Introduction to Version Control and Git(Hub)

Data Wrangling and Husbandry

03/09/2020

Version control

- A *Version control* system is a set of software designed to keep track of changes in documents
- It allows one to go back to a previous version in case problems have been introduced
- It facilitates working with others, providing a framework so that everyone can keep track of the various versions
- *Git* is one implementation of a distributed version control system
  - Originally designed for team development of Linux
  - Repurposed by data scientists for their own use
- *GitHub* is a commercial website where Git repositories can be uploaded (*Bitbucket* and *GitLab* are prominant alternatives)

A conceptual introduction to Git

If you work on a project yourself, you might only use three commands after setting things up

- `git add --all` to start tracking new files and to stage changes to already tracked files
- `git commit -am "<message>"` make a local snapshot of the files that you are tracking. Include a message to make clear what you've done since last time
- `git push origin master` publish the latest local snapshot to a remote repository
  - This last gives you a remote backup as well as potentially making your work available to others.

If you work with other people, there are a few more commands and concepts

- `git clone <url>` to get a local copy of a Git repository, whether just to examine or so you can modify it.
- `git fetch <alias>` to synchronize your repository with a remote repository, fetching all the files it has that you do not.
- `git pull origin <branch>` to pull the most recent changes from a remote branch—if you're working collaboratively and a repo was updated on GitHub but you don't have the changes locally
- Issue: a feature of GitHub often used for bug reports
- Pull requests: Used to suggest a code revision via specific code

Git has big upfront costs (in terms of time and effort) but has many long term advantages. After a while it will become a natural part of your daily workflow

Onward to details. First, potentially painful setup

- Setup your free account at https://github.com
  - Your free account will have public (as in anyone can view) repositories
  - Later on, you can almost certainly arrange for unlimited private repositories via https://education.github.com/
  - You can also pay for private repositories

Install Git

- Is Git installed already?
  - Open a terminal window (Command Prompt on Windows) and type `which git` on Mac and Linux or `where git` on Windows. If you get something like `/usr/bin/git` and not just an empty return, type `git --version`. If you get something like `git version 2.9.3` congratulations! You can ignore this slide and the next.
    - Macintosh users might get something like `xcrun: error: invalid active developer path`. In that case just follow the next step

- Install Git
  - Macintosh users: In a terminal window, type `xcode-select --install`
    - There are other options, such as `brew install git` or https://git-scm.com/downloads
  - Windows users: Install from https://gitforwindows.org
    - Select "Use Git from the Windows Command Prompt" during installation
    - RStudio for Windows prefers for Git to be installed below "C:/Program Files"", for example as"C:/Program Files/Git/bin/git.exe"
  - Linux users: Install via a command like `sudo apt-get install git` or `sudo yum install git`

Setup your Git profile

In the terminal window, type (using your own name and the email address you used for your GitHub account)

```
git config --global user.name 'jasonklusowski'
git config --global user.email 'jason.klusowski@rutgers.edu
git config --global --list
```

Test Run Part 1: Make a Repository on GitHub

1. Log in to https://github.com
2. Click green “New repository” button
3. Make a name, select Public and choose Yes to initialize with a README
4. Click the green button says “Create repository” 1, Copy the address to clone using the green “Clone or Download” button

Clone the directory to your local machine

1. Open up a terminal window
2. Possibly change directories
3. Type `git clone  <the URL you copied>`
4. When Git is done
   - `cd <repository name>`
   - `ls`
   - `cat README.md`
   - `git remote show origin`

Make changes locally

1. Add a line to the README file and check how Git feels about that
   - `echo "A line I wrote on my local computer" >> README.md` (or any other way to change the file)
   - `git status`
2. Use `git add -A` to track all the files in the directory
3. Type `git commit -m "My first commit"` to commit your changes (save a snapshot locally) with an informative message
4. Type `git push` to upload the snapshot to GitHub

Confirm that the local change has shown up on the GitHub remote

1. Refresh your browser window that is still open to the GitHub repository
2. Refresh
3. The README should show the new line
4. Clicking on “commits” should show the one with your commit message

Git and RStudio

1. Even if you already have files, first set up the repository on GitHub.
2. In Rstudio, choose File > New Project > Version Control > Git. Put the URL of the GitHub repo in "repository URL", make up a local directory name, and select the directory in which that directory will sit
3. Click "Open in new session" and then "Create Project"
4. If you have files that you have already been working on, move them into the newly created directory

Git and Rstudio Workflow 1/2

- Every time you've made important changes (think many times a day) you should commit your changes:
  - Open the "Git" panel
  - Click the "Staged" box for files that you want to commit
  - Click "Commit"
  - Type a message in the "Commit message" box
  - Click "Commit"

Git and Rstudio Workflow 2/2

- Push your changes to GitHub less often than commits, but still frequently (think a few times a day) and certainly when you are done with a project for a while
  - First click the blue “Pull”" button in the “Git”" tab (in RStudio). This will incorporate any changes a collaborator might have made before you push your changes to GitHub
  - Next click the green “Push”" button to upload your local changes to GitHub.

Avoiding retyping your password: credentials or ssh-key

- Turn on the credential helper
  - Macintosh:
    - Try `git credential-osxkeychain` to be sure that you get a message like `usage: git credential-osxkeychain <get|store|erase>`
    - Type `git config --global credential.helper osxkeychain`
  - Linux:
    - Type `git config --global credential.helper 'cache --timeout=10000000'` which will store your password for 10000000 seconds
  - Windows
    - In the shell, type `git config --global credential.helper wincred`
    - If that fails, download and run the Git Credential Manager for Windows
- The next time you are asked for your account and password should be your last
- To instead use an ssh-key, see http://r-pkgs.had.co.nz/git.html#git-init, Initial set up, steps 4 and 5

GitHub and Collaboration

We will discuss this next week

Git(Hub) Graphical Interface Clients

- RStudio (limited functionality, but suits me fine)
- GitKraken (Windows, Mac, Linux) https://www.gitkraken.com/
- SourceTree (Window, Mac) http://www.sourcetreeapp.com/
- Don't use the GitHub client, which is too restrictive in its attempt to make things easy.

- This material is *heavily* drawn from Jenny Bryan's work at http://happygitwithr.com
- Hadley Wickham also has a nice condensed set of instructions at http://r-pkgs.had.co.nz/git.html