Command Line Interface

**What is the CLI?**

Windows—Git Bash

Mac/Linux—Terminal

**What can the CLI do?**

* Navigate between folders
* Create files, folders, and programs
* Edit files, folders, and programs
* Run computer programs

**Basics of Directories**

* Directory = folder
* Directories are organized like a tree
* Directories can be inside other directories
* We can navigate the directories using CLI

**Special Directories: Root**

* Directory at the top of the tree is called the root directory
* The root directory contains all other directories
* The name of this directory is represented by a slash: /

**Special Directories: Home**

* Home directory is represented by a tilde: ~
* Usually contains most of your personal files
* Name of home directory is usually the name you use when you log into your computer

**Navigating Directories with the CLI:**

* Windows:
  + Open the start menu
  + Search for Git Bash
  + Open Git Bash

**CLI Basics**

* When you open your CLI you will see your prompt, which looks something like the name of your computer followed by your username, followed by a $
* When you open your CLI you start in your home directory
* Whatever directory you’re currently working with in your CLI is called the “working directory”
* You can imagine tracing all of the directories from your root directory to the directory you’re currently in
* This is called the “path” to your working directory
* In your CLI prompt, type **pwd** and press enter
* This will display the path to your working directory
* As you can see we get the prompt back after entering a command

**CLI Commands**

* CLI commands follow this recipe—command flags argument
* **Command** is the CLI command which does a specific task
* **Flags** are options we give to the **command** to trigger certain behaviors; preceded by a –
* **Arguments** can be what the **command** is going to modify, or other options for the **command**
* Depending on the **command**, there can be zero or more **flags** and **arguments**
* For example, pwd is a **command** that requires no flags or arguments

**CLI Commands Continued**

* **Pwd** displays the path to the current working directory
* **Clear** will clear out the commands in your current CLI window
* **Ls** lists files and folders in the current directory
* **Ls – a** lists hidden and unhidden files and folders
* **Ls – al** lists details for hidden and unhidden files and folders
* Notice that –a and –l are flags (they’re preceded by a -)
* They can be combined into one flag: -al
* **cd** stands for “change directory”
* **cd** takes as an argument the directory you want to visit
* **cd** with no argument takes you to your home directory
* **cd..** allows you to change directory to one level above your current directory
* **mkdir** is “make directory”
* just like “right click” to make a new folder
* **mkdir** takes as an argument the name of the directory you’re creating
* **touch**  creates an empty file
* **cp** stands for “copy”
* **cp** takes as its first argument a file, and as its second argument the path to where you want the file to be copied
* **cp**  can also be used for copying the contents of directories, but you must use the **–r** flag
* the line **cp – r Documents More\_docs** copies the contents of **Documents** into **More\_docs**
* **rm** stands for “remove”
* **rm** takes the name of a file you wish to remove as its argument
* you can also use **rm** to remove entire directories and their contents by using the **–r**  flag
* \*\*be careful doing this because there is no way to undo **rm**\*\*
* **Mv** stands for “move”
* With **mv** you can move files between directories
* You can also use **mv** to rename files
* **Echo** will print whatever arguments you provide
* **Date** will print today’s date

**Introduction to Git**

* Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later
* Git is a free and open source distributed version control system designed to handle everything from very small to very large projects with speed and efficiency
  + Created by the same people who developed Linux
  + The most popular implementation of version control today
  + Everything is stored in local repositories on your computer
  + Operated from the command line

**Introduction to GitHub**

* GitHub is a web-based hosting service for software development projects that use the Git revision control system
* Allows users to “push” and “pull” their local repositories to and from remote repositories on the web
* Provides users with a homepage that displays their public repositories
* Users’ repositories are backed up on the GitHub server in case something happens to the local copies
* Social aspect allows users to follow one another and share projects

**Creating a GitHub Repository**

* You don’t need GitHub to use Git
* Git = local (on your computer) GitHub = remote (on the web)
* GitHub allows you to:
  + Share your repositories with others
  + Access other users’ repositories
  + Store remote copies of your repositories on the GitHub server in case something happens to your local copies on your computer
* Two methods of creating a GitHub repository
  + 1. Start a repository from scratch
  + 2. “Fork” another users’ repository

**Basic Git Commands**

* **Adding**
  + Suppose you add new files to a local repository under version control
  + You need to let Git know that they need to be tracked
    - **git add .** adds all new files
    - **git add –u** updates tracking for files that changed names or were deleted
    - **git add –A** does both of the previous

**Installing R Packages**

* primary location for obtaining R packages is CRAN
  + there are about 5,200 packages available on CRAN
  + find then with the **available.packages ()** function
* for biological applications, many packages are available from the Bioconductor Project
* packages can be installed in R using the **install.packages()** function
* example: **install.packages(“slidify”)**
* installing from Bioconductor
  + source(<http://bioconductor.org/biocLite.R>)
  + biocLite(c(“GenomicFeatures”, “AnnotationDbi”))
* **library()** function needs to be used to load packages into R
  + example: **library(ggplot2)**

**Installing R Tools**

* Rtools is a collection of tools necessary for building R packages in Windows
* Available for download at <http://cran.r-project.org/bin/windows/Rtools/>