USC/test.git
   b83d696..327da7f  master -> master
```

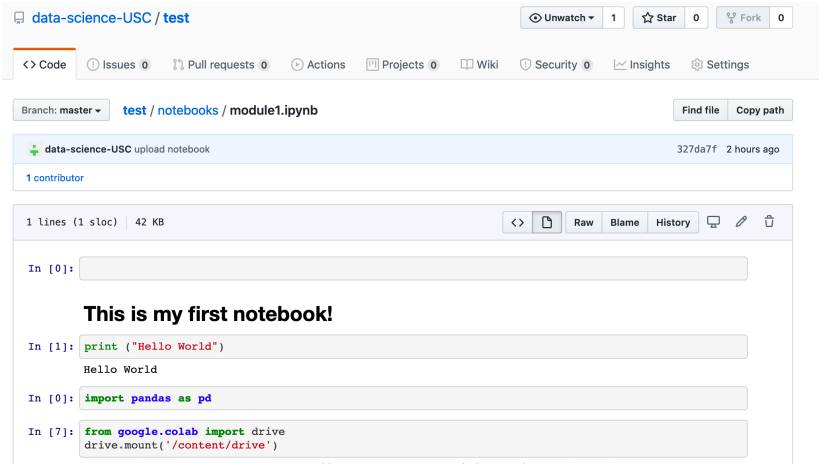
GitHub was updated



GitHub allows to effectively share notebooks



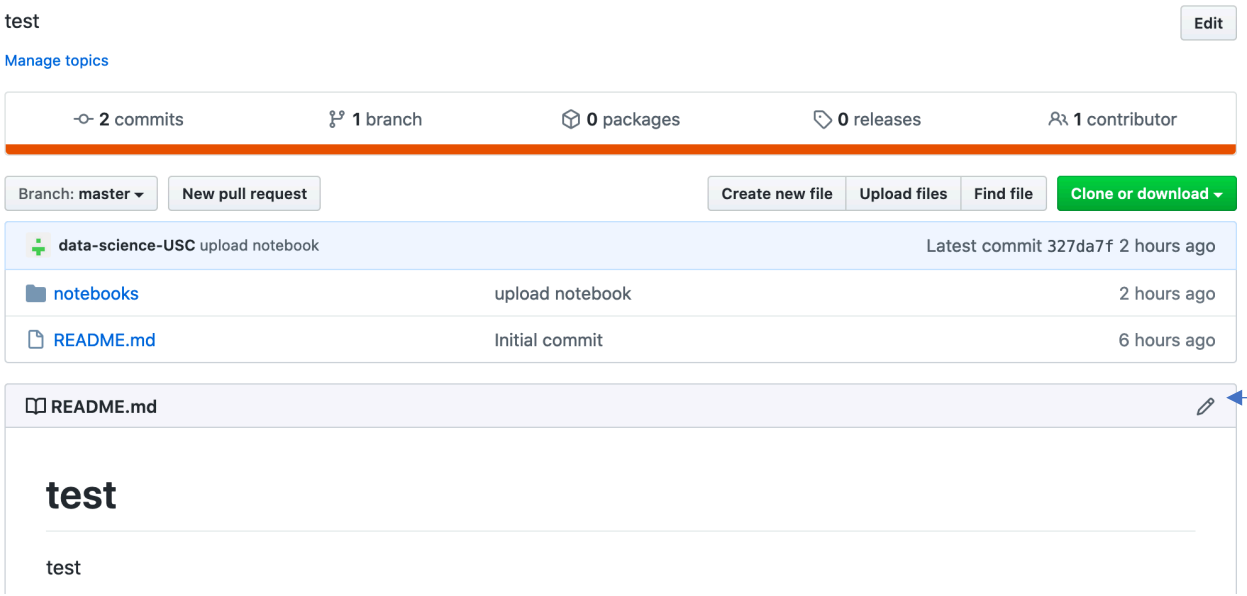
Click



Git pull

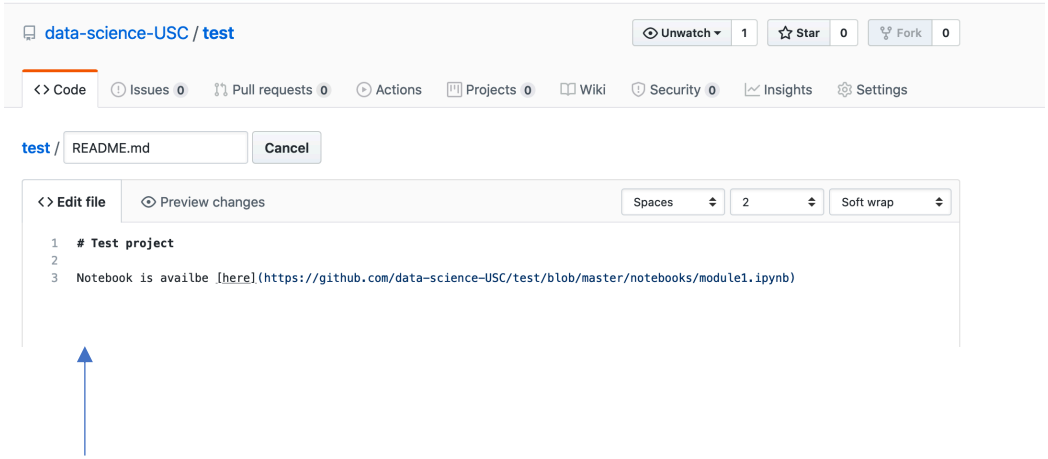
- obtains all updates to the repository on GitHub, and merge them with your local repo
- If there are changes in your local repo, this will potentially create merge conflicts

Pull changes made by others or you



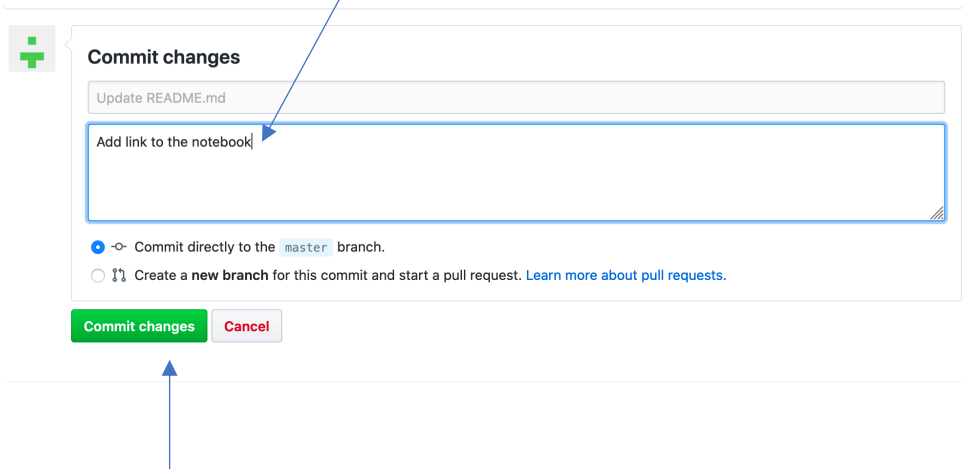
Click

Edit README



Markdown language

Add description of the commit



Click

More about Markdown https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet

Pull changes made by others or you

```
!git pull
```

```
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (6/6), done.
From https://github.com/data-science-USC/test
   b83d696..1682e1c  master     -> origin/master
Updating 327da7f..1682e1c
Fast-forward
 README.md | 5 +++--
 1 file changed, 3 insertions(+), 2 deletions(-)
```

Number of lines added

Number of lines removed

Check README.md in local repo

```
!cat README.md
```

```
# Test project

Notebook is available [here](https://github.com/data-science-USC/test/blob/master/notebooks/module1.ipynb)
```

Organizing your GitHub

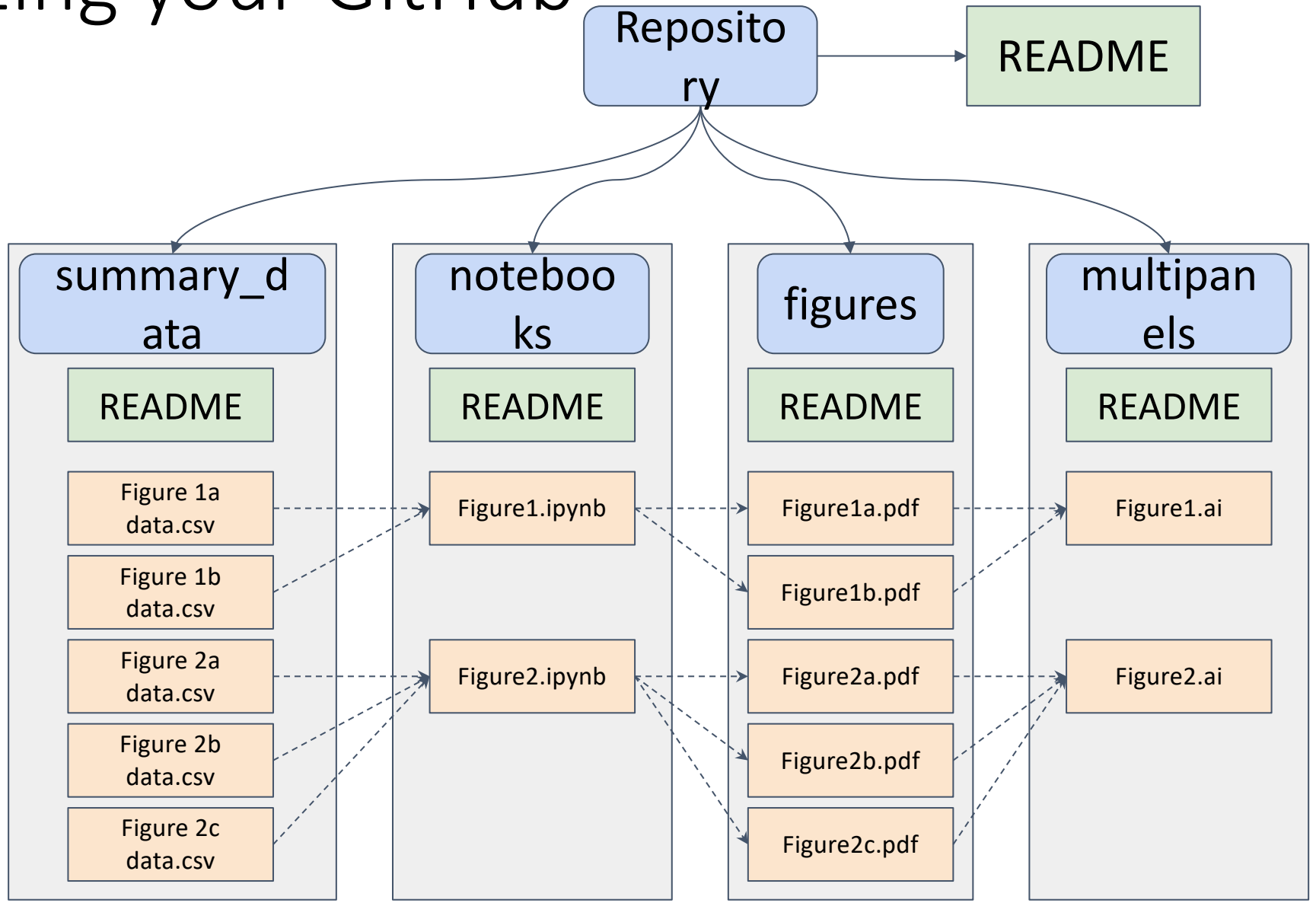
Use a organized structure to store files according to their role

- Create separate folders to store notebooks, summarized data, figures, scripts, etc
- Create README files inside each folder with detailed information about their content

Well organized repositories facilitate the research reproducibility

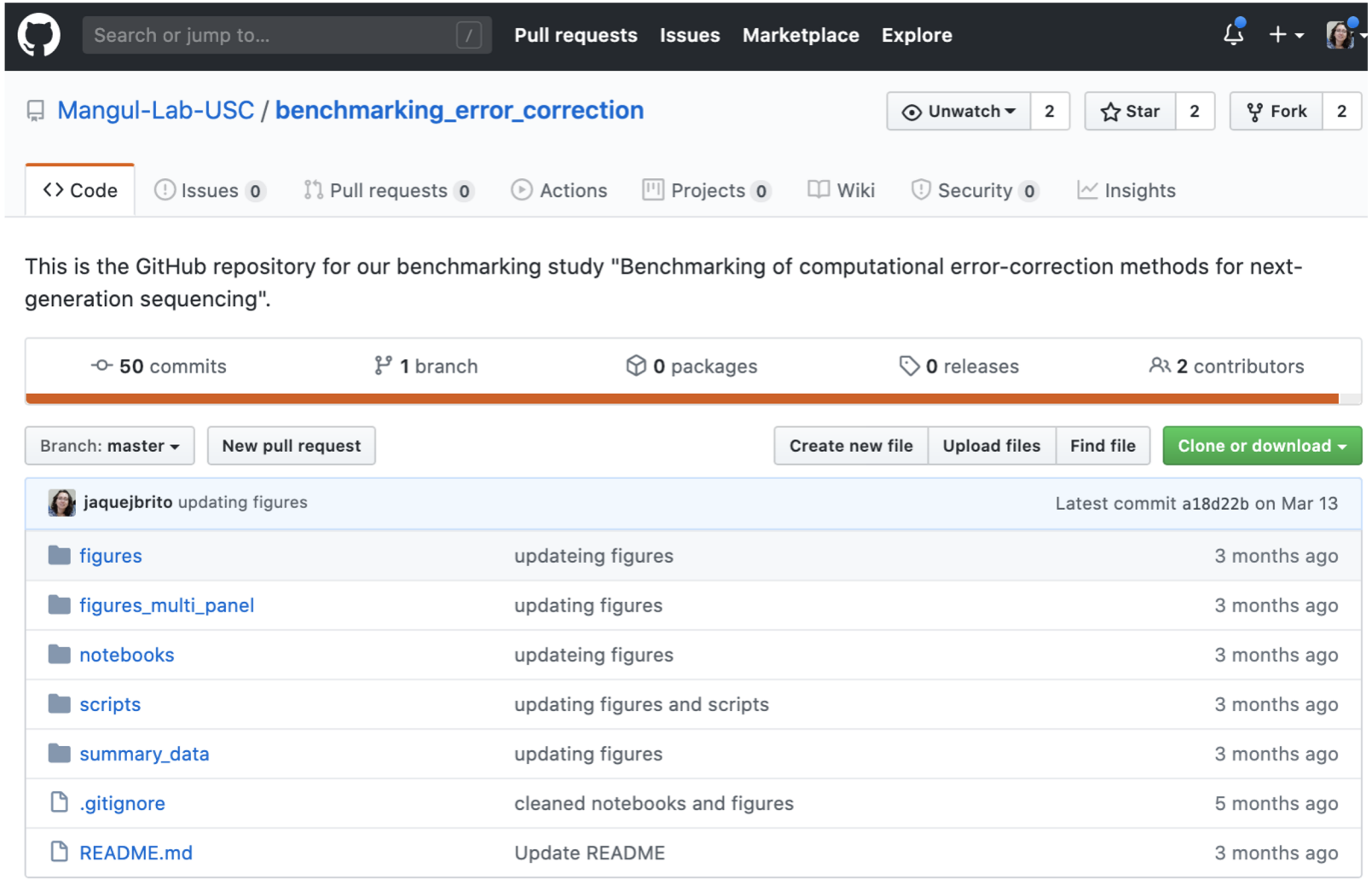
- It is easier to locate information and reuse the research material
- It is simpler for others to contribute to the shared material

Organizing your GitHub



Organizing your GitHub

- Separate folders
- README files inside each folder



Organizing your GitHub

- Example of README file at the root directory providing information about the repository content

README.md

Benchmarking of computational error-correction methods for next-generation sequencing data

Preprint online licence MIT

This project contains the links to the datasets and the code that was used for our study : "Benchmarking of computational error-correction methods for next-generation sequencing data"

Table of contents

How to cite this study

> Mitchell, Keith, et al. "Benchmarking of computational error-correction methods for next-generation sequencing data" bioRxiv, doi: https://doi.org/10.1101/642843

Reproducing results

Tools

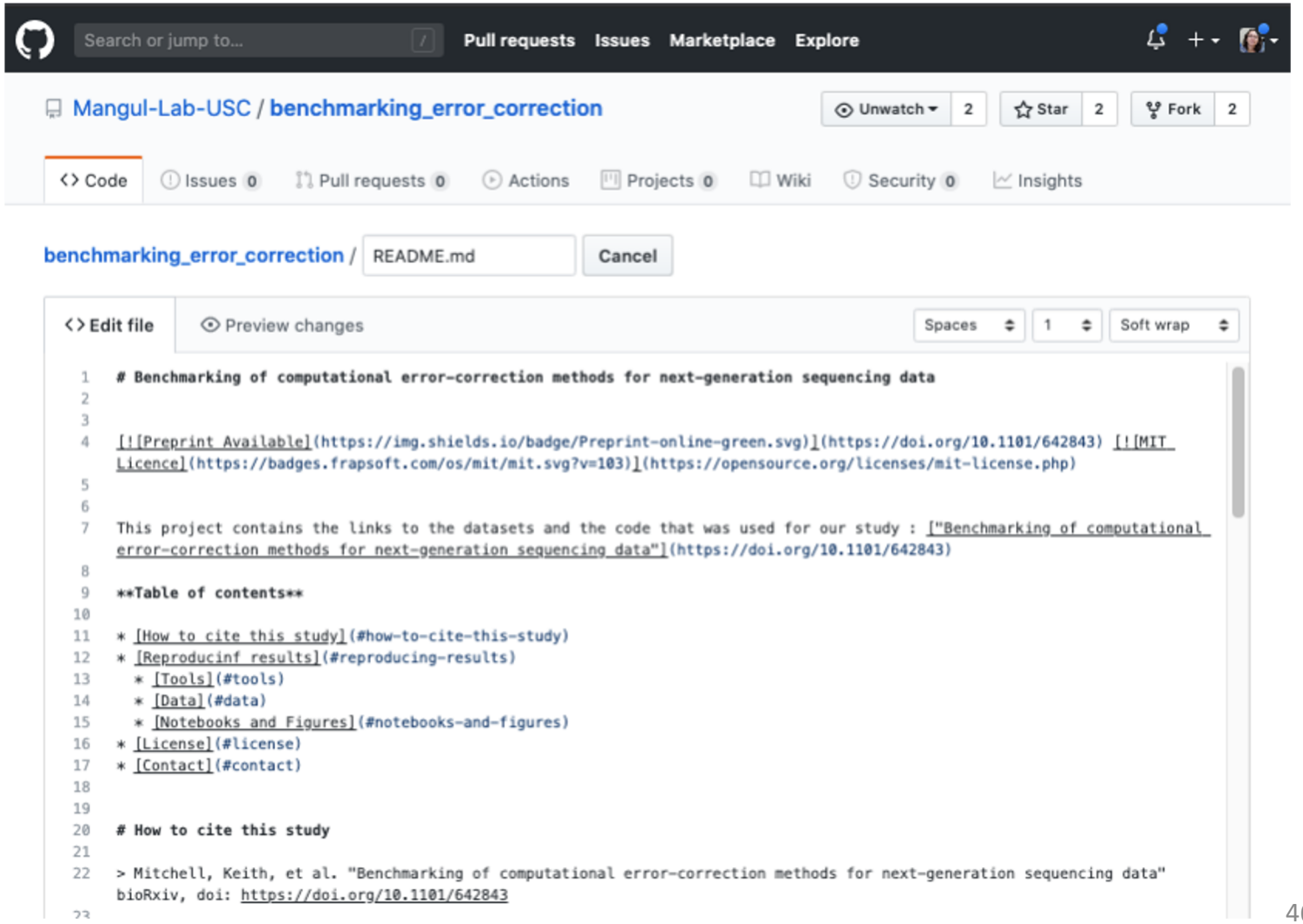
We have evaluated 10 error correction tools: BFC, Bless, Coral, Fiona, Lighter, Musket, Pollux, Reckoner, Racer and SGA. Details about the tools and instructions for running can be found in our "paper".

We have prepared "wrappers" in order to run each of the respective tools as well as create standardized log files.

https://github.com/Mangul-Lab-USC/benchmarking_error_correction

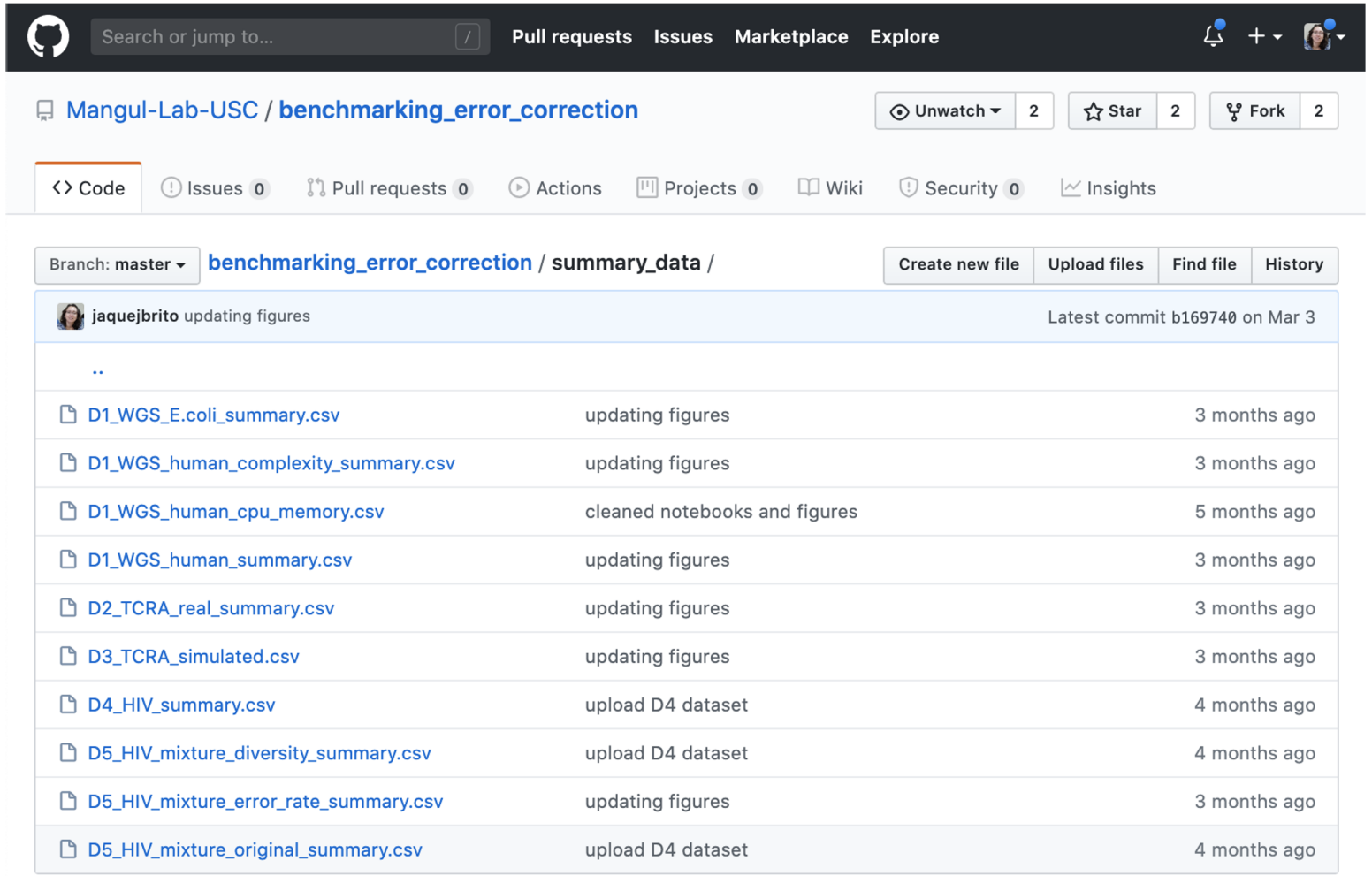
Organizing your GitHub

- The README file is written with markdown language



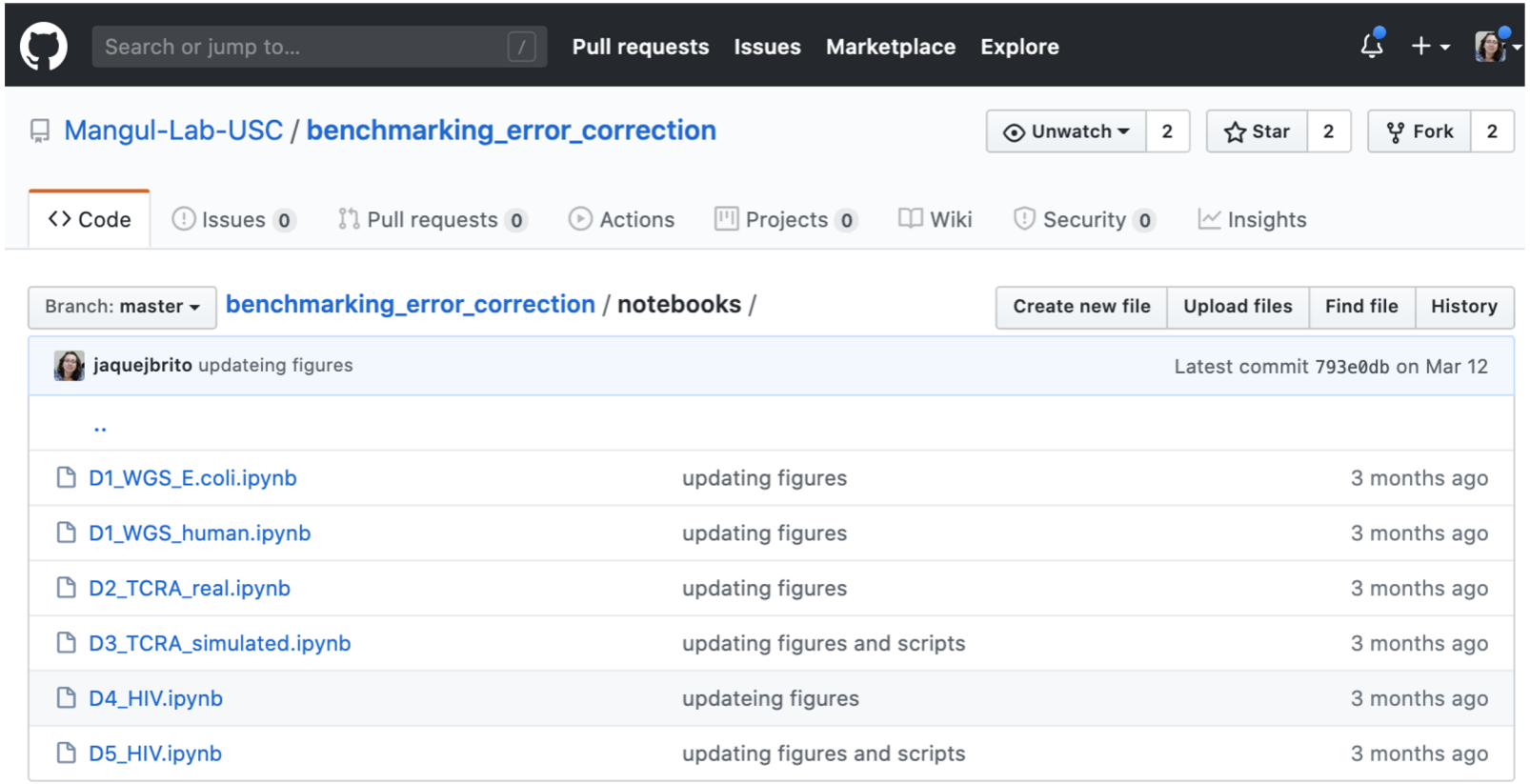
Organizing your GitHub

- The **summary_data** folder stores files that are input to notebooks
  - Generally csv files



Organizing your GitHub

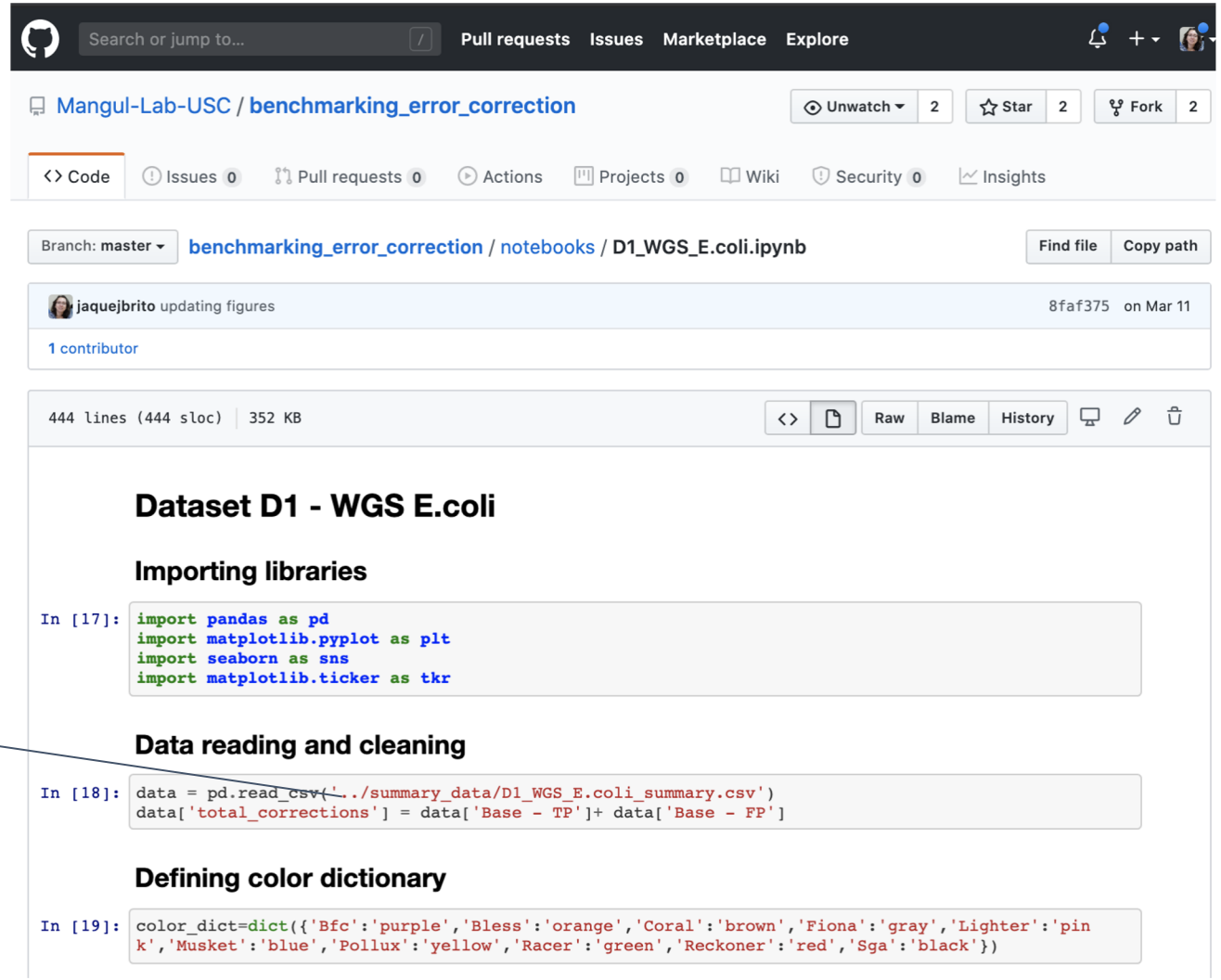
- The **notebook** folder stores notebooks that analyse summarized data and generates figures



Organizing your GitHub

- **Notebooks** can be visualized at the GitHub website

Reading summarized data from summary_data folder



Organizing your GitHub

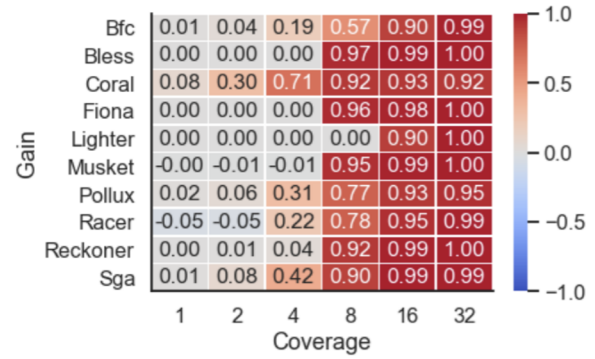
- **Notebooks** generates images that are stored at the figures folder

Saving images in the figures folder

Figure 2g

Heatmap depicting the gain across various coverage settings.
Each row corresponds to an error correction tool, and each column corresponds to a dataset with a given coverage.
For each tool, the best k-mer size was selected.

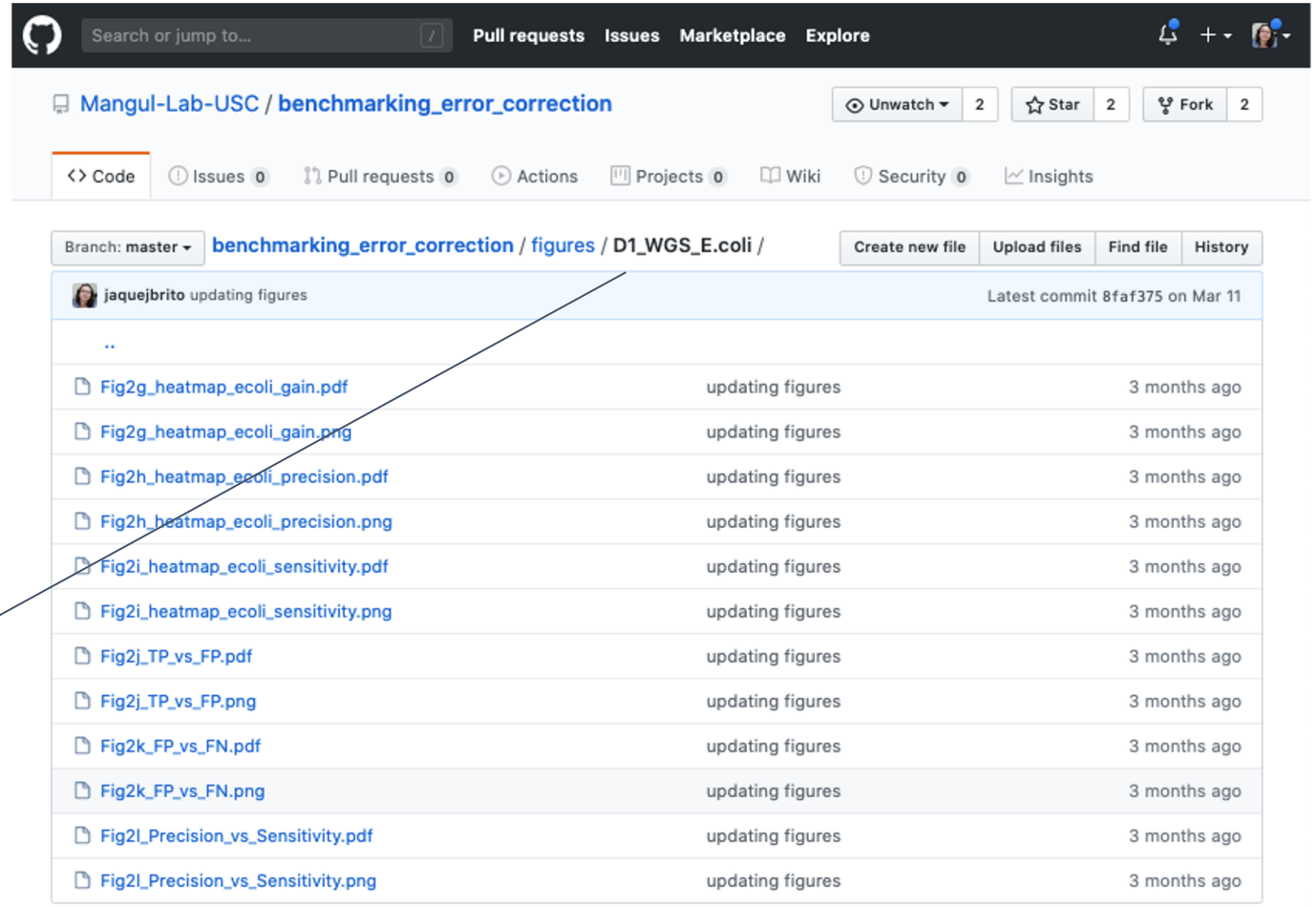
In [21]:
```
result = data_best.pivot(index='Tool', columns='Coverage', values='Base Gain') \
    .sort_values(by="Tool", ascending=False)

sns.set_style("white")
sns.set_context("talk")

g=sns.heatmap(result,
              annot=True,
              cmap='coolwarm',
              center=0,
              linewidths=.5,
              annot_kws={'size':17},
              fmt=".2f",
              vmin=-1,
              vmax=1)

colorbar = g.collections[0].colorbar
colorbar.set_ticks([-1.0, -0.5, 0, 0.5, 1.0])

g.set(xlabel='Coverage', ylabel='Gain')
g.set_ylim(0, 10)


sns.despine()

plt.savefig("../figures/D1_WGS_E.coli/Fig2g_heatmap_ecoli_gain.png", bbox_inches="tight")
plt.savefig("../figures/D1_WGS_E.coli/Fig2g_heatmap_ecoli_gain.pdf", bbox_inches="tight")
```



Organizing your GitHub

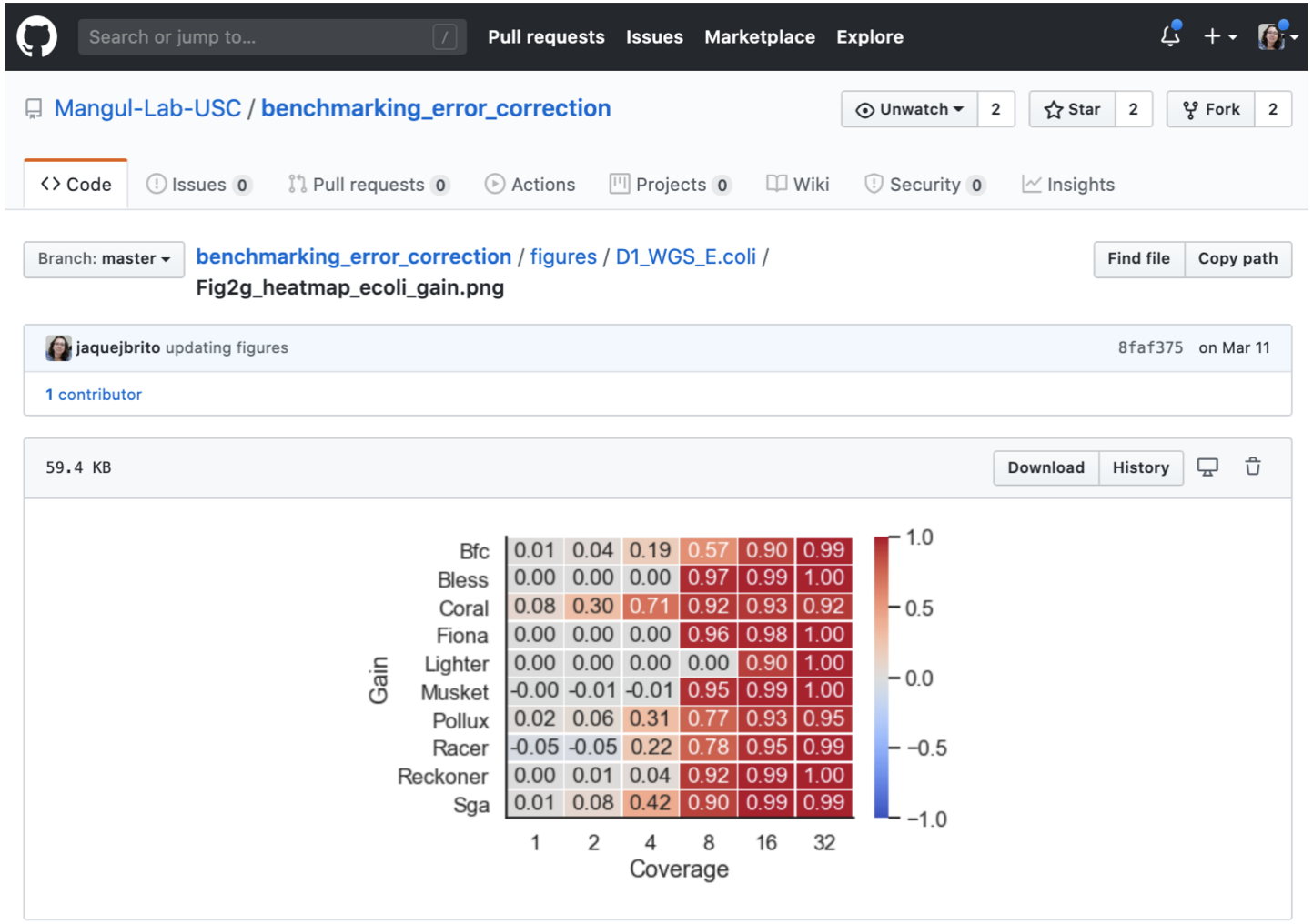
- The **figures** folder stores images of plots generated by the notebooks

Subfolder to improve the organization



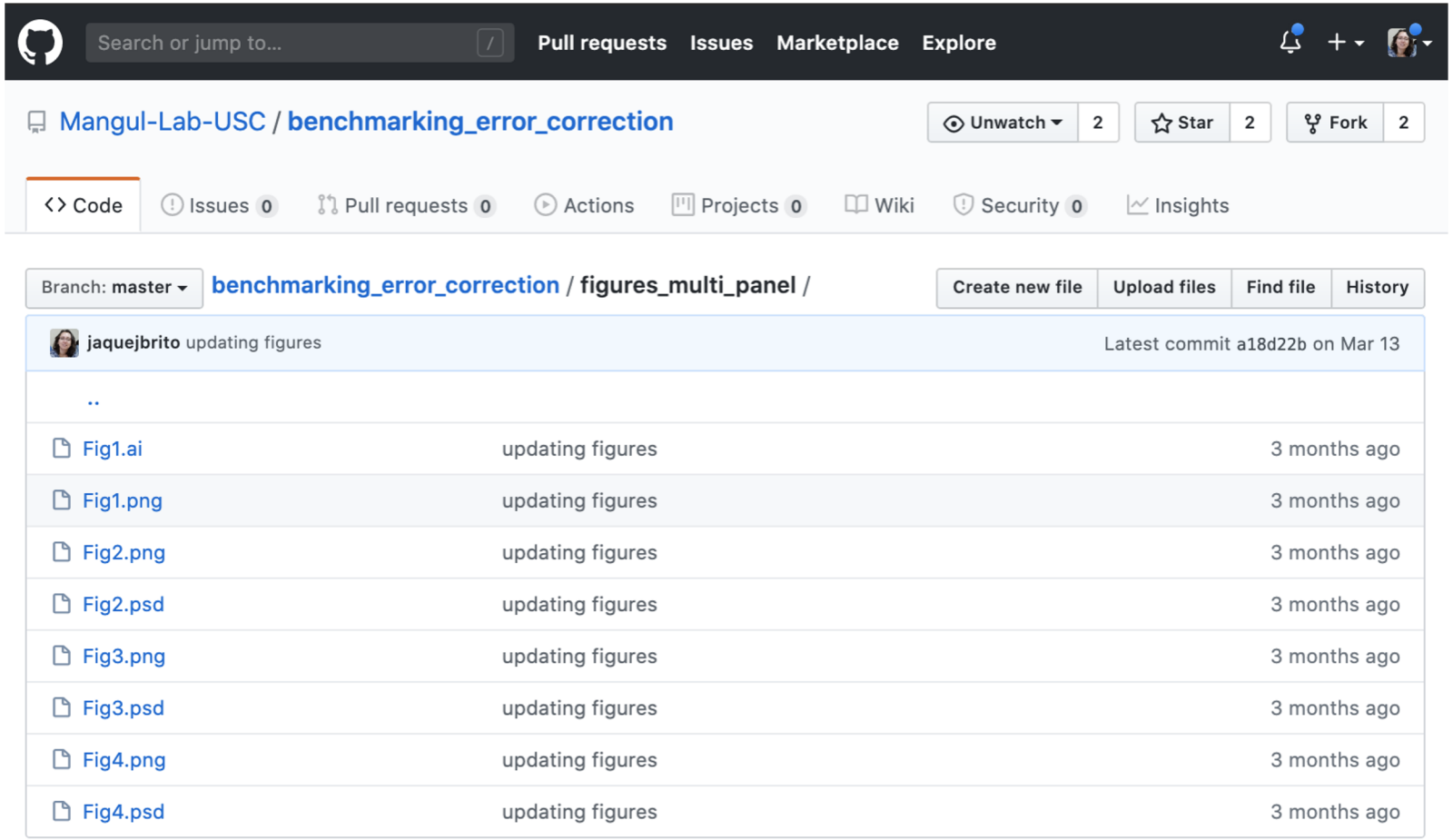
Organizing your GitHub

- Images can also be visualized at the GitHub website



Organizing your GitHub

- The **figure_multi_panel** folder stores source files of panels that combines multiple figures
    - Files generated by image editing software such as Photoshop, Inkscape, Gimp, Illustrator, etc



How to deal with extremely large files?

Raw data are generally large

- Usually gigabytes of size

GitHub has a file size limit of 100MB

- https://help.github.com/en/github/managing-large-files/what-is-my-disk-quota

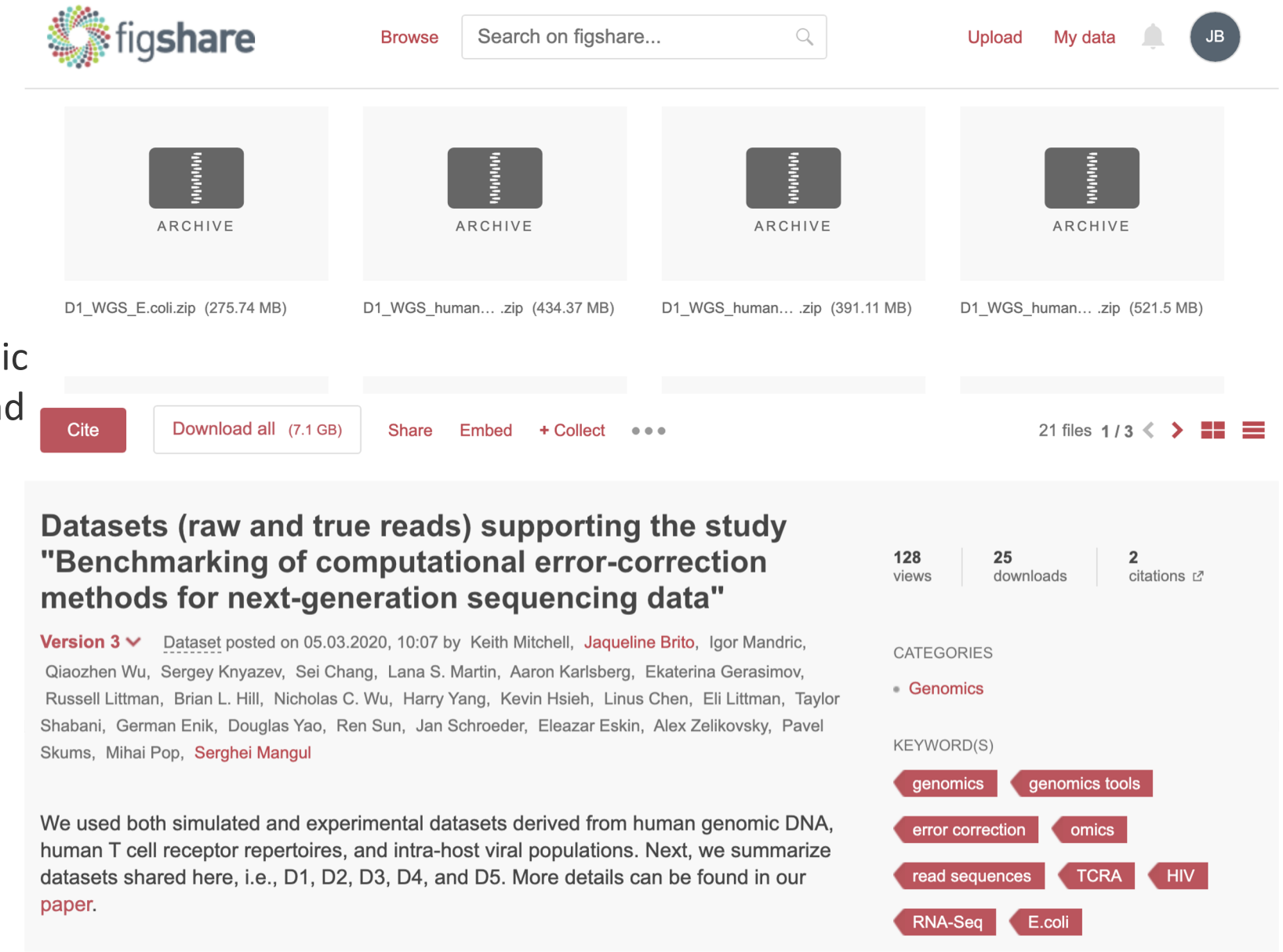
Large files can not be stored at GitHub

- Large files should be shared at external archival platforms
  - FigShare and Zenodo

FigShare

"Figshare is a web-based interface designed for academic research data management and research data dissemination."

https://knowledge.figshare.com



Essential git commands

Git clone creates a copy of an existing remote repository

Git add puts a file into the repo for the first time, i.e. begin tracking it with Version Control

Git commit captures a snapshot of the project

Git push pushes any changes in your local repo to the repo on GitHub

Git pull obtains all updates to the repository on GitHub, and merge them with your local repo

Additional resources

https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004668