Lecture 14: Introduction to GitHub

Managing and Manipulating Data Using R

Introduction

Libraries and data we will use today

Libraries

```
library(tidyverse)
#> -- Attaching packages -----------------------------------------------------
#> v ggplot2 3.2.1          v purrr   0.3.2
#> v tibble  2.1.3          v dplyr   0.8.3
#> v tidyr   1.0.0.9000     v stringr 1.4.0
#> v readr   1.3.1          v forcats 0.4.0
#> -- Conflicts --------------------------------------------------------------
#> x dplyr::filter() masks stats::filter()
#> x dplyr::lag()    masks stats::lag()
library(haven)
library(labelled)
```

Data frame

```
#load dataset with one obs per recruiting event
load(url("https://github.com/ozanj/rclass/raw/master/data/recruiting/recruit_ev
```

Logistics

(LAST!) Lecture 14: Intro to GitHub

- GitHub can be difficult to get the hang of!
- Learning goals: Develop basic understanding; set up a repo on your local machine; get some practice working with GitHub

Teacher Course Evaluations:

- All but one completed as of this morning!

What we will do today

What is Git and GitHub?

What is Git and GitHub?

Git is the most commonly used version-control system to manage code

- Save drafts of code
- Look back at previous versions
- Undo mistakes
- Track your changes

A project managed in **Git** is called a **Git repository**

- local git repository: the repository tracking changes YOU make on your own machine
- remote git repository: local repositories are MOST often connected to a remote repository tracking changes by ALL collaborators

GitHub is the hosting site/service for **Git repositories**

- Stores your local repos in "the cloud"
- You can store files, share code, collaborate with others
- Who uses GitHub? Netflix, Airbnb, Lyft, Coursera
- Competes with Microsoft's and Google's in-house systems

This course is a **Git repository**!

Git Set-up

Create and Verify a Git Account

You should have created a free git account prior to class.

- If you haven't, please create one now: https://github.com/
- Be sure to verify your account via email sent by GitHub

Share your usernames with me:

- ksalazar3

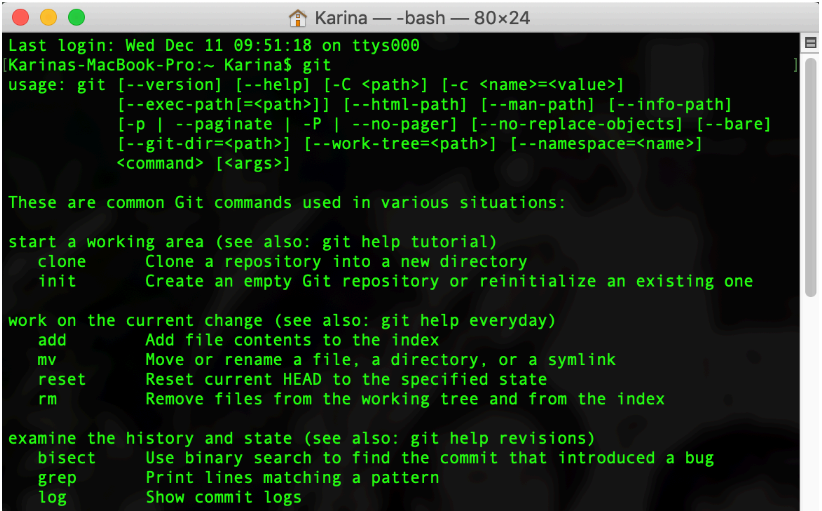
We will setup Git using **Code Academy** instructions: Link

Git Setup for Mac Users

1. Open a Terminal window (command+space bar to open Spotlight Search)

- Terminal is the CLI (command line interface) for Linux and Mac users

1. Type in git to the command line and press enter
2. If you don't already have Git installed, you'll get an warning saying **Git** requires command line developer tools. Click install and agree to terms.



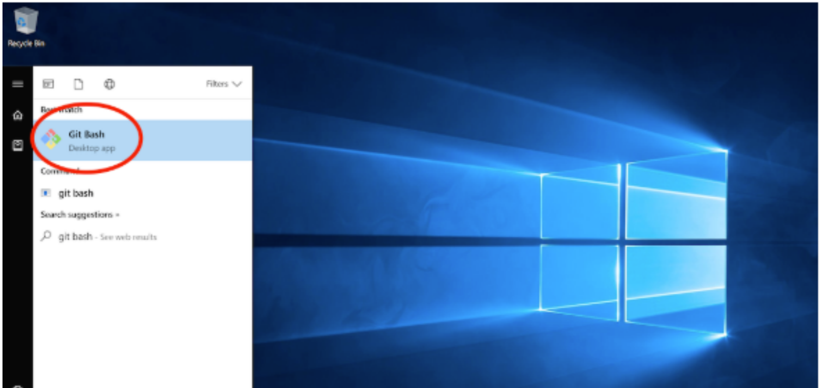
```
Last login: Wed Dec 11 09:51:18 on ttys000
Karinas-MacBook-Pro:~ Karina$ git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone      Clone a repository into a new directory
   init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add        Add file contents to the index
   mv         Move or rename a file, a directory, or a symlink
   reset      Reset current HEAD to the specified state
   rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
   bisect     Use binary search to find the commit that introduced a bug
   grep       Print lines matching a pattern
   log        Show commit logs
```

Git Setup for Windows Users

Warning: I have never worked/installed on a Windows so this may take some time! Mac Users please be patient :)

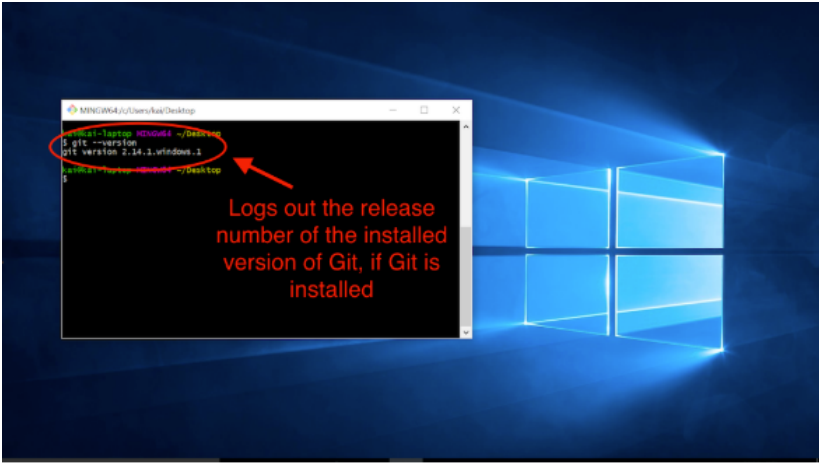
1. Need to first install Git Bash

- *Bash* is the default shell (a specific type of CLI) for Linux and Mac Users so only Windows users need to install this first. Windows default is *Command Prompt*.
- Download and intall Git Bash: https://gitforwindows.org/
- Run the downloaded .exe file and allow the application to make changes to your PC.
- Once it is finished install, check it installed by searching "git bash" in your start menu



Git Setup for Windows Users

2. Open your CLI and verify Git installed
   - Click on the Git Bash icon to open a new CLI window
   - Type in `git --version` and press enter
   - If installed it will return which Git version is installed



```
kai@kai-laptop MINGW64 ~/Desktop
$ git --version
git version 2.14.1.windows.1

kai@kai-laptop MINGW64 ~/Desktop
$
```

Logs out the release number of the installed version of Git, if Git is installed

Authenticating Users in GitHub

There are two ways to authenticate users when making changes to a GitHub repository:

1. HTTPS: Using a URL to "clone" and make changes to a repository
   - Will require you to enter your username and password everytime you make a change
   - Much easier setup but very inefficient/inconvenient
2. SSH (Secure Shell): Using a "keypair" between your computer and GitHub
   - Doesn't require you to enter your username and password everytime you make a change
   - But more time consuing to setup

To make things easier for this introduction to GitHub, we will be using HTTPS. Remember your username and password!

GitHub Work Flow and Basics

Git Workflow

Three components to a Git project:

1. Local working directory
   - This is the area where all your work happens! You are writing Rmd files, debugging R-scripts, adding and deleting files
   - These changes are made on your local machine!
2. Staging Area
   - The area where you list the changes you've made in the workind directory
   - Essentially what changes would you like to keep and save to the repository (or "commiting")
3. The repository
   - This is the actual repository where Git permanently stores the changes you've made in the working directory and listed in the staging area as *different* versions of the project file

Git Basics: Working Directory

First you need to create the working directory on your local machine.

- When working on a collaborative project, one person is usually assigned to creating the repository on GitHub (via browser) and adding all team members as collaborators. Collaborators then `clone` the repository on their local machine.
- You can also create a "folder" on your local machine via the command line, *initialize* it as a git repo, and link it to a remote repository on GitHub

We'll practice doing both!

Basic Git Commands 1. `git clone`: will "clone" a repository into the current directory 1. `git status`: allows you check the status of any changes you've made to the working directory - If you are revising files that already exist in the remote git repository, these changes be listed in red as modified files under `Changes not staged to commit` - If you create a new file that does not already exist in the remote git repository, these changes will be listed in red under `Untracked files`

Git Basics: Staging Area

In order for **Git** to start permanently tracking any changes you've made in the working directory those files need to be added to the working directory

Once you've made changes:

Basic Git Commands

1. `git add filename` : Adds filename (you specify this) to staging area
2. `git status` : Lists filenames in green which have been added to the staging area
3. `git diff filename` : Once a file is added to the staging area you can check differences between the working directory and the staging area
4. `git commit` : Permanently stores changes from the staging area to the repository
    - Always use the `git commit -m "your message here"` option when commiting your changes!
    - The message needs to be in quotation marks; should be brief (up to 50 characters); and in present tense (new to me!)
5. `git log` : logs all commmits made chronologically, uniquely identifies each commit via SHA
    - Helpful when you need to revert your local repository to a previous commit (my safety blanket!)
    - I often need old code (that I've deleted) as a foundation for new code

Git Basics: The Repository

When working on a **remote repository** you will often need to:

“push” your changes

- In order for your collaboraters to see the changes you have made, you need to `push` the changes in your local repository to the remote repository
- Think of as “uploading”

“pull” the changes of your collaborators

- In order for your working directory to contain the changes collaborators have made, you need to `pull` those changes made to the remote repository into your local repository
- Think of as “downloading”

Basic Git Commands

1. `git push`: “Uploads” local repository content to the remote repository
2. `git pull`: Updates the local repository by “downloading” any changes made by collaborators

Git Basics: Other Stuff and “Stylistic Issues”

Always start a git repository with a README file:

```
- Usually includes important information about what the project is, how folks c
- In a remote server, GitHub will display the README file; it's the first thing
- Use other README files as a template!
```

In a collaborative project, you’ll often establish Code of Conduct/Contribution Guidelines together

```
- These are some of the first files to add to a remote repository!
- They set standards for folks using and contributing to the project (necessary
- Generally good practice!
```

Other good practices when working in GitHub:

- Use the issues feature to keep track of bugs, to-do’s, assingments to particular collaborators, etc
- Whenever you make changes to a remote repository, let your collaborators know by issuing a `pull` request
- Sometimes your own work and working collaboratively is not a linear process. Branching and Merging (won’t cover in this intro) are really important for these reasons!

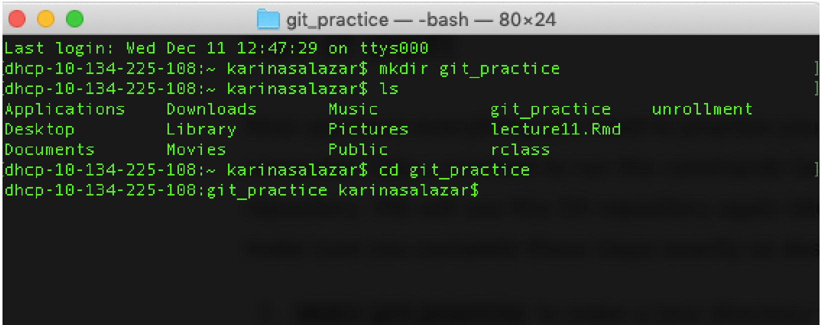
GitHub Practice

Create a Local Git Repository

Let's initialize a Git repository on your local machine to get started!

1. Open your CLI (terminal for Mac Users or Git Bash for Windows Users)
   - CLI should always open to your home directory; indicted by a tilde `~`
   - You can switch directories in the command line via `cd filepath_here`
   - For today and to make things simple, we're going to stay in our home directories
2. Lets make a directory on your local machine (aka a folder)
   - `mkdir git_practice`
3. Change your directory to this new folder so we can work within the new folder
   - `cd git_practice`

Now, your command line should be pointing to this new `git_practice` folder (notice no more `~`)



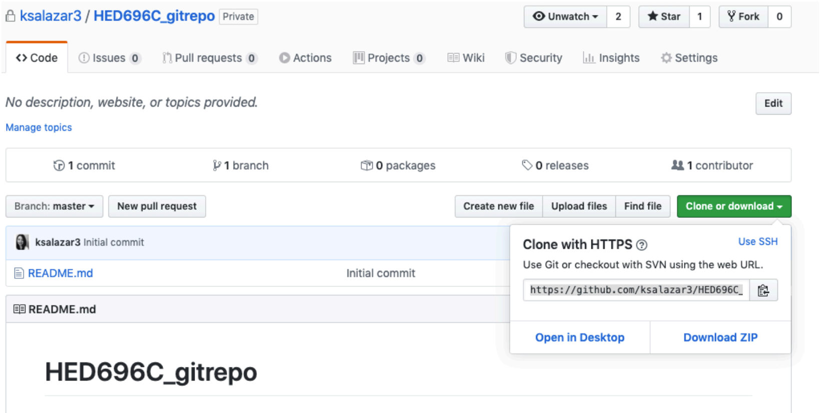
```
Last login: Wed Dec 11 12:47:29 on ttys000
dhcp-10-134-225-108:~ karinasalazar$ mkdir git_practice
dhcp-10-134-225-108:~ karinasalazar$ ls
Applications    Downloads       Music           git_practice    unrollment
Desktop         Library         Pictures        lecture11.Rmd
Documents       Movies          Public          rclass
dhcp-10-134-225-108:~ karinasalazar$ cd git_practice
dhcp-10-134-225-108:git_practice karinasalazar$
```

Create a Local Git Repository

3. Initialize the `git_practice` folder into a *git repository*
    - `git init`
4. ALWAYS start a git repository by creating a README file
    - `echo "Our first read me file" >> README.txt"`
5. Add the new file to the "staging area"
    - `git add README.txt`
6. Commit the changes to your local repository!
    - `git commit -m "add readme file"`

Create a Remote Repository

1. I have created a remote repository for the class `HED696C_gitrepo`
   - Set it as a private repo for now
   - I added you as a collaborator if you submitted your HW with a git username
   - If you all want to create the remote repository we will need to add you! (you can also just follow along)
2. Clone with HTTPS the `HED696C_gitrepo`
   - The URL is located in the top right corner of the GitHub page
   - In your CLI shell: set your directory to where you'd like to copy this project
   - I keep my git projects in my home directory
   - `git clone CLONELINK`


ksalazar3 / HED696C_gitrepo Private

No description, website, or topics provided.

1 commit · 1 branch · 0 packages · 0 releases · 1 contributor

ksalazar3 Initial commit

README.md Initial commit

Clone with HTTPS

Use Git or checkout with SVN using the web URL.

```
https://github.com/ksalazar3/HED696C_
```

HED696C_gitrepo

Create a Remote Repository

Now we'll practice as a class making some changes to this repository!