

Week 2 Lab: Github and Rmarkdown Tutorial

POLI502 Models of Political Analysis

AUTHOR
Howard Liu

PUBLISHED
August 29, 2024

Github

What is Github?

GitHub is a platform and cloud-based service for software development and version control using Git, allowing developers to *store* and *manage* their code for their projects.

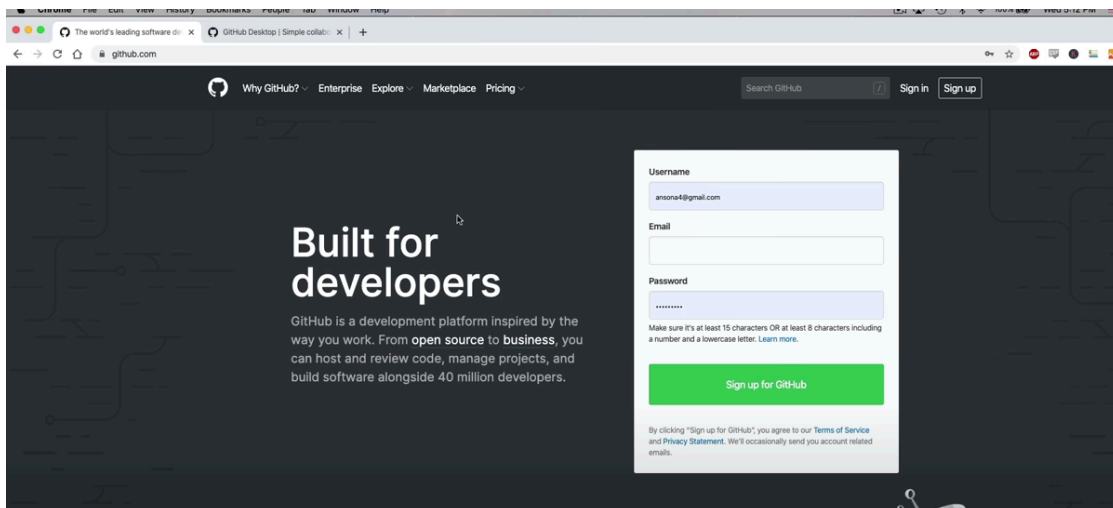
Why do I need to learn Github?

Because every programmer uses it!

GitHub lets users publish and share code effortlessly, making community-driven collaboration all the more productive. Git and GitHub help fuel community-driven software development, and learning how to use these tools will ensure that you, too, can participate in these projects.

- Create a free platform for you and your collaborators to share codes and data (if not large)
- Save your replication files
- Write and host a personal website
- Create a course website

1. Create an account



2. Create a SSH (Secured Shell) key

Why do you need an SSH key to push to GitHub?

SSH keys enable the automation that makes modern cloud services and other computer-dependent services possible and cost-effective. They offer convenience and improved security when properly managed. Functionally SSH keys resemble passwords. They grant access and control who can access what.

Follow this YouTube video and get the SSH key from your local machine added to your Github [here](#).

3. Create a new repository

This is what it will look like when you create a repository.

Q Type to search

Create a new repository

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

Required fields are marked with an asterisk (*).

Owner * / Repository name *

poli502 is available.

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

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.

Initialize this repository with:

Add a README file
This is where you can write a long description for your project. [Learn more about READMEs](#).

Add .gitignore

Choose which files not to track from a list of templates. [Learn more about ignoring files](#).

Choose a license

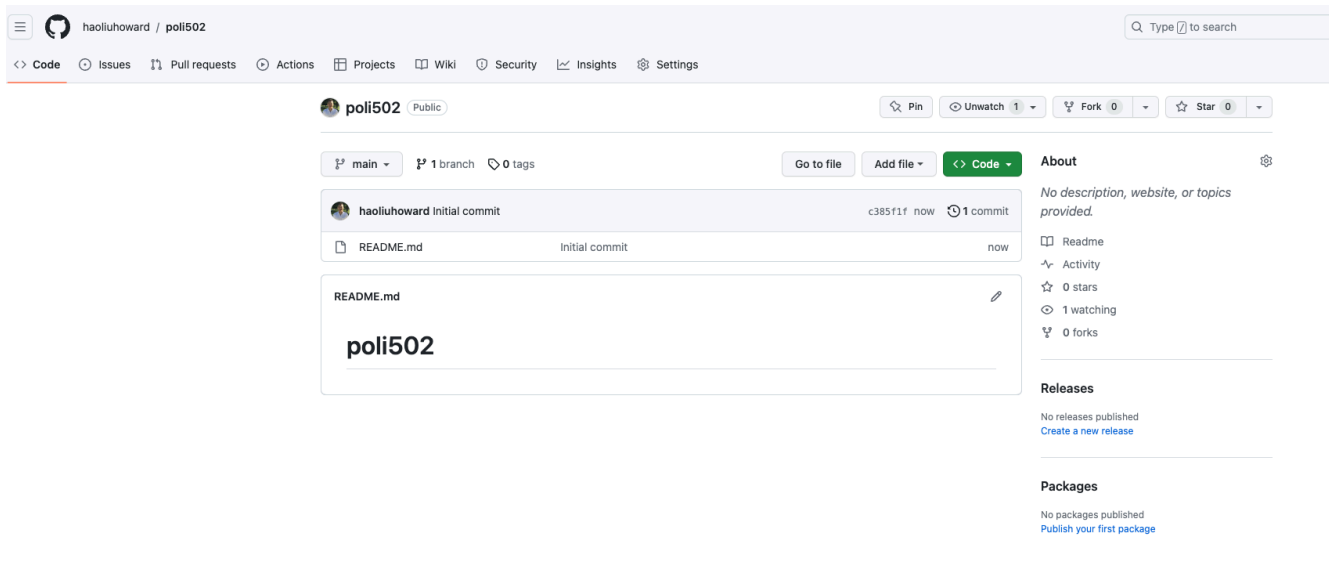
A license tells others what they can and can't do with your code. [Learn more about licenses](#).

This will set [main](#) as the default branch. Change the default name in your [settings](#).

You are creating a public repository in your personal account.

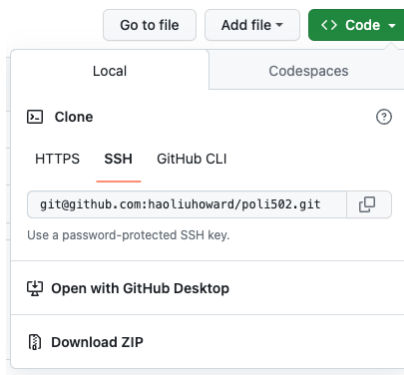
[Create repository](#)

And then you're going to see your newly created repository under your account. A repository usually means a new coding project. You're going to upload all your assignments here for this semester. Here I created a repo for poli502 as an example.



4. Connect your git repository to your local machine.

You first copy the repo link and select **ssh** as the way to clone:



Then you move to your **Terminal** tab next to the **Console** tab in RStudio. We use **Terminal** to connect your remote git repo and local machine rather than *github desktop* because I want you to get familiar with the basic Shell language. Shell (or Bash for Mac computers) is used for UNIX-based operating system. It is useful to learn it because you will need to use it when you learn SQL or Python or how to connect to HPC (High Performance Computing at UofSC).

Okay, back to our github connection:

1. You change the directory by using `cd` (change directory) in Terminal, which redirects you to a new directory where you want to put your repo into.

```
41 |
42 |
42:1 # 3. Connect your git repository to your local machine.
Console Terminal x Render x Background Jobs x
Terminal 1 ~ /Library/CloudStorage/Dropbox/myjobs/UofSC_South-carolina/SC_teaching/POLI502/502_week2/lab
Howards-Mac-Studio:lab howardliu$ cd ~howardliu/
```

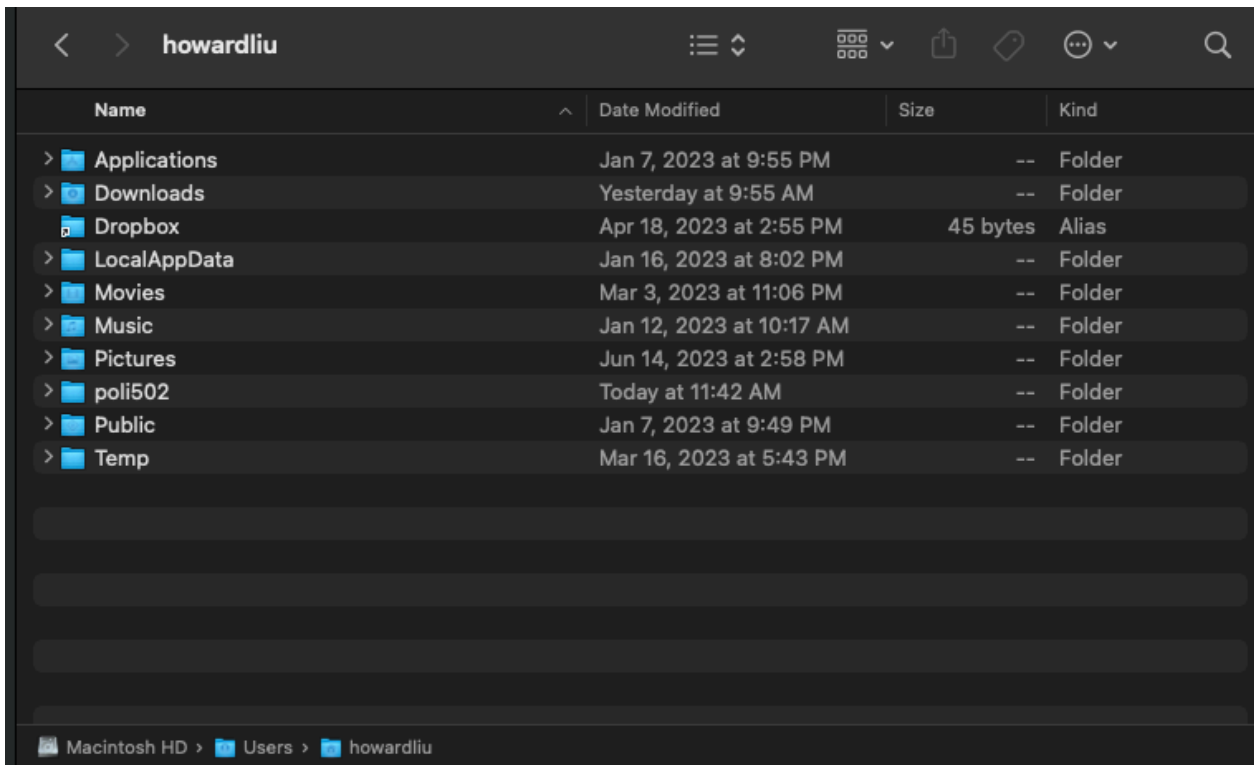
2. You clone your repo to your local machine.

```
42:1 # 3. Connect your git repository to your local machine.
Console Terminal x Render x Background Jobs x
Terminal 1 ~
Howards-Mac-Studio:lab howardliu$ cd ~howardliu/
Howards-Mac-Studio:~ howardliu$ git clone https://github.com/haoliuhoward/poli502.git
Cloning into 'poli502'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
Howards-Mac-Studio:~ howardliu$
```

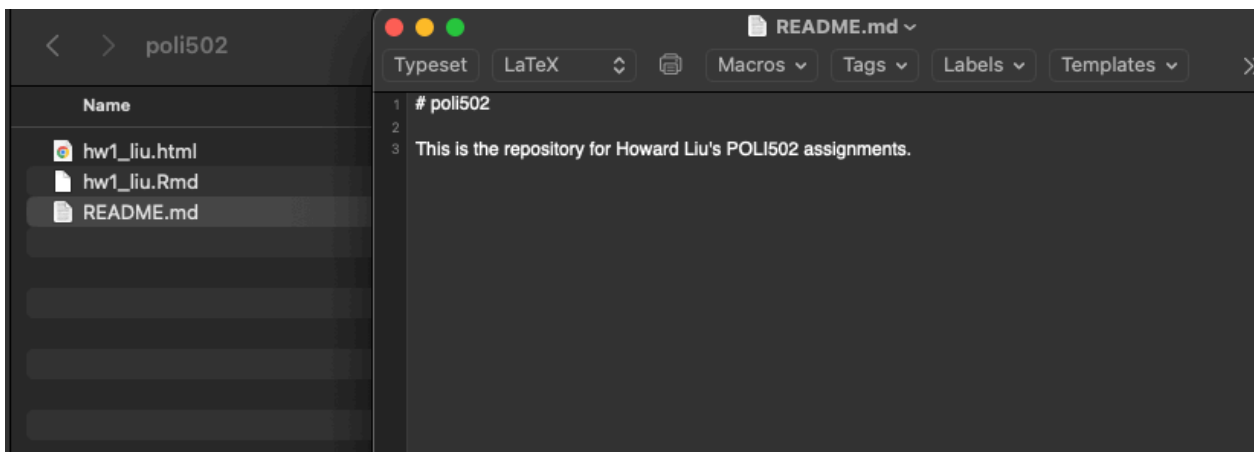
If you use the ssh key to clone, you may be asked if you allow this to connect. Say YES.

```
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

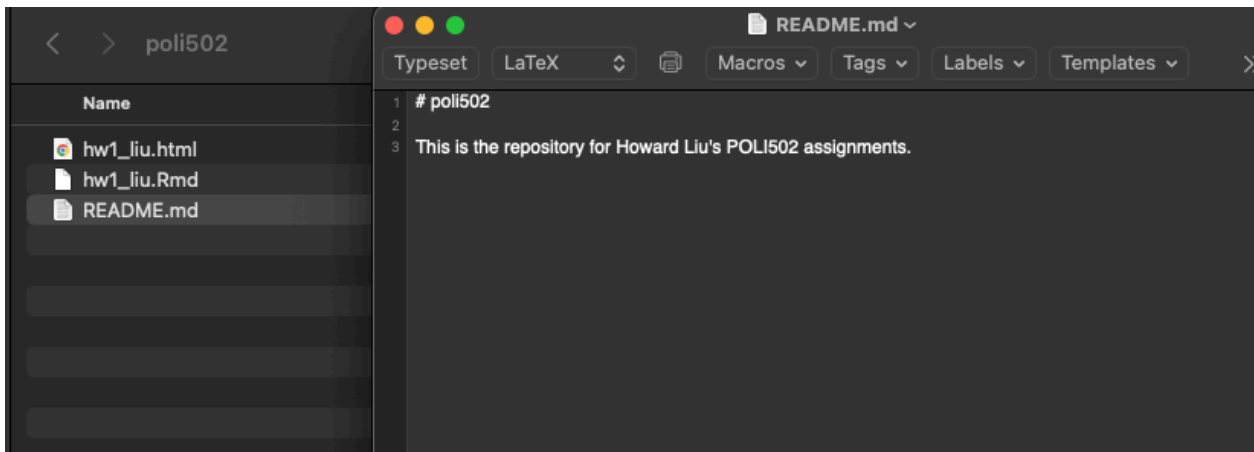
You will see that your repo has been successfully cloned to your local machine.



3. We can make changes locally and then push it back to our remote repo on Github. For example, let's say we edit some description of our repo in the **Readme.md** file and upload your homework 1 files (both of your **Rmd** and **html** files).



4. We **add** our changes, **comment** on what we have done so we know what to track, and **push** our changes onto our remote repo so everyone can see including me and the TA.



Note: `add -A` means that we add all changes.

Note: make sure you are in the local repo directory under **poli502**

5. It is done.

```

on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

3 files changed, 537 insertions(+), 1 deletion(-)
 create mode 100644 hw1_liu.Rmd
 create mode 100644 hw1_liu.html
[howardliu@Howards-Mac-Studio poli502 % git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 20 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 820.39 KiB | 11.72 MiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:haoliuhoward/poli502.git
 c385f1f..eeaeaf main -> main
howardliu@Howards-Mac-Studio poli502 %




```



poli502 Public

 Pin
  Unwatch 1

main 1 branch 0 tags

 Go to file
 Add file
 Code

Howard Liu and Howard Liu my first edit		eeaeaaf 18 minutes ago	🕒 2 commits
	README.md	my first edit	18 minutes ago
	hw1_liu.Rmd	my first edit	18 minutes ago
	hw1_liu.html	my first edit	18 minutes ago

README.md 

poli502

This is the repository for Howard Liu's POLI502 assignments.



6. We want each homework submission in standalone folders. So this is what it should look like:



poli502 Public

 Pin
  Unwatch 1

main 1 branch 0 tags

 Go to file
 Add file
 Code

Howard Liu and Howard Liu folder		c21588e now	🕒 4 commits
	hw1	folder	now
	.DS_Store	folder	now
	README.md	macbook	18 minutes ago

README.md 

poli502

This is the repository for Howard Liu's POLI502 assignments. macbook.

Remember to **name your github repo as POLI502_yourLastName (POLI502_Liu)**

Rmarkdown (rmd) and Quarto (qmd)

What is Rmarkdown?

R Markdown is a framework that combines code, commentary, and results into a single document. It's a variant of Markdown, a markup language that allows users to create plain text files with formatted text, images, headers, and links. R Markdown documents can be used for many purposes, including:

- Communication: Sharing information with decision makers or collaborating with other data scientists
- Data science: Using R Markdown as a lab notebook
- Reproducibility: R Markdown documents are fully reproducible, making them useful for iterative processes where data or models are updated and results need to be compared

R Markdown documents can include the following types of content:

- YAML header: An optional header surrounded by `---`
- R code chunks: Surrounded by ````r`, these can be executed by clicking the icon in the RStudio IDE, which displays the results inline with the file
- Text formatting: Including simple text formatting and inline code, which allows users to directly insert short pieces of code into the text

R Markdown documents can be outputted in many formats, including HTML, PDF, MS Word, Beamer, HTML5 slides, Tufte-style handouts, books, dashboards, Shiny applications, scientific articles, and websites.

What is Quarto?

Very similar but offers more benefits (from [Why should I try out Quarto?](#))

Low-Pain - Quarto is designed to be compatible with existing formats. For example you can render most .Rmd and Jupyter Notebooks with Quarto without modification. This certainly helps with folks making the transition to Quarto – or just trying it out – because they won't have to modify much of their existing work.

The Gain - Quarto is designed to be useful to anyone who wants to create reproducible documents - Your R Markdown doc is also compatible with Python (and bash, Julia, C, SQL), but where R Markdown docs use R and knitr to render your doc, **Quarto does not require R**. Quarto runs computations into separate pluggable language "engines", which helps make this **cross language functionality** easier to support.

Any regular user of R Markdown knows how great it is. It's a lovely notebook (especially in the RStudio IDE or Workbench!), it plays well with version control, which makes it handy for collaboration. I feel it's a standout example today of a great tool for creating reproducible research. And then add in the universe of related tools like bookdown, blogdown, pagedown, distill, pkgdown and more...But R Markdown is really designed for R users and is R first, that is not where Quarto is coming from. Quarto is developed by the same core team as R Markdown and will get the same support and extensions.

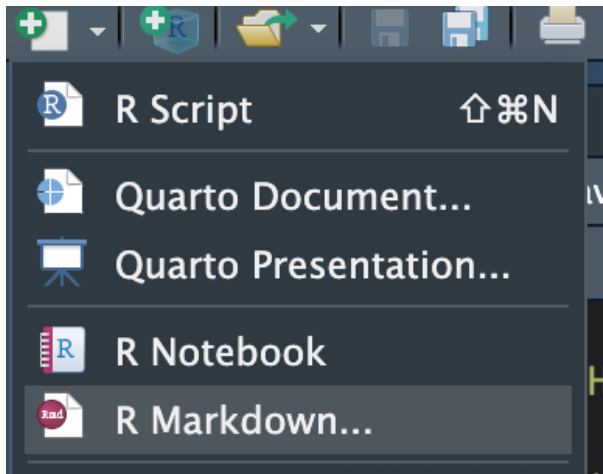
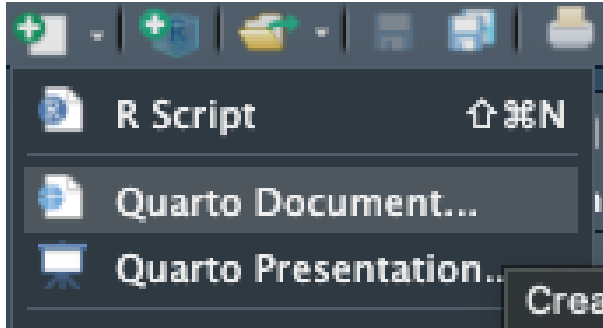
If your goal is to create a document using the language of your preference – one that's reproducible, plays well with version control, one that's easier to share with people who prefer to do their work in a different language – Quarto will be a really handy tool for that purpose.

That being said, R Markdown is not going away. If you prefer it, keep using it.

0. Download and Install Latex

If you would like to create PDF documents from R Markdown, you will need to have a LaTeX distribution installed (can be downloaded [here](#)). R Markdown/Quarto users may also install TinyTeX, which can be installed through RStudio.

1. Create a Quarto file



2. Edit your rmd file

The command below loads the data.

```
# world.data <- read.csv("world.csv")
```

The you can write your interpretation here. You can make them look **nice** by separating lines: Here is the answers:

- (A) XXXX
- (B) YYYY
- (C) ZZZZ
- (D) AAAA

Or you can list:

```
* XXX
```

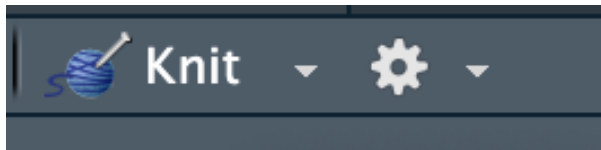
- * YYY
- * ZZZ

You can also do hyperlinks like this:

- * [Github](#)
- * [StackOverflow](#)
- * [Reddit](#)

3. Render/Knit your rmd file

You can knit/compile a html (prettier) or a pdf (more formal) as your output.



4. Some Knitr functionalities can be found here

- [Knitr with R Markdown](#)

Homework submission rules

From now on, all submissions need to have four components:

1. One Quarto (qmd) file, so we can verify your code
2. One pdf
3. Upload your submissions on your private github page. Each homework should be located in one folder (hw1) on your Github repository.

Lastly, you can find my Quarto code using this link [here](#)