

# Task A: Investigating Facebook Data using shell commands

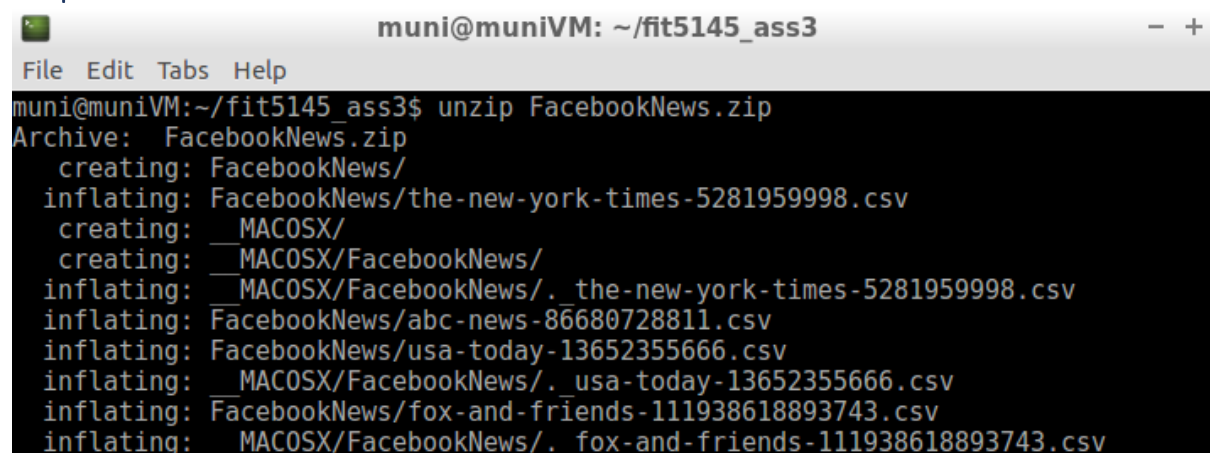
## Q.1) Decompress the file. How big is it (bytes)?

Command: `unzip FacebookNews.zip`

Explanations:

**unzip**: command used to extract from a ZIP file.

Output:

A terminal window titled 'muni@muniVM: ~/fit5145\_ass3' with a menu bar (File, Edit, Tabs, Help). The terminal shows the command 'unzip FacebookNews.zip' and its output: 'Archive: FacebookNews.zip', 'creating: FacebookNews/', 'inflating: FacebookNews/the-new-york-times-5281959998.csv', 'creating: \_\_MACOSX/', 'creating: \_\_MACOSX/FacebookNews/', 'inflating: \_\_MACOSX/FacebookNews/. the-new-york-times-5281959998.csv', 'inflating: FacebookNews/abc-news-86680728811.csv', 'inflating: FacebookNews/usa-today-13652355666.csv', 'inflating: \_\_MACOSX/FacebookNews/. usa-today-13652355666.csv', 'inflating: FacebookNews/fox-and-friends-111938618893743.csv', and 'inflating: \_\_MACOSX/FacebookNews/. fox-and-friends-111938618893743.csv'.

After decompressing “FacebookNews.zip”, we get 2 directories having the csv files,

“FacebookNews”

“\_\_MACOSX/ FacebookNews”

## Finding size of “FacebookNews” directory:

To get the size of “**FacebookNews**” directory, we change current working directory to “FacebookNews”.

Command:

`cd FacebookNews`

`ls -l`

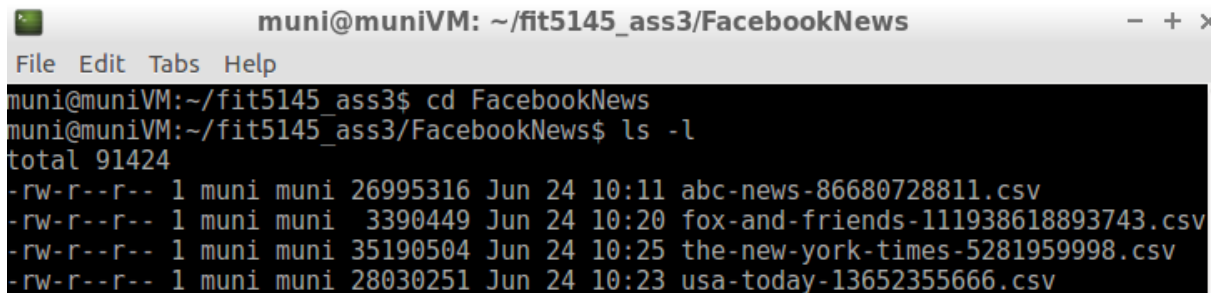
Explanations:

**cd**: command used to change the shell working directory

**ls:** command used to list the contents of directory.

**-l** option to use Long listing format.

Output:



```
muni@muniVM: ~/fit5145_ass3/FacebookNews
File Edit Tabs Help
muni@muniVM:~/fit5145_ass3$ cd FacebookNews
muni@muniVM:~/fit5145_ass3/FacebookNews$ ls -l
total 91424
-rw-r--r-- 1 muni muni 26995316 Jun 24 10:11 abc-news-86680728811.csv
-rw-r--r-- 1 muni muni 3390449 Jun 24 10:20 fox-and-friends-111938618893743.csv
-rw-r--r-- 1 muni muni 35190504 Jun 24 10:25 the-new-york-times-5281959998.csv
-rw-r--r-- 1 muni muni 28030251 Jun 24 10:23 usa-today-13652355666.csv
```

Total size of “**FacebookNews**” directory is 91424 bytes.

### Finding size of “ \_\_MACOSX/FacebookNews” directory:

To get the size of “\_\_**MACOSX/ FacebookNews**” directory, we change current working directory to “\_\_**MACOSX/ FacebookNews**”.

Command:

`cd __MACOSX/FacebookNews`

`ls -la`

Explanations:

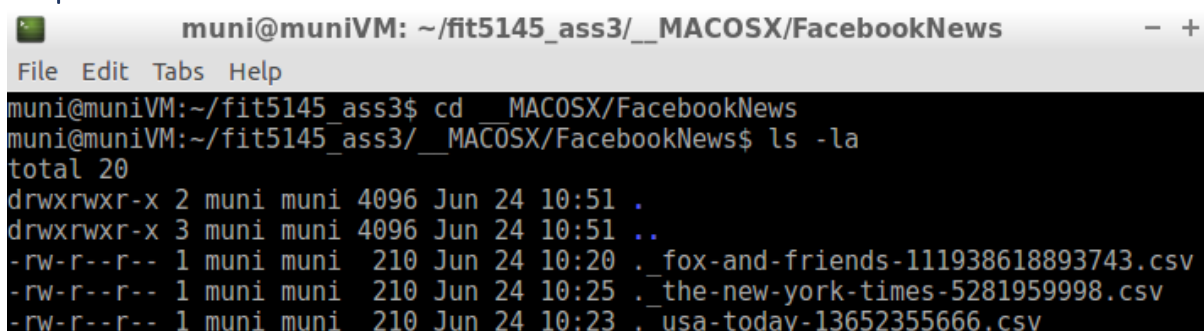
**cd:** command used to change the shell working directory

**ls:** command used to list the contents of directory.

**-l** option to use Long listing format.

**-a** option used to not ignore entries starting with .

Output:



```
muni@muniVM: ~/fit5145_ass3/__MACOSX/FacebookNews
File Edit Tabs Help
muni@muniVM:~/fit5145_ass3$ cd __MACOSX/FacebookNews
muni@muniVM:~/fit5145_ass3/__MACOSX/FacebookNews$ ls -la
total 20
drwxrwxr-x 2 muni muni 4096 Jun 24 10:51 .
drwxrwxr-x 3 muni muni 4096 Jun 24 10:51 ..
-rw-r--r-- 1 muni muni 210 Jun 24 10:20 ._fox-and-friends-111938618893743.csv
-rw-r--r-- 1 muni muni 210 Jun 24 10:25 ._the-new-york-times-5281959998.csv
-rw-r--r-- 1 muni muni 210 Jun 24 10:23 ._usa-today-13652355666.csv
```

Total size of “\_\_**MACOSX/ FacebookNews**” directory is 210 + 210 + 210 = 630 Bytes.

## Computing the total size of the decompressed files:

Total size of decompressed files is the sum of size of the two directories i.e. “**FacebookNews**” and the “**\_\_MACOSX/ FacebookNews**” directory.

**Total size = 91424 + 630 = 92054 bytes.**

## Q.2) What delimiter is used to separate the columns in the file and how many columns are there?

### Command:

```
cat abc-news-86680728811.csv | head | awk -F',' '{print $1,$2,$3}'
```

### Explanations:

**cat:** command used to concatenate files and print on the standard output.

**head:** prints the first 10 lines of file.

**awk:** command used for pattern scanning and text processing.

-F option specifies the delimiter.

### Output:

```
muni@muniVM:~/fit5145_ass3/FacebookNews$ cat abc-news-86680728811.csv | head | awk -F',' '{print $1,$2,$3}'
"id" "page_id" "name"
"86680728811_272953252761568" "86680728811" "Chief Justice Roberts Responds to Judicial Ethics Critics"
"86680728811_273859942672742" "86680728811" "With Reservations"
"86680728811_10150499874478812" "86680728811" "Wishes For 2012 to Fall on Times Square"
"86680728811_244555465618151" "86680728811" "Mitt Romney Vows to Veto Dream Act if President"
"86680728811_252342804833247" "86680728811" "NY Pharmacy Shootout Leaves Suspect"
"86680728811_200661383359612" "86680728811" "The World Rings in 2012"
"86680728811_281125741936891" "86680728811" "Breaking: Magnitude 7.0 Quake Hits Japan"
"86680728811_10150500662053812" "86680728811" "PC"
"86680728811_10150500969563812" "86680728811" "Coast Guard Passenger-Limit Rule Reflects Americans' Weight Gain"
```

As we can get the data of each column using comma (,) as the delimiter using the “awk” command with -F option, we can conclude that the delimiter for the file is comma.

## How many columns are there?

### Command:

```
cat *.csv | head -n 1 | awk -F',' '{print NF}'
```

### Explanations:

**cat:** command used to concatenate files and print on the standard output.

**head:** command to print the first few lines of file.

-n option specifies the first NUM lines to print instead of the first 10.

**awk:** command used for pattern scanning and text processing.

-F option specifies the delimiter.

Output:

```
muni@muniVM:~/fit5145_ass3/FacebookNews$ cat *.csv | head -n 1 | awk -F',' '{print NF}'  
20
```

Total number of Columns is **20**

Q.3) The first column is the unique identifier for each article. What are the other columns?

Command:

```
cat *.csv | head -n 1 | awk -F',' '{print  
$2,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12,$13,$14,$15,$16,$17,$18,$19,$20}'
```

Explanations:

**cat:** command used to concatenate files and print on the standard output.

**head:** command to print the first few lines of file.

-n option specifies the first NUM lines to print instead of the first 10.

**awk:** command used for pattern scanning and text processing.

-F option specifies the delimiter.

Output:

```
muni@muniVM:~/fit5145_ass3/FacebookNews$ cat *.csv | head -n 1 | awk -F',' '{print $2,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12,  
$13,$14,$15,$16,$17,$18,$19,$20}'  
"page_id" "name" "message" "description" "caption" "post_type" "status_type" "likes_count" "comments_count" "shares_co  
t" "love_count" "wow_count" "haha_count" "sad_count" "thankful_count" "angry_count" "link" "picture" "posted at"
```

The other columns are page\_id, 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, link, picture, posted\_at

Q.4) How many articles are there in the file?

Command:

```
cat *.csv | awk -F',' '{l=NR}l>max{max=NR;}END{print max}'
```

Explanations:

**cat:** command used to concatenate files and print on the standard output.

**awk**: command used for pattern scanning and text processing.

-F option specifies the delimiter.

NR is the Number of Records Variable. In our command it gives the row number.

**Output:**

```
muni@muniVM:~/fit5145_ass3/FacebookNews$ cat *.csv | awk -F',' '{l=NR}l>max{max=NR;}END{print max}'  
137650
```

There are total **137650** articles in all the files.

[Q.5\) What is the date range for the articles in this file? \(Assume that the data is in order\)](#)

**Command:**

```
cat *.csv | awk -F',' 'NR==2 {print $20}'
```

**Explanations:**

**cat**: command used to concatenate files and print on the standard output.

**head**: command to print the first few lines of file.

-n option specifies the first NUM lines to print instead of the first 10.

**awk**: command used for pattern scanning and text processing.

-F option specifies the delimiter.

NR is the Number of Records Variable. In our command it gives the row number.

**Output:**

```
muni@muniVM:~/fit5145_ass3/FacebookNews$ cat *.csv | awk -F',' 'NR==2 {print $20}'  
"2012-01-01 00:30:26"
```

Start date range of Articles: 2012-01-01

**Command:**

```
cat *.csv | awk -F',' '{l=NR}l>max{max=NR;}END{print $20}'
```

**Explanations:**

**cat**: command used to concatenate files and print on the standard output.

**awk**: command used for pattern scanning and text processing.

-F option specifies the delimiter.

NR is the Number of Records Variable. In our command it gives the row number.

**Output:**

```
muni@muniVM:~/fit5145_ass3/FacebookNews$ cat *.csv | awk -F',' '{l=NR;l>max{max=NR;}END{print $20}'  
"2016-11-07 23:45:00"
```

End Date range of Articles: 2016-11-07

Therefore, the date range of articles in the files is **2012-01-01** to **2016-11-07**.

### Q.6) How many unique titles are there?

**Command:**

```
awk -F',' '{print $3}' *.csv | sort -u | wc -l
```

**Explanations:**

**awk:** command used for pattern scanning and text processing.

-F option specifies the delimiter.

**sort:** command to sort, merge, or compares all the lines from the given files, or the standard input if no files are given.

-u option used to get first line of an equal run.

**wc:** command used to print newline, word, and byte counts for each file.

-l option used to print the newline counts.

**Output:**

```
muni@muniVM:~/fit5145_ass3/FacebookNews$ awk -F',' '{print $3}' *.csv | sort -u | wc -l  
109248
```

There are **109248** unique titles.

### 7) How many articles don't have a title?

**Command:**

```
cat *.csv | awk -F',' '$3 == NULL {print $1}' | wc -l
```

**Explanations:**

**cat:** command used to concatenate files and print on the standard output.

**awk:** command used for pattern scanning and text processing.

-F option specifies the delimiter.

**wc:** command used to print newline, word, and byte counts for each file.

**-l** option used to print the newline counts.

Output:

```
muni@muniVM:~/fit5145_ass3/FacebookNews$ cat *.csv | awk -F ',' '{ $3 == NULL {print $1} } | wc -l  
2267
```

There are **2267** articles without Title.

8) When was the first mention in the files regarding “Italian food” and what was the title of the post?

Command:

```
grep ".*Italian.*Food.*" *.csv -m1 | awk -F',' '{print $3,$20}'
```

Explanations:

**grep:** command to match a pattern

**-m** option used to stop reading file after NUM matching lines.

**awk -F:** command used for pattern scanning and text processing.

**-F** option specifies the delimiter.

Output:

```
muni@muniVM: ~/fit5145_ass3/FacebookNews  
File Edit Tabs Help  
muni@muniVM:~/fit5145_ass3/FacebookNews$ grep ".*Italian.*Food.*" *.csv -m1 | awk -F',' '{print $3,$20}'  
Marcella Hazan, 1924-2013: Changed the Way Americans Cook Italian Food 2013-09-30 16:25:09"
```

The first post about “Italian Food” was posted on 30-09-2013 having title **Marcella Hazan, 1924-2013: Changed the Way Americans Cook Italian Food.**

9) How many times is “Hillary Clinton” mentioned in the articles? How did you find this? (Do not ignore the case)

Command

```
grep -o "Hillary Clinton" *.csv | wc -l
```

Explanations:

**grep:** command to match a pattern

**-o** option used to print only the matched (non-empty) parts of a matching line, with each such part on a separate output line.

**wc:** command used to print newline, word, and byte counts for each file.

**-l** option used to print the newline counts.

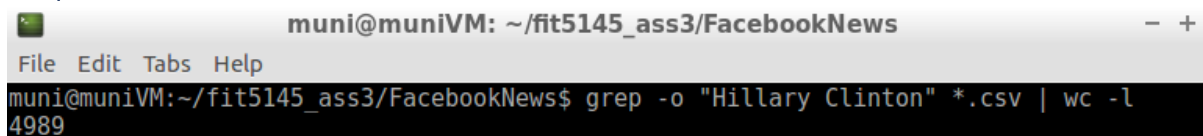
### How did I find this?

I used the **grep** command to match pattern "Hillary Clinton" in all the 4 .csv files.

Then used to **-o** option of **grep** command to print each occurrence of pattern match i.e. "Hillary Clinton" on a new line.

This result is then piped to **wc** command to count the new lines using the **-l** option, thus giving the number of occurrences of "Hillary Clinton" in the given files.

### Output:



```
muni@muniVM: ~/fit5145_ass3/FacebookNews
File Edit Tabs Help
muni@muniVM:~/fit5145_ass3/FacebookNews$ grep -o "Hillary Clinton" *.csv | wc -l
4989
```

"**Hillary Clinton**" is mentioned **4989** times in the articles.

### 10)What about "Donald Trump"? Who is the focus on more articles, Clinton or Trump? (Do not ignore the case)

#### Command:

```
grep -o "Donald Trump" *.csv | wc -l
```

#### Explanations:

**grep:** command to match a pattern.

**-o** option used to print only the matched (non-empty) parts of a matching line, with each such part on a separate output line.

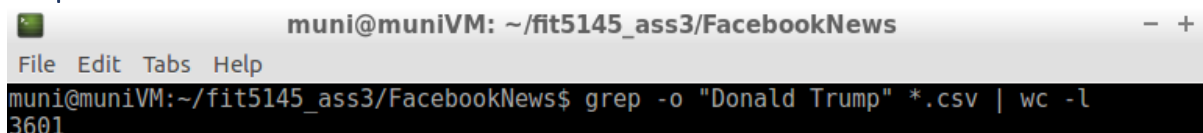
**wc:** command used to print newline, word, and byte counts for each file.

**-l** option used to print the newline counts.

### Who is the focus on more articles?

Looking at the count of occurrence of both the key words i.e. "Hillary Clinton" and "Donald Trump", we can conclude that "**Hillary Clinton**" is has more focus on most of the articles with occurrence of **4989** and "Donald Trump" has 3601 occurances.

### Output:



```
muni@muniVM: ~/fit5145_ass3/FacebookNews
File Edit Tabs Help
muni@muniVM:~/fit5145_ass3/FacebookNews$ grep -o "Donald Trump" *.csv | wc -l
3601
```



“Donald Trump” is mentioned **3601** times in the articles.

11) Select the posts where “Trump” (Ignore the case) is mentioned in the post content and number of likes for those posts are greater than 100. Generate a new file with post\_id and the sorted like\_count and name it “trump.txt”.

Command:

```
cat *.csv | awk -F"," ' $9 > 100 && $4 ~ /\. *Trump\. */ { print $1,$9 } ' > trump.csv  
awk '{ print $0 }' trump.csv | sort -nk2 | cut -f2 -d"\" > trump.txt  
{ echo post_id like_count; cat trump.txt; } > temp.txt && mv temp.txt trump.txt
```

Explanations:

**cat:** command used to concatenate files and print on the standard output.

**awk:** command used for pattern scanning and text processing.

-F option specifies the delimiter.

**sort:** command to sort, merge, or compares all the lines from the given files, or the standard input if no files are given.

-n option used to compare according to string numerical value

-k option used to sort via a key; column 2 in our command is the key.

**cut:** remove sections from each line of files.

-d option used to specify the delimiter.

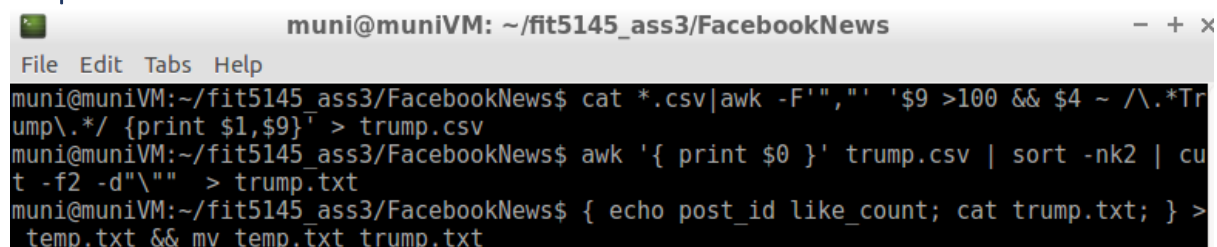
-f option used to select only specified fields. 2nd field is selected in our case.

**echo:** Write arguments to the standard output.

In our case we write the lines to text file.

**mv:** command to move from SOURCE(s) to DIRECTORY.

Output:

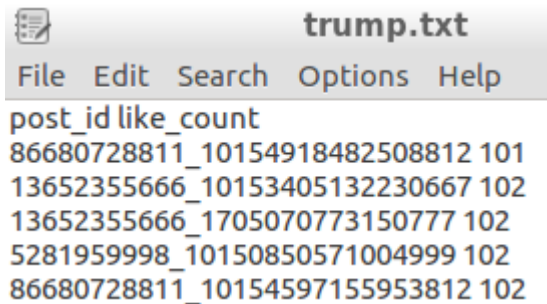


```
muni@muniVM: ~/fit5145_ass3/FacebookNews  
File Edit Tabs Help  
muni@muniVM:~/fit5145_ass3/FacebookNews$ cat *.csv | awk -F"," ' $9 > 100 && $4 ~ /\. *Trump\. */ { print $1,$9 } ' > trump.csv  
muni@muniVM:~/fit5145_ass3/FacebookNews$ awk '{ print $0 }' trump.csv | sort -nk2 | cut -f2 -d"\" > trump.txt  
muni@muniVM:~/fit5145_ass3/FacebookNews$ { echo post_id like_count; cat trump.txt; } > temp.txt && mv temp.txt trump.txt
```

trump.txt

```
post_id like_count  
86680728811_10154918482508812 101
```

```
13652355666_10153405132230667 102
13652355666_1705070773150777 102
5281959998_10150850571004999 102
86680728811_10154597155953812 102
```



Q.12) Find the total number of love\_count and angry\_count for “Donald Trump” and “Hillary Clinton” separately. Who has more positive feeling among people?

Justify your answer

love\_count – Donald Trump

Command:

```
grep "Donald Trump" *.csv | awk -F',' '{sum += $12} END{print sum}'
```

Explanations:

**grep**: command to match a pattern.

**awk**: command used for pattern scanning and text processing.

-F option specifies the delimiter.

Output:

```
muni@muniVM:~/fit5145_ass3/FacebookNews$ grep "Donald Trump" *.csv | awk -F',' '{sum += $12} END{print sum}'
347101
```

Love count for Donald Trump is **347101**

angry\_count - Donald Trump

Command:

```
grep "Donald Trump" *.csv | awk -F',' '{sum += $17} END{print sum}'
```

Explanations:

**grep**: command to match a pattern.

**awk**: command used for pattern scanning and text processing.

-F option specifies the delimiter.

### Output:

```
muni@muniVM:~/fit5145_ass3/FacebookNews$ grep "Donald Trump" *.csv | awk -F',' '{sum += $17} END{print sum}'  
838613
```

Angry count for Donald Trump is **838613**.

### love\_count – Hillary Clinton

#### Command:

```
grep "Hillary Clinton" *.csv | awk -F',' '{sum += $12} END{print sum}'
```

#### Explanations:

**grep**: command to match a pattern.

**awk**: command used for pattern scanning and text processing.

-F option specifies the delimiter.

### Output:

```
muni@muniVM:~/fit5145_ass3/FacebookNews$ grep "Hillary Clinton" *.csv | awk -F',' '{sum += $12} END{print sum}'  
640606
```

Love count for Hillary Clinton is **640606**

### angry\_count - Hillary Clinton

#### Command:

```
grep "Hillary Clinton" *.csv | awk -F',' '{sum += $17} END{print sum}'
```

#### Explanations:

**grep**: command to match a pattern.

**Awk**: command used for pattern scanning and text processing.

-F option specifies the delimiter.

### Output:

```
muni@muniVM:~/fit5145_ass3/FacebookNews$ grep "Hillary Clinton" *.csv | awk -F',' '{sum += $17} END{print sum}'  
799773
```

Angry count for Hillary Clinton is **799773**.

### Who has more positive feeling among people? Justify your answer

“**Hillary Clinton**” has more **love counts** for the articles with total of **640606**, while “Donald Trump” has total love count of only 347101.

On the other hand, “**Donald Trump**” has more **angry counts** for the articles with total of **838613**, while “Hillary Clinton” has total angry count of 799773

Thus, we can say that “**Hillary Clinton**” has more **positive** feeling among people.

### 13)How many articles discussed Trump and Putin? How many discussed Trump but not Clinton?

#### Command: Trump and Putin

```
Cat *.csv | awk '/Trump/ && /Putin/ ' | wc -l
```

#### Explanations:

**cat:** command used to concatenate files and print on the standard output.

**awk:** command used for pattern scanning and text processing.

**wc:** command used to print newline, word, and byte counts for each file.

-l option used to print the newline counts.

#### Output:

```
muni@muniVM:~/fit5145_ass3/FacebookNews$ cat *.csv | awk '/Trump/ && /Putin/ ' | wc -l  
61
```

**61** articles discussed Trump and Putin.

#### Command: Trump but not Clinton

```
grep -i "Trump" *.csv | grep -i -v "Clinton" | wc -l
```

#### Explanations:

**grep:** command to match a pattern.

-i option used to ignore case

-v option used to invert the sense of matching, to select non-matching lines.

**wc:** command used to print newline, word, and byte counts for each file.

-l option used to print the newline counts.

#### Output

```
muni@muniVM:~/fit5145_ass3/FacebookNews$ grep -i "Trump" *.csv | grep -i -v "Clinton" | wc -l  
5533
```

**5533** articles discussed Trump but not Clinton

14) For each publication in trump.txt, find out which month had the most articles about Trump. Try to do this without using grep.

Command: abc-news-86680728811.csv

```
sed -n '/[Tt][Rr][Uu][Mm][Pp]/p' abc-news-86680728811.csv | awk -F"," '{ $9 > 100 {print $20} }' | sed 's/.$// ' | sort -u | cut -c 1-7 | sort | uniq -c | sort -nr | head -n 1
```

#### Explanations:

**sed:** command is a stream editor, used to perform basic text transformations on an input stream.

-n option used to suppress automatic printing of pattern space.

We use a Regular Expression to match the text.

**awk:** command used for pattern scanning and text processing.

-F option specifies the delimiter.

**sort:** command to sort, merge, or compares all the lines from the given files, or the standard input if no files are given.

-u option used to get first line of an equal run.

-n option used to compare according to string numerical value.

**cut:** Print selected parts of lines from each FILE to standard output.

-c option used to select only these characters. 1-7 characters in our case.

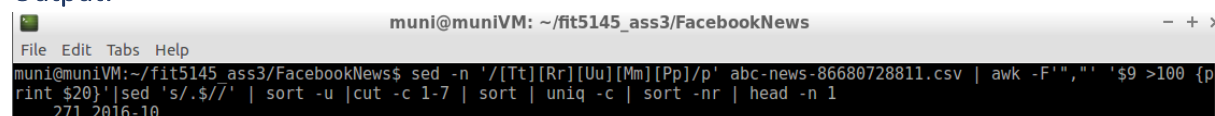
**uniq:** Filter adjacent matching lines from INPUT (or standard input), writing to OUTPUT (or standard output).

-c option used to prefix lines by the number of occurrences.

**head:** command to print the first few lines of file.

-n option specifies the first NUM lines to print instead of the first 10.

#### Output:



```
muni@muniVM: ~/fit5145_ass3/FacebookNews
File Edit Tabs Help
muni@muniVM:~/fit5145_ass3/FacebookNews$ sed -n '/[Tt][Rr][Uu][Mm][Pp]/p' abc-news-86680728811.csv | awk -F"," '{ $9 > 100 {print $20} }' | sed 's/.$// ' | sort -u | cut -c 1-7 | sort | uniq -c | sort -nr | head -n 1
271 2016-10
```

For **ABC news**, maximum articles about Trump were in **October 2016**.

**Command: fox-and-friends-111938618893743.csv**

```
sed -n '/[Tt][Rr][Uu][Mm][Pp]/p' fox-and-friends-111938618893743.csv | awk -F"," '{ $9 > 100 { print $20 } }' | sed 's/.$// ' | sort -u | cut -c 1-7 | sort | uniq -c | sort -nr | head -n 1
```

**Explanations:**

**sed:** command is a stream editor, used to perform basic text transformations on an input stream.

-n option used to suppress automatic printing of pattern space.

We use a Regular Expression to match the text.

**awk:** command used for pattern scanning and text processing.

-F option specifies the delimiter.

**sort:** command to sort, merge, or compares all the lines from the given files, or the standard input if no files are given.

-u option used to get first line of an equal run.

-n option used to compare according to string numerical value.

**cut:** Print selected parts of lines from each FILE to standard output.

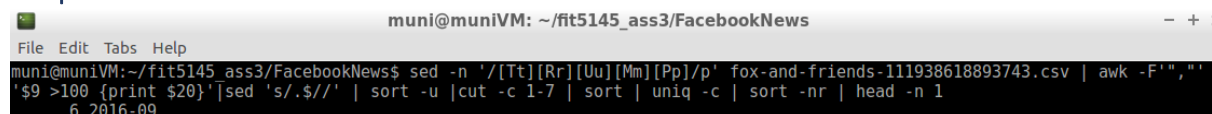
-c option used to select only these characters. 1-7 characters in our case.

**uniq:** Filter adjacent matching lines from INPUT (or standard input), writing to OUTPUT (or standard output).

-c option used to prefix lines by the number of occurrences.

**head:** command to print the first few lines of file.

-n option specifies the first NUM lines to print instead of the first 10.

**Output:**A terminal window titled 'muni@muniVM: ~/fit5145\_ass3/FacebookNews' showing the execution of the command. The output is '6 2016-09'.

```
muni@muniVM: ~/fit5145_ass3/FacebookNews$ sed -n '/[Tt][Rr][Uu][Mm][Pp]/p' fox-and-friends-111938618893743.csv | awk -F"," '{ $9 > 100 { print $20 } }' | sed 's/.$// ' | sort -u | cut -c 1-7 | sort | uniq -c | sort -nr | head -n 1
6 2016-09
```

For **Fox and Friends**, maximum articles about Trump were in **September 2016**.

### Command: the-new-york-times-5281959998

```
sed -n '/[Tt][Rr][Uu][Mm][Pp]/p' the-new-york-times-5281959998.csv | awk -F"," '{ $9 > 100 { print $20 } }' | sed 's/.$// ' | sort -u | cut -c 1-7 | sort | uniq -c | sort -nr | head -n 1
```

### Explanations:

**sed:** command is a stream editor, used to perform basic text transformations on an input stream.

-n option used to suppress automatic printing of pattern space.

We use a Regular Expression to match the text.

**awk:** command used for pattern scanning and text processing.

-F option specifies the delimiter.

**sort:** command to sort, merge, or compares all the lines from the given files, or the standard input if no files are given.

-u option used to get first line of an equal run.

-n option used to compare according to string numerical value.

**cut:** Print selected parts of lines from each FILE to standard output.

-c option used to select only these characters. 1-7 characters in our case.

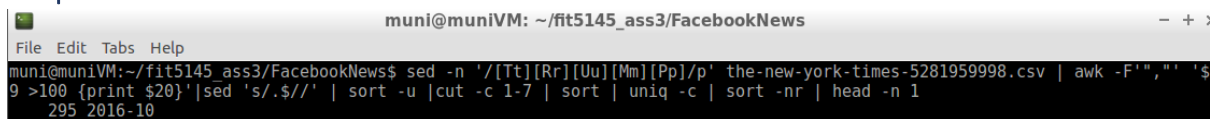
**uniq:** Filter adjacent matching lines from INPUT (or standard input), writing to OUTPUT (or standard output).

-c option used to prefix lines by the number of occurrences.

**head:** command to print the first few lines of file.

-n option specifies the first NUM lines to print instead of the first 10.

### Output:



```
muni@muniVM: ~/fit5145_ass3/FacebookNews
File Edit Tabs Help
muni@muniVM:~/fit5145_ass3/FacebookNews$ sed -n '/[Tt][Rr][Uu][Mm][Pp]/p' the-new-york-times-5281959998.csv | awk -F"," '{ $9 > 100 { print $20 } }' | sed 's/.$// ' | sort -u | cut -c 1-7 | sort | uniq -c | sort -nr | head -n 1
295 2016-10
```

For **The New York Times**, maximum articles about Trump were in **October 2016**.

**Command: usa-today-13652355666**

```
sed -n '/[Tt][Rr][Uu][Mm][Pp]/p' usa-today-13652355666.csv | awk -F"," ' $9 >100 {print $20}' | sed 's/.$$/' | sort -u | cut -c 1-7 | sort | uniq -c | sort -nr | head -n 1
```

**Explanations:**

**sed:** command is a stream editor, used to perform basic text transformations on an input stream.

-n option used to suppress automatic printing of pattern space.

We use a Regular Expression to match the text.

**awk:** command used for pattern scanning and text processing.

-F option specifies the delimiter.

**sort:** command to sort, merge, or compares all the lines from the given files, or the standard input if no files are given.

-u option used to get first line of an equal run.

-n option used to compare according to string numerical value.

**cut:** Print selected parts of lines from each FILE to standard output.

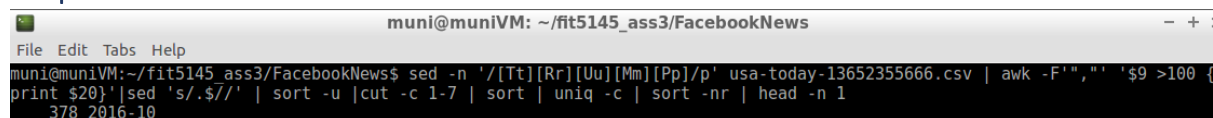
-c option used to select only these characters. 1-7 characters in our case.

**uniq:** Filter adjacent matching lines from INPUT (or standard input), writing to OUTPUT (or standard output).

-c option used to prefix lines by the number of occurrences.

**head:** command to print the first few lines of file.

-n option specifies the first NUM lines to print instead of the first 10.

**Output:**A terminal window titled 'muni@muniVM: ~/fit5145\_ass3/FacebookNews' showing the execution of the command. The output is '378 2016-10', indicating 378 occurrences of the date '2016-10' in the data.

```
muni@muniVM: ~/fit5145_ass3/FacebookNews$ sed -n '/[Tt][Rr][Uu][Mm][Pp]/p' usa-today-13652355666.csv | awk -F"," ' $9 >100 {print $20}' | sed 's/.$$/' | sort -u | cut -c 1-7 | sort | uniq -c | sort -nr | head -n 1
378 2016-10
```

For **USA Today**, maximum articles about Trump were in **October 2016**.



## Task B: Graphing the Data in R

1) How many times does the term 'Trump' appear in the post message? (use Unix shell to answer to this question)

Command:

```
awk -F'"',"' '{print $4}' *.csv | grep Trump | wc -l
```

Output:

```
muni@muniVM:~/fit5145_ass3/FacebookNews$ awk -F'"',"' '{print $4}' *.csv | grep Trump | wc -l
5136
```

Trump appears 5136 times in Post message.

2) We want to consider how the amount of discussion regarding Donald Trump varies over the time period covered by the data file. To answer this question, you will need to extract the timestamps for all posts referring to Trump using shell. You will then need to read them into R and generate a histogram.

Command:

```
cat *.csv | awk -F'"',"' '$4 ~ /\. *Trump\.*/ {print $20}' | sed 's/.$//' | sort -u >
trump_sort_timeline.txt
```

```
sed -i '1d;$d' trump_sort_timeline.txt
```

```
{ echo posted_at; cat trump_sort_timeline.txt; } > temp.txt && mv temp.txt trump_sort_timeline.csv
```

Output:

```
muni@muniVM: ~/fit5145_ass3/FacebookNews
File Edit Tabs Help
muni@muniVM:~$ cd /home/muni/fit5145_ass3/FacebookNews
muni@muniVM:~/fit5145_ass3/FacebookNews$ cat *.csv | awk -F '"',"' '$4 ~ /\. *Trump\.*/ {print $20}' | sed 's/.$//' | sort
> trump_sort_timeline.txt
muni@muniVM:~/fit5145_ass3/FacebookNews$ sed -i '1d;$d' trump_sort_timeline.txt
muni@muniVM:~/fit5145_ass3/FacebookNews$ { echo posted_at; cat trump_sort_timeline.txt; } > temp.txt && mv temp.txt trump_
rt_timeline.csv
```

**trump\_sort\_timeline.csv** file is generated having the timestamp of all the articles having Trump.

## Making a histogram of the data using R:

# 31187366 - Assignment 3

```
library('tidyverse')
```

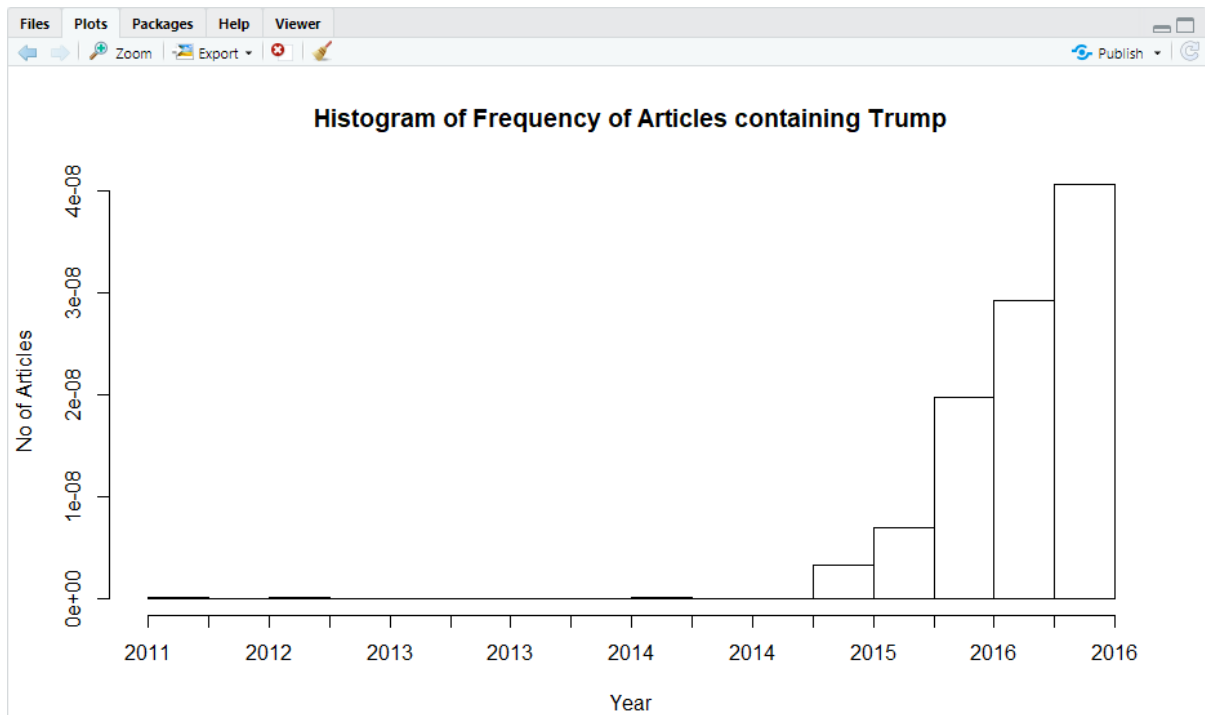
```
df = read_csv(file = 'trump_sort_timeline.csv')
```

```
df$posted_at = strptime(df$posted_at, "%Y-%m-%d %H:%M:%S")
```

```
hist(df$posted_at[,breaks = 12])
```

D:/Data/Monash - Master of Data Science/Semester 2 - July 2020/FIT5145 Introduction to data science/week7/lab/lab\_task\_7 - RStudio

```
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
ass3_script.R*
1 # 31187366 - Assignment 3
2
3 library('tidyverse')
4
5 df = read_csv(file = 'trump_sort_timeline.csv')
6 df$posted_at = strptime(df$posted_at, "%Y-%m-%d %H:%M:%S")
7
8 hist(df$posted_at,
9     breaks = 12,
10    xlab = "Year",
11    ylab = "No of Articles",
12    main = paste("Histogram of" , "Frequency of Articles containing Trump"))
```



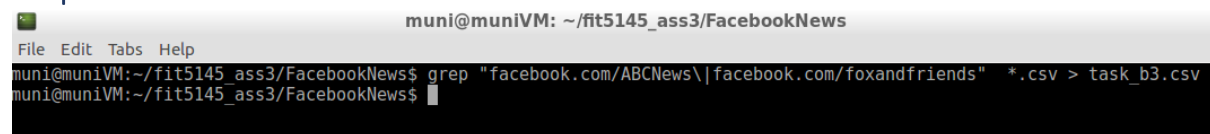
Q.3) In this question, we want to investigate the Facebook posts of a few top media sources. To answer this question, you will need to extract the Facebook posts made on the pages of "abc-news", "cnn" and "fox-news" from your original Facebook dataset.

1. Use the Unix shell to first generate a file containing all the records belonging to "abc-news", "cnn" and "fox-news" only. Then read the resulting file in R.

Command:

```
grep "facebook.com/ABCNews\|facebook.com/foxandfriends" *.csv > task_b3.csv
```

Output:

A terminal window screenshot showing a command being executed. The prompt is 'muni@muniVM: ~/fit5145\_ass3/FacebookNews'. The command is 'grep "facebook.com/ABCNews\|facebook.com/foxandfriends" \*.csv > task\_b3.csv'. The output shows the command being executed successfully, with a cursor at the end of the line.

```
muni@muniVM: ~/fit5145_ass3/FacebookNews
File Edit Tabs Help
muni@muniVM:~/fit5145_ass3/FacebookNews$ grep "facebook.com/ABCNews\|facebook.com/foxandfriends" *.csv > task_b3.csv
muni@muniVM:~/fit5145_ass3/FacebookNews$
```

Task\_b3.csv file is generated having data of posts made by "abc-news" and "fox-news". There is no contribution of "cnn" media to the final csv file.

2. Background: We now want to see if any relationship exists between the number of times a post is shared on Facebook and the number of likes it generates. Task: Use appropriate R code to generate a plot showing the relationship between the number of shares and the number of likes in your dataset. Do you see any relationship?

R Code:

```
# load the required libraries
library("tidyverse")
library("ggplot2")

# read the csv file and ignore con_names
df2 = read_csv("task_b3.csv", col_names = FALSE)

# make list of column names
col_names = list("file_name", "id", "page_id", "name", "message", "description",
  "caption", "status_type", "likes_count",
  "comments_count", "shares_count", "love_count", "wow_count", "haha_count",
  "sad_count", "thankful_count",
  "angry_count", "link", "picture", "posted_at")

# add column names to the dataframe
names(df2) = col_names

# calculate the correlation
cor(df2$shares_count, df2$likes_count)

# make a scatter plot of shares v/s likes
```

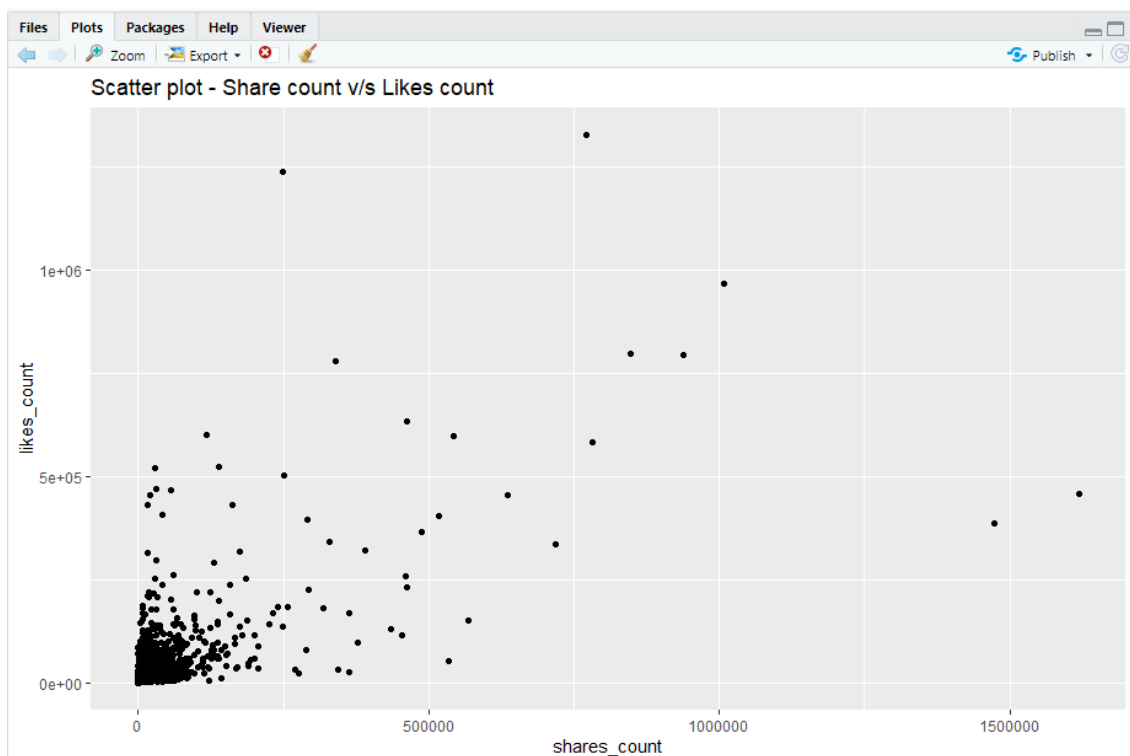
```
ggplot(df2, aes(shares_count, likes_count)) +  
  geom_point() +  
  ggtitle("Iris scatter plot - petals")
```

### Output:

D:/Data/Monash - Master of Data Science/Semester 2 - July 2020/FIT5145 Introduction to data science/week7/lab/lab\_task\_7 - RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

```
1 # 31187366 - Assignment 3 - Task B3-2  
2  
3 # load the required libraries  
4 library("tidyverse")  
5 library("ggplot2")  
6  
7 # read the csv file and ignore con_names  
8 df2 = read_csv("task_b3.csv", col_names = FALSE)  
9  
10 # make list of column names  
11 col_names = list("file_name", "id", "page_id", "name", "message", "description", "caption", "status_type", "likes_count",  
12 "comments_count", "shares_count", "love_count", "wow_count", "haha_count", "sad_count", "thankful_count",  
13 "angry_count", "link", "picture", "posted_at")  
14  
15 # add column names to the dataframe  
16 names(df2) = col_names  
17  
18 # calculate the correlation  
19 cor(df2$shares_count, df2$likes_count)  
20  
21 # make a scatter plot of shares v/s likes  
22 ggplot(df2, aes(shares_count, likes_count)) +  
23   geom_point() +  
24   ggtitle("Iris scatter plot - petals")  
25
```



There is a **linear relationship** between number of share and number of likes.

3. Fit a linear regression model using R to the above data (i.e., shares\_count and likes\_count) and plot the linear fit. Does it look like a good fit to you?

R code:

```
# fit a linear Regression model
```

```
lmMod <- lm(likes_count~shares_count, data=df2)
```

```
# predict the no of likes
```

```
likesPred <- predict(lmMod, newdata = data.frame(shares_count=c(0,100,1000,10000,100000)))
```

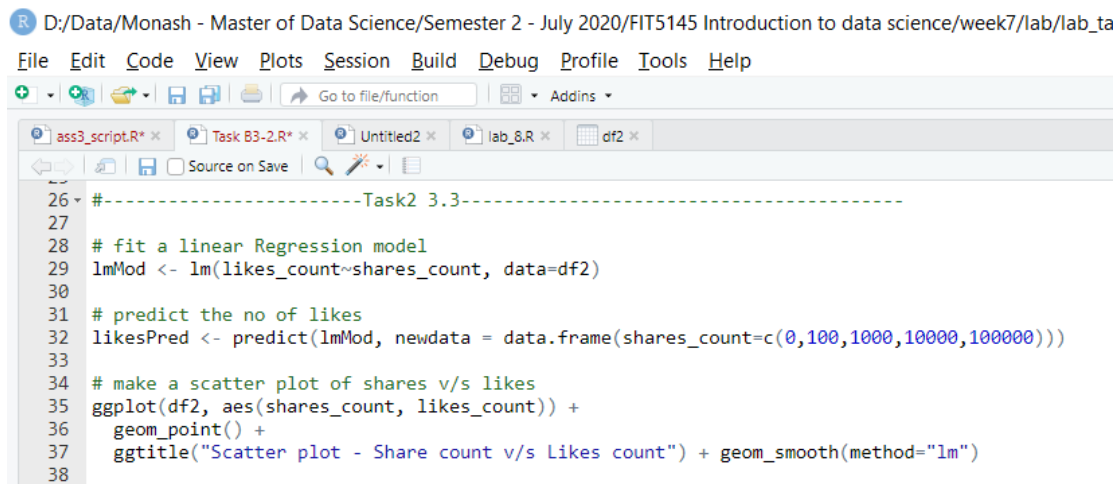
```
# make a scatter plot of shares v/s likes
```

```
ggplot(df2, aes(shares_count, likes_count)) +
```

```
  geom_point() +
```

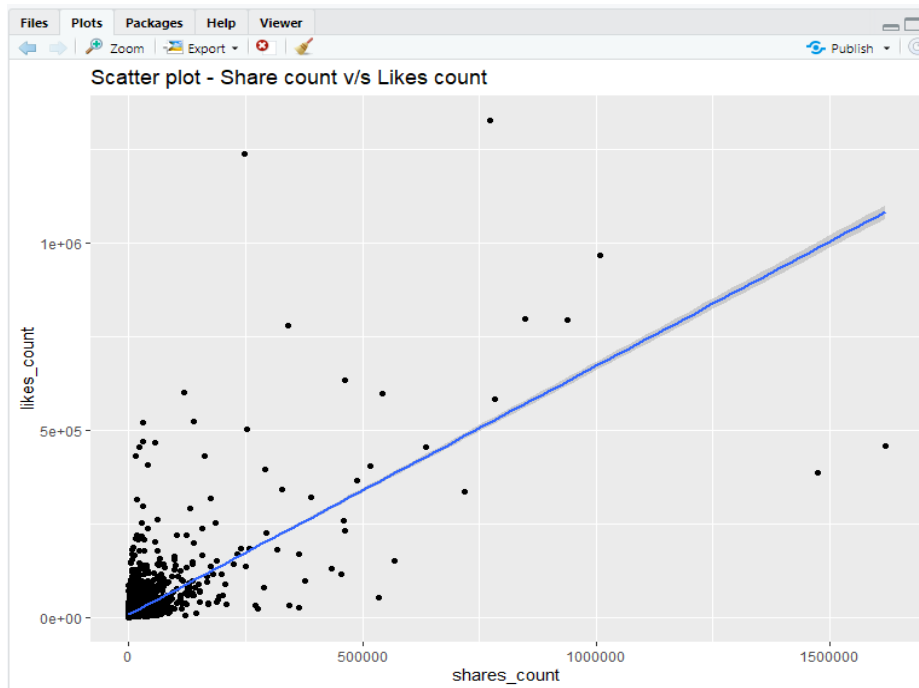
```
  ggtitle("Scatter plot - Share count v/s Likes count") + geom_smooth(method="lm")
```

Output:



The screenshot shows the RStudio interface with the following elements:

- Top bar: File Edit Code View Plots Session Build Debug Profile Tools Help
- Toolbar: Icons for running, saving, and other RStudio functions.
- Tab bar: Shows several open files including 'ass3\_script.R', 'Task B3-2.R', 'Untitled2', 'lab\_8.R', and 'df2'.
- Source editor: Displays the R code from the previous block, starting with a comment line: `26 #-----Task2 3.3-----`.



As we can see that most of the data points are close to the line of best fit, it is a reasonable fit.

4. Use the linear fit to predict the number of likes a post will generate if it is shared 0 times, 100 times, 1000 times, 10000 times and 100000 times on Facebook.

R code:

```
# predict the likes
```

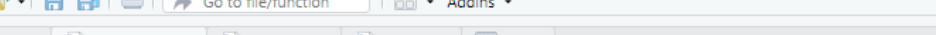
```
likesPred <- predict(lmMod, newdata = data.frame(shares_count=c(0,100,1000,10000,100000)))
```

```
print(likesPred)
```

Output:

D:/Data/Monash - Master of Data Science/Semester 2 - July 2020/FIT5145 Introduction to data science/week7/lab/lab\_task\_7

File Edit Code View Plots Session Build Debug Profile Tools Help



```
40 #-----Task2 3.4-----
41
42 # predict the likes
43 likesPred <- predict(lmMod, newdata = data.frame(shares_count=c(0,100,1000,10000,100000)))
44 print(likesPred)
45
```

```
48:1 Task2 3.4 ↕  
Console Terminal × Jobs ×  
D:/Data/Monash - Master of Data Science/Semester 2 - July 2020/FIT5145 Introduction to data science/ass3/ ↗  
> # predict the likes  
> likesPred <- predict(lmMod, newdata = data.frame(shares_count=c(0,100,1000,10000,100000)))  
> print(likesPred)  
      1      2      3      4      5  
6654 6720 7319 13300 73116  
> |
```

Likes predicted for 0,100,1000,10000,100000 shares are 6654,6720,7319,13300 and 73116 respectively.