Part Four

Shellscripting

Part 4: Outline

1. what are shellscripts
2. hello world
3. how to call a shellscript
4. getting arguments from terminal
5. for loops, if statements

What are shellscripts

Anything you type into your terminal, can be pasted into a file and executed

The code in the shellscript is read line-by-line by the bash interpreter, exactly the same as the lines you type into your terminal

Hello World in shellscript

```
#!/bin/bash
echo “hello world”
```

- Copy the above two lines into a file
- Make it executable (`chmod 755 hw.sh`)
- Call it (`./hw.sh`)

Hashbang (#!)

You need to tell the system what program should interpret your script

Syntax:

```
#! /path/to/executable
#! /bin/bash
```

Calling your script

Example:

```
$ cat scr.sh
  #!/bin/bash
  ls *
$ chmod 755 scr.sh  # make executable
$ ./scr.sh           # execute! (why ‘./’?)
```

Bash for-loops

Syntax:

```
for x in <list>; do
    <code>
done
```

for-loop example

```
# *.fa will expand to space separated list
for q in *.fa; do
    blastp -query $q -db mydb > $q.output
done
```

Command line arguments

```
$ cat scr.sh
#!/bin/bash
echo “$2 $1 $3”
$ ./scr.sh 12 56 89
56 12 89
```

If-else statements

```
if [[ <condition> ]]; then
    <code>
elif [[ <condition> ]]; then
    <code>
else
    <code>
fi
```

for-loop example (2)

```
# Find all pdfs that contain ‘Waldo’
for j in *.pdf; do
    lesspipe $j  |
        grep ‘Waldo' > /dev/null && echo $j
done
```

```
lesspipe  - extracts text data from almost anything
/dev/null - a place where output disappears
```

for-loop (3)

```
# find any .mp3 files that are not real
for j in $(find Home/ -iname “*.mp3”); do
  if [[ ! $(file $j) =~ 'Audio' ]]; then
    echo $j
  fi
done
```

Dying gracefully

```
# If myfile.txt doesn't exist, stop the script
if [[ ! -f myfile.txt ]]; then
    exit 1 # exit code 1 indicates error
fi
```

The spaces around the brackets matter!

Useful tests

```
-r file is readable
-f file exists*
-d directory exists
-s file exists and is not empty
-z test is a variable is empty
```

* **`-f`** tests for existence of a file, but it doesn’t recognize anonymous files, so it prevents command substitution. Generally use **`-r`** instead.

Example shellscript (1)

```
#!/bin/bash
# If the file is ASCII, then use normal less
if [[ $(file $1) =~ 'ASCII text' ]]; then
    less $1
# Otherwise run preprocessor
else
    lesspipe $1 | less
fi
```

Using the right tool

XML - use xmlstarlet

csv - awk if simple else csvtool or csvkit

HTML - some HTML parser

heavy math or statistics - R or matlab

grep, sed and awk are great for data prep