FIT1043 Introduction to Data Science Assignment 3

Name: Vanessa Khoo Ming Yi
Student ID: 31417493
Date: 24th May 2021

Introduction

This assignment is to test the students' knowledge on the use of the BASH (Unix) Shell and R (programming language) to manipulate and work with much larger datasets as compared to the ones used in assignment 1 and 2.

This assignment is to test these abilities:

1. Navigating within the BASH Shell
2. Processing large files using the BASH Shell and using online resources/"man" to aid in our writing of commands
3. Outputting processed files into CSV formats using the bash shell.
4. Reading a processed file in R and conducting visualisation with R

We are given a single dataset (very large size that has been compressed) for this assignment. It contains the data on Facebook posts from the top 15 mainstream media sources. In order to get insight and analysis on this dataset, we are to manipulate the data via the Bash Shell / Read outputted files into R for visualisation for analysis. Thus, this process of properly gaining insight via this data is what I believe entails the end goal of the assignment, along with testing the skills that are mentioned above.

This assignment is split into two parts, Part A: 'Investigating FB Data using shell commands', and Part B: 'Graphing data into R'. Therefore, I will then systematically approach this assignment starting from Part A to Part B.

Note: In the screenshots of my Cygwin terminal and working directory in R, you will see a name 'Aaron Khoo@LAPTOP...' in the file path. The reason for this is because I am using my brother's (Aaron Khoo) hand-me-down laptop.

PART A: Investigating FB Data using shell commands

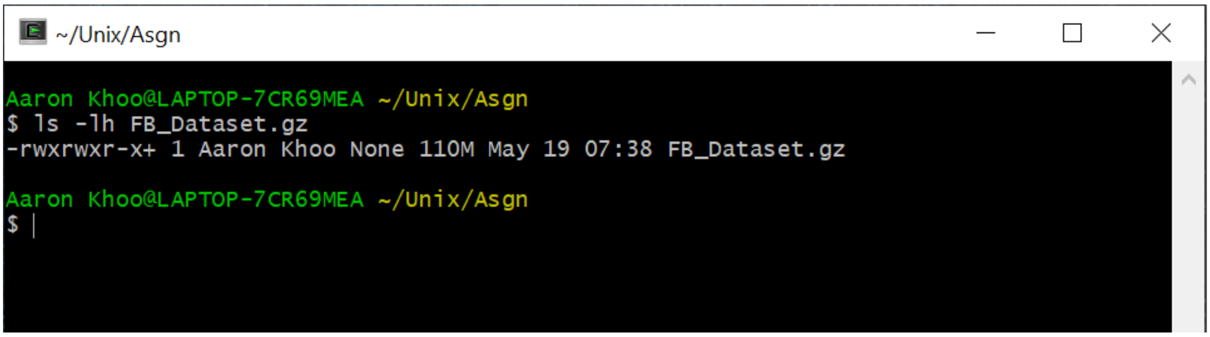
Task A1

(1.1) Original Zipped File Size

Code Input: (Unix)

```
ls -lh FB_Dataset.gz
```

Output:



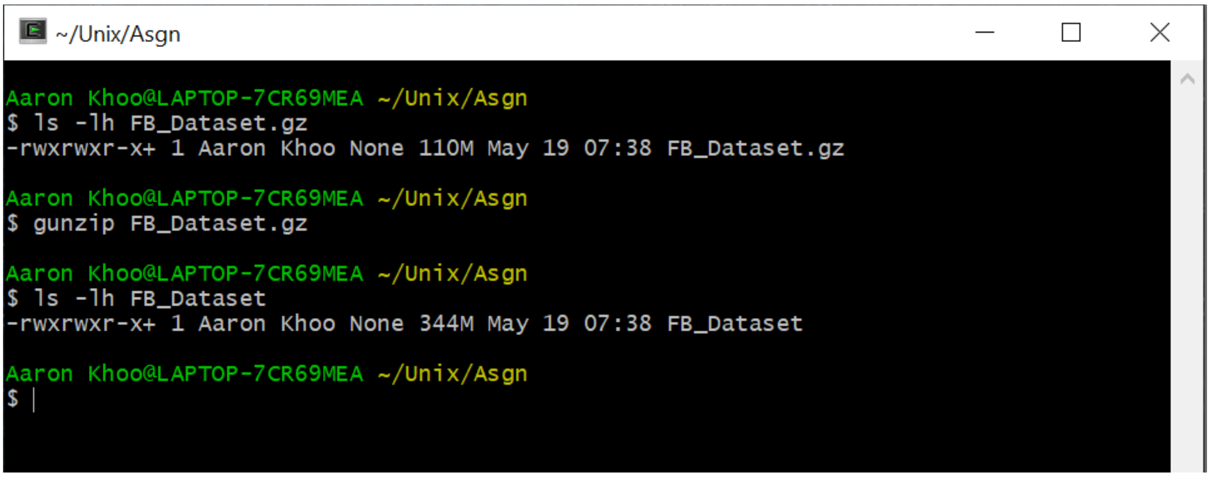
```
Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ ls -lh FB_Dataset.gz
-rwxrwxr-x+ 1 Aaron Khoo None 110M May 19 07:38 FB_Dataset.gz

Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ |
```

(1.2) Decompress file & Get decompressed file size

Code Input: (Unix)

```
gunzip FB_Dataset.gz
ls -lh FB_Dataset
```

Output:



```
Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ ls -lh FB_Dataset.gz
-rwxrwxr-x+ 1 Aaron Khoo None 110M May 19 07:38 FB_Dataset.gz

Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ gunzip FB_Dataset.gz

Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ ls -lh FB_Dataset
-rwxrwxr-x+ 1 Aaron Khoo None 344M May 19 07:38 FB_Dataset

Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ |
```

(1.3) Explanation, Justification & Answer for 1.1 & 1.2:

To check the filesize of a file, the ‘ls’ command is used to list the computer files and ‘-lh’ is a combination of ‘-l’ and ‘-h’ where –‘-l’ lists the files in a long listing format, and ‘-h’ prints human readable sizes.

To decompress the original gunzip file,the ‘gunzip’ command is then used which helps to decompress the file.

We can see that the original gunzipped file (FB_Dataset.gz) is 110Megabytes large, and the decompressed file size (FB_Dataset) is 344Megabytes large.

Let us note that the compression ratio = (uncompressed data size/compressed data size).

Therefore, the compression ratio for FB_Dataset.gz and FB_Dataset is:

344Mb:110Mb = 3.127:1. Thus, the uncompressed data file size is approximately 3.127 times larger than the compressed data file size.

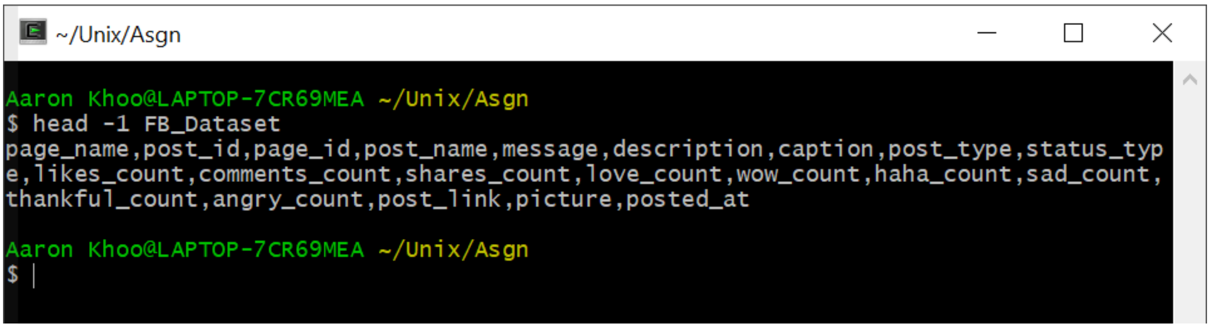
Task A2

(2.1) Delimiter & Number of Rows

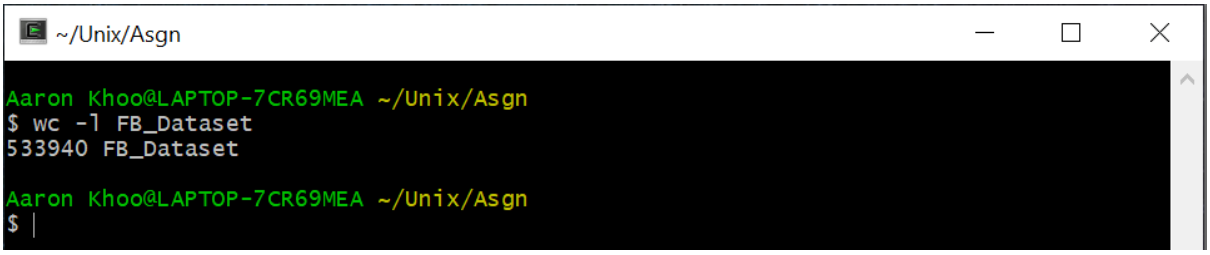
Code Input: (Unix)

```
head -1 FB_Dataset

wc -l FB_Dataset

awk 'END{print NR}' FB_Dataset
```

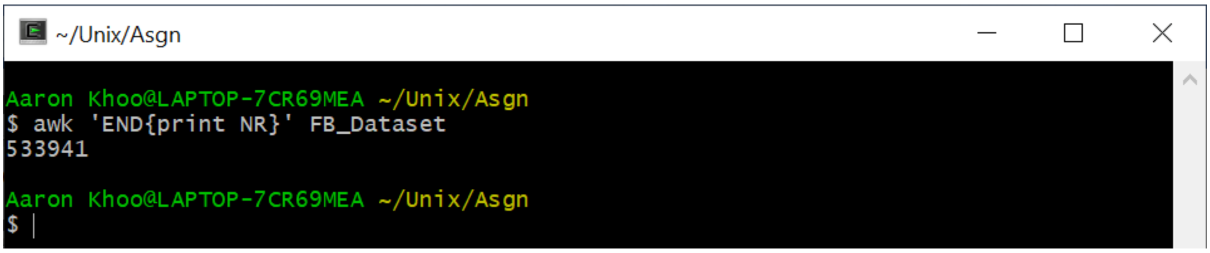
Output:



```
Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ head -1 FB_Dataset
page_name,post_id,page_id,post_name,message,description,caption,post_type,status_typ
e,likes_count,comments_count,shares_count,love_count,wow_count,haha_count,sad_count,
thankful_count,angry_count,post_link,picture,posted_at

Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ |
```



```
Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ wc -l FB_Dataset
533940 FB_Dataset

Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ |
```



```
Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ awk 'END{print NR}' FB_Dataset
533941

Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ |
```

(2.2) Explanation, Justification & Answer:

To find the delimiter of the file, the 'head -1' command is used to show the header (1st row) of the FB_Dataset file. With this, we can then observe and conclude that indeed, the **delimiter is a ','** (a comma), as this is what separates the column names as seen in the above output.

Note that the ‘head’ command is used to output the first part of the file, and ‘-1’ makes sure that it prints only the first line instead of the first 10 lines.

To find the number of rows in the file, the ‘wc -l’ and ‘awk’ command is used. From the output, it is shown that the **total number of rows in the file is 533940, not including the header** (wc command) and **533941 including the header** (awk command). Note that the ‘wc’ prints the number of newline, word & byte counts for the file, but ‘-l’ together with ‘wc’ makes sure that it only prints the newline counts, which gives the total lines/rows count **less one** because the last line probably does not have a ‘\n’ character at the end. Thus, with the awk command, I will be able to accurately find the **TOTAL number of rows in the file to be:**

533941 rows.

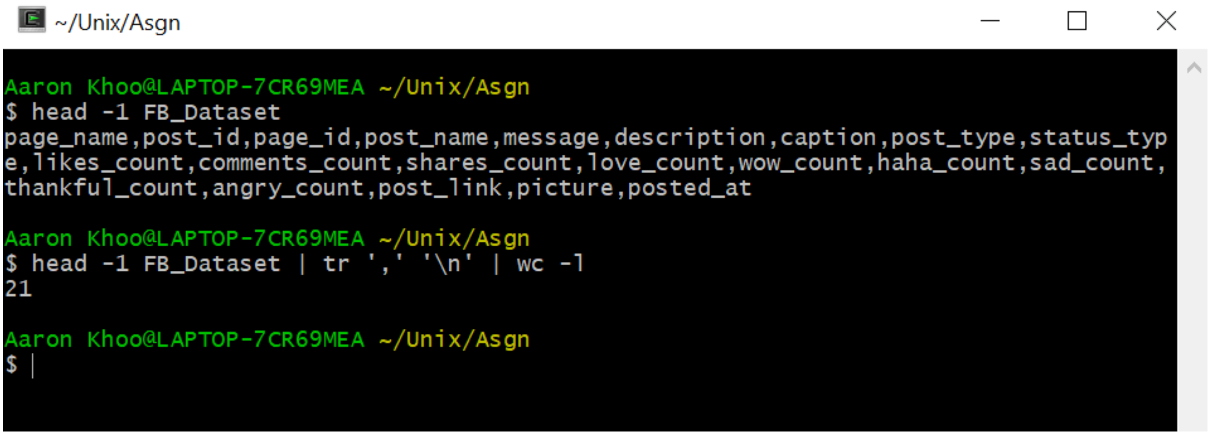
Task A3

(3.1) Column Names & Number of Columns

Code Input: (Unix)

```
head -1 FB_Dataset

head -1 FB_Dataset | tr ',' '\n' | wc -l
```

Output:



~/Unix/Asgn

```
Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ head -1 FB_Dataset
page_name,post_id,page_id,post_name,message,description,caption,post_type,status_typ
e,likes_count,comments_count,shares_count,love_count,wow_count,haha_count,sad_count,
thankful_count,angry_count,post_link,picture,posted_at

Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ head -1 FB_Dataset | tr ',' '\n' | wc -l
21

Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ |
```

(3.2) Explanation, Justification & Answer:

Column names are stored in the header, which is the first row of the file. Thus, the ‘head -1’ command is used to print the first row/header of the file. We can observe from the above output that **the column names are**:

page_name, post_id, page_id, post_name, message, description, caption, post_type, status_type, likes_count, comments_count, shares_count, love_count, wow_count, haha_count, sad_count, thankful_count, angry_count, post_link, picture, posted_at

Note that the ‘head’ command is used to output the first part of the file, and ‘-1’ makes sure that it prints only the first line instead of the first 10 lines.

To get the number of columns, we could always count manually through the header printed above, but in the case of using shell commands, here I used the `tr ',' '\n'` command which helps to delete the ',' characters in from the piped header text and transform/translate these commas to '\n', then inputs this for the next command 'wc -l' which helps to count the number of newline/lines count from the previous output and thus this results in the number of columns. Thus, from the above output, **we can see that the number of columns is 21.**

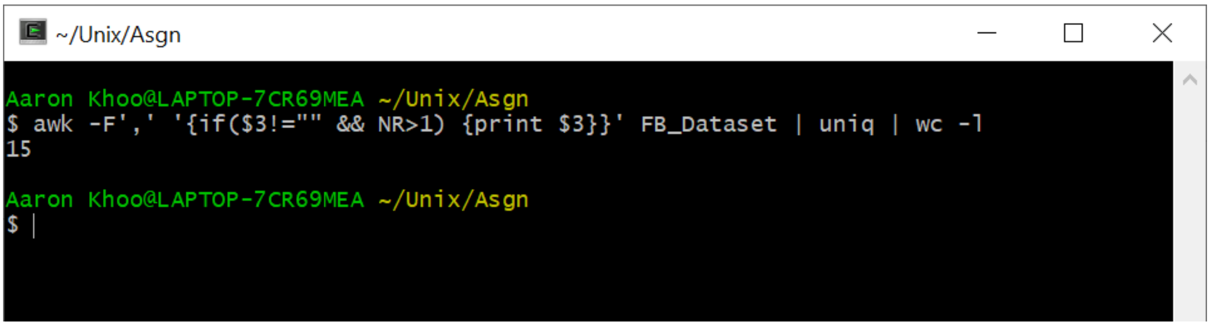
Task A4

(4.1) Unique Pages

Code Input: (Unix)

```
awk -F',' '{if($3!="" && NR>1) {print $3}}' FB_Dataset | uniq | wc -l
```

Output:



```
Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ awk -F',' '{if($3!="" && NR>1) {print $3}}' FB_Dataset | uniq | wc -l
15

Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ |
```

(4.2) Explanation, Justification & Answer:

To find the number of unique pages, let us first establish the definition of unique pages in my interpretation of it. So, **let us define a unique page as one that has a unique page_id.**

Thus, we can then count the number of unique page_id s in the file to find the number of unique pages.

To do this, as seen from the above code input, the first awk command is used to produce an output with all page_id s in all rows except the column header and made sure that empty string page_id s in the page_id column (which is the third column as seen by the use of $3) are excluded. This output is then piped to the next command which filters these lines to omit all repeated lines. Then to count the number of unique page_id s, we again use the 'wc -l' command to count the number of lines of the previous output that included all unique page ids in lines, thus this translates to the number of unique pages.

Therefore, from the above output, we can see that the **number of unique pages = 15.**

How the awk command is used:

-F is used to specify the field separator/delimiter, in this case is ','.

'{if($3!="" && NR>1} {print $3}}' is used to find the rows where the 3rd column is not empty and also to not include the header, then output these rows' 3rd column.

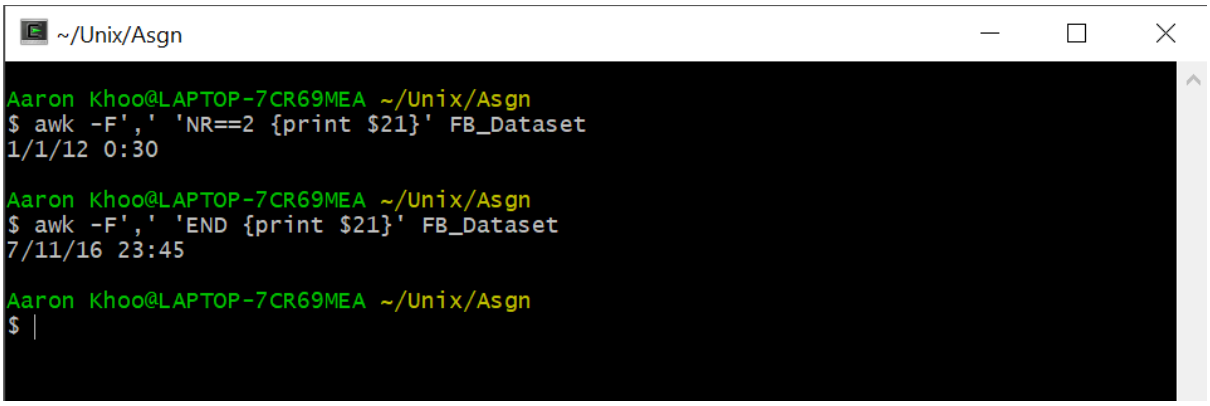
Task A5

(5.1) Date Range

Code Input: (Unix)

```
awk -F',' 'NR==2 {print $21}' FB_Dataset
awk -F',' 'END {print $21}' FB_Dataset
```

Output:



```
Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ awk -F',' 'NR==2 {print $21}' FB_Dataset
1/1/12 0:30

Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ awk -F',' 'END {print $21}' FB_Dataset
7/11/16 23:45

Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ |
```

(5.2) Explanation, Justification & Answer:

To find the date range of the posts in the file, we need to find the earliest post's date and the latest post's date. So, to find these dates, I used the awk command to manipulate the lines and thus find the specific row and column's data and output them.

Since the data is ordered by date in chronological order, we can see that the first awk command from the code I used above outputs the 21st column (posted_at column) in only the second row. Note that the second row is the first row with the data after the header (1st row). This output refers to the earliest post's date.

Then, the second awk command used outputs the 21st column in only the very last row. This output refers to the latest post's date.

Therefore, **the date range is from 1st January 2012 to 7th November 2016.**

And just an additional note, since the first and latest post was posted at difference times, I would like to quickly state the total difference in time for this date range, which is: 1772 days, 23 hours, 15 minutes. (timeanddate, 2021)

How the awk command is used:

-F is used to specify the field separator/delimiter, in this case is ','.

'NR==2 {print $21}' gets the second row and prints its 21st column.

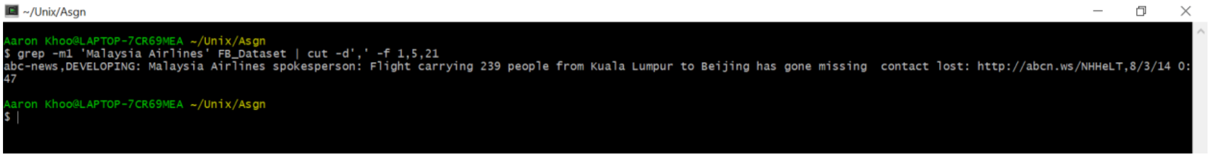
'END {print $21}' gets the last row and prints its 21st column.

Task A6

<u>Code Input: (Unix)</u>

```
grep -m1 'Malaysia Airlines' FB_Dataset | cut -d',' -f 1,5,21
```

<u>Output:</u>



```
Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ grep -m1 'Malaysia Airlines' FB_Dataset | cut -d',' -f 1,5,21
abc-news,DEVELOPING: Malaysia Airlines spokesperson: Flight carrying 239 people from Kuala Lumpur to Beijing has gone missing  contact lost: http://abcn.ws/NHHeLT,8/3/14 0:
47

Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ |
```

Explanation, Justification & Answer:

<u>Datetime of first mention:</u> 8th March 2014 at 12:47am.

<u>Message:</u> “`DEVELOPING: Malaysia Airlines spokesperson: Flight carrying 239 people from Kuala Lumpur to Beijing has gone missing  contact lost: http://abcn.ws/NHHeLT`”

<u>Media source that mentioned it:</u> abc-news

To find this first mention of “Malaysia Airlines”, what I interpret from the question is to find the first mention of it **across all columns,** instead of just finding for the first mention in only the message column.

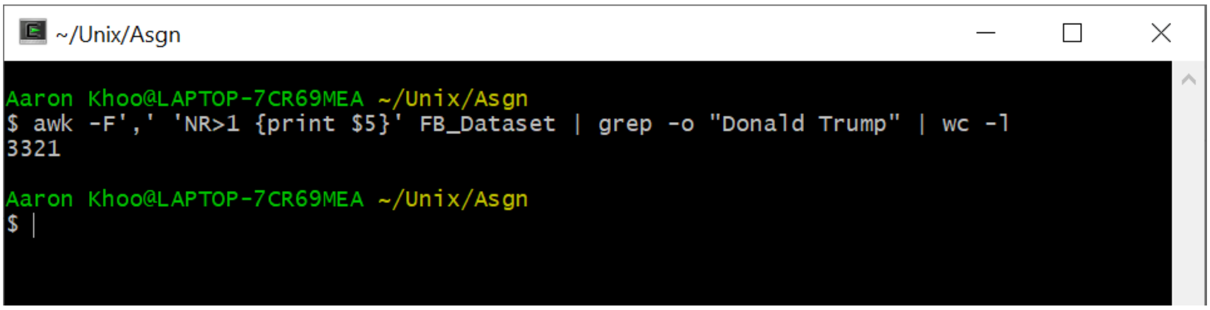
Thus, with the ‘grep -m1’ command used above; it helps find the first matching line with the ‘Malaysia Airlines’ pattern in FB_Dataset. This line output is then piped to the cut command used above to only output the 1st, 5th and 21st column to show the page_name, message and the posted_at data only. These corresponds to the media cource, message and when it was posted.

Task A7

Code Input: (Unix)

```
awk -F',' 'NR>1 {print $5}' FB_Dataset | grep -o "Donald Trump" | wc -l
```

Output:



```
Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ awk -F',' 'NR>1 {print $5}' FB_Dataset | grep -o "Donald Trump" | wc -l
3321

Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ |
```

Explanation, Justification & Answer:

I interpreted 'find the number of times "Donald Trump" is mentioned in message column' with the meaning that one row's message can have multiple mentions of "Donald Trump", and so **one row could have more than one 'mention count'.** For example, a message in one row like this: "Donald Trump has done this…. . Donald Trump is now…" would have a count of 2 mentions of "Donald Trump".

Therefore, how I found this number of times mentioned is by:

(i) First using an awk command to output only the message column of the dataset

(ii) Use grep to find the lines with the pattern of "Donald Trump", here we do NOT ignore the case. I also use '-o' alongside grep just to make sure that we only output the matching parts of the matching line, with each part on a separate output line.

(iii) Thus, since each mention/part takes up an output line, the 'wc -l' command then helps to count the number of lines from the output of the grep command, and this would then translate to the number of total mentions of "Donald Trump" in the message column.

Therefore, according to the output above, the number of mentions of the specific pattern ("Donald Trump") as asked by the question = 3321.

How the awk command is used:

-F is used to specify the field separator/delimiter, in this case is ','.

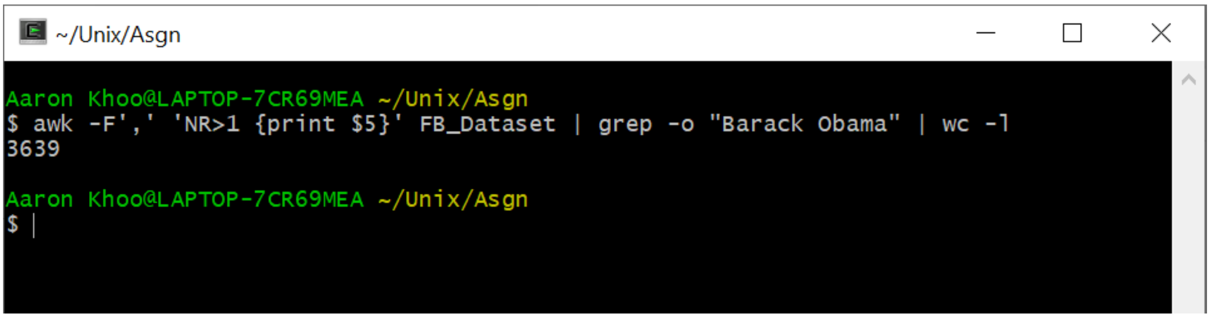
'NR>1 {print $5}' gets the rows not including the header and prints its 5th column.

Task A8

Code Input: (Unix)

```
awk -F',' 'NR>1 {print $5}' FB_Dataset | grep -o "Barack Obama" | wc -l
```

Output:



```
Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ awk -F',' 'NR>1 {print $5}' FB_Dataset | grep -o "Barack Obama" | wc -l
3639

Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ |
```

Explanation

I interpreted 'find the number of times "Barack Obama" is mentioned in message column' with the meaning that one row's message can have multiple mentions of "Barack Obama", and **so one row could have more than one 'mention count'.** For example, a message in one row like this: "Barack Obama and Michelle Obama has done this…. . Barack Obama is now…" would have a count of 2 mentions of "Barack Obama".

Therefore, how I found this number of times mentioned is the same as in A.7, by:

(i) First using an awk command to output only the message column of the dataset

(ii) Use grep to find the lines with the pattern of "Barack Obama", here we do NOT ignore the case. I also use '-o' alongside grep just to make sure that we only output the matching parts of the matching line, with each part on a separate output line.

(iii) Thus, since each mention/part takes up an output line, the 'wc -l' command then helps to count the number of lines from the output of the grep command, and this would then translate to the number of total mentions of "Barack Obama" in the message column.

Therefore, according to the output above, <u>the number of mentions of the specific pattern ("Barack Obama") as asked by the question = 3639.</u>

Let us now analyse the popularity between Trump and Obama:

Before we do that, we must first define what being 'popular' means in my interpretation.

From the oxford dictionary, *popular* means being liked or admired by many people or by a particular person or group. (oxfordlearnersdictionaries, n.d.)

Thus, in accordance with this dictionary's definition, let us define being 'popular' in this context as having a high number of mentions in the message of the FB posts in the dataset given. Thus, being *more* popular would mean having a HIGHER number of mentions in the message of the FB posts in the dataset.

Now, to reiterate, the number of mentions of Trump = 3321, whereas the number of mentions of Obama = 3639. We can then observe that Obama has a higher number of mentions in the message column of the posts for this dataset as compared to Trump. Therefore, we can conclude that **Obama is more popular on Facebook compared to Trump.**

How the awk command is used:

-F is used to specify the field separator/delimiter, in this case is ‘,’.

'NR>1 {print $5}' gets the rows not including the header and prints its 5th column.

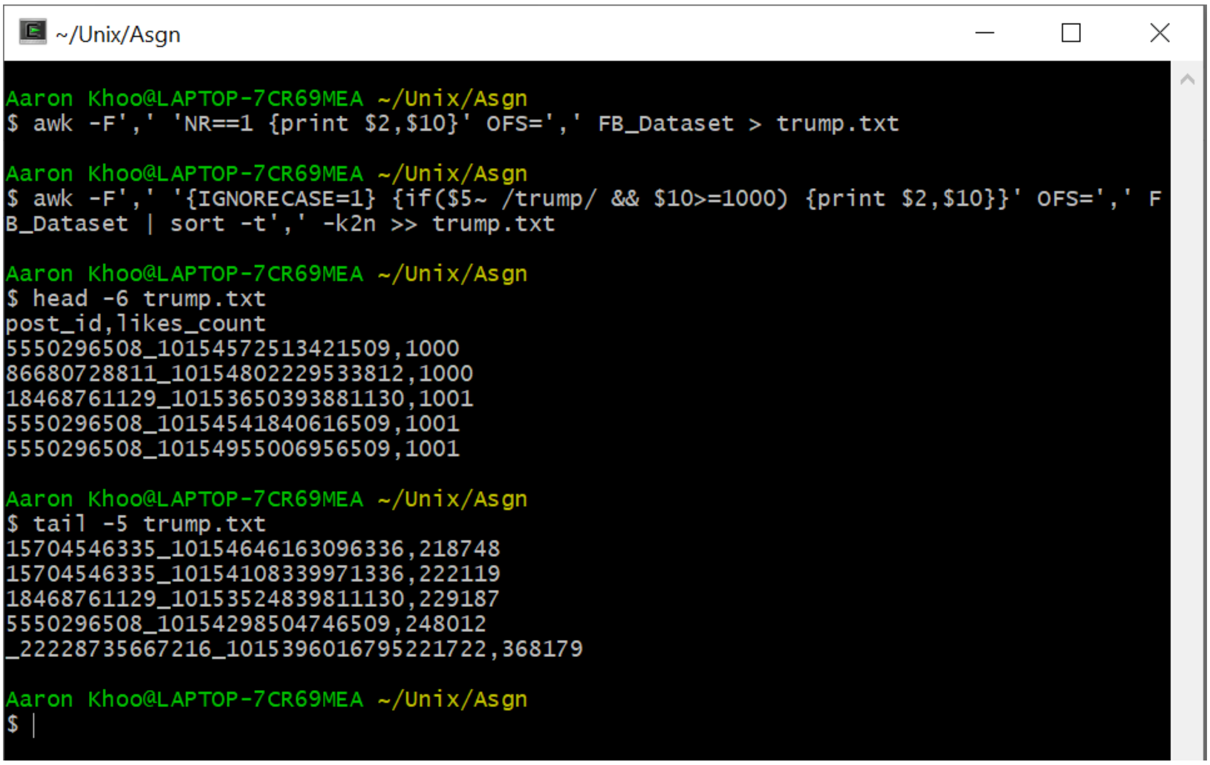
Task A9

Code Input: (Unix)

```
awk -F',' 'NR==1 {print $2,$10}' OFS=',' FB_Dataset > trump.txt

awk -F',' '{IGNORECASE=1} {if($5~ /trump/ && $10>=1000) {print $2,$10}}' OFS=','
FB_Dataset | sort -t',' -k2n >> trump.txt

head -6 trump.txt

tail -5 trump.txt
```

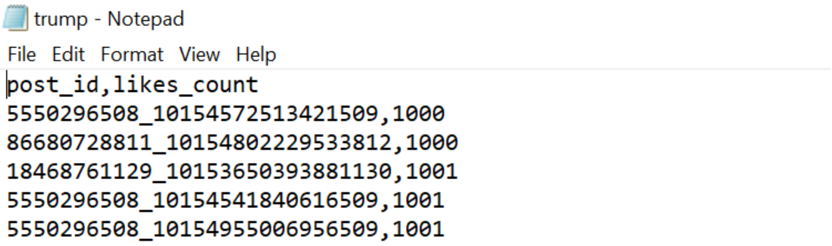
*Note: when copy-pasting the second awk command’s code from the pdf into the terminal, might have to copy the first line first, then input a ‘space’, then only copy the second line (‘FB_dataset | sort…’) after. This is something I encountered when trying to copy-paste multiple lines from pdfs. I think this is because pdfs don’t take the space after “OFS=’,’” into account, since copy-pasting the same code works from the word document.

Output:



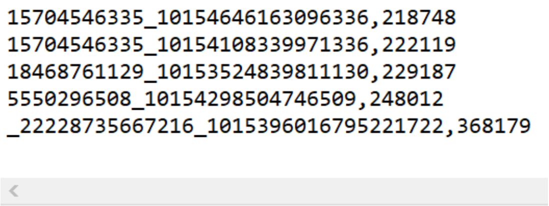
```
Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ awk -F',' 'NR==1 {print $2,$10}' OFS=',' FB_Dataset > trump.txt

Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ awk -F',' '{IGNORECASE=1} {if($5~ /trump/ && $10>=1000) {print $2,$10}}' OFS=',' F
B_Dataset | sort -t',' -k2n >> trump.txt

Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ head -6 trump.txt
post_id,likes_count
5550296508_10154572513421509,1000
86680728811_10154802229533812,1000
18468761129_10153650393881130,1001
5550296508_10154541840616509,1001
5550296508_10154955006956509,1001

Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ tail -5 trump.txt
15704546335_10154646163096336,218748
15704546335_10154108339971336,222119
18468761129_10153524839811130,229187
5550296508_10154298504746509,248012
_22228735667216_10153960167952217222,368179

Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ |
```

Header + first 5 rows



```
post_id,likes_count
5550296508_10154572513421509,1000
86680728811_10154802229533812,1000
18468761129_10153650393881130,1001
5550296508_10154541840616509,1001
5550296508_10154955006956509,1001
```

Last 5 rows



```
15704546335_10154646163096336,218748
15704546335_10154108339971336,222119
18468761129_10153524839811130,229187
5550296508_10154298504746509,248012
_22228735667216_1015396016795221722,368179
```

Explanation, Justification & Answer:

Before adding the filtered data into trump.txt, the trump.txt is first created with by adding the column headers into it via an awk command to get the header only, then inputting this to a trump.txt file created.

Then, using the second awk command as seen above, we select posts where “Trump” (ignorecase) is mentioned in the message and where the likes_count for these posts are >= 1000, and only output the post_id and likes_count column data. The output from this awk command is then piped and thus inputted to a sort command where we specify the field separator as ‘,’. So, this **sorts** the rows by the 2nd column (likes_count). This final sorted by likes_count output is then appended to the trump.txt file created earlier.

Note that the question did NOT state whether the sort is to be ascending or descending, thus I have assumed it to be sorted in *ascending* order.

Then, a head command is used to show the column header & first 5 rows, as well as another head command to show the last 5 entries (in Unix shell). I also added a screenshot above of the trump.txt header + first 5 rows, and last 5 rows, just in case this is needed.

How the awk command is used:

-F is used to specify the field separator/delimiter, in this case is ‘,’.

'NR==1 {print $2,$10}' gets the header and prints its 2nd and 10th column.

OFS=',' specifies the output field separator as ‘,’ (comma)

‘{IGNORECASE=1} …’ IGNORECASE controls case-sensitivity and if IGNORECASE is non-zero (here I put 1), then all operations for string comparisons all ignore case.

‘{if($5~ /trump/ && $10>=1000) {print $2,$10}}' finds the rows where the message column mentions trump and where the likes_count is >= 1000, then prints the 2nd and 10th column.

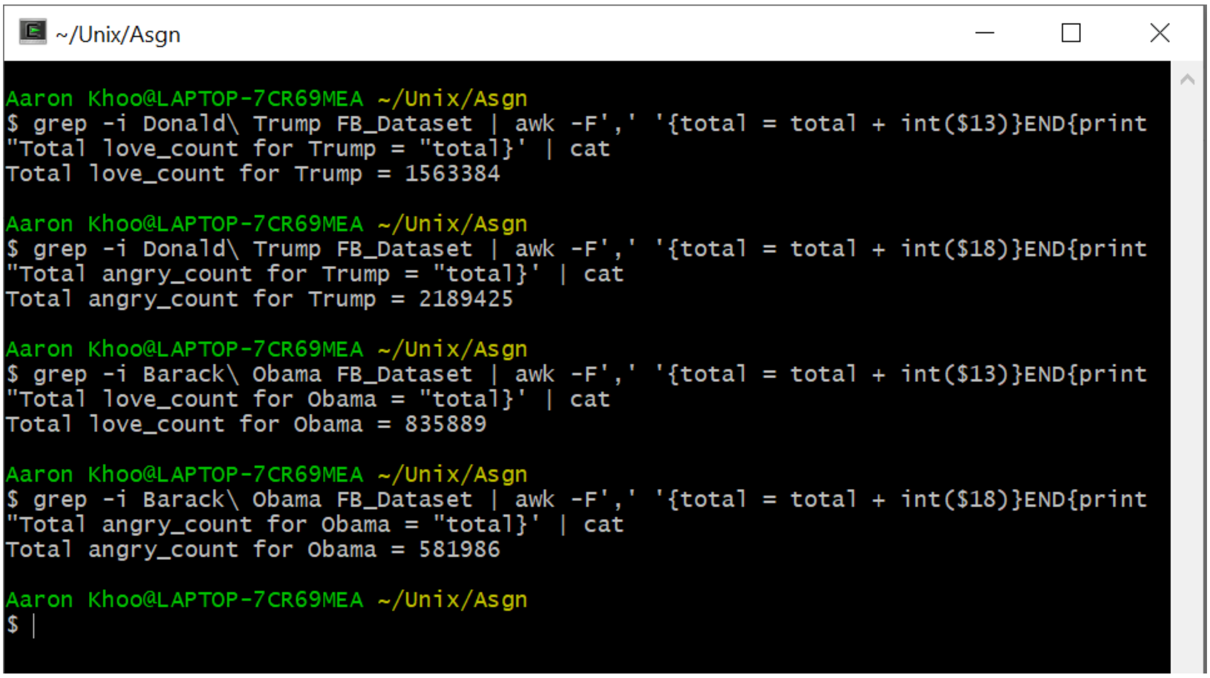
Task A10

Code Input: (Unix)

```
grep -i Donald\ Trump FB_Dataset | awk -F',' '{total = total + int($13)}END{print
"Total love_count for Trump = "total}' | cat


grep -i Donald\ Trump FB_Dataset | awk -F',' '{total = total + int($18)}END{print
"Total angry_count for Trump = "total}' | cat


grep -i Barack\ Obama FB_Dataset | awk -F',' '{total = total + int($13)}END{print
"Total love_count for Obama = "total}' | cat


grep -i Barack\ Obama FB_Dataset | awk -F',' '{total = total + int($18)}END{print
"Total angry_count for Obama = "total}' | cat
```

```
*Note: The multi-line copy pasting thing happens here again; when copy-pasting the
second awk command's code from the pdf into the terminal, might have to copy the
first line first, then enter a 'space', then only copy the second line (eg: "Total
love_count for Trump… ") after. This is something I encountered when trying to
copy-paste multiple lines from pdfs. I think this is because pdfs don't take the
space after the first line "… {print" into account, since copy-pasting the same
code works from the word document.
```

Output:



```
Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ grep -i Donald\ Trump FB_Dataset | awk -F',' '{total = total + int($13)}END{print
"Total love_count for Trump = "total}' | cat
Total love_count for Trump = 1563384

Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ grep -i Donald\ Trump FB_Dataset | awk -F',' '{total = total + int($18)}END{print
"Total angry_count for Trump = "total}' | cat
Total angry_count for Trump = 2189425

Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ grep -i Barack\ Obama FB_Dataset | awk -F',' '{total = total + int($13)}END{print
"Total love_count for Obama = "total}' | cat
Total love_count for Obama = 835889

Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ grep -i Barack\ Obama FB_Dataset | awk -F',' '{total = total + int($18)}END{print
"Total angry_count for Obama = "total}' | cat
Total angry_count for Obama = 581986

Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ |
```

Explanation, Justification & Answer:

From the above output, we can observe that:

Total love_count for Trump = 1563384

Total angry_count for Trump = 2189425

Total love_count for Obama = 835889

Total angry_count for Obama = 581986

Code Explanation

The grep command in each line command used above, is used to get the rows that include the specific pattern with ignorecase (either “Donald Trump” or “Barack Obama” as seen above). This output is then piped and inputted to an awk command that accumulates the values in the specific column (either $13 – love_count, or $18 – angry_count, as seen above) and outputs the result. Then, the cat command is used to print this result on the standard output in the Unix shell.

My approach of my code to find the love and angry counts for each person:

1) In order to compare and find out the overall positive feeling of each person among people, I believe it is better to include the love/angry counts of all posts that include the names while IGNORING the case. This is because mentions of the names via the message, caption, post name, description, or post link, could often either be in lowercase or uppercase. For example, URL links must be in lowercase because a linux server is case sensitive (Goldford, 2011). Therefore, when using the grep command, we use -i to ignore the case.

2) Note that I made sure to only include mentions that have **both** the first name and last name mentioned together. Though this might reduce slightly the number of mentions compared to finding for the first and last name as individual patterns, we should note that finding those that have both first name and last name **helps prevent posts that are about different people who may have similar names, from being counted**. For instance, a post about “Ivanka Trump” or about “McDonalds’” may be counted if we include posts that include only either the individual last name or first name of the person (“Donald Trump”). Or for Obama’s case, a post about “Michelle Obama” may be counted (so we don’t want to count these since it’s not directly related to “Barack Obama”).

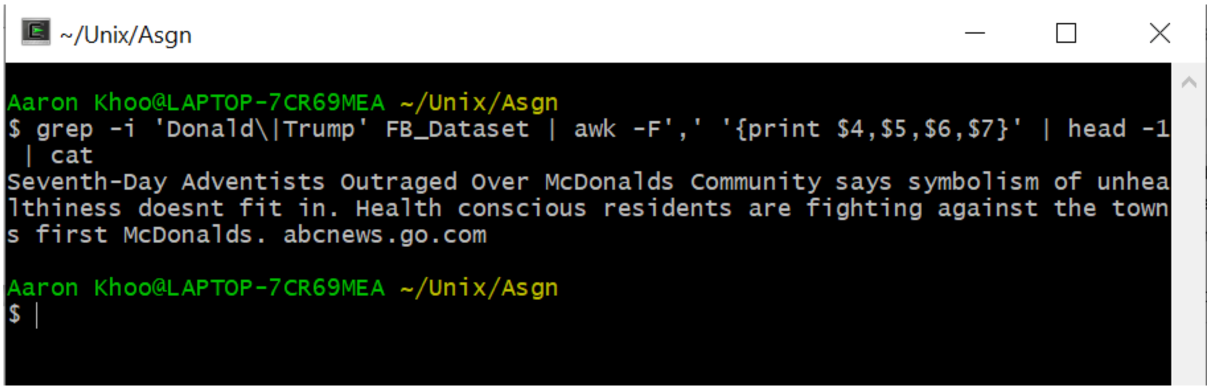
To show an example code & output that includes individual first/last names when searching (which is not what we want as explained right before this):

Example 1 – Donald Trump

Code:

```
grep -i 'Donald\|Trump' FB_Dataset | awk -F',' '{print $4,$5,$6,$7}' | head -1 |
cat
```

Output:



```
Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ grep -i 'Donald\|Trump' FB_Dataset | awk -F',' '{print $4,$5,$6,$7}' | head -1
 | cat
Seventh-Day Adventists Outraged Over McDonalds Community says symbolism of unhea
lthiness doesnt fit in. Health conscious residents are fighting against the town
s first McDonalds. abcnews.go.com

Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ |
```

post_name,message,description,caption from output:

Seventh-Day Adventists Outraged Over McDonalds Community says symbolism of unhealthiness doesnt fit in. Health conscious residents are fighting against the towns first McDonalds.

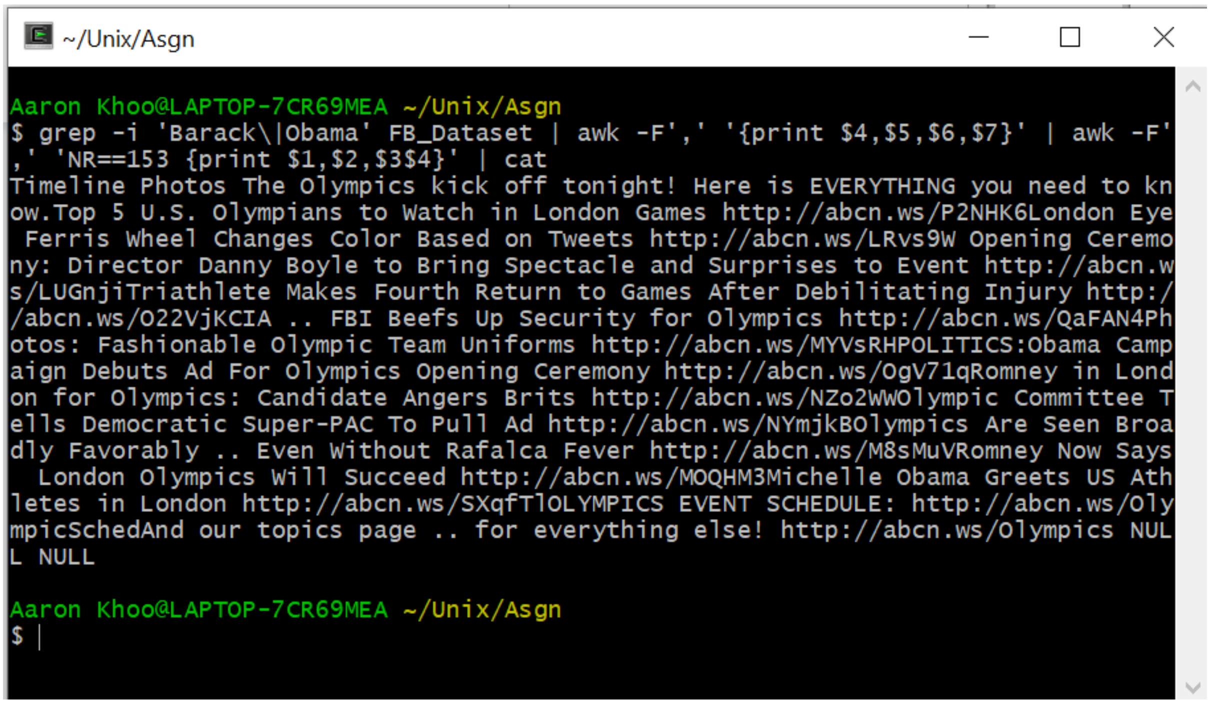
From the above output, we can see that indeed since ‘Donald’ was found in ‘McDonalds’, so this entry was counted as related to Donald Trump when it should not have been.

Example 2 – Barack Obama

Code:

```
grep -i 'Barack\|Obama' FB_Dataset | awk -F',' '{print $4,$5,$6,$7}' | awk -F','
'NR==153 {print $1,$2,$3,$4}' | cat
```

Output:



```
Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ grep -i 'Barack\|Obama' FB_Dataset | awk -F',' '{print $4,$5,$6,$7}' | awk -F'
,' 'NR==153 {print $1,$2,$3$4}' | cat
Timeline Photos The Olympics kick off tonight! Here is EVERYTHING you need to kn
ow.Top 5 U.S. Olympians to Watch in London Games http://abcn.ws/P2NHK6London Eye
 Ferris Wheel Changes Color Based on Tweets http://abcn.ws/LRvs9W Opening Ceremo
ny: Director Danny Boyle to Bring Spectacle and Surprises to Event http://abcn.w
s/LUGnjiTriathlete Makes Fourth Return to Games After Debilitating Injury http:/
/abcn.ws/O22VjKCIA .. FBI Beefs Up Security for Olympics http://abcn.ws/QaFAN4Ph
otos: Fashionable Olympic Team Uniforms http://abcn.ws/MYVsRHPOLITICS:Obama Camp
aign Debuts Ad For Olympics Opening Ceremony http://abcn.ws/OgV71qRomney in Lond
on for Olympics: Candidate Angers Brits http://abcn.ws/NZo2WWOlympic Committee T
ells Democratic Super-PAC To Pull Ad http://abcn.ws/NYmjkBOlympics Are Seen Broa
dly Favorably .. Even Without Rafalca Fever http://abcn.ws/M8sMuVRomney Now Says
  London Olympics Will Succeed http://abcn.ws/MOQHM3Michelle Obama Greets US Ath
letes in London http://abcn.ws/SXqfTlOLYMPICS EVENT SCHEDULE: http://abcn.ws/Oly
mpicSchedAnd our topics page .. for everything else! http://abcn.ws/Olympics NUL
L NULL

Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ |
```

post_name,message,description,caption from output:

Timeline Photos The Olympics kick off tonight! Here is EVERYTHING you need to know.Top 5 U.S. Olympians to Watch in London Games http://abcn.ws/P2NHK6London Eye Ferris Wheel Changes Color Based on Tweets http://abcn.ws/LRvs9W Opening Ceremony: Director Danny Boyle to Bring Spectacle and Surprises to Event http://abcn.ws/LUGnjiTriathlete Makes Fourth Return to Games After Debilitating Injury http://abcn.ws/O22VjKCIA .. FBI Beefs Up Security for Olympics http://abcn.ws/QaFAN4Photos: Fashionable Olympic Team Uniforms http://abcn.ws/MYVsRHPOLITICS:Obama Campaign Debuts Ad For Olympics Opening Ceremony http://abcn.ws/OgV71qRomney in London for Olympics: Candidate Angers Brits http://abcn.ws/NZo2WWOlympic Committee Tells Democratic Super-PAC To Pull Ad http://abcn.ws/NYmjkBOlympics Are Seen Broadly Favorably .. Even Without Rafalca Fever http://abcn.ws/M8sMuVRomney Now Says London Olympics Will Succeed http://abcn.ws/MOQHM3Michelle Obama Greets US Athletes in London http://abcn.ws/SXqfTlOLYMPICS EVENT SCHEDULE: ht

From the above output, we can see that indeed since ‘Obama’ was found in ‘Michelle Obama‘, so this entry was counted as related to Barack Obama when it should not have been.

Therefore, with these examples shown, we have then justified my approach to coding and getting the required posts correctly relating to “Donald Trump “ and “Barack Obama” via searching for the **FULL name in ALL columns.**

Analysis of the overall positive feeling comparison among people between Donald Trump and Barack Obama:

Key observation before analysing the overall positive feeling

One key observation of the counts found is that Barack Obama has a lower count for both love and angry count during this period. This could perhaps be explained by the following:

(i) During Trump’s presidential period that started on June 16, 2015, media analysts had found and reported that many of Trump’s supporters lurked on Facebook, started many support pages, shared news (a lot of them pro-Trump fake news). The number of supporters on Facebook has ultimately led to the steep rise in posts about Donald Trump over his presidency period. (BBCnews, 2016)

(ii) In 2016, Trump’s digital director said the Trump campaign used Facebook to reach clusters of rural voters, by tailoring the background and phrasing of Facebook ads to maximise the impact. (Guardian, 2016)

Thus, the points stated above are what could have caused the increase in the number of trump supporters and thus explains why the engagement of Trump related posts are much higher than Obama’s (number of both love and angry counts for Trump were higher than Obama) during this period. Therefore, this brings us to this conclusion: To analyse the positive feeling, we should then NOT compare the raw difference between the magnitude of the counts of Obama/Trump, since the number of posts vary widely between Trump and Obama due to the nature and environment of US politics topics on Facebook at that time.

So, **let us instead use the ratio of love_counts over angry_counts for each person as a metric for the overall positive feeling**. (Thus, the definition of a ‘good positive feeling among people’ = a high love_counts:angry_counts ratio). So, note that **the higher the ratio, the more positive the feeling among people.**

Ratio of love_counts over angry_counts for Donald Trump = 1563384:2189425 = 0.714:1

Ratio of love_counts over angry_counts for Barack Obama = 835889:581986 = 1.43:1

From the above results, it is observed that Barack Obama has a higher ratio of love_counts over angry_counts as compared to Donald Trump over the time between 1st January 2012 to 7th November 2016.

Therefore, we can indeed conclude that Barack Obama **has a greater positive feeling among people compared to Donald Trump.**

PART B: Graphing Data in R

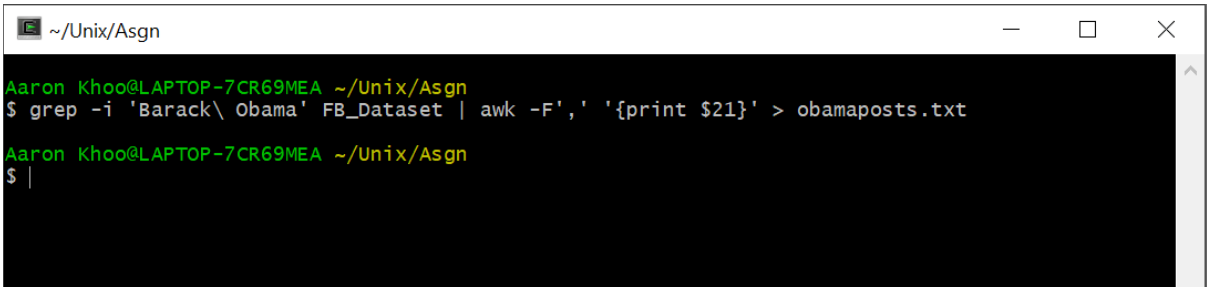
TASK B1

(B1.1) Extracting the timestamps for ALL posts referring to "Barack Obama".

Code Input: (Unix)

```
grep -i 'Barack\ Obama' FB_Dataset | awk -F',' '{print $21}' > obamaposts.txt
```

Output:



```
Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ grep -i 'Barack\ Obama' FB_Dataset | awk -F',' '{print $21}' > obamaposts.txt

Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ |
```

(B1.2) Explanation of Code

Here I have extracted the timestamps (in column $21) for posts relating to "Barack Obama" to a new text file 'obamaposts.txt'

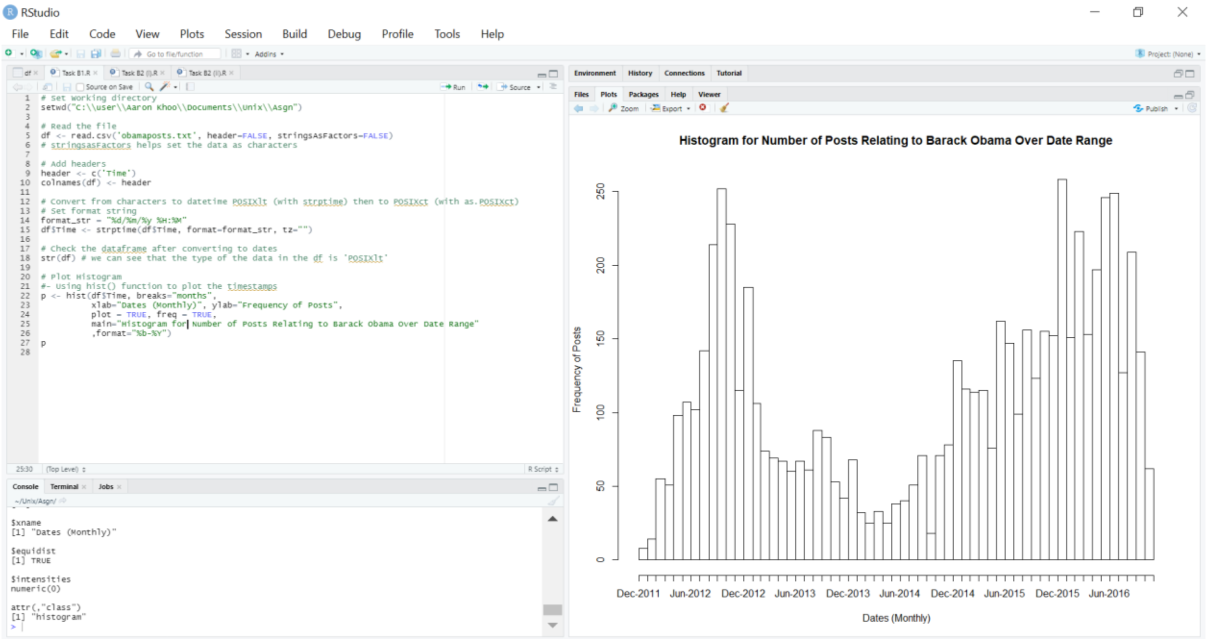
The grep command is used to print/output lines of FB_Dataset that match a specific pattern ('Barack Obama') and in this case we are ignoring the case with -i. This output is then piped and inputted for the next awk command which prints/outputs each row's 21st column (timestamps) from the rows related to "Barack Obama". This output is then added to a newly created file called obamaposts.txt with '> obamaposts.txt'.

(i) Reading the extracted data into R, converting the timestamps and producing a histogram

Code: (R)

```
# Set Working directory
setwd("C:\\user\\Aaron Khoo\\Documents\\Unix\\Asgn")

# Read the file
df <- read.csv('obamaposts.txt', header=FALSE, stringsAsFactors=FALSE)
# stringsasFactors helps set the data as characters

# Add headers
header <- c('Time')
colnames(df) <- header

# Convert from characters to datetime POSIXlt (with strptime) then to POSIXct (with as.POSIXct)
# Set format string
format_str = "%d/%m/%y %H:%M"
df$Time <- strptime(df$Time, format=format_str, tz="")

# Check the dataframe after converting to dates
str(df) # we can see that the type of the data in the df is 'POSIXlt'
```

```
# Plot Histogram
#- Using hist() function to plot the timestamps
p <- hist(df$Time, breaks="months",
          xlab="Dates (Monthly)", ylab="Frequency of Posts",
          plot = TRUE, freq = TRUE,
          main="Histogram for Number of Posts Relating to Barack Obama Over Date Range"
          ,format="%b-%Y")
p
```



```
# Set working directory
setwd("C:\\user\\Aaron Khoo\\Documents\\Unix\\Asgn")

# Read the file
df <- read.csv('obamaposts.txt', header=FALSE, stringsAsFactors=FALSE)
# stringsasFactors helps set the data as characters

# Add headers
header <- c('Time')
colnames(df) <- header

# Convert from characters to datetime POSIXlt (with strptime) then to POSIXct (with as.POSIXct)
# Set format string
format_str = "%d/%m/%y %H:%M"
df$Time <- strptime(df$Time, format=format_str, tz="")

# Check the dataframe after converting to dates
str(df) # we can see that the type of the data in the df is 'POSIXlt'

# Plot Histogram
#- Using hist() function to plot the timestamps
p <- hist(df$Time, breaks="months",
          xlab="Dates (Monthly)", ylab="Frequency of Posts",
          plot = TRUE, freq = TRUE,
          main="Histogram for Number of Posts Relating to Barack Obama Over Date Range"
          ,format="%b-%Y")
p
```

```
$xname
[1] "Dates (Monthly)"

$equidist
[1] TRUE

$intensities
numeric(0)

attr(,"class")
[1] "histogram"
>
```

Explanation & Justification of Code:

I will explain my code in R in steps:

(1) First, I set my working directory to where my extracted timestamps file is at with setwd().

(2) When reading the txt file as a csv into a dataframe, note that with read.csv() the separator would be set as default = ‘,’ which is what we want because our fields in the text file generated had ‘,’ as the separator. Also, I have set stringsasFactors=FALSE which helps set the data as characters instead of as factors.

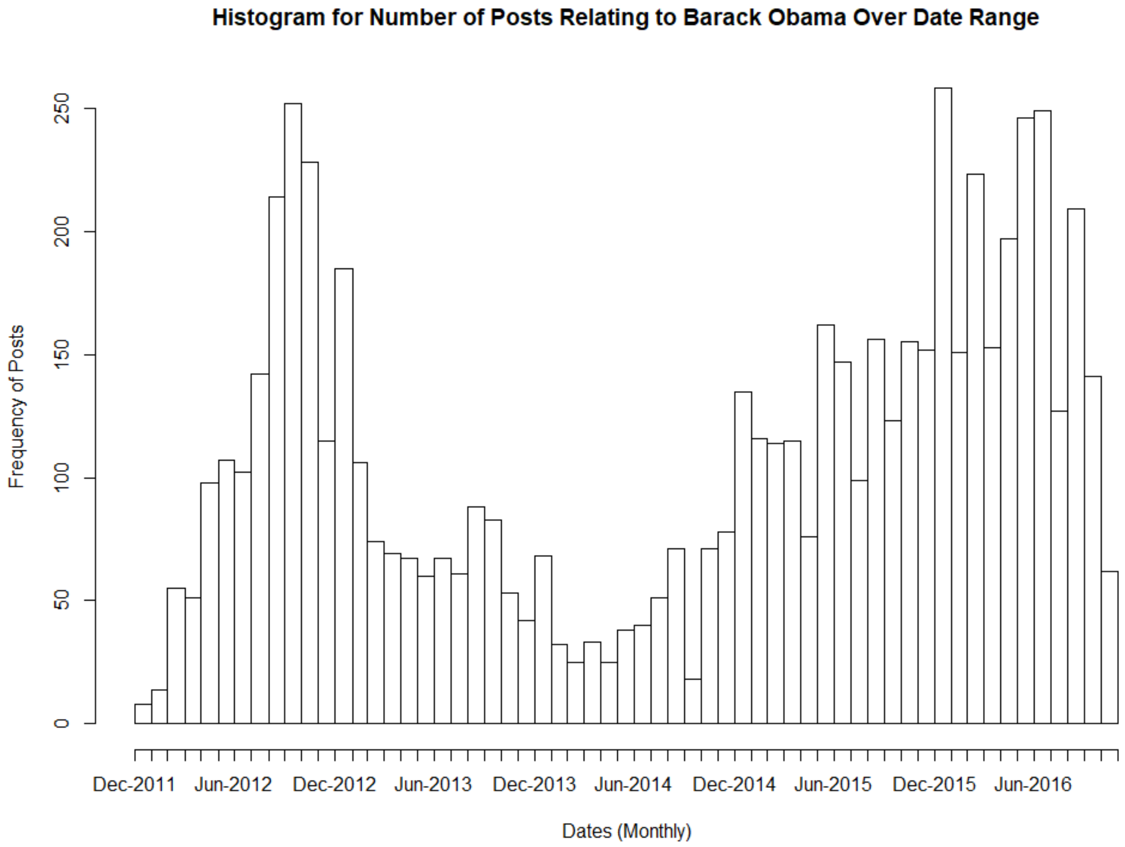
(3) I then add an appropriate header to this dataframe to make the dataframe easier to understand when viewing it and when needing to access the column.

(4) Then, to convert the characters into timestamps so that R will recognise these as timestamps, a format string is first set according to how the time is presented in the original df, then with the use of strptime(), the Time column is converted to a POSXlt format, thus R will be able to recognise each column row as a timestamp.
Note: strptime is a function to directly convert character vectors to **POSIXlt** format.

(5) When plotting the histogram, the hist() function is used. I set the breaks to be by month, thus we can expect 60 bins (from 1/1/2012 to 31/12/2016) in the histogram. I chose it to be in months so that I can see the variation better between them for my analysis later. I also specify the appropriate x/y-labels and plot titles to be on the plot. I set freq=TRUE so that the histogram graphic is a representation of frequencies. In order to see the plot, I set

plot=TRUE, so that a histogram is actually plotted, as otherwise a list of breaks and counts would have been returned. (rdocumentation)

Output Histogram:



(ii) Description/Analysis on histogram pattern & Explanation.

The pattern/shape of the histogram seems to be that:

(1) The shape roughly looks like it has a **bimodal distribution,** which had the first peak in Oct 2012, and then the second peak again in Jan 2016. Note that a **histogram** of a **bimodally distributed** dataset will show two peaks.
Thus, this implies that there would have been two different scenarios or happenings related to Barack Obama during those times (2012 and 2016) that had caused this shape of a graph. With general knowledge, we can see that those two periods for the peaks were on two separate occasions of a US presidential election that Barack Obama was either running for, or was very much involved during the discussion about elections. In 2012, he ran against former governor Mitt Romney. In 2015-2016, he supported Hilary Clinton's campaign, who ran against Republican Donald Trump. (Pengelly, 2015)

(2) From the start of 2012, the number of posts gradually increased and reached the first peak in Oct 2012. In 2012, let us note that there was the 2012 US presidential election period, where Barack Obama was campaigning for re-election. The 2012 US presidential day was held on November 6, 2012, but before that Obama's campaign started from April 4, 2011 till the presidential election day. During this campaign, Obama's team had built a digital data operation that used the data and power of Facebook to target individual voters via an aggressive advertising campaign (Pilkington, Michel, 2012). **Therefore, this explains the**

increase in the number of posts related to Barack Obama during the presidential period in 2012.

(3) However, to explain the peak in October 2012, we can see that this could be because October was the height of the whole presidential campaign due to the presidential debates that were held during October between Obama and Romney, which was then followed by the presidential election day on November 6, 2012. **Therefore, this explains the first peak of the number of posts referring to Barack Obama in Oct 2012.**

(4) For the distribution about the second peak, we see that the number of posts start to rise in 2015, and this can be explained because Hillary Clinton had launched her campaign in June 2015 and Barack Obama showed support and endorsed her during her campaign. Thus, this second peak was then around Jan 2016, and could be mainly due to:

   a. In January 2016, Obama made an actionable speech about gun control and tightening gun control measures in the US, which caused US republicans to attack Obama's moves to tighten gun control. (BBCnews, 2016) This sparked a lot of discussion and debate between the republicans and democrats.

   b. Discussion about the ongoing 2016 presidential election period with topics including Hillary Clinton and Barack Obama.

   Therefore, this explains the second peak of the number of posts referring to Barack Obama in 2016.

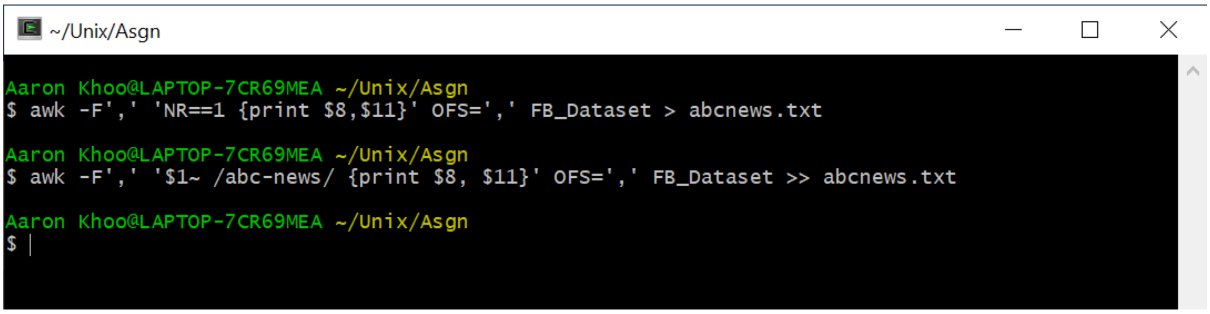
TASK B2

Extracting the comments_count and post_type for abc-news.

Code Input: (Unix)

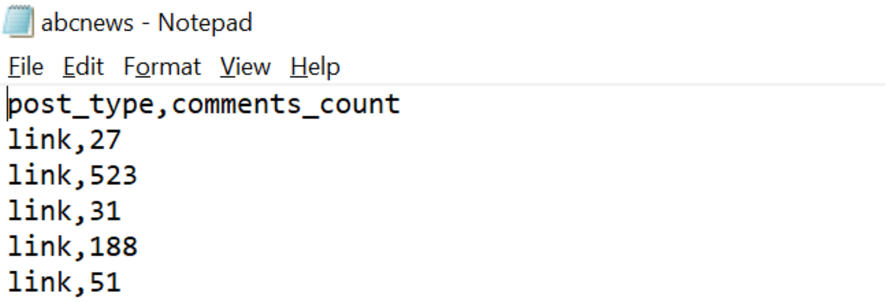
```
awk -F',' 'NR==1 {print $8,$11}' OFS=',' FB_Dataset > abcnews.txt
awk -F',' '$1~ /abc-news/ {print $8, $11}' OFS=',' FB_Dataset >> abcnews.txt
```

Output:



```
Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ awk -F',' 'NR==1 {print $8,$11}' OFS=',' FB_Dataset > abcnews.txt

Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ awk -F',' '$1~ /abc-news/ {print $8, $11}' OFS=',' FB_Dataset >> abcnews.txt

Aaron Khoo@LAPTOP-7CR69MEA ~/Unix/Asgn
$ |
```

Showing the header + first 5 rows of the data extracted.



```
post_type,comments_count
link,27
link,523
link,31
link,188
link,51
```

Explanation & Justification of Code:

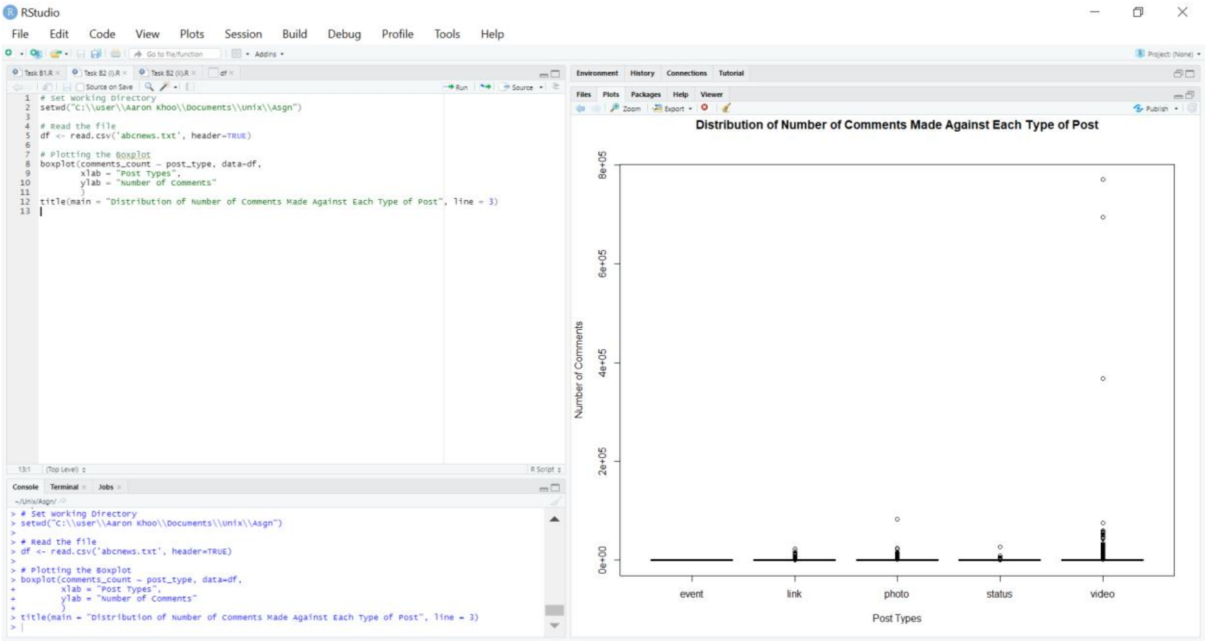
I used an awk command to first get the header of FB_Dataset for the 8th and 11th column (post_type & comments_count respectively), set the field separator for the file as a comma, and inputted this into a newly created file called 'abcnews.txt'.

The next awk command then gets all the rows that **contain abc-news** in the page_name column in FB_Dataset, then outputs the 8th and 11th column (post_type & comments_count respectively), which is then appended to the 'abcnews.txt' file.

(i) Reading the extracted data into R and producing a boxplot for the distribution of comments against each post type.

Code: (R)

```
# Set Working Directory
setwd("C:\\user\\Aaron Khoo\\Documents\\Unix\\Asgn")

# Read the file
df <- read.csv('abcnews.txt', header=TRUE)

# Plotting the Boxplot
boxplot(comments_count ~ post_type, data=df,
        xlab = "Post Types",
        ylab = "Number of Comments"
        )
title(main = "Distribution of Number of Comments Made Against Each Type of Post", line = 3)
```



```
# Set working Directory
setwd("C:\\user\\Aaron Khoo\\Documents\\unix\\Asgn")

# Read the file
df <- read.csv('abcnews.txt', header=TRUE)

# Plotting the Boxplot
boxplot(comments_count ~ post_type, data=df,
        xlab = "Post Types",
        ylab = "Number of Comments"
        )
title(main = "Distribution of Number of Comments Made Against Each Type of Post", line = 3)
```

```
> # Set working Directory
> setwd("C:\\user\\Aaron Khoo\\Documents\\Unix\\Asgn")
>
> # Read the file
> df <- read.csv('abcnews.txt', header=TRUE)
>
> # Plotting the Boxplot
> boxplot(comments_count ~ post_type, data=df,
+         xlab = "Post Types",
+         ylab = "Number of Comments"
+         )
> title(main = "Distribution of Number of Comments Made Against Each Type of Post", line = 3)
> 
```

Explanation of Code

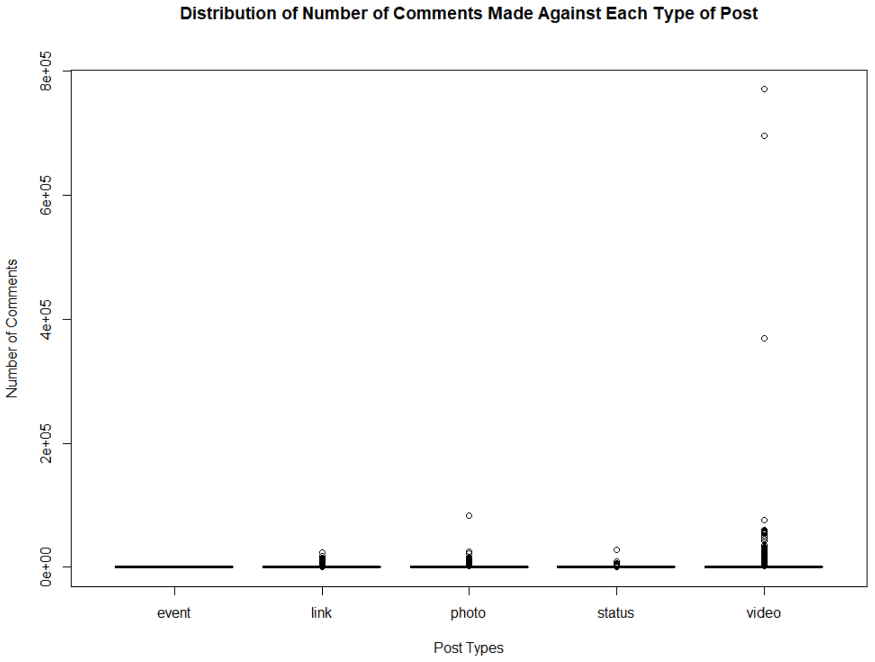
I will explain my code in R in steps:

(1) First, I set my working directory to where my extracted data file is at with setwd().

(2) When reading the txt file as a csv into a dataframe, note that with read.csv() the separator would be set as default = ‘,’ which is what we want because our fields in the text file generated had ‘,’ as the separator. Also, since I have set for the header to be included in the txt file, the header=TRUE helps it to set the first row in the txt as the column names of the df.

(3) Then, the required boxplot is plotted with the boxplot function, by setting the comments_count against the post_type with the ‘~’ formula syntax. This formula syntax helps to automatically group the numeric vector of ‘comments_count’ according to the value of post_type. The x/y-axis labels and plot title are also set appropriately.

Output Boxplot:



(i) Analysis on boxplot:

From the plot, we observe several points & thus infer:

(1) We can see that link, photo, status and video had a quite a few outliers, with video having the highest and most extreme cases of outliers compared to all other post types. This marks a significant number of high outliers marked by the circles/bubbles very much beyond the whiskers of each boxplot.

(2) The outliers have also clearly negatively affected the readability of the boxplot as it is impossible to observe skewness of the distributions.

(3) Since I am also unable to compare the medians of the post_types's comments count, thus to come to a conclusion on which post_type is the most engaging:

   a. I **first define 'most engaging post_type' as: The post_type that has a high amount of posts with a high number of comments.**

   b. I assume that the distribution of each post_type is around the same for now. (due to low readability of current plot)

   c. Thus, from the above definitions, we can conclude and infer that **video is then the most engaging post type** as:

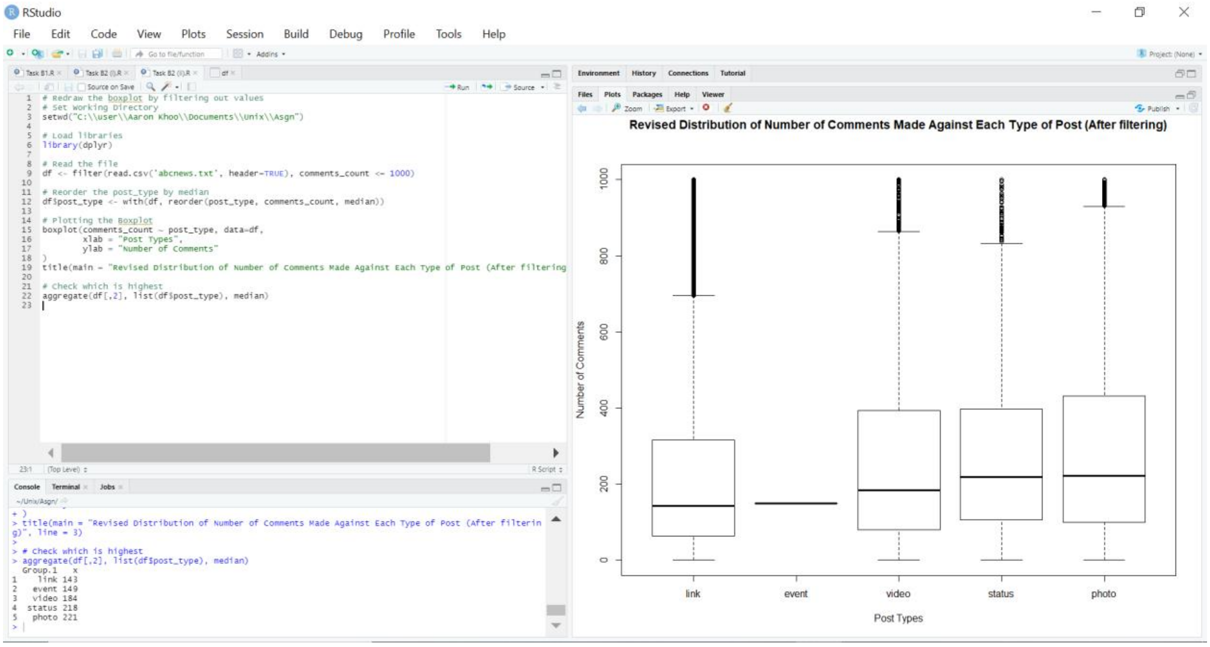
      i. It has a high **number of outliers** compared to the others according to the plot,

      ii. The average **comments count of these outliers is very high** compared to other post_types.

(ii) Redrawing the boxplot with filtered values:

Code: (R)

```
# Redraw the boxplot by filtering out values
# Set Working Directory
setwd("C:\\user\\Aaron Khoo\\Documents\\Unix\\Asgn")

# Load libraries
library(dplyr)

# Read the file
df <- filter(read.csv('abcnews.txt', header=TRUE), comments_count <= 1000)

# Reorder the post_type by median
df$post_type <- with(df, reorder(post_type, comments_count, median))

# Plotting the Boxplot
boxplot(comments_count ~ post_type, data=df,
        xlab = "Post Types",
        ylab = "Number of Comments"
)
title(main = "Revised Distribution of Number of Comments Made Against Each Type of Post (After filtering)", line = 3)

# Check which is highest
aggregate(df[,2], list(df$post_type), median)
```



```
# Redraw the boxplot by filtering out values
# Set working Directory
setwd("C:\\user\\Aaron Khoo\\Documents\\Unix\\Asgn")

# Load libraries
library(dplyr)

# Read the file
df <- filter(read.csv('abcnews.txt', header=TRUE), comments_count <= 1000)

# Reorder the post_type by median
df$post_type <- with(df, reorder(post_type, comments_count, median))

# Plotting the Boxplot
boxplot(comments_count ~ post_type, data=df,
        xlab = "Post Types",
        ylab = "Number of Comments"
)
title(main = "Revised Distribution of Number of Comments Made Against Each Type of Post (After filtering

# Check which is highest
aggregate(df[,2], list(df$post_type), median)
```

```
+ )
> title(main = "Revised Distribution of Number of Comments Made Against Each Type of Post (After filterin
g)", line = 3)
>
> # Check which is highest
> aggregate(df[,2], list(df$post_type), median)
  Group.1   x
1    link 143
2   event 149
3   video 184
4  status 218
5   photo 221
> |
```

Explanation of Code

I will explain my code in R in steps:

(1) First, I set my working directory to where my extracted data file is at with setwd().

(2) The dplyr library is loaded so that we can use its filter method on our read in dataframe later.

(3) When reading the txt file as a csv into a dataframe, note that with read.csv() the separator would be set as default = ',' which is what we want because our fields in the text file generated had ',' as the separator. Also, I have set stringsasFactors=FALSE which helps set the data as characters instead of as factors. Since I have set for the header to be included in the txt file, the header=TRUE helps it to set the first row in the txt as the column names of the df.

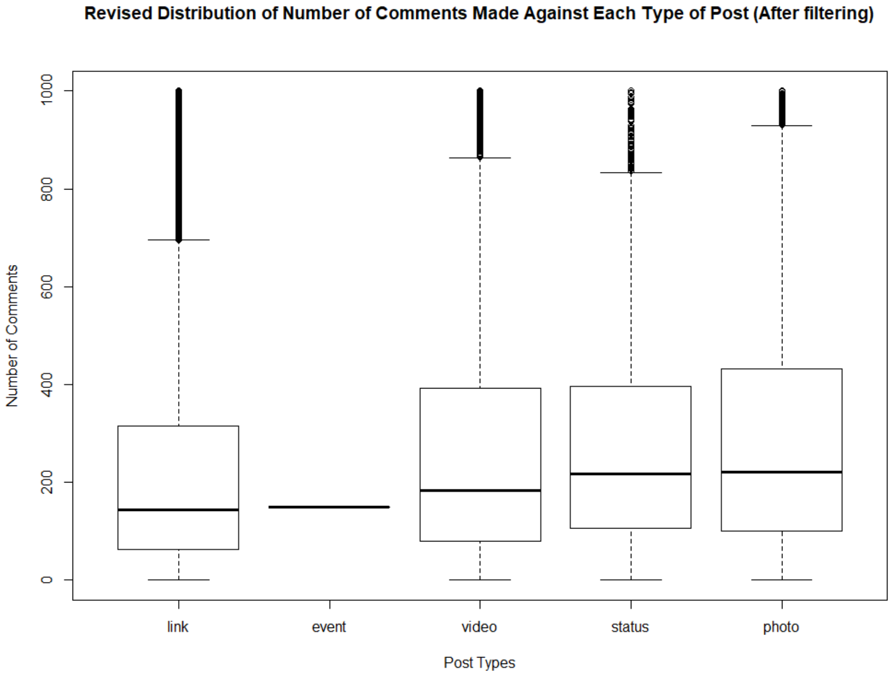
(4) With the filter function, this helps to filter out the rows with comments_count > 1000, and so the remaining rows are put in the dataframe 'df'.

(5) To make it clear which post type's distribution has a higher median (to be discussed further in part (iii)), I reorder the data in the post_type column according to each post_type's median.

(6) The boxplot is then plotted with the boxplot function, by setting the comments_count against the post_type with the '~' formula syntax. This formula syntax helps to automatically group the numeric vector of 'comments_count' according to the value of post_type. The x/y-axis labels and plot title are also set appropriately. Note that the boxplot becomes more readable as compared to the previous boxplot due to the filtering of outliers.

(7) Finally, to reconfirm who has the highest median count (to be discussed further in part (iii)), I have used the aggregate function to aggregate the post_type column and with the numeric comments_count, in order to find the median comments_count of each post_type.

Output Boxplot:



(iii) Analysis on effectiveness of post types

This new boxplot is much more readable and a better plot for us to analyse and work with.

From this boxplot, we can observe and thus infer that:

(1) Photo has the highest median number of comments made as represented by the middle solid line across the boxplots. From my ‘aggregate’ code block’s output in R (refer to section (ii)’s code & output), indeed we can confirm the observation of medians from the boxplot as photo has the highest median value of 221.
Therefore, **photo is on average the most effective post_type among all the other post_types for abc-news.**

Other observations:

(1) ‘Event’ post_type has on average a very low effectiveness; and we should note that not many posts use this post_type as there seems to be only one post with event post_type from abc-news, according to the dataframe and the boxplot as shown. The lack of usage of this post_type already illustrates its low effectiveness, however, there is still a possibility that this post_type may prove to be more effective via future data analysis, if it is used more often by abc-news.
(2) Link is the least effective post_type on average as it has the lowest median.
(2) The distribution of the number of comments made is right skewed. This is because we can observe that the medians for each of the boxplots are closer to the bottom of the plot and are therefore lower than their means, which shows that the distribution of the number of comments made is indeed right skewed. (with event post_type as an exception since we only have one event post)

Conclusion

Therefore, we have reached the end of all the tasks in the assignment.

This assignment has allowed me to explore many BASH shell commands and code for working with large datafiles. I believe the learning and practice while doing this assignment has helped me be more familiar with the different commands used for different purposes, as well as how we can combine commands for text manipulation.

Thus, as a recap, by the end of the assignment we indeed have:

1. Navigated within the BASH Shell
2. Processed a large file using the BASH Shell and using online resources/”man” to aid in our writing of commands
3. Justified the definitions and assumptions while answering the questions
4. Outputted processed files into CSV formats using the bash shell.
5. Read a processed file in R and conducted visualisation with R
6. Analysed the visualisations and provided further insight and analyses on them

Therefore, this assignment has helped me be more well-versed with BASH Scripting as well as writing R Scripts. I also especially find it interesting to analyse a large dataset based on Social Media like the one given for this assignment. Thus, I hope to be able use what I learnt from this assignment and unit with regards to Bash Scripting to try and manipulate other large files for other projects in my own time.

References

Beckett, L. (2017, October 9). Trump digital director says Facebook helped win the White House. Retrieved from https://www.theguardian.com/technology/2017/oct/08/trump-digital-director-brad-parscale-facebook-advertising

BBC News. (2016, November 15). US Election 2016: Trump's 'hidden' Facebook army. Retrieved from https://www.bbc.com/news/blogs-trending-37945486

BBC News. (2015, January 6). US Republicans attack Obama gun control moves. Retrieved from https://www.bbc.com/news/world-us-canada-35239504

Goldford, J. (2011, October 17). Never Use Capital Letters in URLs. Retrieved from https://wiredimpact.com/blog/never-use-capital-letters-urls/

Michel, A., Pilkington, E. (2012, February 17). Obama, Facebook and the power of friendship: the 2012 data election. Retrieved from https://www.theguardian.com/world/2012/feb/17/obama-digital-data-machine-facebook-election

Oxford Learner's Dictionaries. (n.d.). oxfordlearnersdictionaries: popular. Retrieved from https://www.oxfordlearnersdictionaries.com/definition/english/popular

Pengelly, M. (2015, April 12). Barack Obama says Hillary Clinton 'would make an excellent president'. Retrieved from https://www.theguardian.com/us-news/2015/apr/11/barack-obama-hillary-clinton-excellent-president

phoenixmap. (2020, May 5). How to Grep for Multiple Strings, Patterns or Words. Retrieved from https://phoenixnap.com/kb/grep-multiple-strings

rdocumentation. (n.d.). hist: Histograms. Retrieved from https://www.rdocumentation.org/packages/graphics/versions/3.6.2/topics/hist

rdocumentation. (n.d.). strptime: Date-time Conversion Functions to and from Character. Retrieved from https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/strptime

timeanddate. (n.d.). Days Calculator: Days Between Two Dates**.** Retrieved from https://www.timeanddate.com/date/durationresult.html?d1=01&m1=01&y1=2012&d2=07&m2=11&y2=2016&h1=00&i1=30&s1=00&h2=23&i2=45&s2=00

uni.stackexchange. (2013, October 15). Sum the values in a column except the header. Retrieved from https://unix.stackexchange.com/questions/96031/sum-the-values-in-a-column-except-the-header