# Library Carpentry Week Two: Controlling Data (with the Shell)

## Schedule

17:45-18:25 Basics

18:30-19:25 Counting and Mining

19:30-20:25 Cleaning and Transforming

### Basics – navigating the shell

**pwd** print working directory

**ls** list directory

* -l: list file information
* -lh: list human reasable file information

**cd** change directory

**explorer .** open directory in graphical user interface (GUI)

### Basics – interacting with files

**mkdir** make directory

**cat** send file or files to output (in most cases, print to shell)

**head** output first parts of a file or files

**tail** output last parts of a file or files

**mv** rename or move a file or files

* syntax for renaming a file: mv FILENAME NEWFILENAME

**cp** copy a file or files

* syntax: cp FILENAME NEWFILENAME

**>** redirect output

* syntax with cat: cat FILENAME1 FILENAME2 > NEWFILENAME

**rm** remove a file or files. NB: *USE WITH CAUTION!!!*

### Basics – wildcards

**?** a placeholder for one character or number

**\*** a placeholder for zero or more characters or numbers

**[]** defines a class of characters

*Examples*

* foobar?: matches six character filenames starting with foobar
* foobar\*: matches all filenames starting with foobar
* foobar\*.txt: matches all filenames starting with foobar and ending with .txt
* [1-9]foobar?: matches seven character filenames starting that start with a number, have foobar after the number, and end with any character or number.

### Counting and Mining

**wc** word count

* -w: count words
* -l: count lines
* -c: count characters (or m for Mac users)

**grep** global regular expression print

* -c: displays counts of matches for each file
* -i: match with case insensitivity
* -w: match whole words
* -v: exclude match regular expressions
* --file=FILENAME.txt: use the file FILENAME.txt as the source of strings used in query

#### Exercise

With the person next to you, select a word to search for and use what you have learnt do to the following:

* Search for all case sensitive instances of that word in all four derived tsv files in this directory. Print you results to the shell.
* Search for all case sensitive instances of that word in the 'America' and 'Africa' tsv files in this directory. Print you results to the shell.
* Count all case sensitive instances of that word in the 'America' and 'Africa' tsv files in this directory. Print you results to the shell.
* Count all case insensitive instances of that word in the 'America' and 'Africa' tsv files in this directory. Print you results to the shell.
* Search for all case insensitive instances of that word in the 'America' and 'Africa' tsv files in this directory. Print you results to a new .tsv file.
* Search for all case insensitive instances of that whole word in the 'America' and 'Africa' tsv files in this directory. Print you results to a new .tsv file.
* Compare the line counts of the last two files.
* Open both files in a text editor (Notepad++, Atom, Kate, whatever you prefer) or Excel-like program to see the difference between searching strings and searching whole words using grep

### Cleaning and Transforming

#### Exercise

*Work on this exercise with the person next to you*

**Grabbing a text, cleaning it up**

Head to c:\users\DSTraining\desktop\119\text\. You're going to work again with the gulliver.txt file we saw earlier.

The sed command allows you to edit files directly.

Type sed '9352,9714d' gulliver.txt > gulliver-nofoot.txt and hit enter.

The command sed in combination with the d value will look at gulliver.txt and delete all values between the rows specified. The > action then prompts the script to this edited text to the new file specified.

Now type sed '1,37d' gulliver-nofoot.txt > gulliver-noheadfoot.txt and hit enter. This does the same as before, but for the header.

You now have a cleaner text. The next step is to prepare it even further for rigorous analysis.

The tr command is used for translating or deleting characters. Type tr -d [:punct:] < gulliver-noheadfoot.txt > gulliver-noheadfootpunct.txt and hit enter.

This uses the translate command and a special syntax to remove all punctuation. It also requires the use of both the output redirect > we have seen and the input redirect < we haven't seen.

Finally regularise the text by removing all the uppercase lettering. Type tr [:upper:] [:lower:] < gulliver-noheadfootpunct.txt > gulliver-clean.txt and hit enter.

Open the gulliver-clean.txt in Notepad++. Note how the text has been transformed ready for analysis.

**Pulling a text apart, counting word frequencies**

You are now ready to pull the text apart.

Type tr ' ' '\n' < gulliver-clean.txt > gulliver-linebyline.txt and hit enter. **Note: there is a space between the first two single quotation marks**

This uses the translate command again, this time to translate every blank space into \n which renders as a new line. Every word in the file will now have its own line.

This isn't much use, so to get a better sense of the data we need to use another new command called sort. Type sort gulliver-linebyline.txt > gulliver-ordered.txt and hit enter.

This script uses the sort command to rearrange the text from its original order into an alphabetical configuration. Open the file in Notepad++ and after scrolling past some blank space you will begin to see some numbers and finally words, or at least lots of copies of 'a'!

This is looking more useful, but you can go one step further. Type uniq -c gulliver-ordered.txt > gulliver-final.txt and hit enter.

This script uses uniq, another new command, in combination with the -c flag to both remove duplicate lines and produce a word count of those duplicates.

You have now taken the text apart and produced a count for each word in it. Congratulations!

Before you finish, take a look at the data: do you notice anything odd? And if so, what might be at fault and how might you fix it?

### References and further reading

* Command Line Crash Course <http://cli.learncodethehardway.org/book/>
* Computer Science 101, Coursera https://www.coursera.org/course/cs101
* Programming for Everybody (Python), Coursera <https://www.coursera.org/course/pythonlearn>
* *The Programming Historian* <http://programminghistorian.org/>
  + James Baker, 'Preserving Your Research Data'
  + Ian Milligan and James Baker, 'Introduction to the Bash Command Line'
  + James Baker and Ian Milligan, 'Counting and mining research data with Unix'
* Deborah S. Ray and Eric J. Ray, *Unix and Linux: visual quickstart guide* (4th edition, 2009)
* Software Carpentry <http://software-carpentry.org/>
* Al Sweigart, *Automate the Boring Stuff* (2015) <https://automatetheboringstuff.com/>
* William Turkel, 'Basic Text Analysis with Command Line Tools in Linux' (15 June 2013) <http://williamjturkel.net/2013/06/15/basic-text-analysis-with-command-line-tools-in-linux/>
* William Turkel, 'Pattern Matching and Permuted Term Indexing with Command Line Tools in Linux' (20 June 2013) <http://williamjturkel.net/2013/06/20/pattern-matching-and-permuted-term-indexing-with-command-line-tools-in-linux/>
* William Turkel, 'Named Entity Recognition with Command Line Tools in Linux' (30 June 2013) <http://williamjturkel.net/2013/06/30/named-entity-recognition-with-command-line-tools-in-linux/>