Bash Tutorial

What is Bash?

Bash = "**B**ourne **A**gain **SH**ell"

What/who is Bourne?

Stephen Bourne, inventor of Bourne Shell, `sh`.

What is a shell?

Shell is a layer around your operating system, interface with your operating system. It exposes the services (file management, process management, ...) of an operating system to a user.

There are two types of shells:

- Command Line Interface (CLI). You type commands in the *command line*. You do things by **typing**.
- Graphical User Interface (GUI). You have a *graphical* way for manipulating programs. You do things by **clicking**, **dragging**, **dropping**.

Somewhat confusingly, when we talk about shells though, we normally refer to command line interface only, and talk about it as opposed to GUI.

`bash` shell on your machine

- Windows: `git bash`, or `WSL`
- Linux: `Terminal`
- Mac: `Terminal`, or `iTerm`

Why would we want to do this?

Q: Why should we type, when we can click, drag, drop?

A: (Other than getting to feel like a hacker from a movie)

- Speed / automation
- More flexibility / more powerful than GUI / more control / more customizable
- Certain use cases require command line, e.g. git, cloud computing
- Often a prerequisite for a data scientist position
- Unlike with GUI, things can be done the same way on different operating systems
- If your trackpad dies :)

Basic `bash` commands

- *Navigation*: `cd`, `ls`, `pwd`
- *Moving*: `cp`, `mv`, `rm`
- *Inspecting*: `cat`, `less`

BONUS

Other cool things for you to check out / google / learn that may make your life easier.

- wildcards:
  - `*` matches zero or more characters
  - `?` matches a single character
  - `[]` matches any of the characters within the brackets, e.g. `ls l[aeiou]st.txt` will list `last.txt`, `lest.txt`, `list.txt`, `lost.txt`, `lust.txt`
  - `{,}` matches any of the terms inside the curly brackets separated by comma, e.g. `cp {*.pdf, *.ipynb} week_01/` will copy all `.pdf` *and* Jupyter notebook files into the `week_01` folder.
- print out a help page for a command: `man <command>`
- navigating to beginning/end of line: `CTRL-a`, `CTRL-e`
- counting lines, words, characters in a file: `wc -l`, `wc -w`, `wc -m`
- pipe, `|`: takes the output of the first command as an input to the second. A common usecase is wanting to find out the id of a process/program slowing your computer down: `ps aux | grep chrome`. The first part lists all processes, the second gets only those that have the word `chrome` in them and prints them out to the terminal. You can then look up the process id (PID) and kill it with `kill -9 PID`.
- `~/.bashrc` or `~/bash_profile` file: configuration file for `bash`; you can set up `bash` to your liking here and configure things like how `bash` looks, environment variables, aliases...
- aliases: useful ways to save yourself a lot of typing for commands you use often, e.g. instead of typing `cd Documents/spiced_projects` every time you want to access your SPICED folder, you can add a line:

  ```
  alias spiced='cd Documents/spiced_projects'
  ```

  in your `~/.bashr` or `~/bash_profile` file and you'll only have to type `spiced` to get to your SPICED folder.
- `reverse-i-search`: `CTRL-r`
- history expansion: `!$` gets the last parameter of the previous command, e.g. type `ls <directory-name>`, then `cd !$` and it will take you to the directory you `ls`-ed previously.
- `grep`, `cut`