Name: Bryan Jun Kit Wong
Student ID: 32882424

Task A

A1

```
mkdir dataset_TIST2015-001
```

Create a new empty folder named dataset_TIST2015-001 to store the extracted files into

```
tar -xvf dataset_TIST2015-001.tar -C dataset_TIST2015-001
```

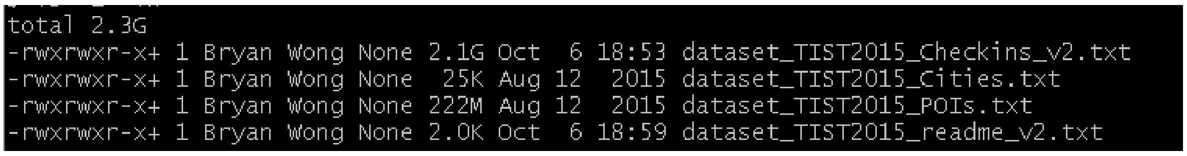
Extract the .tar file and place the extracted files into the folder dataset_TIST2015-001

```
cd dataset_TIST2015-001
```

Move the current directory into the dataset_TIST2015-001 folder to access the files within it

```
ls -1 -lh
```

List out all the files inside the dataset_TIST2015-001 folder. -1 to display each file in a new line in the console and -lh is to display individual sizes of each file.



```
total 2.3G
-rwxrwxr-x+ 1 Bryan Wong None 2.1G Oct  6 18:53 dataset_TIST2015_Checkins_v2.txt
-rwxrwxr-x+ 1 Bryan Wong None  25K Aug 12  2015 dataset_TIST2015_Cities.txt
-rwxrwxr-x+ 1 Bryan Wong None 222M Aug 12  2015 dataset_TIST2015_POIs.txt
-rwxrwxr-x+ 1 Bryan Wong None 2.0K Oct  6 18:59 dataset_TIST2015_readme_v2.txt
```

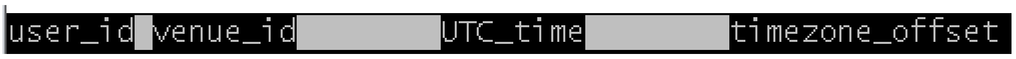
As shown in the output above, there are a total of 4 files extracted from dataset_TIST2015-001.tar, which are dataset_ TIST2015_Checkins_v2.txt, dataset_ TIST2015_Cities.txt, dataset_ TIST2015_POIs.txt and dataset_ TIST2015_readme_v2.txt with a size of 2.1GB, 25KB, 222MB and 2KB respectively.

A2

```
head -1 dataset_TIST2015_Checkins_v2.txt | less
```

Get the first line of the .txt file into the terminal and pipe the output to less to navigate through it.

```
/\t
```



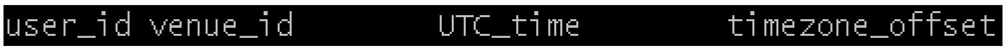
```
user_id venue_id        UTC_time        timezone_offset
```

The command `/\t` above will find all patterns that match `\t`, which is the tab character. Then it will highlight everything that fits the pattern. In this case, the tabs are highlighted between each column, thus showing that tabs are used as the delimiter. Furthermore, as shown in the output above, there are 4 columns in the dataset_TIST2015_Checkins_v2.txt file.

A3

```
head -1 dataset_TIST2015_Checkins_v2.txt
```

Get the first line of the .txt file which are the column names and print them to terminal.

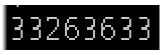


```
user_id venue_id        UTC_time        timezone_offset
```

The other 3 column names are venue_id, UTC_time and timezone_offset.

A4

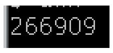
```
awk -F '\t' 'NR > 1' dataset_TIST2015_Checkins_v2.txt | wc -l
```



```
33263633
```

The awk script is used to skip the column header names and start from line 2. wc prints out the word count of the file while –l prints out the number of lines. There are 33263633 Checkins as shown in the output displayed above.

```
awk -F '\t' 'NR > 1 {print $1}'
dataset_TIST2015_Checkins_v2.txt | sort -n | uniq | wc -l
```



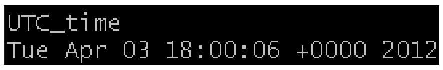
```
266909
```

The awk script will also skip the column headers. Then it will sort the first column, which is the user_id column, and then pipes it into the uniq command to display all unique user ids. Then it will count the number of lines (unique user ids). There are a total of 266909 unique users.

Name: Bryan Jun Kit Wong
Student ID: 32882424

A5

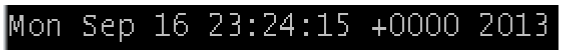
```
cut -f 3 dataset_TIST2015_Checkins_v2.txt | head -2
```



```
UTC_time
Tue Apr 03 18:00:06 +0000 2012
```

The code will remove the tab delimiters and will output the rows in the UTC_time column. Then it will pipe the output and print the first 2 rows, which are the column name and the first row respectively. In this case, the first date is 3rd of April 2012, Tuesday.

```
cut -f 3 dataset_TIST2015_Checkins_v2.txt | tail -1
```

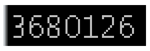


```
Mon Sep 16 23:24:15 +0000 2013
```

The code above will do the same as the previous code with the exception of piping it to the tail command to print the last row of the column. The last date is 16th of September 2013, Monday.

A6

```
cut -f 1 dataset_TIST2015_POIs.txt | sort | uniq | wc -l
```

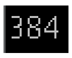


```
3680126
```

The code above will check the venue column, which is the first column, and then sort it. Then it will pipe and output all unique venues. The last pipe will count the amount of lines outputted (unique venues). There are 3680126 unique venues.

A7

```
awk -F '\t' '$5 ~ /FR/ {print $4}' dataset_TIST2015_POIs.txt |
sort | uniq | wc -l
```



```
384
```

Name: Bryan Jun Kit Wong
Student ID: 32882424

An awk script is used to check the fifth column for venues located in France. The 4th column is then outputted and piped into the next command to count the unique venues located in France. There are 384 unique venues in France.

A8

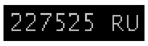
A.

```
awk -F '\t' '$5 ~
/AT|BE|BG|BY|CH|CZ|DE|DK|EE|ES|FI|FR|GB|GR|HU|IE|IT|LV|NL|PL|P
T|RO|RU|SE|UA/' dataset_TIST2015_POIs.txt > POIeu.txt
```

The above command will search through the venue category column (the fifth column) in the dataset_TIST2015_POIs.txt and will pattern match every country code that meets the above criteria. In this case, the country codes listed are countries in Europe. If their fifth column matches the country code listed above, the entire row will be printed. The `’> POIeu.txt’` will capture the printed output and is then stored in a newly created text file named POIeu.txt.

B.

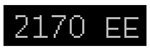
```
cut -f 5 POIeu.txt | sort | uniq -c | sort | tail -1
```



```
227525 RU
```

The code will find the country with the most venues. First it counts the amount of venues held in each country before sorting them in ascending order. Then it is piped into `‘tail -1’` which will output the last row. Here, it will output the country with the most venues, which is Russia at 227525 venues.

```
cut -f 5 POIeu.txt | sort | uniq -c | sort | head -1
```



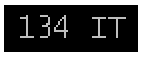
```
2170 EE
```

Similar to the previous code, this will count the amount of venues in each country before sorting them in ascending order. What’s different however, is `’head -1’` will print out the first row, which is the country with the least amount of venues. Estonia has 2170 venues, the least in all of Europe.

Name: Bryan Jun Kit Wong
Student ID: 32882424

C.

```
awk -F '\t' '$4 ~ /Seafood Restaurant/' POIeu.txt | cut -f 5 |
sort | uniq -c | sort | tail -1
```

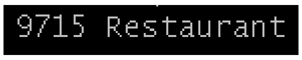


```
134 IT
```

An awk script will be used to search through the fourth column for every Seafood Restaurant. The script will only select rows that contain the pattern "Seafood Restaurant" in the 4th column. Then the output is piped into `'cut -f 5'` to only select the country codes. Then they will be sorted and counted to output the amount of Seafood Restaurants for each country. Then they will be sorted in ascending order and `'tail -1'` will output the country which has the most Seafood Restaurants. Italy has the most Seafood Restaurants in Europe (134 Seafood Restaurants).

D.

```
awk -F '\t' '$4 ~ /Restaurant/' POIeu.txt | cut -f 4 | sort |
uniq -c | sort | tail -1
```



```
9715 Restaurant
```

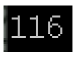
Similar to the code in C, an awk script is used to search through every venue category and find the keyword "Restaurant" inside the POIeu.txt. Then every unique Restaurant class will be counted and sorted in ascending order. Here, the most common Restaurant class in Europe is "Restaurant" with 9715 venues all over Europe

Name: Bryan Jun Kit Wong
Student ID: 32882424

Task B

B1

```
cat Twitter_Data_1.gz | gunzip | cut -f 4 | grep -o "Donald
Trump" | wc -l
```



```
116
```

The above output will read the .gz file, pipe it into gunzip to decompress then check through the tweet column. `grep -o "Donald Trump"` is use to output all occurrences of Donald Trump into a new line each. Then `wc -l` counts the number of Donald Trumps. In this case, there are 116 instances the term “Donald Trump” was used in tweets.

B2

```
cat Twitter_Data_1.gz | gunzip | awk -F '\t' 'BEGIN {print
“Timestamp”} $4 ~ /Donald Trump/ {print $3}' >
Timestamps_of_Tweets.txt
```

The code above will search for lines in which the tweet column has the pattern “Donald Trump”. If a pattern is found, the awk script will print out their respective timestamp. The output is captured and outputted into a .txt file named Timestamps_of_Tweets.txt. The BEGIN will put Timestamp as the column header before reading the file.

```
tweet_timestamps <- read.csv("Timestamps_of_Tweets.txt")
```

Read the Timestamps_of_Tweets.txt file and assign it to variable tweet_timestamps.

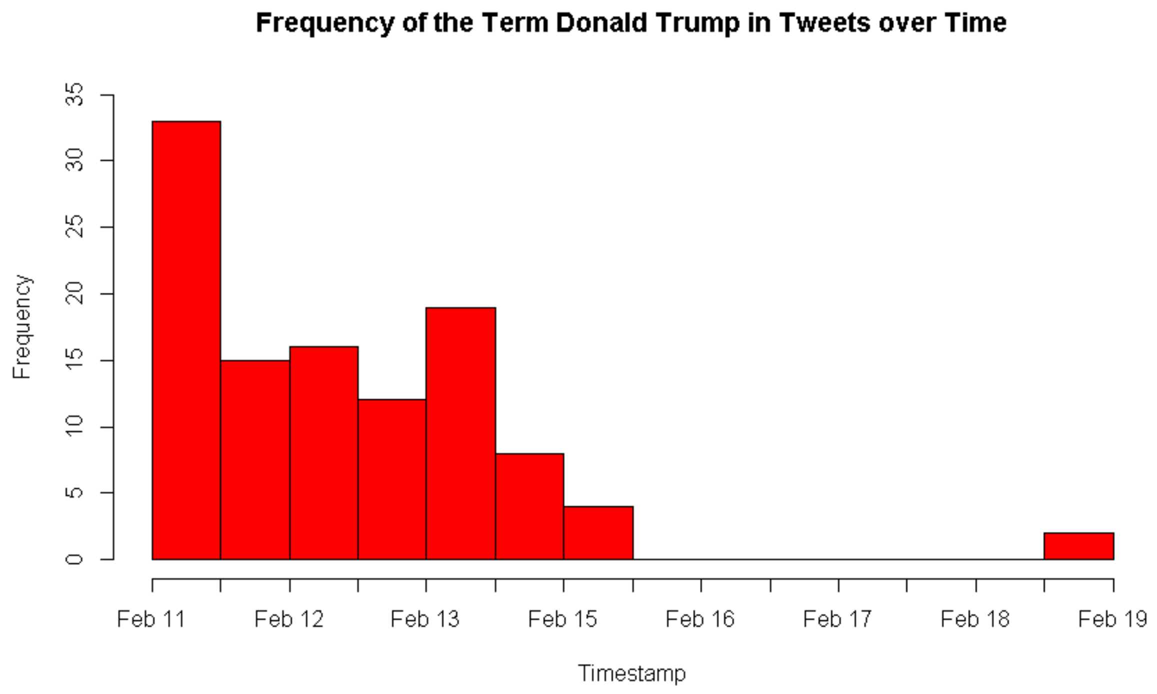
```
new_timestamps <- strptime(tweet_timestamps$Timestamp,
format="%a %b %d %T %z %Y", tz = "UTC")
```

Convert the whole column of strings into the timestamp format using strptime. %a is for shorten day, %b is for shorten month, %d is for decimal day from 1 – 31, %T is for time in Hours:Minutes:Seconds format, %z is for time zone offset from UTC and %Y is the year including the century. Assign the new data frame to new_timestamps variable.

Name: Bryan Jun Kit Wong
Student ID: 32882424

B3

```
hist(new_timestamps, col = "red", main = "Frequency of the
Term Donald Trump in Tweets over Time", xlab = "Timestamp",
"ylab = Frequency", ylim = c(0, 35), freq = TRUE, breaks = 2
```



The code above creates a histogram with the X-axis being the Timestamp/Date and the Y-axis being the Frequency of Donald Trump appearing in Tweets. The y-limit is from 0 to 35. The freq parameter will count for freq of tweets if set to TRUE.

B4

The pattern before February 15 is a general downward trend from Feb 11 having more than 33 mentions of Donald Trump with a sudden increase in February 13 before decreasing to below 5 mentions on February 15. Then there are 0 mentions of Donald Trump until February 19 where there is an unusual and small spike of Donald Trump related tweets.

Name: Bryan Jun Kit Wong
Student ID: 32882424

B5

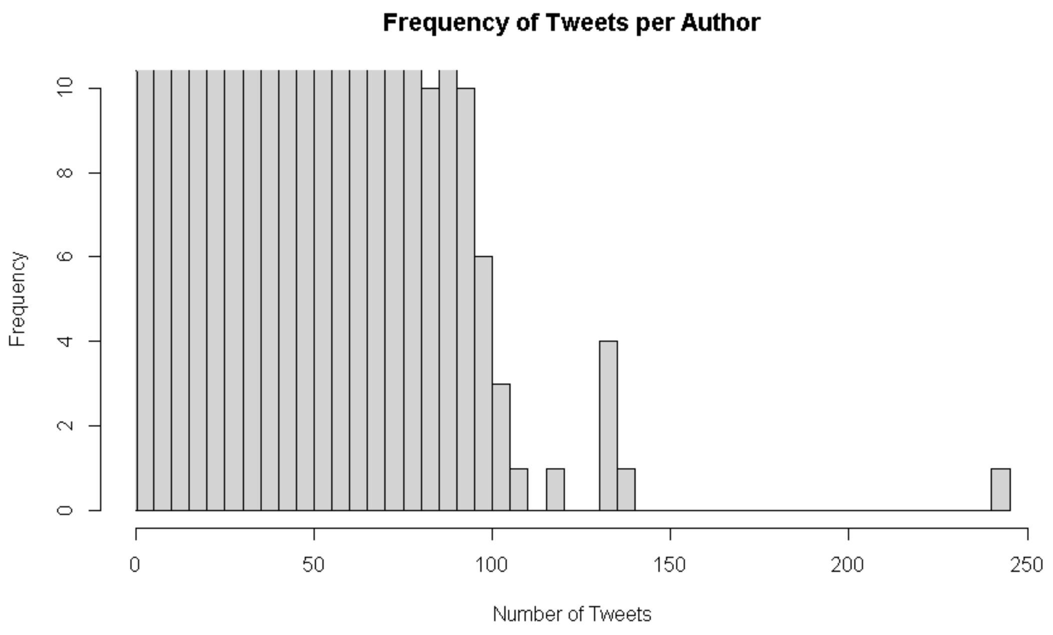
```
cat Twitter_Data_1.gz | gunzip | cut -f 2 | sort | uniq -c |
awk -F ' ' 'BEGIN {printf "user twitter_count\n"} NR > 1
{print $2, $1}' | tr ' ' '\t' | tr "twitter_user" "twitter
user" > Number_of_Tweets_per_Author.txt
```

The code above will check through the tweets' author column and count the amount of unique authors. This will output a table of author and their amount of tweets. Then, add in the column header on the top of the file and rename the delimiter from spaces to tabs using `'tr ' ' '\t''`. The output is captured and stored in a newly created .txt file.

```
author_tweets <- read.csv("Number_of_Tweets_per_Author.txt",
sep = "\t", header = TRUE)
```

Create the data frame using the .txt file created before. The parameter 'header' is used so that the read.csv will skip the first line and treat it as the column headers. 'sep' is used to explicitly state the delimiter of the table. In this case, the delimiter is tabs. Store the created data frame in a variable author_tweets.

```
hist(author_tweets$twitter.count, main = "Frequency of Tweets
per Author", xlab = "Number of Tweets", freq = TRUE, breaks =
50, ylim = c(0, 10))
```



Name: Bryan Jun Kit Wong
Student ID: 32882424

The code plots a histogram of the Number of Tweets and their Frequency. The graph is limited to the top 10 highest number of tweets. In this case, it’s 0 – 100 number of tweets. As the number of tweets increases, the frequency decreases as a general downward trend can be observed.