Table of Contents

[GitHub for Research Methods 2](#_Toc64974291)

[**1.** **What is Git and GitHub?** 2](#_Toc64974292)

[**1.1** **GIT** 2](#_Toc64974293)

[**1.2** **GitHub** 2](#_Toc64974294)

[**1.3** **Creating a Git Account** 3](#_Toc64974295)

[**2.** **Git Repository** 4](#_Toc64974296)

[**2.1** **Creating a repository** 4](#_Toc64974297)

[**3.** **Git Architecture** 5](#_Toc64974298)

[**3.1** **Git Operations** 6](#_Toc64974299)

[**3.2** **Other GIT concepts** 7](#_Toc64974300)

[**4.** **A Sample Git project - using GitHub** 8](#_Toc64974301)

[**4.1** **Create a new repository** 8](#_Toc64974302)

[**4.2** **Create a new branch using main branch** 9](#_Toc64974303)

[**4.3** **Opening a Pull request** 10](#_Toc64974304)

[**4.4** **Merge Pull request** 12](#_Toc64974305)

[**5.** **Creating a GIT profile for researchers and data scientists** 13](#_Toc64974306)

[**5.1** **Default GitHub Profile** 13](#_Toc64974307)

[**5.2** **Profiling a repository** 14](#_Toc64974308)

[**5.3** **GitHub Pages** 15](#_Toc64974309)

[**5.4** **Updating the readme.md File for website customization** 17](#_Toc64974310)

[**6. References** 23](#_Toc64974311)

# GitHub for Research Methods

1. **What is Git and GitHub?**
   1. **GIT**

* Git is the actual software that is used to maintain source control, or track changes for code for collaborations between project teams
* It can be accessed through command line or front-end
  1. **GitHub**
* GitHub is the front-end to the Git, and can be accessed as a website
* It provides an easy Graphical User Interface (GUI) to maintain code for users who are not comfortable using the command line
* It provides a neat interface to the users to document their code, by providing comments and instructions wherever necessary.
* Also provides a personal website (GitHub pages) that can be published to create an impressive profile of their work, providing references to their code and credentials.
* This is how a GitHub page might look with the appropriate configuration changes.

Text

Description automatically generated

* 1. **Creating a Git Account**
* Login to[**https://github.com**](https://github.com/)
* Sign up for GitHub using email
* Once you signup, the following page pops up, select accordingly

Graphical user interface

Description automatically generated

* Once email is verified, we are the home page, and are ready to create a repository to start maintaining code

1. **Git Repository**

* A repository is a central server storing source code (commonly referred to as Repo)
* It is the repository that has the latest working copy of code, and developers are expected to build upon the same
* A user can choose to have one or more Git repositories – as a best practice, for large projects, chose a separate repository, whereas you may maintain smaller projects within the same repository. Nevertheless, it is up to user’s discretion.
* A repository can either be public or private. Public repositories are open for anyone’s viewing, and also support GitHub pages creation. Private repos are only available to the user.
  1. **Creating a repository**

There are several options to create a new repository

* **Create new repository**

This is the default way of creating a repository and allows user to create the repository from scratch. Use this method to start checking in code for a new project. This document will use this method of creating a repo.

* **Import repository**

Use this method to import another Version Control System and start working on it.  Importing repositories from Git, Subversion, Mercurial, and TFS is currently supported in Git

* **Cloning repository**

Use this method to clone a created repository on your local machine for editing

* **Forking repository**

Use this method to create a new repository using an existing one. This allows user to create own implementation of an existing repository, without changing the original repo. This is used when user wants to make changes to the source repository eventually.

1. **Git Architecture**

**Diagram

Description automatically generated**

Image Reference: https://www.geeksforgeeks.org/git-features/

* **Remote Repository**

This is the remote server where code is centrally located. This contains the latest and greatest copy of code which will be available to all users using the Git repository

* **Local Repository**

This is a copy of code on user’s location machine. As shown, each user will have a copy of remote repository on their machines. They have to update it time to time as and when changes are made on the remote repository, so that their code has latest changes

* **Workstation**

User’s local machine

* **GIT**

The technology used to update code either through commands or frontend (GitHub)

* 1. **Git Operations**
* **Update**

Update local repo with latest changes in remote repository

* **Commit**

Commit that changes have been made to user’s local repository. Add a suitable change to indicate what changes have been made over previous versions

* **Push**

Once committed, these changes need to be pushed to remote repo using Push

* **Pull**

Notifies others, including owner that changes were pushed to a branch into a particular repository. Allows repository owner to then merge this change to their repository after reviewing. Requires user to fork the remote repository to make their changes. Once the pull request is accepted and merged, the changes will be added to remote repository. Users can also request a pull from main branch within same repository, if not across forked repositories

* 1. **Other GIT concepts**

**- Branch**

**A picture containing diagram

Description automatically generated**

* Branches are how git enables parallel development.
* They exist within a repository. The main branch is called ‘main’ which would hold the baseline code.
* While starting a new development on top of the existing baseline code, the developer would create a new branch using the main branch and start working on the replicated code.
* **Merging**
* Once development is complete, the new branch can then be merged with main branch. While merging, there could be potential conflicts while checking in code, just in case there are other branches that are in the process to be checked in. Any conflicts are highlighted by Git, and these should be resolved before going ahead in order to have clean code in main.

1. **A Sample Git project - using GitHub**

* Create a GitHub account using steps mentioned earlier
* In the following series of steps, we will create a new repository of name **hello-world** where you store ideas, resources, or even share and discuss things with others.
  1. **Create a new repository**
* Click on the + to the left of the avatar at the top right corner of the screen

**Application

Description automatically generated with medium confidence**

* Name it hello-world
* Click on public and add a README file

Graphical user interface, text, application, email

Description automatically generated

* The readme file of the branch can be edited to add new details
* Click on commit Changes with appropriate comments

Graphical user interface, application, Teams

Description automatically generated

* Thus, we have created a new repository hello-world with a default branch (main)
  1. **Create a new branch using main branch**
* Next, we will create a new branch readme-edits, as if we are adding a new feature to our existing code in main branch

Graphical user interface, text, application

Description automatically generated

* Click on the highlighted link
* This will create a new branch readme-edits containing code from the main branch
* Now we will add a line in the readme.md file of this branch, indicating a change in the original content, and commit the changes

**Note:** Readme.md file uses markdown syntax

Graphical user interface, text, application, email

Description automatically generated

* 1. **Opening a Pull request**
* Next, we can add a pull request in order to request these changes to the main branch. In a real-world scenario, the main branch will be owned by the code administrator, or the production support team. Pulling will send them a notification asking them to review our changes. This will enable them to closely monitor changes to source code versions.
* Click on the **Pull requests** tab on the hello-world branch menu, and then **New Pull requests**

**Graphical user interface, text, application, email

Description automatically generated**

* Here we will select the 2 branches that we want to merge, where base will be the main branch, and compare will be the readme-edits branch

Graphical user interface, text, application, website

Description automatically generated

* Here we will be able to see the differences between the 2 branches, notice that the new lines of code are now visible

Graphical user interface

Description automatically generated with medium confidence

* Now, we will create the “Create pull request”, and add some meaningful comments in order to bring the changes to the reviewer’s notice, and request to take these changes
  1. **Merge Pull request**
* Now click on Merge Pull request

Graphical user interface, text, application

Description automatically generated

* Now click on Confirm Merge to confirm merging the changes.

Graphical user interface, text, application, email

Description automatically generated

* Once confirmed, you should see a message, this indicates that the pull request was successfully closed

Graphical user interface, text, application

Description automatically generated

1. **Creating a GIT profile for researchers and data scientists**

While GitHub is most popular as a version control and collaboration utility, it also provides an excellent framework for developers to proudly display their work profile. In fact, an increasing number of recruiters look forward to candidates’ GitHub profiles in order to understand the nature of their work experience.

This section below will describe a few tips to decorate a GitHub profile.

* 1. **Default GitHub Profile**
* GitHub creates a default profile for every user, this can be seen at –

**Graphical user interface, text, application, chat or text message

Description automatically generatedGraphical user interface, application

Description automatically generated**

* This page shows all the repos owned by user, and the commits made to them over time.
* User can also update profile using the Edit Profile button on the left and add a brief biography and details of concentrations, interests, etc. here.
  1. **Profiling a repository**
* Since our code would mainly reside inside a repository (or multiple, as per choice of the project/user), we can provide useful information inside a repository, in order to provide interesting insight into the project code.
* For the purpose of this document, we will use the hello-world repository that we created above.
* Navigate to the repository

**Graphical user interface, text, application, email

Description automatically generated**

* Click on About to add a brief description of the repository
* Click on Settings. Most of the repository configurations can be done here. Here, we can switch a repository to be either Private or Public, determining whether a website should be available to the internet. This, and some related actions, which can have severe consequences, are located at the end of the page

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

* 1. **GitHub Pages**
* We will make this repository public, since we are going to showcase our work across the internet. This also gives us an option to create a GitHub website, called GitHub pages, which can act as the front-end, and GitHub provides impressive options to customize it, as if it were your own website. We can also add references to code for corresponding sections within GitHub, and also add visualizations. How cool!
* To configure GitHub Pages, ensure that the repo is Public, and go to the GitHub Pages section. On the drop-down, choose where the main code for the project is located. This is most likely going to be the Main branch, since it should have working copy of the code. The other branches are most likely to be Work in Progress, or not merged yet.

Graphical user interface, text, application

Description automatically generated

* Now choose the /(root) folder, since this will be the starting point of the branch and click on Save. More information about the choosing the source can be found at <https://docs.github.com/en/articles/configuring-a-publishing-source-for-your-github-pages-site#choosing-a-publishing-source>
* In case if the settings page closes, navigate to it again. The following message appears, this is the website GitHub created for you, based on the repo/branch provided to it.

Website, timeline

Description automatically generated

* You can now navigate to this website. Sometimes, it might take a few minutes to commission the website, and a 404 Error might show up. In such a case, try sometime later. You can work on the readme.md file in the meantime.

Graphical user interface, text, application

Description automatically generated

* It is a blank template website, which shows the name of the repository and the content of the readme.md file of the branch. In the subsequent sections, we will add meaningful content to our readme.md file so that it shows up on our website
  1. **Updating the readme.md File for website customization**
* Navigate to the main branch
* We will now add sample code and visualizations to this branch
* We will now create a directory called ‘code’ and create new readme.md file inside this new directory to initialize the branch.
* Click on Add file -> Create new file

Graphical user interface, text, application

Description automatically generated

* Add a new directory by typing ‘code/’ – adding a ‘/’ creates a new directory, and not adding It creates a file. We will add readme.md, and add a brief description of the directory, and Click on “Commit New file”

Graphical user interface, text, application, email

Description automatically generated

* A new path with the file inside is created

Graphical user interface, text, application

Description automatically generated

* We will now upload sample code into the new path. Using similar steps above inside the code directory, I added a file Called HelloWorld.py that prints “Hello World” on the screen.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated

* To download the file in its original content, click on the filename -> Raw -> Save Page As (mac) and select location (in windows it will be different).

**Note:** Jupyter Notebooks (ipynyb files ) can be downloaded the same way, even though they show the source code on Clicking Raw.

* Similarly, to add images, create a images folder in the branch. I will add a sample image to the folder

Graphical user interface, text, application, Teams

Description automatically generated

* Now we will edit the readme.md file in the main branch, since this document will be rendered to the website on GitHub Pages.
* Since we will be redirecting the users of the website to the code location, copy the URL of the code directory, and keep it handy since we will be needing it below
* Also, we will be displaying the image we uploaded previously. Copy the link of the image you want to display and keep it handy. For images, provide the internal link (not the url, as if it were in a directory – as shown in figure below)

Graphical user interface, text, application, chat or text message

Description automatically generated

* Click on pen on the branch root readme.md file to edit it
* Delete Existing content of the file
* Edit the file as shown in screenshot using corresponding links

Graphical user interface, text, application, email

Description automatically generated

* Preview changes using the Preview changes pane, if changes look satisfactory, commit them. If done correctly, the readmd.md page would look similar, with the image and link we selected earlier.

A picture containing application

Description automatically generated

* Now we will migrate to the GitHub Pages website we created earlier.

A picture containing application

Description automatically generated

* This website can then be shared with relevant parties, or in your LinkedIn profile

**6. References**

* [My GitHub Page](https://paulamiguha.github.io/Paulami_portfolio/) – Good to refer for beginners
* An advanced website - [https://dhrumil.xyz](https://dhrumil.xyz/)
* A short video for reference - <https://www.youtube.com/watch?v=1aXk2RViq3c>
* GitHub for Data Scientists – By Sara Anstey (requires LinkedIn Premium)
* Markdown CheatSheet: <https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet>