FIT 1043 Introduction to Data Science

Name: Chris Law Zi Qing

ID: 34112804

Introduction

In this assignment, an exploratory analysis is conducted on a large Twitter dataset related to COVID-19. A primary objective of this paper is to demonstrate how to handle and analyze large datasets using UNIX shell commands and the R programming language. Following is a breakdown of the assignment into several key sections:

1. **Inspecting the Data**: The first step involves reviewing the dataset's size, structure, headers, and the number of records in order to determine its overall composition.

2. **Investigating Information from the Data**: This phase is focused on identifying patterns and trends by examining particular aspects of the dataset, such as the number of unique Twitter users and the prevalence of certain keywords.

3. **Data Aggregation**: Datasets are grouped by number of followers each Twitter user has, which is then classified into various ranges to analyze how users are distributed across these counts.

4. **Visual Analysis in R**: The aggregation of data is used to generate visual representations in R that can be utilized to interpret the data and communicate the results effectively.

5. **Comparative Analysis**: The purpose of this section is to filter out retweets, repeat the aggregation and visualization process, and compare the results so that we can observe the impact of retweets on the analysis.

The assignment provides an insight into the public discourse during the COVID-19 pandemic by demonstrating practical data science techniques for managing and analyzing large datasets. It will provide students with an opportunity to gain insights into the impact of the pandemic on public discourse, including topics such as fake news, misinformation, and conspiracy theories. They will also be able to identify trends and develop strategies for mitigating them.

A1: Inspecting the data

Question 1

Code:

ls -lh corona_tweets.csv.gz

Answer:

118MB

Explanation:

The ls -lh command displays the file in plain text format, indicating its size.

Output:

```
(base) chrislaw@Chriss-MacBook-Pro Documents % ls -lh corona_tweets.csv.gz
-rw-r--r--@ 1 chrislaw  staff   118M May 14 12:18 corona_tweets.csv.gz
```

Question 2

Code:

head -1 corona_tweets.csv

Answer:

Created Tweet_ID Text User_ID User User_Location Followers_Count Friends_Count Geo Place_Type Place_Name Place_Country Language

Explanation:

The head -1 command extracts the first line looks to be header.

Output:

```
(base) chrislaw@Chriss-MacBook-Pro Documents % head -1 corona_tweets.csv
Created Tweet_ID Text User_ID User User_Location Followers_Count Friends_Count Geo Place_Type Place_Name Place_Country Language
```

Question 3

Code:

wc -l corona_tweets.csv

Answer:

1143559 lines

Explanation:

The wc -l command calculates the number of lines in the data

Output:

```
(base) chrislaw@Chriss-MacBook-Pro Documents % wc -l corona_tweets.csv
 1143559 corona_tweets.csv
```

A2: Investigating the information from Data

Question 1

Code:

```
cat corona_tweets.csv | awk -F'\t' '{print $4}' | sort | uniq -c | wc -l
```

Answer:

641976 twitter users

Explanation:

In this command, we extract the third column (Twitter user IDs), sort them, determine whether they are unique, and calculate them.

Output:

```
(base) chrislaw@Chriss-MacBook-Pro Documents % cat corona_tweets.csv | awk -F'\t' '{print $4}' | sort | uniq -c | wc -l
   641976
```

Question 2

a)

Code:

```
grep -i "vaccine" corona_tweets.csv | wc -l
```

Answer:

19483 tweets

Explanation:

The grep -i command carries out a case-insensitive lookup for the word "vaccine," while wc -l calculates the number of matching lines.

Output:

```
(base) chrislaw@Chriss-MacBook-Pro Documents % grep -i "vaccine" corona_tweets.csv | wc -l
    19483
```

b)

Code:

```
cat corona_tweets.csv | cut -f3 | grep -iw "vaccine" | grep -v -w "Vaccine" | grep -v -w "vaccine" | wc -l
```

Answer:

273

Explanation:

cat	This command reads the entire content of the file.
cut -f3	This command extracts the third field in the column from each line of the input
grep -iw "vaccine"	**This command selects all lines containing the word "vaccine" in any combination of uppercase or lowercase letters from the previous command's output.** The **-i** option makes the search case-insensitive, and the **-w** option ensures that only whole words are matched.
grep -v -w "Vaccine"	this command excludes any lines that contain the exact word "Vaccine".
grep -v -w "vaccine"	this command excludes any lines that contain the exact word "vaccine".
wc -l	This command counts the number of lines.

Output:

```
(base) chrislaw@Chriss-MacBook-Pro Desktop % cat corona_tweets.csv | cut -f3 | grep -iw "vaccine" | grep -v -w "Vaccine" | grep -v -w "vaccine" | wc -l
     273
```

c)

Code:

```
cat corona_tweets.csv | cut -f3 | grep -iw "vaccine" | grep -v -w "Vaccine" | grep -v -w "vaccine" | wc -l > Result.txt
```

Explanation:

A search is conducted in the corona_tweets.csv file to determine if there are any lines that contain the word "vaccine" in any case, eliminating lines that contain only the word "vaccine", and writing the remainder of the lines to Result.txt.

Output:

```
(base) chrislaw@Chriss-MacBook-Pro Desktop % cat corona_tweets.csv | cut -f3 | grep -iw "vaccine" | grep -v -w "Vaccine" | grep -v -w "vaccine" | wc -l > Result.txt
```

A3: Data aggregation

Question 1

cat	This command reads the entire content of the file.
awk -F'\t'	This command is a text-processing tool that allows for pattern scanning and processing, while sets the field delimiter to a tab character, indicating that columns in the CSV file are separated by tabs.
{print $4}'	If the condition is true, it prints the value in the fourth column which is the User ID
sort	This command sorts the output alphabetically or numerically.
uniq	This command counts the number of occurrences
wc -l	This command counts the number of lines.
| (pipeline)	This operator allows the output of one command to be used as the input for another command, allowing commands to be chained together.

a) Less than or equal to 1500

Code:

```
cat corona_tweets.csv  | awk -F'\t' '$7 <= 1500 {print $4}' | sort | uniq -c | wc -l
```

Answer:

498480

Explanation:

With this command, the file is filtered in order to find Twitter users with 1500 or fewer followers, their user IDs are extracted, and the unique IDs are counted.

Output:

```
(base) chrislaw@Chriss-MacBook-Pro Documents % cat corona_tweets.csv  | awk -F'\t' '$7 <= 1500 {print $4}' | sort | uniq -c | wc -l
  498480
```

b) 1501 to 2500

Code:

```
cat corona_tweets.csv | awk -F'\t' '$7 >= 1501 && $7 <= 2500 {print $4}' | sort | uniq | wc -l
```

Answer:

42891

Explanation:

Similarly, with this command, the file is filtered in order to find Twitter users with 1500 and above as well as less than or equal to 2500 followers, their user IDs are extracted, and the unique IDs are counted.

Output:

```
(base) chrislaw@Chriss-MacBook-Pro Documents % cat corona_tweets.csv | awk -F'\t' '$7 >= 1501 && $7 <= 2500 {print $4}' | sort | uniq | wc -l
    43891
```

c) 2501 to 3500

Code:

```
cat corona_tweets.csv | awk -F'\t' '$7 >= 2501 && $7 <= 3500 {print $4}' | sort | uniq | wc -l
```

Answer:

23620

Explanation:

Similarly, with this command, the file is filtered in order to find Twitter users with 2500 and above as well as less than or equal to 3500 followers, their user IDs are extracted, and the unique IDs are counted.

Output:

```
(base) chrislaw@Chriss-MacBook-Pro Documents % cat corona_tweets.csv | awk -F'\t' '$7 >= 2501 && $7 <= 3500 {print $4}' | sort | uniq | wc -l
    23620
```

d) 3501 to 4500

Code:

```
cat corona_tweets.csv | awk -F'\t' '$7 >= 3501 && $7 <= 4500 {print $4}' | sort | uniq | wc -l
```

Answer:

15165

Explanation:

Similarly, with this command, the file is filtered in order to find Twitter users with 3500 and above as well as less than or equal to 4500 followers, their user IDs are extracted, and the unique IDs are counted.

Output:

```
(base) chrislaw@Chriss-MacBook-Pro Documents % cat corona_tweets.csv | awk -F'\t' '$7 >= 3501 && $7 <= 4500 {print $4}' | sort | uniq | wc -l
    15165
```

e) 4501 to 5500

Code:

```
cat corona_tweets.csv | awk -F'\t' '$7 >= 4501 && $7 <= 5500 {print $4}' | sort | uniq | wc -l
```

Answer:

9297

Explanation:

Similarly, with this command, the file is filtered in order to find Twitter users with 4500 and above as well as less than or equal to 5500 followers, their user IDs are extracted, and the unique IDs are counted.

Output:

```
(base) chrislaw@Chriss-MacBook-Pro Documents % cat corona_tweets.csv | awk -F'\t' '$7 >= 4501 && $7 <= 5500 {print $4}' | sort | uniq | wc -l
    9297
```

f) 5501 to 6500

Code:

```
cat corona_tweets.csv | awk -F'\t' '$7 >= 5501 && $7 <= 6500 {print $4}' | sort | uniq | wc -l
```

Answer:

6848

Explanation:

Similarly, with this command, the file is filtered in order to find Twitter users with 5500 and above as well as less than or equal to 6500 followers, their user IDs are extracted, and the unique IDs are counted.

Output:

```
(base) chrislaw@Chriss-MacBook-Pro Documents % cat corona_tweets.csv | awk -F'\t' '$7 >= 5501 && $7 <= 6500 {print $4}' | sort | uniq | wc -l
    6848
```

g) 6501 to 7500

Code:

```
cat corona_tweets.csv | awk -F'\t' '$7 >= 6501 && $7 <= 7500 {print $4}' | sort | uniq | wc -l
```

Answer:

5076

Explanation:

Similarly, with this command, the file is filtered in order to find Twitter users with 6500 and above as well as less than or equal to 7500 followers, their user IDs are extracted, and the unique IDs are counted.

Output:

```
(base) chrislaw@Chriss-MacBook-Pro Documents % cat corona_tweets.csv | awk -F'\t' '$7 >= 6501 && $7 <= 7500 {print $4}' | sort | uniq | wc -l
    5076
```

h) 7501 to 8500

Code:

```
cat corona_tweets.csv | awk -F'\t' '$7 >= 7501 && $7 <= 8500 {print $4}' | sort | uniq | wc -l
```

Answer:

3855

Explanation:

Similarly, with this command, the file is filtered in order to find Twitter users with 7500 and above as well as less than or equal to 8500 followers, their user IDs are extracted, and the unique IDs are counted.

Output:

```
(base) chrislaw@Chriss-MacBook-Pro Documents % cat corona_tweets.csv | awk -F'\t' '$7 >= 7501 && $7 <= 8500 {print $4}' | sort | uniq | wc -l
     3855
```

i) 8501 to 9500

Code:

```
cat corona_tweets.csv | awk -F'\t' '$7 >= 8501 && $7 <= 9500 {print $4}' | sort | uniq | wc -l
```

Answer:

3072

Explanation:

Similarly, with this command, the file is filtered in order to find Twitter users with 8500 and above as well as less than or equal to 9500 followers, their user IDs are extracted, and the unique IDs are counted.

Output:

```
(base) chrislaw@Chriss-MacBook-Pro Documents % cat corona_tweets.csv | awk -F'\t' '$7 >= 8501 && $7 <= 9500 {print $4}' | sort | uniq | wc -l
     3072
```

j) More than 9500

Code:

```
cat corona_tweets.csv  | awk -F'\t' '$7 > 9500 {print $4}' | sort | uniq -c | wc -l
```

Answer:

32772

Explanation:

Similarly, with this command, the file is filtered in order to find Twitter users with more than 9500 followers, their user IDs are extracted, and the unique IDs are counted.

Output:

```
(base) chrislaw@Chriss-MacBook-Pro Documents % cat corona_tweets.csv  | awk -F'\t' '$7 > 9500 {print $4}' | sort | uniq -c | wc -l
   32772
```

Question 2

Created the table using Microsoft excel

	A	B
1	Range	The Numbers of Twitter Users
2	<= 1500	498480
3	1501 - 2500	43891
4	2501 - 3500	23620
5	3501 - 4500	15165
6	4501 - 5500	9297
7	5501 - 6500	6848
8	6501 - 7500	5076
9	7501 - 8500	3855
10	8501 - 9500	3072
11	> 9500	32772

Question3

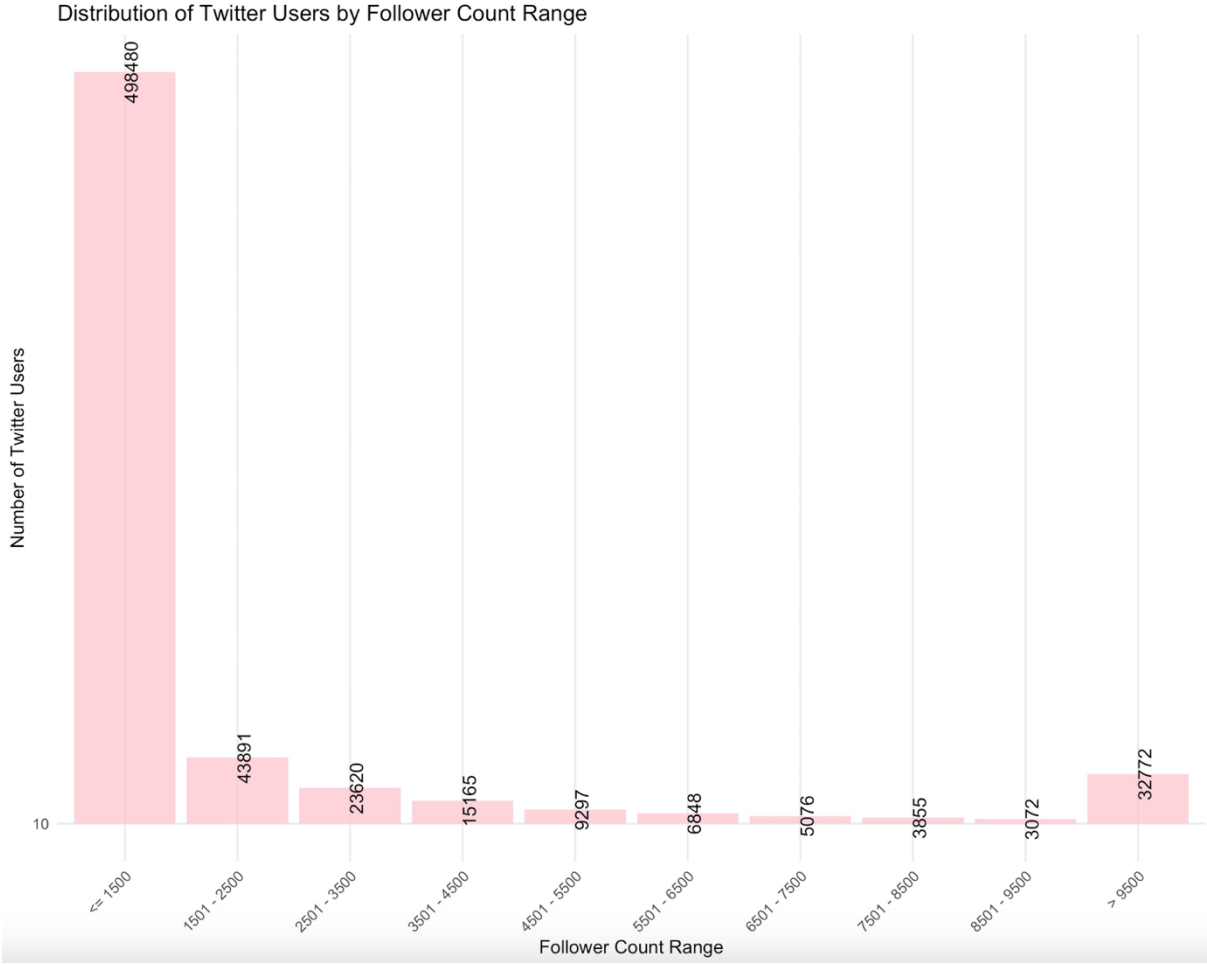
Code:

```
# Set the working directory
setwd("/Users/chrislaw/Documents")

# Read the CSV file
data <- read.csv("twitter.csv")

# Convert the Range column to a factor
data$Range <- factor(data$Range, levels = unique(data$Range))

# Load ggplot2
library(ggplot2)

# Calculate the max value of Number.of.Twitter.Users, handling NA values
max_value <- max(data$Number.of.Twitter.Users, na.rm = TRUE)

# Conditionally set y-axis breaks based on whether max_value is finite
y_breaks <- if (is.finite(max_value)) {
 seq(0, max_value, by = 20000)  # Set breaks up to max_value with intervals of 20000
} else {
c(0, 10)  # Default breaks if max_value is not finite
}

# Plotting a bar chart
p <- ggplot(data, aes(x = Range, y = The.Numbers.of.Twitter.Users)) +
      geom_bar(stat = "identity", fill = 'pink', alpha = 0.7) +
      geom_text(aes(label = The.Numbers.of.Twitter.Users), vjust = 1, angle = 90, hjust = 0.5)
      +  (title = 'Distribution of Twitter Users by Follower Count Range', x = 'Follower Count
      Range', y = 'Number of Twitter Users') +    theme_minimal() +  (axis.text.x =
      element_text(angle = 45, hjust = 1)) +  scale_y_continuous(breaks = y_breaks, labels =
      scales::comma)

# Print the plot
print(p)

# Save the plot as a PNG file to the current working directory
ggsave("TwitterUsers_barchart.png", plot = p, width = 10, height = 8)
```

Explanation:

setwd("/Users/chrislaw/Documents")	This command sets the working directory and any file operations of reading or writing will use this directory as the default location.
data <- read.csv	This command reads the CSV file from the current working directory into a dataframe.
data$Range <- factor(data$Range, levels = unique(data$Range))	This command converts the Range column in the dataframe into a factor, which is useful when dealing with categorical data. In the Range column, the levels are set to the unique values.
library(ggplot2)	This command loads ggplt2 package
max_value <- max(data$Number.of.Twitter.Users, na.rm = TRUE)	Calculates the maximum value of the The.Numbers.of.Twitter.Users column in the data dataframe, while ignoring NA values.
p <- ggplot(data, aes(x = Range, y = Number.of.Twitter.Users))	This command prepares the plot with the data, mapping Range to the x-axis and The.Numbers.of.Twitter.Users to the y-axis.
geom_bar(stat = "identity", fill = 'yellow', alpha = 0.7)	This command adds bars to the plot with a pink color and 70% opacity.
geom_text(aes(label = Number.of.Twitter.Users), vjust = 1, angle = 90, hjust = 0.5)	This command adds labels to the bars, adjusting the position and angle of the text.
labs(title = 'Distribution of Twitter Users by Follower Count Range', x = 'Follower Count Range', y = 'Number of Twitter Users')	This command adds the titles and the labels to the plot.
theme_minimal	This command applies the minimal theme to the plot
theme(axis.text.x = element_text(angle = 45, hjust = 1))	This command rotates the x-axis labels by 45 degrees for better readability.
scale_y_continuous(breaks = seq(0, max(data$Number.of.Twitter.Users, na.rm = TRUE), by = 2000 0), labels = scales::comma)	This command sets the breaks for the y-axis and formats the labels with commas.
print	This command prints the plot p to the graphics device
ggsave("TwitterUsersFollowerCount_bar_chart.png", plot = p, width = 10, height = 8))	This command saves the plot p as a PNG file in the current working directory, with a width of 10 inches and a height of 8 inches.

Question 4


Distribution of Twitter Users by Follower Count Range
Number of Twitter Users
10
498480
43891
23620
15165
9297
6848
5076
3855
3072
32772
<= 1500
1501 - 2500
2501 - 3500
3501 - 4500
4501 - 5500
5501 - 6500
6501 - 7500
7501 - 8500
8501 - 9500
> 9500
Follower Count Range

A4: Small Challenge

Question 1

Code:

```
gzcat corona_tweets.csv.gz | awk -F'\t' '$3 !~ /^RT @/' | gzip > filtered_corona_tweets.gz
```

Explanation:

gzcat	This command decompresses the file and outputs the content.
awk -F'\t'	This command is a text-processing tool that allows for pattern scanning and processing, while sets the field delimiter to a tab character, indicating that columns in the CSV file are separated by tabs.
$3 !~ /^RT @/'	This command checks if the third field of the current record does not start with the string "RT @".
gzip	This command compresses the filtered output.
| (pipeline)	This operator allows the output of one command to be used as the input for another command, allowing commands to be chained together.

As a result, this command is able to filter out retweets those lines in which the third field starts with "RT @" and saves the results to a gzipped file.

Output:

```
(base) chrislaw@Chriss-MacBook-Pro Desktop % gzcat corona_tweets.csv.gz | awk -F'\t' '$3 !~ /^RT @/' | gzip > filtered_corona_tweets.gz
```

Question 2

cat	This command reads the entire content of the file.
awk -F'\t'	This command is a text-processing tool that allows for pattern scanning and processing, while sets the field delimiter to a tab character, indicating that columns in the CSV file are separated by tabs.
{print $4}'	If the condition is true, it prints the value in the fourth column which is the User ID
sort	This command sorts the output alphabetically or numerically.
uniq	This command counts the number of occurrences
wc -l	This command counts the number of lines.
| (pipeline)	This operator allows the output of one command to be used as the input for another command, allowing commands to be chained together.

a)

Code:

```
cat filtered_corona_tweets.gz | gunzip | awk -F'\t' '$7 <= 1500 {print $4}' | sort | uniq | wc -l
```

Answer:

157068

Explanation:

With this command, the file is filtered in order to find Twitter users with 1500 or fewer followers, their user IDs are extracted, and the unique IDs are counted.

Output:

```
(base) chrislaw@Chriss-MacBook-Pro Desktop % cat filtered_corona_tweets.gz | gunzip | awk -F'\t' '$7 <= 1500 {print $4}' | sort | uniq | wc -l
  157068
```

b)

Code:

```
cat filtered_corona_tweets.gz | gunzip | awk -F'\t' '$7 >= 1501 && $7 <= 2500 {print $4}' | sort | uniq| wc -l
```

Answer:

16073

Explanation:

Similarly, with this command, the file is filtered in order to find Twitter users with 1500 and above as well as less than or equal to 2500 followers, their user IDs are extracted, and the unique IDs are counted.

Output:

```
(base) chrislaw@Chriss-MacBook-Pro Desktop % cat filtered_corona_tweets.gz | gunzip | awk -F'\t' '$7 >= 1501 && $7 <= 2500 {print $4}' | sort | uniq| wc -l
    16073
```

c)

Code:

```
cat filtered_corona_tweets.gz | gunzip | awk -F'\t' '$7 >= 2501 && $7 <= 3500 {print $4}' | sort | uniq| wc -l
```

Answer:

9016

Explanation:

Similarly, with this command, the file is filtered in order to find Twitter users with 2500 and above as well as less than or equal to 3500 followers, their user IDs are extracted, and the unique IDs are counted.

Output:

```
(base) chrislaw@Chriss-MacBook-Pro Desktop % cat filtered_corona_tweets.gz | gunzip | awk -F'\t' '$7 >= 2501 && $7 <= 3500 {print $4}' | sort | uniq| wc -l
     9016
```

d)

Code:

```
cat filtered_corona_tweets.gz | gunzip | awk -F'\t' '$7 >= 3501 && $7 <= 4500 {print $4}' | sort | uniq| wc -l
```

Answer:

6071

Explanation:

Similarly, with this command, the file is filtered in order to find Twitter users with 3500 and above as well as less than or equal to 4500 followers, their user IDs are extracted, and the unique IDs are counted.

Output:

```
(base) chrislaw@Chriss-MacBook-Pro Desktop % cat filtered_corona_tweets.gz | gunzip | awk -F'\t' '$7 >= 3501 && $7 <= 4500 {print $4}' | sort | uniq| wc -l
     6071
```

e)

Code:

```
cat filtered_corona_tweets.gz | gunzip | awk -F'\t' '$7 >= 4501 && $7 <= 5500 {print $4}' | sort | uniq| wc -l
```

Answer:

3872

Explanation:

Similarly, with this command, the file is filtered in order to find Twitter users with 4500 and above as well as less than or equal to 5500 followers, their user IDs are extracted, and the unique IDs are counted.

Output:

```
(base) chrislaw@Chriss-MacBook-Pro Desktop % cat filtered_corona_tweets.gz | gunzip | awk -F'\t' '$7 >= 4501 && $7 <= 5500 {print $4}' | sort | uniq| wc -l
     3872
```

f)

Code:

```
cat filtered_corona_tweets.gz | gunzip | awk -F'\t' '$7 >= 5501 && $7 <= 6500 {print $4}' | sort | uniq| wc -l
```

Answer:

2967

Explanation:

Similarly, with this command, the file is filtered in order to find Twitter users with 5500 and above as well as less than or equal to 6500 followers, their user IDs are extracted, and the unique IDs are counted.

Output:

```
(base) chrislaw@Chriss-MacBook-Pro Desktop % cat filtered_corona_tweets.gz | gunzip | awk -F'\t' '$7 >= 5501 && $7 <= 6500 {print $4}' | sort | uniq| wc -l
     2967
```

g)

Code:

```
cat filtered_corona_tweets.gz | gunzip | awk -F'\t' '$7 >= 6501 && $7 <= 7500 {print $4}' | sort | uniq| wc -l
```

Answer:

2187

Explanation:

Similarly, with this command, the file is filtered in order to find Twitter users with 6500 and above as well as less than or equal to 7500 followers, their user IDs are extracted, and the unique IDs are counted.

Output:

```
(base) chrislaw@Chriss-MacBook-Pro Desktop % cat filtered_corona_tweets.gz | gunzip | awk -F'\t' '$7 >= 6501 && $7 <= 7500 {print $4}' | sort | uniq| wc -l
     2187
```

h)

Code:

```
cat filtered_corona_tweets.gz | gunzip | awk -F'\t' '$7 >= 7501 && $7 <= 8500 {print $4}' | sort | uniq| wc -l
```

Answer:

1726

Explanation:

Similarly, with this command, the file is filtered in order to find Twitter users with 7500 and above as well as less than or equal to 8500 followers, their user IDs are extracted, and the unique IDs are counted.

Output:

```
(base) chrislaw@Chriss-MacBook-Pro Desktop % cat filtered_corona_tweets.gz | gunzip | awk -F'\t' '$7 >= 7501 && $7 <= 8500 {print $4}' | sort | uniq| wc -l
     1726
```

i)

Code:

```
cat filtered_corona_tweets.gz | gunzip | awk -F'\t' '$7 >= 8501 && $7 <= 9500 {print $4}' | sort | uniq| wc -l
```

Answer:

1428

Explanation:

Similarly, with this command, the file is filtered in order to find Twitter users with 8500 and above as well as less than or equal to 9500 followers, their user IDs are extracted, and the unique IDs are counted.

Output:

```
(base) chrislaw@Chriss-MacBook-Pro Desktop % cat filtered_corona_tweets.gz | gunzip | awk -F'\t' '$7 >= 8501 && $7 <= 9500 {print $4}' | sort | uniq| wc -l
     1428
```

j)

Code:

```
cat filtered_corona_tweets.gz | gunzip | awk -F'\t' '$7 >9500 {print $4}' | sort | uniq -c | wc -l
```

Answer:

17645

Explanation:

Similarly, with this command, the file is filtered in order to find Twitter users with more than 9500 followers, their user IDs are extracted, and the unique IDs are counted.

Output:

```
(base) chrislaw@Chriss-MacBook-Pro Desktop % cat filtered_corona_tweets.gz | gunzip | awk -F'\t' '$7 >9500 {print $4}' | sort | uniq -c | wc -l
    17645
```

<u>Question 3</u>

Created by using Microsoft Excel

	A	B
1	Ranges	The Numbers of Twitter Users
2	<= 1500	157068
3	1501 - 2500	16073
4	2501 - 3500	9016
5	3501 - 4500	6071
6	4501 - 5500	3872
7	5501 - 6500	2967
8	6501 - 7500	2187
9	7501 - 8500	1726
10	8501 - 9500	1428
11	> 9500	17645

Code:

```
data1 <- read.csv("filteres_twitter.csv")
```

Explanation:

This command reads the file and store into the dataframe

Output:

data1	10 obs. of 3 variables

Question 4

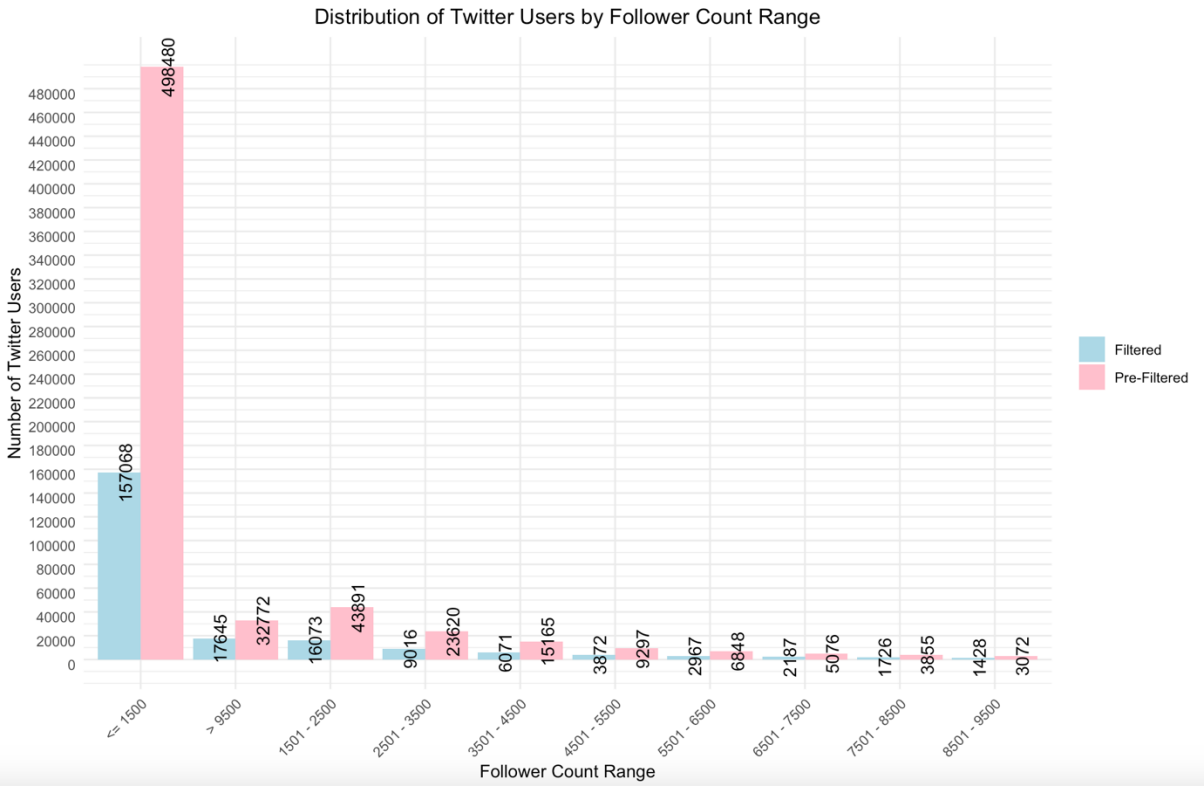
Code:

```
# Set the working directory
setwd("/Users/chrislaw/Documents")

# Read the CSV files
data <- read.csv("twitter.csv")
data1 <- read.csv("filt_twitter.csv")

# Add a new column to indicate the dataset source
data$Dataset <- "Pre-Filtered"
data1$Dataset <- "Filtered"

# Merge the dataframes
merged_data <- rbind(data, data1)

# Plotting the bar chart

p2 <- ggplot(data = merged_data, aes(x = Range, y = The.Numbers.of.Twitter.Users, fill
      Dataset)) + geom_bar(stat = "identity", position = "dodge") +  geom_text(aes(label =
      The.Numbers.of.Twitter.Users), position = position_dodge(width = 0.9), vjust = 1,
      angle = 90, hjust = 0.5) + labs(title = "Distribution of Twitter Users by Follower Count
      Range", x = "Follower Count Range", y = "Number of Twitter Users") +  # Add titles and
      labels scale_fill_manual(values = c("Pre-Filtered" = "pink", "Filtered" = "lightblue"))
      theme_minimal() +  theme(legend.title = element_blank(), plot.background =
      element_rect(fill = "white")) +  theme(axis.text.x = element_text(angle = 45, vjust = 1,
      hjust = 1)) + theme(axis.text.y = element_text(angle = 0, vjust = 1, hjust = 1)) +
      theme(plot.title = element_text(hjust = 0.5)) + scale_y_continuous(breaks = seq(0,
      max(merged_data$The.Numbers.of.Twitter.Users, na.rm = TRUE), by = 20000))

# Display the plot
print(p2)

# Save the plot as a PNG file
ggsave("TwitterUsers_Filtered_barchartV1.png", plot = p2, width = 10, height = 8)
```

Explanation:

setwd("/Users/chrislaw/Documents")	This command sets the working directory and any file operations of reading or writing will use this directory as the default location.
data <- read.csv	This command reads the CSV file from the current working directory into a dataframe.
data$Dataset <- “”	This command adds a new column named Dataset to both dataframes to indicate the source of the data. data is labeled as "Pre-Filtered" and data1 is labeled as "Filtered".
merged_data <- rbind(data, data1)	This command merge two dataframes into one dataframe
ggplot(data = merged_data, aes(x = Range, y = The.Numbers.of.Twitter.Users, fill = Dataset))	This command sets up the plot with **merged_data**, mapping **Range** to the x-axis, **The.Numbers.of.Twitter.Users** to the y-axis, and **Dataset** to fill color.
geom_bar(stat = "identity", position = "dodge")	This command adds bars to the plot with the heights representing values in **The.Numbers.of.Twitter.Users**, placing bars side by side for each **Range** value.
geom_text(aes(label = The.Numbers.of.Twitter.Users), position = position_dodge(width = 0.9), vjust = 1, angle = 90, hjust = 0.5)	This command adds adds title and label to the plot
labs(title = "Distribution of Twitter Users by Follower Count Range", x = "Follower Count Range", y = "Number of Twitter Users")	This command adds labels to the bars, adjusting the position and angle of the text.
scale_fill_manual(values = c("Pre-Filtered" = "pink", "Filtered" = "lightblue"))	This command sets custom colors for the bars to differenciate the d**ataset** column.
theme_minimal	This command applies the minimal theme to the plot
theme(legend.title = element_blank(), plot.background = element_rect(fill = "white"))	This command can customise the legend of the backgroung
theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))	This command rotates the x-axis labels by 45 degrees for better readability.
theme(axis.text.y = element_text(angle = 0, vjust = 1, hjust = 1))	This command will adjust the positionof the y axis label
theme(plot.title = element_text(hjust = 0.5))	This command will make the title to be in the middle
scale_y_continuous	This command formats the labels and sets the breaks for the y-axis.

Output:


Distribution of Twitter Users by Follower Count Range
Number of Twitter Users
Follower Count Range
Filtered
Pre-Filtered
<= 1500
> 9500
1501 - 2500
2501 - 3500
3501 - 4500
4501 - 5500
5501 - 6500
6501 - 7500
7501 - 8500
8501 - 9500
157068
498480
17645
32772
16073
43891
9016
23620
6071
15165
3872
9297
2967
6848
2187
5076
1726
3855
1428
3072

Question 5

Analysis of Twitter User Behavior Based on Follower Count and Retweeting Patterns

According to the graph, Twitter users' behavior is represented in a compelling way, especially in the context of the influence of follower count on content creation. After excluding retweets, the number of users with 1500 or fewer followers decreases dramatically from 498,480 to 157,068. Among users with fewer followers, this sharp decline indicates a heavy reliance on retweeting. There is a noticeable reduction in the number of followers between 1501 and 7500, but it is less severe. Furthermore, in the higher follower brackets between 7501 and over 9500, the decline is minimal, indicating that users with more followers are more likely to produce original content. A noteworthy anomaly appears where the number of users with more than 9500 followers surpasses the number of users in the previous range, indicating a significant increase in the creation of original content.

Based on this data, it is evident that original content generation and follower count are strongly correlated. It has been observed that accounts that post original content tend to attract a greater number of followers, whereas those that rely heavily on retweets have difficulty gaining a significant following. As follower counts increase, the percentage of users relying on retweets decreases, which indicates that more popular accounts are creating original content. In particular, this data is of great value to companies and organizational marketing teams, playing a crucial role in their decision-making process. It is beneficial to collaborate with content creators who create original content, since their followers are usually more emotionally invested and trustworthy. There is often a sense of relatability and personalization among these creators that is appealing to their followers.

In order to make your advertising more impactful and authentic, it is imperative that you engage with original content creators. As a result of the credibility and influence of these creators, marketing campaigns have become more effective. Business can create marketing strategies that resonate deeply with their target audiences by leveraging the trust and engagement resulting from the creation of original content, thus cultivating stronger relationships and achieving better outcomes. In this manner, advertising efforts are able to reach a greater number of consumers while enhancing authenticity and impact, making campaigns more effective and credible in the eyes of consumers.

As a consequence, the data demonstrate the prevalence of retweeting among Twitter users with fewer followers. It can be seen from this trend that it decreases as the number of followers increases, demonstrating that creating original content is a more effective way to attract followers than simply reposting another's content. In order for corporations to establish effective advertising partnerships, it is important to understand this dynamic. A more impactful promotional campaign is enhanced when influencers are identified and marketing strategies are customized to target those who have high levels of original content and engagement, which leverages the trust and emotional investment followers place in content creators who create their own content.