

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



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), CNASEq⁹¹ (v.0.0.6), APOLLOH⁹² (v.0.1.1), SAMtools⁹³ (v.0.1.17), MutationSeq⁹⁴ (v.1.0.2 and v.4.3.5), Strelka⁹⁵ (v.1.0.6), SNPEff⁹⁶ (v.3.2 for somatic and v.4.1 for germline), ABySS⁹⁷ (v.1.3.4), TransABySS^{97,98} (v.1.4.10), Chimerascan⁹⁹ (v.0.4.5), DeFuse¹⁰⁰ (v.0.6.2), Manta¹⁰¹ (v.1.0.0), Delly¹⁰² (v.0.7.3), MAVIS⁷¹ (v.2.1.1), STAR⁷³ (v.2.5.2b), RSEM⁷⁴ (v.1.3.0), MSIsensor⁷⁶ (v.0.2), HRDtools¹⁶ (v.0.0.0.9), BioBloomTools⁷⁸ (v.2.0.11b), EXPANDS⁸⁴ (v.2.1.1), SignIT (<https://github.com/eyzhao/SignIT>), samtools⁹³ (v.0.1.17), ClinVar¹⁰³ (v.20180905), InterVar⁸⁸, ControlFREEC¹⁰⁴ (v.5), CIBERSORT⁸⁹ (v.1.04), Jaguar¹⁰⁵ (v.2.0.3), MiXCR⁹⁰ (Java, v.2.1.2) and VDJtools¹⁰⁶ (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-learn⁸⁶ (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

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



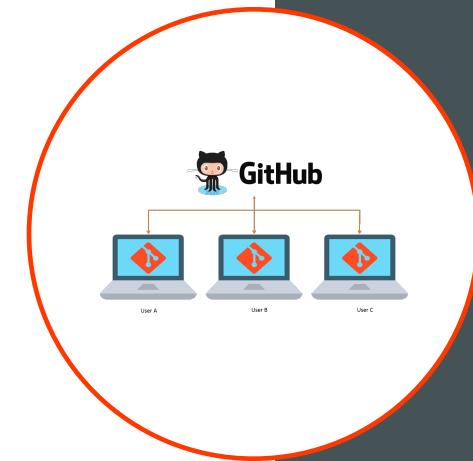
COLLECTION OF ALL
FILES/FOLDERS AND THEIR
HISTORY (ALL SNAPSHOTS)



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



- | | |
|--|---|
| <ol style="list-style-type: none">1. It is a software2. It is installed locally on the system3. It is a command line tool4. It is a tool to manage different versions of edits, made to files5. It provides functionalities like version control and data management | <ol style="list-style-type: none">1. It is a service2. It is hosted on web3. It provides a graphical interface4. It is a space to upload a copy of the git repository5. It provides functionalities of git as well as adding few of its own features |
|--|---|

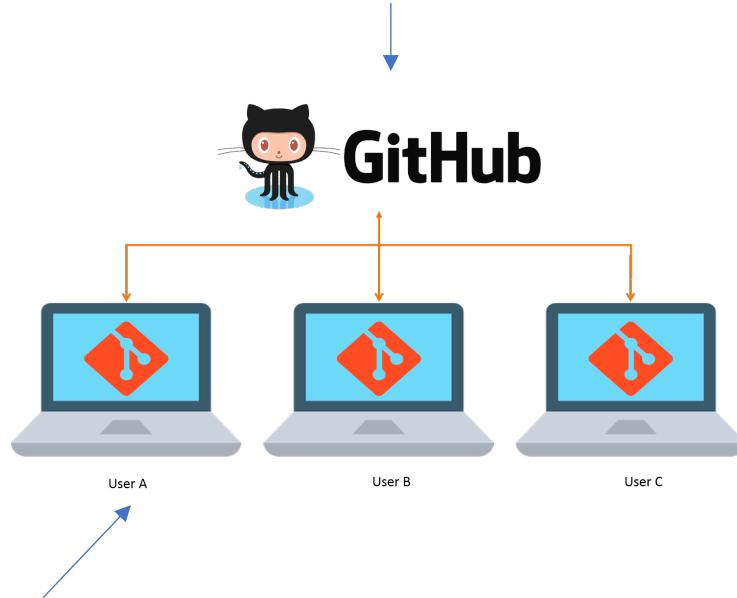
Useful features of version control

- **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/>

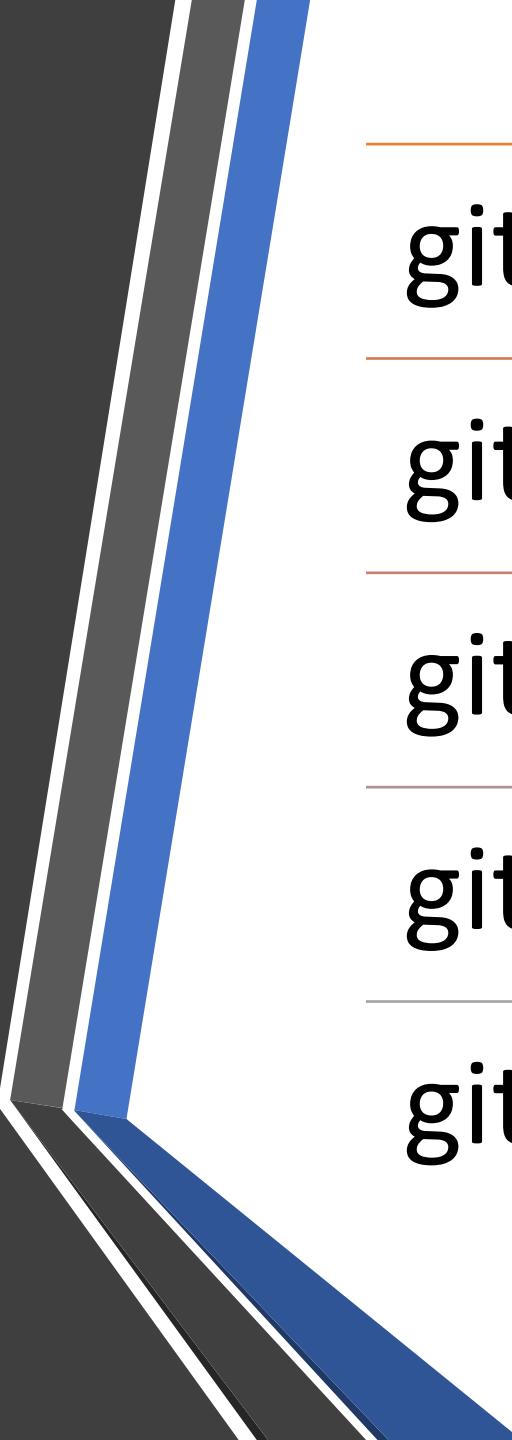
GitHub

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/>

Create your account

Verify your account

Please solve this puzzle so we know you are a real person

[Verify](#)



Email preferences

Send me occasional product updates, announcements, and offers.

[Join a free plan](#)



Please verify your email address

Before you can contribute on GitHub, we need you to verify your email address.

An email containing verification instructions was sent to data.science.usc2@gmail.com.

[Resend verification email](#)

[Change your email settings](#)

[GitHub] Please verify your email address.



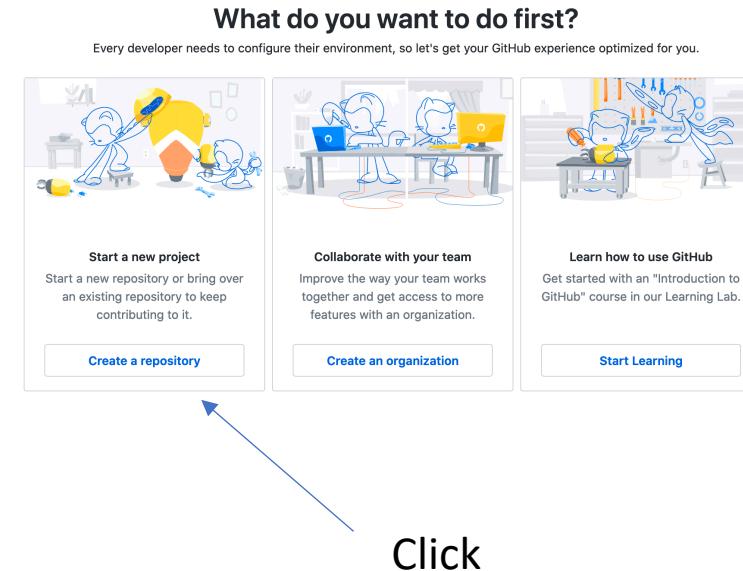
Almost done, **@data-science-USC!** To complete your GitHub sign up, we just need to verify your email address:
data.science.usc2@gmail.com.

Verify email address

Click

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

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Owner / Repository name *

Great repository names are short and memorable. Need inspiration? How about [super-bassoon](#)?

Description (optional)

Public Anyone on the internet can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

Initialize this repository with a README
This will let you immediately clone the repository to your computer.

Add .gitignore: Add a license:

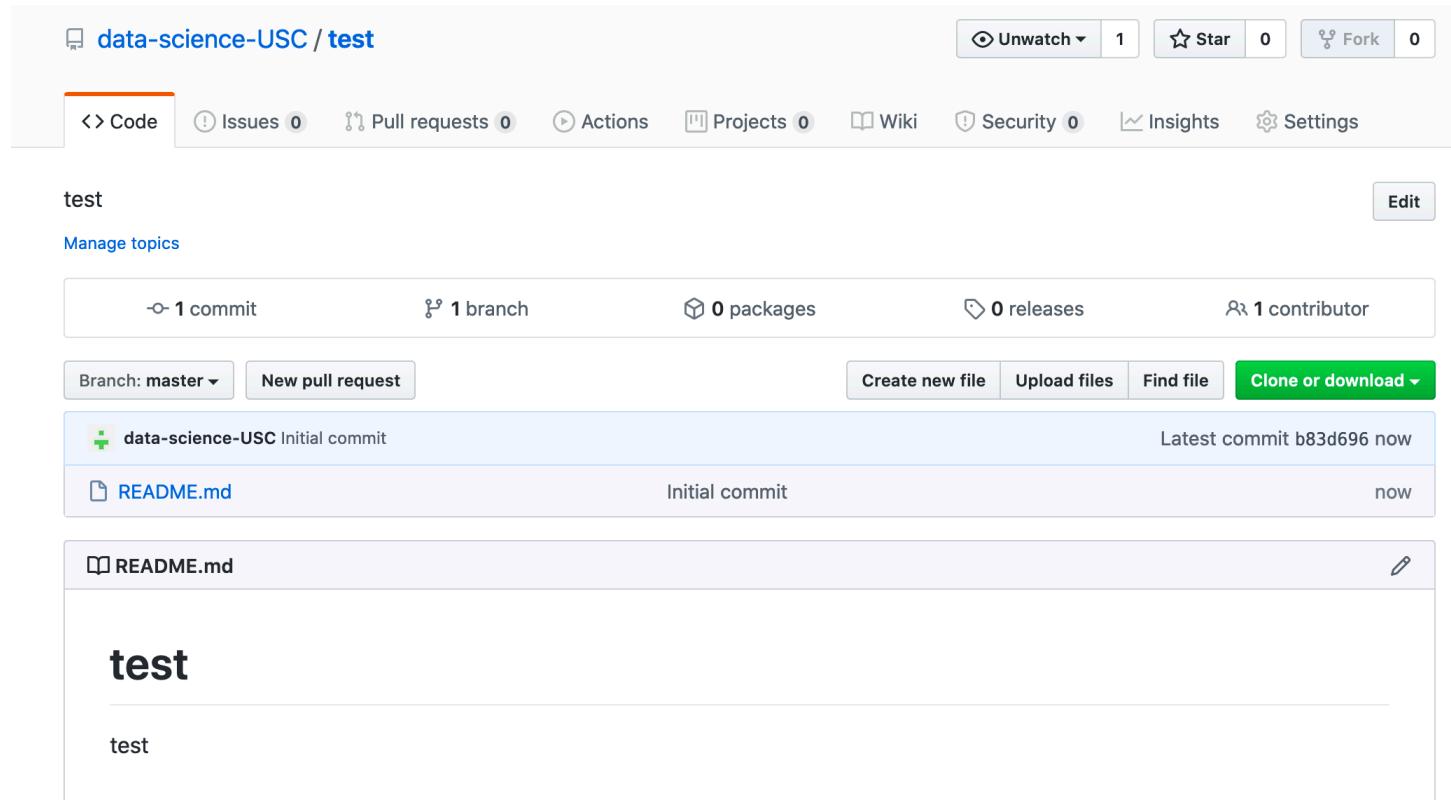
Type

Type

Click

The diagram illustrates the steps required to create a GitHub repository. It shows a screenshot of the GitHub 'Create a new repository' form. Three arrows point from the word 'Type' to different fields: one arrow points from the first 'Type' label to the 'Repository name' input field containing 'test'; another arrow points from the second 'Type' label to the 'Description (optional)' input field containing 'test'; and a third arrow points from the 'Click' label to the checkbox for initializing the repository with a README, which is checked.

Your first GitHub repo



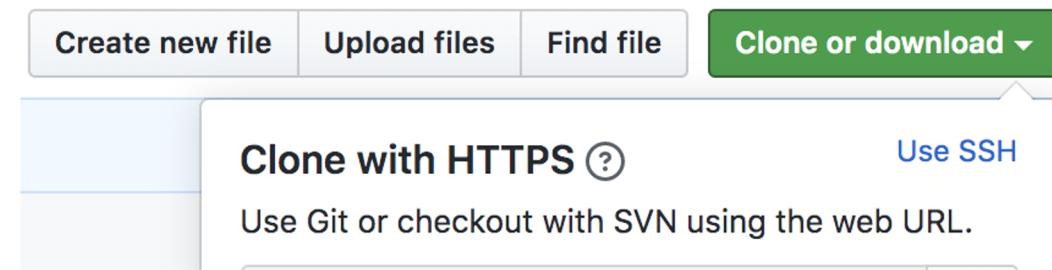
Git clone

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

The screenshot shows a Google Colab notebook interface. The title bar reads "CO GitCommands.ipynb". The menu bar includes File, Edit, View, Insert, Runtime, Tools, Help, and a status message "All changes saved". The toolbar has "Comment", "Share", and "Settings" buttons, along with RAM and Disk status indicators. The code cell contains the following Python code:

```
from google.colab import drive
drive.mount('/content/gdrive')
```

Below the code cell, there is a message: "... Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=9473189". A text input field is present with the placeholder "Enter your authorization code:".

Mounting your Google Drive

- Google Drive is now mounted at /content/gdrive

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** CO GitCommands.ipynb
- Menu Bar:** File Edit View Insert Runtime Tools Help
- Toolbar:** Comment Share Settings
- Code Cell:** + Code + Text
- Runtime Status:** RAM Disk ✓ Editing
- Cell Content:**

```
▶ from google.colab import drive
drive.mount('/content/gdrive')

↳ Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client\_id=9473189

Enter your authorization code:
.....
Mounted at /content/gdrive
```
- Control Buttons:** Up, Down, Run, Stop, Share, Settings, Delete, More

Interacting with GitHub via Google Colab

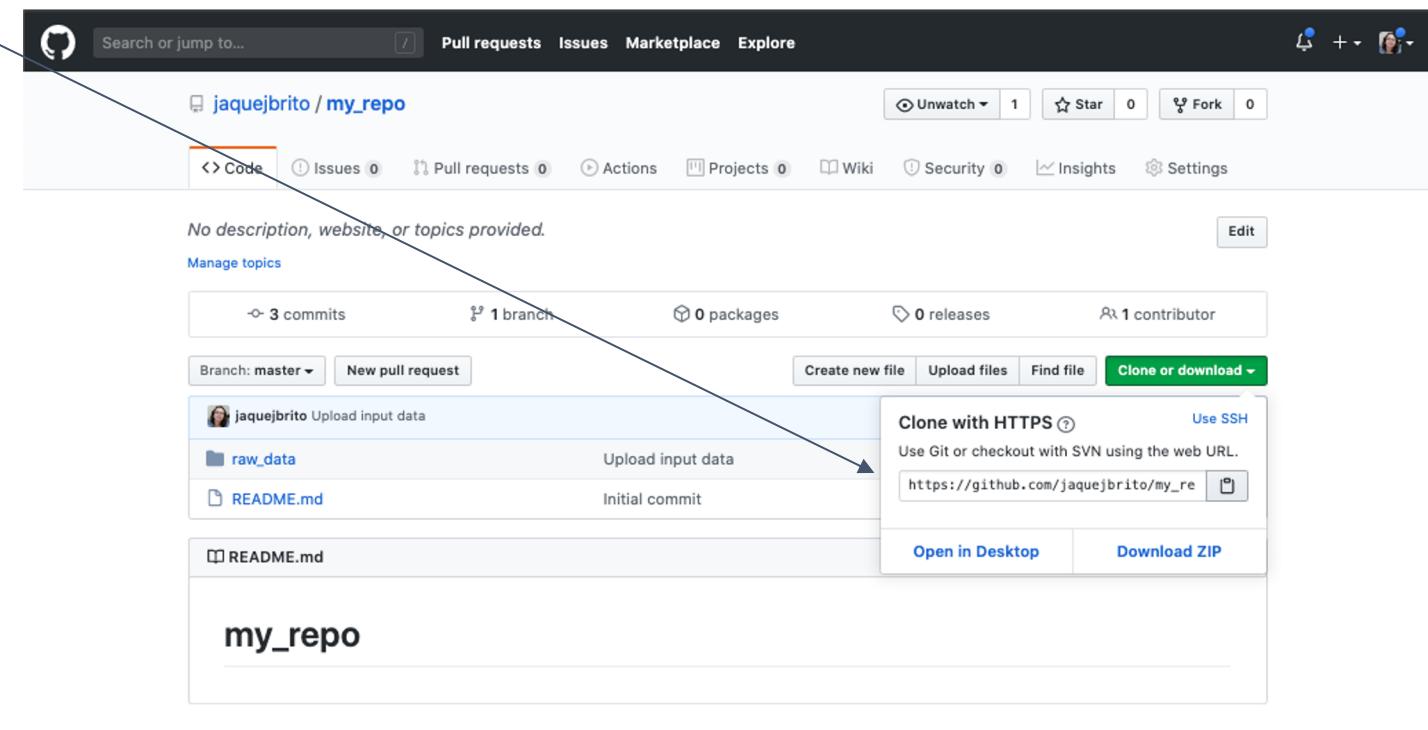
- Mounting your Google Drive
 - Access '/content/drive/My Drive' to clone the repo

```
[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
```

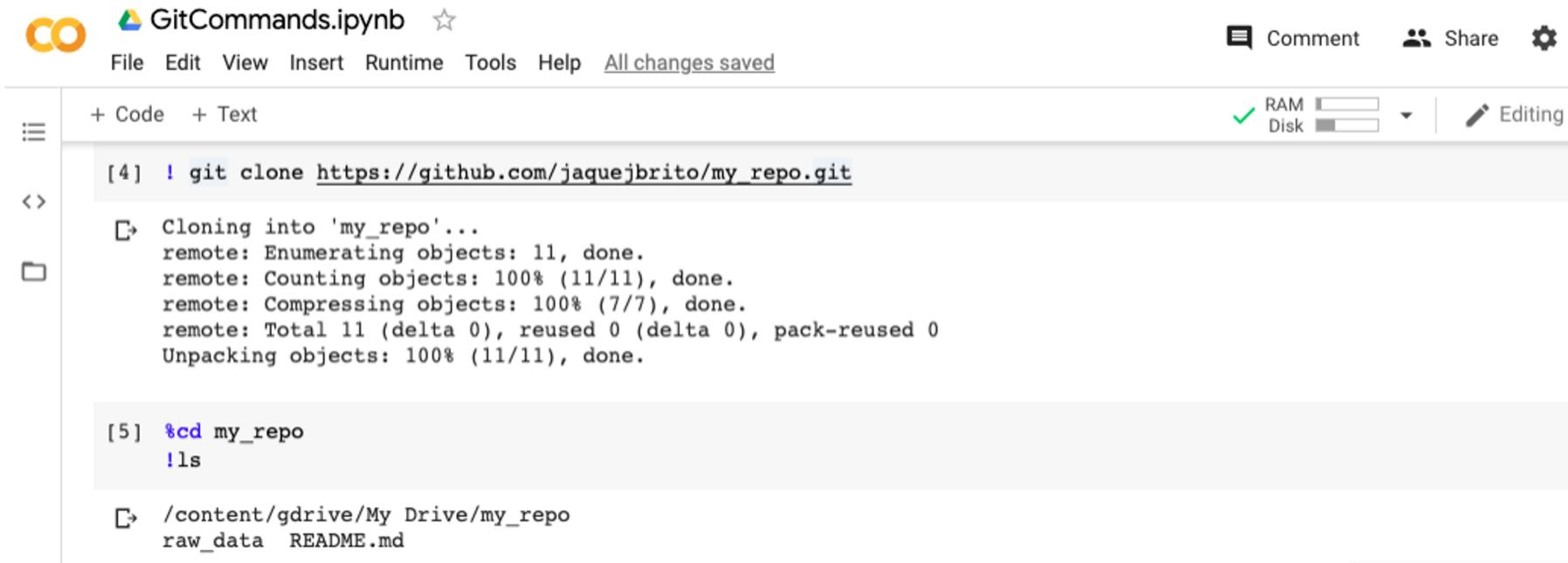
Interacting with GitHub via Google Colab

- 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'



The screenshot shows a Google Colab interface with a Jupyter notebook titled "GitCommands.ipynb". The notebook has a yellow "CO" icon and a star icon. The menu bar includes File, Edit, View, Insert, Runtime, Tools, Help, and a status message "All changes saved". The toolbar on the right shows a green checkmark next to "RAM" and "Disk", and an "Editing" button. The code editor contains two cells:

```
[4] ! git clone https://github.com/jaquejbrito/my_repo.git
[4]: Cloning into 'my_repo'...
      remote: Enumerating objects: 11, done.
      remote: Counting objects: 100% (11/11), done.
      remote: Compressing objects: 100% (7/7), done.
      remote: Total 11 (delta 0), reused 0 (delta 0), pack-reused 0
      Unpacking objects: 100% (11/11), done.

[5] %cd my_repo
[5]: !ls
[5]: /content/gdrive/My Drive/my_repo
      raw_data README.md
```

Create a directory in GitHub repository

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

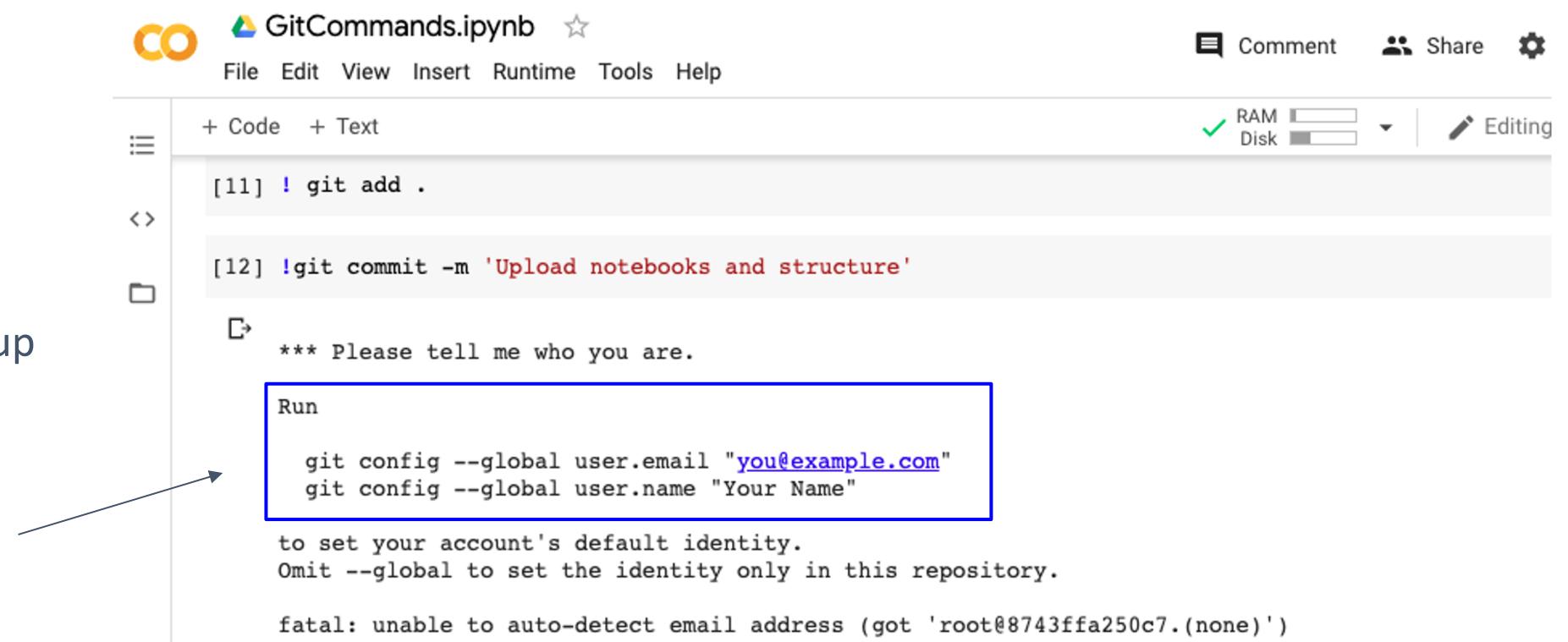
```
!ls  
module1.ipynb
```

```
% 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



The screenshot shows a Google Colab notebook titled "GitCommands.ipynb". The terminal window displays the following commands:

```
[11] !git add .
[12] !git commit -m 'Upload notebooks and structure'
[13] 
*** Please tell me who you are.
```

A blue box highlights the command:

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

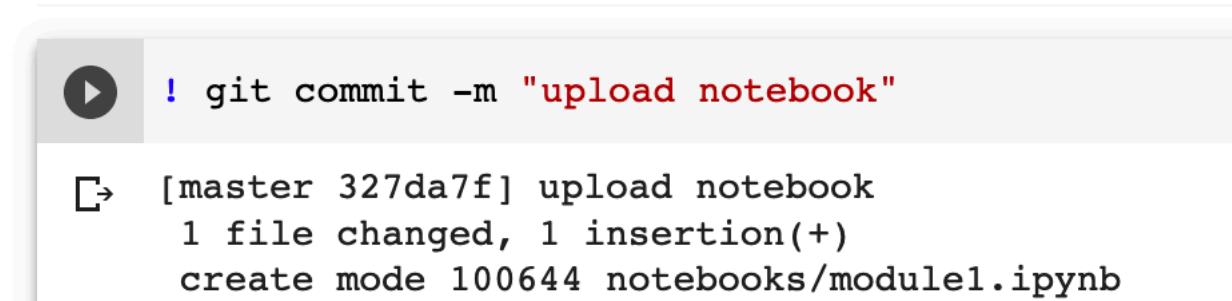
An arrow points from the text "To commit, you may need to setup your GitHub username and email" to the highlighted command in the terminal.

At the bottom of the terminal output, there is an error message:

```
fatal: unable to auto-detect email address (got 'root@8743ffa250c7.(none)')
```

Authorize the changes from local copy to your GitHub

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



A terminal window showing a command and its output. The command is `! git commit -m "upload notebook"`. The output shows a commit message `[master 327da7f] upload notebook`, `1 file changed, 1 insertion(+)`, and `create mode 100644 notebooks/module1.ipynb`.

```
! git commit -m "upload notebook"
[master 327da7f] upload notebook
1 file changed, 1 insertion(+)
create mode 100644 notebooks/module1.ipynb
```

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

```
Your password  
Your username  
! git push 'https://data-science-USC:password123!@github.com/data-science-USC/test.git' --all  
Your username  
Name of repo
```

The diagram illustrates the components of a Git push command. It shows the command `! git push 'https://data-science-USC:password123!@github.com/data-science-USC/test.git' --all` with annotations pointing to specific parts:

- Your password**: Points to the password `password123!` in the URL.
- Your username**: Points to both the source username `data-science-USC` and the destination repository name `test`.
- Name of repo**: Points to the repository name `test.git` at the end of the URL.

```
! git push 'https://data-science-USC:password123!@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

data-science-USC / test

Code Issues 0 Pull requests 0 Actions Projects 0 Wiki Security 0 Insights Settings

test

Manage topics

2 commits 1 branch 0 packages 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

data-science-USC upload notebook Latest commit 327da7f 1 hour ago

notebooks upload notebook 1 hour ago

README.md Initial commit 6 hours ago

README.md

test

test

The screenshot shows a GitHub repository named 'test' under the organization 'data-science-USC'. The repository has 2 commits, 1 branch, 0 packages, 0 releases, and 1 contributor. The latest commit is from 1 hour ago. The repository contains files 'notebooks' and 'README.md'. The README.md file has an 'Initial commit' message and was made 6 hours ago. There is also a note about an 'upload notebook' commit from 1 hour ago.

GitHub allows to effectively share notebooks

data-science-USC / test

Branch: master test / notebooks /

+ data-science-USC upload notebook

module1.ipynb upload notebook 2 hours ago

Click

data-science-USC / test

Branch: master test / notebooks / module1.ipynb

data-science-USC upload notebook 327da7f 2 hours ago

1 contributor

1 lines (1 sloc) 42 KB

In [0]:

This is my first notebook!

In [1]: print ("Hello World")
Hello World

In [0]: import pandas as pd

In [7]: from google.colab import drive
drive.mount('/content/drive')

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

test Edit

Manage topics

-o 2 commits ↗ 1 branch ⚡ 0 packages 🎁 0 releases 🌐 1 contributor

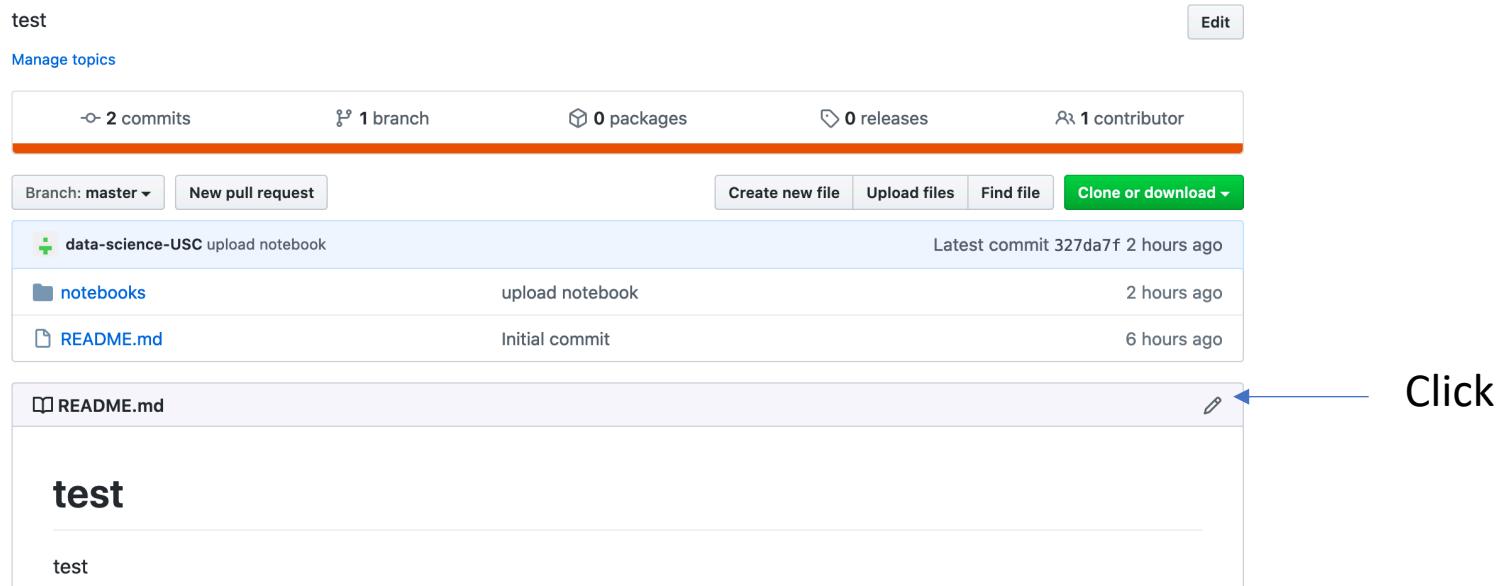
Branch: master ▾ New pull request Create new file Upload files Find file Clone or download ▾

		Latest commit 327da7f 2 hours ago
+	data-science-USC upload notebook	
📁	notebooks	upload notebook
📄	README.md	Initial commit

📄 README.md edit Click

test

test



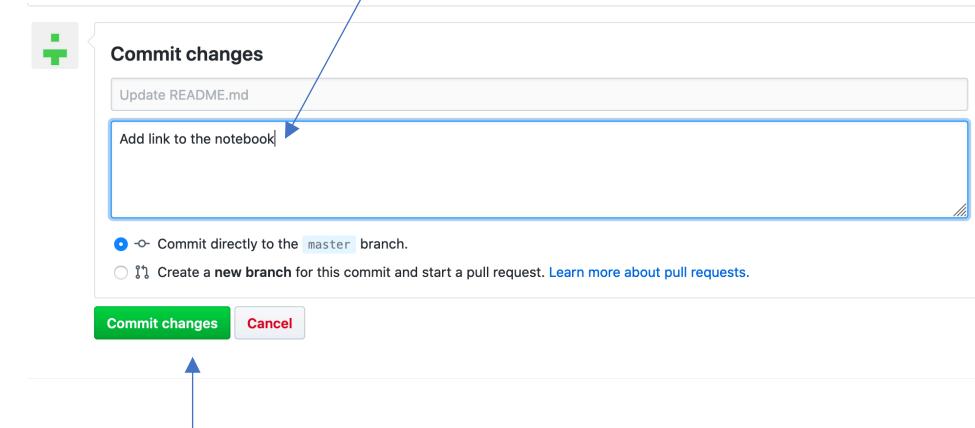
Edit README

A screenshot of a GitHub repository page for 'data-science-USC / test'. The 'Code' tab is selected. Below it, the 'README.md' file is open. The content of the file is:

```
1 # Test project
2
3 Notebook is available [here] (https://github.com/data-science-USC/test/blob/master/notebooks/module1.ipynb)
```

Below the code editor, there is a vertical blue arrow pointing upwards.

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

Number of lines added

```
!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 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

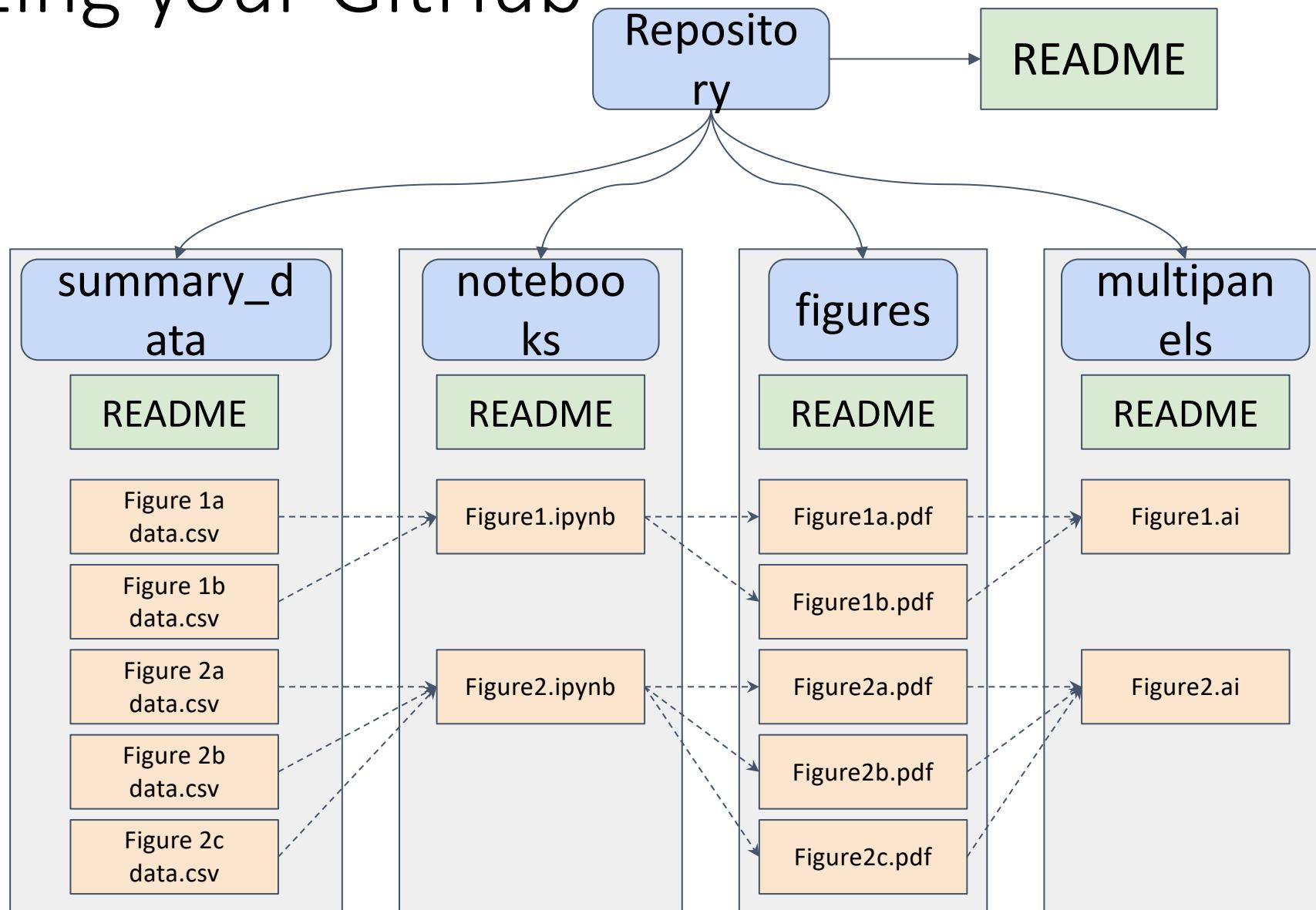
Use a organized structure to store files according to their role

- 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

The screenshot shows a GitHub repository page. At the top, there's a navigation bar with icons for search, pull requests, issues, marketplace, and explore. Below the bar, the repository name 'Mangul-Lab-USC / benchmarking_error_correction' is displayed, along with buttons for unwatching, starring, and forks. A horizontal menu below the repository name includes 'Code' (which is selected), 'Issues 0', 'Pull requests 0', 'Actions', 'Projects 0', 'Wiki', 'Security 0', and 'Insights'. The main content area contains a brief description: 'This is the GitHub repository for our benchmarking study "Benchmarking of computational error-correction methods for next-generation sequencing".' Below this, summary statistics are shown: 50 commits, 1 branch, 0 packages, 0 releases, and 2 contributors. A 'Branch: master' dropdown and a 'New pull request' button are also present. At the bottom, a list of recent commits is shown, each with a user icon, author, message, and timestamp.

Commit	Author	Message	Time
figures	jaquejbrito	updateing figures	3 months ago
figures_multi_panel	jaquejbrito	updating figures	3 months ago
notebooks	jaquejbrito	updateing figures	3 months ago
scripts	jaquejbrito	updating figures and scripts	3 months ago
summary_data	jaquejbrito	updating figures	3 months ago
.gitignore	jaquejbrito	cleaned notebooks and figures	5 months ago
README.md	jaquejbrito	Update README	3 months ago

Organizing your GitHub

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

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

The screenshot shows a GitHub README.md page. At the top, it says "README.md" and has an edit icon. Below the title, there are buttons for "Preprint online", "licence MIT", and "MIT". The main title is "Benchmarking of computational error-correction methods for next-generation sequencing data". Below the title, it says "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"](#)". A "Table of contents" section lists: "How to cite this study", "Reproducinf results" (with sub-points "Tools", "Data", "Notebooks and Figures"), "License", and "Contact". Under "How to cite this study", it says "Mitchell, Keith, et al. "Benchmarking of computational error-correction methods for next-generation sequencing data" bioRxiv, doi: <https://doi.org/10.1101/642843>". Under "Reproducing results", there is a "Tools" section which states: "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."

Organizing your GitHub

- The README file is written with markdown language

A screenshot of a GitHub repository page for 'Mangul-Lab-USC / benchmarking_error_correction'. The page shows the repository's main interface with tabs for Code, Issues (0), Pull requests (0), Actions, Projects (0), Wiki, Security (0), and Insights. Below this, a modal window is open for the 'README.md' file. The modal has tabs for 'Edit file' and 'Preview changes'. The preview area contains the following markdown code:

```
1 # Benchmarking of computational error-correction methods for next-generation sequencing data
2
3
4 [[Preprint Available]](https://img.shields.io/badge/Preprint-online-green.svg)](https://doi.org/10.1101/642843) [[MIT
5 Licence]](https://badges.frapsoft.com/os/mit/mit.svg?v=103)](https://opensource.org/licenses/mit-license.php)
6
7 This project contains the links to the datasets and the code that was used for our study : ["Benchmarking of computational
8 error-correction methods for next-generation sequencing data"](https://doi.org/10.1101/642843)
9 **Table of contents**
10
11 * [How to cite this study] (#how-to-cite-this-study)
12 * [Reproducinf results] (#reproducing-results)
13     * [Tools] (#tools)
14     * [Data] (#data)
15     * [Notebooks and Figures] (#notebooks-and-figures)
16 * [License] (#license)
17 * [Contact] (#contact)
18
19
20 # How to cite this study
21
22 > Mitchell, Keith, et al. "Benchmarking of computational error-correction methods for next-generation sequencing data"
23 bioRxiv, doi: https://doi.org/10.1101/642843
```

Organizing your GitHub

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

The screenshot shows a GitHub repository page for 'Mangul-Lab-USC / benchmarking_error_correction'. The repository has 2 stars and 2 forks. The 'summary_data' folder is selected, showing 10 CSV files. The commits are listed below:

File	Description	Time Ago
D1_WGS_E.coli_summary.csv	updating figures	3 months ago
D1_WGS_human_complexity_summary.csv	updating figures	3 months ago
D1_WGS_human_cpu_memory.csv	cleaned notebooks and figures	5 months ago
D1_WGS_human_summary.csv	updating figures	3 months ago
D2_TCRA_real_summary.csv	updating figures	3 months ago
D3_TCRA_simulated.csv	updating figures	3 months ago
D4_HIV_summary.csv	upload D4 dataset	4 months ago
D5_HIV_mixture_diversity_summary.csv	upload D4 dataset	4 months ago
D5_HIV_mixture_error_rate_summary.csv	updating figures	3 months ago
D5_HIV_mixture_original_summary.csv	upload D4 dataset	4 months ago

Organizing your GitHub

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

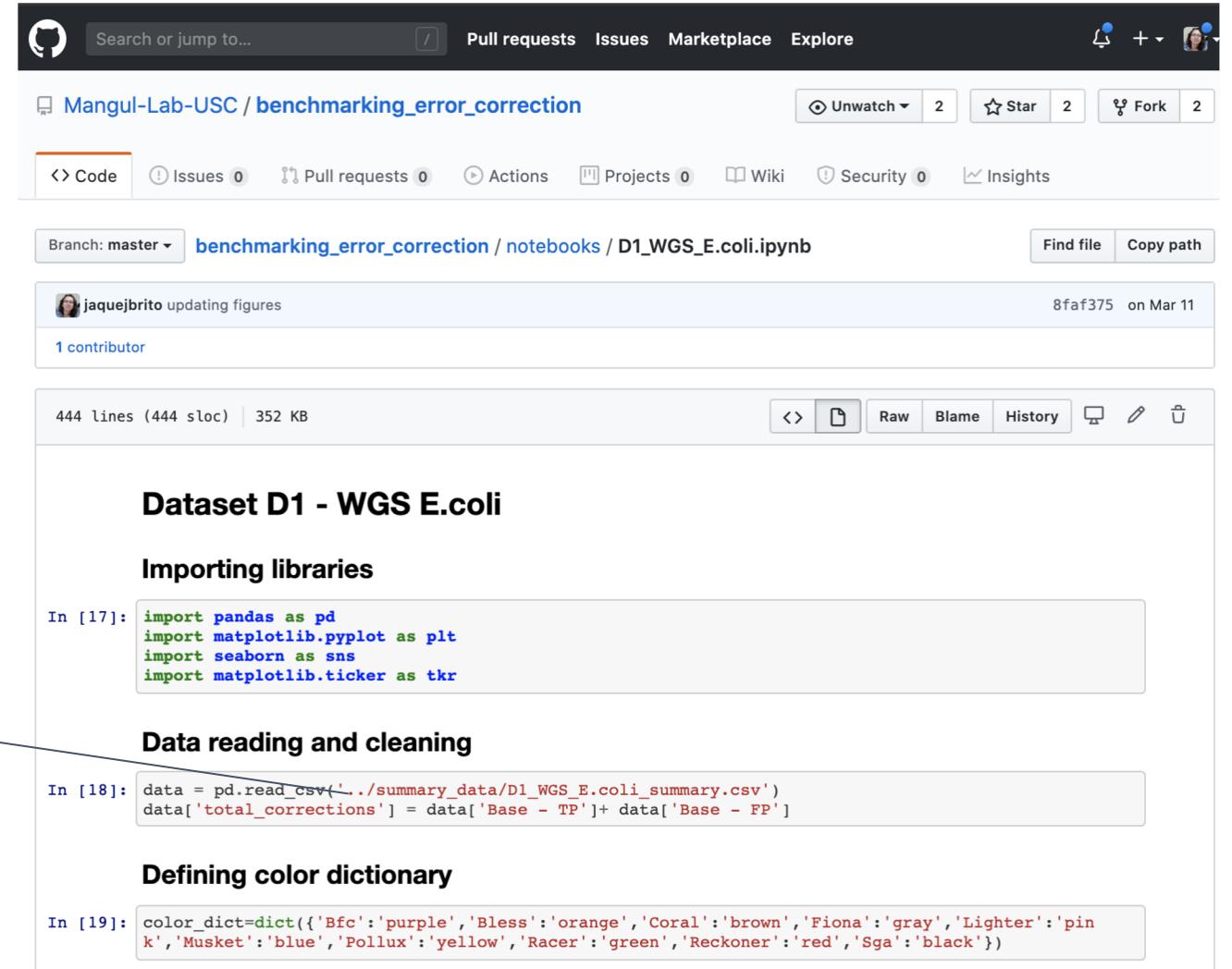
The screenshot shows a GitHub repository page for 'Mangul-Lab-USC / benchmarking_error_correction'. The repository has 2 stars and 2 forks. The 'Code' tab is selected, showing a list of files in the 'benchmarking_error_correction / notebooks /' directory. A commit by 'jaquejbrito' titled 'updateing figures' is shown, with the latest commit being '793e0db' on March 12. The files listed are:

File	Description	Time Ago
D1_WGS_E.coli.ipynb	updating figures	3 months ago
D1_WGS_human.ipynb	updating figures	3 months ago
D2_TCRA_real.ipynb	updating figures	3 months ago
D3_TCRA_simulated.ipynb	updating figures and scripts	3 months ago
D4_HIV.ipynb	updateing figures	3 months ago
D5_HIV.ipynb	updating figures and scripts	3 months ago

Organizing your GitHub

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

Reading summarized data from summary_data folder



The screenshot shows a GitHub repository page for 'Mangul-Lab-USC / benchmarking_error_correction'. The 'Code' tab is selected, displaying the file 'benchmarking_error_correction / notebooks / D1_WGS_E.coli.ipynb'. The notebook content is visible, showing code cells for importing libraries, reading data, and defining a color dictionary.

```
In [17]: import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.ticker as tk
In [18]: data = pd.read_csv('../summary_data/D1_WGS_E.coli_summary.csv')
data['total_corrections'] = data['Base - TP']+data['Base - FP']
In [19]: color_dict=dict({'Bfc':'purple','Bless':'orange','Coral':'brown','Fiona':'gray','Lighter':'pink','Musket':'blue','Pollux':'yellow','Racer':'green','Reckoner':'red','Sga':'black'})
```

Dataset D1 - WGS E.coli

Importing libraries

```
In [17]: import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.ticker as tk
```

Data reading and cleaning

```
In [18]: data = pd.read_csv('../summary_data/D1_WGS_E.coli_summary.csv')
data['total_corrections'] = data['Base - TP']+data['Base - FP']
```

Defining color dictionary

```
In [19]: color_dict=dict({'Bfc':'purple','Bless':'orange','Coral':'brown','Fiona':'gray','Lighter':'pink','Musket':'blue','Pollux':'yellow','Racer':'green','Reckoner':'red','Sga':'black'})
```

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")

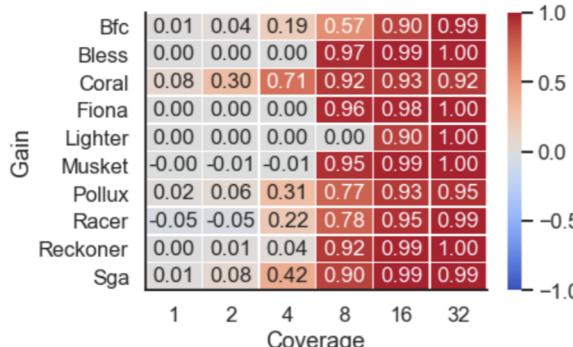
g=sns.heatmap(result,
               annot=True,
               cmap='coolwarm',
               center=0,
               linewidths=.5,
               annot_kws={'size':17},
               fmt=".2f",
               vmin=-1,
               vmax=1)

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_yticks([0, 10])

sns.despine()

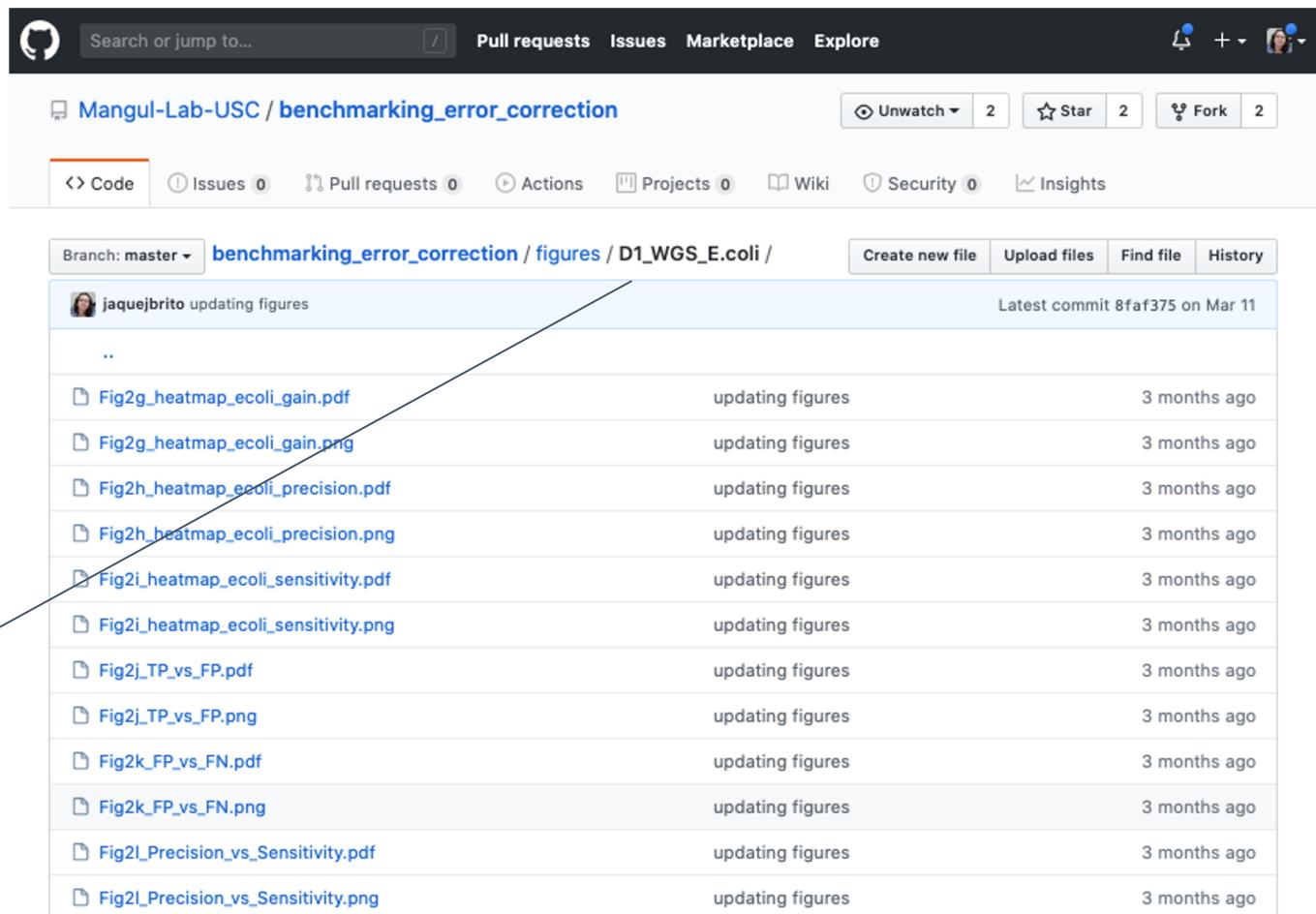
plt.savefig("../figures/D1_WGS_E.coli/Fig2g_heatmap_ecoli_gain.png", bbox_inches="tight")
plt.savefig("../figures/D1_WGS_E.coli/Fig2g_heatmap_ecoli_gain.pdf", bbox_inches="tight")
```



Organizing your GitHub

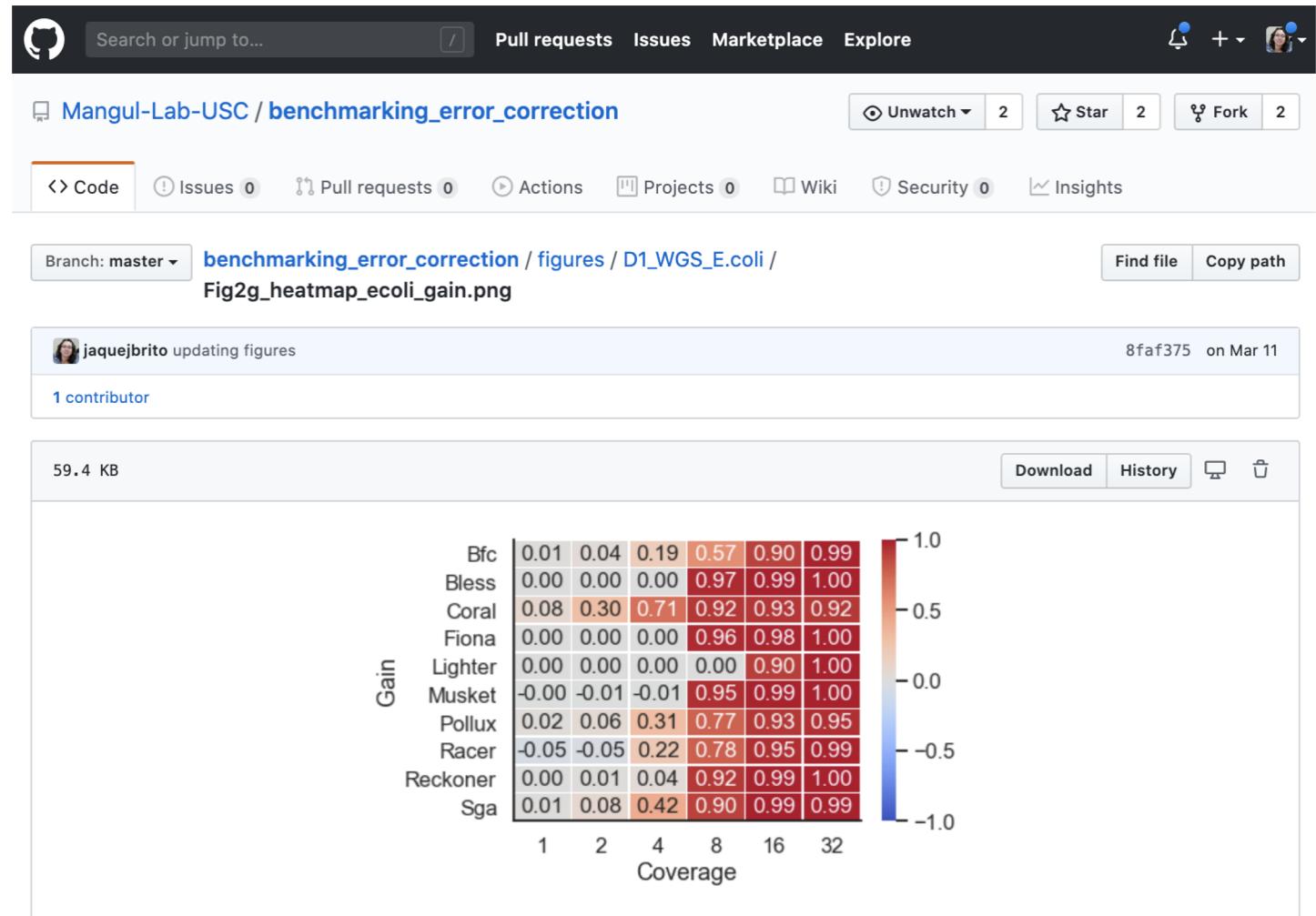
- The **figures** folder stores images of plots generated by the notebooks

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

The screenshot shows a GitHub repository page for 'Mangul-Lab-USC / benchmarking_error_correction'. The repository has 2 stars and 2 forks. The 'Code' tab is selected. A commit from 'jaquejbrito' is shown, updating figures. The commit message is 'updating figures'. The commit was made 3 months ago on March 13, 2023, with the SHA 'a18d22b'. Below the commit, there is a list of files in the 'figures_multi_panel' folder:

File	Description	Time
Fig1.ai	updating figures	3 months ago
Fig1.png	updating figures	3 months ago
Fig2.png	updating figures	3 months ago
Fig2.psd	updating figures	3 months ago
Fig3.png	updating figures	3 months ago
Fig3.psd	updating figures	3 months ago
Fig4.png	updating figures	3 months ago
Fig4.psd	updating figures	3 months ago

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

The screenshot shows a Figshare dataset page. At the top, there's a navigation bar with the Figshare logo, a search bar, and user account links for 'Upload', 'My data', and a notification bell. Below the navigation, four dataset cards are displayed, each featuring a dark grey icon with a vertical line pattern and the word 'ARCHIVE' below it. The datasets are labeled: 'D1_WGS_E.coli.zip (275.74 MB)', 'D1_WGS_human... .zip (434.37 MB)', 'D1_WGS_human... .zip (391.11 MB)', and 'D1_WGS_human... .zip (521.5 MB)'. Below these cards is a toolbar with buttons for 'Cite', 'Download all (7.1 GB)' (which is highlighted in red), 'Share', 'Embed', '+ Collect', and a three-dot menu. To the right of the toolbar, it says '21 files 1 / 3 < > ■ ■ ■'. The main content area features a large title: 'Datasets (raw and true reads) supporting the study "Benchmarking of computational error-correction methods for next-generation sequencing data"'. Below the title, a 'Version 3' section provides details about the dataset's posting date (05.03.2020), author (Keith Mitchell, Jaqueline Brito, Igor Mandric, Qiaozhen Wu, Sergey Knyazev, Sei Chang, Lana S. Martin, Aaron Karlsberg, Ekaterina Gerasimov, Russell Littman, Brian L. Hill, Nicholas C. Wu, Harry Yang, Kevin Hsieh, Linus Chen, Eli Littman, Taylor Shabani, German Enik, Douglas Yao, Ren Sun, Jan Schroeder, Eleazar Eskin, Alex Zelikovsky, Pavel Skums, Mihai Pop, Serghei Mangul). To the right of this text are metrics: '128 views', '25 downloads', and '2 citations'. Further down, under 'CATEGORIES', is a link to 'Genomics'. Under 'KEYWORD(S)', several tags are listed: 'genomics', 'genomics tools', 'error correction', 'omics', 'read sequences', 'TCRA', 'HIV', 'RNA-Seq', and 'E.coli'.

“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

The screenshot shows the header of the PLOS Computational Biology website. The top navigation bar includes links for BROWSE, PUBLISH, and ABOUT. The main title 'PLOS COMPUTATIONAL BIOLOGY' is displayed prominently. Below the title, there are two categories: OPEN ACCESS and EDUCATION.

A Quick Introduction to Version Control with Git and GitHub

John D. Blischak , Emily R. Davenport, Greg Wilson

Published: January 19, 2016 • <https://doi.org/10.1371/journal.pcbi.1004668>

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