Intro to Data Science - Lab 5

Copyright 2022, Jeffrey Stanton and Jeffrey Saltz Please do not post online.

Week 5 - Obtaining and Using Data from a JSON API

```
# Enter your name here: Hongdi Li
```

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)

```
# it's a file full of data
```

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

```
library('RCurl')
library('jsonlite')

station_link <- 'https://gbfs.citibikenyc.com/gbfs/en/station_status.json'
# save the urllink in the variable station_link

apiOutput <- getURL(station_link)
#save the .json file input in the variable apiOutput
```

```
apiData <- fromJSON(apiOutput)
#transform the .json file's data in to R dataframe type

stationStatus <- apiData$data$stations
#transform and save the data in dictionary of data->station in the variable
stationStatus

cols <- c('num_bikes_disabled','num_docks_disabled', 'station_id',
 'num_ebikes_available', 'num_bikes_available', 'num_docks_available')
#save those columns as a vector in variable cols

stationStatus = stationStatus[,cols]
# shows only the select cloumns data
```

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 same style as the origin .json file
#str(apiData)
# shows in R dataframe style
```

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).

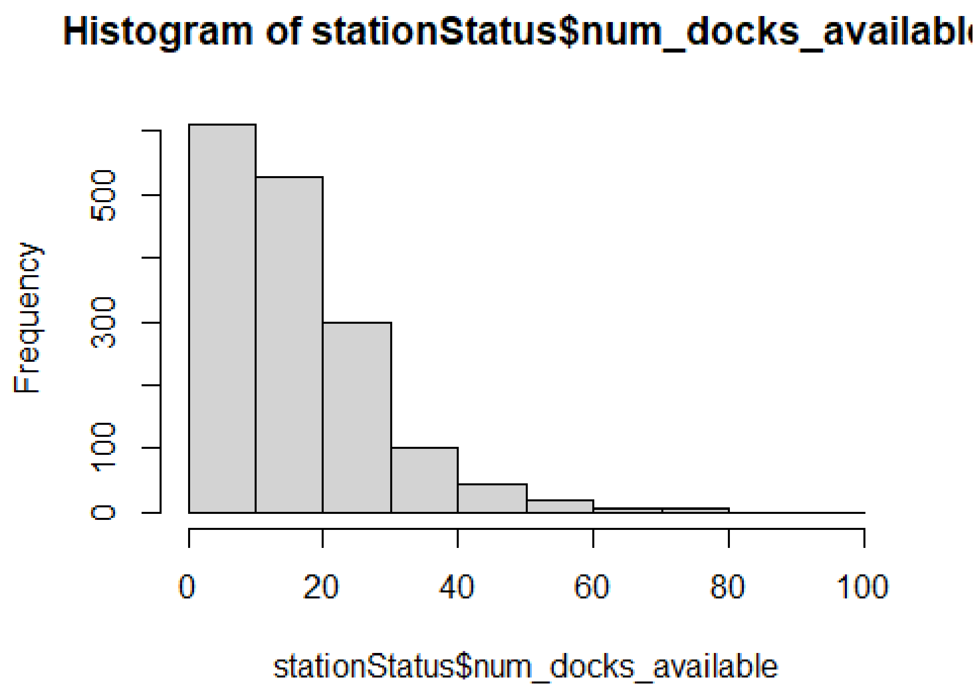
```
#prettify(apiOutput)
# will shows the .json file data in a column/readable style
```

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

```
str(stationStatus)

## 'data.frame':    1619 obs. of  6 variables:
##  $ num_bikes_disabled  : int  3 2 0 4 0 0 1 1 1 3 ...
##  $ 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 4 2 1 2 1 0 0 0 5 ...
##  $ num_bikes_available : int  2 28 24 52 50 50 0 3 54 23 ...
##  $ num_docks_available : int  50 3 3 6 0 3 18 27 1 24 ...

typeof(stationStatus)

## [1] "list"

#it is a list
```

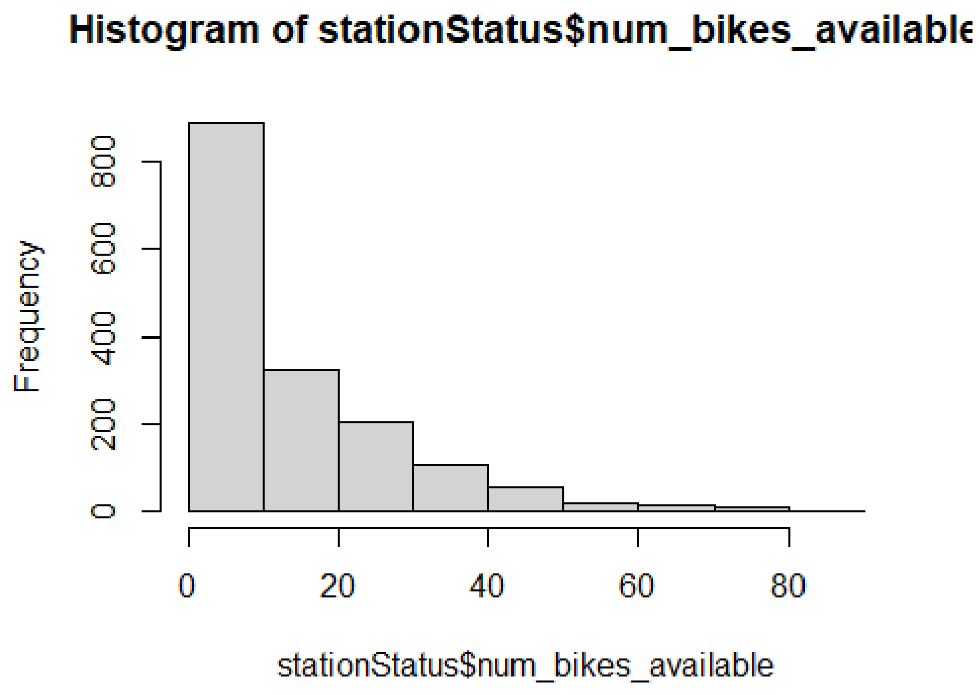
6. Generate a histogram of the number of docks available

```
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?

```
nrow(stationStatus[stationStatus$num_ebikes_available>0,])

## [1] 890
```

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

```
ebike<-stationStatus[stationStatus$num_ebikes_available>0,]
```

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

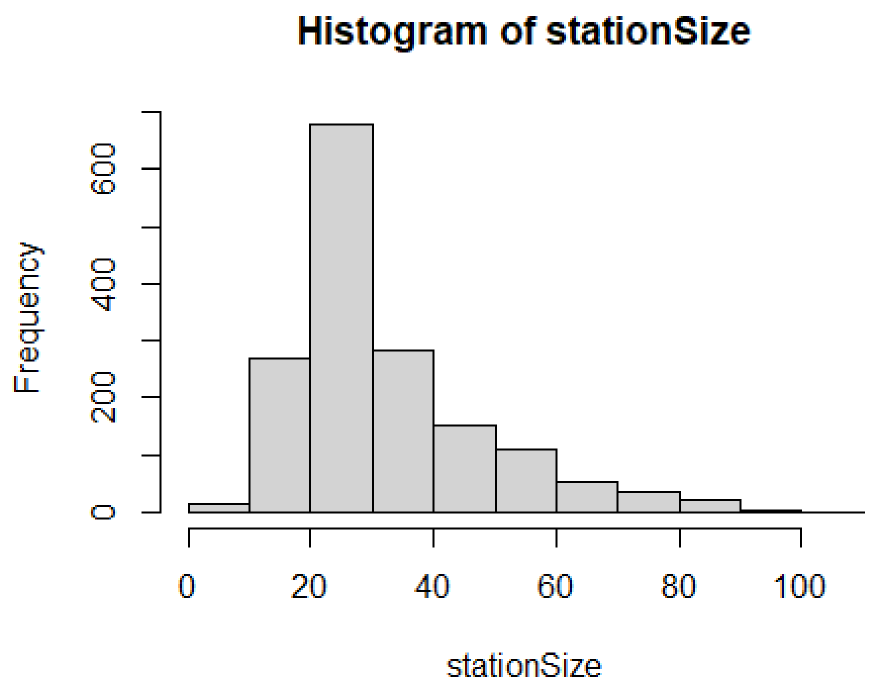
```
mean(ebike$num_docks_available)

## [1] 13.80899
```

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] 15.99876

# one is 12.942, this one is 15.94561,
# the reason is the columns adds and
# the num_docks_available value increased
```

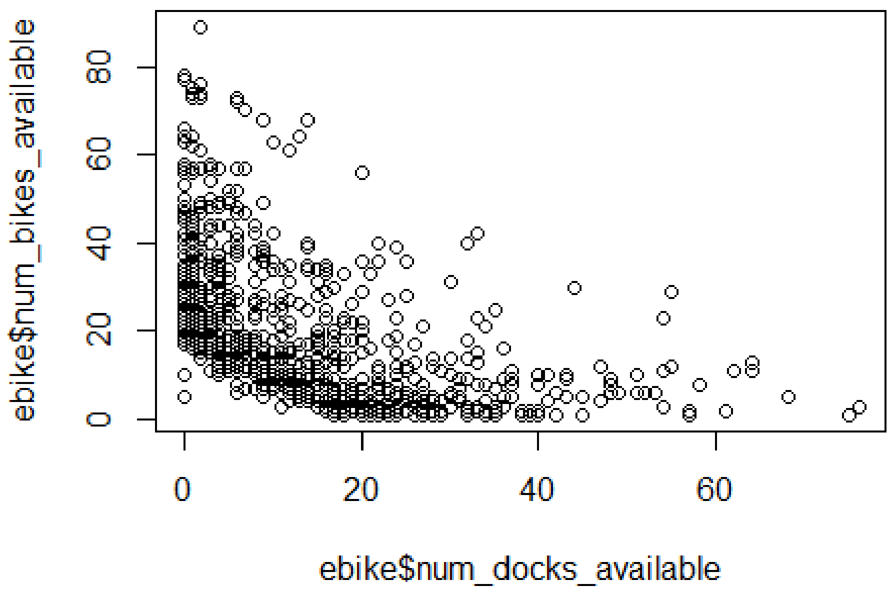
12. Create a new attribute, called ** stationSize **, which is the total number of slots available for a bike (that might, or might not, have a bike in it now). Run a histogram on this variable and review the distribution.

```
stationSize<-
stationStatus$num_bikes_available+stationStatus$num_ebikes_available+stationS
tatus$num_docks_available+stationStatus$num_bikes_disabled+stationStatus$num_
docks_disabled
hist(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(ebike$num_docks_available, ebike$num_bikes_available)
```



```
#num_bike_available >40 and dock aviailable <20
#The distribution of points is relatively dense
```