ECON/DCS 368 Week 2 Notes

Lecture 2: Version Control with Github

- A repository is an entire file and its history
- Any HTML you see you can paste into raw.gethack.com and it will run that HTML code and make it appear in a user friendly way
- R Markdown Presentation
  - Relies on Knitr and Pandoc packages
    - These packages make it more simplistic
    - Can cover a variety of functions, but doesn't cover everything
    - See the R markdown website for helpful guides and information
- A package in code is a file that contains a variety of tools and functions. Note that you must call upon these packages you download when you are starting new code.
- For presentations, use a pdf if you won't have any movement or gifs in the presentation. Speak with prof. If you would like to include such dynamic features.
- Github Presentation
  - Github Desktop allows you to interact with Github using a GUI instead of the command line or web browser
    - GUI is what makes the user view more approachable
  - Git is a version control system that allows developers to see the entire timeline of changes, who made them, when, and why
  - Some common Git commands
    - "Git init" - initializes a new repository
    - "Git clone" - The git clone command copies an existing Git repository
    - "Git commit" - captures a snapshot of the project's currently staged changes. Committed snapshots can be thought of as "safe" versions of a project—Git will never change them unless you explicitly ask it to.
- When you do a pull request, it must be approved by the administrator so you aren't uploading a virus
- Prof. Github Presentation/Demo
  - Git
    - A distributed version control system
    - Git is optimized for data science
  - RStudio has integrated Github and git, which makes everything more seamless
  - Git Commands

    - Stage (or “add”): tell git that you want to add changes to the repo history
    - Commit: Tell git that you are sure these changes should be part of the repo history
    - Pull: get any new changes made on the GitHub Repo by your collaborators or you on another machine
    - Push: push any committed local changes to the GitHub repo
- The Empirical Workflow and Clean Code
    - Prologue: This information will help make your code less messy, which is crucial for getting help and making your work understandable to others
    - Clean Code: code that is easy to understand, modify, debug, etc.
    - Clean Code is underproduced due to:
        - Competitive pressure, deadlines, etc.
        - Nobody else usually looks at your code
    - How to produce good, clean code:
        - Automation - for certain tasks you will do over and over again
        - Version Control
        - Organization of Data and Software Files
        - Abstraction - idea that if it’s a task you have over and over again in the code, you can sort of make a variable shortcut and can just edit that short cut instead of every use of that function
        - Documentation
        - Time/Task management
        - Test-driven development
        - Pair Programming - working with others