

Assignment 3

Rui Qin 30874157

Task A

We first enter the folder which contains the Task A data file.

```
cd C:
cd A3/TaskA
```

1)

We using tar -xf to decompress the tar file

```
tar -xf dataset_TIST2015.tar
```

Then we list the document with 'ls'

```
$ ls
dataset_TIST2015.tar          dataset_TIST2015_POIs.txt
dataset_TIST2015_Checkins_v2.txt  dataset_TIST2015_readme_v2.txt
dataset_TIST2015_Cities.txt
```

There are **4 txt files** in the tar. Now we can check the size of them with -lh, it will print the size and time it created.

```
Qr140@AllEN /cygdrive/c/A3/TaskA
$ ls -lh dataset_TIST2015_POIs.txt
-rwxrwxr-x+ 1 Qr140 Qr140 22M Aug 12 2015 dataset_TIST2015_POIs.txt

Qr140@AllEN /cygdrive/c/A3/TaskA
$ ls -lh dataset_TIST2015_Checkins_v2.txt
-rwxrwxr-x+ 1 Qr140 Qr140 2.1G Oct 6 21:53 dataset_TIST2015_Checkins_v2.txt

Qr140@AllEN /cygdrive/c/A3/TaskA
$ ls -lh dataset_TIST2015_readme_v2.txt
-rwxrwxr-x+ 1 Qr140 Qr140 2.0K Oct 6 21:59 dataset_TIST2015_readme_v2.txt

Qr140@AllEN /cygdrive/c/A3/TaskA
$ ls -lh dataset_TIST2015_Cities.txt
-rwxrwxr-x+ 1 Qr140 Qr140 25K Aug 13 2015 dataset_TIST2015_Cities.txt
```

Based on the code we can know:

- dataset_TIST2015_POIs.txt is **22MB**
- dataset_TIST2015_Checkins_v2.txt is **2.1GB**
- dataset_TIST2015_readme_v2.txt is **2KB**
- dataset_TIST2015_Cities.txt is **25KB**

2)

In this part we can use:

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

To check how many columns and what is the delimiter. The output is:

```
user_id venue_id UTC_time timezone_offset
```

There are **4 columns** in this data frame. Then we use the search function with tab (/^I):

user_id	venue_id	UTC_time	timezone_offset
50756	4f5e3a72e4b053fd6a4313f6	Tue Apr 03 18:00:06	+0000 2012 240
190571	4b4b87b5f964a5204a9f26e3	Tue Apr 03 18:00:07	+0000 2012 180
221021	4a85b1b3f964a520eefe1fe3	Tue Apr 03 18:00:08	+0000 2012 -240
66981	4b4606f2f964a520751426e3	Tue Apr 03 18:00:08	+0000 2012 -300

All the spaces are filled with the highlight, which means the delimiter is **tab**.

3)

We can use the part 2 code to list out all columns:

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

output is:

```
user_id venue_id UTC_time timezone_offset
```

We can conclude the column name is:

- user_id
- venue_id
- UTC_time
- timezone_offset

4)

In this part we can use awk, sort, uniq with word count to calculate how many unique user values are in the file, and due to the delimiter is the tab, so we use -F '\t' after awk.

```
$ awk -F '\t' '{print $1}' dataset_TIST2015_Checkins_v2.txt | sort | uniq | wc -l
266910
```

Due to *user_id* can be counted as 1 in that column. so after subtracting 1, we can know there are **266909** unique users in this file.

Then we can use wc -l to check how many lines are in the file, and get the number of check-in.

```
$ wc -l dataset_TIST2015_Checkins_v2.txt
33263634 dataset_TIST2015_Checkins_v2.txt
```

Due to the title being counted in, so we have to subtract it with 1. We can see there are 33263633 checkin records.

5)

We can use *head -n2* to get extract the second line which starts counting from the head (The first line is the title). We want to get the time, so we know the information we want is from column 3.

```
$ awk -F '\t' '{print $3}' dataset_TIST2015_Checkins_v2.txt | head -n1
UTC_time
Tue Apr 03 18:00:06 +0000 2012
```

We can see the first date is **Tue Apr 03 18:00:06 +0000 2012**

Then we extract the first line from the tail to get the last time

```
$ awk -F '\t' '{print $3}' dataset_TIST2015_Checkins_v2.txt | tail -n1
Mon Sep 16 23:24:15 +0000 2013
```

We can know the last date is **Mon Sep 16 23:24:15 +0000 2013**

6)

We can use the part 4 method to get the number of unique.

```
Qr140@AllEN /cygdrive/c/A3/TaskA
$ awk -F '\t' '{print $1}' dataset_TIST2015_POIs.txt | sort | uniq | wc -l
3680126

Qr140@AllEN /cygdrive/c/A3/TaskA
$ awk -F '\t' '{print $1}' dataset_TIST2015_POIs.txt | head -n1
3fd66200f964a52000e71ee3
```

We can see that there are **3680126** records, and the first row is not the title so we don't need to subtract one from the total.

7)

In this case, we can search the lines which are column 5 only contain FR with the condition **\$5=="FR"**, and then use **sort** and **uniq** to make sure the lines are unique.

```
$ awk -F '\t' ' $5 == "FR" ' dataset_TIST2015_POIs.txt | sort | uniq | wc -l
19837
```

Then we can get there are **19837** unique venue categories including France

8)

a)

We can know the Europe land area can be defined in the range 34° to 72° latitude and -25° to 45° longitude (European, 2020), and the readme points out that **the second column is latitude and third column is longitude**. We can base on this information to sort the data which contain European.

In this code, we sort columns 2 and 3 by using the conditions, and connect the condition with && which means AND in shell, then output the data in POIeu.txt.

```
$ (awk -F '\t' '$2 >= 34 && $2 <= 72 && $3 >= -25 && $3 <= 45'
dataset_TIST2015_POIs.txt) > POIeu.txt
```

b)

Explanation of this part:

- **{arr[\$5]++}**
increment array with column 5 as the key
- **END{for (element in arr) print element, arr[element]}**
The block executed at the end
 - In the block, we create a for loop and go through the elements in the array
 - Print the lines
 - Print the element first
 - Print the count number in the array, with the element as key
- **sort -k2 -n**
Sort the printout data
 - -k2 means we sort column 2 data
 - -n means we use a numeric sort

```
$ awk -F "\t" '{arr[$5]++}END{for (element in arr) print element,
arr[element]}' POIeu.txt | sort -k2 -n
LB 63
EE 2170
BG 2411
DK 2735
CH 2930
```

```

TN 3598
PL 3651
RO 3858
IE 3968
AT 5636
FI 5651
CZ 5707
SE 6389
BY 6693
CY 6804
LV 7924
HU 8681
PT 8721
GR 18259
FR 19837
UA 29276
IT 34332
DE 34713
BE 36826
NL 38536
ES 39187
GB 54278
RU 157378
TR 377302

```

Based on the output list, we can know **TR has the most venues and LB has the least**

c)

In this case what we need to do is add **\$4 == "Seafood Restaurant"** before the **{arr[\$5]++}**. In this way, we can firstly sort out the seafood restaurant, and then process our rest code.

```

awk -F '\t' '$4 == "Seafood Restaurant" {arr[$5]++}END{for (element in arr)
print element, arr[element]}' POIeu.txt | sort -k2 -n
PL 1
BY 2
CH 2
EE 2
FI 2
LB 3
LV 5
BG 6
CZ 6
DK 6
HU 6
RO 6
IE 7
TN 11
SE 15
AT 16
CY 25
UA 26
FR 39
PT 57
BE 63
RU 64
DE 76
NL 94
GB 108
GR 110
ES 123
IT 134
TR 1522

```

Based on the result, we can know **TR has the most seafood restaurants** in their country.

d)

We first using index function to create the condition: string include 'Restaurant' and put it at the front. In this case, we need to use the old method but with a change. We need to put frequency before the restaurant list because the length of the restaurant name is different.

So **arr[element]** is in front of the **element**, which means we print the frequency first and then print the restaurant name. Then we sort the first column with **-k1**.

```
$ awk -F '\t' 'index($4, "Restaurant") {arr[$4]++}END{for (element in arr)
print arr[element], element }' POIeu.txt | sort -k1 -n
23 Filipino Restaurant
27 Mongolian Restaurant
37 Peruvian Restaurant
51 Gluten-free Restaurant
53 Malaysian Restaurant
54 New American Restaurant
57 Southern / Soul Food Restaurant
67 Australian Restaurant
67 Indonesian Restaurant
72 Cajun / Creole Restaurant
77 Ethiopian Restaurant
89 South American Restaurant
95 Cuban Restaurant
95 Latin American Restaurant
96 Dim Sum Restaurant
126 Molecular Gastronomy Restaurant
130 Dumpling Restaurant
130 Paella Restaurant
137 Caribbean Restaurant
137 Swiss Restaurant
150 Moroccan Restaurant
179 Afghan Restaurant
181 Arepa Restaurant
185 Brazilian Restaurant
207 Korean Restaurant
313 Argentinian Restaurant
322 African Restaurant
326 Scandinavian Restaurant
338 Vietnamese Restaurant
422 Portuguese Restaurant
522 Vegetarian / Vegan Restaurant
580 Falafel Restaurant
754 Thai Restaurant
759 Mexican Restaurant
1096 German Restaurant
1292 American Restaurant
1371 Indian Restaurant
1456 Tapas Restaurant
1507 Greek Restaurant
1689 Eastern European Restaurant
1755 Japanese Restaurant
1911 Spanish Restaurant
2077 Mediterranean Restaurant
2124 Sushi Restaurant
2219 Chinese Restaurant
2414 Asian Restaurant
2537 Seafood Restaurant
2716 Middle Eastern Restaurant
2863 French Restaurant
7666 Italian Restaurant
8634 Fast Food Restaurant
10093 Turkish Restaurant
15208 Restaurant
```

Based on the result, we can know the most common restaurant is **Restaurant with no categories**, and the most common restaurant with the category is **Turkish restaurant**.

Reference in case 8:

European, Environment, Agency. (2020). *Global and European temperatures*. [Global and European temperatures — European Environment Agency \(europa.eu\)](https://www.euro.who.int/en/health-topics/communicable-diseases/news/news/2020/04/global-and-european-temperatures)

Task B

To unzip the .gz file we use gunzip.

```
$ gunzip Twitter_Data_1.gz
```

And we get a view of the data

```
head -n1 Twitter_Data_1
433213478539513856      TRY_Sound      Tue Feb 11 12:18:36 +0000 2014   また
たび食べると一時的に楽しくなるし、血行良くなるから頭痛も無くなるけど、覚めた後死ぬ。が食べる。うまい
```

This result suggests that:

- The data frame does not have the header
- The data looks like the record of a Twitter post
- We can assume:
 - The first part is ID
 - The second part is the name of the user
 - The third part is time
 - The fourth part is content

Then we find the delimiter, with the search up function (/^I).

```
head -1 Twitter_Data_1 | less
433213478539513856      TRY_Sound      Tue Feb 11 12:18:36 +0000 2014   また
たび食べると一時的に楽しくなるし、血行良くなるから頭痛も無くなるけど、覚めた後死ぬ。が食べる。うまい
```

The delimiter is the **tab**.

1)

Based on the requirement, we can use grep to extract the word 'Donald Trump'. -oi means ignore case sensitivity and display only the matched string in the next line.

```
$ awk -F '\t' '{print $4}' Twitter_Data_1 | grep -oi "Donald Trump" | wc -l
130
```

Donald Trump appears about **130** times in the data set.

2)

The system begins with ignore case setting with **BEGIN{IGNORECASE=1}**, then we use the method in task A part 8.

```
(awk -F '\t' 'BEGIN{IGNORECASE=1} index($4, "Donald Trump")'
Twitter_Data_1) > Donald.txt
```

After we create a data set we have to verify our data set has the correct amount of 'Donald Trump'.

```
$ awk -F '\t' '{print $4}' Donald.txt | grep -oi "Donald Trump" | wc -l
130
```

The number of “Donald Trump” is the same as in part 1. We can start extracting the time column and output it as .csv, then add the header to the column with the sed method.

- -i means edit file
- -e means the script in command will be executed
- 1i means insert before line 1

```
awk -F '\t' '{print $3}' Donald.txt > Donald.csv  
sed -i -e ' 1i"date"' Donald.csv
```

Now we can start analysing the data with R language.

Initial our R environment and check the data type

- We using ggplot in this assignment
- **rm(list=ls())** is used to clean the R environment
- Set working directory as TaskB
- Using read.csv to read csv file
- Str() can display the data type of column

```
library(ggplot2)  
#read data  
rm(list = ls())  
setwd("C:/A3/TaskB")  
df <- read.csv("Donald.csv")  
str(df)  
  
output:  
'data.frame': 122 obs. of 1 variable:  
 $ date: chr  "Tue Feb 11 12:28:36 +0000 2014" "Tue Feb 11 12:47:26 +0000  
2014" "Tue Feb 11 12:55:09 +0000 2014" "Tue Feb 11 13:22:29 +0000 2014" ...
```

The data type of date is the character, we have to turn it into date time. In this case, we use striptime with a format based on the time's data, and we set the time zone in GMT then replace data in the column.

After using striptime, the data type is POSIXlt, and we need to turn it to POSIXct for making graph. So we use as.POSIXct() to turn it into the data type we want.

```
#turn date from char to date
df[['date']]<-strptime(df[['date']],
                      format = "%a %b %e %H:%M:%S %z %Y",
                      tz = "GMT")
df[['date']]<-as.POSIXct(df[['date']])
str(df)

output:
'data.frame': 122 obs. of 1 variable:
 $ date: POSIXct, format: "2014-02-11 12:28:36" "2014-02-11 12:47:26"
```

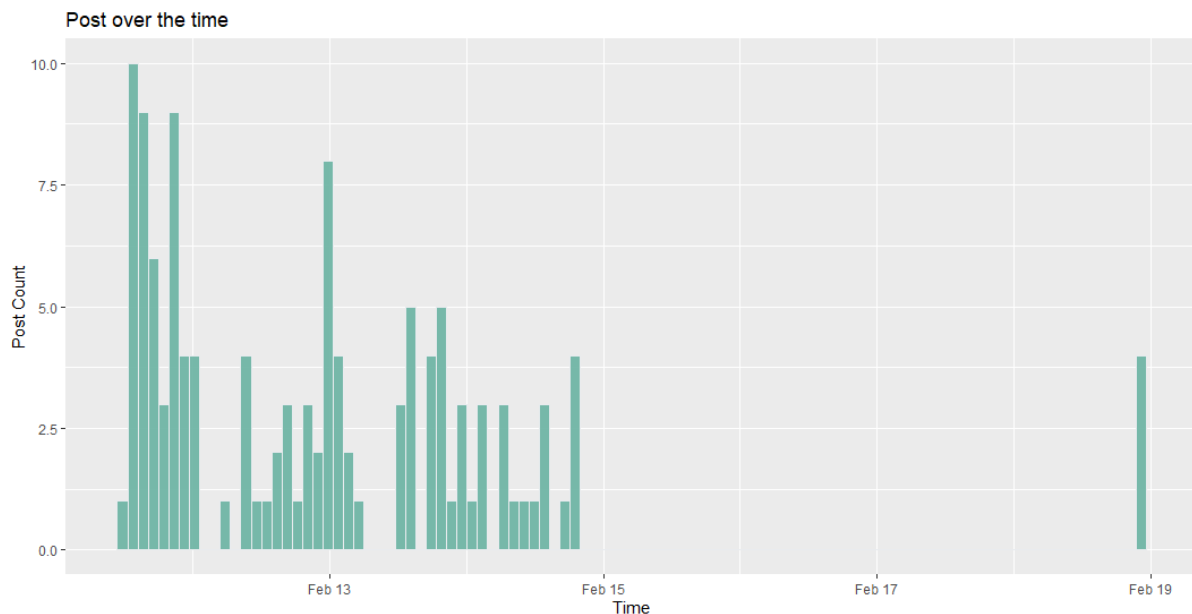
After the process, we can get the date data in POSIXct format.

3)

Plot the previous data with ggplot2, and set the plot title, x-axis and y-axis title with labs. Using geom_histogram to let ggplot plot a histogram and set the bins as 100.

```
ggplot(df,
       aes(x=date)) +
  labs(
    title = "Post over the time",
    x = "Time",
    y = "Post Count")+
  geom_histogram(bins = 100,
                fill="#69b3a2",
                color="#e9ecef",
                alpha=0.9)
```

Then we can get the histogram of posts over the time



4)

Based on the graph we can know:

- The data shows the **positive skewed bimodal distribution**
 - This suggests that before Feb 15, the number of posts reaches its highest point (10 posts) at midnight of Feb 11 and then drops to the bottom at day time of Feb 12. But on Feb 13, the number of posts goes up and then slowly goes down to 0.
 - After Feb 15, there is a little post (around 4 posts) on the night of Feb 18
 - User activity is higher during the day to 1 midnight the next day, with activity highest in the middle of the night.
 - The trend of Trump has been hot for about 3 days.

5)

Before we start our analysis, we prepare our dataset. We use the method in Task A case 8, but we add a comma between the element and count, then output as CSV. And then we add a header in our CSV file.

```
$ awk -F '\t' ' {arr[$2]++}END{for (element in arr) print arr[element],
",",element }' Twitter_Data_1 | sort -k1 -n > user.csv
$ sed -i -e ' 1i"count","user"' user.csv
```

Then we can input the CSV file in R and check the datatype of the column.

```
df2 <- read.csv("user.csv")
str(df2)

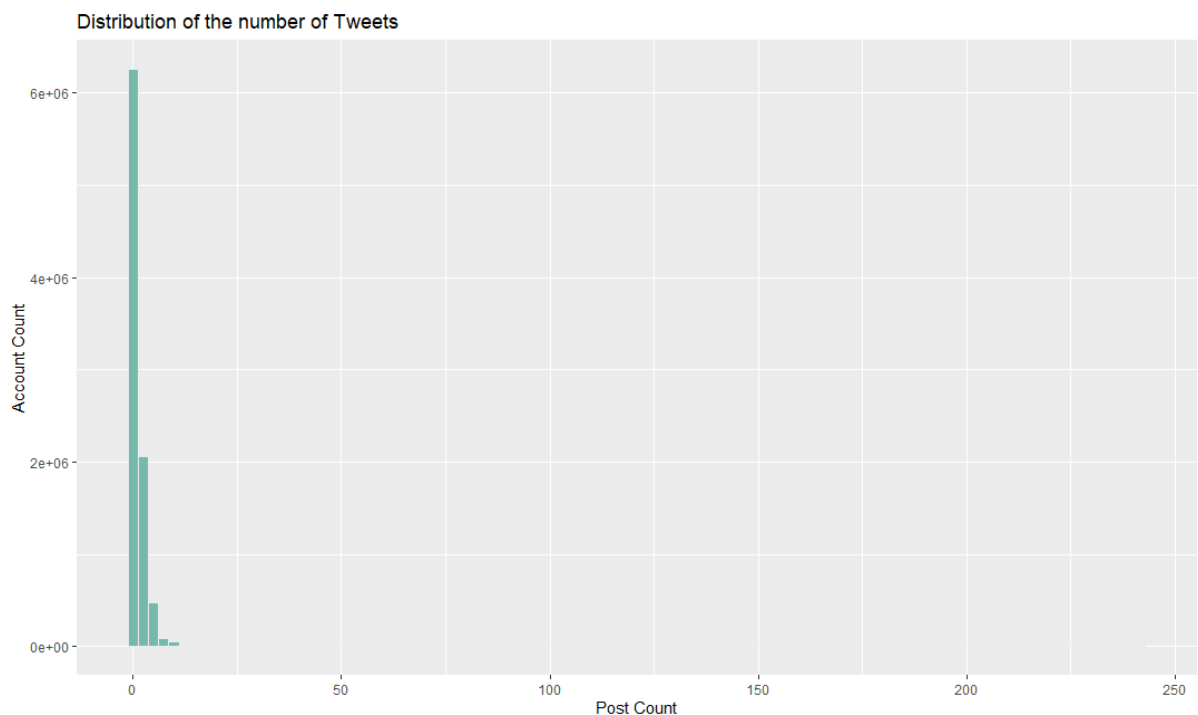
output:
'data.frame': 8977904 obs. of 2 variables:
 $ count: num 1 1 1 1 1 1 1 1 1 1 ...
```

```
$ user : chr " " " 000000000003737" " 0000000000_24" "  
0000000000yours" ...
```

We can see the column 'count' is a number and the 'user' is a character. Then we plot the histogram of the distribution of tweets.

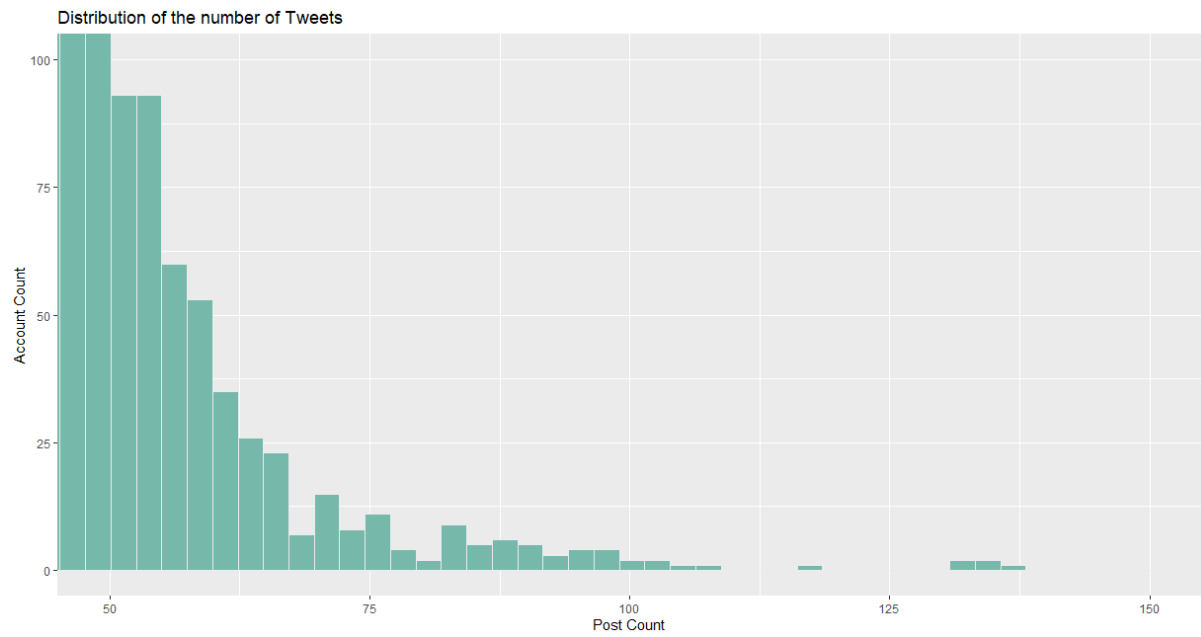
```
p = ggplot(df2,  
           aes(x=count)) +  
  labs(  
    title = "Distribution of the number of Tweets",  
    x = "Post Count",  
    y = "Account Count")+  
  geom_histogram(bins = 100,  
                fill="#69b3a2",  
                color="#e9ecef",  
                alpha=0.9)
```

p



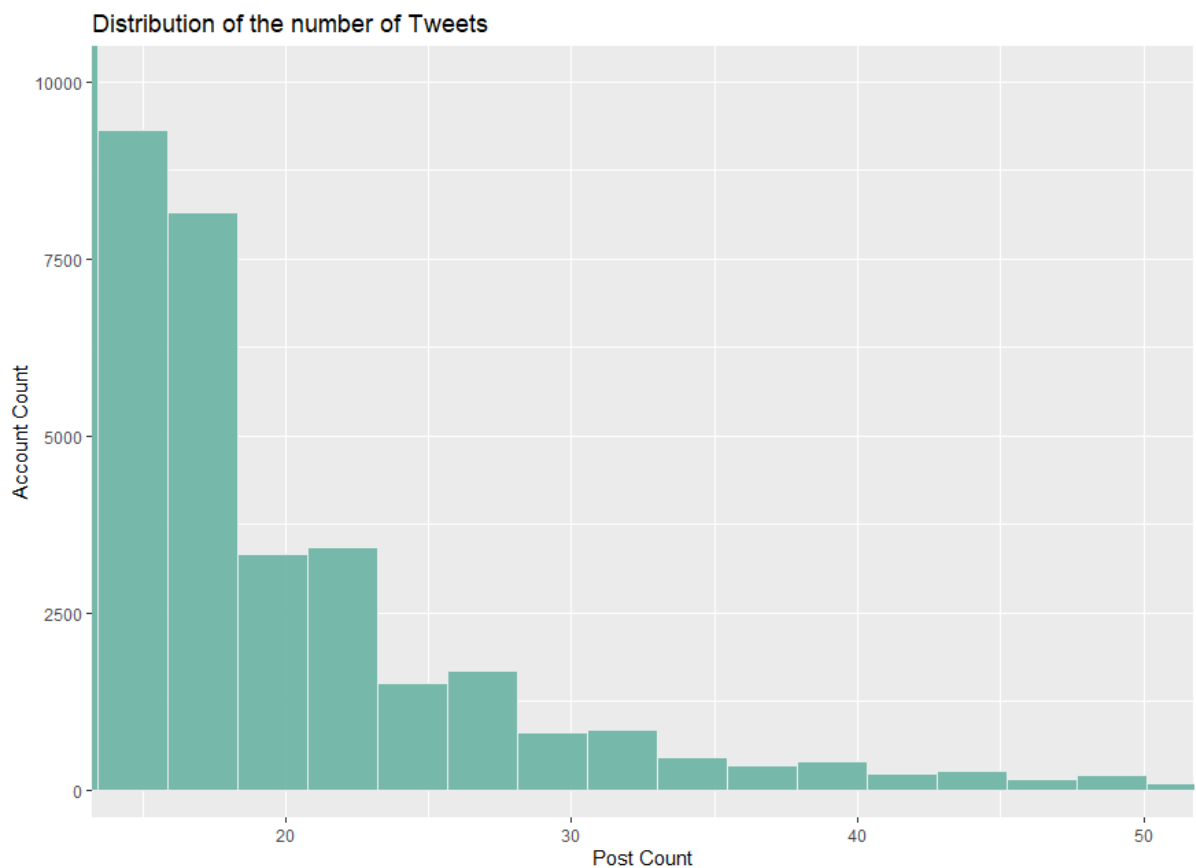
We can see most of the users (Around 6,000,000 users) only post one time. But we also want to see more details of the post count at the right of the graph. We can limit the x and y-axis.

```
p + coord_cartesian( xlim = c(50,150),ylim = c(0,100))
```



The number of accounts with more than 50 posts is all within 100.

```
p + coord_cartesian( xlim = c(15,50),ylim = c(100,10000))
```



The number of accounts within the range of 15-50 posts is all within 10000.