31 July 2017

Wharton PhD Tech Camp

Session 1

Alex Miller

Ph.D. Student, Information Systems

Wharton, OID

What are we doing here?

What are YOU doing here?

Goals for the course

- Have a solid understanding of how to conduct empirical research
- Become familiar with tools available for empirical research
- Be able to collect raw data
- Have some idea of how more advanced methods can fit into your research process

Content of the Course

- First few sessions: lots of info
  - Intro, R, Python, HPCC
  - Could potentially be useful, even for people who are familiar with both R and Python
- Remaining sessions:
  - Collecting and working with data
  - API/databases
  - Crawling/Scraping
  - Text processing
  - NLP/Machine learning

Who is this course for?

- Mostly beginners
  - People who have some or little experience working in Python and R
- Kind of intermediate programmers
  - You may get less out of course, but I will try to focus on things you won’t learn elsewhere
- Not so much for advanced programmers
  - Probably not much benefit from the course
  - Come to Session 3 (Friday) if you want a demonstration of how to use the Grid (High Performance Computing Cluster)

End goal

Proficiency
Hopefully the end of this course
I HAVE NO IDEA WHAT I'M DOING
I have some sort Of idea what I am doing
hacker voice"im in"

Format of the course

- 45-60 minutes of slides/lecture/demonstration
- 30-45 minutes of “lab” time
- This should minimize the amount of time you need to watch me typing into a terminal
- I will provide some brief exercises that will allow you to work on some skills covered during lecture

Teaching/Learning Philosophy

- I am going to throw a lot at you
  - My main goal is for you to know what is possible, i.e., what tools/functions exist and how they can make your life easier
  - Ultimately, you should focus on big picture things and use Google for everything else
- You **must** learn by doing
  - Use lab time to sit down and actually learn
  - I will probably tell you to do something that you don’t know how to do: Use me, Google, Stack Overflow
- I will be experimenting with lab format. Bear with me.

Alex’s Soapbox Time

Unsolicited Advice for 1st Years

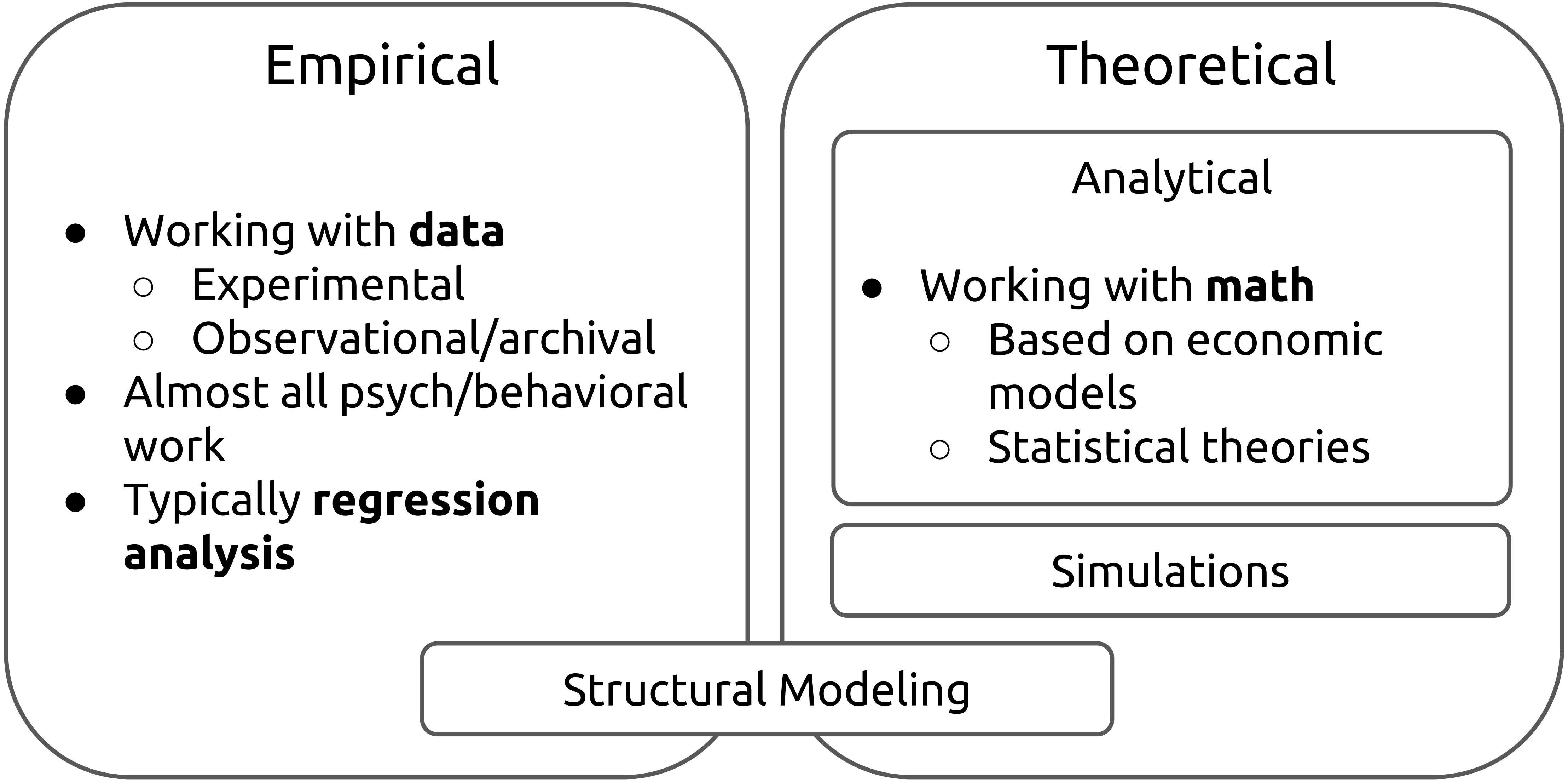
- Your PhD program does not have to be a horrible experience
    - You should (generally) enjoy what you’re doing
- But things WILL be difficult at times
    - Academia is hard work, lots of rejection, very little feedback
- Prioritize your health and well-being!
    - Mental, physical, social
    - Take vacations!
    - Explore Philadelphia!
    - Develop a hobby outside of academics
- Find an advisor that you actually get along with
- When you become rich and famous, be nice to other people

Remainder of class

- Outline:
  - Big Picture
  - Unix/Git
  - R
  - Practice

The Big Picture

What is Empirical Research?



Empirical

- Working with **data**
  - Experimental
  - Observational/archival
- Almost all psych/behavioral work
- Typically **regression analysis**

Theoretical

Analytical

- Working with **math**
  - Based on economic models
  - Statistical theories

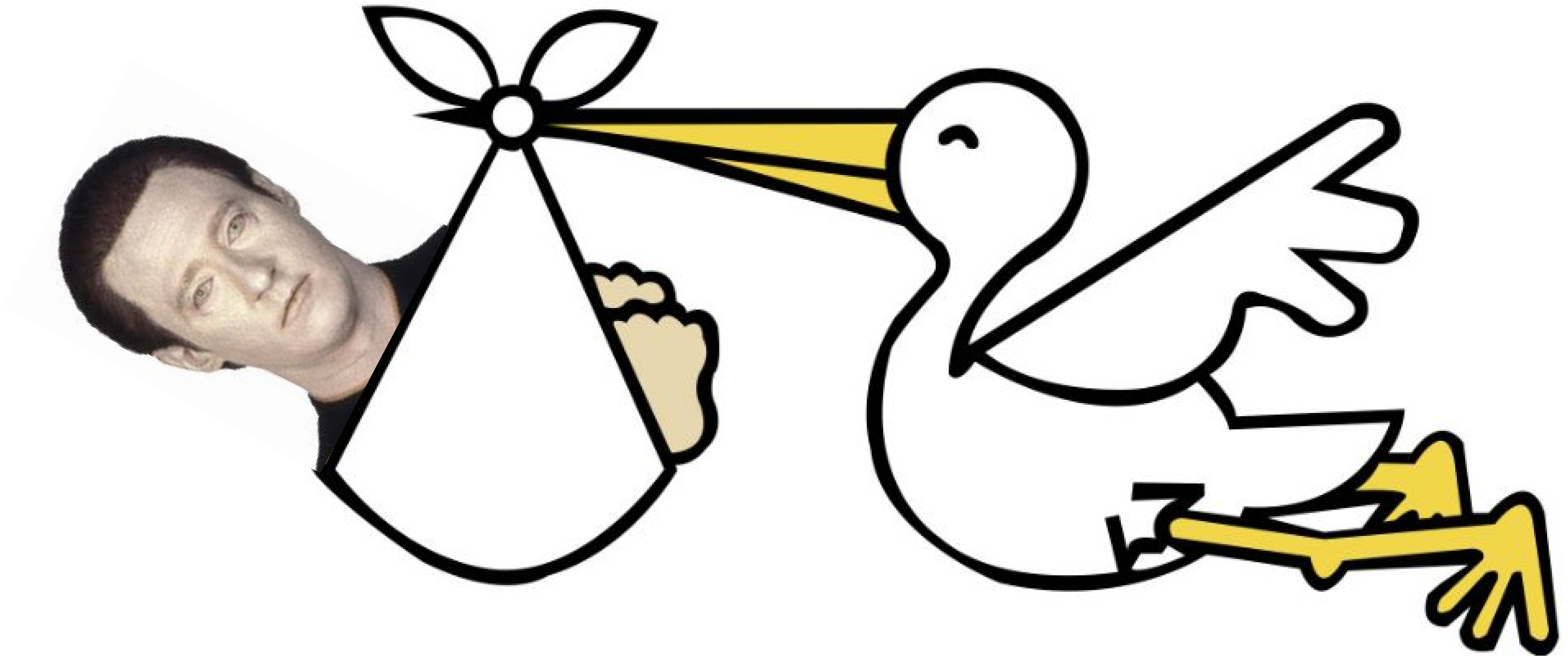
Simulations

Structural Modeling

Regression Analysis

- Trying to show (usually causal) relationship between two (or more variables)
    - Smoking → Lung cancer?
    - College → Higher income?
    - Chocolate and wine → Happiness?
    - X → Y (fill in the blank for your field)
- To answer these questions, you need data

Where does data come from?



Where does data come from?

- Experiments
  - Wharton Behavioral Lab
  - Qualtrics/Amazon Mechanical Turk
- Existing Databases
  - Public: Census, Compustat, etc.
  - Private:
    - Paid access
    - Work directly with private company
  - Use Lippincott Library for data requests (seriously!!!)
- Simulations
  - The Grid (HPCC)
- Scraping

What do you do with data?

- Find it (see previous slide)
- Gather/save it
- Pre-process it
  - “cleaning”, “munging”
- Analyze it

What do you do with data?

- Find it (see previous slide)
- **Gather/save it**
- **Pre-process it**
  - **“cleaning”, “munging”**
- Analyze it

Your entire PhD

This course

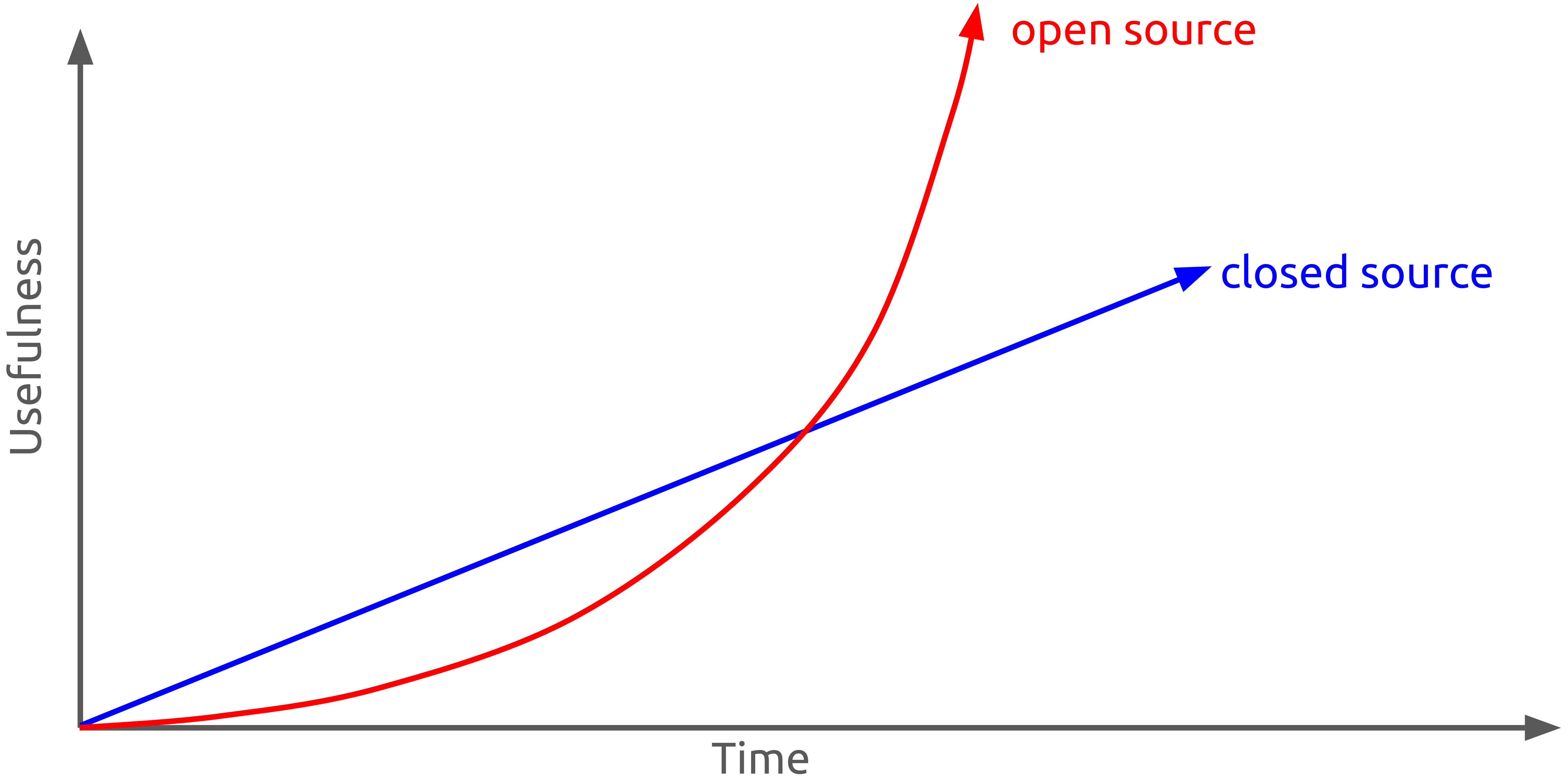
Your econ/stat classes

What tools do people use for this?

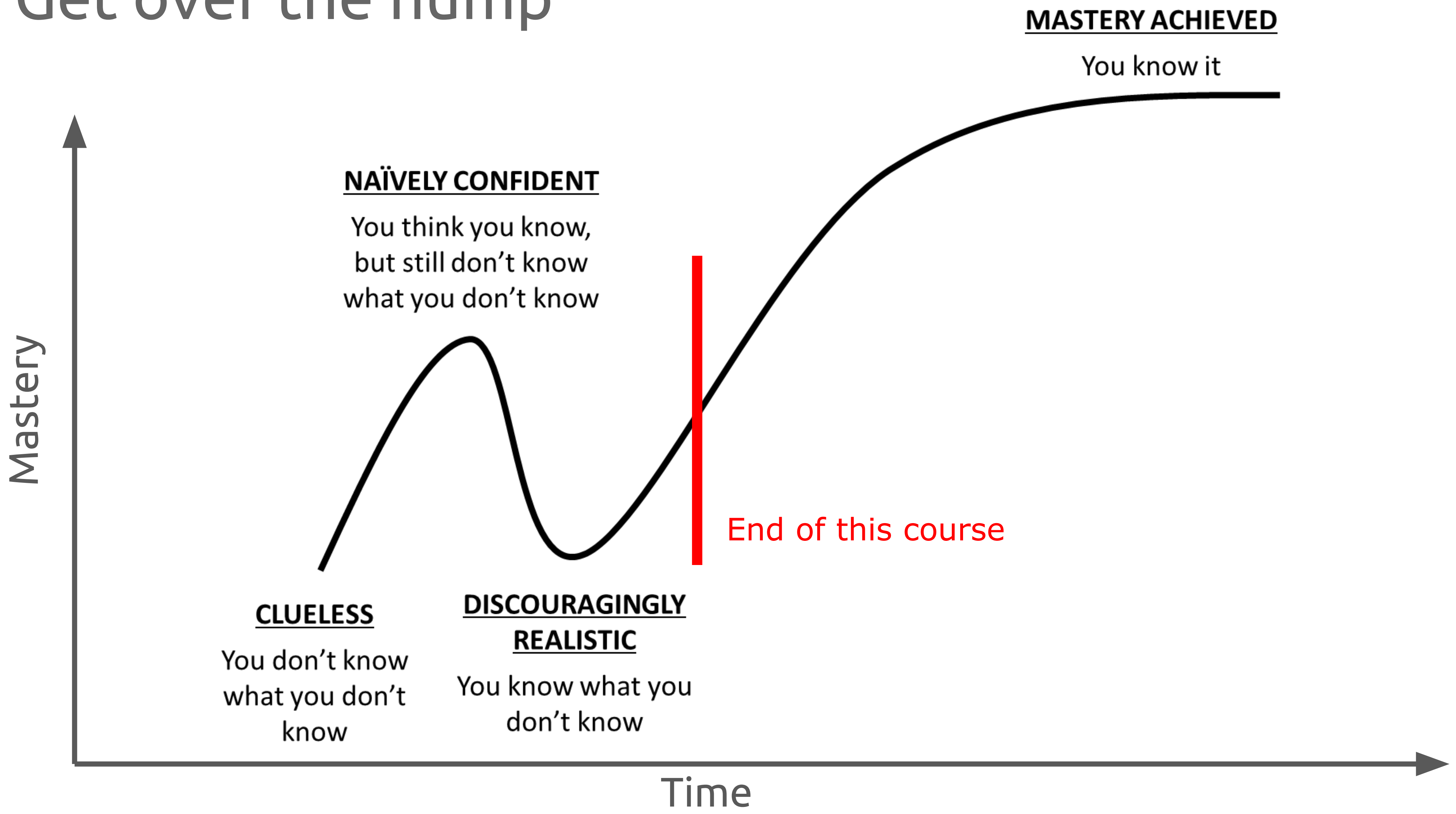
Why focus on Python and R?

- They are powerful, scripting languages
  - Designed to accomplish small tasks quickly
- These languages are open source
  - Leads to much more dynamic, state-of-the-art capabilities
- Most popular among younger and more technical academics
- However, there are tradeoffs
  - Learning curve
  - Documentation can be weaker for newer R/Python packages
  - Things are often in flux; there are many ways to do the same thing; not obvious which one to use

Over time, open source wins



Get over the hump


MASTERY ACHIEVED
You know it
NAÏVELY CONFIDENT
You think you know, but still don’t know what you don’t know
End of this course
CLUELESS
You don’t know what you don’t know
DISCOURAGINGLY REALISTIC
You know what you don’t know
Mastery
Time

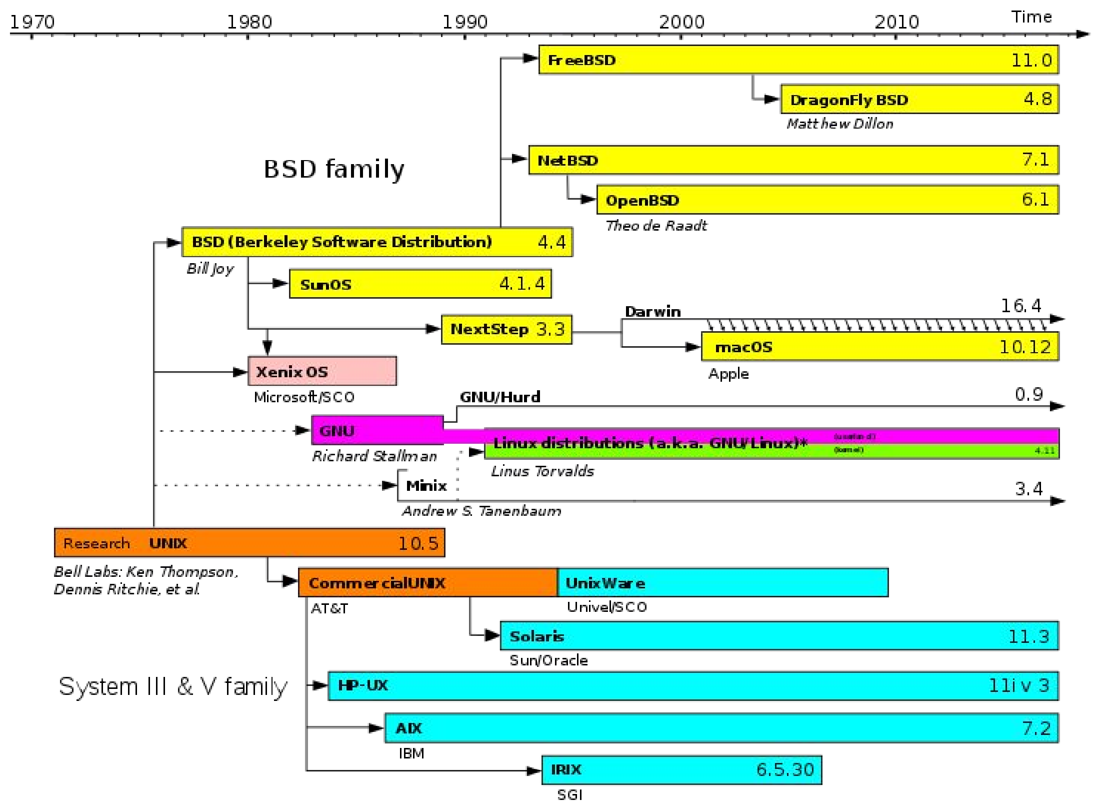
Command Line Basics

Is this really necessary?

- No, but it’s very useful
- Understanding how to use your command line will make you:
    - More technically literate
    - A better programmer
    - Opens up doors to many, many cool things you can do with computers
    - A hacker
- Many modern packages/projects/systems require some familiarity with command line interface (CLI)

Intro: Unix

- Important things for you to know:
    - Unix is an old operating system developed in the 1970s at Bell Labs
    - It quickly spread in Computer Science and academic communities, forking into and inspiring many operating systems



*nix (“Unix-like” environments)

- Important “Unix-like” systems:
  - MacOS
  - Linux. OSes based on Linux kernel:
    - Ubuntu
    - Debian
    - CentOS
    - Android
- Important non-Unix-like systems:
  - Windows

Why should you care?

- Most modern programmers use *nix systems
- Most tutorials, guides, and documentation is for *nix systems

- Windows users should:
  - still be somewhat familiar with the Windows "Command Prompt", but you can…
  - download a Unix-like system shell (i.e., git bash)
- Mac users should:
  - Feel smug

Just for fun

- Mac users:
  - Open “Terminal” and type:
    - “pwd” [enter] (prints working directory)
    - “ls” [enter] (lists files in directory)

- Windows users:
  - Open “Command Prompt” and type:
    - “cd” [enter] (prints current directory)
    - “dir” [enter] (lists files in directory)

Get a *nix-like shell on your machine

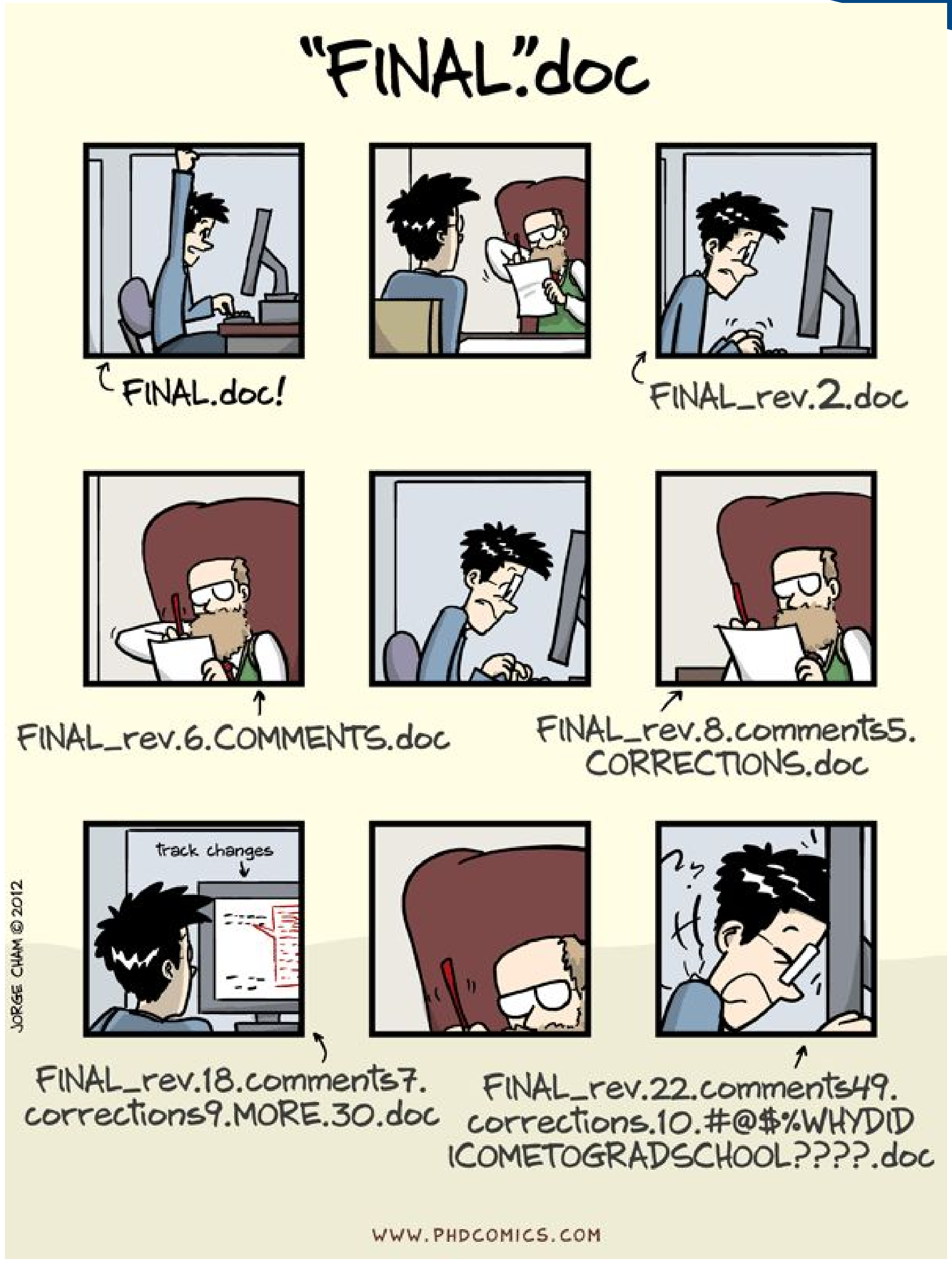
- Windows:
  - Download GitBash: https://git-for-windows.github.io/
- Windows 10:
  - You may be able to use “Bash on Ubuntu on Windows”, which is Microsoft’s attempt to join the future

- I can help you during the exercise portion of today’s lecture

Git, in brief

Git

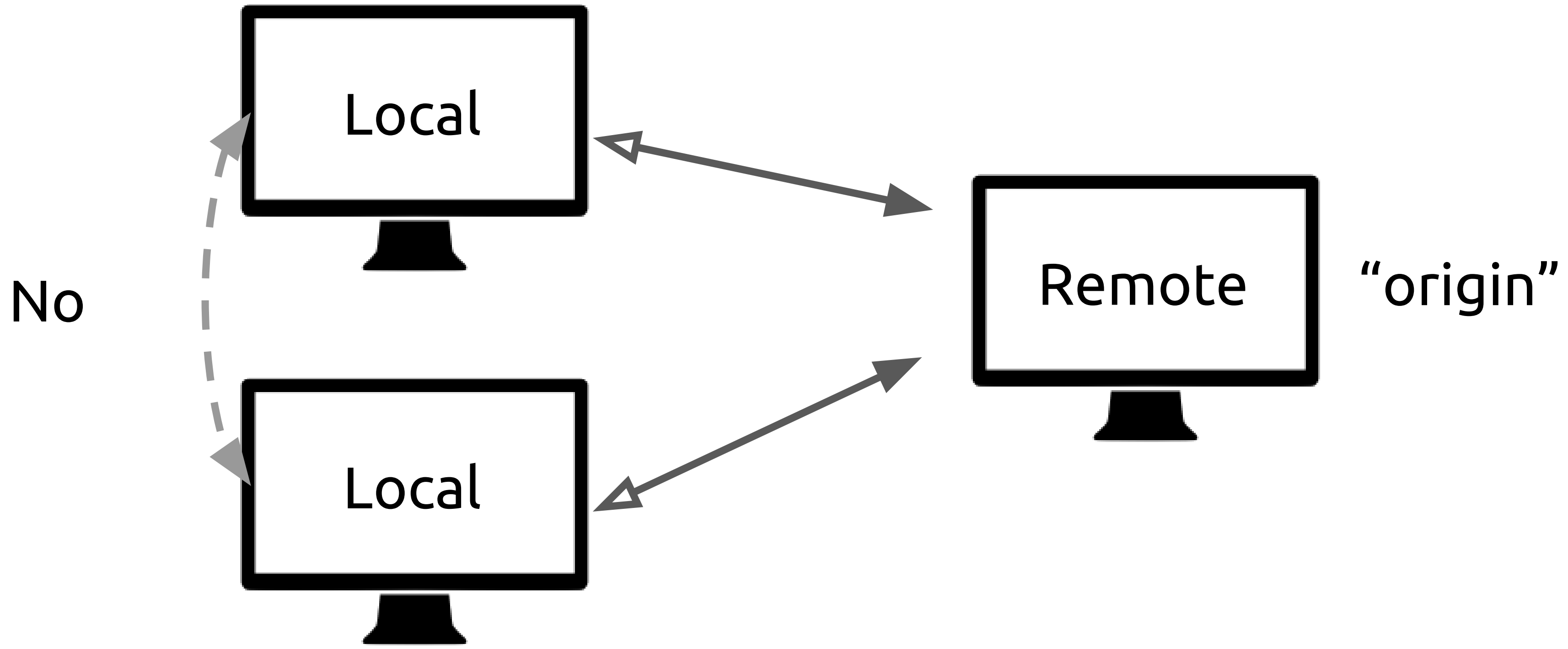
- Allows for efficient collaboration, tracking of changes, and edit history for *any* plaintext-based project
  - Very important tool for software engineering
  - Can be useful for academics, mostly for version history
- Generally good to be somewhat familiar with the basics


"FINAL".doc
FINAL.doc!
FINAL_rev.2.doc
FINAL_rev.6.COMMENTS.doc
FINAL_rev.8.comments5.CORRECTIONS.doc
track changes
FINAL_rev.18.comments7.corrections9.MORE.30.doc
FINAL_rev.22.comments49.corrections.10.#@$%WHYDID ICOMETOGRADSCHOOL????.doc
JORGE CHAM © 2012
WWW.PHDCOMICS.COM

Git basics

- Projects in git are called “repositories”
  - Consists of set of files/subfolders that are all part of the same code-base
- You typically have a “local” repo and a “remote” repo
  - But you don’t have to!
- Remote destinations are useful for backing up
- The most popular repository of repositories is GitHub
  - Essentially a storage service, with some neat features on top of git
  - Bitbucket is also free and has unlimited private repos

How to use git



- This system ensures there is a “master” version of your project
- Also makes it easy to share code, even with non-collaborators

Basics

```
$ cd myrepo/
$ git init   # initializes repository
$ git add --all   # adds files to “staging area”
$ git commit -m “my first commit” # saves changes
$ git log
```

- Will have chance to practice basics during exercise portion

Basics: Working with Remotes

```
$ git remote add origin [remote URL]
$ git remote -v # list remotes
$ git push origin master # push local to remote
```

https://help.github.com/articles/adding-an-existing-project-to-github-using-the-command-line/

- Will have chance to practice basics during exercise portion

Last notes

- Important!!!!!!!
    - Dealing with merge conflicts is VERY frustrating
    - Do your best not to edit the same file in two different places
    - If you do screw up, use StackOverflow:
        - https://stackoverflow.com/questions/161813/how-to-resolve-merge-conflicts-in-git
- RStudio has automatic version control with git built in
    - Will go over this later

R Basics

R

- R is an open-source programming language, based on “S”, a language developed by statisticians at Bell Labs
- RStudio is an IDE based around the R language
  - Importantly, RStudio != R
- Greatest strengths:
  - “Vector-native” calculations
    - Works very naturally with vector/matrix type data
  - R has LOTS of packages for many different stats applications
- Tradeoff/Weakness:
  - Does lots coercing/manipulation without telling you!

Do you need to know R?

- You should be familiar with it
- If other languages are common in your field (e.g., STATA, SAS), you will probably be fine without being an R expert
  - But! Know that R is becoming more popular
- Almost every applied course in the STAT department uses R
- Will only cover basics here so everyone has some experience

This is the boring part

- Unfortunately, R has a lot quirks which you can really only appreciate by seeing them worked through
- More so than other languages, R relies on lots of built-in functions with names you might not be familiar with
- If you are already familiar with R, go ahead and start the exercises (link at Tech Camp website)
  - http://opim.wharton.upenn.edu/techcamp/2017/
  - https://github.com/alexmill/techcamp_week1
  - Do the advanced exercise!

R Basics

- Basic Variable Types
  - Double, Integer, Character, Logical
- Compound Variable types
  - Vector, List, Factor, Matrix, DataFrame
  - Indexing/selecting
- Functions
- For Loops
- Braces/spacing: Reference
- Packages
  - How to discover? Experience + Google!
- Plots

Important things you won’t learn elsewhere

- R Markdown and R Notebooks
  - Use them!
  - (Please!) Never copy your code into a word document again!
  - Even better: Use the `stargazer` or `texreg` packages for reproducible tables!

If we have time today

- tidyverse
  - Important packages: dplyr, ggplot2, readr, stringr
  - Learn it! It will make your life easier
- data.table
  - Alternative to data.frames
  - Makes it easier to manipulate data; but not easier than dplyr
  - Slightly faster performance on big data than dplyr

Exercises

Follow instructions at:

https://github.com/alexmill/techcamp_week1