FIT1043: Assignment 3 Report

Processing large data set using BASH shell scripts and R

Name: Eu Jia Xin

Student ID: 30881676

Date of Submission: 8/6/2020

Introduction

This aim of this assignment is to utilise BASH shell scripts and R programming language to read and process large data set. For this assignment, the data used is a pre-processed twitter dataset for 15th May 2020, filtered by COVID-19 related keywords. This dataset contains variety of information including user information, tweet content, follower counts and related information. The data will be read and processed using BASH shell scripts, and visualisations will be generated using R Studio. Instructions given from the assignment specifications will be included for reference. It should be noted that a $ sign will be used to indicate the start of the shell commands; and the code in blue represents R codes inputted (same format as in the console). The relevant codes and outputs will be shown, followed by analysis or additional information where necessary in the following sections of the report.

Part A: Inspecting the Data

1) Copy the downloaded file to your UNIX (Linux) terminal. State the size (in Bytes or MegaBytes) of the corona_tweets.csv.gz file and provide the shell command that you used to determine the size.

shell command:

```
$ ls -lh corona_tweets.csv.gz
```

output:

```
-rw-rw-r-- 1 jeuu0002 jeuu0002 118M May 31 13:33 corona_tweets.csv.gz
```

Analysis:

Based on the output, 118M represents 118 MegaBytes. Therefore, **the file size is 118 MegaBytes.**

2) The first line of the CSV file contains headers that are "tab" separated. What are the header names and provide the command you used to obtain it. Note that the command provided has to be in one line.

shell command:

```
$ cat corona_tweets.csv.gz | gunzip | head -1 | tr "\t" "\n"
```

output:

```
Created
Tweet_ID
Text
User_ID
User
User_Location
Followers_Count
Friends_Count
Geo
Place_Type
```

```
Place_Name
Place_Country
Language
```

Analysis:

To allow a clearer view of the column headers, every column header is outputted in a new line and can be seen in the output. **Based on the output, there are 13 headers in total, which are: Created, Tweet_ID, Text, User_ID, User, User_Location, Followers_Count, Friends_Count, Geo, Place_Type, Place_Name, Place_Country and Language.**

3) How many lines are there in the dataset? Again, provide the single line code on how you obtained it.

shell command:

```
$ cat corona_tweets.csv.gz | gunzip | wc -l
```

output:

```
1143559
```

Analysis:

There are 1143559 lines in the dataset.

Part B: Information from data

1) How many unique twitter users are there in the dataset. Provide the single line code that uses the “`awk`” and “`uniq`” command. You are also required to read the “`man`” pages of the “`uniq`” command to figure out if it is sufficient to answer the question. Provide explanation on the code you provided.

shell command:

```
$ cat corona_tweets.csv.gz | gunzip | awk -F '\t' '{print $4}' | head -10
```

output:

```
User_ID
3269308680
255800212
803042908345696300
350289248
812218284640608300
3808461979
1161532339052261400
938616431259934700
1243052659944198100
```

First 10 rows are outputted to make sure we are using an appropriate column. Becuase `User_ID` serves as an identifier for different users, this column will be used to find the number of unique twitter users.

shell command:

```
$ cat corona_tweets.csv.gz | gunzip | awk -F '\t' '{print $4}' | grep -v
'User_ID' | sort | uniq | wc -l
```

output:

```
641975
```

Analysis:

The `awk` command is used to extract only the `User_ID` column. Then, `grep` command is used to exclude the column header, as we only want to know the number of unique twitter users. In the manual of the `uniq` command, it says that the command “filters adjacent matching lines”, which means that we are required to `sort` if first to retrieve the actual number of unique twitter users. **The total number of unique twitter users is 641975.**

2) For each of the sub-question below, provide the single line code (one each) and briefly explain your code.

a) How many tweets mentioned the word “death” in any combination of uppercase or lowercase letters.

shell command:

```
$ cat corona_tweets.csv.gz | gunzip | cut -f 3 | grep -i "death" | wc -l
```

output:

```
50911
```

Analysis:

The `cut` command is used to extract the 3rd column of the dataset (`Text`). Then, `grep` command is to find tweets with the word ‘death’ in it. The `-i` flag is used to ignore case as we want the word ‘death’ in any combination of uppercase or lowercase letters. Then, `wc -l` is used to show how many lines there are, which is how many tweets there are. The output shows that there are 50911 tweets containing the word ‘death’.

b) How many of those are not spelt exactly “death”, “deaths”, “Death” or “Deaths” but in other combination of uppercase and lowercase (e.g. DEath, deatHs)

shell command:

```
$ cat corona_tweets.csv.gz | gunzip | cut -f 3 | grep -i 'death' | grep -v
-E 'death|deaths|Death|Deaths' | wc -l
```

output:

```
450
```

Analysis:

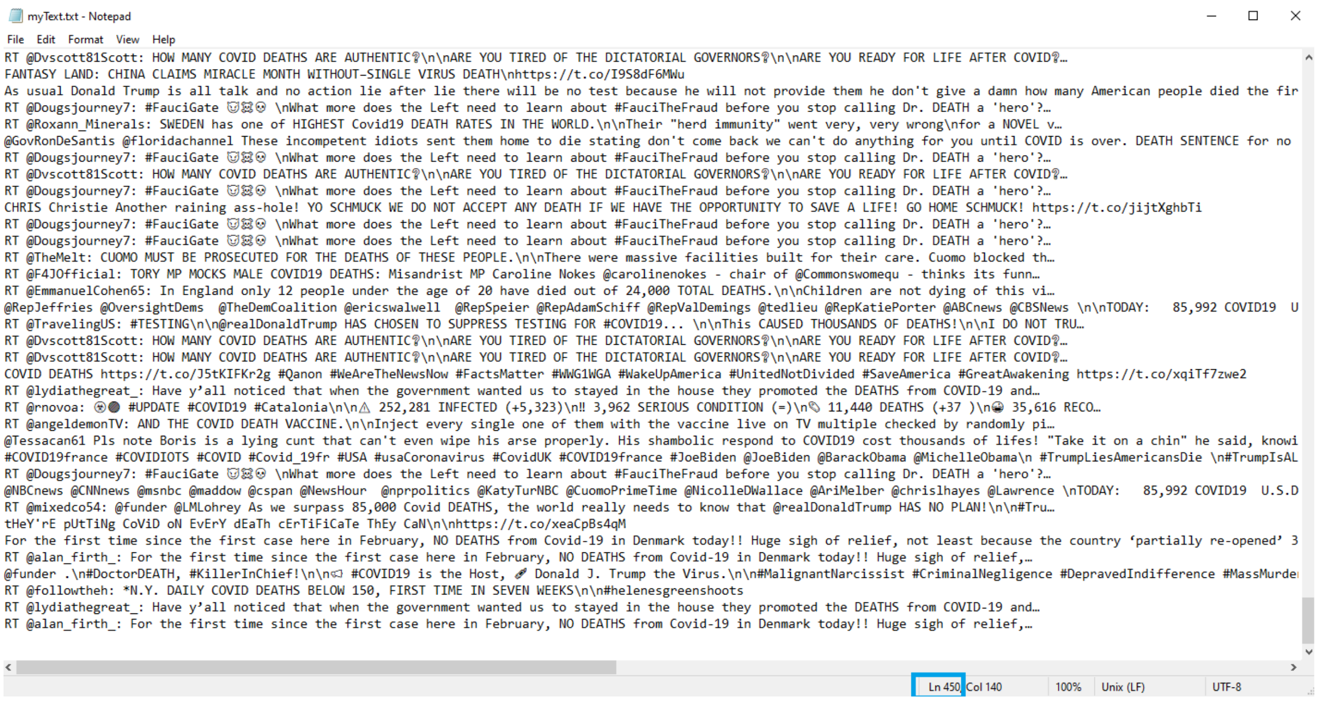
To find the answer, we will modify the code from 2a. There will an additional `grep` command. For the second new `grep` command, the `-v` flag is used to filter out the unwanted words. The `-E` flag is used to make sure that the program searches and excludes 4 different words, instead of searching for a single long word. Based on the output, there are **450 tweets** that contain the word 'death' (case insensitive), but are not spelt exactly as "death", "deaths", "Death" or "Deaths".

c) Output the lines of sub-part 2 (b) into a file called `myText.txt` (not the number of lines but the specific lines).

shell command:

```
$ cat corona_tweets.csv.gz | gunzip | cut -f 3 | grep -i 'death' | grep -v
-E 'death|deaths|Death|Deaths' > myText.txt
```

To check if the `.txt` file is correct, it is opened in notepad. When scrolled to the bottom, the bottom right of the terminal (highlighted in a blue rectangle) shows that there are 450 lines in total, which matches the previous code in 2b.


myText.txt - Notepad

File Edit Format View Help

```
RT @Dvscott81Scott: HOW MANY COVID DEATHS ARE AUTHENTIC❓\n\nARE YOU TIRED OF THE DICTATORIAL GOVERNORS❓\n\nARE YOU READY FOR LIFE AFTER COVID❓…
FANTASY LAND: CHINA CLAIMS MIRACLE MONTH WITHOUT-SINGLE VIRUS DEATH\nhttps://t.co/I9S8dF6MWu
As usual Donald Trump is all talk and no action lie after lie there will be no test because he will not provide them he don't give a damn how many American people died the fir
RT @Dougsjourney7: #FauciGate 😈☠💀 \nWhat more does the Left need to learn about #FauciTheFraud before you stop calling Dr. DEATH a 'hero'?…
RT @Roxann_Minerals: SWEDEN has one of HIGHEST Covid19 DEATH RATES IN THE WORLD.\n\nTheir "herd immunity" went very, very wrong\nfor a NOVEL v…
@GovRonDeSantis @floridachannel These incompetent idiots sent them home to die stating don't come back we can't do anything for you until COVID is over. DEATH SENTENCE for no
RT @Dougsjourney7: #FauciGate 😈☠💀 \nWhat more does the Left need to learn about #FauciTheFraud before you stop calling Dr. DEATH a 'hero'?…
RT @Dvscott81Scott: HOW MANY COVID DEATHS ARE AUTHENTIC❓\n\nARE YOU TIRED OF THE DICTATORIAL GOVERNORS❓\n\nARE YOU READY FOR LIFE AFTER COVID❓…
RT @Dougsjourney7: #FauciGate 😈☠💀 \nWhat more does the Left need to learn about #FauciTheFraud before you stop calling Dr. DEATH a 'hero'?…
CHRIS Christie Another raining ass-hole! YO SCHMUCK WE DO NOT ACCEPT ANY DEATH IF WE HAVE THE OPPORTUNITY TO SAVE A LIFE! GO HOME SCHMUCK! https://t.co/jijtXghbTi
RT @Dougsjourney7: #FauciGate 😈☠💀 \nWhat more does the Left need to learn about #FauciTheFraud before you stop calling Dr. DEATH a 'hero'?…
RT @Dougsjourney7: #FauciGate 😈☠💀 \nWhat more does the Left need to learn about #FauciTheFraud before you stop calling Dr. DEATH a 'hero'?…
RT @TheMelt: CUOMO MUST BE PROSECUTED FOR THE DEATHS OF THESE PEOPLE.\n\nThere were massive facilities built for their care. Cuomo blocked th…
RT @F4JOfficial: TORY MP MOCKS MALE COVID19 DEATHS: Misandrist MP Caroline Nokes @carolinenokes - chair of @Commonswomequ - thinks its funn…
RT @EmmanuelCohen65: In England only 12 people under the age of 20 have died out of 24,000 TOTAL DEATHS.\n\nChildren are not dying of this vi…
@RepJeffries @OversightDems  @TheDemCoalition @ericswalwell  @RepSpeier @RepAdamSchiff @RepValDemings @tedlieu @RepKatiePorter @ABCnews @CBSNews \n\nTODAY:   85,992 COVID19  U
RT @TravelingUS: #TESTING\n\n@realDonaldTrump HAS CHOSEN TO SUPPRESS TESTING FOR #COVID19... \n\nThis CAUSED THOUSANDS OF DEATHS!\n\nI DO NOT TRU…
RT @Dvscott81Scott: HOW MANY COVID DEATHS ARE AUTHENTIC❓\n\nARE YOU TIRED OF THE DICTATORIAL GOVERNORS❓\n\nARE YOU READY FOR LIFE AFTER COVID❓…
RT @Dvscott81Scott: HOW MANY COVID DEATHS ARE AUTHENTIC❓\n\nARE YOU TIRED OF THE DICTATORIAL GOVERNORS❓\n\nARE YOU READY FOR LIFE AFTER COVID❓…
COVID DEATHS https://t.co/J5tKIFKr2g #Qanon #WeAreTheNewsNow #FactsMatter #WWG1WGA #WakeUpAmerica #UnitedNotDivided #SaveAmerica #GreatAwakening https://t.co/xqiTf7zwe2
RT @lydiathegreat_: Have y'all noticed that when the government wanted us to stayed in the house they promoted the DEATHS from COVID-19 and…
RT @rnovoa: ☢● #UPDATE #COVID19 #Catalonia\n\n⚠ 252,281 INFECTED (+5,323)\n‼ 3,962 SERIOUS CONDITION (=)\n📎 11,440 DEATHS (+37 )\n😀 35,616 RECO…
RT @angeldemonTV: AND THE COVID DEATH VACCINE.\n\nInject every single one of them with the vaccine live on TV multiple checked by randomly pi…
@Tessacan61 Pls note Boris is a lying cunt that can't even wipe his arse properly. His shambolic respond to COVID19 cost thousands of lifes! "Take it on a chin" he said, knowi
#COVID19france #COVIDIOTS #COVID #Covid_19fr #USA #usaCoronavirus #CovidUK #COVID19france #JoeBiden @JoeBiden @BarackObama @MichelleObama\n #TrumpLiesAmericansDie \n#TrumpIsAL
RT @Dougsjourney7: #FauciGate 😈☠💀 \nWhat more does the Left need to learn about #FauciTheFraud before you stop calling Dr. DEATH a 'hero'?…
@NBCnews @CNNnews @msnbc @maddow @cspan @NewsHour  @nprpolitics @KatyTurNBC @CuomoPrimeTime @NicolleDWallace @AriMelber @chrislhayes @Lawrence \nTODAY:   85,992 COVID19  U.S.D
RT @mixedco54: @funder @LMLohrey As we surpass 85,000 Covid DEATHS, the world really needs to know that @realDonaldTrump HAS NO PLAN!\n\n#Tru…
tHeY'rE pUtTiNg CoViD oN EvErY dEaTh cErTiFiCaTe ThEy CaN\n\nhttps://t.co/xeaCpBs4qM
For the first time since the first case here in February, NO DEATHS from Covid-19 in Denmark today!! Huge sigh of relief, not least because the country 'partially re-opened' 3
RT @alan_firth_: For the first time since the first case here in February, NO DEATHS from Covid-19 in Denmark today!! Huge sigh of relief,…
@funder .\n#DoctorDEATH, #KillerInChief!\n\n📢 #COVID19 is the Host, 🖊 Donald J. Trump the Virus.\n\n#MalignantNarcissist #CriminalNegligence #DepravedIndifference #MassMurde
RT @followtheh: *N.Y. DAILY COVID DEATHS BELOW 150, FIRST TIME IN SEVEN WEEKS\n\n#helenesgreenshoots
RT @lydiathegreat_: Have y'all noticed that when the government wanted us to stayed in the house they promoted the DEATHS from COVID-19 and…
RT @alan_firth_: For the first time since the first case here in February, NO DEATHS from Covid-19 in Denmark today!! Huge sigh of relief,…
```

Ln 450, Col 140 | 100% | Unix (LF) | UTF-8

Part C: Data aggregation

1) Let's group the twitter user (ID) by the number of followers that they have into the following ranges. Provide the code for them. One line of code for each of them below (11 lines).

a) <= 1000

shell command:

```
$ cat corona_tweets.csv.gz | gunzip | awk -F'\t' '$7<=1000 {print $4,$7}'
| sort -u | less
```

The `less` command is used to show the user ID and follower count. A manual inspection was done to find possible repeating user IDs. A specific user ID 100003276 is shown to be outputted twice. And the follower count is 280 and 281. For a clearer output, the grep command is used to show only the specific user ID using the first attempted code.

shell command:

```
$ cat corona_tweets.csv.gz | gunzip | awk -F'\t' '$7<=1000 {print $4,$7}'
| sort -u | grep '100003276'
```

output:

```
100003276 280

100003276 281
```

This shows that the user_id has an increase in follower by 1, which proved the problem cannot be solved by directly sorting with a -u flag as above.

From this, we can deduce that there are duplicated user IDs even though we have attempted to sort it with the `-u` flag. This is most likely because the same user might have tweeted multiple times a day, creating multiple records. And within the time intervals, they might have gained or lost followers, causing the follower count to be different. Therefore, we will obtain the most accurate follower count by taking the latest entry of the record.

We will try to rectify this using the following codes and checking if the specific user ID chosen earlier will only output follower count of 281.

shell command:

```
$ cat corona_tweets.csv.gz | gunzip | awk -F'\t' '{print $4,$7}' | sort -r
-u | awk '!seen[$1]++' | awk -F' ' '$2<=1000 {print $1,$2}' | grep
'100003276'
```

output:

```
100003276 281
```

Code breakdown:

The first awk command will select the `User_ID` and `Followers_Count` only. Then, we will sort using two flags, `-r` and `-u` to make sure that the latest record of unique users with duplicate records will be shown at the top. Then, awk `'!seen[$1]++` is used to only take the first record of any user IDs that are repeated. The final awk command will filter out user IDs based on the follower count range intended.

Since this code takes the latest record for the specific user ID chosen, we assume this is correct for now, and will use the same approach to find the rest of the followers range. Instead of outputting the user IDs and follower count, we will directly count the number of records using wc `-l`. Afterwards, the sum of all 11 lines results is totalled up and compared to answer from b2 (641975 unique users) to double confirm if the code is correct.

a) <= 1000

shell command:

```
$ cat corona_tweets.csv.gz | gunzip | awk -F'\t' '{print $4,$7}' | sort -r
-u | awk '!seen[$1]++' | awk -F' ' '$2<=1000  {print $1,$2}' | wc -l
```

output:

```
455751
```

b) 1001 to 2000

shell command:

```
$ cat corona_tweets.csv.gz | gunzip | awk -F'\t' '{print $4,$7}' | sort -r
-u | awk '!seen[$1]++' | awk -F' ' '$2>1000 && $2<=2000  {print $1,$2}' |
wc -l
```

output:

```
68527
```

c) 2001 to 3000

shell command:

```
$ cat corona_tweets.csv.gz | gunzip | awk -F'\t' '{print $4,$7}' | sort -r
-u | awk '!seen[$1]++' | awk -F' ' '$2>2000 && $2<=3000  {print $1,$2}' |
wc -l
```

output:

```
31264
```

d) 3001 to 4000

shell command:

```
$ cat corona_tweets.csv.gz | gunzip | awk -F'\t' '{print $4,$7}' | sort -r
-u | awk '!seen[$1]++' | awk -F' ' '$2>3000 && $2<=4000  {print $1,$2}' |
wc -l
```

output:

```
18798
```

e) 4001 to 5000

shell command:

```
$ cat corona_tweets.csv.gz | gunzip | awk -F'\t' '{print $4,$7}' | sort -r
-u | awk '!seen[$1]++' | awk -F' ' '$2>4000 && $2<=5000  {print $1,$2}' |
wc -l
```

output:

```
11689
```

f) 5001 to 6000

shell command:

```
$ cat corona_tweets.csv.gz | gunzip | awk -F'\t' '{print $4,$7}' | sort -r
-u | awk '!seen[$1]++' | awk -F' ' '$2>5000 && $2<=6000  {print $1,$2}' |
wc -l
```

output:

```
7975
```

g) 6001 to 7000

shell command:

```
$ cat corona_tweets.csv.gz | gunzip | awk -F'\t' '{print $4,$7}' | sort -r
-u | awk '!seen[$1]++' | awk -F' ' '$2>6000 && $2<=7000  {print $1,$2}' |
wc -l
```

output:

```
5874
```

h) 7001 to 8000

shell command:

```
$ cat corona_tweets.csv.gz | gunzip | awk -F'\t' '{print $4,$7}' | sort -r
-u | awk '!seen[$1]++' | awk -F' ' '$2>7000 && $2<=8000  {print $1,$2}' |
wc -l
```

output:

```
4367
```

i) 8001 to 9000

shell command:

```
$ cat corona_tweets.csv.gz | gunzip | awk -F'\t' '{print $4,$7}' | sort -r
-u | awk '!seen[$1]++' | awk -F' ' '$2>8000 && $2<=9000  {print $1,$2}' |
wc -l
```

output:

```
3522
```

j) 9001 to 10000

shell command:

```
$ cat corona_tweets.csv.gz | gunzip | awk -F'\t' '{print $4,$7}' | sort -r
-u | awk '!seen[$1]++' | awk -F' ' '$2>9000 && $2<=10000  {print $1,$2}' |
wc -l
```

output:

```
2738
```

k) More than 10000

shell command:

```
$ cat corona_tweets.csv.gz | gunzip | awk -F'\t' '{print $4,$7}' | sort -r
-u | awk '!seen[$1]++' | awk -F' ' '$2>10000  {print $1,$2}' | grep -v
'User_ID' | wc -l
```

output:

```
31470
```

Analysis:

By summing up all the outputs from a to k, we know that there is a total of 641975 users, which proves that the code is correct.

2) Use (modified) the code above and include them into a shell script (with the extension `.sh`) to output a CSV file. The CSV file should contain two columns, the first column is the range and the second column is the number of twitter users.

```
$ nano c2shell.sh
```

Inside the nano editor:

```
#!/bin/bash


echo "<=1k,`cat corona_tweets.csv.gz | gunzip | awk -F'\t' '{print $4,$7}' | sort
-r -u | awk '!seen[$1]++' | awk -F' ' '$2<=1000  {print $1,$2}' | wc -l`" >
follower_count.csv
echo "1-2k,`cat corona_tweets.csv.gz | gunzip | awk -F'\t' '{print $4,$7}' | sort
-r -u | awk '!seen[$1]++' | awk -F' ' '$2>1000 && $2<=2000  {print $1,$2}' | wc -
l`" >> follower_count.csv
echo "2-3k,`cat corona_tweets.csv.gz | gunzip | awk -F'\t' '{print $4,$7}' | sort
-r -u | awk '!seen[$1]++' | awk -F' ' '$2>2000 && $2<=3000  {print $1,$2}' | wc -
l`" >> follower_count.csv
echo "3-4k,`cat corona_tweets.csv.gz | gunzip | awk -F'\t' '{print $4,$7}' | sort
-r -u | awk '!seen[$1]++' | awk -F' ' '$2>3000 && $2<=4000  {print $1,$2}' | wc -
l`" >> follower_count.csv
echo "4-5k,`cat corona_tweets.csv.gz | gunzip | awk -F'\t' '{print $4,$7}' | sort
-r -u | awk '!seen[$1]++' | awk -F' ' '$2>3000 && $2<=4000  {print $1,$2}' | wc -
l`" >> follower_count.csv
echo "5-6k,`cat corona_tweets.csv.gz | gunzip | awk -F'\t' '{print $4,$7}' | sort
-r -u | awk '!seen[$1]++' | awk -F' ' '$2>3000 && $2<=4000  {print $1,$2}' | wc -
l`" >> follower_count.csv
echo "6-7k,`cat corona_tweets.csv.gz | gunzip | awk -F'\t' '{print $4,$7}' | sort
-r -u | awk '!seen[$1]++' | awk -F' ' '$2>3000 && $2<=4000  {print $1,$2}' | wc -
l`" >> follower_count.csv
```

```
echo "7-8k,`cat corona_tweets.csv.gz | gunzip | awk -F'\t' '{print $4,$7}' | sort
-r -u | awk '!seen[$1]++' | awk -F' ' '$2>3000 && $2<=4000  {print $1,$2}' | wc -
l`" >> follower_count.csv
echo "8-9k,`cat corona_tweets.csv.gz | gunzip | awk -F'\t' '{print $4,$7}' | sort
-r -u | awk '!seen[$1]++' | awk -F' ' '$2>3000 && $2<=4000  {print $1,$2}' | wc -
l`" >> follower_count.csv
echo "9-10k,`cat corona_tweets.csv.gz | gunzip | awk -F'\t' '{print $4,$7}' | sort
-r -u | awk '!seen[$1]++' | awk -F' ' '$2>9000 && $2<=10000  {print $1,$2}' | wc -
l`" >> follower_count.csv
echo ">10k,`cat corona_tweets.csv.gz | gunzip | awk -F'\t' '{print $4,$7}' | sort
-r -u | awk '!seen[$1]++' | awk -F' ' '$2>10000  {print $1,$2}' | grep -v
'User_ID' | wc -l`" >> follower_count.csv
```

The shell script is made using the nano editor. The line `#!/bin/bash` at the start is used to tell the program that the script should be interpreted and run by BASH shell. Then, there are 11 lines, each producing a record into the csv file. For every line, the echo command will be used to pipe the output into the csv file. The first column will be the follower count range, whereas the second column will be the number of users that are within the range. The second column is the code used to find the number of lines as in part C1. The code is enclosed with `` to make sure that the output of the code is piped into the csv. Aside from the first record, the remaining 10 lines uses >> instead of > to ensure that consecutive records are appended instead of overwriting previous lines.

shell command:

```
$ chmod a+x c2shell.sh
$ ./c2shell.sh
```

After executing the shell script, the csv file is created, we will check the output and open the file in Excel to ensure the shell script performed the tasks correctly.

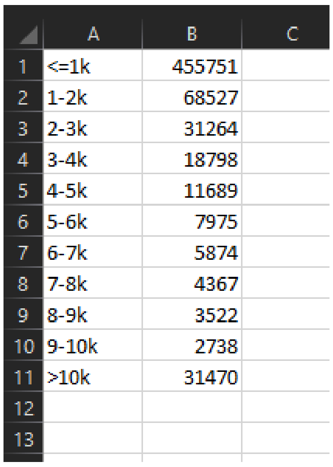
shell command:

```
$ cat follower_count.csv
```

output:

```
<=1k,455751
1-2k,68527
2-3k,31264
3-4k,18798
4-5k,11689
5-6k,7975
6-7k,5874
7-8k,4367
8-9k,3522
9-10k,2738
>10k,31470
```

Screenshot of the file opened in Excel:



	A	B	C
1	<=1k	455751	
2	1-2k	68527	
3	2-3k	31264	
4	3-4k	18798	
5	4-5k	11689	
6	5-6k	7975	
7	6-7k	5874	
8	7-8k	4367	
9	8-9k	3522	
10	9-10k	2738	
11	>10k	31470	
12			
13			

As the output and Excel file seems alright, we will proceed to visualise the follower count range in a bar chart. This will be done in R studio.

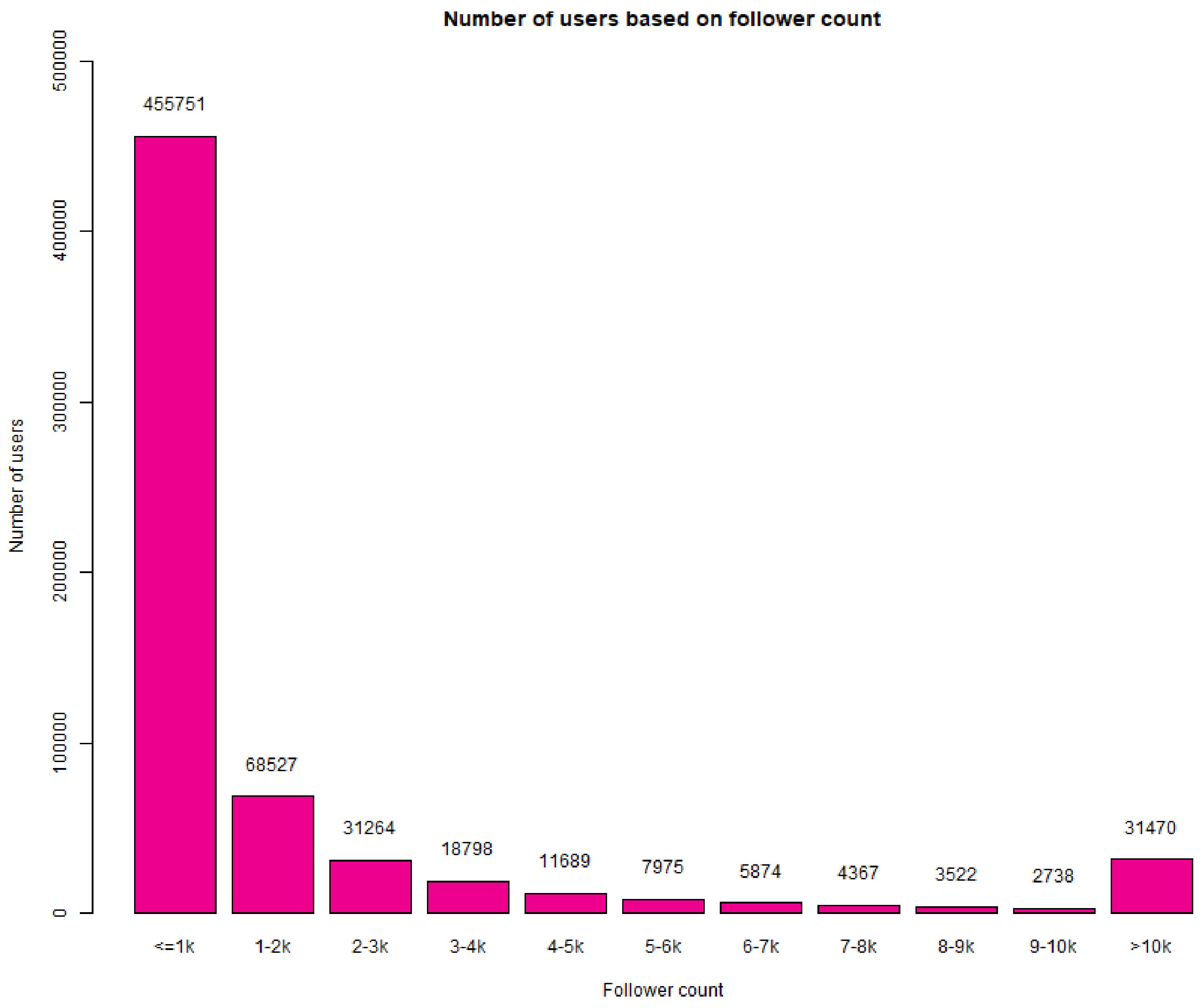
Visualisation

Checking the file in R:

```
> follower_count = read.table("follower_count.csv",header=FALSE,sep=',')
> follower_count
     V1     V2
1   <=1k 455751
2   1-2k  68527
3   2-3k  31264
4   3-4k  18798
5   4-5k  11689
6   5-6k   7975
7   6-7k   5874
8   7-8k   4367
9   8-9k   3522
10 9-10k   2738
11  >10k  31470
```

To make sure the y-axis values are shown nicely without the scientific notations, scipen option is adjusted to a larger value.

```
> options(scipen=100000)

> png("graph1.png",width= 850, height = 700)
> follower_graph = barplot(follower_count$V2,ylim = c(0,500000),main= "Number of users based on follower count",names.arg = follower_count$V1,xlab="Follower count",ylab= "Number of users",col = "#ED008E")
> text(x = follower_graph, y = follower_count$V2+20000, label=follower_count$V2)
> dev.off()
```



Part D: Small challenge

Having done Part C: Data aggregation, let's assume that we want to compare against tweets that are not retweets. We can assume that this is indicated at the beginning of the text by the "RT @".

1) Provide a single line code that filters out the tweets that contain "RT @" and output the results into a compressed gz file.

2) Do the same process as Part C (1) - (3) with this new file. Your R code can be a continuation from Part C.

3) Plot a side by side bar chart to visualise it. The bars should be coloured.

Similar to part c, a new user ID is randomly chosen from manual inspection beforehand to make sure that unique users are not double counted. The user ID selected is 100011309

shell command:

```
$ cat corona_tweets.csv.gz | gunzip | cut -f 3,4,7 | grep -v 'RT @' | awk
-F'\t' '{print $2,$3}' | grep '100011309'
```

output:

```
100011309 9417

100011309 9417

100011309 9417

100011309 9417

100011309 9417

100011309 9419
```

To ensure that only the last line is outputted, the following code is used:

Shell command:

```
$ cat corona_tweets.csv.gz | gunzip | cut -f 3,4,7 | grep -v 'RT @' | awk
-F'\t' '{print $2,$3}' | sort -r -u | awk '!seen[$1]++' | grep '100011309'
```

Output:

```
100011309 9419
```

Outputting the results into a compressed gz file:

shell command:

```
$ cat corona_tweets.csv.gz | gunzip | cut -f 3,4,7 | grep -v 'RT @' | awk
-F'\t' '{print $2,$3}' | sort -r -u | awk '!seen[$1]++' | gzip >
no_rt.csv.gz
```

a) Less than or equal to 1000

shell command:

```
$ cat no_rt.csv.gz | gunzip | awk -F' ' '$2<=1000 {print $1,$2}' | wc -l
```

output:

```
142248
```

b) 1001 to 2000

shell command:

```
$ cat no_rt.csv.gz | gunzip | awk -F' ' '$2>1000 && $2<=2000 {print
$1,$2}' | wc -l
```

output:

```
24030
```

c) 2001 to 3000

shell command:

```
$ cat no_rt.csv.gz | gunzip | awk -F' ' '$2>2000 && $2<=3000 {print
$1,$2}' | wc -l
```

```
11770
```

d) 3001 to 4000

shell command:

```
$ cat no_rt.csv.gz | gunzip | awk -F' ' '$2>3000 && $2<=4000 {print
$1,$2}' | wc -l
```

output:

```
7348
```

e) 4001 to 5000

shell command:

```
$ cat no_rt.csv.gz | gunzip | awk -F' ' '$2>4000 && $2<=5000 {print
$1,$2}' | wc -l
```

output:

```
4724
```

f) 5001 to 6000

shell command:

```
$ cat no_rt.csv.gz | gunzip | awk -F' ' '$2>5000 && $2<=6000 {print
$1,$2}' | wc -l
```

output:

```
3459
```

g) 6001 to 7000

shell command:

```
$ cat no_rt.csv.gz | gunzip | awk -F' ' '$2>6000 && $2<=7000 {print
$1,$2}' | wc -l
```

output:

```
2498
```

h) 7001 to 8000

shell command:

```
$ cat no_rt.csv.gz | gunzip | awk -F' ' '$2>7000 && $2<=8000 {print
$1,$2}' | wc -l
```

output:

```
1908
```

i) 8001 to 9000

shell command:

```
$ cat no_rt.csv.gz | gunzip | awk -F' ' '$2>8000 && $2<=9000 {print
$1,$2}' | wc -l
```

output:

```
1611
```

j) 9001 to 10000

shell command:

```
$ cat no_rt.csv.gz | gunzip | awk -F' ' '$2>9000 && $2<=10000 {print
$1,$2}' | wc -l
```

output:

```
1279
```

k) More than 10000

shell command:

```
$ cat no_rt.csv.gz | gunzip | awk -F' ' '$2>10000  {print $1,$2}' | grep -
v 'User_ID' | wc -l
```

output:

```
17015
```

Now, by using the same approach as in part c, a new shell script to create a new csv file containing the follower count range and number of users for non-retweets.

shell command:

```
$ nano d2shell.sh
```

Inside nano editor:

```
#!/bin/bash


echo "<=1k,`cat no_rt.csv.gz | gunzip | awk -F' ' '$2<=1000 {print $1,$2}' | wc -
l`" > no_rt_fcount.csv

echo "1-2k,`cat no_rt.csv.gz | gunzip | awk -F' ' '$2>1000 && $2<=2000 {print
$1,$2}' | wc -l`" >> no_rt_fcount.csv

echo "2-3k,`cat no_rt.csv.gz | gunzip | awk -F' ' '$2>2000 && $2<=3000 {print
$1,$2}' | wc -l`" >> no_rt_fcount.csv
```

```
echo "3-4k,`cat no_rt.csv.gz | gunzip | awk -F' ' '$2>3000 && $2<=4000 {print
$1,$2}' | wc -l`" >> no_rt_fcount.csv
echo "4-5k,`cat no_rt.csv.gz | gunzip | awk -F' ' '$2>4000 && $2<=5000 {print
$1,$2}' | wc -l`" >> no_rt_fcount.csv
echo "5-6k,`cat no_rt.csv.gz | gunzip | awk -F' ' '$2>5000 && $2<=6000 {print
$1,$2}' | wc -l`" >> no_rt_fcount.csv
echo "6-7k,`cat no_rt.csv.gz | gunzip | awk -F' ' '$2>6000 && $2<=7000 {print
$1,$2}' | wc -l`" >> no_rt_fcount.csv
echo "7-8k,`cat no_rt.csv.gz | gunzip | awk -F' ' '$2>7000 && $2<=8000 {print
$1,$2}' | wc -l`" >> no_rt_fcount.csv
echo "8-9k,`cat no_rt.csv.gz | gunzip | awk -F' ' '$2>8000 && $2<=9000 {print
$1,$2}' | wc -l`" >> no_rt_fcount.csv
echo "9-10k,`cat no_rt.csv.gz | gunzip | awk -F' ' '$2>9000 && $2<=10000 {print
$1,$2}' | wc -l`" >> no_rt_fcount.csv
echo ">10k,`cat no_rt.csv.gz | gunzip | awk -F' ' '$2>10000  {print $1,$2}' | grep
-v 'User_ID' | wc -l`" >> no_rt_fcount.csv
```

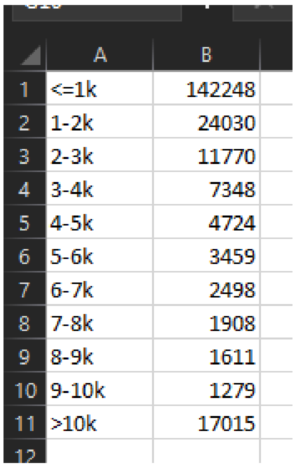
shell commands:

```
$ chmod a+x d2shell.sh
$ ./d2shell.sh
$ cat no_rt_fcount.csv
```

output:

```
<=1k,142248
1-2k,24030
2-3k,11770
3-4k,7348
4-5k,4724
5-6k,3459
6-7k,2498
7-8k,1908
8-9k,1611
9-10k,1279
>10k,17015
```

Checking the Excel file:



	A	B
1	<=1k	142248
2	1-2k	24030
3	2-3k	11770
4	3-4k	7348
5	4-5k	4724
6	5-6k	3459
7	6-7k	2498
8	7-8k	1908
9	8-9k	1611
10	9-10k	1279
11	>10k	17015
12		

The output and Excel file both looks alright, so we will proceed to visualising the results.

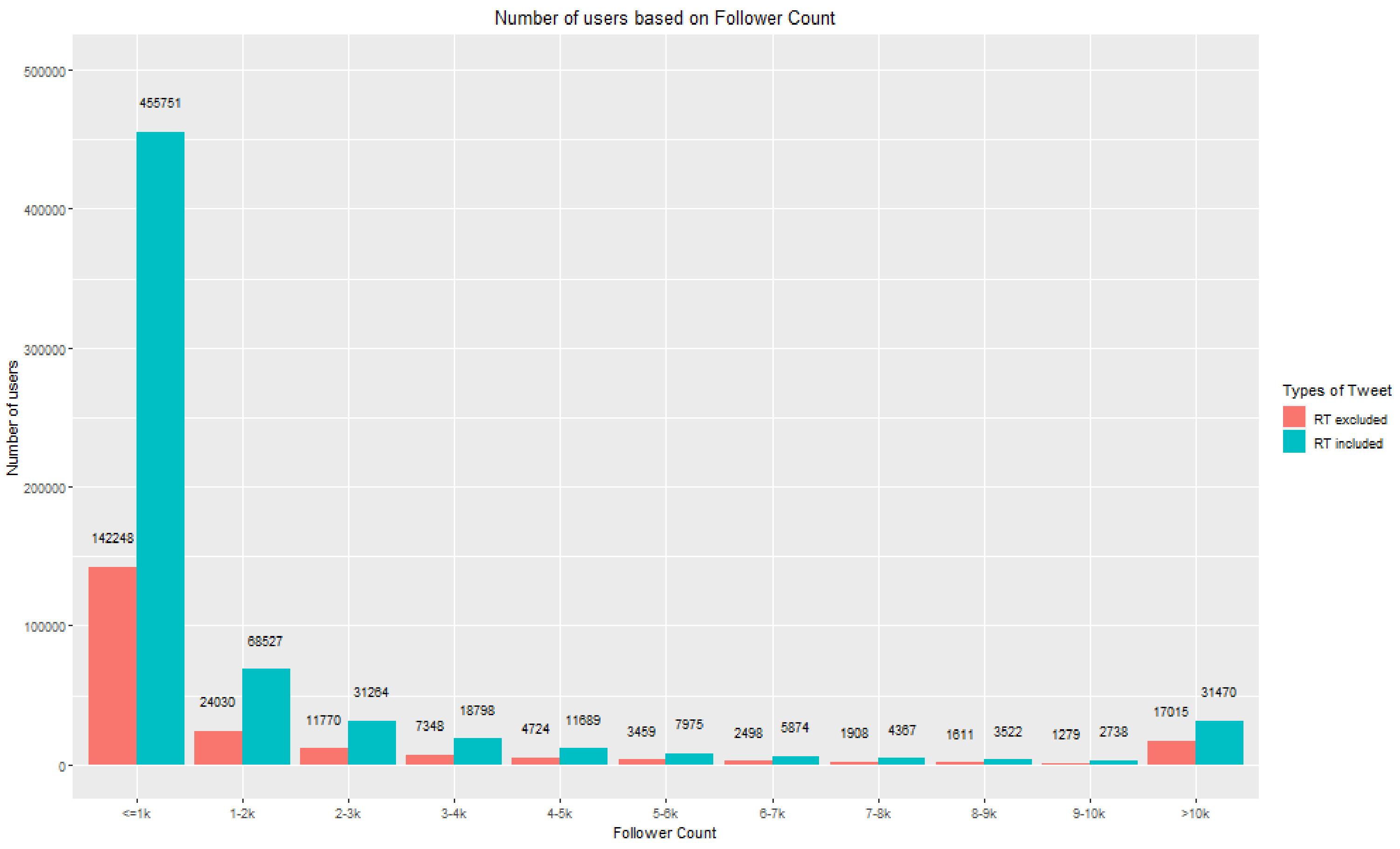
Visualisation (side-by-side bar chart)

For this section, we will be using ggplot to plot the side by side bar chart.

First, we will import the new csv file made, and create a new dataframe based on the two csv files. The necessary libraries are then installed to help plot the bar chart. The following codes are used to plot the graph. Some outputs are shown to display the flow of code.

```
> no_rt_fcount = read.table("no_rt_fcount.csv",header=FALSE,sep=',')
> no_rt_fcount
     V1     V2
1   <=1k 142248
2   1-2k  24030
3   2-3k  11770
4   3-4k   7348
5   4-5k   4724
6   5-6k   3459
7   6-7k   2498
8   7-8k   1908
9   8-9k   1611
10 9-10k   1279
11  >10k  17015

> both_fcount = data.frame(follower_count,no_rt_fcount$V2)
> both_fcount
     V1     V2 no_rt_fcount.V2
1   <=1k 455751          142248
2   1-2k  68527           24030
3   2-3k  31264           11770
4   3-4k  18798            7348
5   4-5k  11689            4724
6   5-6k   7975            3459
7   6-7k   5874            2498
8   7-8k   4367            1908
9   8-9k   3522            1611
10 9-10k   2738            1279
```

```
11  >10k  31470         17015

> names(both_fcount)[names(both_fcount) == 'V1'] <- 'Follower Count'
> names(both_fcount)[names(both_fcount) == 'V2'] <- 'RT included'
> names(both_fcount)[names(both_fcount) == 'no_rt_fcount.V2'] <- 'RT excluded'

> both_fcount
  Follower Count RT included RT excluded
1          <=1k     455751      142248
2          1-2k      68527       24030
3          2-3k      31264       11770
4          3-4k      18798        7348
5          4-5k      11689        4724
6          5-6k       7975        3459
7          6-7k       5874        2498
8          7-8k       4367        1908
9          8-9k       3522        1611
10        9-10k       2738        1279
11         >10k      31470       17015

> install.packages("tidyr")
> library(tidyr)
> library(ggplot2)

> df <- gather(both_fcount,'Types of Tweet', 'Number of users', 'RT included':'RT excluded')
> df
  Follower Count Types of Tweet Number of users
1          <=1k    RT included          455751
2          1-2k    RT included           68527
3          2-3k    RT included           31264
4          3-4k    RT included           18798
5          4-5k    RT included           11689
6          5-6k    RT included            7975
7          6-7k    RT included            5874
8          7-8k    RT included            4367
9          8-9k    RT included            3522
10        9-10k    RT included            2738
11         >10k    RT included           31470
12         <=1k    RT excluded          142248
13         1-2k    RT excluded           24030
14         2-3k    RT excluded           11770
15         3-4k    RT excluded            7348
16         4-5k    RT excluded            4724
17         5-6k    RT excluded            3459
18         6-7k    RT excluded            2498
19         7-8k    RT excluded            1908
20         8-9k    RT excluded            1611
21        9-10k    RT excluded            1279
22         >10k    RT excluded           17015

> df$`Follower Count`<- as.character(df$`Follower Count`)
> df$`Follower Count`<- factor(df$`Follower Count`,levels=unique(df$`Follower Count`))

> png("both_follower_graph.png",width= 1000, height = 600)
> plot <- ggplot(df, aes(x=`Follower Count`,y=`Number of users`,fill=`Types of Tweet`),width=10) +
coord_cartesian(ylim = c(0,500000))
> plot <- plot + geom_col(position = 'dodge') + ggtitle('Number of users based on Follower Count') +
theme(plot.title = element_text(hjust = 0.5))
> plot <- plot + geom_text(aes(label=`Number of users`, y= `Number of users`+
20000),position=position_dodge(width=0.9),vjust=0.25,size = 3)
> plot
> dev.off()
RStudioGD
        2
```



Conclusion

By using BASH shell scripts, large datasets can be processed in a short amount of time. The data can further be aggregated and manipulated efficiently, moreover we are able to generate filtered data for further processing. For example, a new compressed file which contains far less redundant data is created quickly, and the data is extracted into a csv file efficiently. This shows that the inter-process communication of piping in UNIX is extremely convenient for the manipulation of large data sets. Combined with R programming language, we are able to visualise the findings we obtained through visualisations like bar charts.