FIT1043 Introduction to Data Science

Week 9

Ts. Dr. Sicily Ting

School of Information Technology
Monash University Malaysia

Week 9 Outline

- Characterising data and "big data"
  - the V's
  - Metadata
  - Dimensions of data
  - Growth laws

- Introduction to Unix Shell for data science
  - Why Unix shell
  - Useful commands to read/manipulate large data files

Learning Outcomes

Week 9

By the end of this week you should be able to:

- *Characterize data sets used to assess a data science project*
- *Explain what Big data is*
- *Understand the V's in Big data*
- *Understand and analyse the growth laws: Moore's Law, Koomey's Law, Bell's Law and Zimmerman's Law*
- Analyze and use shell commands to read and manipulate big data

What is a UNIX Shell

```
• • •                    🏠 mcarman — bash — 80×24
Last login: Mon Aug 29 12:55:11 on ttys002
dyn-49-127-70-237:~ mcarman$ ▊
```

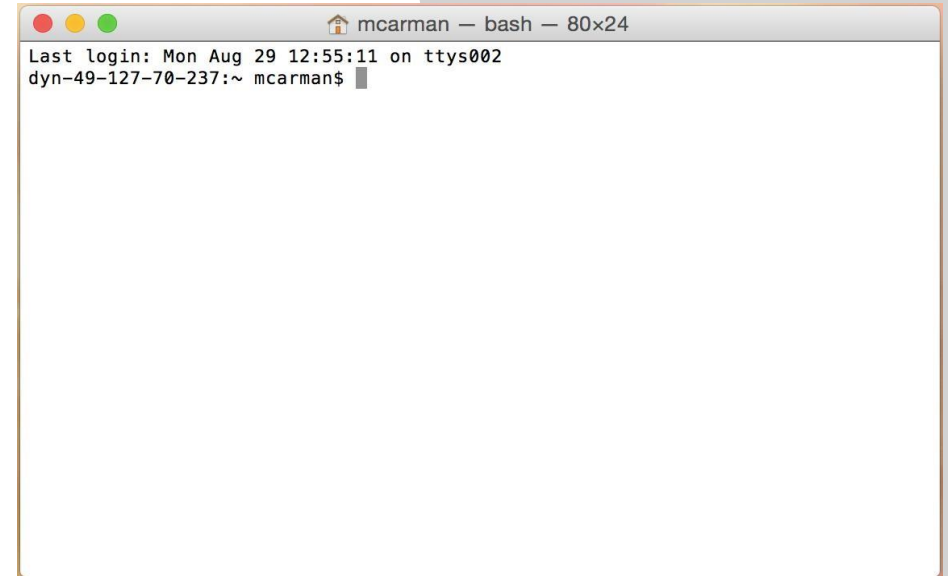Command line interface to a Unix computer
• Different shells have been around since the 70s

Why are shells interesting for Data Scientists?
• Provide **powerful** & **easy** way to **manipulate large** data files
• And **move** data around a network

Available on most Unix based operating systems
• Linux
• MacOS (BSD based)

Why Shell Scripting

Super-computers are typically UNIX/Linux based

- Explore data before you use it in Python or R

- Easier to manipulate (aka wrangle) Big Data

  - Simple and easy to learn.

  - Ideal for textual data, e.g. unstructured data for social networks, life sciences, system logs and so on.

  - Quick to sort, search, match, replace and clean your data.

Getting Started

Installing on Windows:

Windows Linux Subsystem

- For Windows 10, you can enable the Linux sub-system under (Control Panel → Programs → Turn Windows features on or off) after you have enabled the Developer mode on Windows.

[Alternative] Dual Booting your system

Using a Virtual Machine

Running a Shell:

- In Linux, click on the black square at the top left of the screen (depending on your interface).

- In MacOSX, go to Applications -> Utilities -> Terminal.

BASH
Basics I

Navigating the Filesystem

Change directory:
```
cd [destination]
cd /my/favourite/place
```

Special cases:
```
cd ..    <- Takes you up one directory
cd       <- Without argument, takes you to home directory
```

List files in the current directory:
```
ls
```

Copy files from one location to another:
```
cp [source] [destination]
cp /Downloads/myfile .
```

Reading a Text File

Open a text file for reading using less:

`less myfile.txt`

Navigate within the text file using

`[up/down]`	<- move one line the file
`[space]`	<- move down a whole page
`q`	<- quit
`[shift]+g`	<- skip to end of file
`/keyword`	<- search for the first occurrence of "keyword"
`/`	<- find the next occurrence of keyword

Some Useful Commands

Count the number of words/lines in a file

```
wc myfile.txt
```

Find lines in a file containing a keyword

```
grep "elephant" myfile.txt
```

Print the first/last few lines of a file

```
head myfile.txt
tail myfile.txt
```

Print the contents of a file to the screen

```
cat myfile.txt
```

MONASH University

Flags and Arguments

Many programs take flags and command line arguments that modify their behaviour, for example:

Sort the contents of a file lexicographically(alphabetically)

```
sort myfile.txt
```

Sort the lines of a file by numeric value

```
sort -n myfile.txt
```

Sort the data by column one, then column two and finally column three:

```
sort -k1,3 myfile.txt
```

BASH
Basics II

```
#!/bin/bash
echo "Hello world!"
```

Pipes

Sometimes we'd like the output of one program to be used as the input to another.

Doing this is super easy in the shell. We just use the pipe operator "|"

```
program1 | program2
```

We can chain as many programs together as we want, for example:

```
cat hourly_44201_2014-06.csv.gz | gunzip | less
```

MONASH University

Pipes (Notes)

The pipe is buffered

Each program in the list only generates data as it is needed by the next stage in the pipeline.

Thus **memory requirement** for processing the data is limited

Crucial for **scaling** up processing to **enormous data files**

Redirects

If we want to save the results in a file rather than pipe them to a new program:

Just change the pipe operator "|" to be a greater than symbol ">" and provide a filename:

```
cat hourly_44201_2014-06.csv.gz | gunzip > newFile.txt
```

Wildcards

Some unix commands can take multiple files as input, for example:

```
cat myfile1.txt myfile2.txt
```

In order to avoid listing large number of files, we can use the wildcard syntax to specify all files in a directory with a certain pattern, e.g.:

```
cat myfile*.txt
```

awk

In the tutorial, we'll have a look at a powerful command for processing text files one line at a time called `awk`

`awk` syntax:

```
awk '[select line?] {do something}'
```

Example

```
awk 'rand()<1/100 {print $6,$7,$14}'
```

Since `awk` processes data one line a time, it can **scale** up to **massive datasets**!

MONASH University

Scripts & Parallel Execution

You can also use scripts written in Python as programs in the shell (you are encouraged to explore this).

And find out how to run programs in parallel using the ampersand notation:

```
myprogram &
```

End of Introduction

We'll be experimenting with the Unix shell in this week's tutorial (or you may already have ☺   )

There are MANY excellent shell tutorials online if you'd like to learn more!

Re-cap: Learning Outcomes

Week 9

By the end of this week you should be able to:

- *Characterize data sets used to assess a data science project*
- *Explain what Big data is*
- *Understand the V's in Big data*
- *Understand and analyse the growth laws: Moore's Law, Koomey's Law, Bell's Law and Zimmerman's Law*
- Analyze and use shell commands to read and manipulate big data