Quiz 2: Working with Large Files

Casey Owen

CS119 Big Data

Fall 2024

Question 1a

What does the man pwd command tell you about the pwd command? You should learn what the man command is if you tried with different options, such as man ls, man cd.

The man command shows the manual for a given command, in this case the pwd command. It gives a synopsis of how to use the command, as well as a description of all option flags and what they do

Question 1b

Explain piping in Linux.

The pipe | character in Linux provides a way to direct input from one command to the next – the output of the command on the left is used as the input to the command on the right. It is useful for chaining commands together without the need to create temporary files.

Question 1b

Explain the concepts of stdin, stdout and stderr.

These are three data streams within Linux – stdin is standard input, stdout is standard output, and stderr is standard error. Commands typically take input from stdin and direct output to stdout (the terminal), sending error messages to stderr. You can redirect stdout to a file with the > character.

Question 2

Find the column names in the Opioid dataset. The naive way is to gunzip the .gz file and run head -1 on the result, but you likely don't have enough disk space. Conveniently, zcat can read the file and write the unzipped contents into stdout, which can be piped into head -1.

Command: zcat arcos_all_washpost.tsv.gz | head -n 1

This command runs zcat – a form of gunzip that directs the output directly to stdout on the file, and pipes the output to stdin of head. Head then uses the -n option to specify that only the first line should be displayed. This results in the first line, which is the column names, being printed to the terminal window.

Question 3

Find the number of rows in the Opioid dataset by processing the zcat output, stripping the header row, and counting the remaining lines using wc.

Command: zcat arcos_all_washpost.tsv.gz | tail -n +2 | wc --lines

Again running zcat, the output is this time piped into tail, which is the opposite of head – counts all remaining lines. Using the option flag -n +2, the command inverts the number of lines selctor, to go from number of lines to exclude. +2 selects all but the top 1 line, since +1 would select the whole file. This result is then piped into wc with the lines option, which returns the number of lines in the input. The result is 178598026.

Question 4

Find the names of all the drugs named in the dataset.

First, to find out what column number drug name is, I consulted the output of question 2.

To confirm the column name was correct before running the large computation, I used the command: zcat arcos_all_washpost.tsv.gz | head -n 1 | awk -F '\t' '{print $24}'

Which gave the result DRUG_NAME, column I am interested in, and confirms that the 24th column is correct. awk is used here to select the 24th column in the input, and the -F flag shows that the command should expect the input to be tab-seperated.

To find the answer to the question, I used:

Command: zcat arcos_all_washpost.tsv.gz | awk -F '\t' '{print $24}' | tail -n +2 | sort | uniq

Similar to question 3, and the above, this starts by selecting the DRUG_NAME column, then stripping the header. To then find the uniq values, the uniq command is used with the default options. Since this command expects the input to be sorted, it is preceded by sort, which acts as the name implies and sorts the input.

Result:

HYDROCODONE

OXYCODONE

Question 5

Estimate the number of rows for each year in the dataset. There may be enough space in the shell, but we'd prefer not to rely on that. So here's a potential strategy: Use the shuf command to extract, say, random 7,500 rows from the output of zcat. Find the proportion of rows for each year in this extract. Assuming that the distribution of the random 7,500 rows is similar to the distribution in the whole file, estimate the number of rows for each year.

Command: zcat arcos_all_washpost.tsv.gz | tail -n +2 | shuf -n 7500 | awk -F '\t' '{print $31}' | cut -c 5-8 | sort | uniq -c | awk '{$1*=(178598026/7500)}1'

The process for this one is:

Remove headers → sample 7500 points (with shuf and -n argument) → select column 31 (TRANSACTION_DATE) → select characters 5-8 of that column, which represent the year as a string → sort the years → get unique years, with -c flag to also return counts in a new column → multiply the counts column by the total number of rows over the number of samples, 178598026/7500, which gives the expected number of rows of each year in the full file

Result:

```
2.11698e+07 2006

2.33606e+07 2007

2.41941e+07 2008

2.55038e+07 2009

2.75993e+07 2010

2.78137e+07 2011

2.89567e+07 2012
```

Question 6

Extract any 4,000 rows for the year 2012 from arcos_all_washpost.tsv.gz. Write the resulting lines to arcos_2012.tsv.

Based on the piazza post about this question, my understanding is that the header must be printed as well, and the file should be 4,000 lines long exactly – so it should contain 3,999 records.

Command:
```
zcat arcos_all_washpost.tsv.gz | head -n 1 > arcos_2012.tsv; zcat arcos_all_washpost.tsv.gz | tail -n +2 | awk -F '\t' '$31 ~ /2012$/ {print $0}' | shuf -n 3999 >> arcos_2012.tsv
```

This contains two commands, the first simply gets the header and prints it to the file.

The second command appends the records to the created file – it does this by first stripping headers, then using an awk command to perform the matching. It uses regex to print all columns ($0) where the TRANSACTION_DATE column ($31) ends with ($) 2012. Then it shuffles the result and picks 3,999 rows, redirecting stdout in append mode to the new file.

Testing:

I ran several simple commands to verify that this file has the properties I expect.

Command:
```
cat arcos_2012.tsv | wc -l
```

returned 4000, as expected (the file is 4,000 lines)

Command:
```
cat arcos_2012.tsv | awk -F '\t' '{print $31}' | tail -n +2 |cut -c 5-8 | sort | uniq
```

returned 2012, as expected (the list of all unique years)

Command: cat arcos_2012.tsv | tail -n +2 | sort | uniq -d

returns nothing, as expected (the duplicate lines)

Command: diff <(cat arcos_2012.tsv | head -n 1) <(zcat arcos_all_washpost.tsv.gz | head -n 1)

returns nothing, as expected (the difference in headers between this file and the orignial)