ceu-economics-and-business.github.io

Command Line Exercises. – ECBS 5146 SQL and Different Shapes of Data

6–7 minutes

Overview

Teaching: 90 min

Questions

- How can an analyst use bash to explore/modify data stored in a flat file?
- How can an analyst build simple bash tool to perform basic (but fast) analytics?
- Can be Linux bash used on Windows/MacOS?

Objectives

- Basic understanding of Linux bash
- Practice of viewing/editing/modifying a large csv file
- Quick intro to Linux scripts

Keywords

#BASH

#GIT BASH

#SENTIMENT ANALYSIS

Prerequisites for this chapter

In this chapter we will use Git bash.

Instructions for Windows:

- Install Git for Windows: https://gitforwindows.org/
- This installs Git bash too. More info on Git bash: https://www.atlassian.com/git/tutorials/git-bash
- Please run and check (and double check) that Git Bash works properly. Type `pwd` and see if you get a response.

Instructions for Mac:

- You should have Git installed by default
- Open a Terminal application and type: `git --version` - if you get back something like `git version 2.24.3 (Apple Git-128)` you should be ok.
- If you don't have git or you have an older version, type `brew install git`

Preparing the exercise artifacts

First, let's check some file navigation. Your current location:

List files in your current folder in two different formats:

To get the required artifacts (flat files for exercises) navigate to a folder suitable for exercises (like Documents, CEU class folder etc) and clone the course repo:

Example:

```
cd Documents/
mkdir bash
git clone https://github.com/CEU-Economics-and-Business/ECBS-5146-Different-Shapes-of-Data
cd ECBS-5146-Different-Shapes-of-Data/artifacts/bash/
```

Basic commands

`cat` - Print the file to the screen.

head - Print the first 3 lines to the file to the screen

```
head -n 3 birdstrikes.csv
```

What is `-n`? Check it in the manual

`tail` - Check the last 5 lines of the file

```
tail -n 5 birdstrikes.csv
```

> - Save the first 10 lines into another file (note that instead of -n 10, we used a simplified version)

```
head -10 birdstrikes.csv > first10.csv
```

list the result

| - concatenating command. Show the 10th line only:

```
head -n 10 birdstrikes.csv | tail -n 1
```

Exercise 1

Copy the 6th and 7th line of birdstrikes into the myprettypony.csv

Filtering

`grep` - Only show incidents from California

```
cat birdstrikes.csv | grep California
```

`grep -v` - Only show incidents NOT with California

```
cat birdstrikes.csv | grep -v California
```

`grep -i` - Ignore case

```
cat birdstrikes.csv | grep -i CALIFORNIA
```

Word count

`wc` - show the line, word and character count of birdstrikes

`wc -l` shows only the line count

> ### Exercise 2
>
> How many words we have in the first 10 lines of Hamlet.txt?

> ### Exercise 3
>
> How many incidents were in California (only output line count)

Cut / printing out parts of a line

What is name of the 1st column:

```
cat birdstrikes.csv | cut -d ';' -f1 | head -1
```

- `cut` - Display the first 10 records of *damage* (column 4) and the *cost* (column 10) columns

```
cat birdstrikes.csv | cut -d ';' -f4,10 | head -11
```

Similar to previous example, but now writing all columns, except *damage* (column 4) and *cost* (column 10) into a new file:

```
cat birdstrikes.csv | cut -d ';' --complement -f4,10
> new.csv
cat birdstrikes.csv | cut -d ';' -f1-3,5-9,11- >
new.csv

head -5 new.csv
```

Exercise 4

Write *state* and the *bird size* columns of the 17th record from birdstrikes, in file called *onerepublic.csv*. What is the result if run 'cat onerepublic.csv'?

Sorting

```
sort birdstrikes.csv | head -10
sort -n birdstrikes.csv | head -10
```

Reverse sort by *damage* (column 4)

```
cat birdstrikes.csv | sort -t ';' -k4 -r | head -10
```

Exercise 5

What was the cost of the most expensive incident?

Distinct/Unique values

`sort | uniq` - Distinct states in birdstrikes:

```
cat birdstrikes.csv | cut -d ';' -f6 | sort | uniq |
wc -l
```

`uniq -c` - How many incidents were there by state?

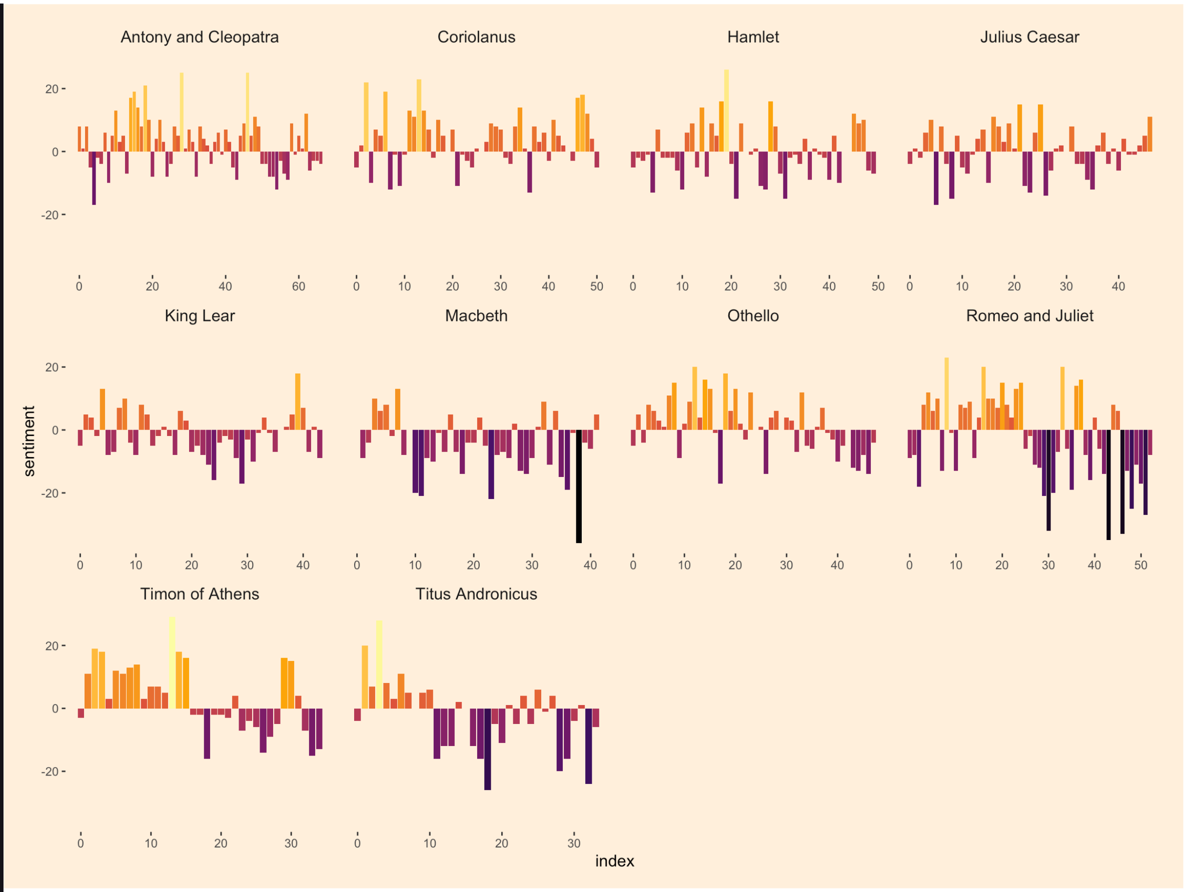
```
tail -n+2 birdstrikes.csv | cut -d ';' -f6 | sort |
uniq -c
```

Scripts

Sentiment analysis

In the next example, we would like to present a small script for sentiment analysis of books. You will see, that using the commands above and adding a bit of procedural logic by bash scripting, we can easily create a basic sentiment analyzer.

This script is focusing on counting certain negative words and calculating the ratio against the number of words in a book. The script could be improved (as the image below suggests) to look for context, distribution in time and ultimately check the positive words as well, for a bit more balanced analytics.



Source of image: [Fair is foul, and foul is fair: a tidytext sentiment analysis of Shakespeare's tragedies]

First, we create the file for the script:

Then, here is the script itself:

```
dictionary=(sad sorrow death dead pain poor misery)
totalcount=0

for i in "${dictionary[@]}";
do
        ((wordcount=$(grep -i -c $i $1)))
        ((totalcount+=$wordcount))
        echo $i:$wordcount
done
```

```
words=($(wc -w $1))
ratio=$(($words/$totalcount))

echo ---
echo total:$totalcount
echo words:$words
echo ratio:$ratio

echo ---
if [ "$ratio" -gt 1000 ]; then
        echo Sentiment: this book is not sad
else
        echo Sentiment: this book is sad
fi
```

On the end, we run the script with a book stored in txt format:

```
sh sentiment.sh Hamlet.txt
```

Book recommendation

If you want to get deeper into the magic land of command lines, a new book is available for free: Data Science at the Command Line

Homework (Optional, no need to submit)

- Show the first 3 Helicopter incidents outside of Colorado
- How many incidents did happen were cost is bigger than 0
- In which Area did the most expensive incident happen that was caused by a Small bird?