Intro to Data Science - Lab 5

Week 5 - Obtaining and Using Data from a JSON API

```
# Enter your name here: Hendi Kushta
```

Please include nice comments.

Instructions:

Run the necessary code on your own instance of R-Studio.

Attribution statement: (choose only one and delete the rest)

```
# 1. I did this lab assignment by myself, with help from the book and the professor.
```

JSON (JavaScript Object Notation) provides a transparent, user friendly data exchange format that many organizations use to make their data available over the web. JSON is human readable, but is also highly structured, and its support for nested data structures makes it highly flexible. Today we will use JSON to obtain data from the **NYC CitiBike project**. The CitiBike project provides an application programming interface (API) that members of the public can access to get up-todate information on the status of more than 800 bike stations. You may need to install the **RCurl** and **jsonlite** packages to get the code to work.

```
station_link <- 'https://gbfs.citibikenyc.com/gbfs/en/station_status.json'
apiOutput <- getURL(station_link)
apiData <- fromJSON(apiOutput)
stationStatus <- apiData$data$stations
cols <- c('num_bikes_disabled','num_docks_disabled', 'station_id',
 'num_ebikes_available', 'num_bikes_available', 'num_docks_available')
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)

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

library(RCurl)
library(jsonlite)

# station_link URL will show JSON data, data that are online,
# shown in a list of dictionaries
```

2. Paste the code from above here and provide a comment explaining each line of code.

```
station_link <- 'https://gbfs.citibikenyc.com/gbfs/en/station_status.json'
# link for the online data
apiOutput <- getURL(station_link) # request the data
apiData <- fromJSON(apiOutput) # pass the structure of data to
# the function fromJSON()
stationStatus <- apiData$data$stations # select stations from our data
cols <- c('num_bikes_disabled','num_docks_disabled', 'station_id',
 'num_ebikes_available', 'num_bikes_available', 'num_docks_available')
# name the columns of our dataframe
```

```
stationStatus = stationStatus[,cols] # create dataframe with all the
# columns and rows
```

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) # shows the structure of keys and values for the data.
```

```
##  chr "{\"data\":{\"stations\":[{\"last_reported\":1656077825,\"is_installed\":0,\"num_docks_available
```

```
# It is truncated, not comlete
# str(apiData) # shows the structure of the data. Nr of observations, attributes.
# shows types of attributes.
```

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) # shows better the structure of the data with
# the keys and the values for each station.
```

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

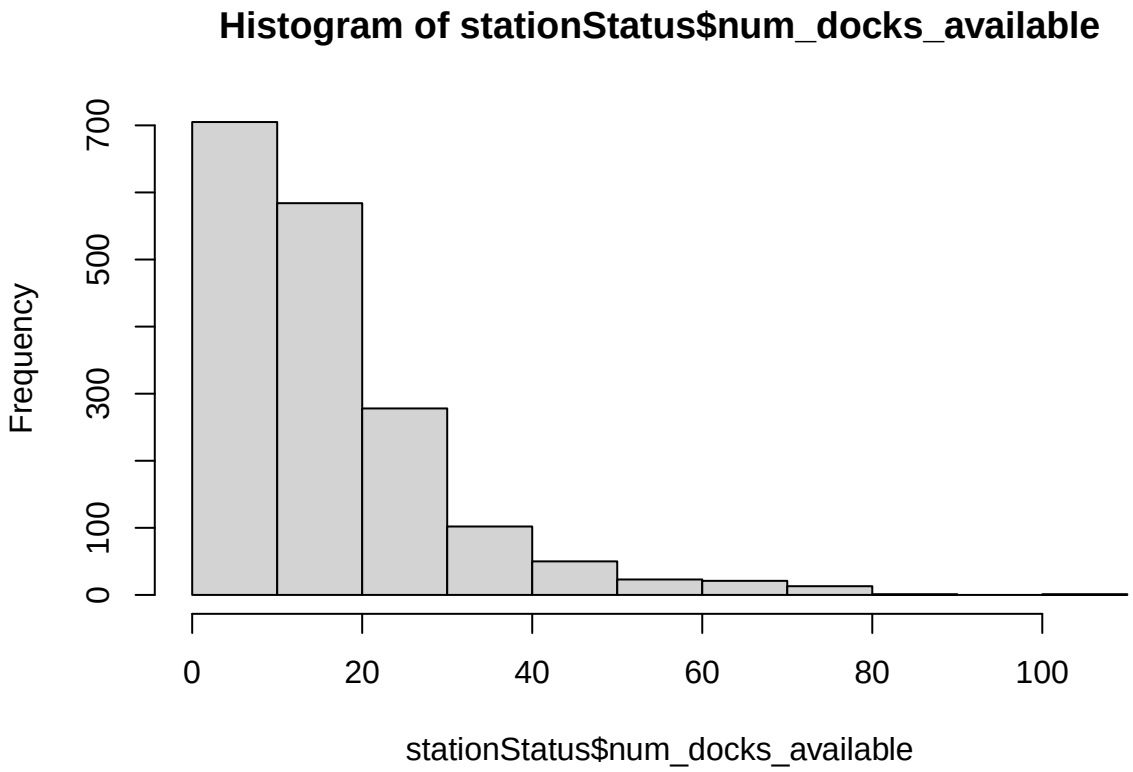
```
str(stationStatus)
```

```
## 'data.frame':    1778 obs. of  6 variables:
##  $ num_bikes_disabled  : int  0 3 2 4 2 1 1 1 4 3 ...
##  $ num_docks_disabled  : int  0 0 0 1 0 0 0 0 0 0 ...
##  $ station_id          : chr  "72" "79" "82" "83" ...
##  $ num_ebikes_available: int  0 0 1 0 1 0 2 3 2 0 ...
##  $ num_bikes_available : int  0 15 14 14 3 35 11 16 49 30 ...
##  $ num_docks_available : int  19 13 11 42 69 17 6 13 1 16 ...
```

```
# it is a dataframe with 1778 observations and 6 variables.
```

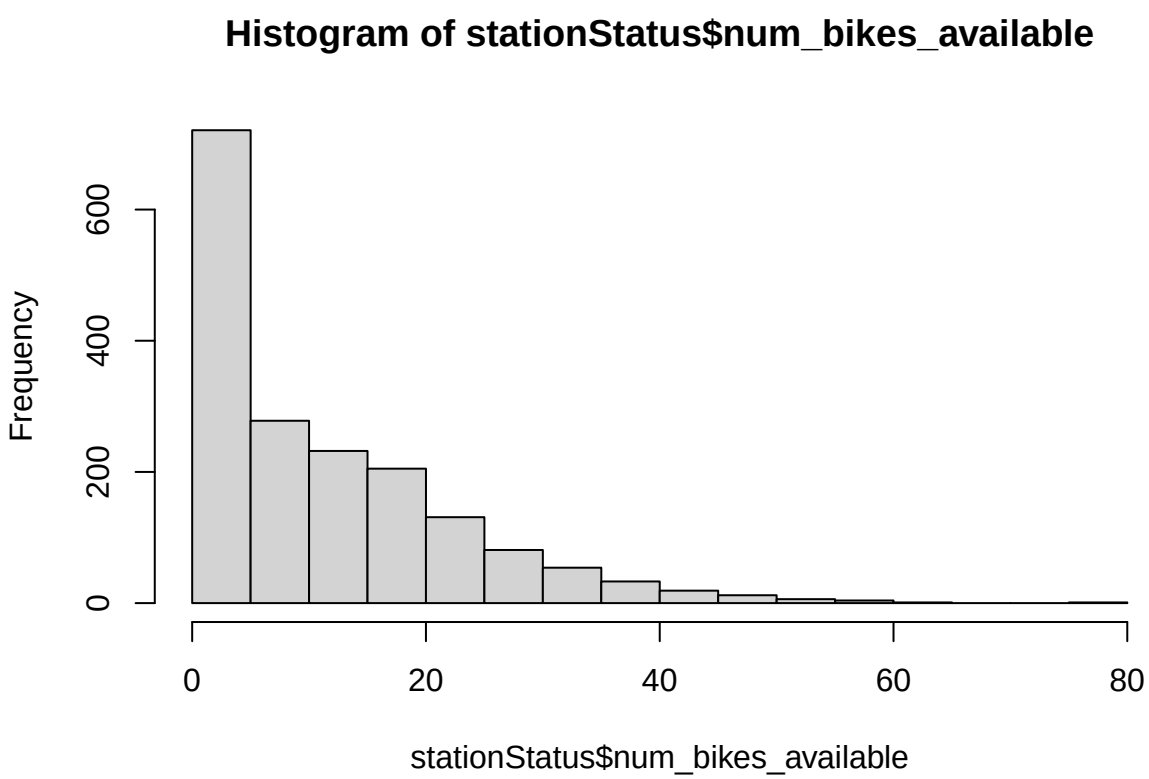
6. Generate a histogram of the number of docks available

```
hist(stationStatus$num_docks_available)
```



7. Generate a histogram of the number of bikes available

```
hist(stationStatus$num_bikes_available)
```



8. How many stations have at least one ebike?

```
sum(stationStatus$num_ebikes_available > 0)
```

```
## [1] 784
```

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

```
atLeastOneBike <- stationStatus[stationStatus$num_ebikes_available > 0,]
# atLeastOneBike
```

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

```
mean(atLeastOneBike$num_docks_available)
```

```
## [1] 14.3699
```

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] 15.83127
```

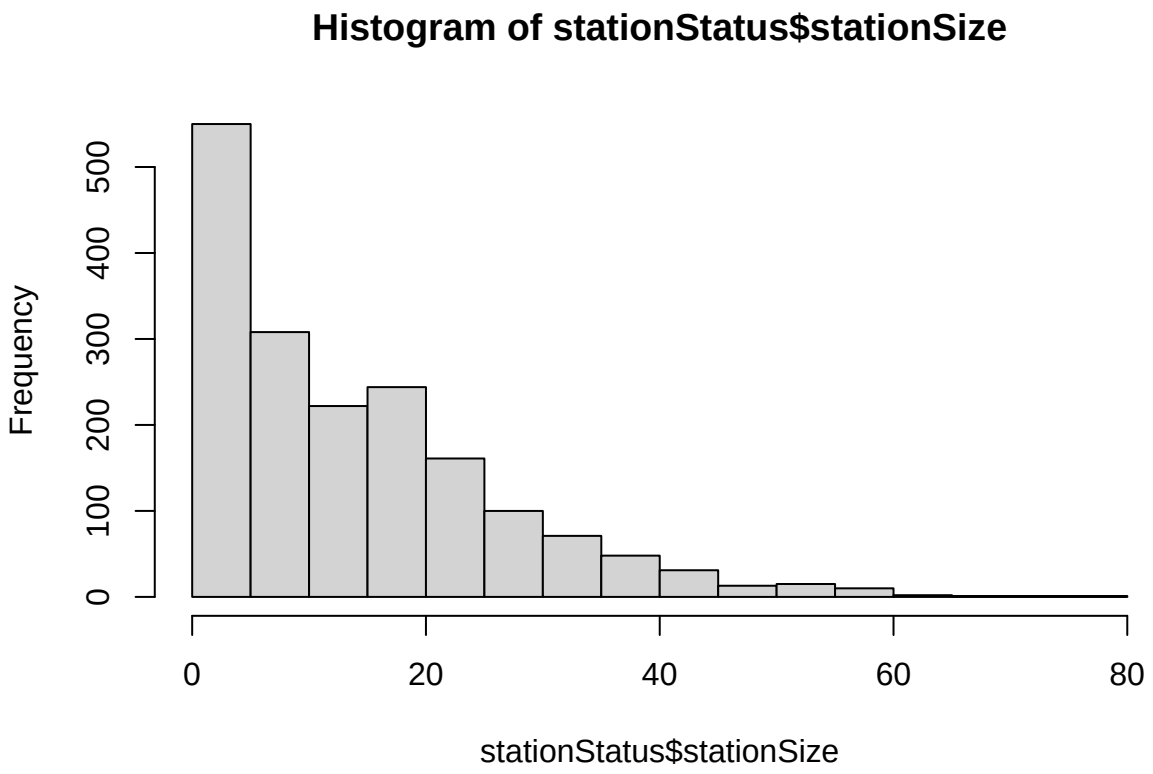
```
# it changes with almost 1
```

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.

```
# install.packages("tidyverse")
library(tidyverse)
```

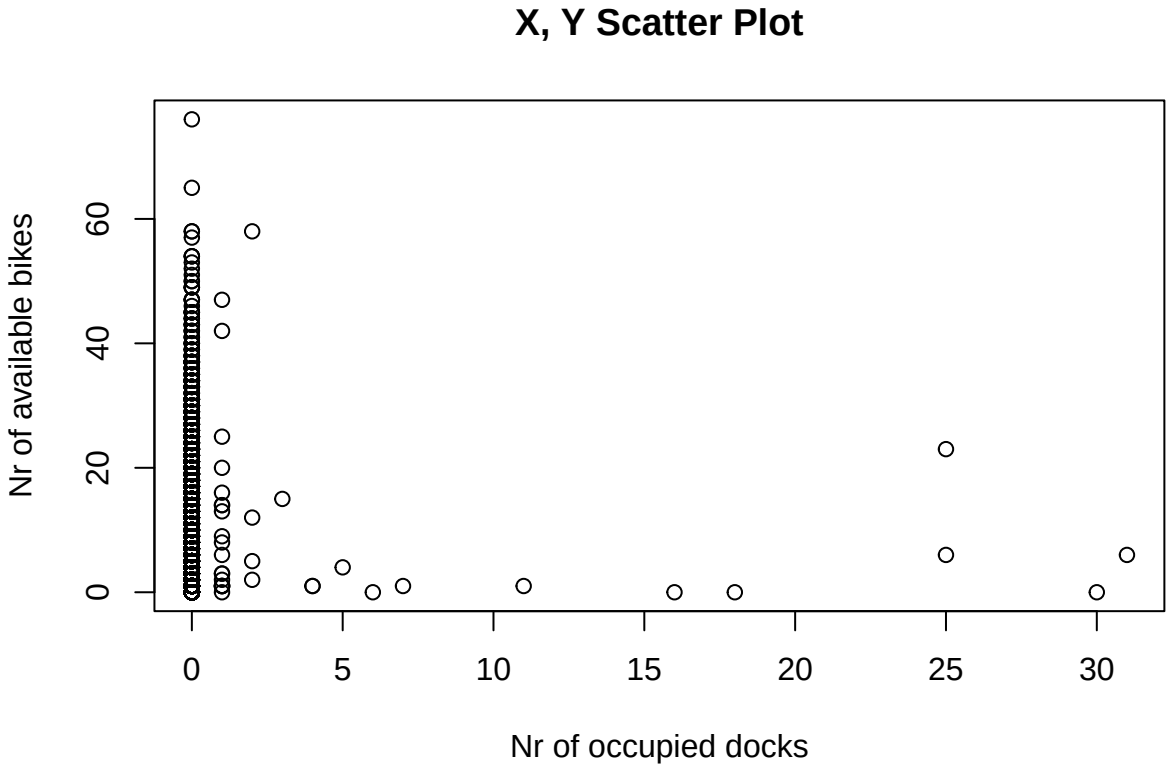
```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.2      v forcats 0.5.2
## -- Conflicts ------------------------------------------ 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()
```

```
stationStatus <- mutate(stationStatus, stationSize <- stationStatus$num_bikes_disabled + stationStatus$n
  stationStatus$num_ebikes_available + stationStatus$num_bikes_available + stationStatus$num_docks_avai
  (stationStatus$num_docks_disabled + stationStatus$num_docks_available))
hist(stationStatus$stationSize)
```



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.

```
plot(stationStatus$num_docks_disabled, stationStatus$num_bikes_available,
     main="X, Y Scatter Plot",
   xlab="Nr of occupied docks", ylab="Nr of available bikes")
```


X, Y Scatter Plot
Nr of available bikes
0
20
40
60
0
5
10
15
20
25
30
Nr of occupied docks