WHY USE GITHUB?

There are a number of reasons, the two that we'll be most interested in:

(1) It's a great place to store sequential VERSIONS of a project in a safe place and in a really organized way

(2) It's ideal for data science collaborators

SOME NOTES ON GITHUB

"Repositories" are basically the github equivalent of an RStudio "Project" - it's just a place where you can easily store all information/data/etc. related to whatever project you're working on.

When we create a Repository in github, and have it communicating with a Project in RStudio, then we can easily get (pull) information from github to RStudio, or push information from RStudio to github where it is safely stored and/or our collaborators can access it. The beauty is that it keeps a complete history of updated versions that can be accessed/reviewed by you and your collaborators.

Other important things: As a student, you get a limited number of private repos. If you need more, you can always upgrade. This can be very important when you're working on projects with confidential data/analyses/etc (yay open science, but you don't always want to share everything all the time).

We'll just be using publicly available repos.

First, make a folder on your computer (maybe in your ESM 206 folder, or your home directory) called 'github'. Just have a really clear place where you're going to put your git-related business.

Now let’s get started:

A. GET SOMEONE ELSE'S STUFF FROM GITHUB INTO YOUR OWN REPO, WORK WITH IT IN RSTUDIO, THEN PUSH IT BACK

Sometimes, you'll want to use a model/data/etc. that someone else has stored on github. Let's say that's a person you don't know, but luckily they have a public repo because they want a lot of people to use their model. How can we get information from *their repo* into *our github* and then *work with it in RStudio* before *sending out updated version back to github*? That's what we'll do first.

This requires a couple of steps:

1. **FORK.** As you can imagine, people you don't know usually don't want you to be able to update their files without them knowing it. That's why you can't actual change a file in a 'master branch'

unless you've been added as a collaborator by the "owner" (more on collaborating later). So instead of using and changing their originals, you need to ***fork*** their repo first. This essentially makes a copy of all information in their repo, and stores it as your own.

- In github, navigate to the user andrewheiss's github page to find the 'Harry-Potter-aggression' (you can search for it)
- Press '**Fork**' in the top right
- Now navigate back to your github account (if you click on your user picture in the top right, it'll bring down a menu where you can select "Your repositories"), and you'll see the forked repo exists in your world where you can mess with it

2. **CLONE.** Once you've forked a repo into your own github, you probably want to start working with it in RStudio. How?

The easiest way is to get the repo URL from github, THEN create the corresponding project in RStudio that it'll communicate with. When you're in your forked repo that you want to work with:

- Click on 'Clone or Download'
- Select and Copy the ENTIRE URL
- Open a new RStudio session
- Choose New Project > **Version Control** > **Git** > Paste the URL from github into the 'Repository URL' bar > Press TAB to auto-populate the project name > Make sure you're in the subdirectory you want to be in > Select 'Open in a new session' > Create project!

Now, it kind of seems like you're back to working in an RStudio project like normal, right? And all of the files that WERE in the repo you forked are now in your working directory.

But we **also** have version control power now.

What does that mean? Let's make some changes in RStudio and find out.

- Create a new markdown document
- Load the tidyverse
- Read in the Harry Potter aggression .csv file
- Create a simplified version of 'aggressions per mention' just for Draco and Harry
- Make a clustered column graph for the two (follow along)
- Save it into the project folder (working directory)

3. **STAGE/COMMIT/PUSH.** Now your project in RStudio has been updated, but we want this new 'version' of the Project to be stored in github (that's kind of the point). We follow the process of Stage > Commit > Push to send versions back to github.

Once you've saved your .Rmd (feel free to knit, too), it shows up in your working directory.

- Click on the 'Git' tab (up by Environment/History)
- Select files waiting to be committed (stage all of them)
- Click 'Commit'
- Add a short commit message (required), then press 'Commit'
- No error = working
- Press 'Push' - the green 'UP' icon to send the commits back to github
- Refresh your github repo, and see that the file(s) YOU created now exist in your repo. Cool! Cloud storage and version control!

B. MAKE YOUR OWN GITHUB REPO/RSTUDIO PROJECT FROM SCRATCH

You don't always want to use a repo that someone else has created. Usually, you'll have some data that you want to analyze in RStudio, and you might want to use github to help with version control + collaboration.

The easiest way is to CREATE THE REPO FIRST, then follow the steps above (copy/paste URL for github repo to set up R project with version control).

Let's say we want to create a new version control project to explore more about penguins at Palmer Archipeligo (from LTER data portal). You should have the penguins.csv data saved to your computer.

- In github, click on the 'plus' sign up by your user picture, and choose 'New Repository'
- Give it a name (something simple, avoid spaces and symbols, much like naming files/variables in R...like 'palmer-penguins')
- Choose 'Public' (default)
- Choose 'Initialize this repository with a README'
- Click 'Create repository'
- Drop the 'penguins.csv' file into github, 'Commit Changes'

Now, to get working with it in R as a Project with version control:

- Click Clone or Download
- Select the entire URL, and copy to clipboard
- Go to new RStudio session
- Create New Project > Version Control > Git > Paste the URL from github into the 'Repository URL' bar > Press TAB to auto-populate the project name > Make sure you're in the subdirectory you want to be in > Select 'Open in a new session' > Create project!
- Now make some edits (we'll run a Levene's Test and a one-way ANOVA to compare sizes for the 3 penguin species...follow along)
- Save, commit and push the changes. Then they'll show up in your github repo.

We probably won't get here in Week 6, but will include information anyway...

C. HOW DO I START COLLABORATING?

So far, we've been the sole "author" on some repo/project we're working in. Often, we'll want to be collaborating on a paper or data analysis work. If that's the case, we'll want to add a collaborator.

IMPORTANT: Collaborators will be able to push/pull changes directly to/from the repo you share with them, so make sure it's someone you want to allow!

OK, once you have confirmed that you want to add someone as a collaborator:

- Figure out what name or username or email they have associated with their github account
- While in the github repo that you want to add them to as a collaborator, click on 'Settings' (under 'Watch')
- Click 'Collaborators'
- Enter their name(s) (any of the above to search)
- When you find the right one, click 'Add Collaborator'
- And now they can clone and push/pull directly, and you can both be working on the same project

A Practice Task: Creating a repo/project, and adding a collaborator who can work on it with you.

Primary user:

- In github, create a new repo called 'test-yourname' (like 'test-allison' is mine...)
- Drop the "sleep.csv" file into the repo
- Click on 'Clone or Download' to copy the repo URL
- Go to RStudio, create a new Project with version control, linked to the repo you just created using the URL (as described above)
- In RStudio, open a new markdown document
- Write code to prepare a basic scatterplot graph of animal gestation time v. lifespan
- Save the markdown document (and knitted doc, if you want...)
- Stage, commit, and push the files to github (and ensure they show up in the Repo)
- Add your neighbor as a collaborator in the repo

Collaborator:

- Clone/download the repo (same name) that you're now collaborating on
- Update something obvious about the graph (e.g. point color)
- Commit and push your changes back to github
- Check out what happens in the repo you're sharing