THE NEW YORK CITY TAXI AND LIMOUSINE TRIP

Vikrant Patil

About the dataset:

The New York City Taxi and Limousine trip record dataset consists of readings recorded from 1st September 2015 to 30th September 2015. The dataset has records for pickup and drop-off timings along with the date of the month. The latitudes and longitudes of pickup and drop-off locations are also included for respective trips. The different charges incurred during the trip are divided into categories along with the tip amount the driver received at the end of the tip.

Libraries used:

```
library(RCurl)
library(tidyverse)
library(rpart)
library(data.table)
library(ggplot2)
library(dplyr)
library(datetime)
library(lubridate)
library(class)
library(randomForest)
```

Programmatically download and load into your favorite analytical tool the trip data for September 2015.
Report how many rows and columns of data you have loaded.

Code:

Here, I have used the 'RCurl' library in R to fetch the dataset. The variable URL stores the fetched dataset while fread () provides faster and convenient reading of the dataset into the 'context1' variable.

```
URL <- "https://s3.amazonaws.com/nyc-tlc/trip+data/green_tripdata_2015-09.csv"
context1 <- fread(URL)
Read 1494926 rows and 21 (of 21) columns from 0.223 GB file in 00:00:15
```

The glimpse () function is an alternative to str() function, used to print the columns and the data contained in it. To use this function, we need to have 'dplyr' package installed.

```
glimpse(context3)
Observations: 1,494,926
Variables: 21

$ VendorID              <int> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, ...
$ lpep_pickup_datetime  <chr> "2015-09-01 00:02:34", "2015-09-01 00:04:20",
"2015-09-01 00:01:50", "2015-09-01 00:02:36"...
$ Lpep_dropoff_datetime <chr> "2015-09-01 00:02:38", "2015-09-01 00:04:24",
"2015-09-01 00:04:24", "2015-09-01 00:06:42"...
$ Store_and_fwd_flag    <chr> "N", "N", "N", "N", "N", "N", "N", "N", "N",
"N", "N", "N", "N", "N", "N", "N", "N", "N", ...
$ RateCodeID            <int> 5, 5, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
$ Pickup_longitude      <dbl> -73.97948, -74.01080, -73.92141, -73.92139, -
73.95548, -73.94530, -73.89088, -73.94670, -7...
$ Pickup_latitude       <dbl> 40.68496, 40.91222, 40.76671, 40.76668,
40.71405, 40.80819, 40.74643, 40.79732, 40.69383, ...
$ Dropoff_longitude     <dbl> -73.97943, -74.01078, -73.91441, -73.93143, -
73.94441, -73.93767, -73.87692, -73.93764, -7...
$ Dropoff_latitude      <dbl> 40.68502, 40.91221, 40.76469, 40.77158,
40.71473, 40.82120, 40.75631, 40.80452, 40.68053, ...
$ Passenger_count       <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1,
1, 1, 1, 1, 1, 5, 1, 6, 1, 1, 1, 1, 1, 1, 1, ...
$ Trip_distance         <dbl> 0.00, 0.00, 0.59, 0.74, 0.61, 1.07, 1.43,
0.90, 1.33, 0.84, 0.80, 0.70, 1.01, 0.39, 0.56, ...
$ Fare_amount           <dbl> 7.8, 45.0, 4.0, 5.0, 5.0, 5.5, 6.5, 5.0, 6.0,
5.5, 5.0, 4.0, 5.5, 3.5, 4.0, 7.5, 7.5, 5.0,...
$ Extra                 <dbl> 0.0, 0.0, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, ...
$ MTA_tax               <dbl> 0.0, 0.0, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, ...
$ Tip_amount            <dbl> 1.95, 0.00, 0.50, 0.00, 0.00, 1.36, 0.00,
0.00, 1.46, 0.00, 0.00, 1.06, 0.00, 0.00, 0.00, ...
$ Tolls_amount          <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
$ Ehail_fee             <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
$ improvement_surcharge <dbl> 0.0, 0.0, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3,
0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, ...
$ Total_amount          <dbl> 9.75, 45.00, 5.80, 6.30, 6.30, 8.16, 7.80,
6.30, 8.76, 6.80, 6.30, 6.36, 6.80, 4.80, 5.30,...
```

```
$ Payment_type          <int> 1, 1, 1, 2, 2, 1, 1, 2, 1, 2, 2, 1, 2, 2, 2,
2, 2, 1, 2, 2, 2, 1, 1, 2, 2, 2, 1, 2, 2, 2, ...
$ Trip_type             <int> 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
```

Finally, the dim() function is used to display the dimensions of the dataset. The output of dim() is in 'rows – columns' form. Here, the dataset has 1494926 rows and 21 columns.

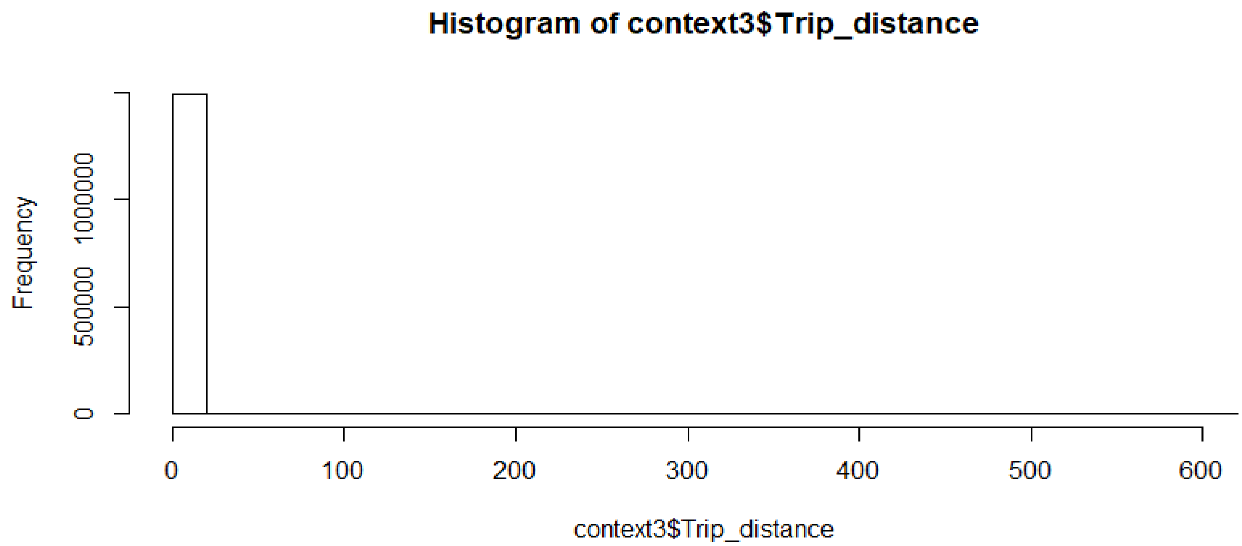
```
dim(context1)
[1] 1494926      21
```

Plot a histogram of the number of the trip distance ("Trip Distance").
Report any structure you find and any hypotheses you have about that structure.

Code:

Let's see how the histogram looks initially, without using any filters.

```
hist(context1$Trip_distance)
```



Now, that we know how the histogram looks, we will find the median and standard deviation of the Trip Distance readings.

```
median(context1$Trip_distance)
[1] 1.98

sd(context1$Trip_distance)
[1] 3.076621
```

Approximately, 99.9% of readings always lie within 3-standard deviations of the median observation. Here, 3 standard deviations are approximately equal to 12 and hence we filter the variable Trip_distance by excluding the readings which lie beyond 12 units.

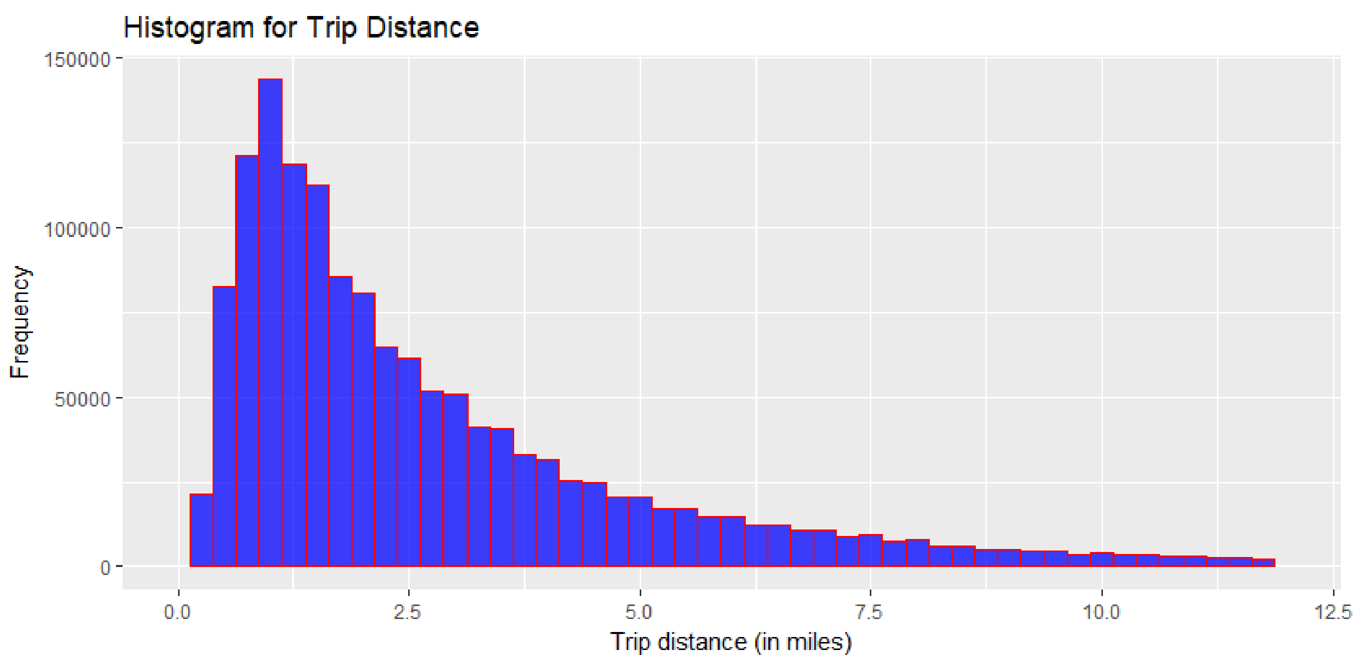
```
a <- filter(context1, context1$Trip_distance<=12)
```

We confirm our operation by using summary () to check the maximum value.

```
summary (a$Trip_distance)

Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.000   1.080   1.920   2.695   3.590  12.000
```

Finally, we plot the histogram using ggplot2.

```
qplot(a$Trip_distance, geom="histogram", xlim = c(0,12), binwidth = 0.25,
main = "Histogram for Trip Distance", xlab = "Trip distance (in miles)",
ylab = "Frequency", fill=I("blue"), col=I("red"), alpha=I(0.75))
```



From the histogram, we observe that the histogram is a right skewed histogram. This tells us that the data has a lower bound. As the distribution is not symmetrical we can say that the values are not random. Also, the median is lower than the mean of the data and both are lower than the standard deviation of the data.

We can hypothesize that many people travel a specific amount of distance every day which we may attribute to a location where these people have their workplaces.

Report mean and median trip distance grouped by hour of day.
We'd like to get a rough sense of identifying trips that originate or terminate at one of the NYC area airports. Can you provide a count of how many transactions fit this criteria, the average fare, and any other interesting characteristics of these trips.

Method:

- In the dataset, the pickup times and drop-off times both have the data type char.
- For smooth calculations, I have converted the variable 'lpep_pickup_datetime' into hours and then changed its data type into 'numeric' by using the as.numeric () function.
- The same method is used to convert the variable 'Lpep_dropoff_datetime' which corresponds to drop-off times of the rides, into hours and numeric data type.
- Then by using mutate () and group_by () functions, I grouped the mean and median Trip distance by hours and plotted the corresponding results in form of line graphs.
- For the second part of the sum, I created a data frame which consisted of Rate Code ID, Payment Type, Fare Amount, Total Amount and the Tip Amount.
- I created two subsets, subset1 having trips with Rate Code ID's as 2 and 3 while subset2 having the remaining values of Rate Code ID.

Code:

A variable called pickup_hour is created which has the contents of 'lpep_pickup_datetime'

```
#convert pickup time into hours
pickup_hour<- context1$lpep_pickup_datetime
```

Converting datetime in standard format

```
strptime(pickup_hour, format = '%Y-%m-%d %H:%M:%S', tz = '')

[1] "2015-09-01 00:02:34 CDT" "2015-09-01 00:04:20 CDT" "2015-09-01
00:01:50 CDT" "2015-09-01 00:02:36 CDT"
[5] "2015-09-01 00:00:14 CDT" "2015-09-01 00:00:39 CDT" "2015-09-01
00:00:52 CDT" "2015-09-01 00:02:15 CDT"
```

```
is.character(pickup_hour)
[1] TRUE
```

Converting pickup time into hours

```
pickup_hour1 <- hour(pickup_hour)
pickup_hour1
[1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[60] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Changing the data type of pickup hour

```
as.numeric(pickup_hour1)

[1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[60] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
summary(pickup_hour1)
Min. 1st Qu.  Median   Mean    3rd Qu.    Max.
0.00    9.00   15.00   13.53   19.00    23.00
```

```
is.numeric(pickup_hour1)
[1] TRUE
```

Moving the data values to original variable. Now the original variable has time in hours.

```
context1$lpep_pickup_datetime <- pickup_hour1

context1$lpep_pickup_datetime

[1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[60] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

summary(context1$Trip_distance)

Min.    1st Qu.  Median    Mean   3rd Qu.    Max.
0.000    1.100   1.980    2.968   3.740    603.100
```

Same as above method of converting time into hours

```
#convert dropoff time into hours

x <- context1$Lpep_dropoff_datetime
```

```
strptime(x, format = '%Y-%m-%d %H:%M:%S', tz = '')

[1] "2015-09-01 00:02:38 CDT" "2015-09-01 00:04:24 CDT" "2015-09-01 00:04:2
 4 CDT" "2015-09-01 00:06:42 CDT"
[5] "2015-09-01 00:04:20 CDT" "2015-09-01 00:05:20 CDT" "2015-09-01 00:05:5
 0 CDT" "2015-09-01 00:05:34 CDT"
```

```
e <- hour(x)

as.numeric(e)

[1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[60] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
is.numeric(e)

[1] TRUE

context1$Lpep_dropoff_datetime <- e
```

Grouping the variables with respect to hours

```
v <- mutate(context1, Hours= context1$lpep_pickup_datetime)

hoursofday <- group_by(v, Hours)

summary(hoursofday$Trip_distance)
Min.    1st Qu.  Median    Mean   3rd Qu.    Max.
0.000    1.100    1.980   2.968   3.740   603.100
```

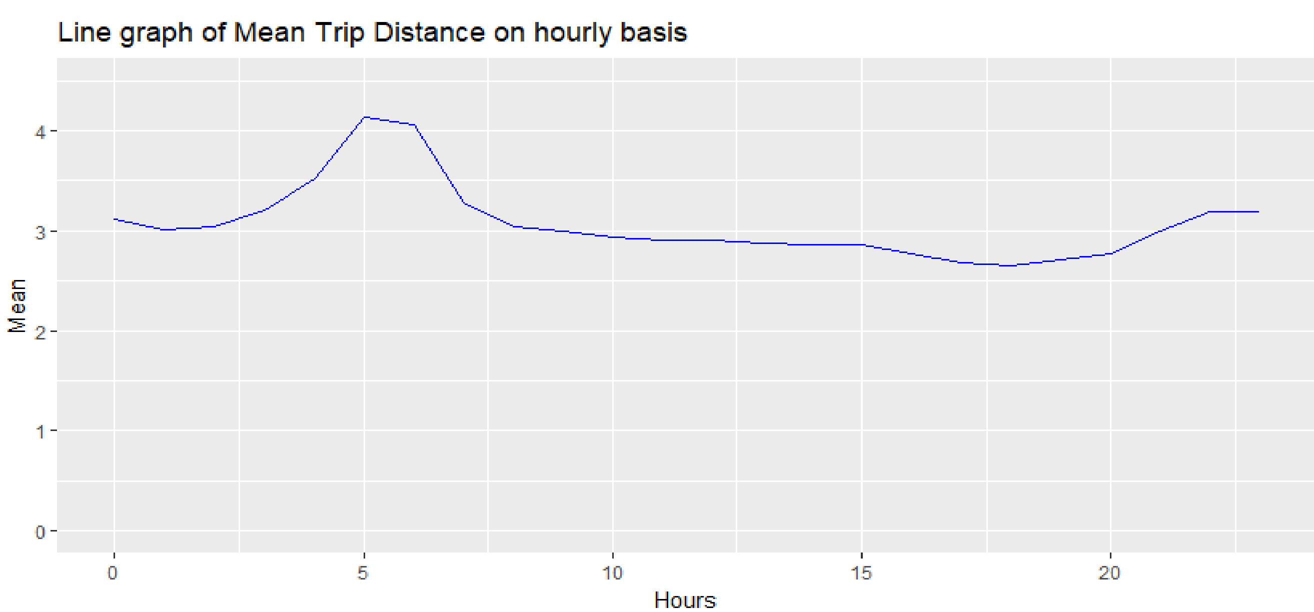
Summarizing the values with hour, mean and median

```
result <- summarize(hoursofday, Mean=mean(Trip_distance), Median=median(Trip_distance))

result
# A tibble: 24 x 3
   Hours     Mean Median
   <int>    <dbl>  <dbl>
 1     0 3.115276   2.20
 2     1 3.017347   2.12
 3     2 3.046176   2.14
 4     3 3.212945   2.20
 5     4 3.526555   2.36
 6     5 4.133474   2.90
 7     6 4.055149   2.84
 8     7 3.284394   2.17
 9     8 3.048450   1.98
10     9 2.999105   1.96
# ... with 14 more rows
```

```
data.table(result)
    Hours      Mean Median
 1:     0 3.115276   2.20
 2:     1 3.017347   2.12
 3:     2 3.046176   2.14
 4:     3 3.212945   2.20
 5:     4 3.526555   2.36
 6:     5 4.133474   2.90
 7:     6 4.055149   2.84
 8:     7 3.284394   2.17
 9:     8 3.048450   1.98
10:     9 2.999105   1.96
11:    10 2.944482   1.92
12:    11 2.912015   1.88
13:    12 2.903065   1.89
14:    13 2.878294   1.84
15:    14 2.864304   1.83
16:    15 2.857040   1.81
17:    16 2.779852   1.80
18:    17 2.679114   1.78
19:    18 2.653222   1.80
20:    19 2.715597   1.85
21:    20 2.777052   1.90
22:    21 2.999189   2.03
23:    22 3.185394   2.20
24:    23 3.191538   2.22
    Hours      Mean Median
```

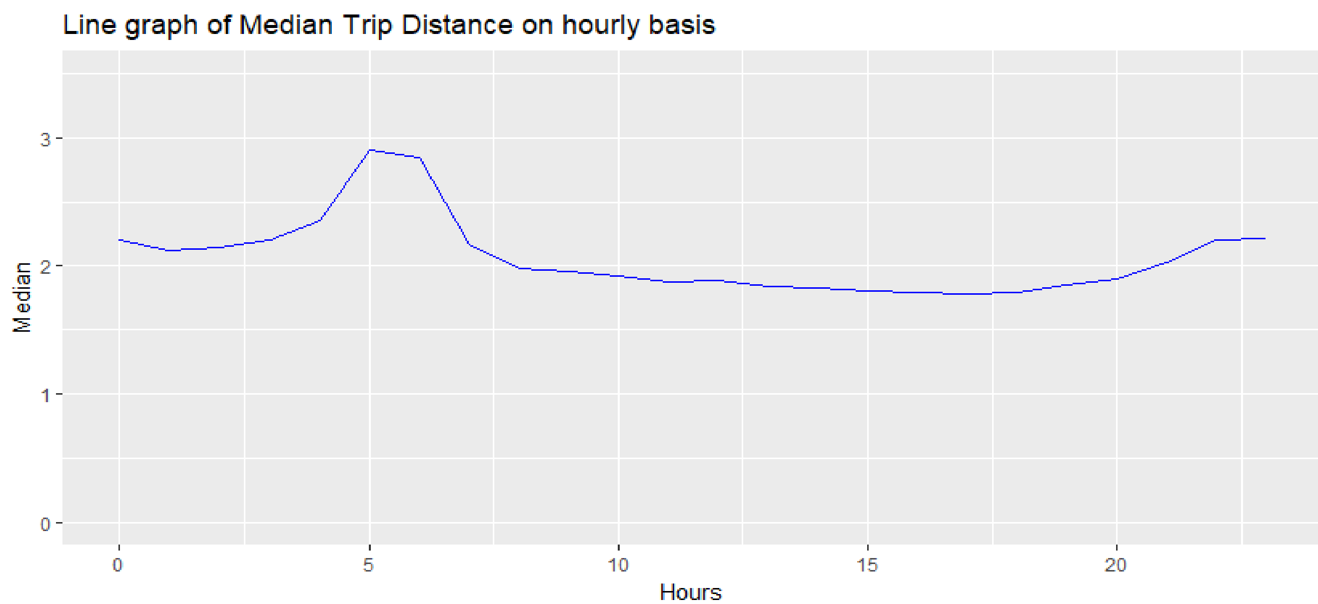
Plotting Mean trip distance per hour

```
ggplot(data=result, aes(x=Hours, y=Mean, group=1)) +
+    geom_line(color = "blue") +
+    ylim(0,4.5)
```



Plotting median trip distance per hour

```
ggplot(data=result, aes(x=Hours, y=Median, group=1)) +
+   geom_line(color = "blue") +
+   ylim(0,3.5)
```



Creating a data frame with RateCodeID, Payment type, Fare amount, Total amount and Tip amount

```
m <- data.frame(Code=hoursofday$RateCodeID, Payment_Type=hoursofday$Payment
 _type, Fare_amount=hoursofday$Fare_amount, Total_amount = hoursofday$Total
 _amount, Tip_amount = hoursofday$Tip_amount)
```

Creating a subset with RateCodeID = 2 or 3. These are the IDs for airports as mentioned in the data dicti
onary

```
subset1 <- subset(m, (Code==2 | Code == 3), drop = FALSE)
```

Number of trips originating or terminating at one of the airports

```
dim(subset1)

[1] 5552    5
```

Average fare of airport trips

```
average_fare1 <- mean(subset1$Fare_amount)
average_fare1
[1] 48.97695
```

Average total amount of airport trips (including other charges)

```
average_total_amount1 <- mean(subset1$Total_amount)
average_total_amount1
[1] 57.20842
```

Creating subset without trips to or from airports

```
subset2 <- subset(m, !(Code==2 | Code == 3), drop = FALSE)
dim(subset2)
[1] 1489374       5
```

Average fare of those trips

```
average_fare2 <- mean(subset2$Fare_amount)
average_fare2
[1] 12.40738
```

Average total amount of those trips

```
average_total_amount2 <- mean(subset2$Total_amount)
average_total_amount2
[1] 14.87492
```

Average tip amount

```
average_tip_amount2 <- mean(subset2$Tip_amount)

average_tip_amount2

[1] 1.224104
```

We can see that only 5552 taxi rides either originate or end at the airports. Moreover, the average fare, average total amount and average tip amount of these rides are nearly 4 times that of the rides which do not originate or terminate at any of the airports.

Build a derived variable for tip as a percentage of the total fare.
Build a predictive model for tip as a percentage of the total fare. Use as much of the data as you like (or all of it). We will validate a sample.

Method:

- To start with, I added columns to the dataset by converting pickup times and drop-off times into different formats and then converted them into numeric data type.
- Using pick up times and drop off times in seconds, the total trip time was calculated and from that the average speed was calculated and added to the dataset.
- The tip percentage variable was derived by dividing tip amount by total amount and multiplying the result by 100.
- The data was cleaned by removing the unwanted columns. The NA values in Trip_type variable were replaced by most occurring value in the column, i.e. 1 while RateCodeID variable had values of 99 which according to data definition stand for nothing. These values were replaced by 2 as it was the most occurring value.
- The columns storing different amounts like Total_amount, Fare_amount, Improvement_surcharge and Tip_amount had negative values. As amount cannot be negative, there was an option to omit them fully or convert them into their absolute values. These columns were converted into their absolute values.
- I added another column which stores 0 if there is no tip given or 1 if the tip amount is greater than 0.
- This column was created to help in the classification process.
- The classification was carried out by using Random Forest method. A sample of first 10000 values was used for the same.

Preparing data by converting pickup time and drop-off time into day of week(%W), timings in hour(%H), date(%d) and week(%V) and adding it to dataset.

```
context4$dayofweek <- strftime(context4$lpep_pickup_datetime, format = '%W'
, tz = '')

context4$pickup_in_hours <- strftime(context4$lpep_pickup_datetime, format
= '%H', tz = '')

context4$dropoff_in_hours <- strftime(context4$lpep_pickup_datetime, format
= '%H', tz = '')

context4$date_only <- strftime(context4$lpep_pickup_datetime, format = '%d'
, tz = '')

context4$week <- strftime(context4$lpep_pickup_datetime, format = '%V', tz
= '')
```

Converting pickup time into seconds and numeric data type. Strptime is used to get the data in standard datetime format.

```
strptime(context4$lpep_pickup_datetime, format = '%Y-%m-%d %H:%M:%S', tz =
'')

[1] "2015-09-01 00:02:34 CDT" "2015-09-01 00:04:20 CDT" "2015-09-01 00:01:5
 0 CDT" "2015-09-01 00:02:36 CDT"
[5] "2015-09-01 00:00:14 CDT" "2015-09-01 00:00:39 CDT" "2015-09-01 00:00:5
 2 CDT" "2015-09-01 00:02:15 CDT"
```

Removing the '-' from date and creating a substring of time in 'Hours:Minutes:Seconds'

```
gsub("-", "", context4$Lpep_dropoff_datetime)
[1] "20150901 00:02:38" "20150901 00:04:24" "20150901 00:04:24" "20150901 0
 0:06:42" "20150901 00:04:20"
[6] "20150901 00:05:20" "20150901 00:05:50" "20150901 00:05:34" "20150901 0
 0:07:20" "20150901 00:07:23"

lpep_pickup_datetime_substr <- substr(context4$lpep_pickup_datetime, 12, 19
)
```

Converting the substring into time format using as.time()

```
as.time(lpep_pickup_datetime_substr)

[1] 00:02 00:04 00:01 00:02 00:00 00:00 00:00 00:02 00:02 00:02 00:01 00:04
 00:03 00:05 00:04 00:00 00:01 00:04 00:00
[20] 00:01 00:00 00:04 00:04 00:01 00:01 00:03 00:00 00:08 00:06 00:02 00:0
 0 00:07 00:05 00:02 00:00 00:03 00:01 00:02

is.character(lpep_pickup_datetime_substr)

[1] TRUE
```

Converting the time format into hours only format; checking the data type
of the original variable and then converting it to numeric type having data in hours only format.

```
lpep_pickup_datetime1 <- hms(lpep_pickup_datetime_substr)

lpep_pickup_datetime <- as.numeric(lpep_pickup_datetime1)

is.numeric(context4$lpep_pickup_datetime)

[1] FALSE

context4$pickup_in_seconds <- lpep_pickup_datetime
```

Convert dropoff time into seconds and numeric data type(Same method used above)

```
strptime(context4$Lpep_dropoff_datetime, format = '%Y-%m-%d %H:%M:%S', tz =
'')

[1] "2015-09-01 00:02:38 CDT" "2015-09-01 00:04:24 CDT" "2015-09-01 00:04:2
 4 CDT" "2015-09-01 00:06:42 CDT"
[5] "2015-09-01 00:04:20 CDT" "2015-09-01 00:05:20 CDT" "2015-09-01 00:05:5
 0 CDT" "2015-09-01 00:05:34 CDT"
```

```
gsub("-", "", context4$Lpep_dropoff_datetime)
[1] "20150901 00:02:38" "20150901 00:04:24" "20150901 00:04:24" "20150901 0
 0:06:42" "20150901 00:04:20"
[6] "20150901 00:05:20" "20150901 00:05:50" "20150901 00:05:34" "20150901 0
 0:07:20" "20150901 00:07:23"

Lpep_dropoff_datetime_substr <- substr(context4$Lpep_dropoff_datetime, 12,
19)
```

```
as.time(Lpep_dropoff_datetime_substr)

[1] 00:02 00:04 00:04 00:06 00:04 00:05 00:05 00:05 00:07 00:07 00:05 00:06
 00:07 00:07 00:07 00:07 00:08 00:07 00:07
[20] 00:07 00:07 00:08 00:10 00:08 00:10 00:09 00:08 00:08 00:09 00:10 00:1
 1 00:07 00:08 00:07 00:08 00:08 00:08 00:05

is.character(Lpep_dropoff_datetime_substr)

[1] TRUE
```

```
Lpep_dropoff_datetime1 <- hms(Lpep_dropoff_datetime_substr)

Lpep_dropoff_datetime <- as.numeric(Lpep_dropoff_datetime1)

is.numeric(context4$lpep_dropoff_datetime)

[1] FALSE

context4$dropoff_in_seconds <- Lpep_dropoff_datetime
```

Converting the newly added columns in numeric data type

```
#data type conversion
context4$dropoff_in_seconds <- Lpep_dropoff_datetime
context4$dayofweek <- as.numeric(context4$dayofweek)
context4$pickup_in_hours <- as.numeric(context4$pickup_in_hours)
context4$dropoff_in_hours <- as.numeric(context4$dropoff_in_hours)
context4$date_only <- as.numeric(context4$date_only)
context4$week <- as.numeric(context4$week)
```

Taking difference between drop-off time (secs) and pickup time (secs) to calculate total trip time.

```
#total trip time calculation
context4$totaltriptime <- context4$dropoff_in_seconds - context4$pickup_in_
seconds
```

Calculating average speed in miles/hour

```
#average speed calculation
context4$average_speed <- 3600*(context4$Trip_distance/context4$totaltripti
me)
```

Creating a variable for tip percentage

```
context4$tip_percentage <- 100*(context4$Tip_amount/context4$Total_amount)
```

Removing columns which would not affect tip percentage

```
#data cleaning

context4$VendorID <- NULL
context4$lpep_pickup_datetime <- NULL
context4$Lpep_dropoff_datetime <- NULL
context4$Store_and_fwd_flag <- NULL
context4$Pickup_longitude <- NULL
context4$Pickup_latitude <- NULL
context4$Dropoff_longitude <- NULL
context4$Dropoff_latitude <- NULL
context4$Ehail_fee <- NULL
context4$Total_amount <- abs(context4$Total_amount)
context4$Fare_amount <- abs(context4$Fare_amount)
context4$improvement_surcharge <- abs(context4$improvement_surcharge)
context4$Tip_amount <- abs(context4$Tip_amount)
context4$RateCodeID <- replace(context4$RateCodeID, 99, 2)
context4$Trip_type <- replace(context4$Trip_type, NA, 1)
```

Creating factors. If tip percentage = 0 then add 0 else 1.

```
#add factors

dataset$class1 <- (dataset$tip_percentage>0)*1

fdata <- factor(dataset$class1)

dataset$class1 <- fdata

dataset$class1
[1] 1 0 0 1 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 1 1 0 0 0 1 0 0 0 1 0 1 0 0 0 0 1
 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1 0 1 0 0 1
[60] 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 0 0 1 0 0 1 0 1 1 0 1 0 0
 0 0 1 0 1 1 0 1 0 1 0 0 0 0 0 1 0 1 0 0 1 1 0 0

Levels: 0 1
```

More of an intuitive process. Looking at the data labels, only the variables which would have the most e ffect on the tip percentage are kept in the dataset.

```
#more data cleaning

dataset$RateCodeID<- NULL
dataset$Passenger_count<- NULL
dataset$Extra<- NULL
dataset$MTA_tax<- NULL
dataset$Tolls_amount<- NULL
dataset$improvement_surcharge<- NULL
dataset$Trip_type<- NULL
dataset$dropoff_in_seconds<- NULL
dataset$pickup_in_seconds<- NULL
dataset$week<- NULL
dataset$date_only<- NULL
dataset$dropoff_in_hours<- NULL
dataset$pickup_in_hours<- NULL
dataset$dayofweek<- NULL
```

Classification using Random Forest Algorithm.

```
#Classification using Random Forest

dataset1 <- dataset[1:10000,]

tip_class <- randomForest(class1~.,data=dataset1, proximity = TRUE)

print(tip_class)

Call:
 randomForest(formula = class1 ~ ., data = dataset1, proximity = TRUE)
              Type of random forest: classification
                    Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 0.01%
Confusion matrix:

      0     1   class.error
0  5898     0  0.0000000000

1     1  4101  0.0002437835
```

From the above Random Forest Classification, we see that there was a very little error in classifying the data into right classes. This shows that the variables chosen heavily affect the probability that the driver will receive a tip or not.

Build a derived variable representing the average speed over the course of a trip.
Can you perform a test to determine if the average trip speeds are materially the same in all weeks of September? If you decide they are not the same, can you form a hypothesis regarding why they differ?
Can you build up a hypothesis of average trip speed as a function of time of day?

Method:

- Initially I created two copies of the dataset namely context5 and context6.
- Using the method discussed in the previous questions, I converted the pickup and drop-off times into seconds and numeric data type.
- The reason to convert it into seconds is for smoother calculations of the total trip time.
- The obtained trip time is then filtered for any unwanted values and the average speed is calculated in miles/hour.
- The speed readings are cleaned for any outliers and histogram is plotted for average speed over the course of the trip.
- For the second part of the question, I grouped the readings by weeks.
- The t-test test is used to find any equality or significance between the values and hence I used it to form my hypothesis.
- In the third part of the question, I created a subset with drop-off hours and average speeds. The hours were then converted into levels from 0 to 23 and bar graphs for mean and median were plotted.

Code:

add column for dropoff time in hours

```
context6 <- context1
inhours <- context6$Lpep_dropoff_datetime

strptime(inhours, format = '%Y-%m-%d %H:%M:%S', tz = '')

[1] "2015-09-01 00:02:38 CDT" "2015-09-01 00:04:24 CDT" "2015-09-01 00:04:2
 4 CDT" "2015-09-01 00:06:42 CDT"
[5] "2015-09-01 00:04:20 CDT" "2015-09-01 00:05:20 CDT" "2015-09-01 00:05:5
 0 CDT" "2015-09-01 00:05:34 CDT"

dropofftime_inhours <- hour(inhours)

context5$Lpep_dropoff_Hours <- dropofftime_inhours

context5$Lpep_dropoff_Hours
[1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[60] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Converting the drop-off time into units of weeks (decimal number as defined in ISO 8601) by using %V argument. Here the week starts from Monday.

```
#add column for number of weeks

week1 <- strftime(context1$Lpep_dropoff_datetime, format = "%V")
```

```
week1
[1] "36" "36" "36" "36" "36" "36" "36" "36" "36" "36" "36" "36" "36" "36" "
 36" "36" "36" "36" "36" "36" "36" "36" "36"
[24] "36" "36" "36" "36" "36" "36" "36" "36" "36" "36" "36" "36" "36" "36"
 "36" "36" "36" "36" "36" "36" "36" "36" "36"

context5$weeks <-week1
```

Using the same method as in Question 4 to convert pickup time into seconds

```
#convert pickup time into seconds and numeric data type

strptime(context5$lpep_pickup_datetime, format = '%Y-%m-%d %H:%M:%S', tz =
 '')

[1] "2015-09-01 00:02:34 CDT" "2015-09-01 00:04:20 CDT" "2015-09-01 00:01:5
 0 CDT" "2015-09-01 00:02:36 CDT"
[5] "2015-09-01 00:00:14 CDT" "2015-09-01 00:00:39 CDT" "2015-09-01 00:00:5
 2 CDT" "2015-09-01 00:02:15 CDT"

gsub("-", "", context5$lpep_pickup_datetime)

[1] "20150901 00:02:34" "20150901 00:04:20" "20150901 00:01:50" "20150901 0
 0:02:36" "20150901 00:00:14"
[6] "20150901 00:00:39" "20150901 00:00:52" "20150901 00:02:15" "20150901 0
 0:02:36" "20150901 00:02:13"

lpep_pickup_datetime_substr <- substr(context5$lpep_pickup_datetime, 12, 19
 )

as.time(lpep_pickup_datetime_substr)

[1] 00:02 00:04 00:01 00:02 00:00 00:00 00:00 00:02 00:02 00:02 00:01 00:04
 00:03 00:05 00:04 00:00 00:01 00:04 00:00
[20] 00:01 00:00 00:04 00:04 00:01 00:01 00:03 00:00 00:08 00:06 00:02 00:0
 0 00:07 00:05 00:02 00:00 00:03 00:01 00:02


lpep_pickup_datetime1 <- hms(lpep_pickup_datetime_substr)

lpep_pickup_datetime <- as.numeric(lpep_pickup_datetime1)

is.numeric(context5$lpep_pickup_datetime)

[1] FALSE

context5$lpep_pickup_datetime <- lpep_pickup_datetime
```

Using the same method as used in Question 4 to convert drop-off time into seconds.

```
#convert dropoff time into seconds and numeric data type

strptime(context5$Lpep_dropoff_datetime, format = '%Y-%m-%d %H:%M:%S', tz =
 '')

[1] "2015-09-01 00:02:38 CDT" "2015-09-01 00:04:24 CDT" "2015-09-01 00:04:2
 4 CDT" "2015-09-01 00:06:42 CDT"
[5] "2015-09-01 00:04:20 CDT" "2015-09-01 00:05:20 CDT" "2015-09-01 00:05:5
 0 CDT" "2015-09-01 00:05:34 CDT"

gsub("-", "", context5$Lpep_dropoff_datetime)

[1] "20150901 00:02:38" "20150901 00:04:24" "20150901 00:04:24" "20150901 0
 0:06:42" "20150901 00:04:20"
[6] "20150901 00:05:20" "20150901 00:05:50" "20150901 00:05:34" "20150901 0
 0:07:20" "20150901 00:07:23"


Lpep_dropoff_datetime_substr <- substr(context5$Lpep_dropoff_datetime, 12,
 19)

as.time(Lpep_dropoff_datetime_substr)

[1] 00:02 00:04 00:04 00:06 00:04 00:05 00:05 00:05 00:07 00:07 00:05 00:06
 00:07 00:07 00:07 00:07 00:08 00:07 00:07
[20] 00:07 00:07 00:08 00:10 00:08 00:10 00:09 00:08 00:08 00:09 00:10 00:1
 1 00:07 00:08 00:07 00:08 00:08 00:08 00:05

Lpep_dropoff_datetime1 <- hms(Lpep_dropoff_datetime_substr)

Lpep_dropoff_datetime <- as.numeric(Lpep_dropoff_datetime1)

is.numeric(context5$lpep_dropoff_datetime)

[1] FALSE

context5$Lpep_dropoff_datetime <- Lpep_dropoff_datetime
```

Calculating Total trip time by subtracting drop-off time (secs) and pickup time (secs)

```
#total trip time

triptime <- Lpep_dropoff_datetime - lpep_pickup_datetime

triptime

[1]    4    4  154  246  246  281  298  199  284  310  231  126  274  130
 148  416  434  198  432  355  416  235  352
[24]  391  550  358  476   21  180  474  633   15  147  323  515  334  395
 146  228  190  116  138  393  256    8  129

context5$triptime <- triptime
```

Filtering out total trip time readings which are less than 90 secs. A cab ride lasting just 90 secs is rarity or is a result of wrong data entries.

```
#cleaning undesirable time readings

triptime_cleaned <- filter(context5, triptime > 90)
```

Creating subset of total trip time (secs) and total trip distance (miles)

```
subset <- select(triptime_cleaned, triptime, Trip_distance)
```

Calculating the average speed in miles/hour

```
average_speed <- 3600*(subset$Trip_distance/subset$triptime)

average_speed
[1] 13.792208 10.829268  8.926829 13.708185 17.275168 16.281407 16.859155
 9.754839 12.467532 20.000000 13.270073
[12] 10.800000 13.621622 15.576923 12.110599 16.000000 14.416667  8.315493
 15.663462 20.527660 18.715909 13.166240

summary(average_speed)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max. 
   0.000    9.429   11.809   12.993   15.051 3479.423 
```

Adding average speed column to both the datasets for ease in calculations

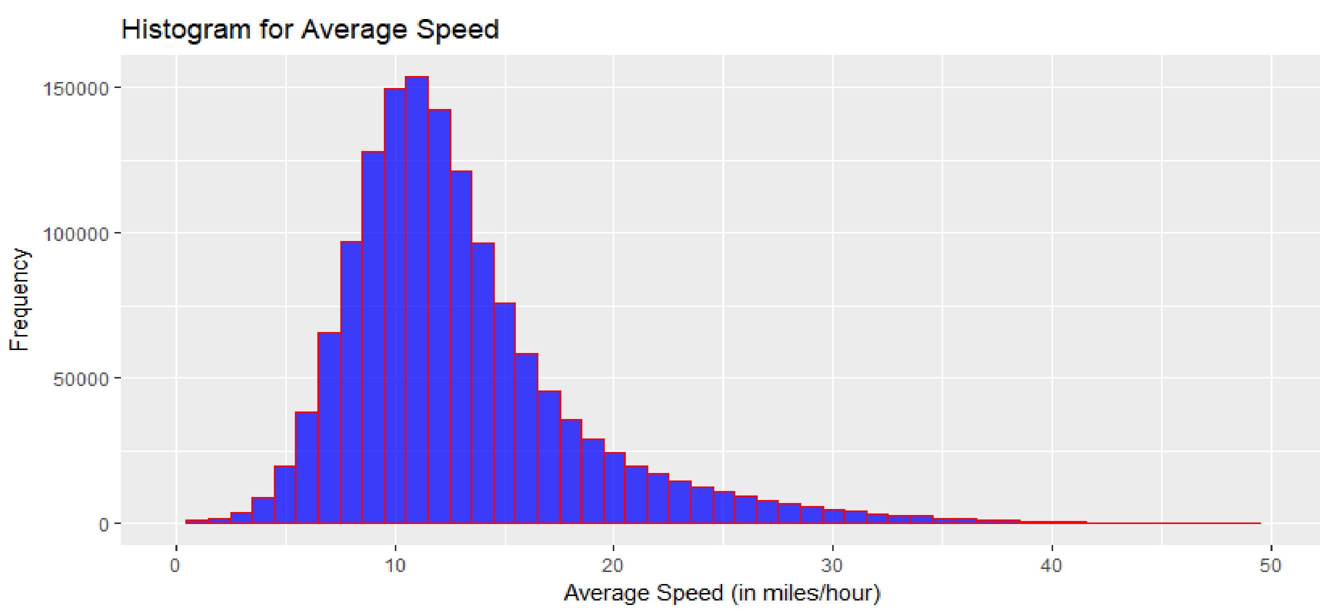
```
triptime_cleaned$avg_speed <- average_speed

subset$average_speed <- average_speed
```

Filtering out speed readings greater than 50. In NYC (all types of roads included) the average speed limit is 50mph, hence the filter.

```
#cleaning speed readings

subset <- filter(subset, average_speed < 50)

subset$average_speed

[1] 13.792208 10.829268  8.926829 13.708185 17.275168 16.281407 16.859155
 9.754839 12.467532 20.000000 13.270073
[12] 10.800000 13.621622 15.576923 12.110599 16.000000 14.416667  8.315493
 15.663462 20.527660 18.715909 13.166240


summary(subset$average_speed)
Min.        1st Qu.       Median    Mean       3rd Qu.         Max.
0.000        9.429        11.807    12.971     15.048         49.986

subset2 <- filter(triptime_cleaned, avg_speed < 50)
```

#plotting histogram for Ques 5)a)

```
qplot(subset$average_speed, geom="histogram", xlim = c(0,50), binwidth = 1,
 main = "Histogram for Average Speed", xlab = "Average Speed (in miles/hour
)", ylab = "Frequency", fill=I("blue"), col=I("red"), alpha=I(0.75))
```



```
#Ques 5)b)

subset4 <- filter(triptime_cleaned, avg_speed < 50)

subset5 <- select(subset4, weeks, avg_speed)

result2 <- group_by(subset5, weeks) %>%
+   summarise(
+     count = n(),
+     mean = mean(avg_speed, na.rm = TRUE),
+     median = median(avg_speed, na.rm = TRUE),
+     sd = sd(avg_speed, na.rm = TRUE)
+   )


result2

# A tibble: 5 x 5
  weeks  count     mean   median       sd
  <chr>  <int>    <dbl>    <dbl>    <dbl>
1    36 287913 13.36865 12.11374 5.758512
2    37 349798 12.74523 11.61876 5.645801
3    38 345485 12.76858 11.65468 5.488254
4    39 325481 13.24778 12.04461 5.698794
5    40 127308 12.53714 11.43646 5.507692
```

Conducting t-test for mean and weeks.

```
t.test(result2$mean, result2$weeks)

 Welch Two Sample t-test

data:  result2$mean and result2$weeks
t = -34.582, df = 4.4053, p-value = 1.556e-06
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -27.00805 -23.12500
sample estimates:

mean of x   mean of y
12.93348    38.00000
```

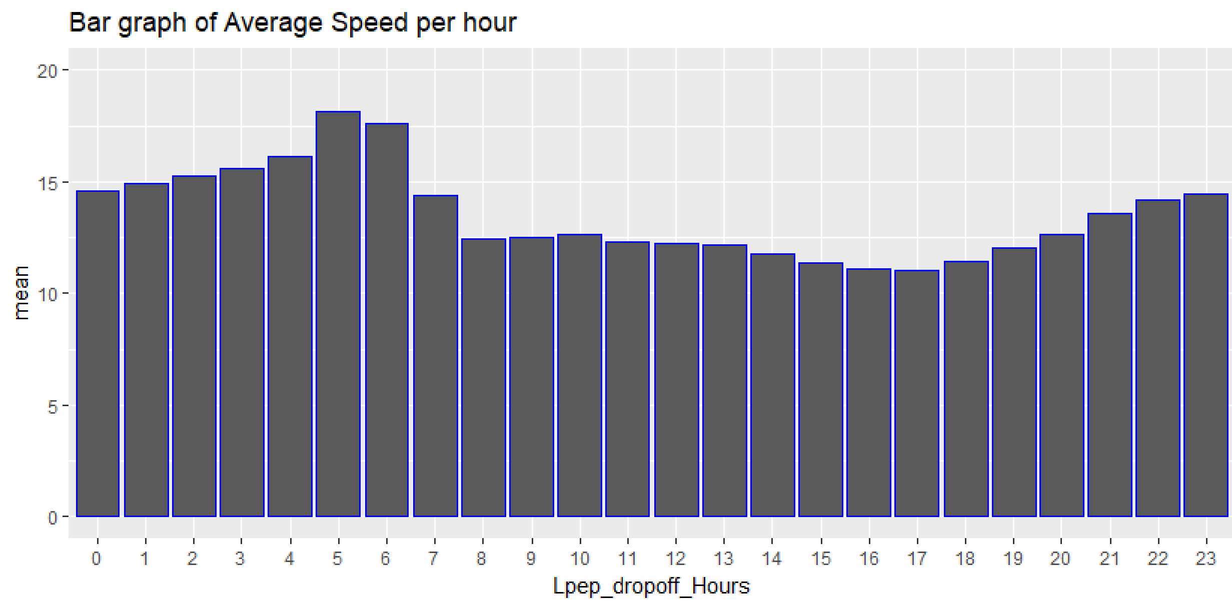
Factoring the dataset into levels from 0 to 23 (for hours)

```
#Ques 5)c)

subset3 <- select(subset2, Lpep_dropoff_Hours, avg_speed)

fdata <- factor(subset3$Lpep_dropoff_Hours)

subset3$Lpep_dropoff_Hours <- fdata

levels(subset3$Lpep_dropoff_Hours)
[1] "0"  "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "
14" "15" "16" "17" "18" "19" "20" "21" "22" "23"
```

```
result <- group_by(subset3, Lpep_dropoff_Hours) %>%
+   summarise(
+     count = n(),
+     mean = mean(avg_speed, na.rm = TRUE),
+     median = median(avg_speed, na.rm = TRUE),
+     sd = sd(avg_speed, na.rm = TRUE)
+   )

result

# A tibble: 24 x 5
   Lpep_dropoff_Hours count     mean   median       sd
               <fctr> <int>    <dbl>    <dbl>    <dbl>
 1                  0 53023 14.56772 13.34118 5.588746
 2                  1 55088 14.94019 13.68821 5.824380
 3                  2 42160 15.25304 13.98058 5.970701
 4                  3 31452 15.60309 14.27922 6.196688
 5                  4 27059 16.16334 14.53726 6.943104
 6                  5 17012 18.15698 16.09784 7.925645
 7                  6 19100 17.63916 15.74783 7.354322
 8                  7 34757 14.40144 12.78764 6.531830
 9                  8 53583 12.46814 11.14551 5.906674
10                  9 60806 12.53565 11.26257 5.808504
# ... with 14 more rows
```

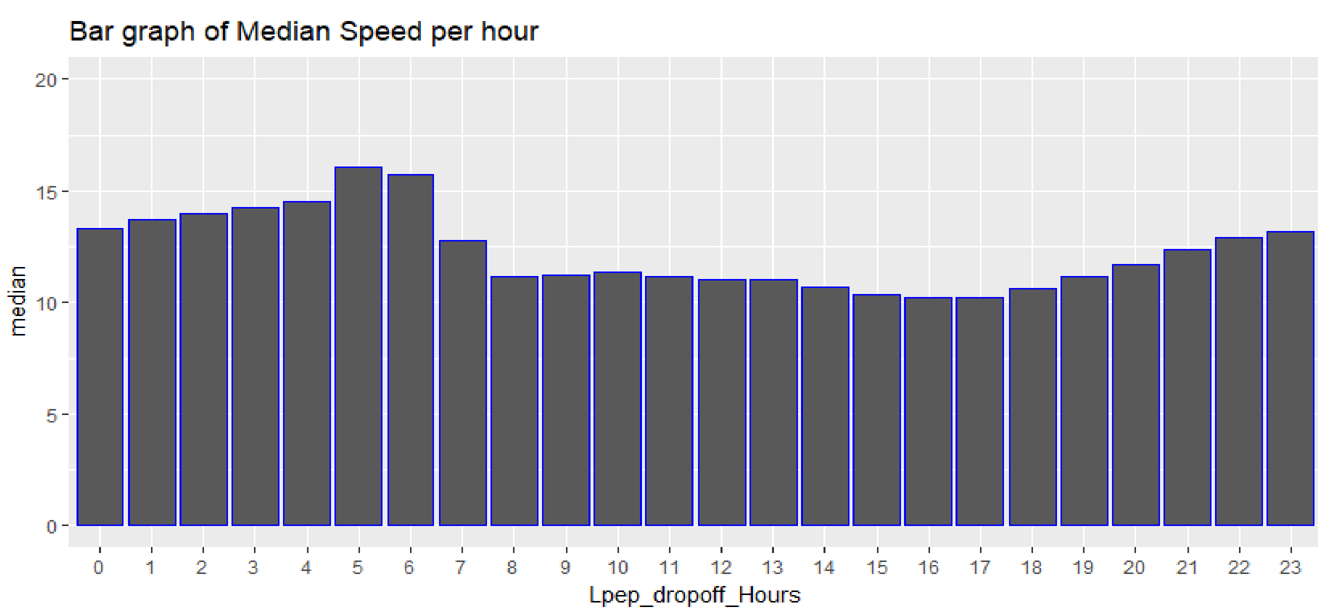
Plot of mean speed per hour

```
ggplot(data=result, aes(x=Lpep_dropoff_Hours, y=mean)) +
+  geom_bar(stat="identity") +
+   ylim(0,20)
```



Plot of median speed per hour

```
ggplot(data=result, aes(x=Lpep_dropoff_Hours, y=median)) +
+   geom_bar(stat="identity") +
+   ylim(0,20)
```



Findings:

It is found out in the first part that the average speed of 12 miles/hour occurs most frequently in the dataset. We can say that the in September 2015, the green taxis most frequently ran with a speed of 12 miles/hour.
From the histogram, we can also observe that most of the average speed readings lie between 9 miles/hour to 14 miles/hour.

From the t-test we can conclude that there are significant differences between the average speed of the taxis in all the weeks of September 2015. A very low p-value indicates that we reject our null hypothesis that the means have no significant differences.

In the third part of the question, we can observe that the average speeds and median speeds are very high in the morning hours. We can attribute this to people hailing taxis to get to their work destinations. Both the average and median speeds fall in the afternoon hours but rise gradually starting from 7 pm which we can attribute to people going back from their work destinations.