ESM 206 Lab 4 - Meet GitHub and work between RStudio & GitHub

Materials by Allison Horst
Bren School of Environmental Science and Management (UCSB)

Keys for all parts of Lab 4 here: https://github.com/allisonhorst/meet-github-esm-206

Why Git?

Two reasons that we'll be most interested in (to start):

- It's a great way to safely store organized versions of a project
- It's ideal for data science collaborators who need to share & edit code together

Some notes:

Repositories (repos) on GitHub are the same unit as an RStudio Project - it's 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 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. It also keeps a complete history of updated versions that can be accessed/reviewed by you and your collaborators at any time, from anywhere, as long as you have the internet.

A couple of general tips:

- Pull frequently (if working with anyone else, and when you start working on a project again, even if on the same device)
- Commit/push in small, meaningful increments and with useful (searchable, descriptive) Commit messages
- The best way to deal with merge conflicts is to avoid creating merge conflicts. Communicate with collaborators so you're not all working on the same piece of code at the same time.

PART 1. Fork & clone an existing repo on GitHub, make edits, push back

a. Go to github.com and log in (you probably are already)
b. In the Search bar, look for repo **allisonhorst/meet-github-esm-206**
c. Click on the repo name, and check out the existing repo structure

d. FORK the repo
e. Press Clone/download, copy the URL, and create a new version controlled project in RStudio (follow along with instructor)
f. Check out the *some_cool_animals.Rmd* document, and the html
g. Add your name to the top of the document (where prompted)
h. BUT WAIT. We have forgotten to add a great image and facts about a very important species - the ringtail cat! Follow along to add a new tab for the ringtail cat, including an image (*ringtail_cat.jpg*, courtesy of the VA zoo) and some fun facts. Like these from the Arizona-Sonora Desert Museum:

    - They can rotate their hind feet 180 degrees for awesome climbing
    - Not endangered, just very elusive
    - Closely related to raccoons, not cats

i. Once you've added the ringtail cat tab, knit (to save & update html)
j. Stage, commit & push all files
k. On GitHub, refresh and see that files are updated. Cool! Now you've used something someone else has created, customized it, and saved your updated version.

PART 2. Create your own repo & version controlled R Project from scratch

a. Go back to your GitHub account
b. Click on the plus sign (upper right, by your profile pic/icon) to create a new repository
c. Name the repo **esm206-lab4-part1-yourinitials** (like esm206-lab4-part1-ah), and select to initialize with a ReadMe
d. Edit the ReadMe (however you want - notice that markdown formatting works & you can see a preview) & commit
e. **Clone** to create a connected R Project in RStudio
f. Create a new R Markdown document
g. Attach the {tidyverse}, {DT}, and {plotly} packages in a hidden code chunk (include = FALSE)
h. Create an interactive plot of Edgar Anderson's *iris* data with {plotly}, showing the output but not the code or messages
i. Create an interactive table of mammal sleep data (*msleep*) using {DT}
j. Knit & save your .Rmd
k. Stage, commit & push back to GitHub

PART 3. Git enabled R Projects with subfolders

a. Create a new repo named **esm206-lab4-part3-yourinitials**, like *esm206-lab4-part3-ah*
b. Clone to work locally in RStudio
c. In your local project folder, create subfolders ‘data’ and ‘final_graphs’
d. Drop the file *disease_burden.csv* into the ‘data’ subfolder
e. In RStudio, open a new .R script
f. Attach the {tidyverse} and {here} packages
g. Read in the *disease_burden.csv* data
h. Stage, commit & push at this point - notice that the empty folder ‘final_graphs’ doesn’t show up (won’t commit an empty folder)
i. Back in the script, follow along with instructor to do some cleaning, wrangling & viz
j. Update your graph with direct labels (using *annotate*) and vertical or horizontal lines with *geom_vline* or *geom_hline*
k. Use ggsave() to write your graph to a .png in the ‘final_graphs’ subfolder
l. Save, stage, commit & push
m. Check that changes are stored on GitHub

END LAB