Intro to Data Science - Lab 5

IST687 Section M002

Professor Anderson

Enter your name here: Chaithra Kopparam Cheluvaiah

#Select one of the below and add needed information

1. I did this homework by myself, with help from the book and the professor.

```
# install.packages("RCurl")
# install.packages("jsonlite")
```

```
# loading the RCurl package which was installed in the previous step
# used for accessing internet data
library(RCurl)

# loading the jsonlite package which was installed in the previous step
# required for decoding json
library(jsonlite)

# assigning the Citi bike URL of JSON data to station_link variable
station_link <- 'https://gbfs.citibikenyc.com/gbfs/en/station_status.json'

# getting the JSON data
apiOutput <- getURL(station_link)

# parse the JSON data to R object
apiData <- fromJSON(apiOutput)

# getting stations data frame which is inside 'data' list
stationStatus <- apiData$data$stations

# keeping the required columns from JSON data
cols <- c('num_bikes_disabled','num_docks_disabled', 'station_id',
          'num_ebikes_available', 'num_bikes_available', 'num_docks_available')

# sub-setting data frame to have only the required columns that are selected
# in 'cols' vector
stationStatus = stationStatus[,cols]
```

1. Explain what you see if you type in the station_link URL into a browser (in a comment, write what you see)

```
# We have JSON data (key: value pairs) of citi bikes. Each JSON node is
# representing a station showing the information related to the availability of
# bikes, e-bikes, docks.
```

2. Provide a comment explaining each line of code.

3. Use str( ) to find out the structure of apiOutput and apiData. Report (via a comment) what you found and explain the difference between these two objects.

```
# str(apiOutput) #apiOutput is a string having entire JSON data

# str(apiData) #apiData is a R object having stations data frame inside data list
```

4. The apiOutput object can also be examined with a custom function from the jsonlite package called prettify( ). Run this command and explain what you found (in a comment).

```
#prettify(apiOutput)
#It formats JSON data which is helpful to look at key-value pairs inside the
#JSON nodes. Basically makes the JSON data more readable.
```

5. Explain stationStatus (what type of object, what information is available)

```
summary(stationStatus)
```

```
##  num_bikes_disabled num_docks_disabled  station_id        num_ebikes_available
##  Min.   : 0.000     Min.   : 0.0000    Length:1578        Min.   : 0.000
##  1st Qu.: 0.000     1st Qu.: 0.0000    Class :character   1st Qu.: 0.000
##  Median : 1.000     Median : 0.0000    Mode  :character   Median : 0.000
##  Mean   : 1.956     Mean   : 0.1496                       Mean   : 0.474
##  3rd Qu.: 3.000     3rd Qu.: 0.0000                       3rd Qu.: 1.000
##  Max.   :16.000     Max.   :45.0000                       Max.   :10.000
##  num_bikes_available num_docks_available
##  Min.   : 0.00       Min.   : 0.0
##  1st Qu.: 3.00       1st Qu.: 5.0
##  Median : 9.00       Median :13.0
##  Mean   :13.13       Mean   :14.6
##  3rd Qu.:19.00       3rd Qu.:20.0
##  Max.   :79.00       Max.   :75.0
```

```
str(stationStatus)
```

```
## 'data.frame':    1578 obs. of  6 variables:
##  $ num_bikes_disabled  : int  2 2 3 5 3 4 3 3 6 1 ...
##  $ num_docks_disabled  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ station_id          : chr  "72" "79" "82" "83" ...
##  $ num_ebikes_available: int  0 0 0 0 0 1 2 0 0 1 ...
##  $ num_bikes_available : int  40 27 24 50 45 46 9 27 49 13 ...
##  $ num_docks_available : int  13 4 0 7 2 3 7 1 1 0 ...
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.1     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x tidyr::complete() masks RCurl::complete()
## x dplyr::filter()   masks stats::filter()
## x purrr::flatten()  masks jsonlite::flatten()
## x dplyr::lag()      masks stats::lag()
```

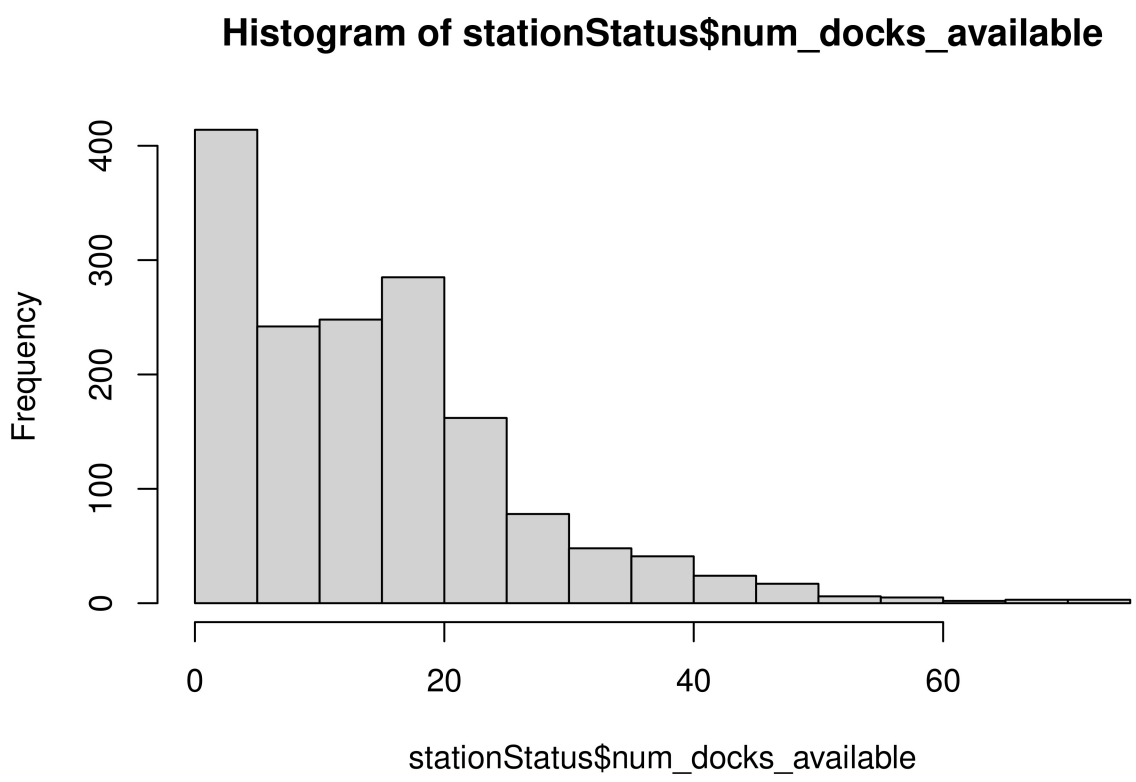
```
glimpse(stationStatus)
```

```
## Rows: 1,578
## Columns: 6
## $ num_bikes_disabled   <int> 2, 2, 3, 5, 3, 4, 3, 3, 6, 1, 2, 5, 4, 2, 1, 3, 1~
## $ num_docks_disabled   <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1~
## $ station_id           <chr> "72", "79", "82", "83", "116", "119", "120", "127~
## $ num_ebikes_available <int> 0, 0, 0, 0, 0, 1, 2, 0, 0, 1, 0, 0, 2, 0, 0, 0, 1~
## $ num_bikes_available  <int> 40, 27, 24, 50, 45, 46, 9, 27, 49, 13, 54, 44, 17~
## $ num_docks_available  <int> 13, 4, 0, 7, 2, 3, 7, 1, 1, 0, 2, 6, 35, 4, 30, 6~
```

```
# stationStatus is a data frame. It has only 6 attributes:num_bikes_disabled,
# num_docks_disabled, station_id,num_ebikes_available, num_docks_available, and
# num_docks_available
```

6. Generate a histogram of the number of docks available

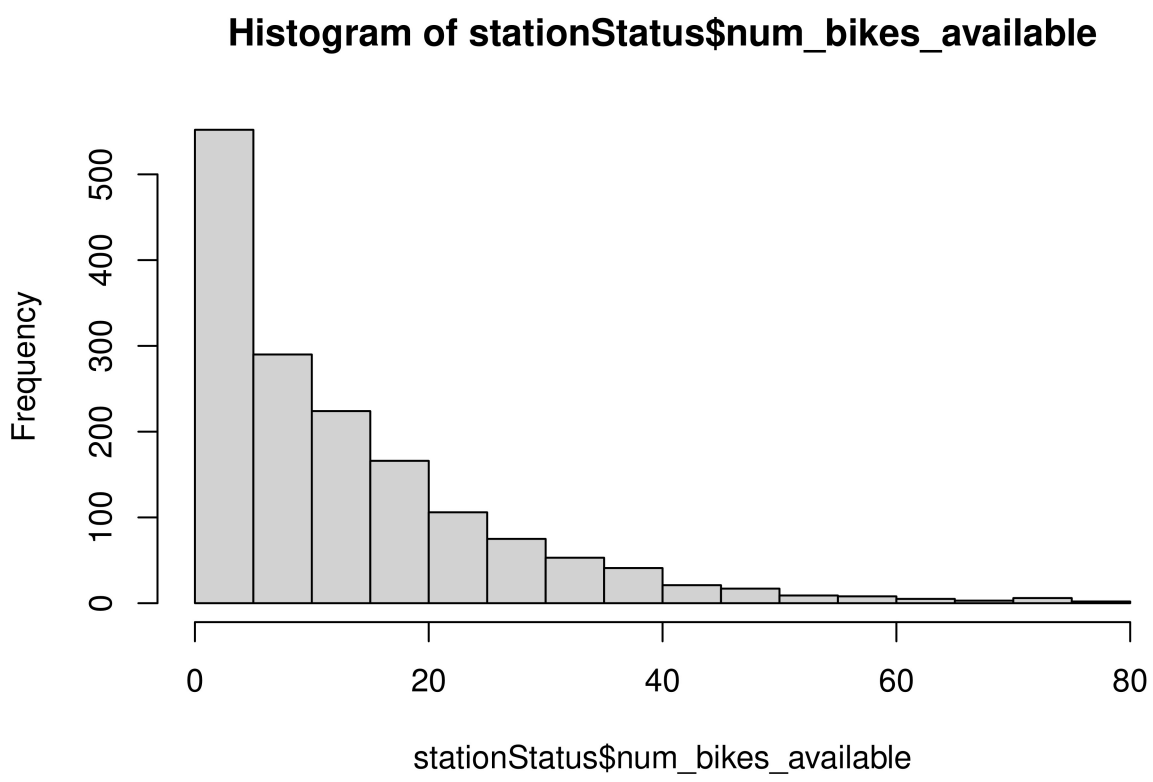
```
hist(stationStatus$num_docks_available) # creates histogram plot of
```




```
# num_docks_available
```

7. Generate a histogram of the number of bikes available

```
hist(stationStatus$num_bikes_available) # creates histogram plot of
```




```
# num_bikes_available
```

8. How many stations have at least one ebike?

```
# sub-setting the data frame with logical condition to find observations that
# are having one or more ebikes availability
ebikesStation <- stationStatus[stationStatus$num_ebikes_available>0,]

nrow(ebikesStation) # returns the number of rows in a data frame
```

```
## [1] 477
```

```
length(ebikesStation$num_ebikes_available) #returns the number of items present
```

```
## [1] 477
```

```
# in the column
```

9. Explore stations with at least one ebike by create a new dataframe, that only has stations with at least one eBike.

```
# sub-setting the data frame with boolean condition to find observations that
# are having one or more ebikes availability
ebikesStation <- stationStatus[stationStatus$num_ebikes_available>0,]
```

10. Calculate the mean of 'num_docks_available' for this new dataframe.

```
mean(ebikesStation$num_docks_available) # finding average number of docks
```

```
## [1] 13.32075
```

```
# available in ebikes stations
```

11. Calculate the mean of 'num_docks_available' for for the full 'stationStatus' dataframe. In a comment, explain how different are the two means?

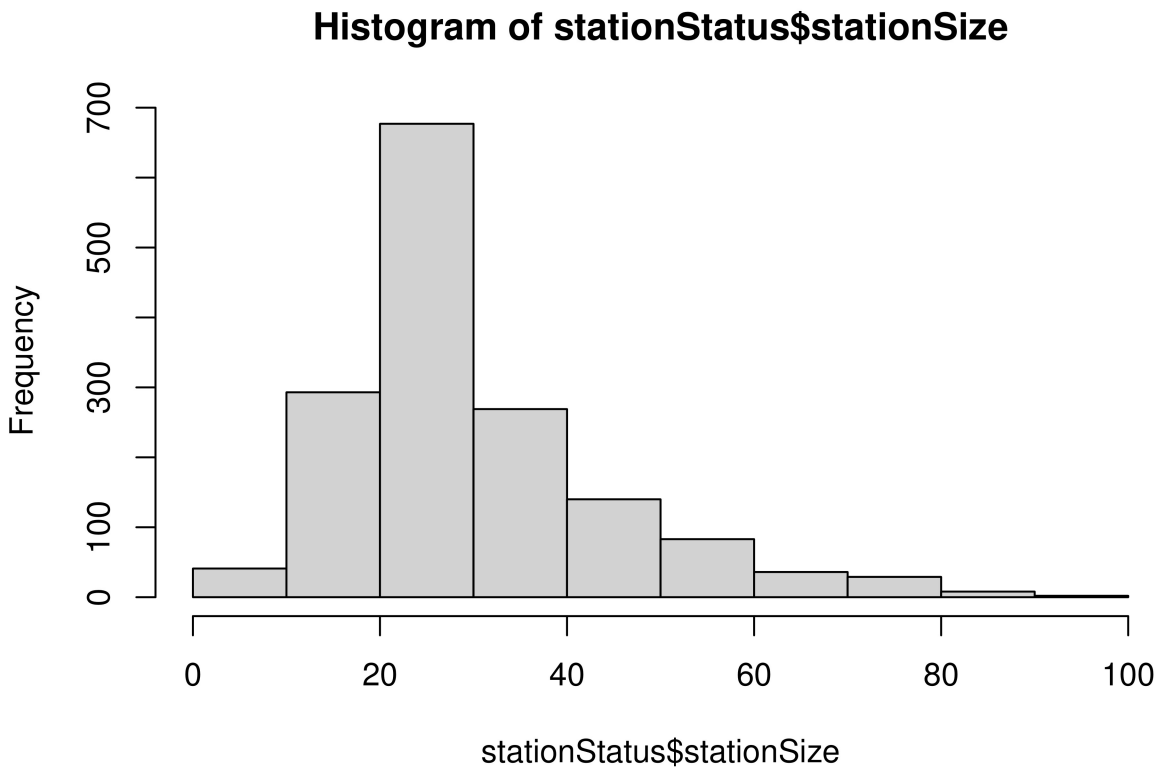
```
mean(stationStatus$num_docks_available)
```

```
## [1] 14.59759
```

```
# average that was found earlier includes only the docks available in the
# stations that have at least one e-bike.

# average that is calculated now includes the docks available in the stations
# that have zero e-bikes as well
```

12. Create a new attribute, called 'stationSize', which is the total number of "slots" available for a bike (that might, or might not, have a bike in it now). Run a histogram on this variable and review the distribution.

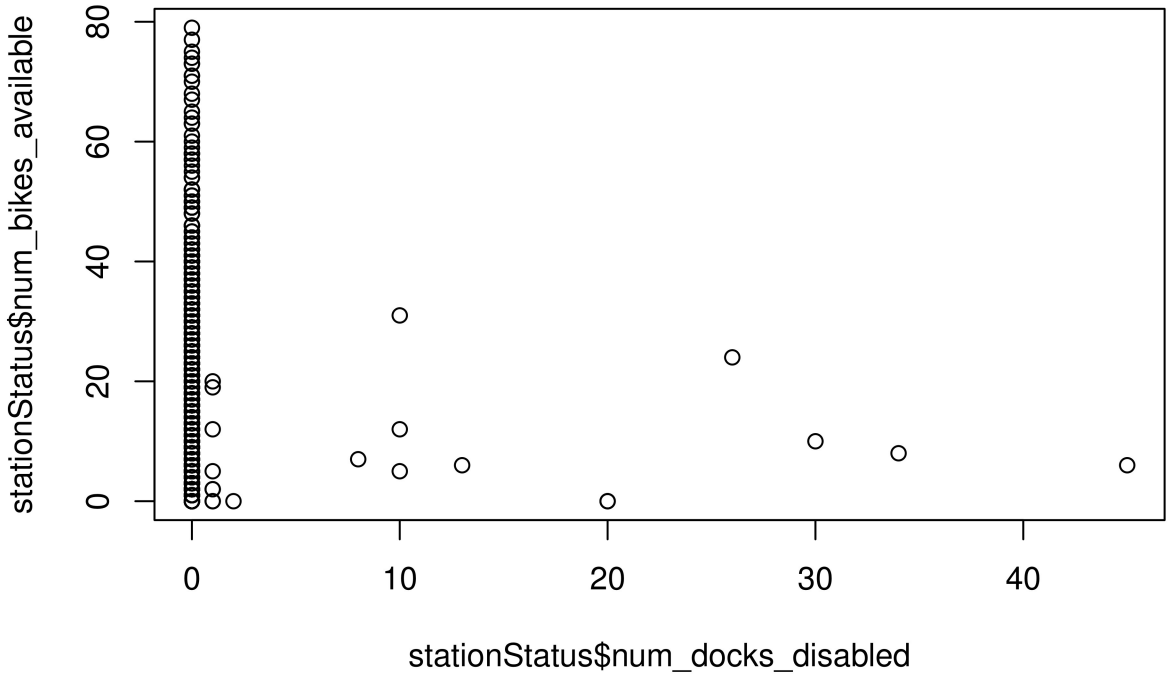
```
# finding total number of slots available for a bike by adding disabled docks,
# disabled bikes, available e-bikes, bikes, and docks
slots <- stationStatus$num_bikes_disabled +
  stationStatus$num_docks_disabled +
  stationStatus$num_ebikes_available +
  stationStatus$num_bikes_available +
  stationStatus$num_docks_available

# adding new column to dataframe
stationStatus$stationSize <- slots

#creating histogram plot
hist(stationStatus$stationSize)
```




```
# stationSize results in right-skewed distribution
```

13. Use the plot( ) command to produce an X-Y scatter plot with the number of occupied docks on the X-axis and the number of available bikes on the Y-axis. Explain the results plot.

```
# creating scatter plot
plot(stationStatus$num_docks_disabled,stationStatus$num_bikes_available)
```


stationStatus$num_bikes_available
0
20
40
60
80
0
10
20
30
40
stationStatus$num_docks_disabled