INFO7374 - US On-time Performance Flight Data

Bowei Wang, Dongyue Li, Sarthak Agarwal, Sriram Chandramouli

21 June 2016

Contents

1. Introduction

The flight dataset which is used for our investigation is derived from the Bureau of Transportation Statistics. The Bureau of Transportation Statistics (BTS) was established as a statistical agency in 1992. Its main function is to administer data collection, analysis, and reporting and to ensure the most cost-effective use of transportation-monitoring resources.

As a statistical agency, BTS:

- is policy-neutral-an objective broker for the facts
- covers all of transportation; BTS is cross-modal in nearly everything we do
- does independent data collection and analysis, but BTS also serves all the other modes to help them be more effective and efficient
- sets standards for transportation data
- has special statutory protections (essentially the same as those for Census Bureau and Bureau of Labor Statistics) for the confidentiality of data we collect
- has unique competencies in statistics, economics, information technology, geographic information systems, and transportation

The main purpose of BTS' work is to help advance Department of Transportation (DOT) strategic goals. But they also aim to anticipate future needs and policy issues. Their challenge is to develop data and analyses that are relevant, high quality, timely, comparable, complete, and accessible to strategic goals for transportation statistics.

Analysis Goals

Our goal is to investigate one month of data and characterize flight delays. We are looking at various factors associated with flight data and how these factors have an impact on delay. To begin with, we do our analysis on

- how the departure delay and arrival delay have an influence on flights connecting to particular airport
- how various delays such as carrier delay, weather delay, security delay etc. contribute to the overall delay percentage?
- the time during which more number of delays occur

Using network analysis, we find various centrality measures such as cities with highest number of degree, cities that have highest closeness and farness value, cities that have highest betweenness, overall transitivity measure.

2. Data Profile

In the following two sub sections we understand the data and its structure. We further identify the relevant variables within the Flights dataset that are important for our analysis.

2.1 Dataset description

The airline data consists of several variable that provides information about flights; origin, destination, delays, cancellation etc. The data is available from the Research and Innovative Technology Administration (RITA) website. For our analysis we have chosen the data for January 2016 and selected the following variables. It has approximately 445827 records.

Variable	Type	Unit	Description
YEAR	Number		Year of the flight date
QUARTER	Number		Quarter of the flight date
MONTH	Number		Month of the flight date
DAY_OF_MONTH	Number		Day of the month of the flight date
DAY_OF_WEEK	Number		Day of the week of the flight date
FL_DATE	Date	YYYY-MM-DD	Flight date

Variable	Type	Unit	Description
UNIQUE_CARRIER	Character		Code that represent an airline carrier
FL_NUM	Number		Flight number
ORIGIN	Character		Origin of the flight
DEST	Character		Destination of the flight
CRS_DEP_TIME	Time	local time(hhmm)	Scheduled departure time
DEP_TIME	Time	local time(hhmm)	Actual departure time
DEP_DELAY	Number	minutes	Difference between scheduled and actual departure time
DEP_DELAY_NEW	Number	minutes	Difference between scheduled and actual departure time
DEP_TIME_BLK	Time	hour	Scheduled time between gate in and gate out
TAXI_OUT	Number	minutes	Time between departure from the origin airport gate and wheels off
TAXI_IN	Number	minutes	Time between wheels down and arrival at the destination airport gate
CRS_ARR_TIME	Time	local time(hhmm)	The actual time at which flight is scheduled to arrive
ARR_TIME	Time	local time(hhmm)	Actual Arrival Time of the flight
ARR_DELAY	Number	minutes	Difference between scheduled and actual arrival time
CANCELLED	Number		Value 1 indicates that flight is cancelled
DIVERTED	Number	Value 1 indicates that flight is diverted	
CRS_ELAPSED_TIME	Number	minutes	Time from gate departure time to gate arrival time
ACTUAL_ELAPSED_TIME	Number	minutes	Time taken by the flight from gate departure to gate arrival
AIR_TIME	Number	minutes	Flight time taken to travel
DISTANCE	Number	miles	Distance between airports
CARRIER_DELAY	Number	minutes	Delays due to maintenance or crew problems
WEATHER_DELAY	Number	minutes	Delays due to meteorological conditions
NAS_DELAY	Number	minutes	Delays due to non-extreme weather conditions
SECURITY_DELAY	Number	minutes	Delays or cancellations due to security reasons
LATE_AIRCRAFT_DELAY	Number	minutes	Delay caused by late arrival of previous flight

3. Dataset preparation

To prepare the data for analysis, we:

1. load the necessary libraries,
2. load the flights dataset into a dataframe,
3. create edges dataframe

3.1 Load libraries

The following libraries are used in this report:

- `dplyr`
- `magrittr`
- `ggplot2`
- `igraph`
- `RNeo4j`

3.2 Load the flights dataset

We load the flights data into a dataframe from the csv file downloaded from the server.

```
flight.df =
  read.csv("D:/GRAD_SCHOOL/Summer2016/A2/data/209001837_T_ONTIME/209296560_T_ONTIME_2016_1.csv")
flight.df$X=NULL
dim(flight.df)
```

```
## [1] 445827     37
```

We see that there are 445827 number of flights in the particular month and there are 37 variables selected.

3.3 Create edges dataframe

For network analysis in NEO4j, we create some dataframes that are required for our analysis.

We group by `origin` and `unique_carrier` and use aggregate function to calculate the average departure delay. We write this dataframe into a csv.

```
flight.df %>%
  group_by(ORIGIN, UNIQUE_CARRIER) %>%
  summarize(avg_dep_delay=mean(DEP_DELAY_NEW,na.rm=TRUE))  %>%
  group_by()   %>%
  {.} -> edges_dep_delay.df

write.csv(edges_dep_delay.df,
          "C:/Users/Sarthak/Documents/Neo4j/default.graphdb/import/edges_dep_delay.csv")
```

We group by `origin` and `destination` and aggregate the `CARRIER_DELAY`, `WEATHER_DELAY`, `NAS_DELAY`, `SECURITY_DELAY`, `LATE_AIRCRAFT_DELAY` to calculate their total. We write this dataframe into a csv.

```
flight.df %>%
  group_by(ORIGIN,DEST) %>%
  summarize(carr_delay = sum(CARRIER_DELAY,na.rm=TRUE),
            weather_delay = sum(WEATHER_DELAY,na.rm=TRUE),
            nas_delay = sum(NAS_DELAY,na.rm=TRUE),
            sec_delay = sum(SECURITY_DELAY,na.rm=TRUE),
            aircraft_delay = sum(LATE_AIRCRAFT_DELAY,na.rm=TRUE),
            total_delay = sum(ARR_DELAY_NEW, na.rm=TRUE)) %>%
  filter( total_delay != 0) %>%
  group_by() %>%
  {.} -> edges_delays.df

write.csv(edges_delays.df,
          "C:/Users/Sarthak/Documents/Neo4j/default.graphdb/import/edges_delays.csv")
```

To create this dataframe we group by `origin`, `destination` and `carrier` and calcualte the count. We write this dataframe into a csv.

```
flight.df %>%
  filter(CANCELLED == "1")  %>%
  group_by(ORIGIN, UNIQUE_CARRIER, DEST) %>%
  summarize(count=n()) %>%
  group_by()            %>%
  {.} -> edges_canceled.df

write.csv(edges_canceled.df,
          "C:/Users/Sarthak/Documents/Neo4j/default.graphdb/import/edges_canceled.csv")
```

origin, carrier and origin state [needs editing]

```
#flight1.df %>%
#  group_by(ORIGIN, UNIQUE_CARRIER, ORIGIN_STATE_NM) %>%
#  summarize(avg_taxi_out=mean(TAXI_OUT,na.rm=TRUE))  %>%
#  group_by()   %>%
#  {.} -> edges_taxiin_time.df

#write.csv(edges_taxiin_time.df,
#          "C:/Users/Sarthak/Documents/Neo4j/default.graphdb/import/edges_taxiin_time.csv")
```

4. Variable summaries and visualizations

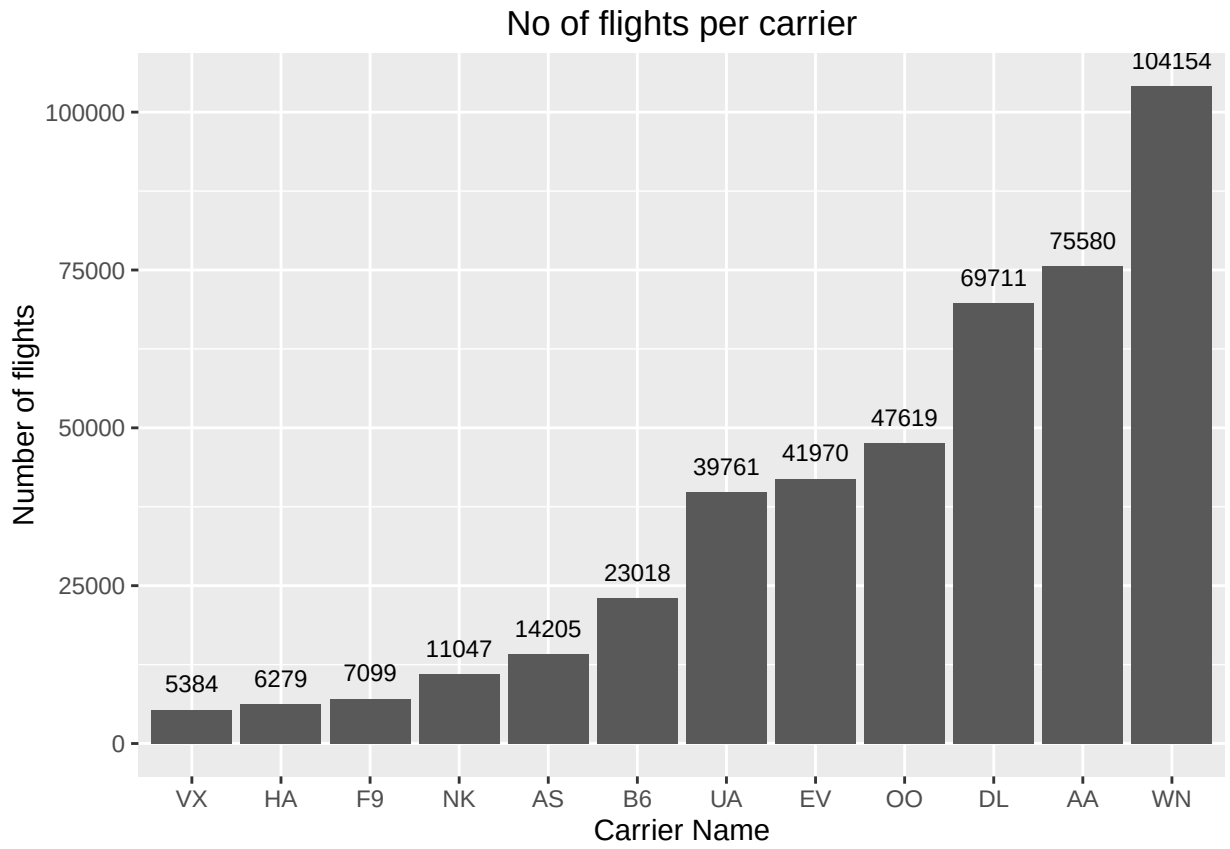
In the next few sections, we examine the significant single and multiple variables for our analysis.

4.1 Number of flights per carrier

We analyze the total number of flights for each carrier to get an understanding on the carriers and how many flights they are operating per month.

We visualize this using a bar graph.

```
carrier_count <- count(flight.df, UNIQUE_CARRIER)
ggplot(carrier_count, aes(x = reorder(UNIQUE_CARRIER, n), y = n)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = n), vjust=-1, position = position_dodge(0.9),size = 3) +
  ggtitle("No of flights per carrier") +
  xlab("Carrier Name") + ylab("Number of flights")
```

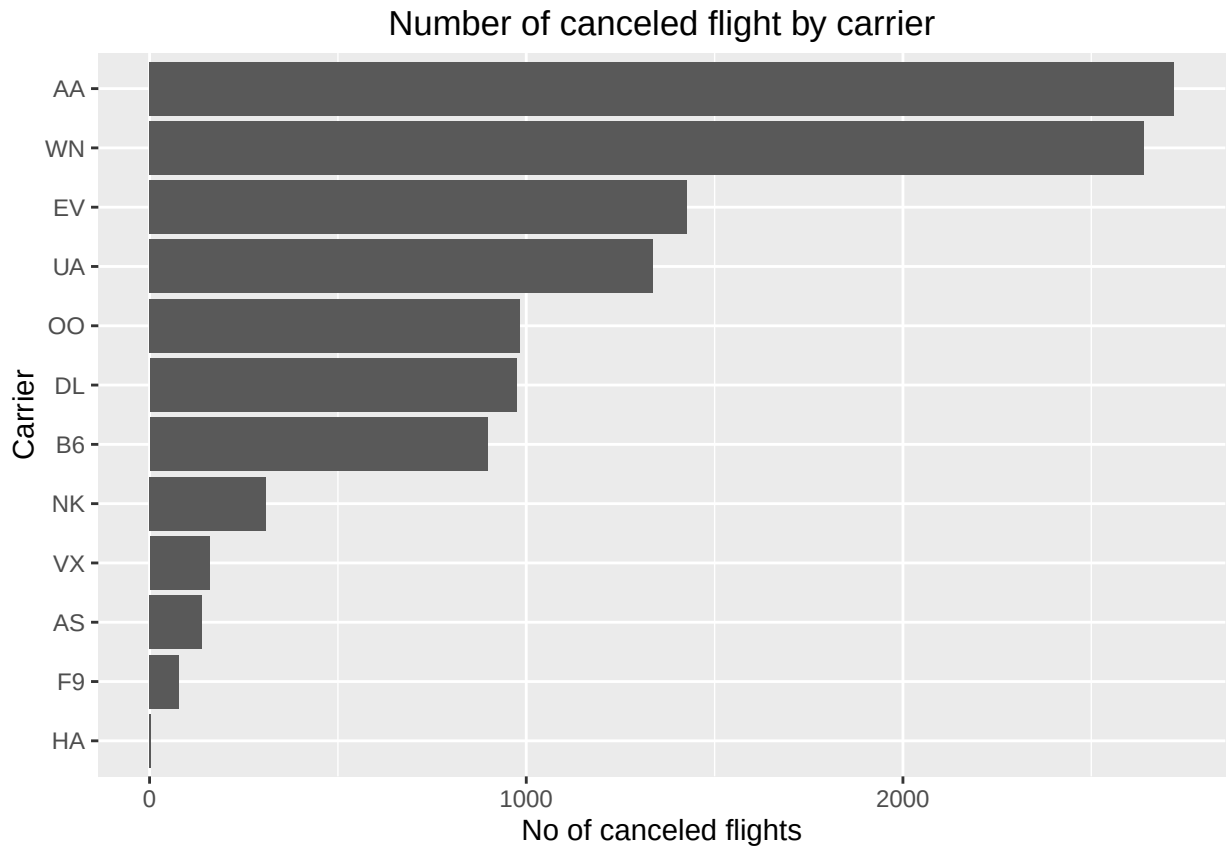


We observe that Southwest Airlines has the largest number of flights followed by American Airlines. Virgin America has the least.

4.2 Number of canceled flights by carrier

We represent the number of canceled flights per carrier using bar graph.

```
flight.df %>%
  subset(CANCELLED == 1) %>%
  ggplot(aes(x = reorder(UNIQUE_CARRIER,
                         UNIQUE_CARRIER, function(x)+length(x))))+
  geom_bar() + coord_flip() +
  ggtitle("Number of canceled flight by carrier") +
  xlab("Carrier") + ylab("No of canceled flights")
```

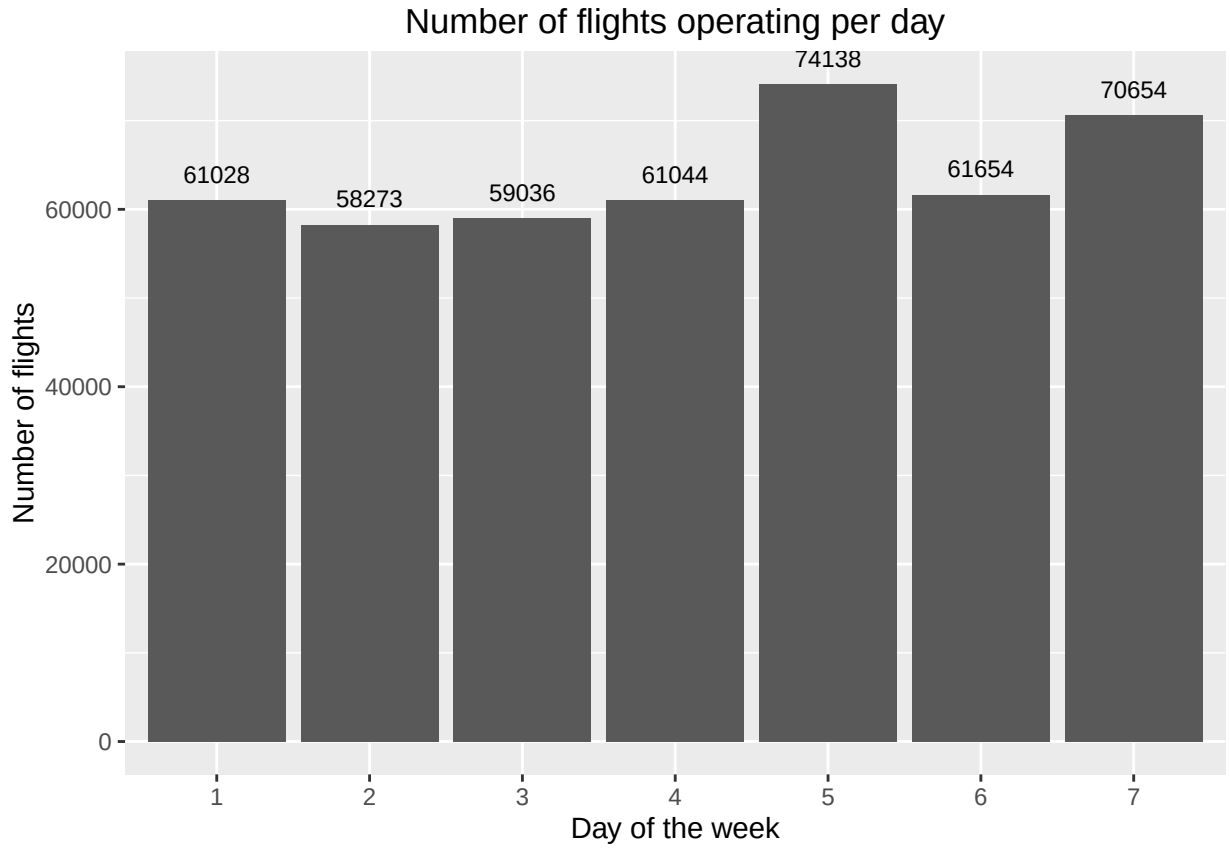


We observe that American Airlines has the maximum number of cancelations followed by Southwest Airlines which is very close to AA. Hawaiian Airlines has the least number of cancelations.

4.3 Number of flights operating per day

To understand the distribution of number of flights over a week, we visualize the number of flights operating per day using bar chart. This will help us relate the effect that number of flights has on flight delay.

```
flight_count <- count(flight.df, DAY_OF_WEEK)
ggplot(flight_count, aes(x = reorder(DAY_OF_WEEK, DAY_OF_WEEK), y = n)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = n), vjust=-1, position = position_dodge(0.9),size = 3) +
  ggtitle("Number of flights operating per day") +
  xlab("Day of the week") + ylab("Number of flights")
```



We observe that higher number of flights are operated on 5th and 7th day of the week i.e. Friday and Sunday.

4.4 Destination airport and arrival delay

We analyze the average delay time and percent delayed for top 10 destination airports with the largest arrive delay time.

First, we calculated average arrive delay time and percent delayed for each destination airport and merged these two data frame by `DEST` by using `aggregate` and `merge`. Here are the top 6 records.

```
##   DEST des_average_delay_time des_percent_delayed(%)
## 1  ABE                      11                  39.77
## 2  ABQ                       8                  28.40
## 3  ABR                       3                  29.03
## 4  ABY                      18                  32.50
## 5  ACT                       4                  16.44
## 6  ACV                      19                  57.14
```

Second, we sorted above data frame by choosing the top 10 records that have the largest average delay time by using `arrange` and `head`.

```
##    DEST des_average_delay_time des_percent_delayed(%)
## 1   EKO                      43                  57.45
## 2   HIB                      27                  48.10
## 3   CMX                      25                  50.00
## 4   OTH                      25                  60.00
```

```
## 5   BGM                    24                40.79
## 6   SBN                    23                38.40
## 7   SFO                    23                44.17
## 8   MQT                    22                38.78
## 9   PAH                    22                37.29
## 10  BTM                    21                36.84
```

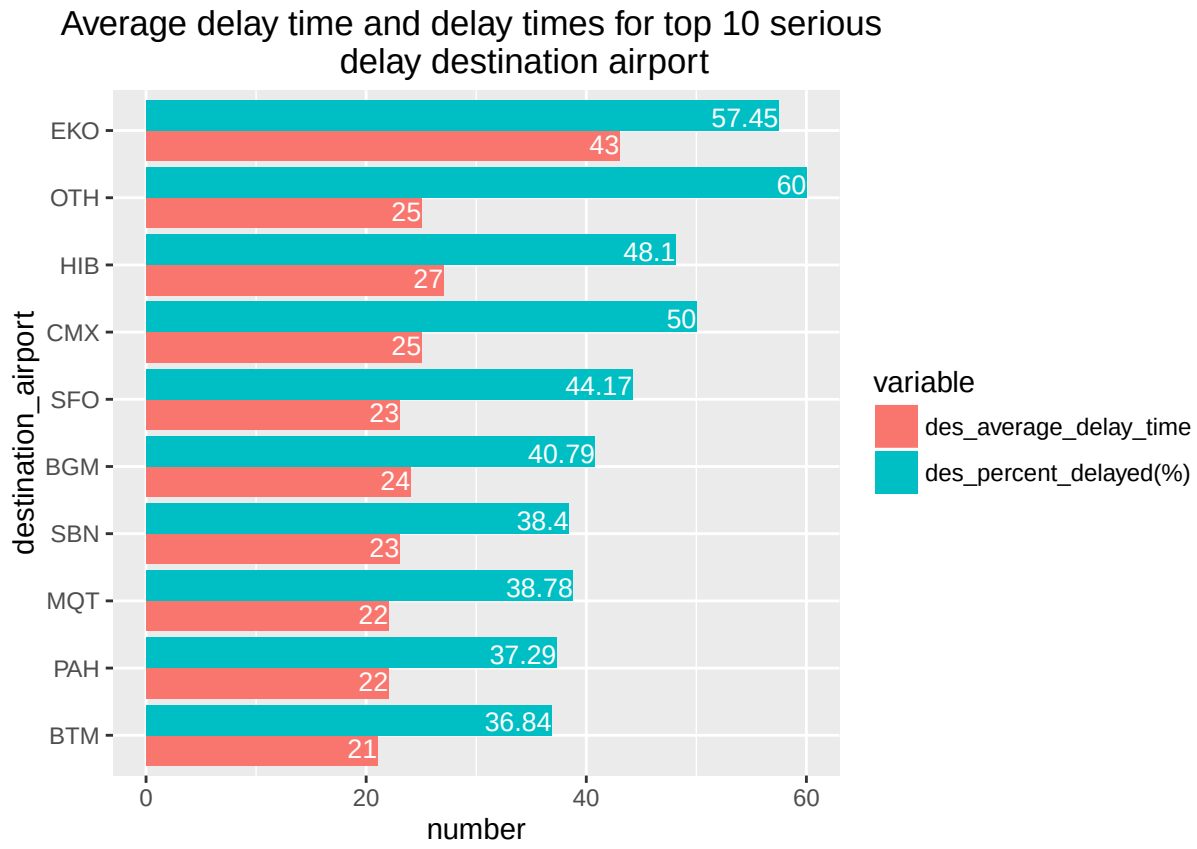
Finally, in order to prepare the data for the graph we changed the structure of above data frame by using `melt`.

```
##    DEST               variable value
## 1   EKO des_average_delay_time 43.00
## 2   HIB des_average_delay_time 27.00
## 3   CMX des_average_delay_time 25.00
## 4   OTH des_average_delay_time 25.00
## 5   BGM des_average_delay_time 24.00
## 6   SBN des_average_delay_time 23.00
## 7   SFO des_average_delay_time 23.00
## 8   MQT des_average_delay_time 22.00
## 9   PAH des_average_delay_time 22.00
## 10  BTM des_average_delay_time 21.00
## 11  EKO des_percent_delayed(%) 57.45
## 12  HIB des_percent_delayed(%) 48.10
## 13  CMX des_percent_delayed(%) 50.00
## 14  OTH des_percent_delayed(%) 60.00
## 15  BGM des_percent_delayed(%) 40.79
## 16  SBN des_percent_delayed(%) 38.40
## 17  SFO des_percent_delayed(%) 44.17
## 18  MQT des_percent_delayed(%) 38.78
## 19  PAH des_percent_delayed(%) 37.29
## 20  BTM des_percent_delayed(%) 36.84
```

Finally, we draw a bar chart showing average delay time and percent delayed for top 10 destination airports that have the largest arrive delay time. Below code represents the bar graph which represents total number along x-axis and destination airport along y-axis.

```
flight.df %>%
  aggregate(ARR_DELAY_NEW ~ DEST, data = .,FUN =. %>% mean(na.rm = TRUE) %>% round()) %>%
  merge(aggregate(ARR_DELAY_NEW ~ DEST, data=flight.df,
                  function(x) round(c(length(which(x > 0)))*100/length(x),2)),
        by = "DEST")%>%
  set_colnames(.,c("DEST","des_average_delay_time","des_percent_delayed(%)")) %>%
  arrange(desc(des_average_delay_time)) %>%
  head(n = 10) %>%
  melt(id.vars = c("DEST")) %>%
  ggplot(aes(x = reorder(DEST, value),
             y = value, fill = variable) ) +
  geom_bar(stat="identity", position = 'dodge') +
  ggtitle("Average delay time and delay times for top 10 serious
          delay destination airport") +
  geom_text(aes(label=value), vjust=0.5, hjust=1, color="white",
            position = position_dodge(0.8), size=3.5)+
  xlab("destination_airport") +
```

```
    ylab("number") +
  coord_flip()
```



We observe that in January 2016, all top 10 airports that have highest arrive delay time have more than 20 minutes arrive delay time and more than 36% flights arrived late. EKO (Elko Regional Airport) as the destination airport has the maximum average delay time of 43 minutes and its percent delayed is also very high that is 57.45% of flights that arrived late at this airport.

4.5 Origin airport and departure delay

We analyze the average delay time and percent delayed for top 10 origin airports that have the largest departure delay time.

First, we calculated average departure delay time and percent delayed for each origin airport and merged these two data frame by `ORIGIN` by using `aggregate` and `merge`. Here are the top 6 records.

```
##   ORIGIN origin_average_delay_time origin_percent_delayed(%)
## 1    ABE                        13                      29.65
## 2    ABQ                         9                      29.89
## 3    ABR                        13                      27.42
## 4    ABY                         9                      18.99
## 5    ACT                         5                      13.10
## 6    ACV                        17                      45.38
```

Second, we sorted above data frame by choosing the top 10 records that have the largest average delay time by using `arrange` and `head`.

```
##    ORIGIN origin_average_delay_time origin_percent_delayed(%)
## 1     COD                        58                      36.36
## 2     MQT                        49                      37.74
## 3     APN                        39                      27.08
## 4     BRD                        39                      19.23
## 5     HIB                        39                      33.33
## 6     CMX                        37                      38.60
## 7     OTH                        26                      47.06
## 8     SWF                        25                      41.54
## 9     FAR                        24                      32.97
## 10    INL                        24                      21.15
```

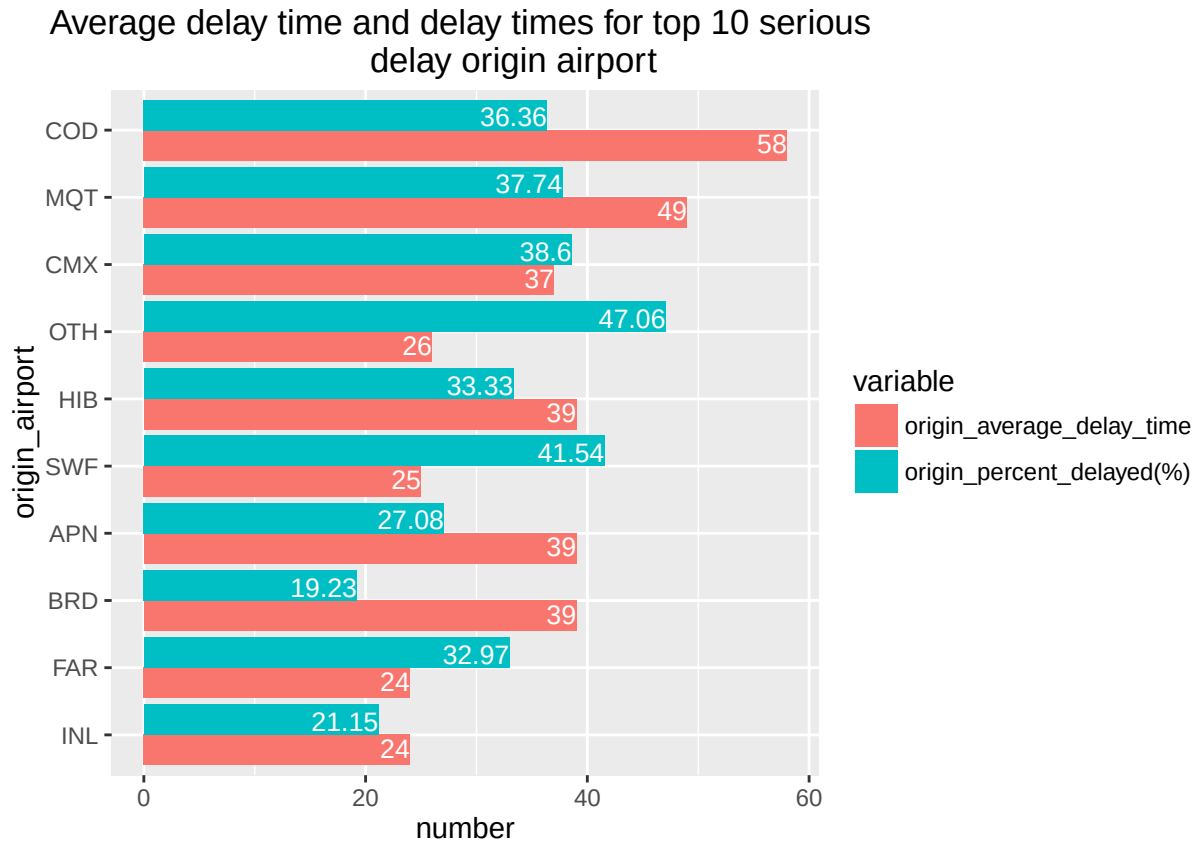
Finally, in order to prepare the data for the graph we changed the structure of above data frame by using `melt`.

```
##    ORIGIN                   variable value
## 1     COD origin_average_delay_time 58.00
## 2     MQT origin_average_delay_time 49.00
## 3     APN origin_average_delay_time 39.00
## 4     BRD origin_average_delay_time 39.00
## 5     HIB origin_average_delay_time 39.00
## 6     CMX origin_average_delay_time 37.00
## 7     OTH origin_average_delay_time 26.00
## 8     SWF origin_average_delay_time 25.00
## 9     FAR origin_average_delay_time 24.00
## 10    INL origin_average_delay_time 24.00
## 11    COD origin_percent_delayed(%) 36.36
## 12    MQT origin_percent_delayed(%) 37.74
## 13    APN origin_percent_delayed(%) 27.08
## 14    BRD origin_percent_delayed(%) 19.23
## 15    HIB origin_percent_delayed(%) 33.33
## 16    CMX origin_percent_delayed(%) 38.60
## 17    OTH origin_percent_delayed(%) 47.06
## 18    SWF origin_percent_delayed(%) 41.54
## 19    FAR origin_percent_delayed(%) 32.97
## 20    INL origin_percent_delayed(%) 21.15
```

Finally, we draw a bar chart showing average delay time and percent delayed for Top 10 origin airports that have the largest departure delay time. Below code represents the bar graph which represents total number along x-axis and origin airport along y-axis.

```
flight.df %>%
  select(ORIGIN, DEP_DELAY_NEW) %>%
  aggregate(.~ ORIGIN, data = .,FUN =. %>% mean(na.rm = TRUE) %>% round()) %>%
  merge(aggregate(DEP_DELAY_NEW ~ ORIGIN, data=flight.df,
                  function(x) round(c(length(which(x > 0)))*100/length(x),2)),
        by = "ORIGIN") %>%
  set_colnames(.,c("ORIGIN","origin_average_delay_time","origin_percent_delayed(%)")) %>%
  arrange(desc(origin_average_delay_time)) %>%
  head(n = 10) %>%
  melt(id.vars = c("ORIGIN")) %>%
  ggplot(aes(x = reorder(ORIGIN,value),
             y = value, fill = variable) ) +
```

```
geom_bar(stat="identity", position = 'dodge') +
ggtitle("Average delay time and delay times for top 10 serious
        delay origin airport") +
geom_text(aes(label=value), vjust=0.5, hjust=1, color="white",
          position = position_dodge(0.8), size=3.5)+
xlab("origin_airport") +
ylab("number") +
coord_flip()
```



We observe that January 2016, all top 10 airports that have highest arrive delay time have more than 24 minutes arrive delay time and more than 19% flights arrived late. COD (Yellowstone Regional Airport) as the origin airport has the maximum average delay time of 58 minutes and its percent delayed is also very high that is 36.36% of flights arrived late at this airport.

4.6 On-Time Arrival Performance

In this section, we analyze the airline service quality performance. We researched the Airline On-Time Statistics and Delay Causes and observed that the displayed numbers are rounded and may not add up to the total. Later, we visualize them on a pie-chart.

First, we calculate 5 delay reasons weighted count. While doing this calculation we considered that a flight is considered delayed when it arrived 15 minutes or more than the scheduled time and when multiple causes are assigned to one delayed flight, each cause is prorated based on delayed minutes it is responsible for.

For example, we got this record for the 5 delay reasons.

```
##   CARRIER_DELAY WEATHER_DELAY NAS_DELAY SECURITY_DELAY LATE_AIRCRAFT_DELAY
## 1             0             0        47              0                  66
```

The weighted count of NAS_DELAY for this flight is:

```
47 / (0+0+47+0+66)
```

```
## [1] 0.4159292
```

Then we calulate all NAS_DELAY weighted count for all the flight and after summing them we get the weighted count of NAS_DELAY that is the percentage of NAS_DELAY in total delay reasons. Similarly, we get the 5 delay reasons weighted count.

```
delay_reason_weighted_count <- flight.df %>%
  filter(ARR_DELAY_NEW >= 15 ) %>%
  select(CARRIER_DELAY,WEATHER_DELAY, NAS_DELAY,
         SECURITY_DELAY,LATE_AIRCRAFT_DELAY) %>%
  mutate(rowSum = rowSums(.)) %>%
  transmute(CARRIER_DELAY_COUNT = CARRIER_DELAY / rowSum,
         WEATHER_DELAY_COUNT = WEATHER_DELAY / rowSum,
         NAS_DELAY_COUNT = NAS_DELAY / rowSum,
         SECURITY_DELAY_COUNT = SECURITY_DELAY / rowSum,
         LATE_AIRCRAFT_DELAY_COUNT = LATE_AIRCRAFT_DELAY / rowSum) %>%
  colSums()
delay_reason_weighted_count
```

```
##       CARRIER_DELAY_COUNT       WEATHER_DELAY_COUNT
##                 22571.6526                2243.1402
##           NAS_DELAY_COUNT      SECURITY_DELAY_COUNT
##                 21692.9216                 153.6523
## LATE_AIRCRAFT_DELAY_COUNT
##                 24220.6333
```

Next, we calculate the count of arrive on time, cancelled and diverted.

```
arrive_on_time_count <- length(which(flight.df$ARR_DELAY_NEW<15))
arrive_on_time_count
```

```
## [1] 362416
```

```
cancelled_count <- length(which(flight.df$CANCELLED==1))
cancelled_count
```

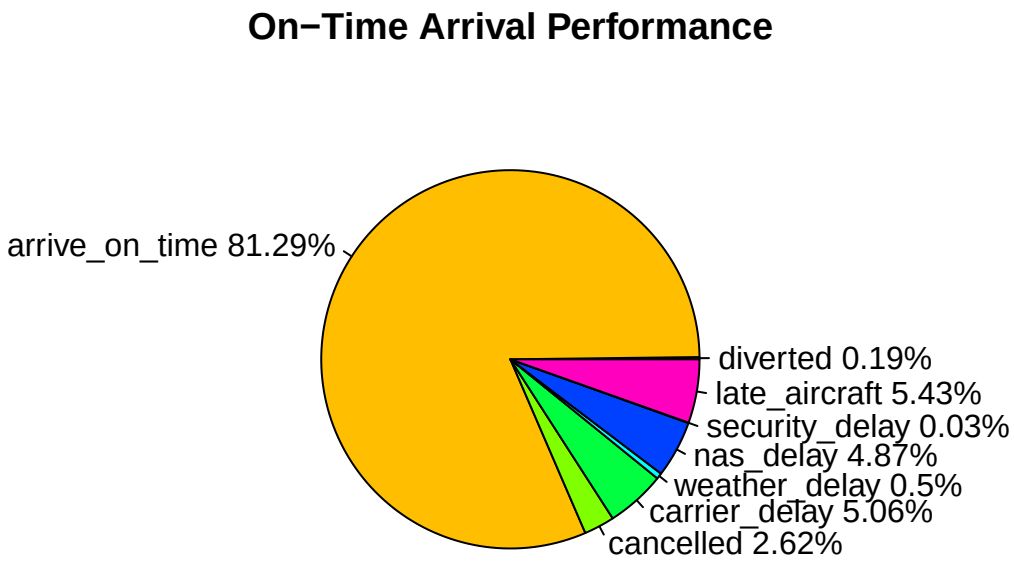
```
## [1] 11665
```

```
diverted_count <- length(which(flight.df$DIVERTED==1))
diverted_count
```

```
## [1] 864
```

Finally, we used above data to draw a pie chart.

```
slices <- c(diverted_count,arrive_on_time_count,cancelled_count,
            delay_reason_weighted_count[1],
            delay_reason_weighted_count[2],
            delay_reason_weighted_count[3],
            delay_reason_weighted_count[4],
            delay_reason_weighted_count[5])
lbls <- c("diverted","arrive_on_time","cancelled",
          "carrier_delay", "weather_delay" , "nas_delay",
          "security_delay", "late_aircraft")
pct <- round(slices/sum(slices)*100,2)
lbls <- paste(lbls, pct) # add percents to labels
lbls <- paste(lbls,"%",sep="") # ad % to labels
pie(slices,labels = lbls, col=rainbow(length(lbls)),
    main="On-Time Arrival Performance")
```



We observe that 81.29% flights arrived on time, 2.62% flights were cancelled and 0.19% flights were diverted. About 15.9% flights are delayed and most of the delays occur due to `late aircraft` which contributes to 5.43% of all flights. It is followed by `carrier delay`, `nas delay`, `weather delay` and `security delay`.

4.7 Departure delay distribution over the day period

We plot the departure delays calculated for each period of the day using a bar graph.

First, we make a copy of `DEP_TIME_BLK` and regroup the `DEP_TIME_PERIOD` set to MIDNIGHT, MORNING, AFTERNOON and EVENING.

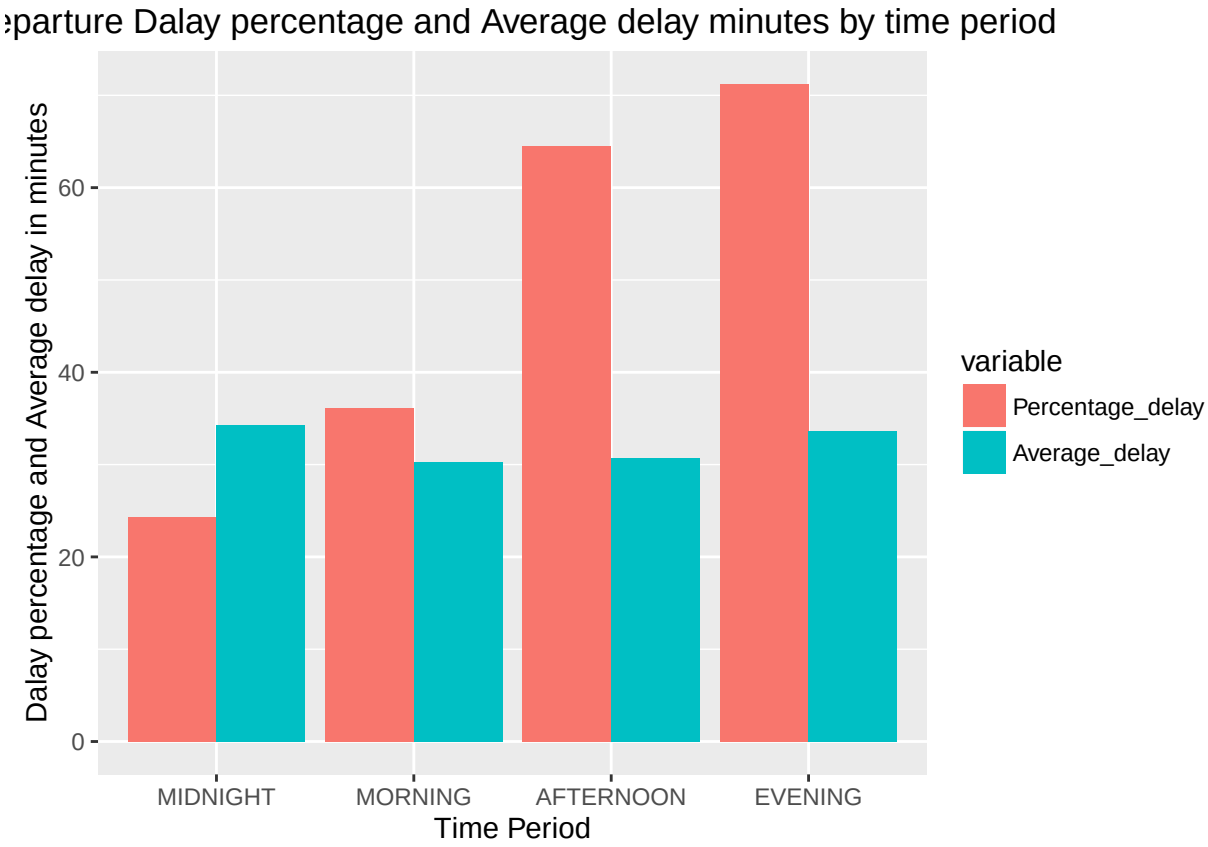
```
flight.df$DEP_TIME_PERIOD <- flight.df$DEP_TIME_BLK

levels(flight.df$DEP_TIME_PERIOD) <- c("MIDNIGHT", "MORNING", "MORNING", "MORNING", "MORNING",
                                       "MORNING", "MORNING", "AFTERNOON", "AFTERNOON", "AFTERNOON"
                                       , "AFTERNOON", "AFTERNOON", "AFTERNOON", "EVENING",
                                       "EVENING","EVENING", "EVENING", "EVENING", "EVENING")
```

We filter out the rows where DEP_DELAY_NEW is NA then select DEP_TIME_PERIOD and DEP_DELAY_NEW and mutate a new column where DEP_DELAY_NEW is greater than 0 or not.

We groupby the dataframe based on DEP_TIME_PERIOD and newly created BOOLEAN and unmelt based on DEP_TIME_PERIOD and BOOLEAN.

Finally, we calculate the average delay time and merge it to the dataframe, rename and melt the dataframe for multi category bar plot.

```
flight.df %>%
  subset(!is.na(DEP_DELAY_NEW)) %>%
  select(DEP_TIME_PERIOD, DEP_DELAY_NEW) %>%
  mutate(BOOLEAN = ifelse(DEP_DELAY_NEW >0, 1, 0)) %>%
  group_by(DEP_TIME_PERIOD, BOOLEAN) %>%
  tally() %>%
  dcast(DEP_TIME_PERIOD ~ BOOLEAN) %>%
  set_colnames(c("DEP_TIME_PERIOD", "total_flight_by_period", "delay_flight")) %>%
  mutate(Quotient = delay_flight * 100 / total_flight_by_period) %>%
  merge(aggregate(DEP_DELAY_NEW ~ DEP_TIME_PERIOD ,
                  data = subset(flight.df, DEP_DELAY_NEW>0), mean)) %>%
  select(DEP_TIME_PERIOD, Quotient, DEP_DELAY_NEW) %>%
  set_colnames(c("Time_Period", "Percentage_delay", "Average_delay")) %>%
  melt(id.vars = c('Time_Period')) %>%
  ggplot(aes(x = Time_Period, y = value, fill = variable) ) +
  geom_bar(stat="identity", position = 'dodge') +
  ggtitle("Departure Dalay percentage and Average delay minutes by time period") +
  xlab("Time Period") +
  ylab("Dalay percentage and Average delay in minutes")
```



We observe that average departure delay is high during midnight and maximum percentage of delays occur during evening.

5. Network analysis

In the next two sections we analyze the flights delay using centrality measures and graph database NEO4j. The flight dataset has details about airport and hence network analysis can be performed by considering airports as nodes and flights between them as edges.

5.1 Centrality Measures

The igraph dataframe interprets first two fields as the nodes and rest as edges. Therefore we arrange the origin city and destination city as first two fields and create the dataframe `flights.df.graph`.

```
flights.df.graph <- data.frame(ORIGIN_AIRPORT = flight.df$ORIGIN,
                               DEST_AIRPORT = flight.df$DEST,
                               flight.df[, c(-10, -12)])
```

We create a directed graph from the above dataframe. The edges of the graph have a direction from origin to destination, hence the property `directed` is set to TRUE.

```
g <- graph.data.frame(flights.df.graph,
                      directed = TRUE)
```

There are `vcount(g)` nodes and `ecount(g)` edges in the graph.

We analyze the network of flights using the following indicators of centrality: 1) Degree 2) Betweenness 3) Farness and closeness 4) Transitivity 5) Eigenvector

5.1.1 Degree

The degree of a vertex is the number of ties that a node has. For our analysis of flight dataset, we calculate the degree of the airport cities and get the number of flights that are connected to them. We sort the result in descending and ascending order to get the top 5 airports with higher and lower number of flights respectively.

```
head(sort(degree(g), decreasing = TRUE))
```

```
##   ATL   ORD   DEN   DFW   LAX   SFO
## 59724 37215 35046 33137 32833 26413
```

```
head(sort(degree(g), decreasing = FALSE))
```

```
## UST ROW BGR ADK PPG OTH
##   6   8  10  18  20  34
```

We observe that ATL (Atlanta) has the highest number of flights and UST (St. Augustine Florida) has the lowest. It can be inferred that Atlanta is the busiest airport whereas St. Augustine Florida is the idlest.

To analyze the flights coming in and going out, we calculate the in-degree and out-degree of the airport nodes.

```
head(sort(degree(g, mode = "out"),
          decreasing = TRUE))
```

```
##   ATL   ORD   DEN   DFW   LAX   SFO
## 29870 18610 17519 16565 16427 13207
```

```
head(sort(degree(g, mode = "out"),
          decreasing = FALSE))
```

```
## UST ROW BGR ADK PPG OTH
##   3   4   6   9  10  17
```

```
head(sort(degree(g, mode = "in"),
          decreasing = TRUE))
```

```
##   ATL   ORD   DEN   DFW   LAX   SFO
## 29854 18605 17527 16572 16406 13206
```

```
head(sort(degree(g, mode = "in"),
          decreasing = FALSE))
```

```
## UST ROW BGR ADK PPG OTH
##   3   4   4   9  10  17
```

We observe that Atlanta has the highest and St. Augustine Florida has the lowest number of flights flown in and flown out.

5.1.2 Closeness and Farness

In this section, farness and closeness centrality measures are calculated for flights dataset. Based on the number of flights that connect the airports, closeness and farness is determined which gives us the understanding of how well the cities are connected to each other by flights. It's important to note that the farness and closeness values calculated here has nothing to do with the actual distance between the airports or the cities.

Closeness centrality measures the ease of access that one airport has to all other airports in the network. We calculate the closeness of an airport which gives us the measure of the degree to which an airport is connected to all other airports. We calculate the closeness of an airport and sort it in decreasing order to get the top 6 airports that have the maximum closeness. In other words we get the airports that are closest to other airports.

```
c <- closeness(g)
head(sort(c, decreasing = TRUE))
```

```
##         ATL         ORD         DEN         MSP         IAH         DFW
## 0.002262443 0.002202643 0.002150538 0.002061856 0.002032520 0.002028398
```

We observe that the airports in Atlanta, Chicago, Denver, Minneapolis, Houston and Dallas are most well connected to other airports in the United States and have relatively simple flight paths to all other airports.

We calculate the farness of an airport and sort it in decreasing order to get the top 6 airports that have the maximum farness. In other words we get the airports that have the higher sum of the distances from the other airports.

```
f <- 1/c
head(sort(f, decreasing = TRUE))
```

```
##  WRG  YAK  SIT  PSG  UST  ADQ
## 1093 1083 1083 1083  972  931
```

From the above results we observe that Wrangell, AK is farthest from all the airports in terms of connectivity with flights.

5.1.3 Diameter of a graph

The diameter of the graph is the length of the longest geodesic. We calculate the diameter of our airport city nodes which will give us the largest route.

```
diameter(g)
```

```
## [1] 5
```

This value depicts that there are total `diameter(g)` number of airports that needs to be traversed to get to the farthest airport.

To get the nodes(cities) which are farthest to each other in terms of airport connectivity we calculate `farthest.nodes`.

```
farthest_nodes <- farthest.nodes(g)
farthest_nodes
```

```
## $vertices
## + 2/294 vertices, named:
## [1] ADQ PIB
##
## $distance
## [1] 5
```

From the above result we can conclude that while travelling from Kodiak airport in Alaska to Hattiesburg-Laurel airport at Mississippi one need to traverse through 5 airports, which is the maximum for any travel in the United States.

We observed earlier that Atlanta has the highest degree. We will compute the shortest paths between Atlanta and other airports. This will give us insight about the airports that are closest to Atlanta. We sort the result in ascending and descending order and get the closest and farthest airport from Atlanta.

```
sp <- shortest.paths(graph=g)['ATL',]
head(sort(sp, decreasing = FALSE))
```

```
## ATL DFW DTW SEA JFK SJC
##   0   1   1   1   1   1
```

```
head(sort(sp, decreasing = TRUE))
```

```
## ADQ BET BRW SCC YAK CDV
##   3   3   3   3   3   3
```

We see that Detroit, Seattle, New York and San Jose are amongst many airports that are connected to Atlanta with just 1 flight. Whereas Kodiak, Bethel, Barrow, Deadhorse, Yakutat and Cordova are amongst many airports that are connected to Atlanta with 3 airports in between.

5.1.4 Clique of a graph

Theoretically cliques are fully connected subgraphs of a graph. In flights dataset we use this technique to model cliques of airports; that is, groups of airports that are all adjacent to each other.

The size of the largest clique in the graph is calculated here.

```
larg_cliques <- largest.cliques(g)
larg_cliques[c(1,2)]
```

```
## [[1]]
## + 23/294 vertices, named:
##  [1] ATL ORD DTW EWR MSP DFW DEN IAH LAS PHX LAX BOS MCO SFO CLT BWI STL
## [18] SEA MIA SLC SAN MSY MCI
##
## [[2]]
## + 23/294 vertices, named:
##  [1] ATL ORD DTW EWR MSP DFW DEN IAH LAS PHX LAX BOS MCO SFO CLT BWI STL
## [18] SEA MIA SLC SAN MSY AUS
```

From the above results we can observe two cliques of airports. All the airports in a clique are connected to each other and one can fly directly between them.

5.1.5 Betweenness

Betweeness centrality measures the significance that a node has in connecting all other nodes in the network. Nodes with high betweeness are critical in connecting nodes throughout the network to each other. Nodes that must be passed through to reach other nodes, or articulation points, generally have high betweeness scores.

In the below code we find the particular airport which is involved in most connections.

```
b <- betweenness(g)
head(sort(b, decreasing = TRUE))
```

```
##       ATL       ORD       DEN       DFW       MSP       SEA
## 23848.347 12258.216 11955.895  9690.446  8006.950  6064.213
```

The result shows that ATL (Atlanta) has the highest betweenness score, meaning it is involved in more connections between various nodes of airport. It is interesting to note the high difference between the betweenness score of Atlanta and ORD (O'Hare Chicago Airport).

5.1.6 Transitivity

We calculate the transitivity measure and measure the probability that the adjacent airport of an airport are connected.

```
transitivity(g)
```

```
## [1] 0.3421134
```

For any network, transitivity index range between 0.3 and 0.6 is considered to be ideal. The probability value of 0.34 obtained above is a good value and indicates that airports are well connected to each other.

5.1.7 Eigen vector centrality

Eigenvector centrality is a measure of the influence of a node in a network. It measures the approximate importance of each node in a graph. Its function is similar to that of Google's PageRank algorithm. It assigns relative scores to all nodes in the network based on the concept that connections to high scoring nodes contribute more to the score than connections to low scoring nodes.

```
head(sort(evcent(g)$vector, decreasing = TRUE))
```

```
##       ATL       LAX       ORD       DEN       DFW       SFO
## 1.0000000 0.9631554 0.8543206 0.7984406 0.7708810 0.7697084
```

Eigenvector centrality scores correspond to the values of the first eigenvector of the graph adjacency matrix; these scores may, in turn, be interpreted as arising from a reciprocal process in which the centrality of each actor is proportional to the sum of the centralities of those actors to whom he or she is connected. In general, vertices with high eigenvector centralities are those which are connected to many other vertices which are, in turn, connected to many others (and so on). Here, Chicago has the highest eigenvector centrality as compared to other cities, implying that it is connected with other highly connected nodes.

5.2 Analysis in graph database

We use the information obtained till now and answer the following questions with cypher queries of the graph database. 1. What are the airports and carriers that face the maximum flight delays? 2. Analyze the reasons for delay of flights 3. Analysis on cancelled flights

In this section we load the flight dataset into Neo4j graph database. We create nodes and then build relation between them. Finally we run cypher queries to get our desired results.

We start the graph database

```
graph1 = startGraph("http://localhost:7474/db/data/",
                    username = "neo4j",
                    password = "sarthak12")
#clear(graph1)
```

5.2.1 Departure delay analysis for origin airports and carrier

In this section, we calculate the origin airports and carriers that have the highest average departure delays. We compare them with the important airports we obtained from centrality measures and observe interesting findings.

There are 3 subsections for this analysis: 1. Environment setup - Create nodes and build relationship between the nodes. 2. Run cypher queries

1. We use the `edges_dep_delay.df`, `edges_dep_delay.df` and `edges_delays.df` dataframes created earlier and make the origin airport, carrier and destination airport as nodes.

```
addConstraint(graph1, "origin_airport", "name")
lapply(unique(c(as.character(edges_dep_delay.df$ORIGIN))),
       function(a.airport) {
         createNode(graph1, "origin_airport", name=a.airport)
       }
)

addConstraint(graph1, "Carrier", "name")
lapply(unique(c(as.character(edges_dep_delay.df$UNIQUE_CARRIER))),
       function(a.carrier) {
         createNode(graph1, "Carrier", name=a.carrier)
       }
)

addConstraint(graph1, "dest_airport", "name")
lapply(unique(c(as.character(edges_delays.df$DEST))),
       function(a.airport) {
         createNode(graph1,
                    "dest_airport",
                    name=a.airport)
       }
)
```

We create the relationships between the nodes using the below cypher queries. First, we create edges between the origin and carrier nodes and set average departure delay as the edge property. Second, we create edges

between origin and destination airports and set the sum of `carrier_delay`, `weather_delay`, `nas_delay`, `security_delay` and `aircraft_delay` as the property of edges. And third, we create edge property as the number of flights cancelled between origin, destination and carrier.

```
cypher(graph1,
       "LOAD CSV WITH HEADERS FROM 'file:/edges_dep_delay.csv' AS row
       MATCH (origin:origin_airport {name: row.ORIGIN}),
       (carrier:Carrier {name: row.UNIQUE_CARRIER})
       CREATE (origin)-[:delayed_by {dep_delay: row.avg_dep_delay}]->(carrier)
       ")

cypher(graph1,
       "LOAD CSV WITH HEADERS FROM 'file:/edges_delays.csv' AS row
       MERGE (origin:origin_airport {name: row.ORIGIN})
       MERGE (destination:dest_airport {name: row.DEST})
       MERGE (origin)-[r:delayed_by]->(destination)
       SET  r.carr_delay=row.carr_delay, r.weather_delay=row.weather_delay,
       r.nas_delay=row.nas_delay, r.sec_delay=row.sec_delay,
       r.aircraft_delay=row.aircraft_delay, r.total_delay=row.total_delay
       ")

cypher(graph1,
       "LOAD CSV WITH HEADERS FROM 'file:/edges_canceled.csv' AS row
       MATCH (origin:origin_airport {name: row.ORIGIN}),
       (destination:dest_airport {name: row.DEST}),
       (carrier:Carrier {name: row.UNIQUE_CARRIER})
       CREATE (origin)-[:cancelled_by {cancellation: row.count}]->(carrier)
       CREATE (origin)-[:cancelled_by {cancellation: row.count}]->(destination)
       CREATE (origin)-[:operated_by {carrier: row.UNIQUE_CARRIER}]->(carrier)
       ")
```

2. After setting up the Neo4j environment, we do our analysis by running the following cypher queries.

Query 1: The query below gives the maximum average departure delay that a carrier has in a particular airport.

```
q1.df = cypher(graph1,
       "MATCH (oa:origin_airport)-[d:delayed_by]->(c:Carrier)
        RETURN oa.name AS Airport, c.name  AS Carrier, d.dep_delay As Departure_Delay
        ORDER BY Departure_Delay DESC
        LIMIT 10")
head(q1.df)
```

```
##   Airport Carrier  Departure_Delay
## 1     SLC      OO  9.9843315643164
## 2     TPA      DL 9.97950377562028
## 3     GPT      DL 9.97260273972603
## 4     DTW      UA 9.96703296703297
## 5     PHL      EV 9.94736842105263
## 6     HSV      EV 9.94160583941606
```

We observe that SLC (Salt Lake City Airport) and OO (SkyWest airlines) has the highest average departure delay of 9.98 minutes.

Query 2: The below query gives us the carriers that have the maximum average departure delay.

```
q2.df = cypher(graph1,
       "MATCH ()-[d:delayed_by]->(c:Carrier)
        RETURN c.name  AS Carrier,
        AVG(toFloat(d.dep_delay)) As Departure_Delay
        ORDER BY Departure_Delay DESC
        LIMIT 10")
head(q2.df)
```

```
##   Carrier Departure_Delay
## 1      NK        18.56820
## 2      B6        15.96185
## 3      VX        15.83741
## 4      OO        15.22071
## 5      DL        11.71215
## 6      AA        10.84875
```

We observe that NK (Spirit airlines), B6 (JetBlue) and VX (Virgin America) have the maximum average departure delay. It is interesting to note that these 3 carriers are not in top 5 maximum average departure delay that a carrier has in a particular airport.

Query 3: In the below query, we calculate airports that have the higher average departure delay.

```
q3.df = cypher(graph1,
       "MATCH (oa:origin_airport)-[d:delayed_by]->()
        RETURN oa.name  AS Airports,
        AVG(toFloat(d.dep_delay)) As avg
        ORDER BY avg DESC
        LIMIT 10")
head(q3.df)
```

```
##   Airports      avg
## 1      COD 58.21212
## 2      MQT 48.90566
## 3      BRD 39.13462
## 4      HIB 38.95062
## 5      APN 38.70833
## 6      FNT 36.67645
```

From the above results, we conclude that COD (Yellowstone Regional Airport) airport has the highest average departure delay. It is interesting to note that ATL (Atlanta airport) which has the highest degree centrality is not in the top 6 list obtained here. In the next query we calculate the average departure delay of ATL.
Query 4:

```
q4.df = cypher(graph1,
       "MATCH (oa:origin_airport {name:'ATL'} )-[d:delayed_by]->()
        RETURN oa.name  AS Airports,
        AVG(toFloat(d.dep_delay)) As avg")
head(q4.df)
```

```
##   Airports      avg
## 1      ATL 11.17716
```

We see that ATL being a very busy airport has only 11.1 minutes of average departure delay, which is low as compared to the maximum average departure delay of 58 minutes.

Query 5: In the below query we calculate the top 10 delays between source and destination

```
q5.df = cypher(graph1,
               "MATCH (oa:origin_airport)-[r:delayed_by]->(da:dest_airport)
                RETURN oa.name  AS Origin,
                        avg(tofloat(r.sec_delay)) AS Security_Delay
                ORDER BY (Security_Delay) DESC
                LIMIT 10")
head(q5.df)
```

```
##   Origin Security_Delay
## 1    ADK      80.000000
## 2    STX      12.500000
## 3    JFK      12.188679
## 4    PHX      10.701299
## 5    ACY       9.666667
## 6    FCA       9.333333
```

We observe that ADK (Adak Island Airport) has the maximum average security delay.

Query 6: In the below query, we analyze the percentage distribution of all the delay types for origin airports.

```
q6.df = cypher(graph1,
               "MATCH (oa:origin_airport)-[r:delayed_by]->(da:dest_airport)
                RETURN oa.name  AS Origin,
                sum(tofloat(r.carr_delay)) / sum(tofloat(r.total_delay)) * 100
                AS Carrier_Delay_per,
                sum(tofloat(r.weather_delay)) / sum(tofloat(r.total_delay)) * 100
                AS weather_delay_per,
                sum(tofloat(r.nas_delay)) / sum(tofloat(r.total_delay)) * 100
                AS NAS_Delay_per,
                sum(tofloat(r.sec_delay)) / sum(tofloat(r.total_delay)) * 100
                AS Security_Delay_per,
                sum(tofloat(r.aircraft_delay)) / sum(tofloat(r.total_delay)) * 100
                AS Aircraft_Delay_per,
                sum(tofloat(r.total_delay)) AS Total_delay
                ORDER BY (Total_delay) desc
                LIMIT 10")
head(q6.df)
```

```
##   Origin Carrier_Delay_per weather_delay_per NAS_Delay_per
## 1    ATL          38.60297         12.718115      16.06739
## 2    ORD          33.66013          9.759089      18.59472
## 3    SFO          26.89582          1.226817      12.69777
## 4    LAX          30.26840          1.934773      21.98615
## 5    DEN          31.83012          3.065073      22.87610
## 6    LAS          25.46564          1.048951      19.33492
##   Security_Delay_per Aircraft_Delay_per Total_delay
## 1        0.006637643           21.57820      256115
## 2        0.112187873           30.01109      238885
## 3        0.021290689           52.20950      211360
```

```
## 4        0.150983530           35.64968      210619
## 5        0.026652807           30.65206      150078
## 6        0.077098917           44.12696      147862
```

We see that carrier delay and aircraft delay are the main reasons for which the flights are delayed; that is, carriers are the main reasons for delay of flights. In the next query we see which carriers have canceled the most number of flights.

Query 7:

```
q7.df = cypher(graph1,
               "MATCH ()-[ca:cancelled_by]->(c:Carrier)
                WITH SUM(toFloat(ca.cancellation)) As total
                MATCH ()-[ca:cancelled_by]->(c:Carrier)
                RETURN c.name  AS Carrier,
                100.0 * SUM(toFloat(ca.cancellation)) / total AS percent_cancellation
                ORDER BY percent_cancellation DESC")
head(q7.df)
```

```
##   Carrier percent_cancellation
## 1      AA            23.317617
## 2      WN            22.631805
## 3      EV            12.233176
## 4      UA            11.453065
## 5      OO             8.435491
## 6      DL             8.349764
```

We observe that AA (American Airlines) has the highest percentage of cancellations.

In the next query, we calculate which route is the worst in terms of total delay time. *Query 8:*

```
q8.df = cypher(graph1,
               "MATCH (oa:origin_airport)-[r:delayed_by]->(de:dest_airport)
                RETURN oa.name  AS Origin,
                      de.name  AS Destination,
                sum(tofloat(r.total_delay)) AS Total_delay_time
                ORDER BY (Total_delay_time) desc
                LIMIT 10")
head(q8.df)
```

```
##   Origin Destination Total_delay_time
## 1    LAX         SFO            35964
## 2    SFO         LAX            28734
## 3    LAS         SFO            22907
## 4    LAS         LAX            16710
## 5    SFO         LAS            16255
## 6    SAN         SFO            14838
```

From the results above, the data analyzed suggests that travelling from LAX (Los Angeles airport) to SFO (San Francisco airport) and back has the highest probability of getting delayed.

7. Conclusions