

Handling Compressed Data

For this part of the quiz, you may use your own laptop. On a Mac or a Chromebook, use Terminal. On Windows laptops, we recommend using the ubuntu¹. If you don't have enough space on your laptop, you may use the eecs Linux servers². The eecs Linux servers already have the compressed file available as `/comp/119/arcos_all_washpost.tsv.gz`.

I ran my code on Mac os Bash.

Programs I used (including installed ones):

- Gzcat
- Pv (installed to see progress)
- Pigz (installed for parallelized process)
- Gshuf (installed coreutils to shuffle)

¹ If you don't have ubuntu on your Windows laptop, you may have to enable it; the instructions are [here](#).

² Visit <http://systems.eecs.tufts.edu/managing-your-password/> to obtain a login and password, then ssh `linux.eecs.tufts.edu` to access your linux account.

1. [15 points] 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.

- Code : `gzcat arcos_all_washpost.tsv.gz | head -n 1`
 - head -n 1 gives you the first row, which is useful to quickly check the header structure without processing the entire large file.
- Result :

REPORTER_DEA_NO	REPORTER_BUS_ACT	REPORTER_NAME	REPORTER_ADDL_CO_INFO		
REPORTER_ADDRESS1	REPORTER_ADDRESS2	REPORTER_CITY	REPORTER_STATE	REPORTER_ZIP	
REPORTER_COUNTY	BUYER_DEA_NO	BUYER_BUS_ACT	BUYER_NAME	BUYER_ADDL_CO_INFO	
BUYER_ADDRESS1	BUYER_ADDRESS2	BUYER_CITY	BUYER_STATE	BUYER_ZIP	BUYER_COUNTY
TRANSACTION_CODE	DRUG_CODE	NDC_NO	DRUG_NAME	QUANTITYUNIT	ACTION_INDICATOR
ORDER_FORM_NO	CORRECTION_NO	STRENGTH	TRANSACTION_DATE	CALC_BASE_WT_IN_GM	
DOSAGE_UNIT	TRANSACTION_ID	Product_Name	Ingredient_Name	Measure	MME_Conversion_Factor
Combined_Labeler_Name	Revised_Company_Name	Reporter_family	dos_str		

- Output :

```
yi.hs@yihui-MacBookPro week2 % gzcat arcos_all_washpost.tsv.gz | head -n 1
REPORTER_DEA_NO REPORTER_BUS_ACT REPORTER_NAME REPORTER_ADDL_CO_INFO REPORTER_ADDRESS1 REPORTER_ADDRESS2 REPORTER_CITY REPORTER_STATE REPORTER_ZIP REPORTER_COUNTY BUYER_DEA_NO BUYER_
BUS_ACT BUYER_NAME BUYER_ADDL_CO_INFO BUYER_ADDRESS1 BUYER_ADDRESS2 BUYER_CITY BUYER_STATE BUYER_ZIP BUYER_COUNTY TRANSACTION_CODE DRUG_CODE NDC_NO DRUG_NAME QUANTI
TY UNIT ACTION_INDICATOR ORDER_FORM_NO CORRECTION_NO STRENGTH TRANSACTION_DATE CALC_BASE_WT_IN_GM DOSAGE_UNIT TRANSACTION_ID Product_Name Ingredient_Name Measure MME_Co
nversion_Factor Combined_Labeler_Name Revised_Company_Name Reporter_family dos_str
```

2. [15 points] 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.

- Code : `pv arcos_all_washpost.tsv.gz | pigz -dc | cut -f1 | tail -n +2 | wc -l`
 - `tail -n +2`: Would skip the first line (header) and start from the second line
 - `wc -l`: Would count the total number of lines

***** for this question I made some changes for efficiency.**

1. Used `pigz` instead of `gzcat` which parallelizes the decompression process by utilizing multiple CPU cores
 2. Added `cut -f1` which significantly improved performance, likely because:
 - Processing a single column reduces the memory overhead
 - Less data needs to be processed through the pipeline
 3. The combined improvements showed substantial performance gains:
 - Before changes: 2.67 MB/s (ETA 40 min)
 - After changes: 13.4 MB/s (ETA 10 min)
- Result : 178598026
 - Output:

```
yi.hs@yihui-MacBookPro week2 % pv arcos_all_washpost.tsv.gz | gzcat | tail -n +2 | wc -l
^C.0MiB 0:00:04 [2.67MiB/s] [>
```

```
yi.hs@yihui-MacBookPro week2 % pv arcos_all_washpost.tsv.gz | pigz -dc | cut -f1 | tail -n +2 | wc -l
6.41GiB 0:08:08 [13.4MiB/s] [=====]
178598026
```

3. [20 points] Find the names of all the drugs named in the dataset³.

- Code : `pv arcos_all_washpost.tsv.gz | pigz -dc | cut -f24 | tail -n +2 | sort -u`
 - `cut -f24`: Extracts column 24 which contains the drug names (verified by checking the header)
 - `tail -n +2`: Skips the header row to get just the data
 - `sort -u`: Gets unique drug names by removing duplicates
- Result : HYDROCODONE, OXYCODONE
- Output:

```
yi.hs@yihui-MacBookPro week2 % pv arcos_all_washpost.tsv.gz | pigz -dc | cut -f24 | tail -n +2 | sort -u
6.41GiB 0:11:39 [9.39MiB/s] [=====]
HYDROCODONE
OXYCODONE
```

³ There are multiple ways of solving this problem, but many of them result in “write failed: No space left on device.” The task is to find one that works!

Analysis of the Opioid Dataset Using Sampling

4. [20 points] Estimate the number of rows for each year in the dataset⁴. There may be enough space in the shell, but this exercise requires you to assume that that's not the case. 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
- Code : `pv arcos_all_washpost.tsv.gz | pigz -dc | cut -f31 | awk '{print substr($1,5,4)}' | gshuf -n7500 | sort -n | uniq -c | awk '{print $2, $1}'`
 - `awk '{print substr($1,5,4)}'`: Extracts the year (last 4 digits) from the date format
 - `gshuf -n7500`: Randomly selects 7,500 rows from the dataset
 - `sort -n`: Sorts the years numerically
 - `uniq -c`: Counts how many times each year appears
 - `awk '{print $2, $1}'`: Rearranges output to show "year count" instead of "count year"
 - Result : this is the result from 7500 samples.
2006 875
2007 1022
2008 1025
2009 1039
2010 1110
2011 1188
2012 1241

From this, I can estimate the whole file.

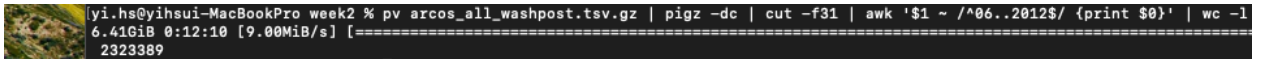
```
2006 20,842,390
2007 24,342,911
2008 24,414,351
2009 24,735,827
2010 26,432,508
2011 28,289,927
2012 29,557,973
```

⁴ There are no month or year fields in the dataset. That information is embedded in the TRANSACTION DATE.

- Output :

```
yi.hs@yihisui-MacBookPro week2 % pv arcos_all_washpost.tsv.gz | pigz -dc | cut -f31 | awk '{print substr($1,5,4)}' | gshuf -n7500 | sort -n | uniq -c | awk '{print $2, $1}'
6.41GiB 0:12:08 [9.01MiB/s] [=====]
2006 875
2007 1022
2008 1025
2009 1039
2010 1110
2011 1188
2012 1241
```

5. [15 points] Obtain the count of rows for June 2012 by extracting all such rows from `arcos_all_washpost.tsv.gz` and running `wc` on the extracted rows.
- Code : `pv arcos_all_washpost.tsv.gz | pigz -dc | cut -f31 | awk '$1 ~ /^06..2012$/ {print $0}' | wc -l`
 - `cut -f31`: Extracts column 31 which contains the transaction dates
 - `awk '$1 ~ /^06..2012$/ {print $0}'`: Matches dates from June 2012 (06..2012 where dots match any day)
 - `wc -l`: Counts the total number of matching rows
 - Result : 2323389
 - Output :

A terminal window screenshot showing a command being executed. The command is `pv arcos_all_washpost.tsv.gz | pigz -dc | cut -f31 | awk '$1 ~ /^06..2012$/ {print $0}' | wc -l`. The output shows the file size and transfer speed: `6.41GiB 0:12:10 [9.00MiB/s]`, followed by a separator line of equals signs, and then the final count: `2323389`.

```
yi.hs@yihui-MacBookPro week2 % pv arcos_all_washpost.tsv.gz | pigz -dc | cut -f31 | awk '$1 ~ /^06..2012$/ {print $0}' | wc -l
6.41GiB 0:12:10 [9.00MiB/s] [=====]
2323389
```

6. [15 points] Estimate the count of rows for June 2012 based on answers to questions 1, 2, and 4 and compare that count with your findings from Q5.

From previous results we have:

- Total rows for 2012: 29,557,973
- Sample proportion analysis showed 2012 had 1241/7500 entries
- Actual June 2012 count: 2,323,389

To estimate June 2012 rows:

- Assuming uniform distribution across months:
 - $29,557,973 \text{ (2012 total)} \div 12 \text{ months} = 2,463,164 \text{ rows expected per month}$

Comparing:

- Estimated (uniform distribution): 2,463,164
- Actual count: 2,323,389
- Difference: 139,775 fewer rows than estimated

The actual count (2,323,389) is reasonably close to our uniform distribution estimate, about 5.7% lower than expected.