R projects and Connecting with Github

When considering our data science workflow, we want to have a seamless way to

- work on our project
- utilize version control
- use local file paths for ease of sharing code
- collaborate with others

All of this can be accomplished using R projects with git and github!

R Projects

First, it is important to know that we are going to be reading in some files *locally*. That is, from a folder somewhere on your computer (as opposed to reading a file in from the web). That means we need to be able to tell R where a file exists on our computer.

Locating Files

How does R locate the file?

- We can give a *full path name* to the file
    - ex: `C:/Users/jbpost2/Documents/Repos/ST-558/datasets/`
    - ex: `C:\\\\Users\\\\jbpost2\\\\Documents\\\\Repos\\\\ST-558\\\\datasets`

Note: The `\` character is called an *escape* character. This allows us to use different symbols and things when we are working within a **string** of text. For instance, `\n` is a line break. The `\` tells `R` to interpret the next character(s) in a different way than usual.

For example (cat is kind of like a different version of a print function):

```r
cat("Hi my names is Justin. I work at NC State.")
```

```
Hi my names is Justin. I work at NC State.
```

with the `\n` in there we get a line break:

```r
cat("Hi my names is Justin.\nI work at NC State.")
```

```
Hi my names is Justin.
I work at NC State.
```

Therefore, when we specify a path to a file as a string, if we try to use a `\` we actually need two `\\` so that R knows we actually want a `\`! Confusing I know. But, we can just replace `\` with `/` in paths to files to avoid that.

- Full path names are not good to use generally!
    - If you share your code with someone else, they don't have the same folder structure, username, etc.
    - Instead, use a *relative* path. That is, a path from `R`'s current working directory (the place it looks by default)
- Determine `R`'s working directory via `getwd()` (get working directory)

```
getwd()
```

```
[1] "C:/Users/jbpost2/repos/ST-558/Notes"
```

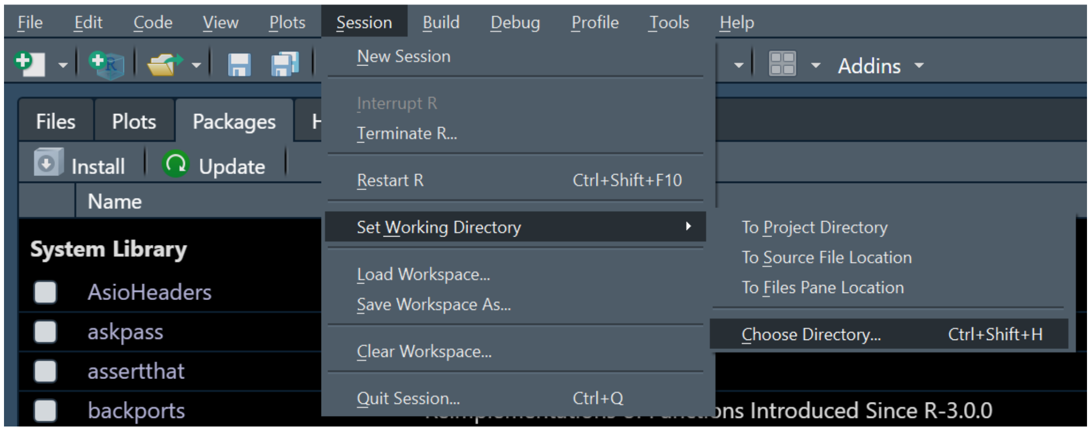
Now if I had my file in the same folder as my working directory, I don't need to use a full path as R is looking in that folder by default. If I had the file `chickens.csv` in my working directory, I could tell R where it is via something like:

```
read.csv("chickens.csv")
```

- Great! How do we set that working directory? Via `setwd()` (set working directory)

```
setwd("C:/Users/jbpost2/Documents/Repos/ST-558/datasets/")
```

- We can also set it via menus:



- Ok, but our goal is to share our code with others so they can run it. We could say to our collaborators, "just to update that one line of code to change your working directory and make sure to have all the files in the same directory that we use for this analysis." Clearly, that isn't an efficient way to work....
- Instead we can use R projects!

Using an R Project

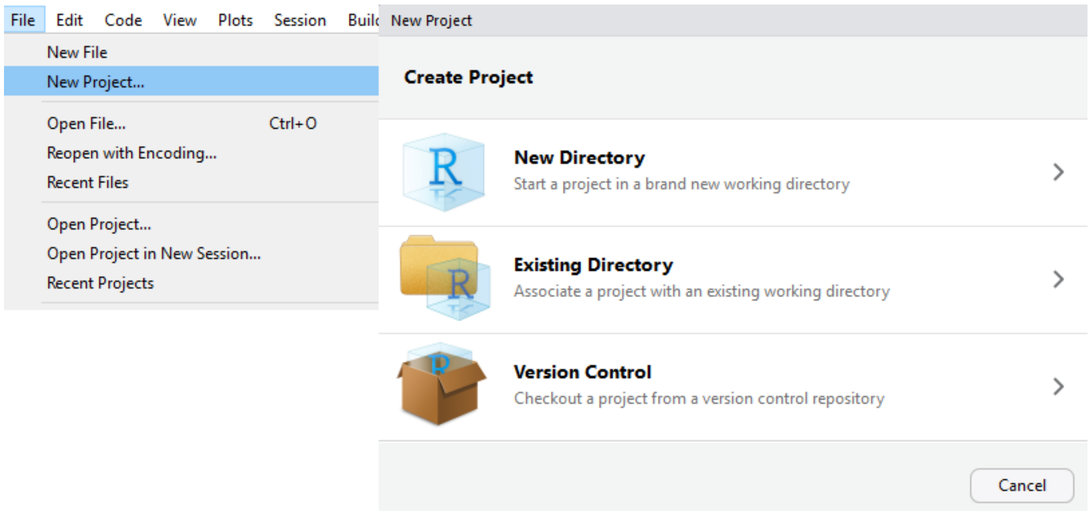
- As we often have many files associated with each analysis, it can make keeping analysis separate difficult. The project feature in RStudio is made to alleviate this!

R projects provide a straightforward way to divide your work into multiple contexts. Each with their own:

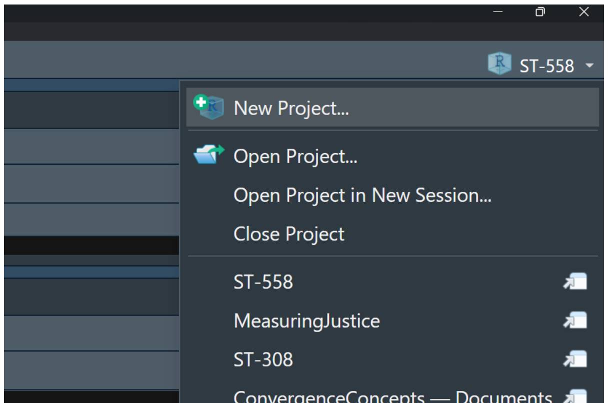
Each with their own:

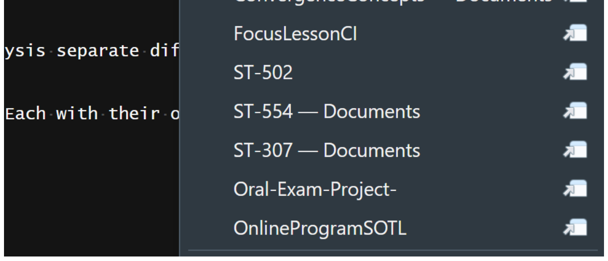
- Working directory
- Workspace (environment with stored data/objects)
- History
- Folder structure & source documents

They are very easy to create!



- When you create an R project, you might note that it gets associated with a directory (or folder or repo). That folder is what the project uses as the working directory.
    - This is important! This means if we can share an entire folder (with subfolders and everything else in the same *relative* place), another user can pick up an R project and the paths to things should work! (Assuming we've used relative paths for everything.)
    - That's exactly how we'll use R projects with github! Github is a remote folder (that can have subfolders and what-not). If we associate an R project with that folder and upload that, another user can download it and open that R project, allowing them to work seamlessly!
- You might create a new project for materials related to this course or for each homework assignment etc. It is up to you how much clarity you want on a specific folder/project you are working on.
- Note you can quickly switch between projects in the upper right hand of RStudio or via the File menu.




```
ysis separate dif

Each with their o
```

Connecting Git/GitHub with RStudio

Ideally we want to document our process, easily collaborate, and widely share our work

- For reproducibility, ideally we would save different versions of our analysis, write-up, etc. along the way
- Remember that git is a version control software to help:
    - Track the changes we `commit` to the files
    - Allow multiple users to work on the **same** project
- github is a hosting service that allows us to do Git-based projects on the internet and share them widely!

Recall our basic workflow. First we create a repo on github (remotely). We then associate a folder on our local computer with that repo using git. Then we:

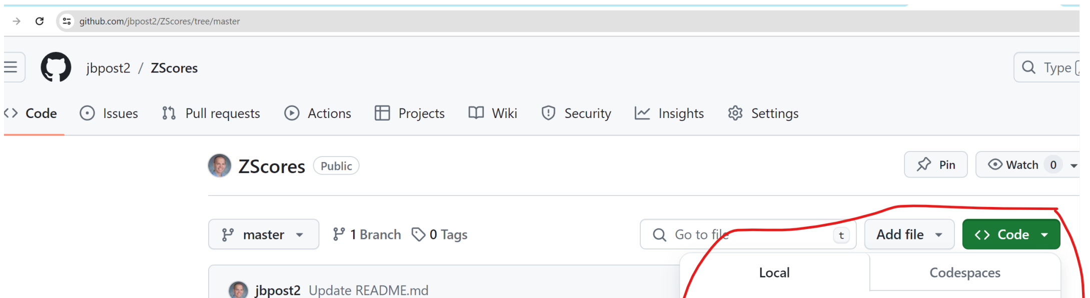
- Pull down most recent files (`git pull`) or do initial download (`git clone`)
- Add files you want to keep changes to (`git add`)
- Commit to the changes (`git commit`)
- Push the changes to the remote repo (`git push`)

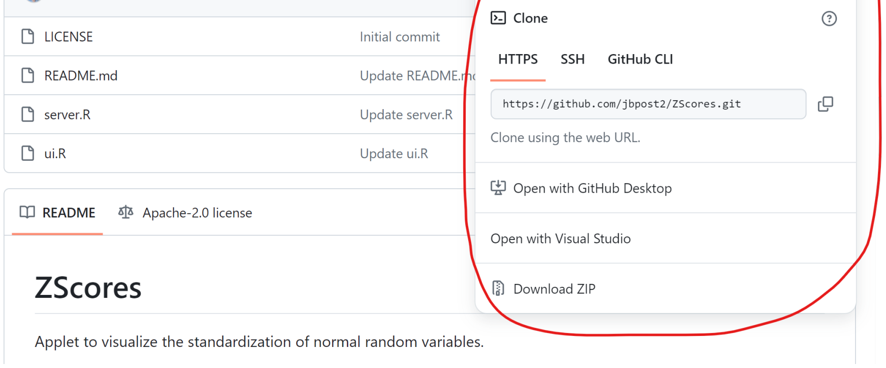
Let's go through some explicit steps to do this! **Make sure you've downloaded git to your machine already (see earlier material on git/github)!**

Cloning a Repo & Working in RStudio

We don't want to use github.com web interface as that is inefficient.

- We can `clone` the repo (i.e. download the entire repo locally).
- Repo main page has a green button. Click on that.
    - Can download a zip and unzip it to an appropriate folder





https://github.com/jbpost2/ZScores.git

ZScores

Applet to visualize the standardization of normal random variables.

- Better to clone the repo via the URL and use RStudio! Open RStudio,
    - go to File –> new project
    - select from version control
    - choose Git
    - paste in the repo link
    - select a directory to save this repo in
    - hit create project!
- Now have the files locally and this associates an R project with a git repo!
- Try it with my repo here: `https://github.com/jbpost2/ZScores.git`
    - You won't have access to push files to this repository though!

Try this out on your own! Create your own repo on github. Then try to download it as an R project.

Communication Between Github and RStudio

We need to make sure RStudio and github can communicate. This can sometimes be tough to get working! Do the following:

- Modify a file in the github repo you just created and downloaded (the one you own).
- Go to the `Git` tab in your `Environment` area
- You should see any files with changes
- Can `add` files that you'd like to `commit` up to the remote repo
- Click on all of the boxes (equivalent to `git add -A` when using the command line) and click the `Commit` button
- A window pops up with a comparison of files. When satisfied, write a commit message in the box and click the `commit` button (equivalent to `git commit -m "message"` when using the command line)
- Hit close on that window
- Click the push button in the top right (equivalent to `git push` when using the command line)
- You may be prompted to log-in in some way. If not, repo on gitub.com should show the changes!

Using the Command Line Interface (CLI)

When working by myself on a repo, I'm not worried about merge conflicts with other people's changes. As such, my workflow is as follows:

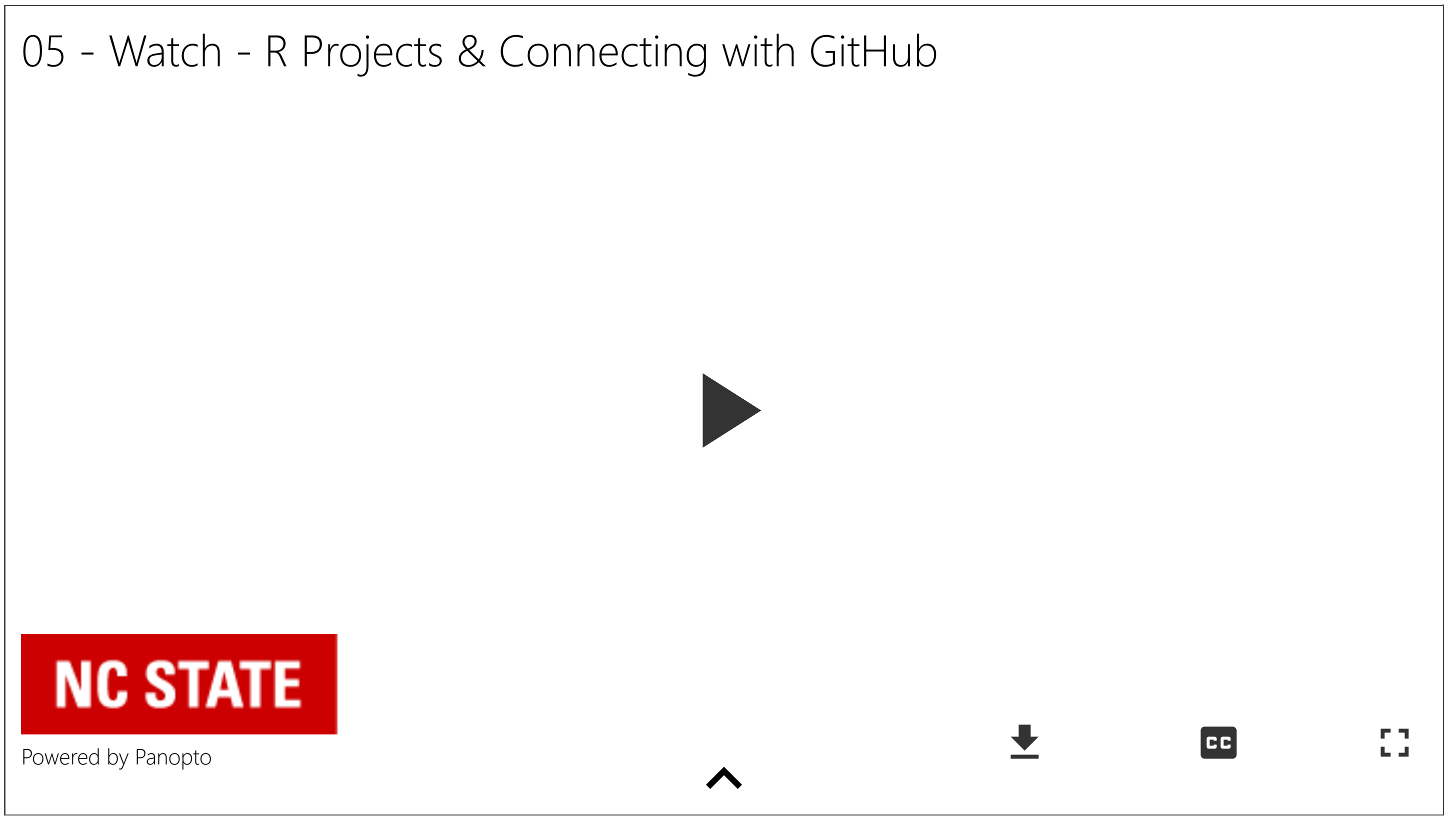
- Open the appropriate project in RStudio
- Go to the Terminal and type `git pull`
- Work… at a good spot for saving, back to the terminal
- Type `git add -A` to add all files that have been modified
- Type `git commit -m "Message"` to stage a commit
- Type `git push` to push the local changes to the remote repo

Creating a Repo From an Existing R Project

Sometimes you'll have an R project that already exists but you don't have a corresponding repo on GitHub. The easiest way to get that project into a repo is to do the following:

- Create a new repository on GitHub
- Clone it to your computer via RStudio new project (in a different folder than your current project)
- Move all the files from the R Project you already have into the new project folder
- `add`, `commit`, and `push` up the files

Please pop this video out and watch it in the full panopto player!


05 - Watch - R Projects & Connecting with GitHub

NC STATE

Powered by Panopto