Cpts 575 Data Science: Assignment 4

Md Muhtasim Billah

10/7/2020

Question 1

1 (a)

Filtering the dataset (using a left join) to display the tail number, year, month, day, hour, origin, and humidity for all flights heading to Tampa International Airport (TPA) on the afternoon of November 1, 2013.

```
# load packages
library(nycflights13)
library(tidyverse)
library(dplyr)
```

```
Humidity = weather %>%
           select(origin, humid)
TampaFlights = flights %>%
               select(dest, dep_time, tailnum, year, month, day, hour, origin) %>%
               filter(dest=='TPA' & month==11 & day==1 & year==2013 & dep_time>=1200) %>%
               select(-dest, -dep_time) %>%
               left_join(Humidity, by = 'origin')
TampaFlights
```

```
## # A tibble: 87,051 x 7
##    tailnum  year month   day  hour origin humid
##    <chr>   <int> <int> <int> <dbl> <chr>  <dbl>
##  1 N580JB   2013    11     1    14 JFK     59.4
##  2 N580JB   2013    11     1    14 JFK     59.4
##  3 N580JB   2013    11     1    14 JFK     59.5
##  4 N580JB   2013    11     1    14 JFK     62.2
##  5 N580JB   2013    11     1    14 JFK     61.6
##  6 N580JB   2013    11     1    14 JFK     64.3
##  7 N580JB   2013    11     1    14 JFK     64.4
##  8 N580JB   2013    11     1    14 JFK     59.5
##  9 N580JB   2013    11     1    14 JFK     59.5
## 10 N580JB   2013    11     1    14 JFK     59.6
## # ... with 87,041 more rows
```

1 (b)

By definition, `anti_join(x, y)` drops all observations in x that have a match in y. To explore the differences between the two joins provided in the questions, we'll run them both. But before that, let's look at the number of rows and columns of these two tables.

Now, we run the first command.

```
join1 = anti_join(flights, airports, by = c("dest" = "faa"))
join1
```

```
## # A tibble: 7,602 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      544            545        -1     1004           1022
##  2  2013     1     1      615            615         0     1039           1100
##  3  2013     1     1      628            630        -2     1137           1140
##  4  2013     1     1      701            700         1     1123           1154
##  5  2013     1     1      711            715        -4     1151           1206
##  6  2013     1     1      820            820         0     1254           1310
##  7  2013     1     1      820            820         0     1249           1329
##  8  2013     1     1      840            845        -5     1311           1350
##  9  2013     1     1      909            810        59     1331           1315
## 10  2013     1     1      913            918        -5     1346           1416
## # ... with 7,592 more rows, and 11 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

Now, let's run the second command.

```
join2 = anti_join(airports, flights, by = c("faa" = "dest"))
join2
```

```
## # A tibble: 1,357 x 8
##    faa   name                      lat    lon   alt    tz dst   tzone
##    <chr> <chr>                   <dbl>  <dbl> <dbl> <dbl> <chr> <chr>
##  1 04G   Lansdowne Airport        41.1  -80.6  1044    -5 A     America/New_Yo...
##  2 06A   Moton Field Municipal A...  32.5  -85.7   264    -6 A     America/Chicago
##  3 06C   Schaumburg Regional      42.0  -88.1   801    -6 A     America/Chicago
##  4 06N   Randall Airport          41.4  -74.4   523    -5 A     America/New_Yo...
##  5 09J   Jekyll Island Airport    31.1  -81.4    11    -5 A     America/New_Yo...
##  6 0A9   Elizabethton Municipal ...  36.4  -82.2  1593    -5 A     America/New_Yo...
##  7 0G6   Williams County Airport  41.5  -84.5   730    -5 A     America/New_Yo...
##  8 0G7   Finger Lakes Regional A...  42.9  -76.8   492    -5 A     America/New_Yo...
##  9 0P2   Shoestring Aviation Air...  39.8  -76.6  1000    -5 U     America/New_Yo...
## 10 0S9   Jefferson County Intl    48.1 -123.    108    -8 A     America/Los_An...
## # ... with 1,347 more rows
```

```
#number of dropped observations (rows) for join1
DroppedObs1 = nrow(flights) - nrow(join1)
DroppedObs1
```

```
## [1] 329174
```

```
#number of dropped observations (rows) for join2
DroppedObs2 = nrow(airports) - nrow(join2)
DroppedObs2
```

```
## [1] 101
```

```r
#checking that the number of columns after the join equals
#the number of columns of the first table
ncol(flights) == ncol(join1)
```

```
## [1] TRUE
```

```r
ncol(airports) == ncol(join2)
```

```
## [1] TRUE
```

By looking at the results, the following facts (differences) are noticed.

- For the first join, all observations from the `flights` table were dropped that have a match in the `airports` table.
- On the other hand, for the second join, all observations from the `airports` table were dropped that have a match in the `flights` table.
- For the first join, the antijoin has been made by the mapping the variable `dest` (from the `flights` table) onto the `faa` variable (from the `airports` table).
- On the other hand, for the second join, the antijoin has been made by the mapping the variable `faa` (from the `airports` table) onto the `dest` variable (from the `flights` table).
- While the number of columns after both joins remain exactly the same as the first table in the join, the number of observations (rows) are different for each join.
- After the first join, `329,174` observations were dropped and the number of remaining observations are `7,602`.
- After the second join, `101` observations were dropped and the number of remaining observations are `1,357`.

1(c)

Selecting the origin and destination airports and their latitude and longitude for all fights in the dataset.

```r
Flights_OriginDest = flights %>%
             select(origin,dest)
Airports_LatLong = airports %>%
                  select(faa,lat,lon)

AllFlights = inner_join(Flights_OriginDest, Airports_LatLong, by = c("dest"="faa"))
AllFlights
```

```
## # A tibble: 329,174 x 4
##    origin dest    lat   lon
##    <chr>  <chr> <dbl> <dbl>
##  1 EWR    IAH    30.0 -95.3
##  2 LGA    IAH    30.0 -95.3
##  3 JFK    MIA    25.8 -80.3
##  4 LGA    ATL    33.6 -84.4
##  5 EWR    ORD    42.0 -87.9
##  6 EWR    FLL    26.1 -80.2
##  7 LGA    IAD    38.9 -77.5
##  8 JFK    MCO    28.4 -81.3
##  9 LGA    ORD    42.0 -87.9
## 10 JFK    PBI    26.7 -80.1
## # ... with 329,164 more rows
```

1(d)

Use group_by and count to get the number of flights to each unique origin/destination combination. Hint: There should be 217 of these total.

```
Num_UniqueFlights = AllFlights %>%
                    group_by(origin,dest) %>%
                    count(sort = TRUE)
Num_UniqueFlights
```

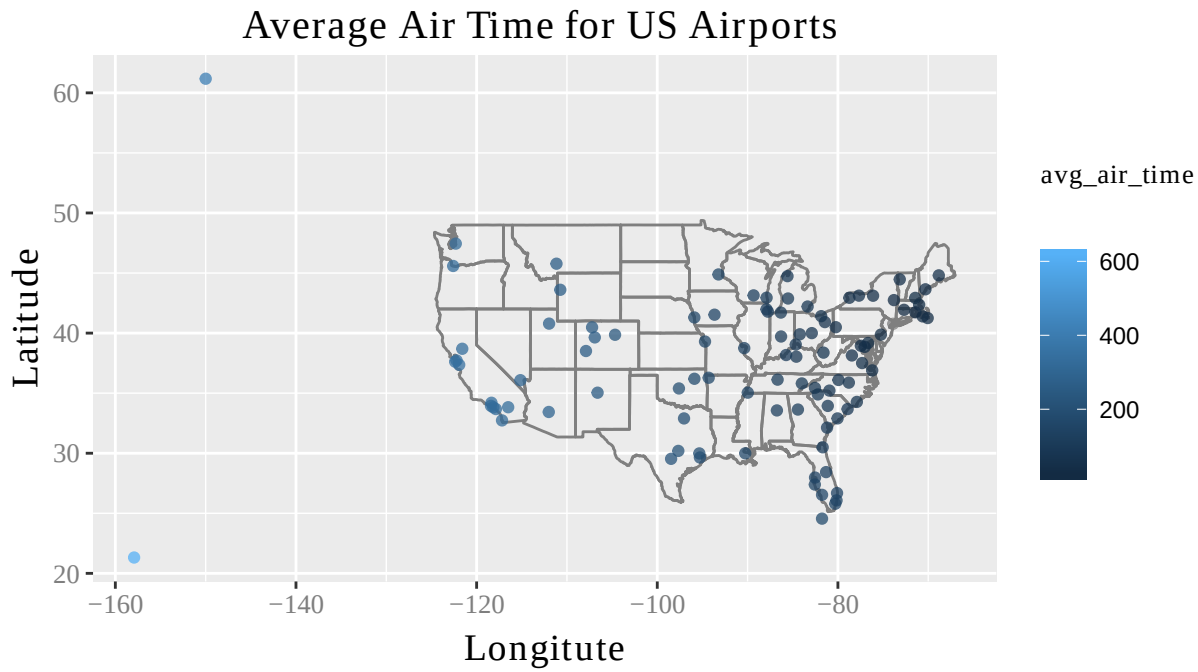
```
## # A tibble: 217 x 3
## # Groups:   origin, dest [217]
##    origin dest      n
##    <chr>  <chr> <int>
##  1 JFK    LAX   11262
##  2 LGA    ATL   10263
##  3 LGA    ORD    8857
##  4 JFK    SFO    8204
##  5 LGA    CLT    6168
##  6 EWR    ORD    6100
##  7 JFK    BOS    5898
##  8 LGA    MIA    5781
##  9 JFK    MCO    5464
## 10 EWR    BOS    5327
## # ... with 207 more rows
```

1(e)

Produce a map that colors each destination airport by the average air time of its incoming flights. Here is a code snippet to draw a map of all flight destinations, which you can use as a starting point. You may need to install the maps packages if you have not already. Adjust the title, axis labels and aesthetics to make this visualization as clear as possible. Hint: You may find it useful to use a different type of join in your solution than the one in the snippet.

```
Flights_DestAirtime = flights %>%
                      group_by(dest) %>%
                      summarise(
                                count = n(),
                                avg_air_time = mean(air_time, na.rm = TRUE)
                               )
Airports_LatLong = airports %>%
                   select(faa,lat,lon)

DestAirports = inner_join(Flights_DestAirtime, Airports_LatLong, by = c("dest"="faa")) %>%
               select(dest, lat, lon, avg_air_time)

DestAirports %>% ggplot(aes(lon, lat)) +
borders("state") + geom_point(aes(color = avg_air_time),alpha = 0.75) + coord_quickmap() +
  ggtitle("Average Air Time for US Airports ") + xlab("Longitute") + ylab("Latitude") +
  labs(fill = "Airtime") +
  theme(
    plot.title = element_text(size=15, hjust=0.5, vjust = 1.5, family = "Palatino",
```

```
                              colour = "Black", margin = margin(10, 0, 0, 0)),
    axis.title.x = element_text(size=14, vjust = -0.3, family = "Palatino",
                                colour = "Black", margin = margin(0, 0, 20, 0)),
    axis.text.x = element_text(size = 10, family = "Palatino", colour = "grey50"),
    axis.title.y = element_text(size=14, vjust = 1.5, family = "Palatino",
                                colour = "Black", margin = margin(10, 0, 10, 10)),
    axis.text.y = element_text(size = 10, family = "Palatino", colour = "grey50"),
    legend.title = element_text(size=10, vjust = -0.0, family = "Palatino",
                                colour = "Black", margin = margin(0, 0, 20, 0))
  )
```

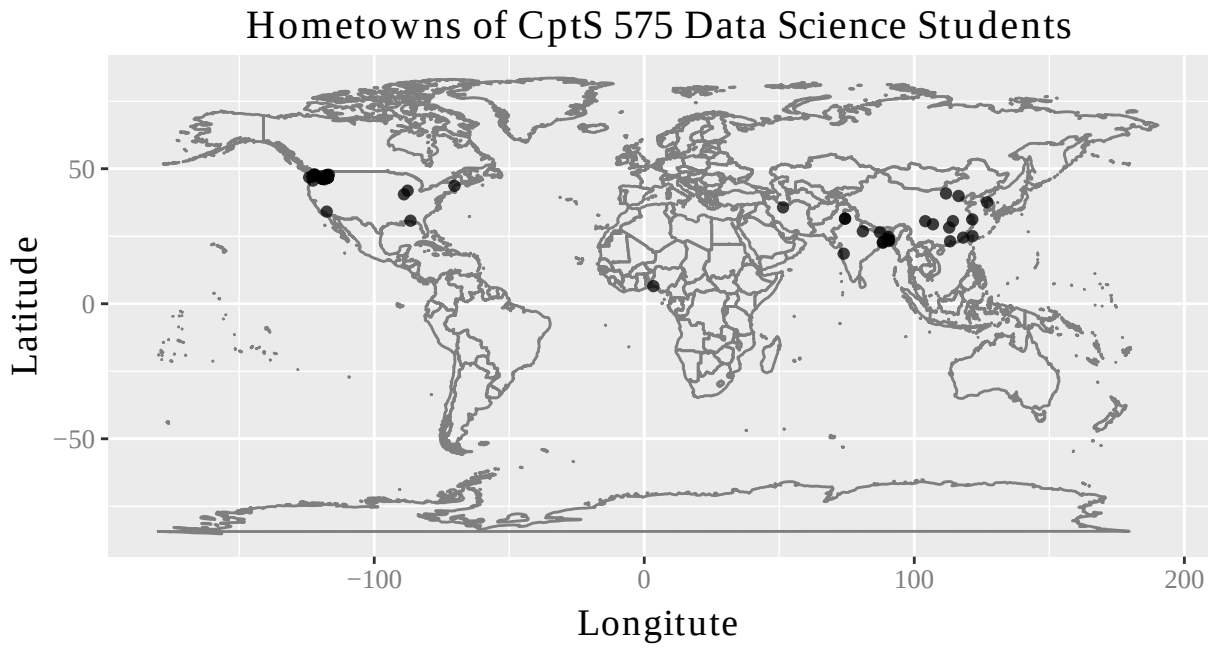


Question 2.

The coordinates for all the cities (lat and long) were collected manually (one by one) to complete the dataset which contain the columns `City Name`, `Country`, `Longitude` and `Latitude`. Then the cities were plotted in the world map.

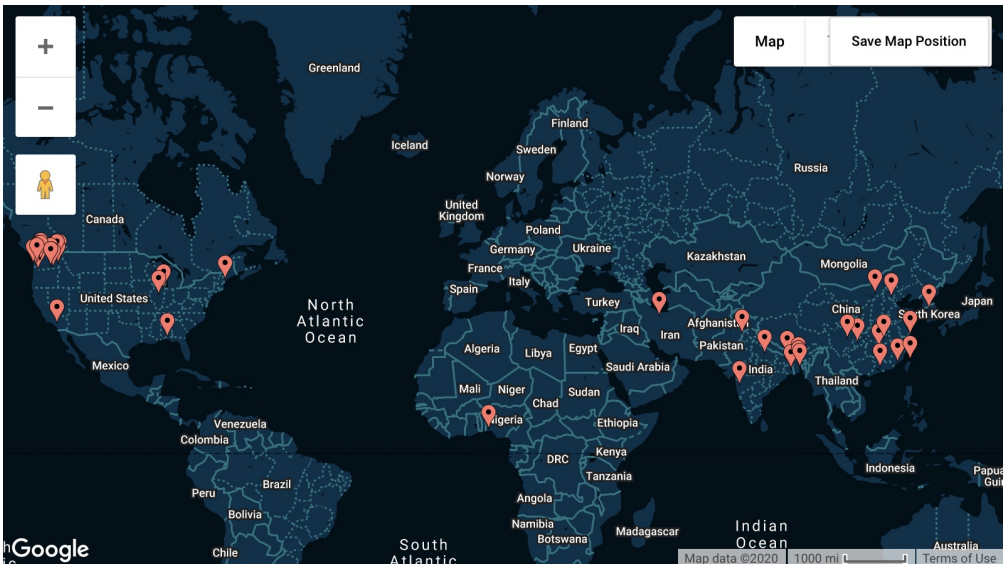
```
mapdata = read.csv("/Users/muhtasim/Desktop/hometowns.csv", header = TRUE)

mapdata %>% ggplot(aes(Longitude, Latitude)) +
borders("world") + geom_point(aes(),alpha = 0.75) + coord_quickmap() +
  #geom_polygon(fill="lightgray", colour = "white") +
  ggtitle("Hometowns of CptS 575 Data Science Students") + xlab("Longitute") + ylab("Latitude") +
  #labs(fill = "Airtime") +
  theme(
    plot.title = element_text(size=15, hjust=0.5, vjust = 1.5, family = "Palatino",
                              colour = "Black", margin = margin(10, 0, 0, 0)),
    axis.title.x = element_text(size=14, vjust = -0.3, family = "Palatino",
                                colour = "Black", margin = margin(0, 0, 20, 0)),
    axis.text.x = element_text(size = 10, family = "Palatino", colour = "grey50"),
```

```
    axis.title.y = element_text(size=14, vjust = 1.5, family = "Palatino",
                                colour = "Black", margin = margin(10, 0, 10, 10)),
    axis.text.y = element_text(size = 10, family = "Palatino", colour = "grey50"),
    legend.title = element_text(size=10, vjust = -0.0, family = "Palatino",
                                colour = "Black", margin = margin(0, 0, 20, 0))
)
```



The same map created using the https://batchgeo.com/features/map-coordinates/ website as below.

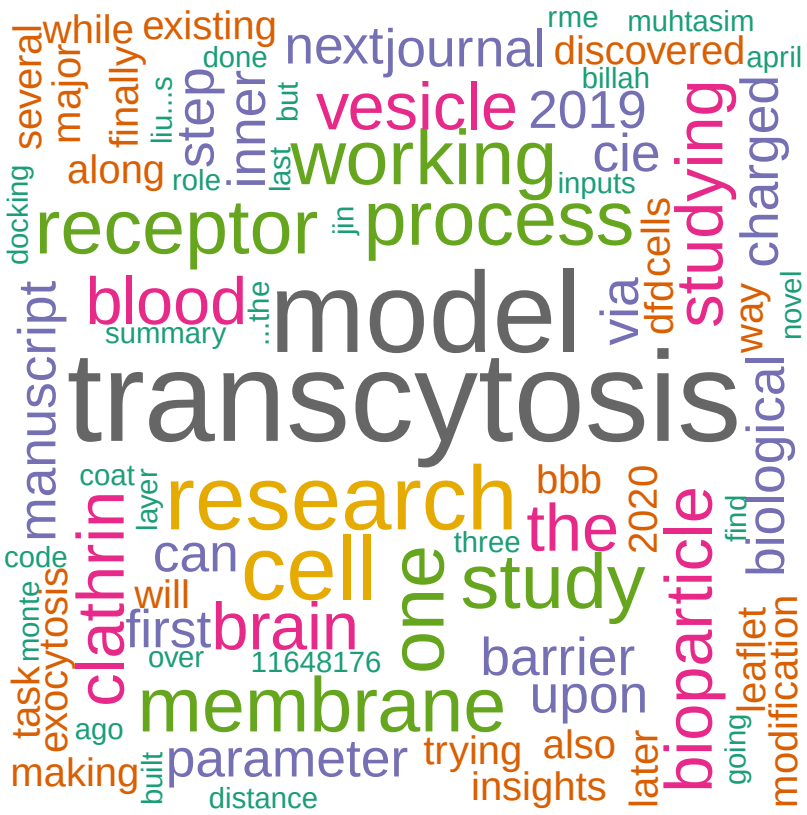
```
knitr::include_graphics("/Users/muhtasim/Desktop/map")
```



Question 3.

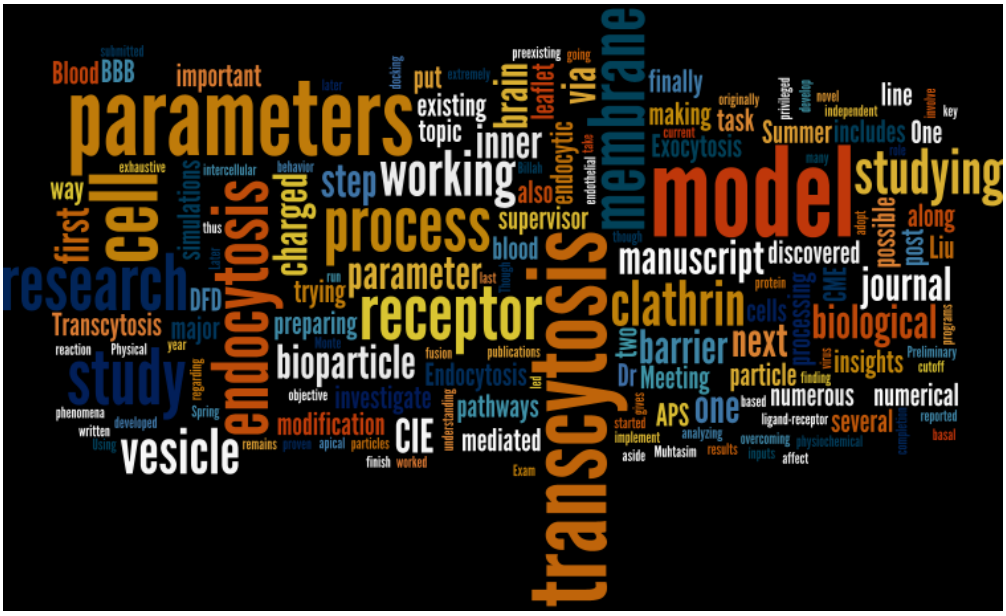
The document chosen for making a word cloud is a two page long summary of my research for the year 2019.

```
library(wordcloud)
library(RColorBrewer)
text = readLines("/Users/muhtasim/Desktop/text.txt")
wordcloud(text, min.freq = 1,
          max.words=200, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2"))
```



Caption: Md Muhtasim Billah's annual Research Review, written in February 2020. (using Wordcloud library in R)

```
knitr::include_graphics("/Users/muhtasim/Desktop/wordle1.png")
```



Caption: Md Muhtasim Billah's annual Research Review, written in February 2020. (using Wordle)