Software Development Principles for Statistical Modelling
The Shell

Mick Cooney
mickcooney@gmail.com

`https://www.github.com/kaybenleroll/training_courses`

Code is available in the `sdpsm_intro` directory.

Content in this workshop is based on the 'Software Carpentry' course: `http://software-carpentry.org/`

1. Shell Basics

In this part of the workshop we introduce the Unix shell and get familiar with filesystem navigation and manipulation.

With these core concepts established, we move on to pipes, allowing the creation of filters and pipelines of work. This is part of the Unix philosophy of single-purpose utilities doing one job well.

Finally, we extend the notion of this concept to shell scripts: mini-programs that allow the codifying of workflows and other commands.

Exercise 1.1 Use the shell to determine your current username and home directory.

Exercise 1.2 Get a list of all the files in the current directory using the `ls` command.

Exercise 1.3 Use the internal help/manual system to see what options are available for the `ls` command.

Exercise 1.4 Use tab-completion to navigate the file system.

Exercise 1.5 Using a text editor, create a small text file named `draft.txt` and write some text into it. Save the file in the directory.

Exercise 1.6 Delete the file `draft.txt`.

Exercise 1.7 Calculate the number of lines and words in the `data/` folder.

Exercise 1.8 Store the output of the word counts to a file `lengths.txt`

Exercise 1.9 Sort the entries of `lengths.txt` in an appropriate way and save the output to `sorted_lengths.txt`

2. Shell Pipes and Filters

Exercise 2.1 Using a pipe (the `|` character), create a one-liner that outputs the file with the lowest number of lines in it.

Exercise 2.2 Using redirection (the `>` character), create a file called `lowest_lines.txt` from the one-liner shell command you wrote.

Exercise 2.3 Output the contents of the `us_states.txt` file to the terminal.

Exercise 2.4 From the length of the file, determine how many duplicates are in the file.

Exercise 2.5 Using the `uniq` script, try to remove duplicates. Why does the naive approach not work?

Exercise 2.6 Determine a way to create a new file, `us_states_uniq.txt` where each US state only appears once.

Exercise 2.7 Append the last alphabetical state to the `us_states_uniq.txt`.

Exercise 2.8 Check if `uniq` works with the file now.

3. Shell Scripts and Pipelines

Exercise 3.1 Using a `for` loop, find the top 3 lines of each of the files:

- `expectpayoff_100_1_0.2.mesh`
- `expectpayoff_200_1_0.4.mesh`
- `expectpayoff_400_2_0.2.mesh`

Exercise 3.2 Using a combination of `head` and tail, output lines 81–100.

Exercise 3.3 Using the `for` loop we rename the above files to have the prefix `sample_`

Exercise 3.4 Use a variable to allow the creation of prefix variable to allow the setting of the prefix at a single location.

Exercise 3.5 Use the `history` command to repeat the previous renaming. This should not alter the files you have renamed.

Exercise 3.6 Nest the loops to allow the renaming of all files with the `_200_` and `_2_` in the filename.

Exercise 3.7 Using the `echo` command, create a dry run script that renames all `200` and `0.2` files to a given prefix.

Exercise 3.8 Create a shell script called `middle.sh` that returns lines 81–100 from any input script.

Exercise 3.9 Rewrite the script so that it takes the starting line and the number of subsequent lines as input parameters.

4. Finding Files

Exercise 4.1 Find all states in the US that contain the word 'North' in them.

Exercise 4.2 Using the sorted file of US states, use the `-n` command-line argument to find the line numbers of the above states.

Exercise 4.3 Using extended regular expressions, find all US states that have more than one word in their name.

Exercise 4.4 Find all PDF files on your filesystem.