

### FIT1043 Assignment 3

Student ID: 29389356

Student Name: Anisha Shrestha

Tutorial Code:01-p1

Tutor: Heshan and Vijaya

1.

<b>Code</b>	<code>unzip FB_Dataset.csv.zip</code>
<b>Output</b>	<pre>KC-G@DESKTOP-GE0BIA4 /cygdrive/d/FIT1043tutorial/ass3 \$ unzip FB_Dataset.csv.zip Archive:  FB_Dataset.csv.zip   inflating: FB_Dataset.csv    creating: __MACOSX/   inflating: __MACOSX/._FB_Dataset.csv</pre>
<b>Explanation</b>	This code is used to unzip/uncompress the zipped/compressed file.

<b>Code</b>	<code>ls -lh FB_Dataset.csv</code>
<b>Output</b>	<pre>KC-G@DESKTOP-GE0BIA4 /cygdrive/d/FIT1043tutorial/ass3 \$ ls -lh FB_Dataset.csv -rw-r--r--+ 1 KC-G None 344M Sep 13  2019 FB_Dataset.csv</pre>
<b>Explanation</b>	ls command is used to get the list of contents in the directory. -lh is one of the options for ls command. This code is used to get the file size of the unzipped dataset.

The file size of the unzipped dataset is 344M.

2.

<b>Code</b>	<code>head -n1 FB_dataset.csv</code>
<b>Output</b>	<pre>KC-G@DESKTOP-GE0BIA4 /cygdrive/d/FIT1043tutorial/ass3 \$ head -n1 FB_dataset.csv 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</pre>
<b>Explanation</b>	In the above code head specifies to print out the first parts of the FB_dataset and -n1 is used to get the first row/line of the dataset.

By reviewing the output of the code below we see that “,” is used to separate the columns in the file. The columns in the dataset are as follows: 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.

3. While exploring the columns of the dataset, we find a column name called page\_id which would be unique for each page. Therefore by columning the number of unique values in this column we can calculate the number of unique pages in the dataset.

<b>Code</b>	<code>awk -F ',' '{print \$3}' FB_dataset.csv sort uniq</code>
<b>Output</b>	<pre>\$ awk -F ',' '{print \$3}' FB_dataset.csv sort uniq 1.12E+14 1.31E+11 1.56E+14 10606591490 10643211755 13652355666 15704546335 18468761129 2.29E+11 5281959998 5550296508 5863113009 6250307292 8304333127 86680728811 page_id</pre>
<b>Explanation</b>	The code is used to get the list of unique values in the 3rd column which displays data of page_id. “awk” processes the csv file one line at a time. In the code we’re specifying the delimiter is “,” to separate the line by the comma into columns. Then we apply the “sort” command to sort the lines and remove successive duplicates. The uniq command then gives the list of unique values in column 3 and wc -l counts the lines.

Since from the above code, we know that column 3 also includes the header name called page\_id which would be counted as a unique value in the column. Therefore, we need to subtract 1 from the total count number resulting from the query. and counts them

<b>Code</b>	<code>\$ awk -F ',' '{print \$3}' FB_dataset.csv sort uniq wc -l</code>
<b>Output</b>	<pre>KC-G@DESKTOP-GE0BIA4 /cygdrive/d/FIT1043tutorial/ass3 \$ awk -F ',' '{print \$3}' FB_dataset.csv sort uniq wc -l 16</pre>
<b>Explanation</b>	The above code is an extension to the previous code, it is used to get the unique ids from the page_id column and count them. In this code wc -l is added and it is used to print the new line counts of the result shown in the previous code i.e. the list of unique page_ids. .

Thus, there are 15 unique page\_id in the dataset.

#### 4.

Assuming that the data in FB\_dataset is ordered, the first row of the dataset would be the earliest post made and the last record would be the latest post made, therefore by getting the values of the first and last row from the posted\_at column will give us the date range of the facebook posts on this file.

<b>Code</b>	<code>awk -F ',' '{print \$21}' FB_dataset.csv head -n2</code>
<b>Output</b>	<pre>KC-G@DESKTOP-GE0BIA4 /cygdrive/d/FIT1043tutorial/ass3 \$ awk -F ',' '{print \$21}' FB_dataset.csv head -n2 posted_at 1/1/12 0:30</pre>
<b>Explanation</b>	This code is used to get the first 2 values of column 21 which would be the column name and the date of when the post was posted. Here we specify that the delimiter is a ',' and print out column 21 of FB_dataset using awk and then print out the first 2 values using head -n2.

<b>Code</b>	<code>awk -F ',' '{print \$21}' FB_dataset.csv tail -n1</code>
<b>Output</b>	<pre>KC-G@DESKTOP-GE0BIA4 /cygdrive/d/FIT1043tutorial/ass3 \$ awk -F ',' '{print \$21}' FB_dataset.csv tail -n1 7/11/16 23:45</pre>
<b>Explanation</b>	This code is used to get the last value of column 21. Similar to the previous code, we use awk to specify the delimiter and print column 21 and use tail to output the last parts of the file. Then by specifying -n1 we print the last value of the posted_at column.

The date for the Facebook posts ranges from 1/1/12 to 7/11/16.

#### 5.

Looking at the output of question 2 we know that column 4 contains data of the page\_name, Therefore in the following code we'll be printing out column no. 4.

<b>Code</b>	<code>awk -F ',' '{print \$4}' FB_dataset.csv grep -o -i "Donald Trump" wc -l</code>
<b>Output</b>	<pre>KC-G@DESKTOP-GE0BIA4 /cygdrive/d/FIT1043tutorial/ass3 \$ awk -F ',' '{print \$4}' FB_dataset.csv grep -o -i "Donald Trump" wc -l 7610</pre>
<b>Explanation</b>	<p>This code is used to count the number of times "Donald Trump" is mentioned in the contents under post_names, which is the 4th column.</p> <p>The code uses awk, a program that will process the csv file one line at a time. In the code, we specify to break up each line of column 4 separated by the delimiter: ",". We use grep to filter the text by the word "Donald</p>

	Trump” and use -o print each matched word in a separate line and -i to ignore case sensitivity. Then we finally used -i to print out the count number of lines filtered by grep.
--	--

The word “Donald Trump” appears 7610 times in the content of the post names.

The following code is used to print the post name and the date of the first mention of “Donald Trump” in the post names.

<b>Code</b>	<pre>awk -F ',' '{print \$2,\$4,\$21}' FB_dataset.csv grep -i "Donald Trump" head -n1</pre>
<b>Output</b>	<pre>KC-G@DESKTOP-GE0BIA4 /cygdrive/d/FIT1043tutorial/ass3 \$ awk -F ',' '{print \$2,\$4,\$21}' FB_dataset.csv grep -i "Donald Trump" head -n1 86680728811_325466700825405 Donald Trump Staff Reaching Out to Financers .. Campaign Managers to Explore Third Party Bid 30/1/12 21:07</pre>
<b>Explanation</b>	The code uses awk, a program that will process the csv file one line at a time. In the code, we specify to break up each line of column 2, column 4 and column 21 separated by the delimiter: “,”. We use grep to filter the text by the word “Donald Trump” and -i to ignore case sensitivity. Then we finally used head -n1 print out the first line filtered by grep.

The first mention of “Donald Trump” was made on 30/1/12 21:07 on the post named “Donald Trump Staff Reaching Out to Financers .. Campaign Managers to Explore Third Party Bid”.

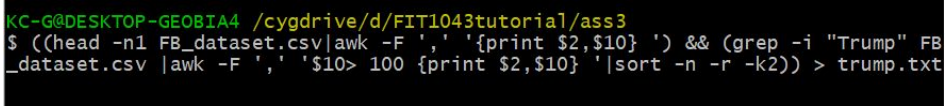
Exploring the columns, data from columns like\_count, love\_count, haha\_count and thankful\_count can be considered as positive reactions since these counts express positive feelings users got from reading posts and angry\_count can be considered as negative reactions as it can be accounted as negative feelings users got while reading the post.

In the following code we use the value of post\_id (86680728811\_325466700825405) we got from the previous code to extract columns of the row to determine the positive and negative reactions of the post as post\_id is unique for every post.

<b>Code</b>	<pre>awk -F ',' ' \$2 == "86680728811_325466700825405"    NR==1 {print \$2,\$10,\$13,\$14,\$15,\$17,\$18}' FB_dataset.csv</pre>
<b>Output</b>	<pre>KC-G@DESKTOP-GE0BIA4 /cygdrive/d/FIT1043tutorial/ass3 \$ awk -F ',' ' \$2 == "86680728811_325466700825405"    NR==1 {print \$2,\$10,\$13,\$14,\$15,\$17,\$18}' FB_dataset.csv post_id likes_count love_count wow_count haha_count thankful_count angry_count 86680728811_325466700825405 174 0 0 0 0 0</pre>
<b>Explanation</b>	In this code we use awk to specify the delimiter which is a ‘,’ and the columns that is needed to be printed out (column 2, 10, 13, 14, 15, 17, 18) that match the condition \$2=="86680728811_325466700825405" to print values of the row that matched the post_id and NR==1, which states to print the first row to the output.

After exploring the row we see that all columns that account for reactions except for likes\_count have the value 0. According to the posted\_date it is clear that these columns have the value 0 because this post was posted before Facebook had introduced allowing reactions to the post. Facebook only added reactions to their post from February 24, 2016 onwards ([reference](#)). Therefore, we see 0 negative reactions against this post. While we can see that 349 liked the post which shows that they responded positively to it, however we don't know the amount of people who viewed the post and were dissatisfied with the post. Therefore, it's difficult to conclude whether people's reactions were positive or negative to this.

## 6.

<b>Code</b>	<pre>((head -n1 FB_dataset.csv awk -F ',' '{print \$2,\$10} ') &amp;&amp; (grep -i "Trump" FB_dataset.csv  awk -F ',' '\$10&gt; 100 {print \$2,\$10} ' sort -n -r -k2)) &gt; trump.txt</pre>
<b>Output</b>	
<b>Explanation</b>	<p>This code is used to print the post_id and like_count columns of posts that have mentioned the word “Trump” on them, sort it in descending order and save the results to the file called trump.txt.</p> <p>In the code we see two conditions separated by ‘&amp;&amp;’ wrapped inside an outer parentheses which says that the combined results of the conditions to be saved inside trump.txt by using the &gt; command.</p> <p>In the first condition we print out the the 2nd and the 10th column names by using head -n1 to print out the 1st row of the dataset and using awk program to specify the columns are separated by “,”. In the second condition we’re saying the for the rows with the word “Trump ” which is filtered using grep to print the 2nd and 10th columns only if the value for the 10th column is greater than 100 in the awk statement.</p> <p>Then we use sort function to sort the data by greatest number of likes. We use -n to perform numeric sort, -r to sort in reverse order ( descending order) and -k2 to specify that the data should be sorted by the second column.</p> <p>Lastly, we save the sorted results to trump.txt using the command “&gt;”.</p>

<b>Code</b>	<pre>head -n6 trump.txt</pre>
-------------	-------------------------------

<b>Output</b>	<pre>KC-G@DESKTOP-GE0BIA4 /cygdrive/d/FIT1043tutorial/ass3 \$ head -n6 trump.txt post_id likes_count _22228735667216_1015396016795221722 368179 5550296508_10154298504746509 248012 18468761129_10153524839811130 229187 15704546335_10154108339971336 222119 15704546335_10154646163096336 218748</pre>
<b>Explanation</b>	In this code by using head -n6 we're printing out the header row and first 5 rows of the dataset.

## B1.

The following code was written in cygwin to filter the FB\_dataset with appropriate rows required for the tasks under B1.

<b>Code</b>	<pre>awk -F ',' '\$1== "the-wall-street-journal"    NR==1 {print \$8","\$11 }' FB_dataset.csv &gt; wall-street-journal.csv</pre>
<b>Output</b>	<pre>KC-G@DESKTOP-GE0BIA4 /cygdrive/d/FIT1043tutorial/ass3 \$ awk -F ',' '\$1== "the-wall-street-journal"    NR==1 {print \$8","\$11 }' FB_data set.csv &gt; wall-street-journal.csv</pre>
<b>Explanation</b>	The above code is used to extract the 8th and 11th column i.e. post_type and comments_count columns having the page_name "the-wall-street-journal" and save the results to "wall-street-journal.csv" file which is filtered by the command \$1=="the-wallstreet-journal"

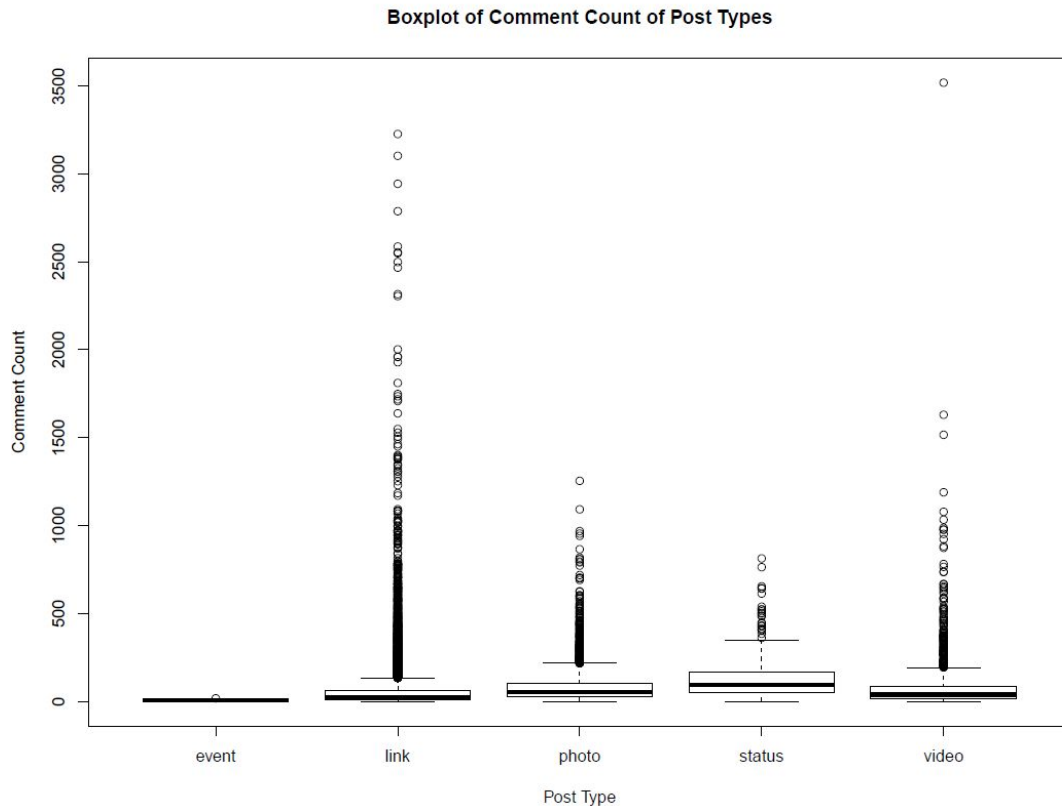
## 7.

<b>Code</b>	<pre>wallstreetJ= read.csv("wall-street-journal.csv" ) head(wallstreet)  &gt; head(wallstreetJ)   post_type comments_count 1      link              60 2      link              22 3      link              19 4      link              36 5      link             149 6      link              31 ~    #creating a dataframe with post types having less than 4000 comments wallstreetJ &lt;- wallstreetJ[(wallstreetJ\$comments_count&lt; 4000),] head(wallstreetJ)</pre>
-------------	---

	<pre> &gt; head(wallstreetJ)   post_type comments_count 1      link             60 2      link             22 3      link             19 4      link             36 5      link            149 6      link             31  # ordering dataset in a ascending order by comments_counts wallstreetJ = wallstreetJ[order(wallstreetJ\$comments_count, decreasing = TRUE),] head(wallstreetJ) </pre>
Output	<pre> &gt; head(wallstreetJ)       post_type comments_count 33796      video          3519 5864       link          3227 20475       link          3103 7777       link          2944 4731       link          2788 4943       link          2587 ~   </pre>
Explanation	<p>The above code we read the csv file and filter the dataframe with rows having comment_count of less than 4000. Then we order the dataframe in an ascending order.</p>

Code	<pre> #creating a boxplot boxplot(wallstreetJ\$comments_count ~ wallstreetJ\$post_type, main="Boxplot of Comment Count of Post Types", xlab="Post Type", ylab="Comment Count") </pre>
Explanation	<p>The above code is used to create boxplots for different post types against the comment count with appropriate labels.</p>





From the above figure we see that for post type “event” has the lowest median while “status” has the highest median among the post types. We can also see that the spread of the box plot varies across the post types. For the post type “link” and “video” we see a high variation in the number of comments, “photo” and “status” have lower variations on them than “link” and “video”, while for “event” the range is small and constricted.

Looking at the shape of the box plot for link, photo, status and video we can see that the median lies closer to the Q1 value. they have a longer upper whisker and have a series of outliers after the upper boundary which suggests that the data are skewed to the right. The long upper tail says that there are few posts that belong to these post types that have a high number of comments i.e values higher than the max value determined by the IQR rule. Also because of the high variations in the number of comments across the other post types we cannot read the boxplot for “event”. Since the values for most of the post types are skewed to the right with a high number of outliers, median would be a better measure to determine the center as it is not affected by the outliers. Assuming that the most engaging post type is the type that has the highest average number of comments on the posts , we can look at the median value to determine it.

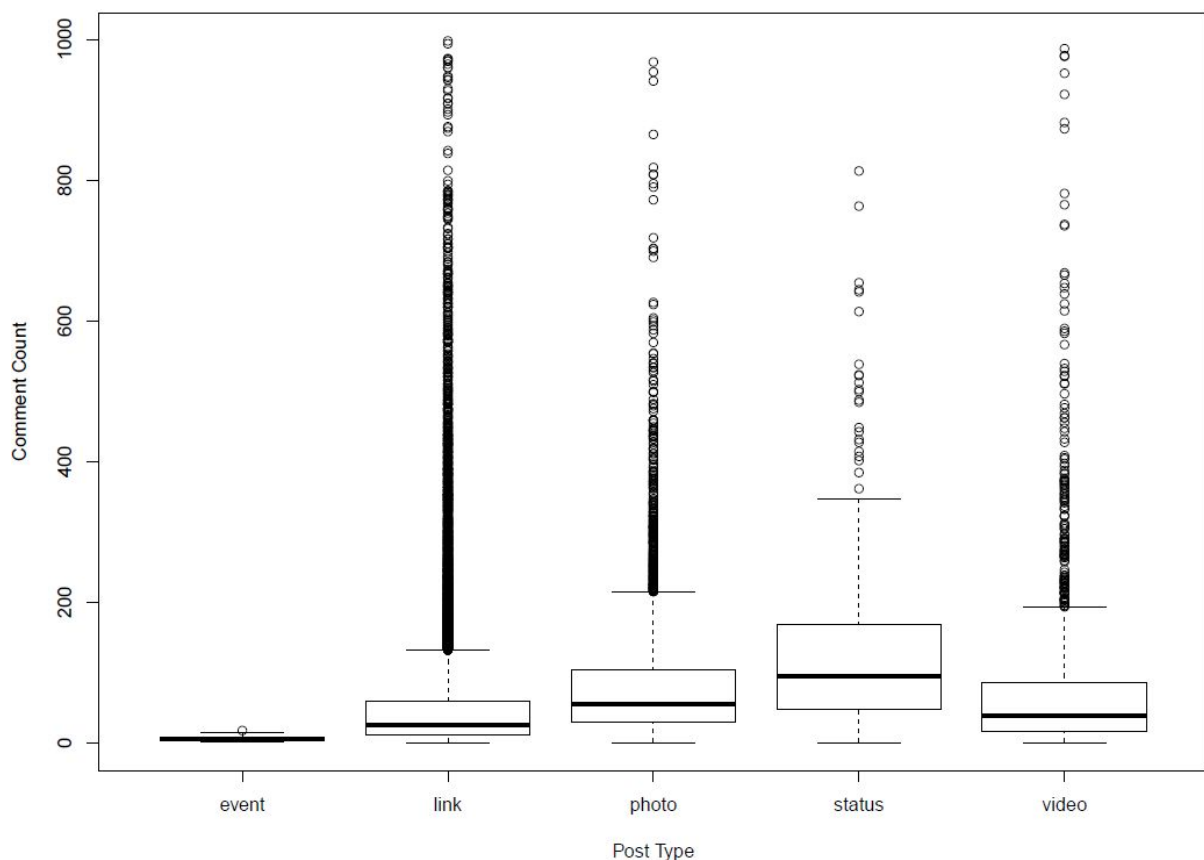
In the above figure, the medians for the boxplots seem fairly close to one another where status has the highest median and event has the lowest. While we see variations of comments with high count in "link" suggesting higher user engagement on these posts, the median for it is lower than status. Therefore , since "status" has the highest median than the rest of the post types it is the most engaging post type.



8.

<b>Code</b>	<pre> #Q8 filter_wsJ = filter(wallstreetJ, wallstreetJ\$comments_count &lt;= 1000) head(filter_wsJ) &gt; head(filter_wsJ)   post_type comments_count 23717    link           998 34085    link           994 29439    video          987 8112     video          977 22276    video          976 35358    link           973 . .  #creating a boxplot boxplot(filter_wsJ\$comments_count ~ filter_wsJ\$post_type, main="Boxplot of Comment Count of Post Types", xlab="Post Type", ylab="Comment Count") </pre>
<b>Explanation</b>	<p>In the above code fragment, we create a dataframe with post_types with comment count of less than or equal to 1000. Then we create boxplots for different post types against the comment counts and add necessary labels to the diagram.</p>

Boxplot of Comment Count of Post Types



9.

The post type status has the highest median, therefore, on average has been the most effective for “the-wall-street-journal”.

## B2.

In the given dataset, I consider likes\_count, love\_count, wow\_count, haha\_count, sad\_count, thankful\_count and anger\_count as reactions as it depicts the emotions a user perceives from a particular post. Furthermore, Facebook officially considers like, love, wow, haha, sad, thankful and angry as reactions provided by them for users to use to express how they feel. Therefore, through the mentioned reaction icons a user would be expressing their emotions.

<https://en.facebookbrand.com/facebookapp/assets/reactions/>

<b>Code</b>	<pre>KC-G@DESKTOP-GE0BIA4 /cygdrive/d/FIT1043tutorial/ass3 \$ awk -F ' ' '\$1=="abc-news"    NR==1 {print \$1", "\$2", "\$5", "\$10", "\$13", "\$14", "\$15", "\$16", "\$17", "\$18", "\$21}' FB_dataset.csv &gt; abc_news_d.csv  KC-G@DESKTOP-GE0BIA4 /cygdrive/d/FIT1043tutorial/ass3 \$ head -n5 abc_news_d.csv</pre>
<b>Output</b>	<pre>KC-G@DESKTOP-GE0BIA4 /cygdrive/d/FIT1043tutorial/ass3 \$ awk -F ' ' '\$1=="abc-news"    NR==1 {print \$1", "\$2", "\$5", "\$10", "\$13", "\$14", "\$15", "\$16", "\$17", "\$18", "\$21}' FB_dataset.csv &gt; abc_news_d.csv  KC-G@DESKTOP-GE0BIA4 /cygdrive/d/FIT1043tutorial/ass3 \$ head -n5 abc_news_d.csv page_name,post_id,message,likes_count,love_count,wow_count,haha_count,sad_count,thankful_count,angry_count,posted_at abc-news,86680728811_272953252761568,Roberts took the unusual step of devoting the majority of his annual report to the issue of judicial ethics.,61,0,0,0,0,0,0,1/1/12 30 abc-news,86680728811_273859942672742,Do you agree with the new law?,120,0,0,0,0,0,0,1/1/12 1:08 abc-news,86680728811_10150498674478812,Some pretty cool confetti will rain down on New York City celebrators.,271,0,0,0,0,0,0,1/1/12 2:00 abc-news,86680728811_244555465618151,NULL,140,0,0,0,0,0,0,1/1/12 2:35</pre>
<b>Explanation</b>	<p>This code is used to filter the dataset by “abc_news” in column 1 which is post_names and print out a subset of the dataset with columns 1, 5, 10, 13, 14, 15, 16, 17, 18, 21 which are post_names,post_id, message, like_counts, love_count,wow_count,haha_count,sad_count,thankful_count,angry_count and posted_at and save the result to acb_news_d.csv.</p>

10.

<b>Code</b>	<pre>abc_news = read.csv("abc_news_d.csv")  filter_abc_news = abc_news[(grep("Donald Trump", abc_news\$message, ignore.case = TRUE)), ]</pre>
-------------	---

### Output

page_name	post_id	message	likes_count	love_count	wow_count	haha_count	sad_count	thankful_count	angry_count	po
abc-news	86680728811_10152623555133812	Vera Coking became a folk hero for resisting decades-long e...	1149	0	0	0	0	0	0	0
abc-news	86680728811_10152623774158812	The 91-year-old woman once called Donald Trump 'a maggo...	1348	0	0	0	0	0	0	0
abc-news	86680728811_10152638605648812	Donald Trump has a message for the two Atlantic City casino...	678	0	0	0	0	0	0	0
abc-news	86680728811_10152981635443812	In an appearance tonight at the Economic Club in Washingto...	3484	0	0	0	0	0	0	0
abc-news	86680728811_10153511140788812	JUST IN: Is Donald Trump actually running for president this ...	2093	1	0	0	0	0	0	0

<b>Explanation</b>	In the above code fragment, we read the CSV file in R as <code>abc_news</code> then filter the rows of the dataset by matching the word “Donald Trump” in the message column of the <code>abc_news</code> data frame and print out all columns of the matched row and save the results into the data frame <code>filter_abc_news</code> .
--------------------	---

<b>Code</b>	<pre>filter_abc_news = filter_abc_news[c(2,4:11)] head(filter_abc_news)</pre>
-------------	---

### Output

	post_id	likes_count	love_count	wow_count	haha_count	sad_count	thankful_count	angry_count	posted_at
14415	86680728811_10152623555133812	1149	0	0	0	0	0	0	0 31/7/14 8:08
14416	86680728811_10152623774158812	1348	0	0	0	0	0	0	0 31/7/14 10:48
14556	86680728811_10152638605648812	678	0	0	0	0	0	0	0 6/8/14 9:24
17120	86680728811_10152981635443812	3484	0	0	0	0	0	0	0 16/12/14 3:34
20920	86680728811_10153511140788812	2093	1	0	0	0	0	0	0 16/6/15 15:55
20928	86680728811_10153511641463812	2088	0	0	0	0	0	0	0 16/6/15 19:56

<b>Explanation</b>	In the above code fragment, we're replacing the old data frame by binding columns 2, 4, 5, 6, 7, 8, 9, 10 and 11 and storing it back as <code>filter_abc_news</code> .
--------------------	--

<b>Code</b>	<pre>filter_abc_news\$total_reactions = rowSums(filter_abc_news[,3:9])</pre>
-------------	--

### Output

	post_id	likes_count	love_count	wow_count	haha_count	sad_count	thankful_count	angry_count	posted_at	total_reactions
14415	86680728811_10152623555133812	1149	0	0	0	0	0	0	0 31/7/14 8:08	1149
14416	86680728811_10152623774158812	1348	0	0	0	0	0	0	0 31/7/14 10:48	1348
14556	86680728811_10152638605648812	678	0	0	0	0	0	0	0 6/8/14 9:24	678
17120	86680728811_10152981635443812	3484	0	0	0	0	0	0	0 16/12/14 3:34	3484
20920	86680728811_10153511140788812	2093	1	0	0	0	0	0	0 16/6/15 15:55	2094
20928	86680728811_10153511641463812	2088	0	0	0	0	0	0	0 16/6/15 19:56	2088

<b>Explanation</b>	In the above code fragment, we're summing the values from column 3 to column 9 for all rows and storing the output value in the respective cells in the newly added column called <code>total_reactions</code> .
--------------------	--

<b>Code</b>	<pre>library(lubridate)  filter_abc_news\$date= as.Date(filter_abc_news\$posted_at,format("%d/%m/%Y %H:%M"))</pre>
-------------	--

	<pre>filter_abc_news\$time= format(as_datetime(filter_abc_news\$posted_at,format("%d/%m/%Y %H:%M")),format="%H:%M") head(filter_abc_news)</pre>
	<pre>post_id likes_count love_count wow_count haha_count sad_count thankful_count angry_count posted_at total_reactions date 86680728811_10152623555133812 1149 0 0 0 0 0 0 31/7/14 8:08 1149 0014-07-31 86680728811_10152623774158812 1348 0 0 0 0 0 0 31/7/14 10:48 1348 0014-07-31 86680728811_10152638605648812 678 0 0 0 0 0 0 6/8/14 9:24 678 0014-08-06 86680728811_10152981635443812 3484 0 0 0 0 0 0 16/12/14 3:34 3484 0014-12-16 86680728811_10153511140788812 2093 1 0 0 0 0 0 16/6/15 15:55 2094 0015-06-16 86680728811_10153511641463812 2088 0 0 0 0 0 0 16/6/15 19:56 2088 0015-06-16 time 08:08 10:48 09:24 03:34 15:55 19:56</pre>
<b>Explanation</b>	<p>In the above code we're setting the class type of the new column date as type date and giving R the current format of the data in the cell which is d/m/y h:m. Likewise, using the lubridate library's as_datetime function we're converting the data from posted_at to date-time in the format d/m/y h:m and using format function to extract the h:m values to the new column time.</p>

<b>Code</b>	<pre>filter_abc_news\$weekdays = weekdays(filter_abc_news\$date) head(filter_abc_news)</pre>
<b>Output</b>	<pre>post_id likes_count love_count wow_count haha_count sad_count thankful_count angry_count posted_at total_reactions date 86680728811_10152623555133812 1149 0 0 0 0 0 0 31/7/14 8:08 1149 0014-07-31 86680728811_10152623774158812 1348 0 0 0 0 0 0 31/7/14 10:48 1348 0014-07-31 86680728811_10152638605648812 678 0 0 0 0 0 0 6/8/14 9:24 678 0014-08-06 86680728811_10152981635443812 3484 0 0 0 0 0 0 16/12/14 3:34 3484 0014-12-16 86680728811_10153511140788812 2093 1 0 0 0 0 0 16/6/15 15:55 2094 0015-06-16 86680728811_10153511641463812 2088 0 0 0 0 0 0 16/6/15 19:56 2088 0015-06-16 time weekdays 08:08 Thursday 10:48 Thursday 09:24 Wednesday 03:34 Tuesday 15:55 Tuesday 19:56 Tuesday</pre>
<b>Explanation</b>	<p>In this code we're using the weekdays method to populate the rows of the new column weekdays with the day derived from the date column.</p>

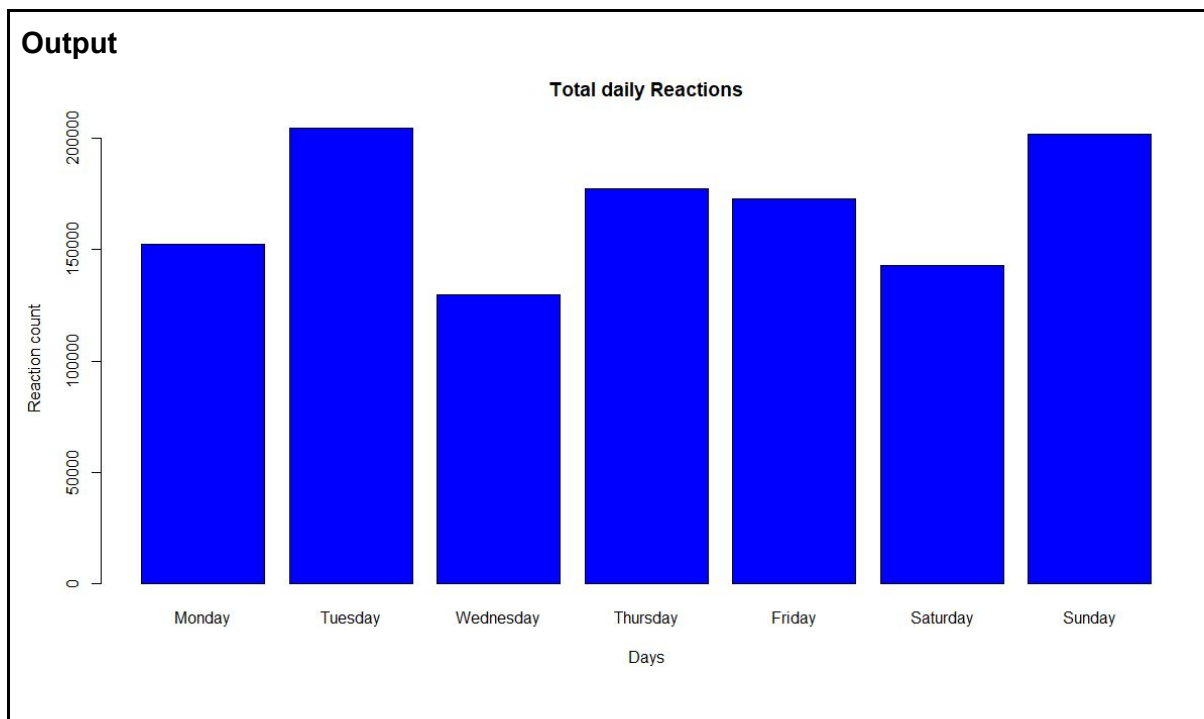
<b>Code</b>	<p><a href="https://stackoverflow.com/questions/1660124/how-to-sum-a-variable-by-group">https://stackoverflow.com/questions/1660124/how-to-sum-a-variable-by-group</a></p> <pre>weekdays_treaction = aggregate(filter_abc_news\$total_reactions, by=list(Weekdays=filter_abc_news\$weekdays),FUN=sum) data.frame(weekdays_treaction)</pre>
<b>Output</b>	<pre>weekdays      x 1  Friday 172716 2  Monday 152459 3 Saturday 142866 4   Sunday 201967 5 Thursday 177277 6  Tuesday 204462 7 Wednesday 129759</pre>
<b>Explanation</b>	<p>In the above code we use aggregate function to create subsets of unique</p>

	values in the weekdays column and apply the function sum to the data in the total_reactions column of the to all the data subsets created. We're then saving the result to a new dataframe called weekdays_reactions.
--	---

<b>Code</b>	<pre>names(weekdays_reaction)[names(weekdays_reaction) == "x"]="total_reaction"  data.frame(weekdays_reaction)</pre>
<b>Output</b>	<pre>  weekdays total_reaction 1   Friday      172716 2   Monday      152459 3  Saturday      142866 4    Sunday      201967 5  Thursday      177277 6   Tuesday      204462 7  Wednesday      129759</pre>
<b>Explanation</b>	This code is used to rename the column of the weekdays_reaction data frame from "x" to "total reaction".

<b>Code</b>	<p><a href="https://stackoverflow.com/questions/10309564/reorder-factor-levels-by-day-of-the-week-in-r">https://stackoverflow.com/questions/10309564/reorder-factor-levels-by-day-of-the-week-in-r</a></p> <pre>weekdays_reaction\$Weekdays = ordered(weekdays_reaction\$Weekdays, levels=c("Monday", "Tuesday", "Wednesday", "Thursday",           "Friday", "Saturday", "Sunday"))</pre>
<b>Explanation</b>	In the above code we're using the ordered function which will treat the column Weekdays as an ordered factor and print the results of operations applied in the order that we want. For instance, after running the above code the bar-graph created next will have the order mentioned in the code i.e. Monday, Tuesday, Wednesday, Thursday, Friday, Saturday and Sunday.

<b>Code</b>	<pre>barplot(weekdays_reaction\$total_reaction~weekdays_reaction\$W eekdays, col="blue",         main="Total daily Reactions",         xlab="Days",         ylab="Reaction count" )</pre>
<b>Explanation</b>	This code is used to create a bar graph of total_reaction column against Weekdays column with the appropriate labels.



**11.** From the above graph we can decipher that the users have shown the most reactions to the posts on Tuesday and Sunday which is about 200,000 for both days. On weekdays, Tuesday had the highest total reactions while Wednesday had the lowest total reaction count. During the weekdays we can see that the total number of reactions mostly were greater than 150,000, except for on Wednesday. During the weekend, Saturday we see a total number of reactions less than 150,000. However, we do see a high number of reactions to posts on Sunday. Since both weekdays and weekends have days with higher reaction count and lower reaction count and don't show any drastic difference in the number of reactions, we can say that the number of reactions during individual weekends and weekdays are fairly similar.

**12.**

From the graph we see that Tuesday and Sunday are the days where users have shown the most reaction. Therefore, to visualize the data for these days, a subset of the dataset called abc\_news\_d.csv was created.

<b>Code</b>	<pre> #creating a dataset where weekday is Tuesday tuesday_tr = filter_abc_news[(filter_abc_news\$weekdays=="Tuesday"),] head(tuesday_tr)  #creating a dataset where weekday is Sunday sun_tr = filter_abc_news[(filter_abc_news\$weekdays=="Sunday"),] head(sun_tr) </pre>
-------------	---



## Output

```
> head(tuesday_tr)
      post_id likes_count love_count wow_count haha_count sad_count thankful_count angry_count posted_at total_reactions
17120 86680728811_10152981635443812      3484         0         0         0         0         0         0 16/12/14 3:34      3484
20920 86680728811_1015351140788812      2093         1         0         0         0         0         0 16/6/15 15:55      2094
20928 86680728811_10153511641463812      2088         0         0         0         0         0         0 16/6/15 19:56      2088
21760 86680728811_10153617708413812      4249         0         0         0         0         0         0 21/7/15 18:11      4249
22111 86680728811_10153654255963812         775         0         0         0         0         0         0  4/8/15 22:11        775
22660 86680728811_10153714884588812     27627         5         2         2         0         0         0 11 25/8/15 23:14     27647

      date time weekdays
17120 0014-12-16 03:34 Tuesday
20920 0015-06-16 15:55 Tuesday
20928 0015-06-16 19:56 Tuesday
21760 0015-07-21 18:11 Tuesday
22111 0015-08-04 22:11 Tuesday
22660 0015-08-25 23:14 Tuesday

> head(sun_tr)
      post_id likes_count love_count wow_count haha_count sad_count thankful_count angry_count posted_at total_reactions
21578 86680728811_10153591677273812      8395         0         0         0         0         0         0 12/7/15 3:28      8395
21730 86680728811_10153610911963812      5324         0         0         0         0         0         0 19/7/15 13:19      5325
21731 86680728811_1015361042723812      2252         0         0         0         0         0         0 19/7/15 13:47      2252
22060 86680728811_10153648302138812      2337         0         0         0         0         0         0  2/8/15 17:43      2337
22425 86680728811_10153687284818812         554         0         0         0         0         0         0 16/8/15 13:36        554
23134 86680728811_10153767063873812      2060         0         0         0         0         0         0 13/9/15 13:10      2060

      date time weekdays
21578 0015-07-12 03:28 Sunday
21730 0015-07-19 13:19 Sunday
21731 0015-07-19 13:47 Sunday
22060 0015-08-02 17:43 Sunday
22425 0015-08-16 13:36 Sunday
23134 0015-09-13 13:10 Sunday
```

## Explanation

The above code is used to create a subsets of the dataframe filter\_abc\_news called tuesday\_tr and sun\_tr where we're filtering the rows by the word "Tuesday" and "sunday" respectively from the weekdays column and rendering all the columns for the matching rows to the new subset.

## Code

```
#adding a column hours to the tuesday_tr subset
tuesday_tr$hours = format(as_datetime(tuesday_tr$posted_at,
                                     "%d/%m/%Y %H:%M"), format = "%H")

head(tuesday_tr)

#adding a column hours to the sun_tr subset
sun_tr$hours = format(as_datetime(sun_tr$posted_at,
                                  "%d/%m/%Y %H:%M"),
                      format = "%H")

head(sun_tr)
```

## Output

```
> head(tuesday_tr)
      post_id likes_count love_count wow_count haha_count sad_count thankful_count angry_count posted_at total_reactions
17120 86680728811_10152981635443812      3484         0         0         0         0         0         0 16/12/14 3:34      3484
20920 86680728811_1015351140788812      2093         1         0         0         0         0         0 16/6/15 15:55      2094
20928 86680728811_10153511641463812      2088         0         0         0         0         0         0 16/6/15 19:56      2088
21760 86680728811_10153617708413812      4249         0         0         0         0         0         0 21/7/15 18:11      4249
22111 86680728811_10153654255963812         775         0         0         0         0         0         0  4/8/15 22:11        775
22660 86680728811_10153714884588812     27627         5         2         2         0         0         0 11 25/8/15 23:14     27647

      date time weekdays hours
17120 0014-12-16 03:34 Tuesday 03
20920 0015-06-16 15:55 Tuesday 15
20928 0015-06-16 19:56 Tuesday 19
21760 0015-07-21 18:11 Tuesday 18
22111 0015-08-04 22:11 Tuesday 22
22660 0015-08-25 23:14 Tuesday 23

> head(sun_tr)
      post_id likes_count love_count wow_count haha_count sad_count thankful_count angry_count posted_at total_reactions
21578 86680728811_10153591677273812      8395         0         0         0         0         0         0 12/7/15 3:28      8395
21730 86680728811_10153610911963812      5324         0         0         0         0         0         0 19/7/15 13:19      5325
21731 86680728811_1015361042723812      2252         0         0         0         0         0         0 19/7/15 13:47      2252
22060 86680728811_10153648302138812      2337         0         0         0         0         0         0  2/8/15 17:43      2337
22425 86680728811_10153687284818812         554         0         0         0         0         0         0 16/8/15 13:36        554
23134 86680728811_10153767063873812      2060         0         0         0         0         0         0 13/9/15 13:10      2060

      date time weekdays hours
21578 0015-07-12 03:28 Sunday 03
21730 0015-07-19 13:19 Sunday 13
21731 0015-07-19 13:47 Sunday 13
22060 0015-08-02 17:43 Sunday 17
22425 0015-08-16 13:36 Sunday 13
23134 0015-09-13 13:10 Sunday 13
```



	In the above code fragment, we're adding a column called hours to the dataframe tuesday_tr and sun_tr by converting the data in the posted_at to date-time and extracting hours from the posted_at column to the new column hours.
--	--

<b>Code</b>	<pre>#changing the class of hours to integer for both subsets class(tuesday_tr\$hours) &gt; class(tuesday_tr\$hours) [1] "character"  tuesday_tr\$hours= as.integer(tuesday_tr\$hours) class(tuesday_tr\$hours) &gt; class(tuesday_tr\$hours) [1] "integer"  sun_tr\$hours= as.integer(sun_tr\$hours) class(sun_tr\$hours) &gt; class(sun_tr\$hours) [1] "integer"</pre>
<b>Explanation</b>	In the above code we're changing the class of the hours column from character to integer by using as.integer method .

<b>Code</b>	<pre>#aggregating hours column from tuesday_tr tues_h_treaction = aggregate(tuesday_tr\$total_reactions,                              by=list(hours=tuesday_tr\$hours), FUN=sum)  names(tues_h_treaction)[names(tues_h_treaction) == "x"]="total_reaction"  head(tues_h_treaction,10)  #aggregating hours column from sun_tr sun_h_treaction = aggregate(sun_tr\$total_reactions,                              by=list(hours=sun_tr\$hours),                              FUN=sum) names(sun_h_treaction)[names(sun_h_treaction) == "x"]="total_reaction" head(sun_h_treaction,10)</pre>
<b>Output</b>	<pre>&gt; head(tues_h_treaction,10) &gt; head(sun_h_treaction,10)   hours total_reaction  hours total_reaction 1     0         21327  1     0             544 2     1           640  2     1          1370 3     2         14190  3     2         14747 4     3          5694  4     3          8395 5     4          2280  5     4          3123 6     5          1815  6     5         16360 7     6          4000  7     6          1319 8     11           147  8     7         28178 9     12          8767  9     8          2778 10    13          1441 10    9         31207</pre>

<b>Explanation</b>	In the above code we're aggregating the rows by the unique values in the hour column and applying a function sum to return a summation value of all the data under the column total_reaction for a unique hour and saving the results as a new dataframe called tues_h_treaction and sun_h_treaction .
--------------------	--

Looking at the output of the head we can see that there are hours in which there were no posts being reacted to, therefore, to add those missing hours, a df with values from 0 to 24 was created and joined to the dataframe tues\_h\_reaction and sun by using the function full\_join from the library dplyr.

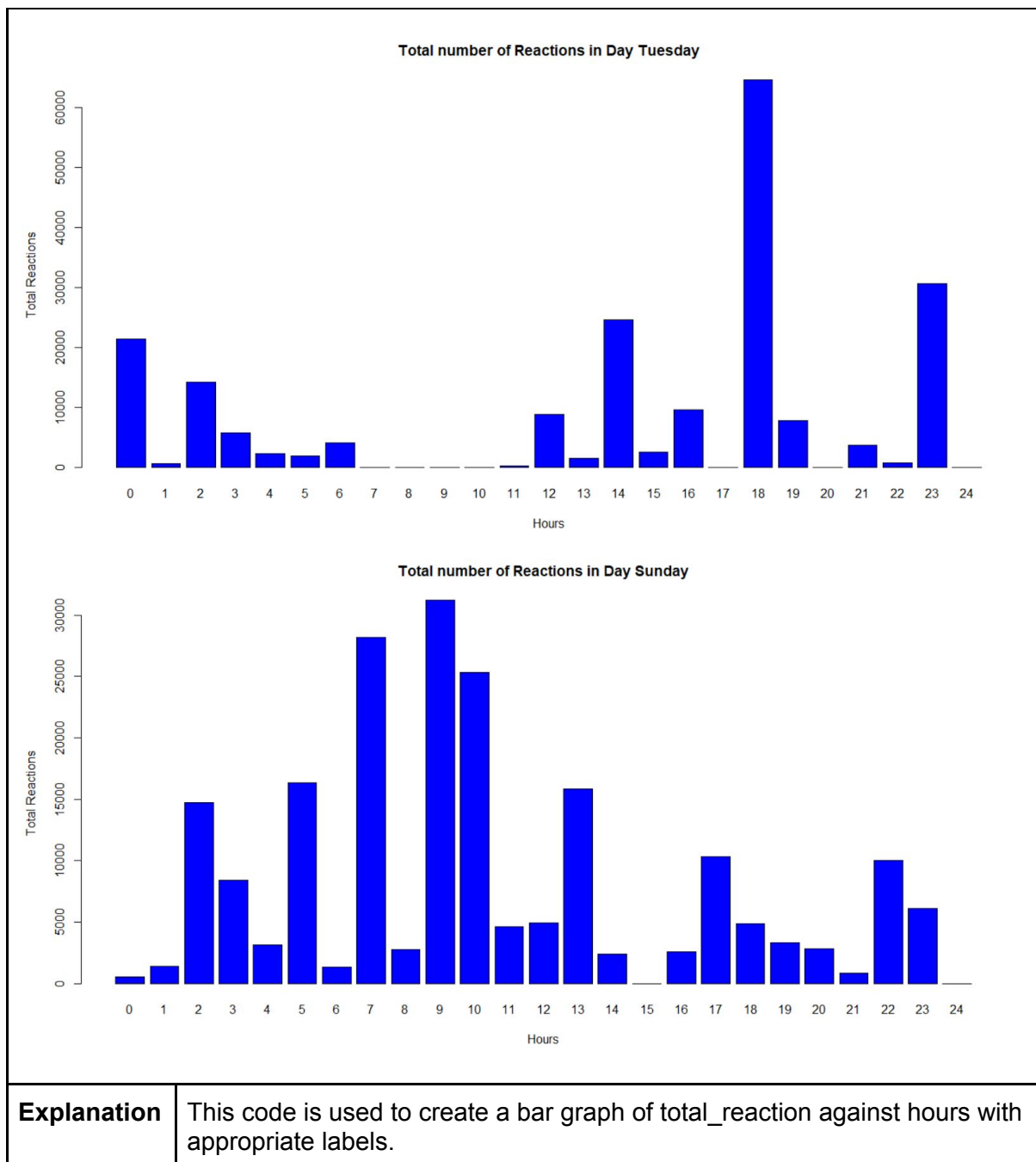
<b>Code</b>	<pre>df = data.frame(hours =0:24) head(df)  class(df\$hours)</pre>
<b>Output</b>	<pre>&gt; head(df)   hours 1     0 2     1 3     2 4     3 5     4 6     5  &gt; class(df\$hours) [1] "integer"</pre>
<b>Explanation</b>	This code is used to create a dataframe with one column called hours with values from 0 to 24 and check the class of the integer.

<b>Code</b>	<pre>install.packages("dplyr") library(dplyr)  final_tues_tr = full_join(df,tues_h_treaction) head(final_tues_tr,10)  final_sun_tr = full_join(df,sun_h_treaction) tail(final_sun_tr,10)</pre>
<b>Output</b>	<pre>&gt; head(final_tues_tr,10)   hours total_reaction 1     0          21327 2     1           640 3     2          14190 4     3           5694 5     4           2280 6     5           1815 7     6          4000 8     7            NA 9     8            NA 10    9            NA  &gt; tail(final_sun_tr,10)   hours total_reaction 16    15             NA 17    16           2567 18    17          10344 19    18           4875 20    19           3317 21    20           2819 22    21            854 23    22          10043 24    23           6095 25    24             NA</pre>

<b>Explanation</b>	This code is used to join two dataframes <code>tues_h_reactions</code> and <code>sun_h_treaction</code> with the dataframe <code>df</code> by the column <code>hours</code> and have the rows and columns as a new dataframe <code>final_tues_tr</code> and <code>final_sun_tr</code> .
--------------------	---

Code	<pre>final_tues_tr[is.na(final_tues_tr)] = 0 head(tues_h_treaction,10)  final_sun_tr[is.na(final_sun_tr)] = 0 tail(final_sun_tr,10)</pre>																																																																		
Output	<pre>&gt; head(final_tues_tr,10) &gt; tail(final_sun_tr,10)</pre> <table><thead><tr><th></th><th>hours</th><th>total_reaction</th><th></th><th>hours</th><th>total_reaction</th></tr></thead><tbody><tr><td>1</td><td>0</td><td>21327</td><td>16</td><td>15</td><td>0</td></tr><tr><td>2</td><td>1</td><td>640</td><td>17</td><td>16</td><td>2567</td></tr><tr><td>3</td><td>2</td><td>14190</td><td>18</td><td>17</td><td>10344</td></tr><tr><td>4</td><td>3</td><td>5694</td><td>19</td><td>18</td><td>4875</td></tr><tr><td>5</td><td>4</td><td>2280</td><td>20</td><td>19</td><td>3317</td></tr><tr><td>6</td><td>5</td><td>1815</td><td>21</td><td>20</td><td>2819</td></tr><tr><td>7</td><td>6</td><td>4000</td><td>22</td><td>21</td><td>854</td></tr><tr><td>8</td><td>7</td><td>0</td><td>23</td><td>22</td><td>10043</td></tr><tr><td>9</td><td>8</td><td>0</td><td>24</td><td>23</td><td>6095</td></tr><tr><td>10</td><td>9</td><td>0</td><td>25</td><td>24</td><td>0</td></tr></tbody></table>		hours	total_reaction		hours	total_reaction	1	0	21327	16	15	0	2	1	640	17	16	2567	3	2	14190	18	17	10344	4	3	5694	19	18	4875	5	4	2280	20	19	3317	6	5	1815	21	20	2819	7	6	4000	22	21	854	8	7	0	23	22	10043	9	8	0	24	23	6095	10	9	0	25	24	0
	hours	total_reaction		hours	total_reaction																																																														
1	0	21327	16	15	0																																																														
2	1	640	17	16	2567																																																														
3	2	14190	18	17	10344																																																														
4	3	5694	19	18	4875																																																														
5	4	2280	20	19	3317																																																														
6	5	1815	21	20	2819																																																														
7	6	4000	22	21	854																																																														
8	7	0	23	22	10043																																																														
9	8	0	24	23	6095																																																														
10	9	0	25	24	0																																																														
Explanation	This code is used to replace the missing values in the column total_reaction with 0s.																																																																		

<b>Code</b>	<pre>#bar graph of total _reations against hours for Tuesday barplot(final_tues_tr\$total_reaction~final_tues_tr\$hours, col="blue",       main="Total number of Reactions in Day Tuesday",       xlab="Hours",       ylab="Total Reactions" )  #bar graph of total _reations against hours for Sunday barplot(final_sun_tr\$total_reaction~final_sun_tr\$hours, col="blue",       main="Total number of Reactions in Day Sunday",       xlab="Hours",       ylab="Total Reactions" )</pre>
<b>Output</b>	



From the two graphs above, we can deduce that on Tuesday most reactions happened around 18 pm of the day i.e around 60,000 reactions and on Sunday users reacted most to the posts at 9am which is around 30,000 reactions. Looking at the peaks of the two graphs we can see that on Tuesday the highest peak is double the peak of Sundays graph.

On these two days, the hours when a high number of users react to the posts vary from one another. On Tuesday there are many hours where users haven't reacted to any posts, for instance, in hour 7, 8, 9 and 10. This might be due to the fact that users are mostly busy getting ready for work and on the way to their work. Whereas for Sunday, we can see that users react to posts almost throughout the day. Also, on Sunday we see a high number of reactions from users in the hour 7, 9 and 10 whereas on Tuesday no users reacted to any posts during that time. Similarly on hours where there are high numbers of reactions from

users on Tuesday we see a lower number of reactions during those hours on Sunday. For example, on 14 and 18th hour we see around 20,000 and 60,000 reactions respectively on Tuesday while on Sunday on the 14th and 18th hour less than 500 reactions were given to the posts by users.

**13.**

**a.**

Tuesday is the day and hour 18 (6:00pm -6:59pm) is the time which has the maximum number of reactions.

**b.**

Yes, I do think it's a good idea publishing general posts about Trump on Tuesdays and Sundays as they have moderately higher reaction counts than the rest of the days which can be identified from the graph in Q10. The higher number of reactions also suggests that there are a higher number of users online and viewing posts on "abc-news" page during those days.

On Tuesday we see about 60,000 reactions on the peak hour which is in hour 18 (6:00pm-6:59pm). The high peak at this hour may suggest users getting off work and checking their social media therefore if users follow the "abc-news" page this would be a good hour to post the general post about Trump as this increases the likelihood of viewing the post on their home page. On Sundays, if the post is released around 9 am it is likely that the post will receive the highest level of engagement they can on that day as the graph shows in Q12. This might be because it's a day off for most people thus, they don't need to go to work and might start off their day by checking their social media platforms.