

Computational Social Science

File management and Github

Dr. Thomas Davidson

Rutgers University

February 2, 2022

Plan

- ▶ Course updates
- ▶ File management and Github

Course updates

- ▶ Homework 1 released on Github Classroom
 - ▶ Follow link on Canvas in the Module for this week
 - ▶ Select your Rutgers NetID
 - ▶ This will take you to a Github repository
- ▶ I will show you how to download it at the end of the lecture

File management



Source: The Verge, 2021.

File management

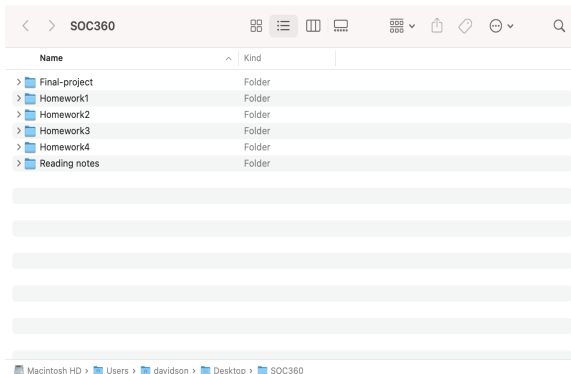
- ▶ Organizing the class materials
- ▶ Navigating files using RStudio

Organizing your files

- ▶ Make a directory to contain all materials for this class
- ▶ Store this somewhere practical
 - ▶ e.g, `/Users/me/Documents/SOC360`,
`/Users/me/Desktop/Classes/SOC360`
 - ▶ Do not just leave files in your Downloads directory!

Organizing your files

- ▶ Within this directory, make a separate directory for the class materials, readings, and homework assignments. It might look something like this.



Github

- ▶ Github is a version-control system
 - ▶ This allows you to easily control and manage changes to your code (similar to Track Changes in Word)
 - ▶ It can facilitate collaboration
 - ▶ Version-control helps to ensure reproducibility
 - ▶ It makes it easy to share code
- ▶ Github is *not* designed as a place to store large datasets (100Mb file size limit)

Terminology

- ▶ A Github *repository* (or *repo* for short) contains all files and associated history
 - ▶ A repository can be public or private
 - ▶ Files should be organized into folders
 - ▶ Github can render Markdown files (suffix `.md` in Markdown), useful for documentation
- ▶ Github repositories exist online and you can *clone* them to your local computer

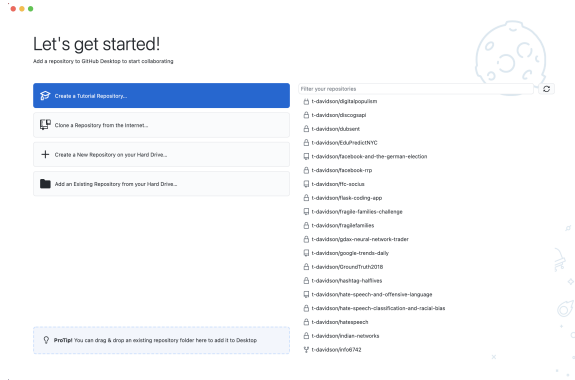
Using Github

- ▶ You can interact with Github in several different ways
 - ▶ Github Desktop (recommended)
 - ▶ Through your browser (not recommended)
 - ▶ Using the command line
 - ▶ RStudio integration
 - ▶ See <https://happygitwithr.com/index.html> for a guide

Install Github Desktop

- ▶ Install Github Desktop by using this link:
<https://desktop.github.com/>

Using Github Desktop



Using Github Desktop

Clone a Repository

×

GitHub.com	GitHub Enterprise	URL
------------	-------------------	-----

Repository URL or GitHub username and repository
(hubot/cool-repo)

https://github.com/t-davidson/css-spring-2022

Local Path

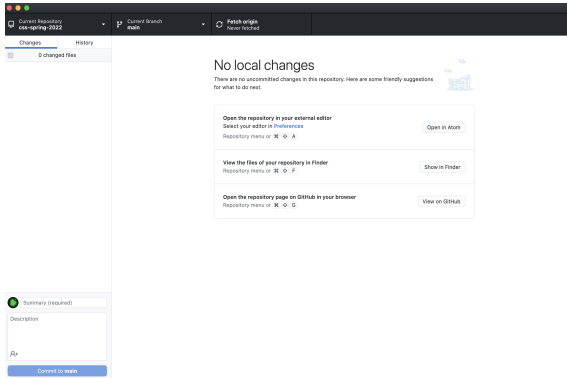
/Users/davidson/Desktop/SOC360/css-spring-2022

Choose...

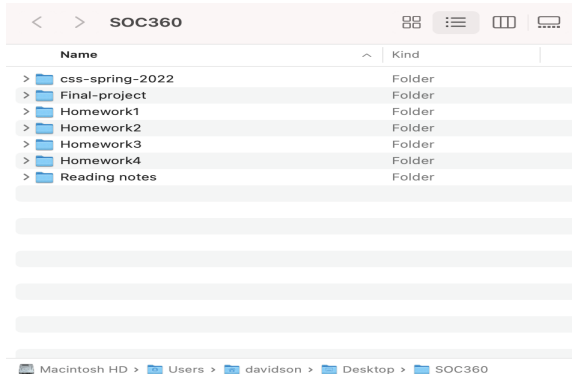
Cancel

Clone

Using Github Desktop



Using Github Desktop



Opening this .Rmd file

- ▶ Navigate the repository you just created and open this .Rmd file.
- ▶ The file will be located in the following *path*:
 - ▶ `~/SOC360/css-spring-2022/slides/lecture5-file-management`
 - ▶ The ~ stands in for the location of the SOC360 directory on your computer.

Navigating files using RStudio

Run the `getwd()` command to show the current working directory in R.

```
getwd()
```

```
## [1] "/Users/davidson/Desktop/Classes/css-spring-2022/slides"
```

Navigating files using RStudio

Run the `setwd()` command to change your working directory. You can then run `getwd()` to verify it has changed.

```
setwd("/Users/davidson/Desktop/")  
getwd()
```

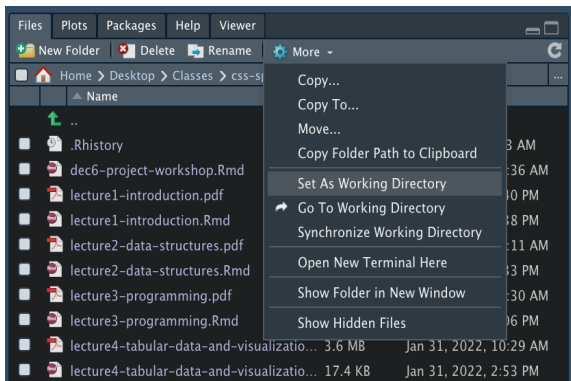
```
## [1] "/Users/davidson/Desktop"
```

Navigating files using RStudio

- ▶ RMarkdown defaults to the directory where the `.Rmd` file is contained.
 - ▶ If you run `setwd()` it will change within a chunk, but other chunks will revert back to the working directory.

Navigating files using RStudio

You can also use the Files pane in RStudio to navigate folders and set a working directory.



Navigating files using RStudio

The working directory is important when considering loading files. You can use the `list.files` function to see a list of the files in the current directory.

```
setwd("/Users/davidson/Desktop/Classes/css-spring-2022/")  
list.files()
```

```
## [1] "file_outer.txt" "images"          "README.md"      "slides"  
## [5] "syllabus"
```

Using file paths

- ▶ The working directory is important because it defines a path we need to use to load files. Different files will fall into one of three groups:
 - ▶ File contained in the working directory.
 - ▶ Files contained in outer directories.
 - ▶ Files contained in inner, nested directories.

Files contained in the working directory

If a file exists in the working directory, you will see it when running `list.files`. It can be loaded by using the file name.

```
library("tidyverse")  
f <- read_file("file.txt")  
print(f)
```

```
## [1] "I'm a file in your current directory!"
```

Files contained in an outer directory

If a file is contained in an upper level directory, you need to use the `..` to escape the current directory. For each step out from the current directory you need to add another `/..` to the file path. In this case, the file is contained one level above the current directory.

```
f <- read_file("../file_outer.txt")  
print(f)
```

```
## [1] "I'm a file in an outer directory!"
```


Files contained in an inner directory

If a file is contained in an inner directory, you need to add the name of the directory to the file path.

```
f <- read_file("nested/file_nested.txt")  
print(f)
```

```
## [1] "Hi, I'm a file in a nested directory!"
```

Files contained in an inner directory

If a file is contained in an inner directory, you need to add the name of the directory to the file path.

```
f <- read_file("nested/nested2/file_nested2.txt")  
print(f)
```

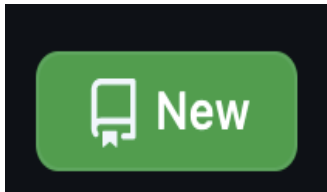
```
## [1] "Hi, I'm a file in a nested directory, nested inside another dir"
```

Errors

- ▶ It is common to get errors if you misspecify a path when trying to load a file. You will get an error like the following:
 - ▶ Error: 'filename' does not exist in the current working directory ('directory').
- ▶ If this happens, check whether the file is in your current working directory.
 - ▶ You can always use your Files tab or your normal file viewer to verify the location.

Making a new Github repository

Go to Github and click the New button in the top-left corner.



Making a new Github repository

Give the repository and informative name.

Create a new repository

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

Repository template

Start your repository with a template repository's contents.

No template ▾

Owner *

Repository name *


t-davidson ▾


 / test-repository ✓

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

Description (optional)

This is a test

☒  **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.

Making a new Github repository

Check the Add .gitignore box and select R from the dropdown. This will prevent unnecessary files from being committed (more later).



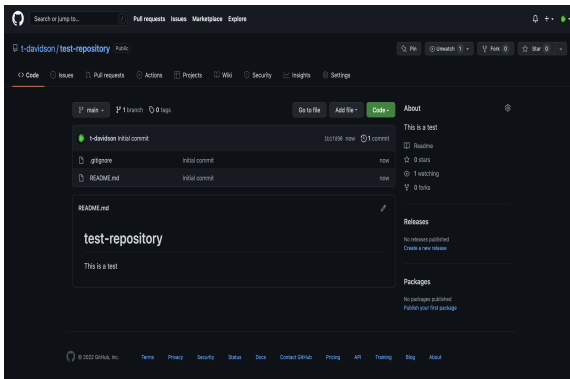
Making a new Github repository

When you're ready, click `Create repository`.



Making a new Github repository

You will be taken to the Github page for your repository.

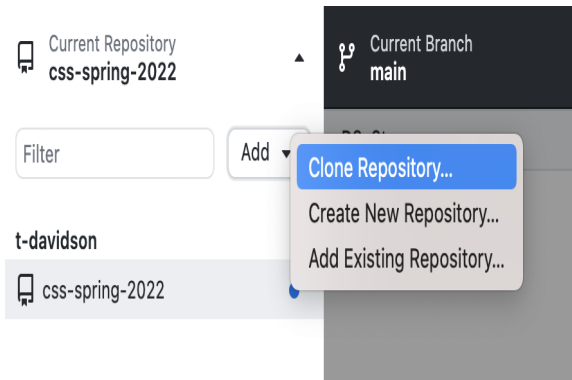


Cloning a new repository

- ▶ You can clone this repository to your computer by copying the URL at the top,
e.g. `https://github.com/t-davidson/test-repository`
- ▶ Cloning creates a local version of the repository.

Cloning a new repository

Go to Github Desktop and click the Add dropdown and select Clone Repository...



Cloning a new repository

Paste the URL into the relevant box and select the path. In this case, the repository will be contained within the SOC360 folder.

Clone a Repository

GitHub.com

GitHub Enterprise

URL

Repository URL or GitHub username and repository
(hubot/cool-repo)

Local Path

Cancel

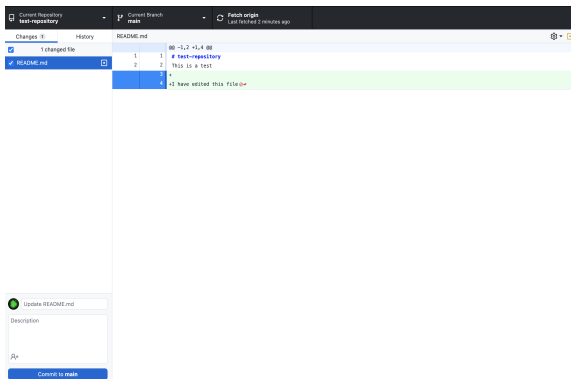
Clone

Editing the repository

- ▶ Navigate to the directory and open the README.md file
- ▶ Add some new text to this file
- ▶ Return to Github Desktop

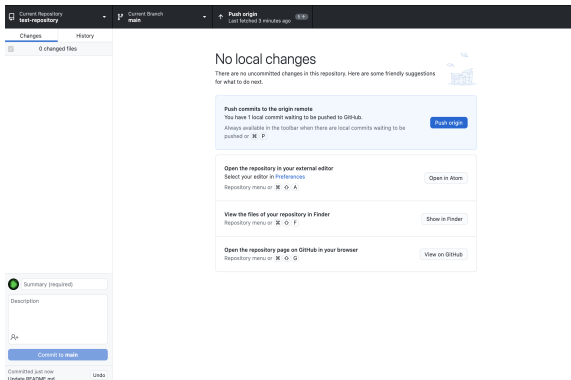
Committing a change

You can see the file has changed. Click **Commit** to **main** to prepare to update the online version using a *commit*.



Committing a change

You will now see that there are no uncommitted changes.





Pushing a change

Finally, click Push origin to *push* the changes

Push commits to the origin remote

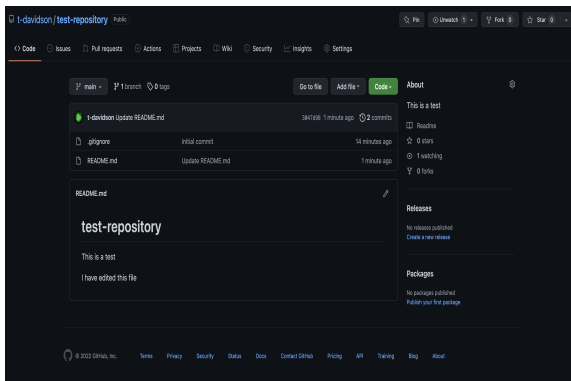
You have 1 local commit waiting to be pushed to GitHub.

Always available in the toolbar when there are local commits waiting to be pushed or  

Push origin

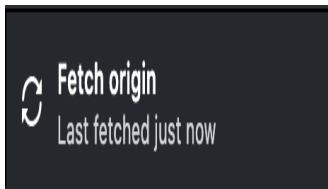
Verifying the update

Refresh the Github page in your browser. You should now be able to see the update.



Pulling changes

- ▶ Just like you use `push` to send your changes from your computer to the internet, you need to use `pull` to bring changes from the internet to the local version.
- ▶ Before each lecture, navigate to the class Github repository and click `Fetch origin`. This will allow you to download the new slides and other files.



Github in the command line

- ▶ Let's say you want to make changes to a repository, in this case adding a single file called `myfile.txt`:
- ▶ Make changes to `myfile.txt` and save the file.
- ▶ `git status` will show information about the status of your repo.
- ▶ `git add myfile.txt` will stage the file to be added to the online repo.
 - ▶ Avoid using `git add *` (only check files you want in Github Desktop)
- ▶ `git commit -m "Adding a new file"` commits the file to the repo, along with an informative message.

Github in the command line

- ▶ `git push origin main` then tells Github to push the local changes to the main branch of the online repository
 - ▶ Conversely, `git pull origin main` will pull the latest updates from your main branch to your local machine
- ▶ Now visit the web page for your repository and you should see the changes.

Github Classroom and homework assignments

- ▶ We will be using *Github Classroom* for the homework assignments
 - ▶ You will each have template repository containing the homework
 - ▶ Clone this repository using Github Desktop
 - ▶ Submit your homework by pushing your updates
- ▶ Full instructions are contained in the repository

Github Student Developer Pack

- ▶ If you haven't already, log in and apply for the Github Student Developer pack
 - ▶ <https://education.github.com/pack>
- ▶ This allows you to make unlimited private repositories and gives access to many other tools

Next week

- ▶ Application Programming Interfaces (APIs)
- ▶ Collecting and analyzing data from Spotify and other platforms

Advanced Github

Branches and merging

- ▶ A *branch* consists of a particular version of the repo
 - ▶ All repos start with a single branch called *main* (formerly *master*)
 - ▶ You can create separate branches for particular tasks
 - ▶ This is particularly useful for collaboration
 - ▶ You can then *merge* the branch back into main
 - ▶ But be careful of *merge conflicts*
- ▶ A *pull request* is a mechanism for merging content into a repository
 - ▶ This can enable the code to be reviewed before it is integrated
- ▶ The *issue* function can be used to note any issues with the code and to bring them to the repo owner's attention (e.g. <https://github.com/tidyverse/ggplot2>)

Advanced Github

Forks

- ▶ A *fork* is a copy of another repository (usually from another user)
 - ▶ This allows you to easily copy the repository and modify it without changing the original content