R Version Control

Vincent Arumadri

2025-07-08

R Version Control with Git/GitHub

Git/GitHub enables continuous software/code development by tracking code changes to the **initial commit**, tracking who made the changes and enabling code collaboration. Here I will go through the basics on how to step-up **Git/GitHub** for version control with **R/RStudio**. Although the focus is on R/RStudio, the same procedure can be followed by those who use other softwares for data analysis e.g **Python**, **LaTex**, **Julia** etc.

Required programs

To start off, ensure you have the following programs installed:

1. **Git** – the base program for tracking code changes.
    - **Windows**: You can download Git here. Double-click the downloaded .exe file to start the installation process.
    - **macOS**: First you need to install **Homebrew**. Copy and paste the Homebrew download link from (https://brew.sh/) to the Terminal and run.

    If you already have Homebrew installed, open Terminal and type:

    ```
    brew install git
    ```

2. **R** – the base R program
    - **Windows**: You can download R here
    - **macOS**: You can download R here
3. **RStudio** – IDE (integrated development environment) for R
    - **Windows and macOS**: You can download RStudio here
4. **GitHub Desktop** – for version control and collaboration
    - **Windows and macOS**: You can download GitHub Desktop here

Getting started

1. **Sign up for GitHub** – Got to GitHub and create a free account:

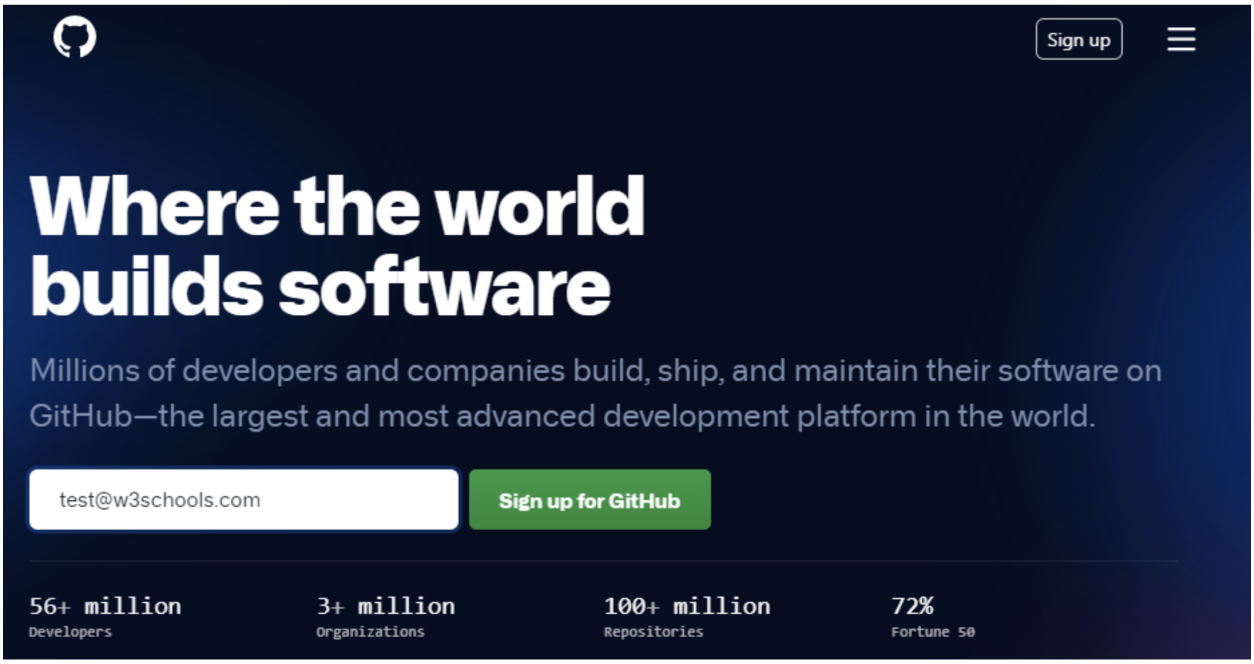
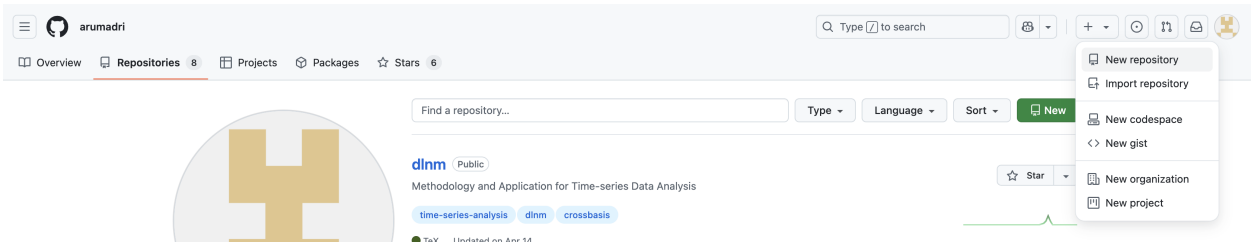


Figure 1: GitHub account setup

2. **Create a Repository** After signing in, click the + button to create a new repository:



3. **Fill in the repository details** (name, description, public/private, etc.) and click Create repository:

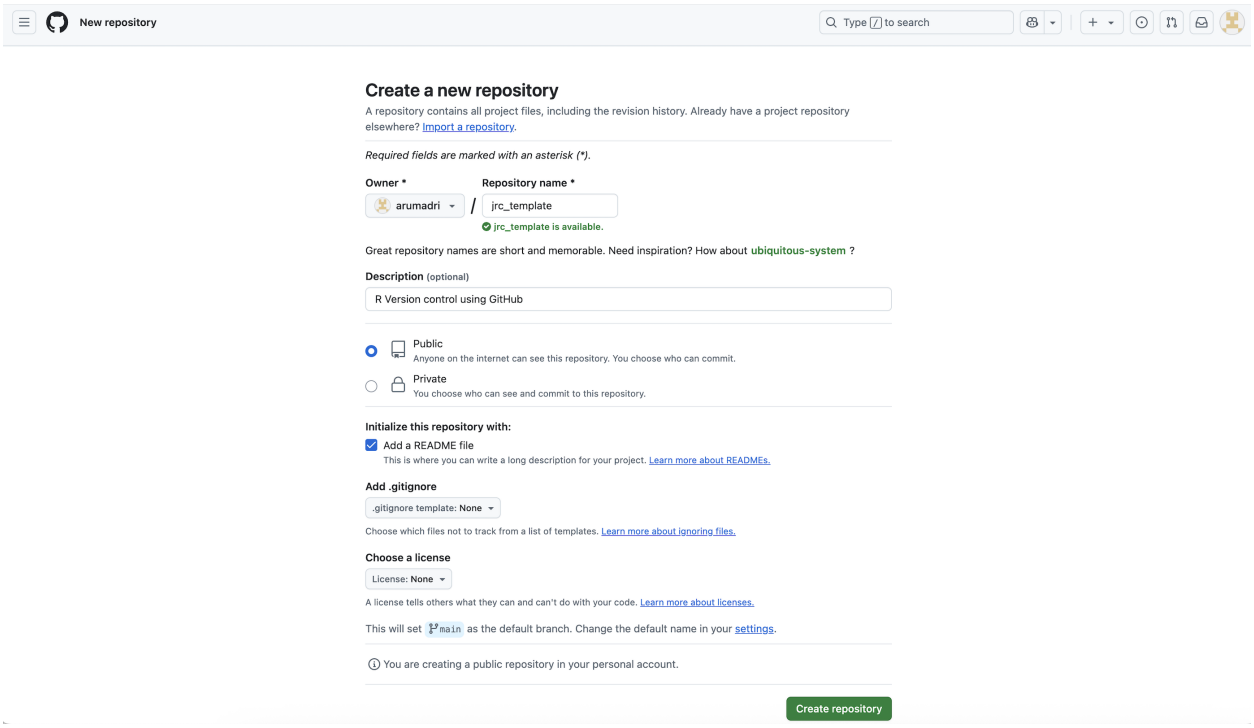


Figure 2: GitHub new repository details

Now you have set up your **remote repository**!!

4. **Clone remote repository to local machine (laptop/desktop)**

Inside the repository click the **<>Code** button and select **Open with GitHub Desktop** in the drop down menu.

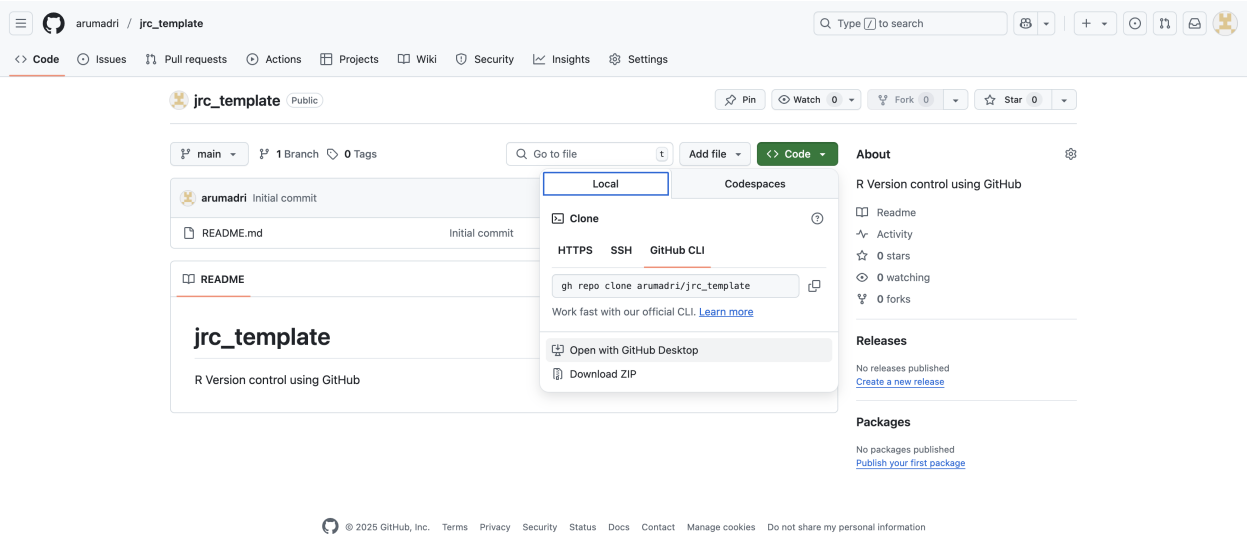
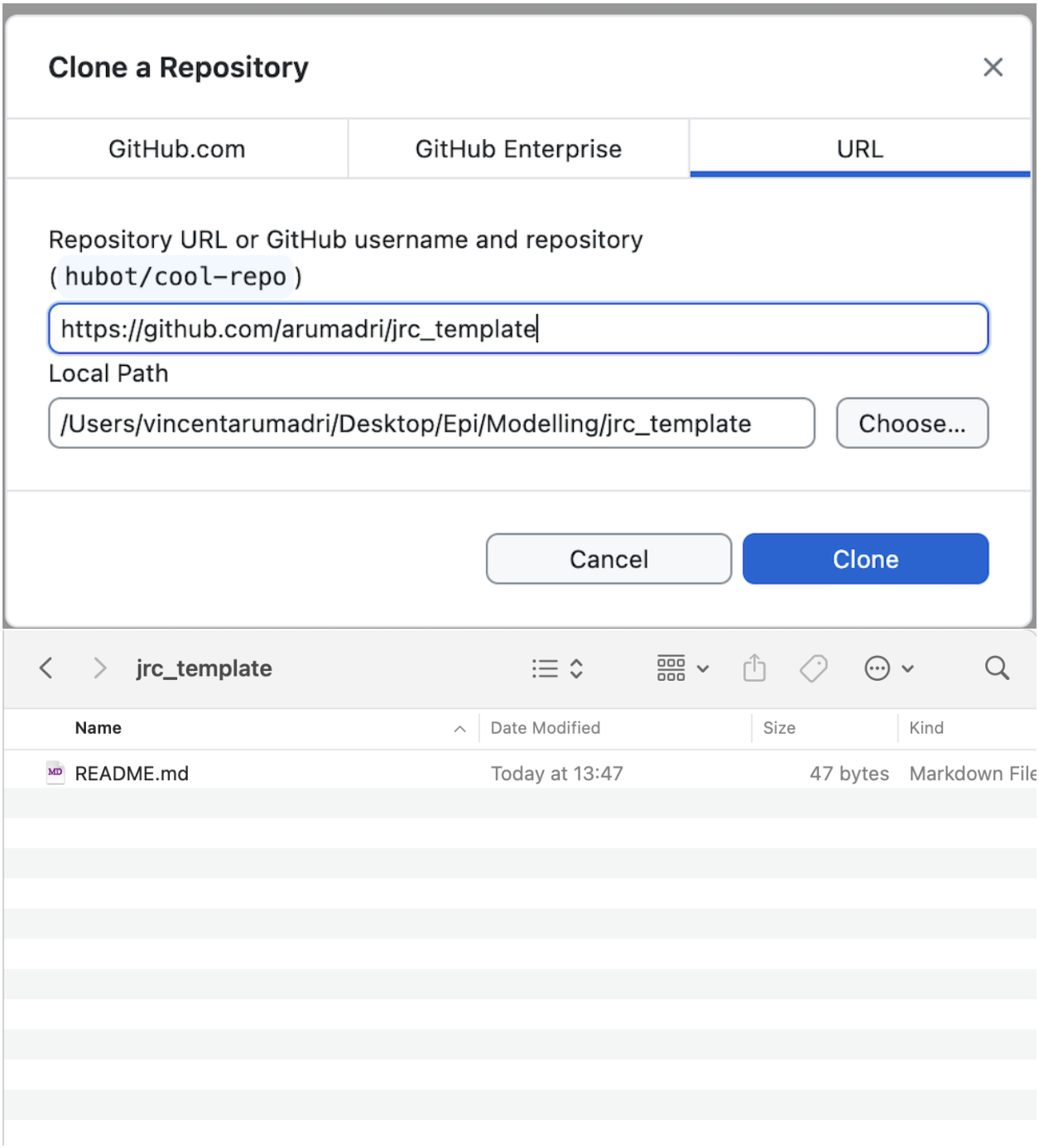


Figure 3: Clone remote to local

5. **Select path for local repository**

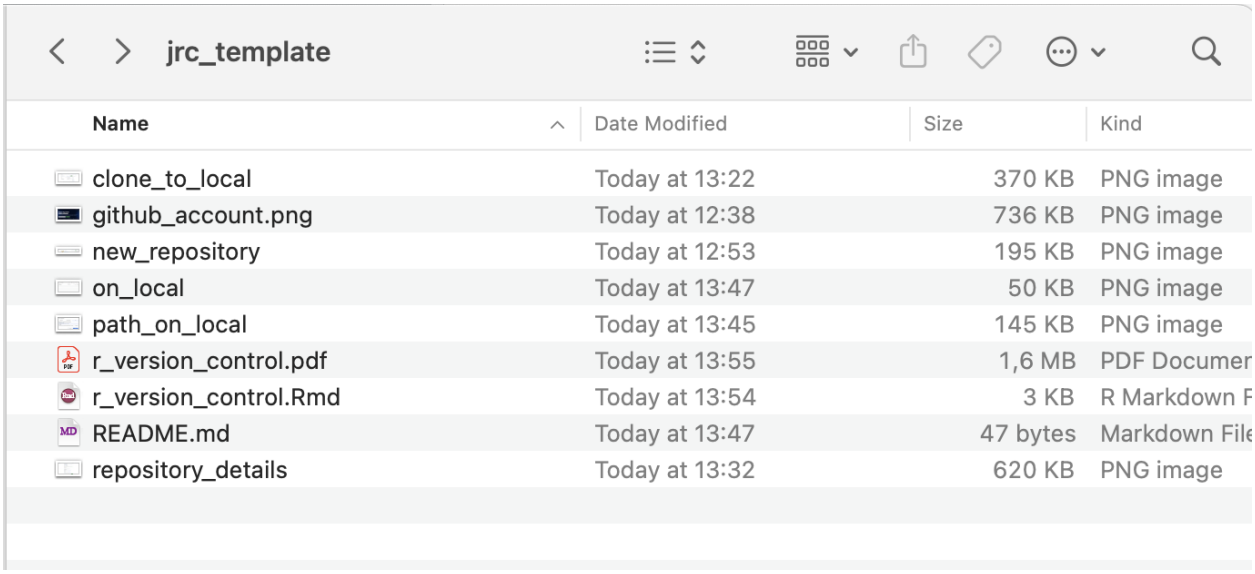


6. **Add or (save your new r scripts/files) to this folder**

This will initialize the **commit** and **push** prompt in GitHub Desktop to commit and push the changes respectively to the **remote repository**.

These prompts enable changes made to the folder (adding new files) or files (new edits to code) on the **local** machine to be **committed and pushed** to the **remote** for tracking and hence **version control**.

Here I add new files to the **local** folder.



These changes appear on **GitHub Desktop** ready to be **committed** and **pushed** to the **remote**.

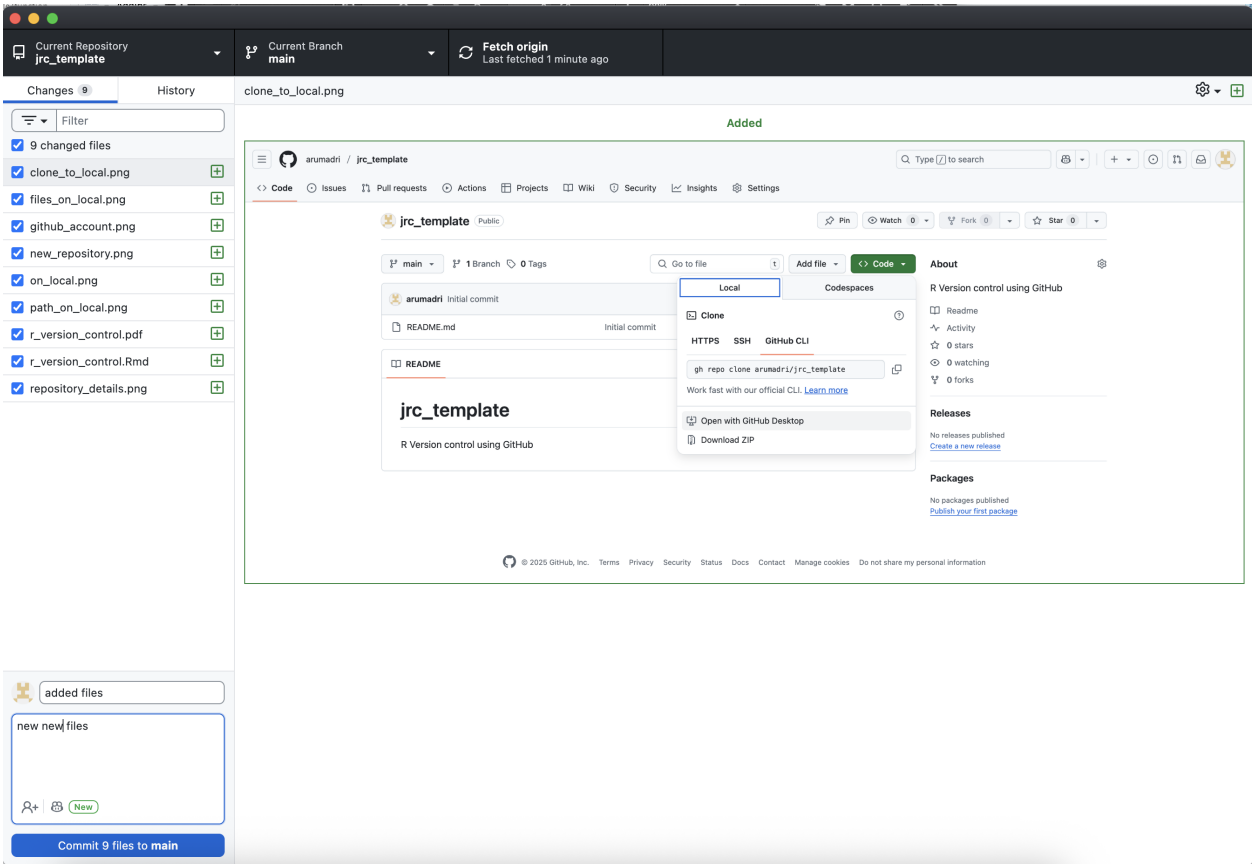
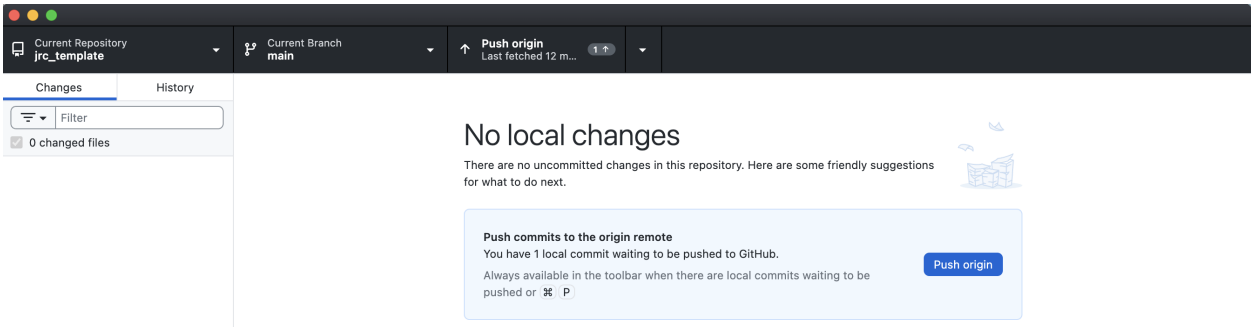
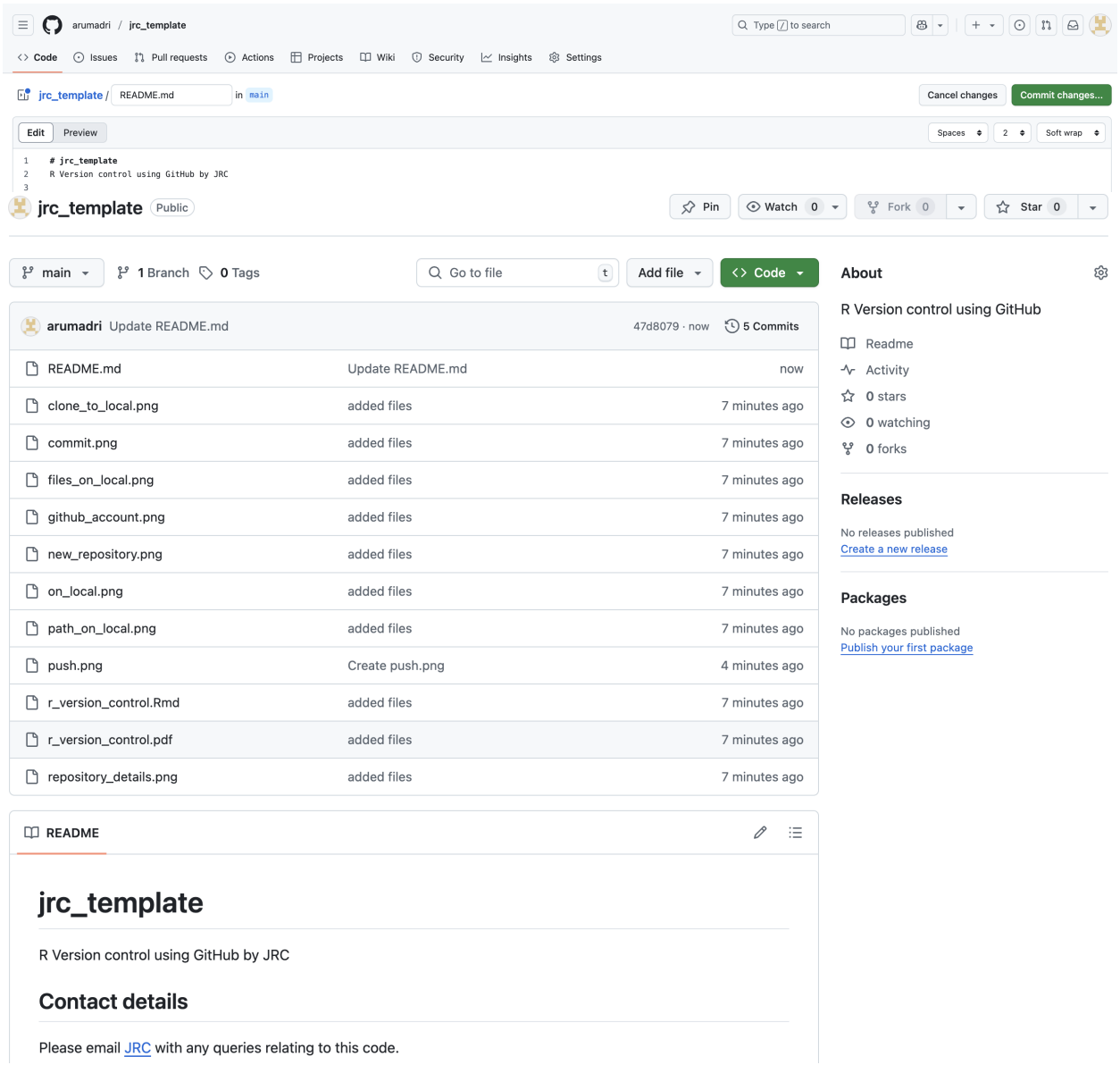


Figure 4: Commit

7. **Push** and **pull** changes to or from **GitHub**. After **committing** the changes, they are ready to be **pushed**.



8. Changes on the **remote** can be **pulled** to the **local**. This is especially useful in projects with multiple contributors to the code. Changes made by one person to the remote can be pulled to the local repository of another person to update their code to the latest version. Here I make some changes on the README file on **remote**, **commit** then **pull** them to **local** .



Within **Github Desktop**, these changes will be reflected and a prompt to pull them to your **local** will appear when the application is opened while connected to the internet.

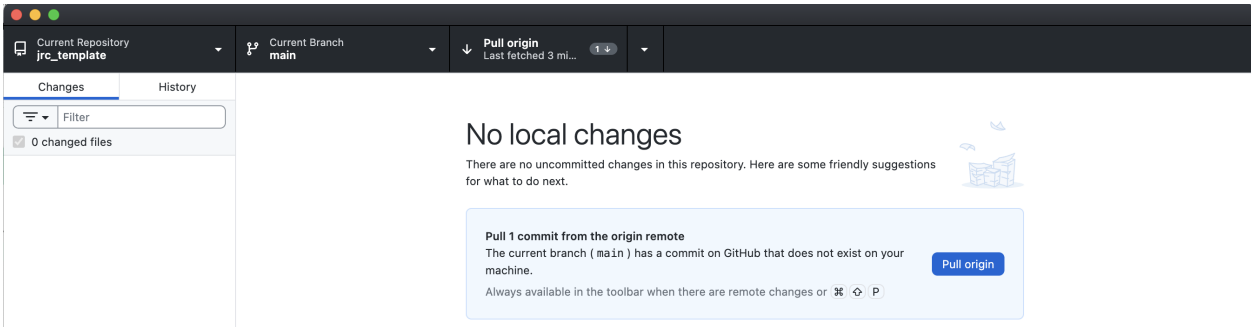


Figure 5: Pull to local

Changes made on the **remote** now **pulled** to the **local** as seen in **RStudio** here.

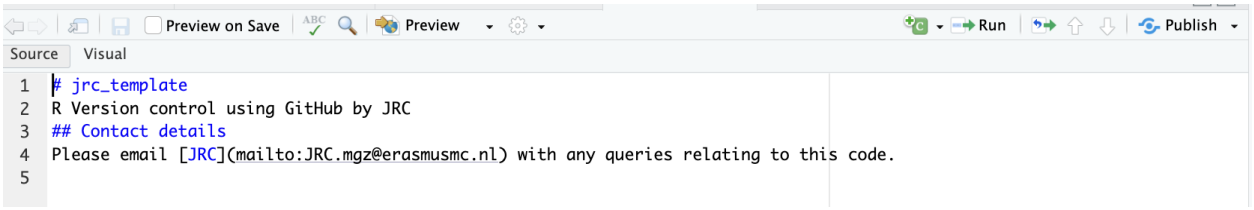


Figure 6: Commit changes on local

We hope you find this helpful and start using version control to keep track of your code changes. Do not hesitate to conduct **JRC** with any queries relating to this code. Good luck!