# Foundations Overview

```
$ echo "Data Sciences Institute"
$ echo "Rachael Lam"
```

# Why take this course?

- Unix shells - more specifically bash - is a powerful tool for quickly and easily navigating and manipulating files, scaling automated tasks, accessing Git and processing data.
- Git is extremely important for reproducibility of your personal work and collaborating with others on group projects.
- Git is incredible at keeping a historical reference of the changes you make to your work and debugging your code.
- Github has an amazing community with educational resources, open-sourced projects, and events.

# Learning Outcomes

1. Become comfortable with Unix basics and more complicated functions
2. Learn how to use Git and Github in solo and group projects
3. Navigate how to solve problems that you encounter
4. Understand why reproducibility is important and how to make your code reproducible

5. Grasp the ethical considerations of who is and isn’t in our datasets
6. Recognize past abuses of power and their continued influence
7. Learn professional skills and how to work within a team

## Prerequisites

Please come prepared with a Github account.

# Assessments

- A number of formative assessments that continuously put in practice what we learned in class
- Attitudinal assessments to help understand how students feel about the material and any areas for additional review
- One summative assessment that compiles everything we have learned
- Written reflections

# Course Reading Material

- Chacon and Straub, 2014, Pro Git, 2nd Edition.
- Newham and Rosenblatt, 2005, Learning the bash shell: Unix shell programming, O'Reilly.
- Timbers, Campbell, Lee, 2021, Data Science: A First Introduction, https://ubc-dsci.github.io/introduction-to-datascience/.
- William E. Shotts, Jr., 2009, The Linux Command Line
- Wilson, 2021, Building Software Together, https://buildtogether.tech/.

# Unix

# Intro to Unix and Linux

**Readings:** Newham et al. chapter 1 & Scotts chapter 1

Unix encompasses many features. In this section, we will look at what Unix/Unix shells are and the differences between Unix and Linux, introduce Bash and understand why it's important to learn. We will also get our environment set up so that we can try a few initial commands.

# Navigate Files and Directories

**Readings:** Newham et al. chapter 1.6 & Scotts chapters 3-4

To begin, we'll start with a bit of theory to understand directories and the differences between the types of paths and files. We'll then look at some tools that will help us navigate our files and directories using different options and arguments. We'll also learn a few commands that will help us quickly and easily navigate our system.

```
$ ls
```

```
$ cd
```

```
$ pwd
```

# Working with Files and Directories

**Readings:** Newham et al. chapter 1.7 & Scotts chapters 5, 7, 11

In this section we'll start manipulating files and directories. This includes creating, copying, moving and more. We'll then introduce inputs and outputs and how combine them into command pipes.

```
$ cp
```

```
$ mv
```

```
$ mkdir
```

```
$ rm
```

```
$ ln
```

# Pipes and Filters

**Readings:** Newham et al. chapters 1.9-2 & Scotts chapters 8-9 + 13

Continuing from last lesson, we'll expand on pipes and introduce some filter commands that will help us gain more shell experience. We'll also cover some important expansions and command line editing tips.

```
$ cat
```

```
$ sort
```

```
$ uniq
```

```
$ grep
```

```
$ find
```

# Shell Scripts

**Readings:** Scotts chapter 25

Now we'll learn how to group together commands and compile them into shell scripts. This avoids writing commands one by one on the command line. We'll build our first script and in the process discover how to write, run and store shell scripts.

# Shell Functions

**Readings:** Newham et al. chapter 4 & Scotts chapters 26, 33, 35

A good practice in programming is to create functions which separate larger tasks into smaller tasks. We'll learn the basic structure of functions, how to use self contained variables and parameters and further expand upon expansions.

```
function name {
        commands
        returns
}
```

```
name () {
        commands
        returns
}
```

# Flow Control

**Readings:** Newham et al. chapter 5 & Scotts chapters 28, 30

In the final lesson of Unix Shell, we'll introduce more advanced topics: if statements and loops. In doing so, we'll be able to write scripts that make decisions based on true/false statements and allow portions of our program to repeat.

```
x=5
if [ $x = 5 ]; then
       echo "x equals 5."
else
       echo "x does not equal 5."
fi
```

# Git and GitHub

# Intro to Git and Github

**Readings:** Wilson chapter 1 & Timbers chapter 12.3-12.4, 13.3.1

Git and Github are extremely important for code revisions, reproducibility and collaboration. This introduction will discuss local, centralized and distributed version control as well as how to get started with Git using some of our knowledge from the Unix lessons.

# Git Basics

**Readings:** Wilson chapter 2

At this point we'll create our first repository in an existing directory or by cloning a directory. We'll also introduce how to ignore files and why we might want to do that.

# Git Commands

**Readings:** Wilson chapter 2 & Timbers chapter 12.5-12.6

This lesson will contain the most important Git commands. We'll learn how to pull any changes, check the status of our work, commit changes, push them to our repository and even more commands and options. We'll also discuss why adding messages to our commits is important and should be added to our practice.

# Remote Repositories

**Readings:** Chacon and Straub chapter 2 & Timbers chapter 12.5-12.6

Here we will be focusing on remote repositories and the workflow you should adopt to appropriately collaborate with teammates. We'll learn commands such as `git remote`, `git pull` and `git push` during this section.

# Branching and Pull Requests

**Readings:** Wilson chapter 3

Git is extremely useful for separating work that you want to develop and test from the main line to avoid damage. It does this through a feature called branching. We'll be learning how to create these branches and merge them when our work is sufficient.

# Collaborating

**Readings:** Wilson chapter 3 & Timbers chapter 12.8

Another amazing feature of Git is the ability to collaborate with others. We'll discuss how to grant access to our repositories and how branching can help us collaborate. We'll also go through some of the best practices when collaborating with others.

# Dealing with Conflicts

**Readings:** Wilson chapter 6

Collaborating with others can produce several conflicts. We'll explore some practices to deal with merge conflicts when multiple individuals are working on a project, as well as Github Issues as another tool in collaboration. We'll end with some debugging using annotations and binary searches.

# Important Considerations

# Problem Solving

Problem solving is a necessary skill when writing code. In this section we'll learn how to identify the problem and effectively search for our solution using Google and Stack Overflow. We'll also begin a discussion how how reproducibility makes a difference when asking for help.

## Reproducibility

Reproducibility is extremely important in reducing and solving errors and increasing trust and transparency. We'll have thorough discussions on the significance of reproducibility and how to practice it through code commenting, documentation writing and proper folder structures.

# Ethics

When looking at open-source projects on Github or other libraries, it’s important to not take the information and results we see at face-value. We’ll be discussing what to look for, and who might be inappropriately excluded from our data. We’ll be examining at several datasets to analyze the ethics of the project and what might be missing.

## Inequality

When ethics are not taken into consideration, massive inequality can take place. We'll further our understanding of what happens when ethics are dismissed and the past abuses of power that have occured under this massively harmful failure.

## Professional Skills

We'll end this module with a lesson on pertinent and tech based professional skills. This includes healthy work habits, time management and best practices in meetings. We'll also discuss some team collaboration skills such as code reviews and sprint methodology.

# Discussion/Questions