DA5020.P3.Hsiao-Yu.Peng

Chandani Shrestha, Joshua Okon, and Hsiao-Yu Peng

2023-12-11

Question 1

CRISP-DM: Data Understanding

• **Load** the NYC Green Taxi Trip Records data into a data frame or tibble. • **Data exploration**: explore the data to identify any patterns and analyze the relationships between the features and the target variable i.e. tip amount. At a minimum, you should analyze: 1) the distribution, 2) the correlations 3) missing values and 4) outliers — provide supporting visualizations and explain all your steps.

Data Exploration

```
tripdata_df<-read_csv("2018_Green_Taxi_Trip_Data.csv")
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)

## Rows: 1048575 Columns: 19
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (3): lpep_pickup_datetime, lpep_dropoff_datetime, store_and_fwd_flag
## dbl (15): VendorID, RatecodeID, PULocationID, DOLocationID, passenger_count,...
## lgl  (1): ehail_fee
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
dim(tripdata_df)
```

```
## [1] 1048575      19
```

```
head(tripdata_df)
```

```
## # A tibble: 6 x 19
##   VendorID lpep_pickup_datetime lpep_dropoff_datetime store_and_fwd_flag
##      <dbl> <chr>                <chr>                 <chr>
```

```
## 1        2 1/1/2018 0:18        1/1/2018 0:24         N
## 2        2 1/1/2018 0:30        1/1/2018 0:46         N
## 3        2 1/1/2018 0:07        1/1/2018 0:19         N
## 4        2 1/1/2018 0:32        1/1/2018 0:33         N
## 5        2 1/1/2018 0:32        1/1/2018 0:33         N
## 6        2 1/1/2018 0:38        1/1/2018 1:08         N
## # i 15 more variables: RatecodeID <dbl>, PULocationID <dbl>,
## #   DOLocationID <dbl>, passenger_count <dbl>, trip_distance <dbl>,
## #   fare_amount <dbl>, extra <dbl>, mta_tax <dbl>, tip_amount <dbl>,
## #   tolls_amount <dbl>, ehail_fee <lgl>, improvement_surcharge <dbl>,
## #   total_amount <dbl>, payment_type <dbl>, trip_type <dbl>
```

```
glimpse(tripdata_df)
```

```
## Rows: 1,048,575
## Columns: 19
## $ VendorID              <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, ~
## $ lpep_pickup_datetime  <chr> "1/1/2018 0:18", "1/1/2018 0:30", "1/1/2018 0:07~
## $ lpep_dropoff_datetime <chr> "1/1/2018 0:24", "1/1/2018 0:46", "1/1/2018 0:19~
## $ store_and_fwd_flag    <chr> "N", "N", "N", "N", "N", "N", "N", "N", "N", "N"~
## $ RatecodeID            <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ PULocationID          <dbl> 236, 43, 74, 255, 255, 255, 189, 189, 129, 226, ~
## $ DOLocationID          <dbl> 236, 42, 152, 255, 255, 161, 65, 225, 82, 7, 129~
## $ passenger_count       <dbl> 5, 5, 1, 1, 1, 1, 5, 5, 1, 1, 2, 2, 1, 1, 2, 1, ~
## $ trip_distance         <dbl> 0.70, 3.50, 2.14, 0.03, 0.03, 5.63, 1.71, 3.45, ~
## $ fare_amount           <dbl> 6.0, 14.5, 10.0, -3.0, 3.0, 21.0, 8.5, 14.5, 10.~
## $ extra                 <dbl> 0.5, 0.5, 0.5, -0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.~
## $ mta_tax               <dbl> 0.5, 0.5, 0.5, -0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.~
## $ tip_amount            <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 3.16, ~
## $ tolls_amount          <dbl> 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, ~
## $ improvement_surcharge <dbl> 0.3, 0.3, 0.3, -0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.~
## $ total_amount          <dbl> 7.30, 15.80, 11.30, -4.30, 4.30, 22.30, 9.80, 18~
## $ payment_type          <dbl> 2, 2, 2, 3, 2, 2, 2, 1, 2, 2, 2, 2, 2, 1, 1, 1, ~
## $ trip_type             <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
```

```
str(tripdata_df)
```

```
## spc_tbl_ [1,048,575 x 19] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ VendorID             : num [1:1048575] 2 2 2 2 2 2 2 2 2 2 ...
##  $ lpep_pickup_datetime : chr [1:1048575] "1/1/2018 0:18" "1/1/2018 0:30" "1/1/2018 0:07" "1/1/2018 0
##  $ lpep_dropoff_datetime: chr [1:1048575] "1/1/2018 0:24" "1/1/2018 0:46" "1/1/2018 0:19" "1/1/2018 0
##  $ store_and_fwd_flag   : chr [1:1048575] "N" "N" "N" "N" ...
##  $ RatecodeID           : num [1:1048575] 1 1 1 1 1 1 1 1 1 1 ...
##  $ PULocationID         : num [1:1048575] 236 43 74 255 255 255 189 189 129 226 ...
##  $ DOLocationID         : num [1:1048575] 236 42 152 255 255 161 65 225 82 7 ...
##  $ passenger_count      : num [1:1048575] 5 5 1 1 1 1 5 5 1 1 ...
##  $ trip_distance        : num [1:1048575] 0.7 3.5 2.14 0.03 0.03 5.63 1.71 3.45 1.61 1.87 ...
##  $ fare_amount          : num [1:1048575] 6 14.5 10 -3 3 21 8.5 14.5 10 7.5 ...
##  $ extra                : num [1:1048575] 0.5 0.5 0.5 -0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
##  $ mta_tax              : num [1:1048575] 0.5 0.5 0.5 -0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
##  $ tip_amount           : num [1:1048575] 0 0 0 0 0 0 0 3.16 0 0 ...
##  $ tolls_amount         : num [1:1048575] 0 0 0 0 0 0 0 0 0 0 ...
```

```
##  $ ehail_fee            : logi [1:1048575] NA NA NA NA NA NA ...
##  $ improvement_surcharge: num [1:1048575] 0.3 0.3 0.3 -0.3 0.3 0.3 0.3 0.3 0.3 0.3 ...
##  $ total_amount         : num [1:1048575] 7.3 15.8 11.3 -4.3 4.3 ...
##  $ payment_type         : num [1:1048575] 2 2 2 3 2 2 2 1 2 2 ...
##  $ trip_type            : num [1:1048575] 1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   VendorID = col_double(),
##   ..   lpep_pickup_datetime = col_character(),
##   ..   lpep_dropoff_datetime = col_character(),
##   ..   store_and_fwd_flag = col_character(),
##   ..   RatecodeID = col_double(),
##   ..   PULocationID = col_double(),
##   ..   DOLocationID = col_double(),
##   ..   passenger_count = col_double(),
##   ..   trip_distance = col_double(),
##   ..   fare_amount = col_double(),
##   ..   extra = col_double(),
##   ..   mta_tax = col_double(),
##   ..   tip_amount = col_double(),
##   ..   tolls_amount = col_double(),
##   ..   ehail_fee = col_logical(),
##   ..   improvement_surcharge = col_double(),
##   ..   total_amount = col_double(),
##   ..   payment_type = col_double(),
##   ..   trip_type = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
anyNA(tripdata_df)
```

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

```
# look for NAs in each column
sapply(tripdata_df, function(x) sum(is.na(x)))
```

```
##              VendorID  lpep_pickup_datetime lpep_dropoff_datetime
##                     0                     0                     0
##    store_and_fwd_flag           RatecodeID          PULocationID
##                     0                     0                     0
##          DOLocationID       passenger_count         trip_distance
##                     0                     0                     0
##           fare_amount                 extra              mta_tax
##                     5                     0                     0
##            tip_amount          tolls_amount             ehail_fee
##                     0                     0               1048575
## improvement_surcharge          total_amount          payment_type
##                     0                     5                     0
##             trip_type
##                     3
```

```
summary(tripdata_df)
```

```
##     VendorID     lpep_pickup_datetime lpep_dropoff_datetime store_and_fwd_flag
##  Min.   :1.000   Length:1048575       Length:1048575        Length:1048575
##  1st Qu.:2.000   Class :character     Class :character      Class :character
##  Median :2.000   Mode  :character     Mode  :character      Mode  :character
##  Mean   :1.827
##  3rd Qu.:2.000
##  Max.   :2.000
##
##    RatecodeID      PULocationID  DOLocationID   passenger_count
##  Min.   : 1.000   Min.   :  1   Min.   :  1.0   Min.   :0.000
##  1st Qu.: 1.000   1st Qu.: 49   1st Qu.: 61.0   1st Qu.:1.000
##  Median : 1.000   Median : 82   Median :129.0   Median :1.000
##  Mean   : 1.072   Mean   :110   Mean   :128.5   Mean   :1.359
##  3rd Qu.: 1.000   3rd Qu.:166   3rd Qu.:191.0   3rd Qu.:1.000
##  Max.   :99.000   Max.   :265   Max.   :265.0   Max.   :9.000
##
##  trip_distance      fare_amount          extra            mta_tax
##  Min.   :  0.000   Min.   :-183.00   Min.   :-4.5000   Min.   :-0.5000
##  1st Qu.:  0.990   1st Qu.:   6.00   1st Qu.: 0.0000   1st Qu.: 0.5000
##  Median :  1.700   Median :   9.00   Median : 0.0000   Median : 0.5000
##  Mean   :  2.662   Mean   :  11.72   Mean   : 0.3445   Mean   : 0.4882
##  3rd Qu.:  3.260   3rd Qu.:  14.00   3rd Qu.: 0.5000   3rd Qu.: 0.5000
##  Max.   :140.620   Max.   : 999.99   Max.   : 4.5000   Max.   : 0.5000
##                    NA's   :5
##    tip_amount       tolls_amount      ehail_fee      improvement_surcharge
##  Min.   : -2.720   Min.   :  0.0000   Mode:logical   Min.   :-0.3000
##  1st Qu.:  0.000   1st Qu.:  0.0000   NA's:1048575   1st Qu.: 0.3000
##  Median :  0.000   Median :  0.0000                  Median : 0.3000
##  Mean   :  1.037   Mean   :  0.0858                  Mean   : 0.2932
##  3rd Qu.:  1.760   3rd Qu.:  0.0000                  3rd Qu.: 0.3000
##  Max.   :295.000   Max.   :557.5500                  Max.   : 0.3000
##
##   total_amount      payment_type     trip_type
##  Min.   :-183.00   Min.   :1.000   Min.   :1.000
##  1st Qu.:   7.80   1st Qu.:1.000   1st Qu.:1.000
##  Median :  10.80   Median :1.000   Median :1.000
##  Mean   :  13.99   Mean   :1.471   Mean   :1.018
##  3rd Qu.:  16.80   3rd Qu.:2.000   3rd Qu.:1.000
##  Max.   : 999.99   Max.   :5.000   Max.   :2.000
##  NA's   :5                         NA's   :3
```

We loaded the NYC Green Taxi Trip Records data on R and did some data exploration. There are 1,048,575 rows and 19 columns. Out of 19 variables all are numerics except 4 variables. lpep_pickup_datetime,lpep_dropoff_datetime and store_and_fwd_flag are character type, and ehail_fee is logical type.

We can see from our glimpse, str and summary, and as well as inforamtion from data dictionary, we have lots of categorical variables in our data. And we should choose our visualization accordingly for data exploration. VendorID, RatecodeID, store_and_fwd_flag, PULocationID, DOLocationID, payment_type and trip_type are categorical varaibles.

There are some NA values in the data. fare_amount and total_amount has 5, trip_type has 3 and ehail_fee entirely consists of NAs only. With the summary data, we can see that trip_distance, fare_amount,

tip_amount, tolls_amount, total_amount have very huge difference in maximum and minimum values - some minimum value in negative and maximum value in hundreds, this indicate that those categories may have outliers.

In order to further analysis, we first splitted the data into 3 data frames - categorical_df contains all categorical values and tip_amount. numeric_df contains all continuous varaibles and tip_amount. And finally, date_df which contains pickup, dropoff dates and location with tip_amount. Further, since we are analyzing the data from 2018, we observed that there are some data from 2009 which we filtered out as well.We have decided to not to include mta_tax,improvement_surcharge, extra in our numeric data as they both have fixed certain values and we will ne creating bar plots for those. ehail_fee has only NAs so there is no point in creating a plot.

```
# select only the categorical variables and target variable - tip_amount
categorical_df <- tripdata_df %>%
  select(VendorID, RatecodeID, store_and_fwd_flag,payment_type, trip_type, tip_amount)
head(categorical_df)
```

```
## # A tibble: 6 x 6
##   VendorID RatecodeID store_and_fwd_flag payment_type trip_type tip_amount
##      <dbl>      <dbl> <chr>                     <dbl>     <dbl>      <dbl>
## 1        2          1 N                             2         1          0
## 2        2          1 N                             2         1          0
## 3        2          1 N                             2         1          0
## 4        2          1 N                             3         1          0
## 5        2          1 N                             2         1          0
## 6        2          1 N                             2         1          0
```

```
# select only the continuous varaible and target varaible - tip_amount
numeric_df <- tripdata_df %>%
  select(passenger_count, trip_distance, fare_amount, tip_amount, tolls_amount, total_amount)

head(numeric_df)
```

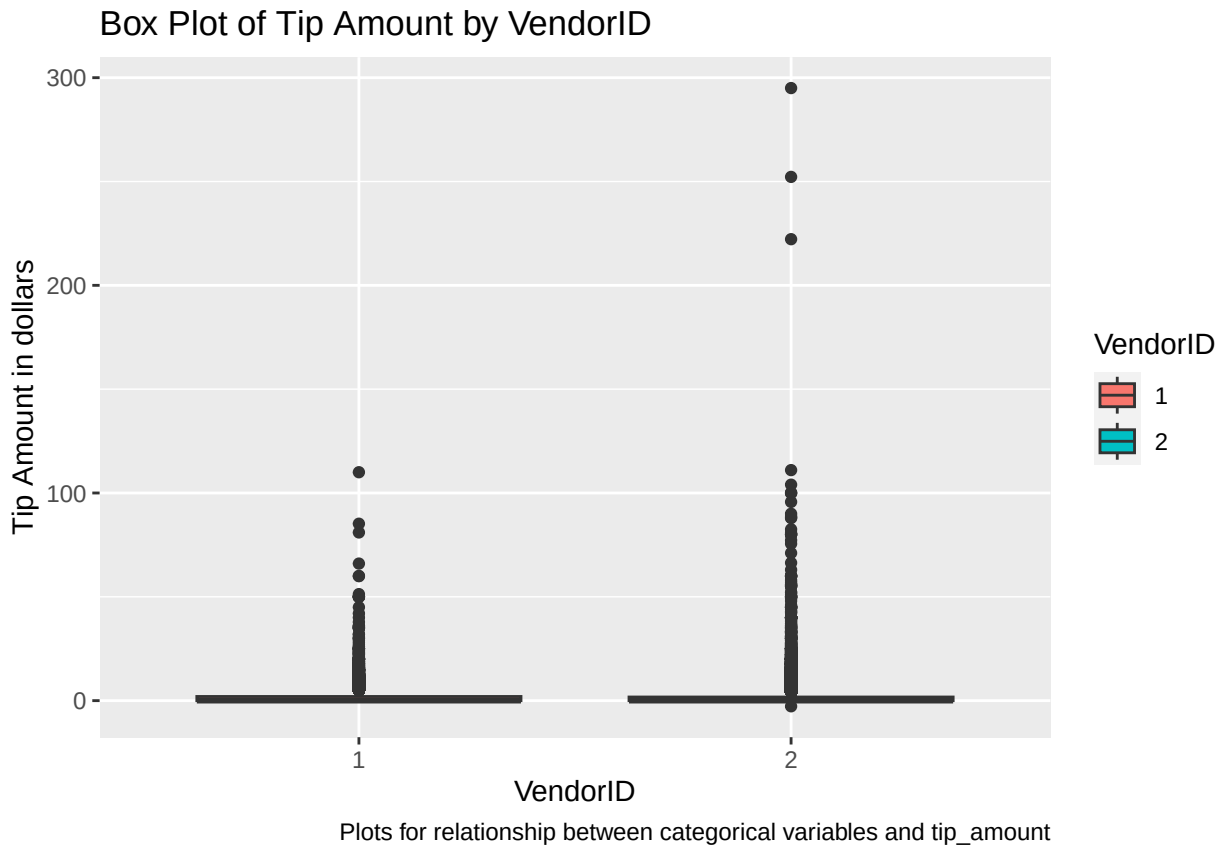
```
## # A tibble: 6 x 6
##   passenger_count trip_distance fare_amount tip_amount tolls_amount total_amount
##             <dbl>         <dbl>       <dbl>      <dbl>        <dbl>        <dbl>
## 1               5          0.7           6          0            0          7.3
## 2               5          3.5          14.5        0            0         15.8
## 3               1          2.14         10          0            0         11.3
## 4               1          0.03         -3          0            0         -4.3
## 5               1          0.03          3          0            0          4.3
## 6               1          5.63         21          0            0         22.3
```

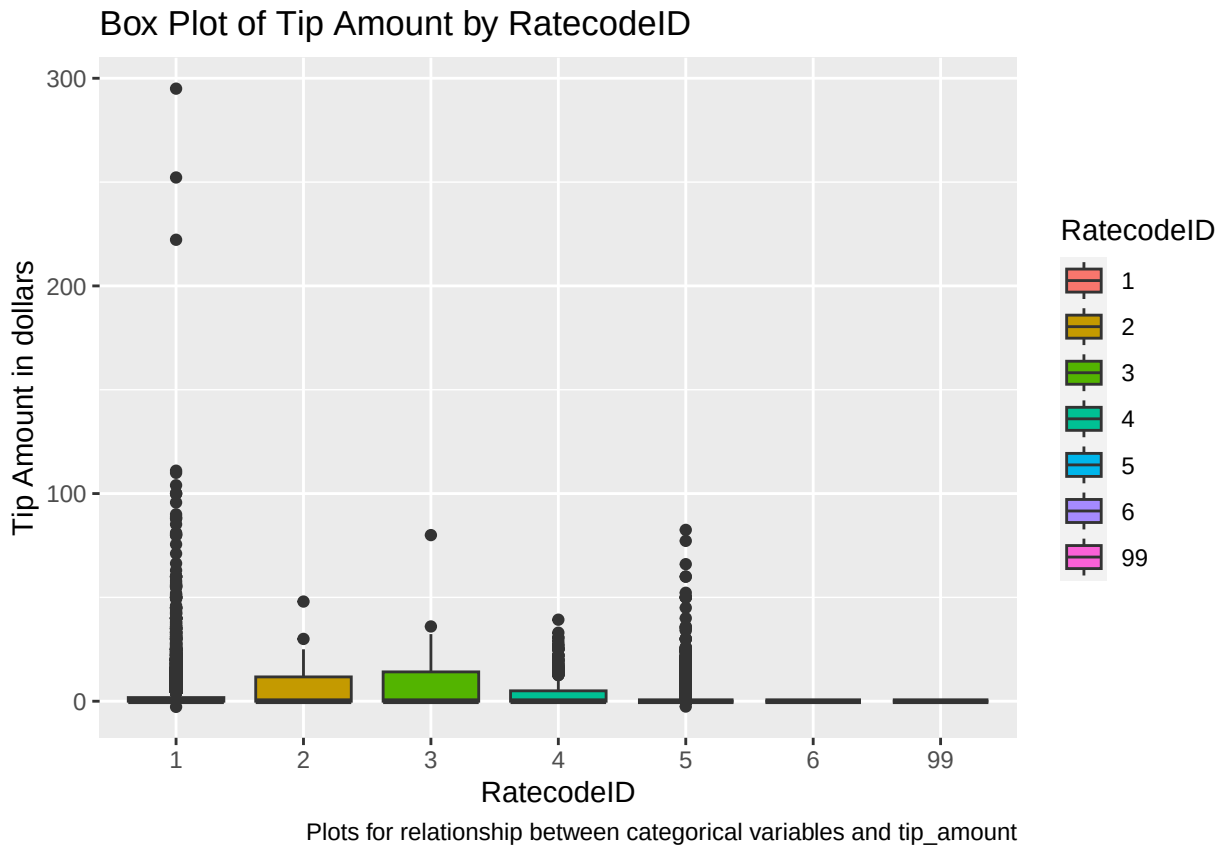
```
# relation between categorical variables and tip amount
col_to_fac <- c("VendorID","RatecodeID","store_and_fwd_flag", "payment_type", "trip_type")

factor_df <- categorical_df %>%
  mutate(across(col_to_fac, as.factor))
```

Data exploration and visualization for categorical variables and target variable

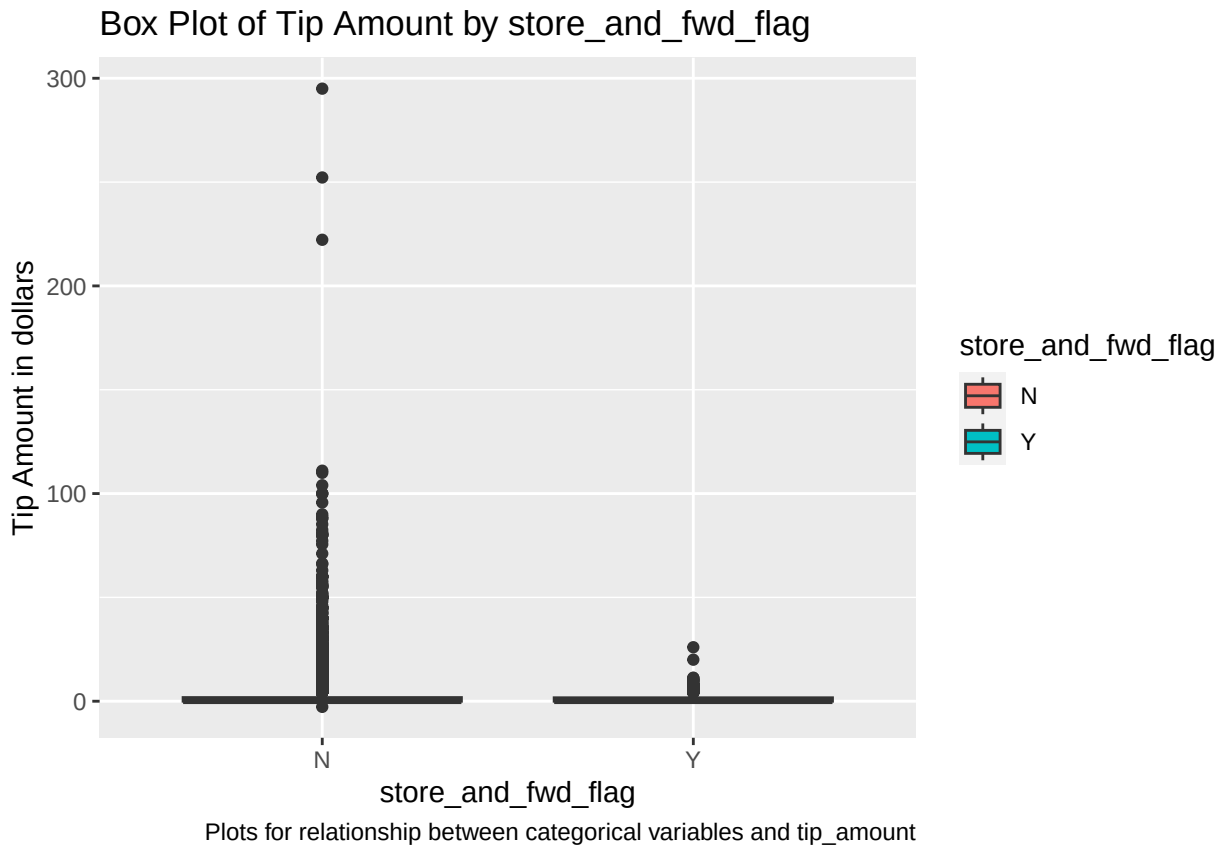
```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `across(col_to_fac, as.factor)`.
## Caused by warning:
## ! Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(col_to_fac)
##
##   # Now:
##   data %>% select(all_of(col_to_fac))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
```

```
for (col in col_to_fac) {
  print(ggplot(factor_df, aes(x = .data[[col]], y = tip_amount, fill = .data[[col]])) +
    geom_boxplot() +
    labs(title = paste("Box Plot of Tip Amount by", col), x = col, y = "Tip Amount in dollars", caption
}
```

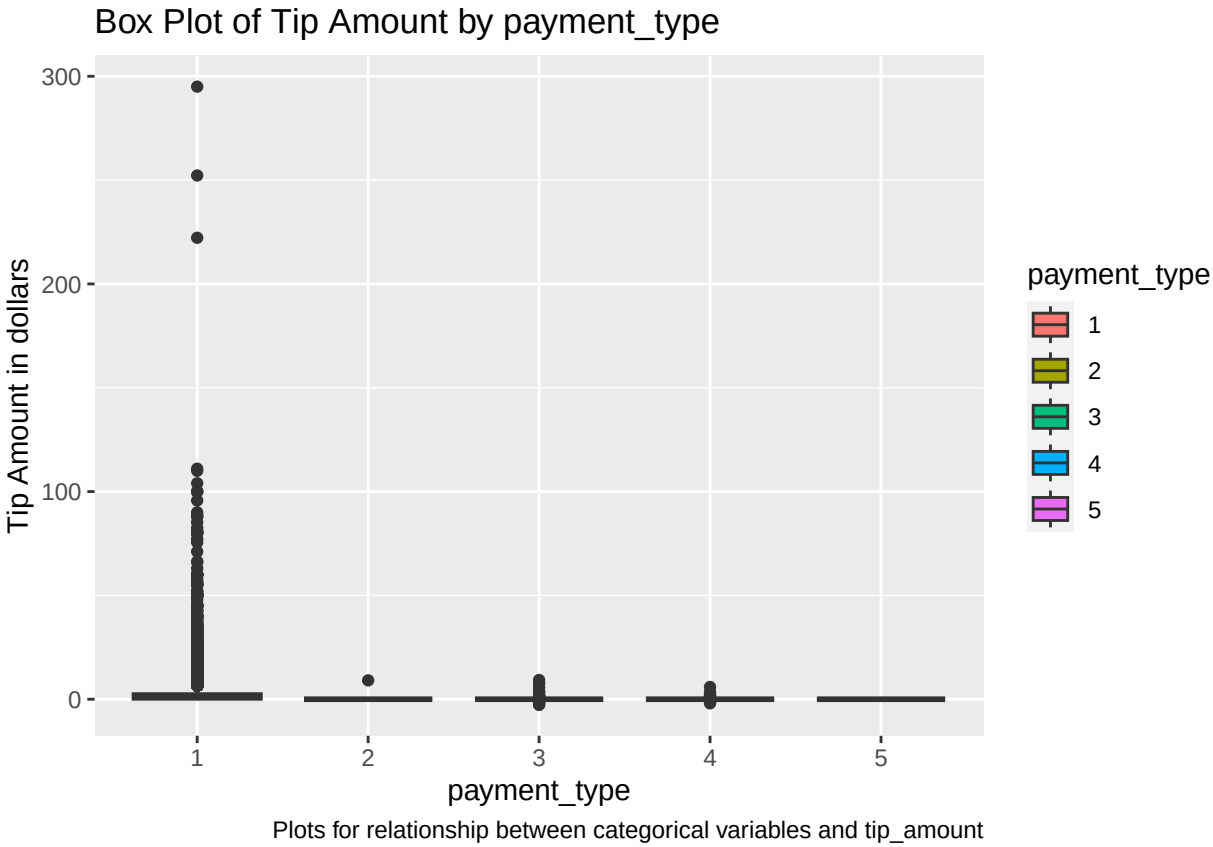




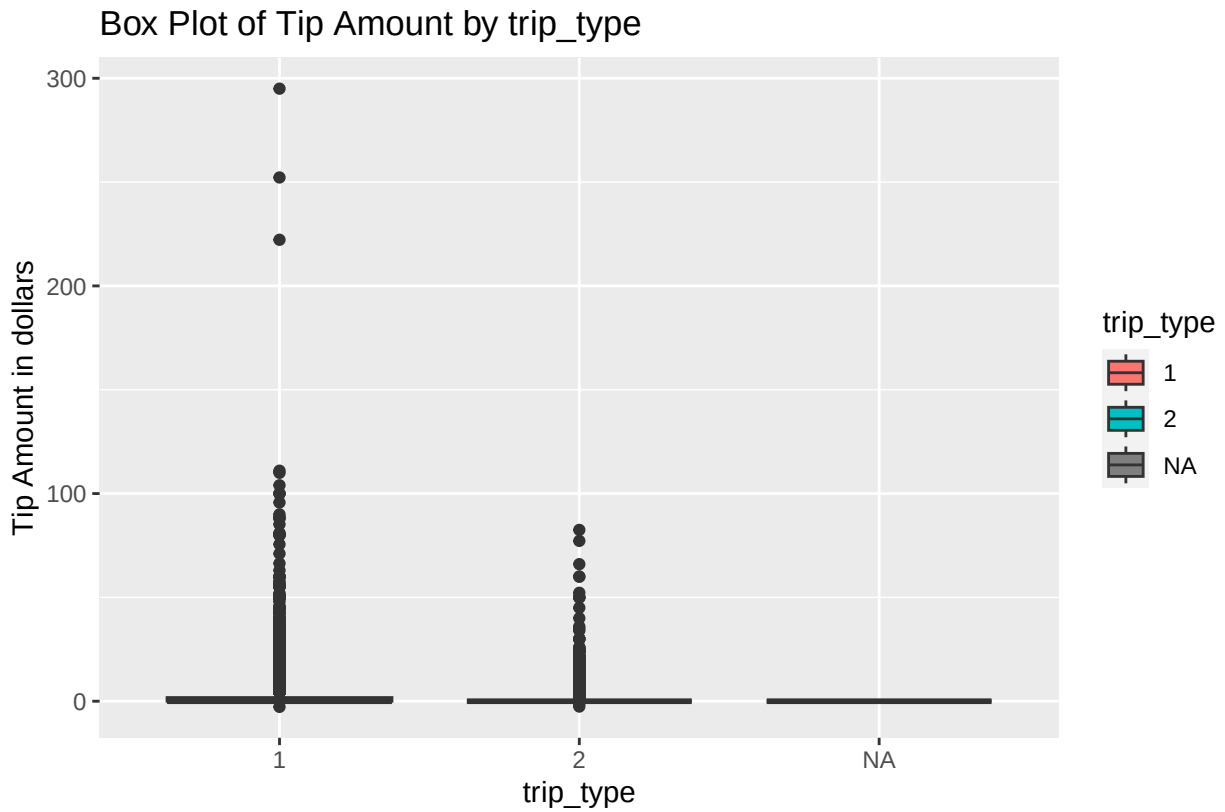
Plots for relationship between categorical variables and tip_amount



Plots for relationship between categorical variables and tip_amount



Plots for relationship between categorical variables and tip_amount



Plots for relationship between categorical variables and tip_amount

Based on the box plot for VendorID, we can see that the shape of the box is similar which indicate similar data dispersion in both ID 1 and ID 2. However, both the IDs have outliers and the outlier tip amounts are on the higher side.

Based on the box plot for RatecodeID, we can see that except for ID 6 and 99 all have outliers and ID 1 and 5 also have negative tip amounts as outliers. The the boxes for 2 , 3 and 4 seems longer which indicate that there is variability in the data.

Based on the store_and_fwd_flg box plot, we can observe that category 'N' has higher tip_amount as outliers than 'Y' category.

Based on the payment_type plot, we can see that type 1 is the one with highest outliers. Type 5 dont have any outliers.

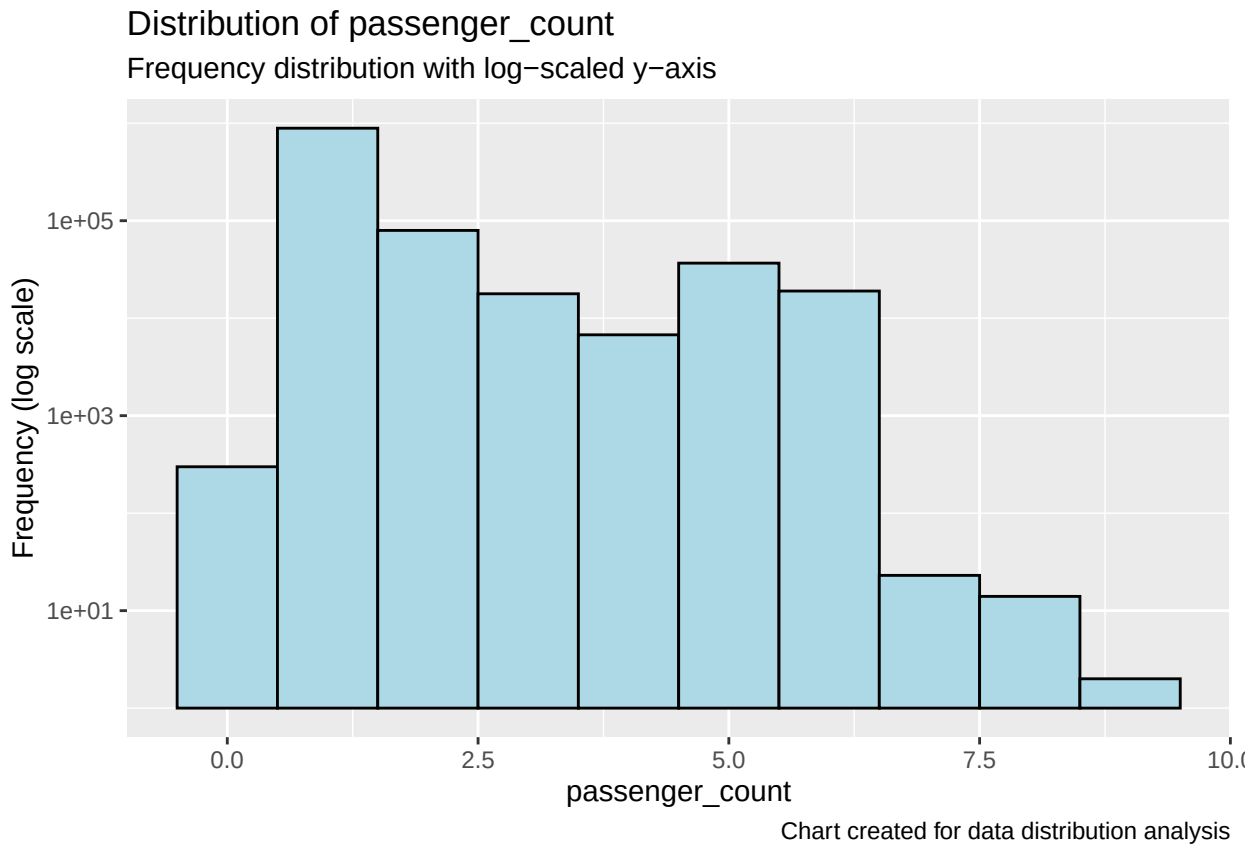
Based on trip_type plot, we can see that type 1 has more outliers than type 2, and both categories have negative tip amount as oultiers as well. And we can see that there are some NA values as well.

Distribution of the numeric features For better visualization of the data we are using log scale. We used the numeric_df to look at the distribution of the data.

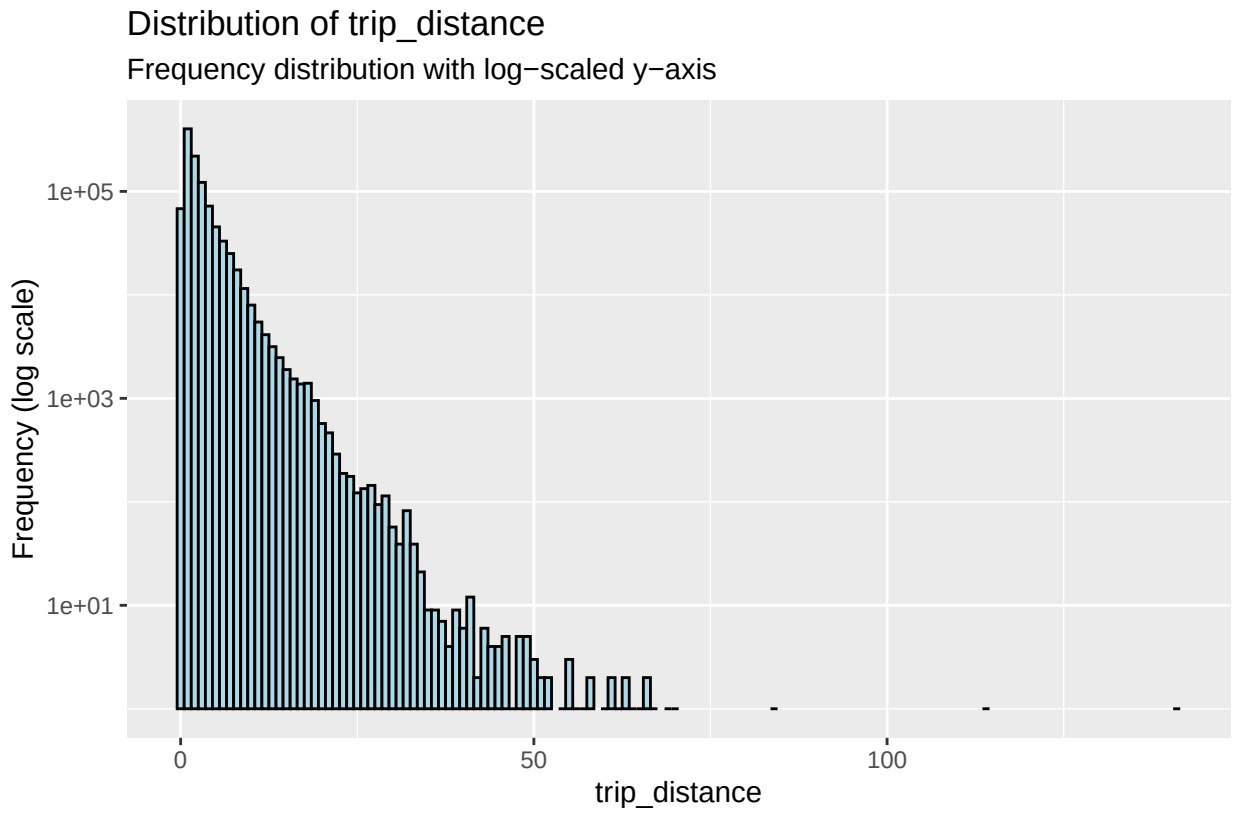
```
# rescale the y-axis using a log scale to improve the visualization
histograms <- lapply(1:ncol(numeric_df), function(i) {
  ggplot(data = numeric_df, aes(x = numeric_df[[i]])) +
    geom_histogram(binwidth = 1, fill = "lightblue", color = "black") +
    labs(
        title = paste("Distribution of",names(numeric_df)[i]),
        subtitle = "Frequency distribution with log-scaled y-axis",
        x = paste(names(numeric_df)[i]),
```

```
        y = "Frequency (log scale)",
        caption = "Chart created for data distribution analysis") +
    scale_y_log10()  # Add log scale to the y-axis
})

print(histograms)
```

```
## [[1]]

## Warning: Use of `numeric_df[[i]]` is discouraged.
## i Use `.data[[i]]` instead.
```




```
##
## [[2]]

## Warning: Use of `numeric_df[[i]]` is discouraged.
## i Use `.data[[i]]` instead.

## Warning: Transformation introduced infinite values in continuous y-axis

## Warning: Removed 71 rows containing missing values (`geom_bar()`).
```




```
##
## [[3]]
```

```
## Warning: Use of ‘numeric_df[[i]]‘ is discouraged.
## i Use ‘.data[[i]]‘ instead.
```

```
## Warning: Removed 5 rows containing non-finite values (‘stat_bin()‘).
```

```
## Warning: Transformation introduced infinite values in continuous y-axis
```

```
## Warning: Removed 940 rows containing missing values (‘geom_bar()‘).
```

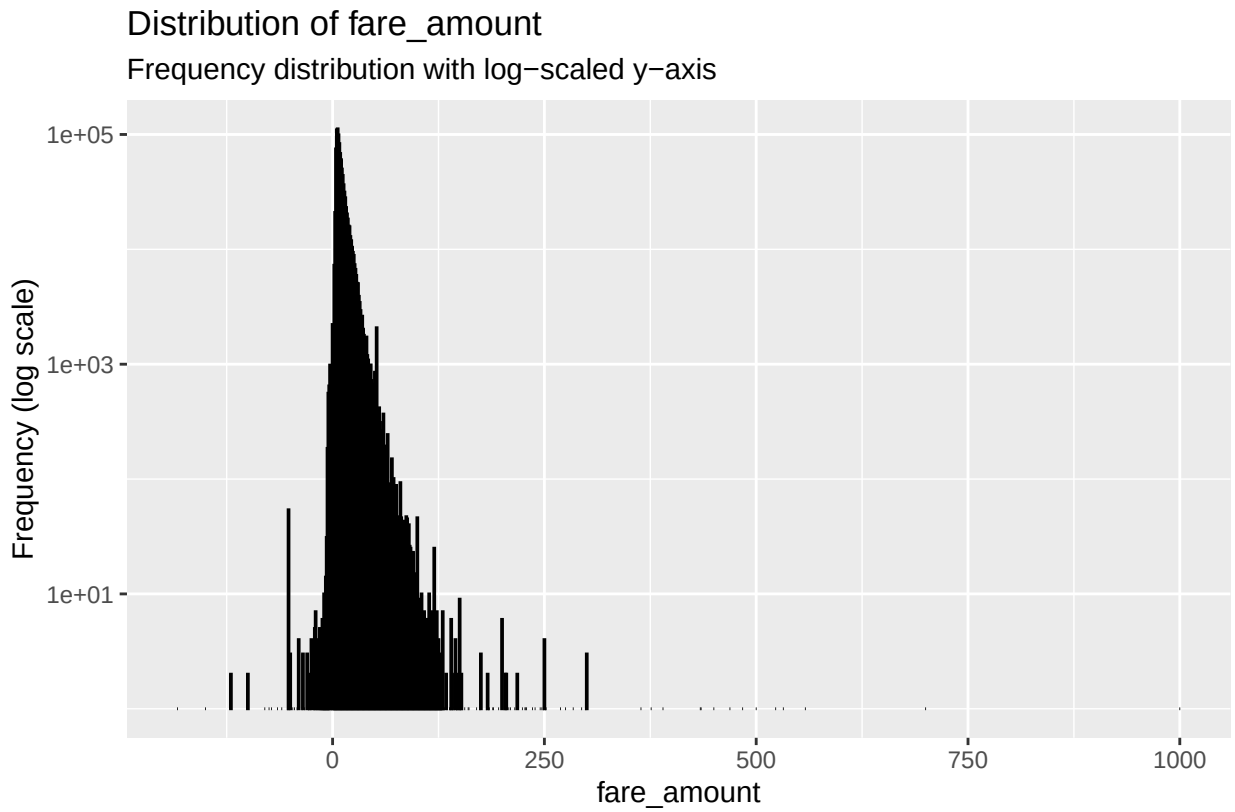
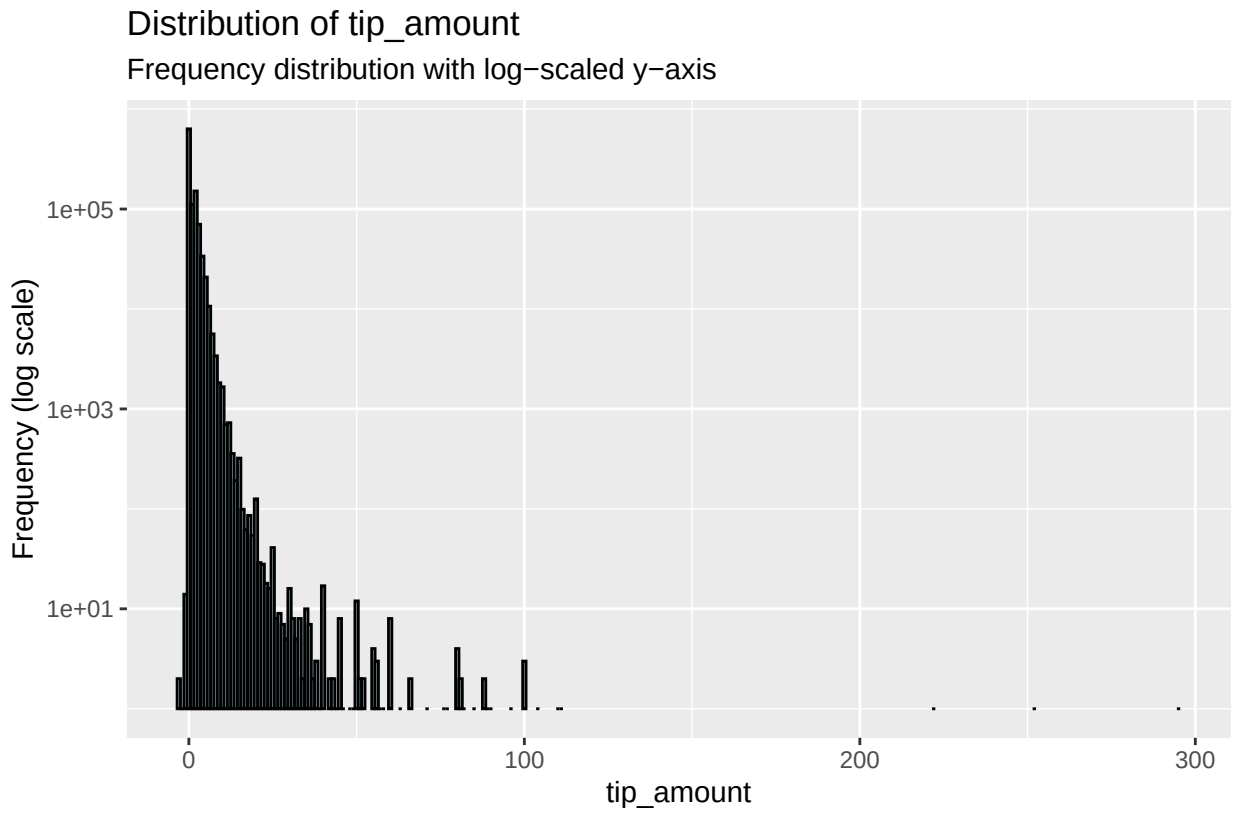
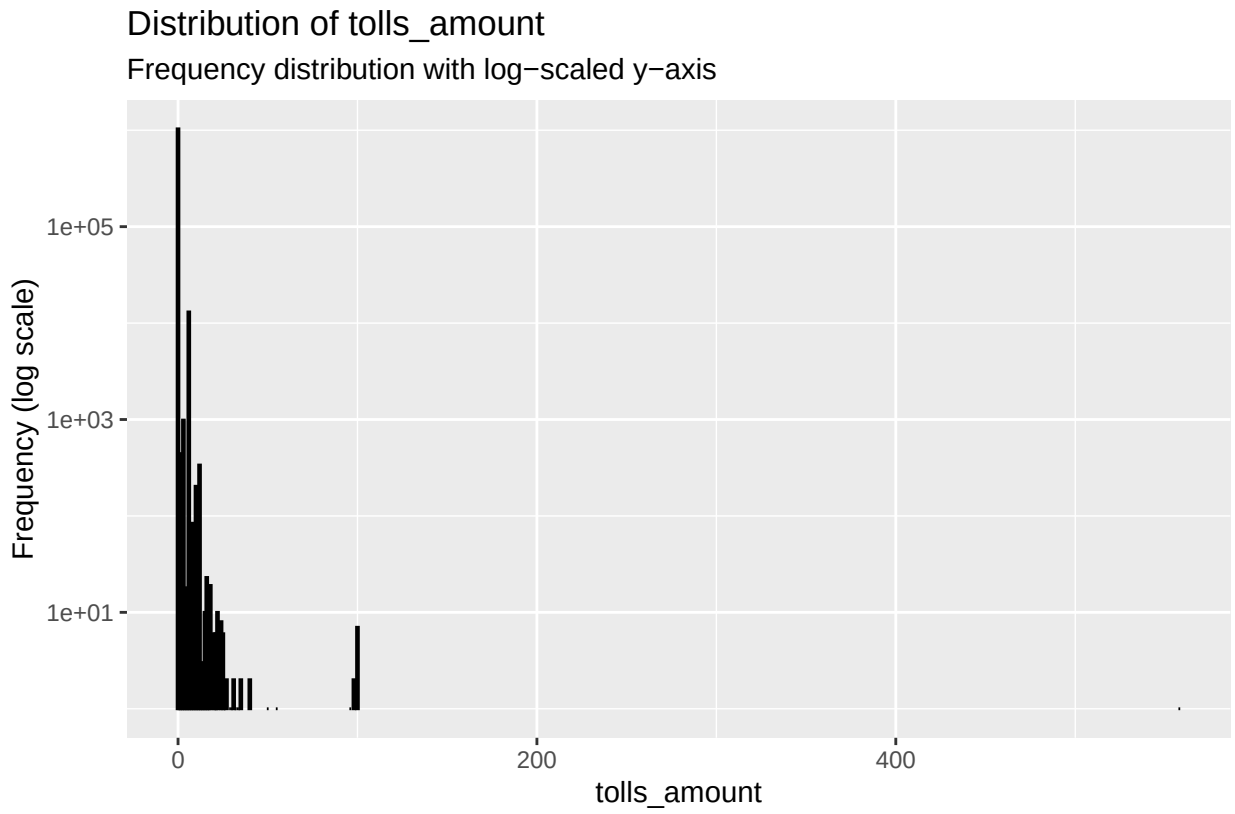


Chart created for data distribution analysis

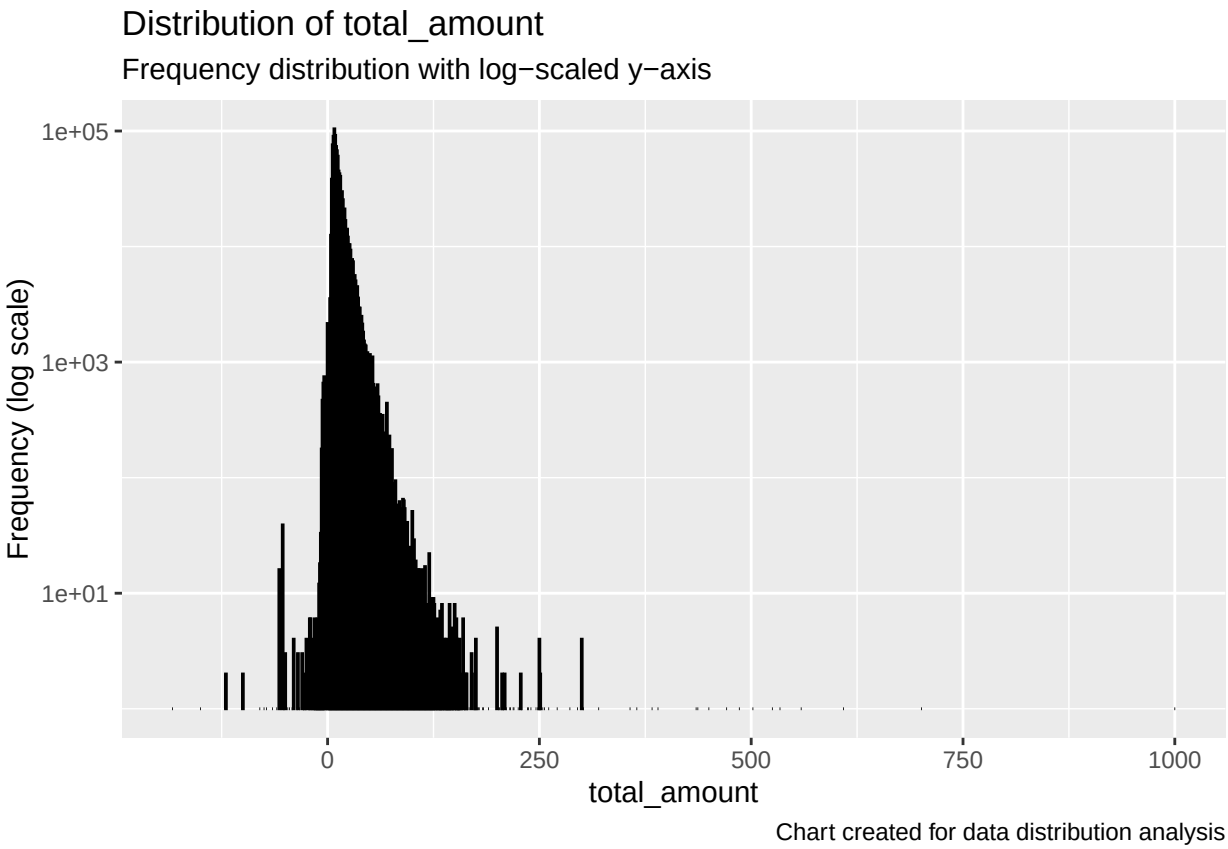
```
##
## [[4]]

## Warning: Use of ‘numeric_df[[i]]‘ is discouraged.
## i Use ‘.data[[i]]‘ instead.

## Warning: Transformation introduced infinite values in continuous y-axis

## Warning: Removed 220 rows containing missing values (‘geom_bar()‘).
```




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

## Warning: Use of ‘numeric_df[[i]]‘ is discouraged.
## i Use ‘.data[[i]]‘ instead.

## Warning: Transformation introduced infinite values in continuous y-axis

## Warning: Removed 519 rows containing missing values (‘geom_bar()‘).
```




```
##
## [[6]]

## Warning: Use of ‘numeric_df[[i]]‘ is discouraged.
## i Use ‘.data[[i]]‘ instead.

## Warning: Removed 5 rows containing non-finite values (‘stat_bin()‘).

## Warning: Transformation introduced infinite values in continuous y-axis

## Warning: Removed 920 rows containing missing values (‘geom_bar()‘).
```



From the passenger_count histogram that, lesser passengers count(fewer thab 2) is more common than more than 5 passengers.

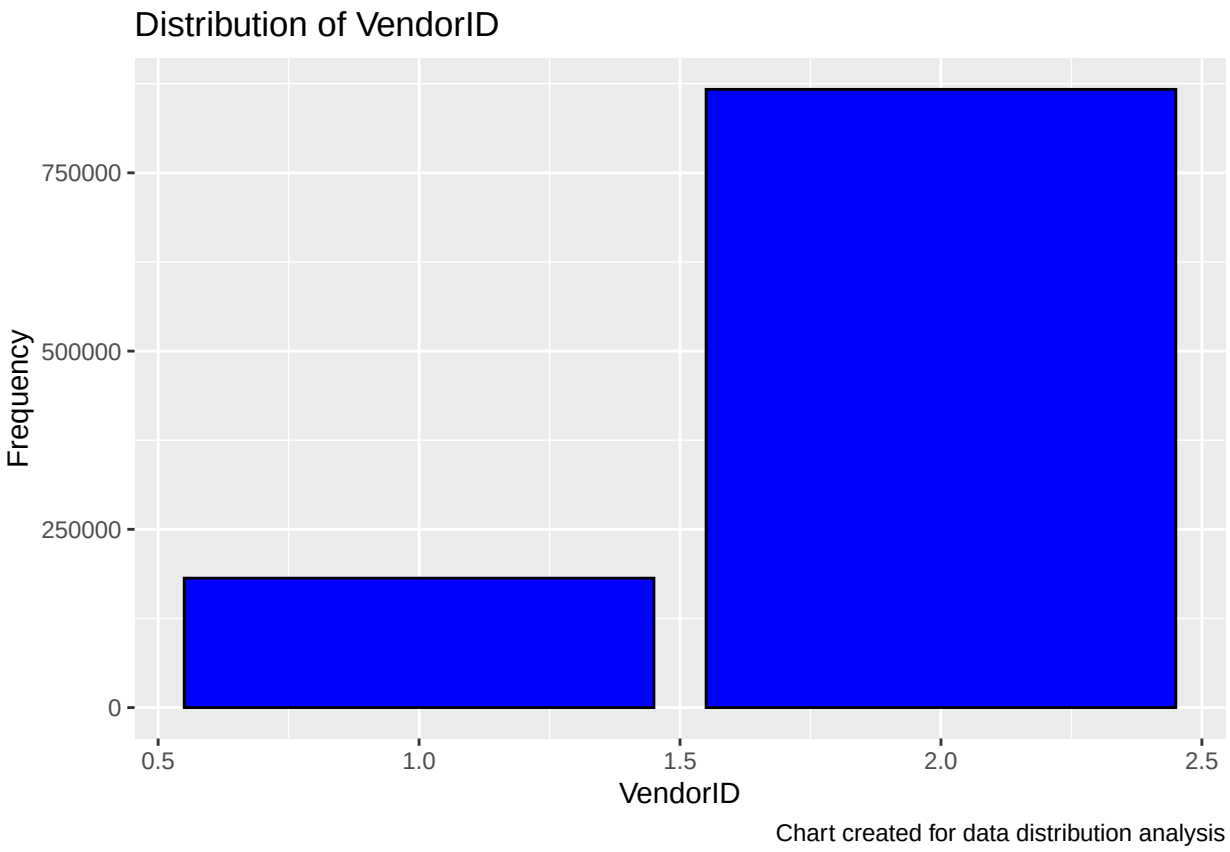
Our target variable, tip_amount is right skewed, it is due to higher number of occurrence of lower values(i.e. on the left side) and lesser occurrences of the larger values which are on the right side, thats why it forms a tail like structure on the right.

Similar pattern of distribution is seen for trip_distance, fare_amount, tolls_amount. However, distribution for total_amount looks fairly normal, looks peak somewhat centered.

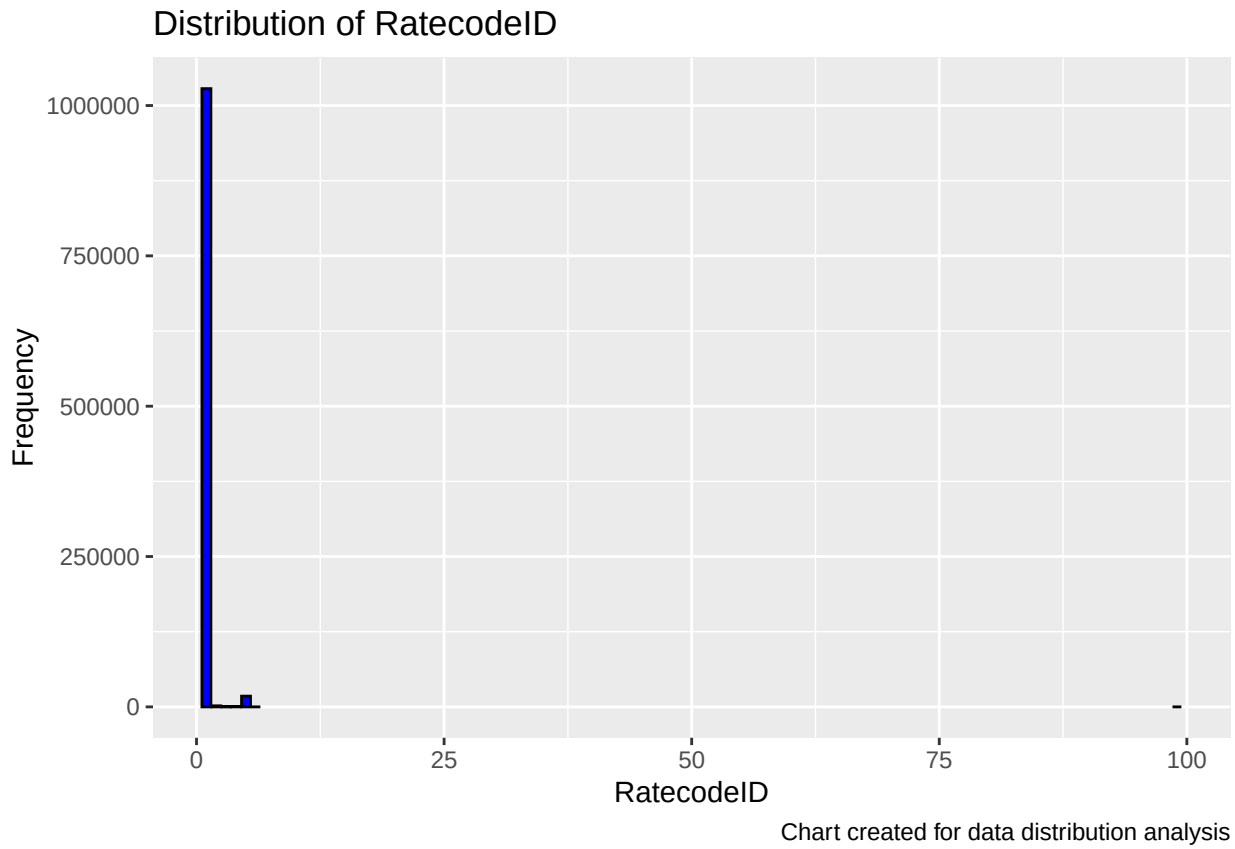
```
bar_df <- tripdata_df %>%
  select(VendorID, RatecodeID, store_and_fwd_flag,payment_type, trip_type, mta_tax, improvement_surcharg

barplots <- lapply(1:ncol(bar_df), function(i) {
  ggplot(data = categorical_df, aes(x = bar_df[[i]])) +
    geom_bar(fill = "blue", color = "black") +
    labs(
      title = paste("Distribution of", names(bar_df)[i]),
      x = paste(names(bar_df)[i]),
      y = "Frequency",
      caption = "Chart created for data distribution analysis"
    )  # Add log scale to the y-axis
})

print(barplots)
```

Bar plots for categorical variables

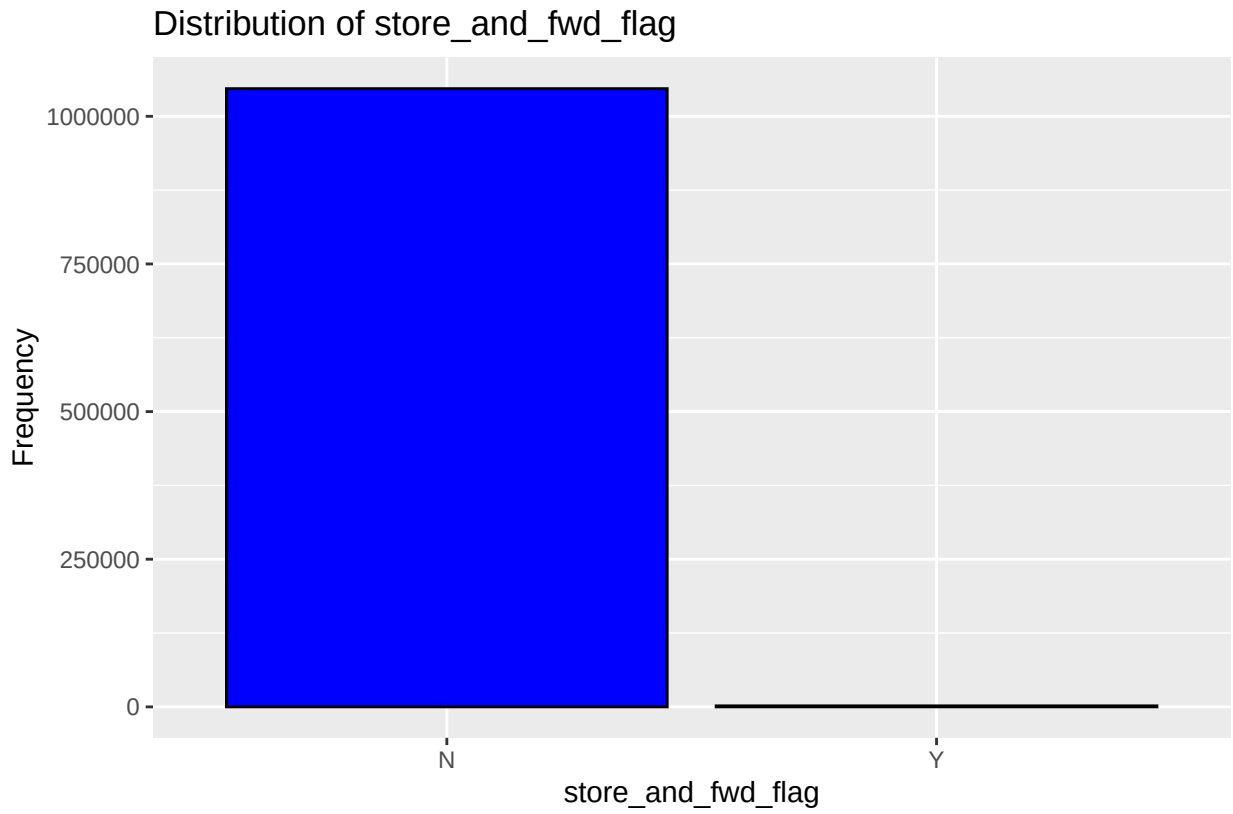
```
## [[1]]
```




```
##
## [[2]]
```


Distribution of RatecodeID
1000000
750000
500000
250000
0
Frequency
0
25
50
75
100
RatecodeID
Chart created for data distribution analysis

```
##
## [[3]]
```


Distribution of store_and_fwd_flag
1000000
750000
500000
250000
0
Frequency
N
Y
store_and_fwd_flag
Chart created for data distribution analysis

```
##
## [[4]]
```

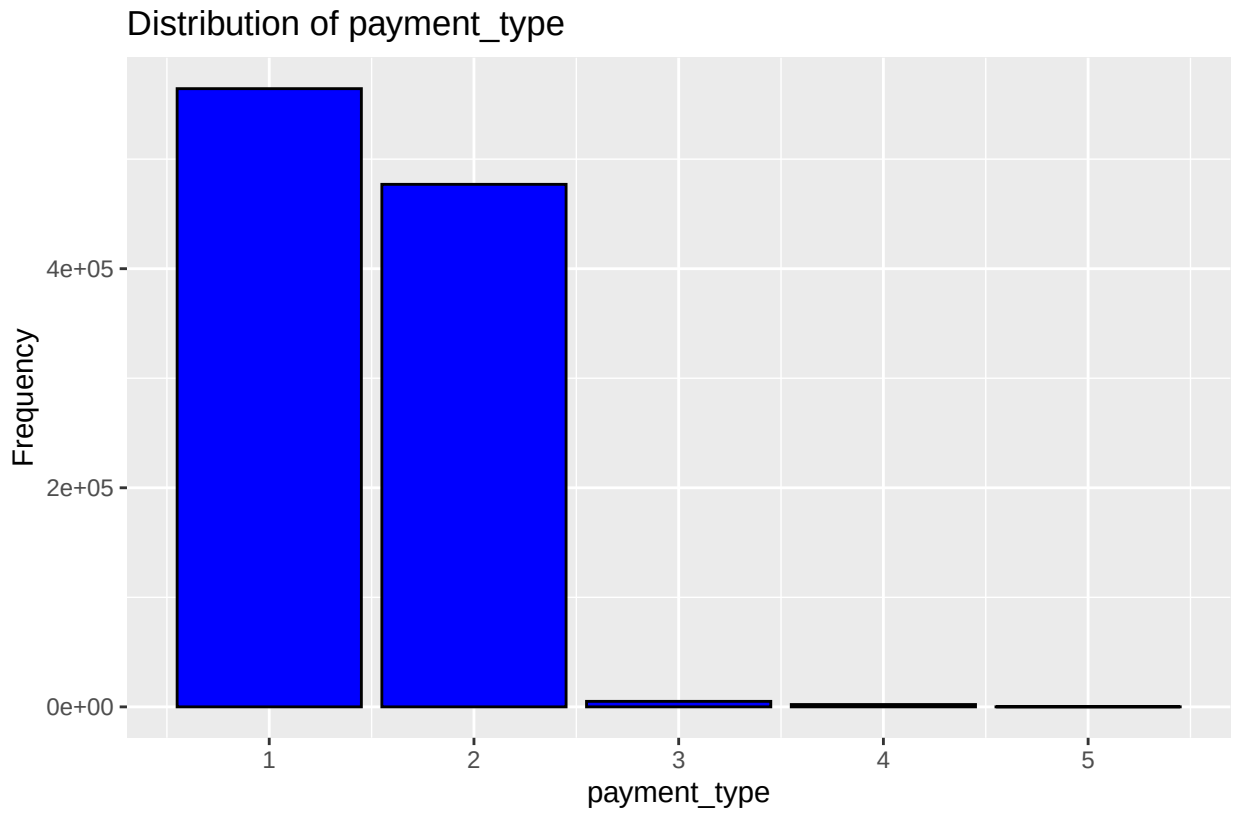
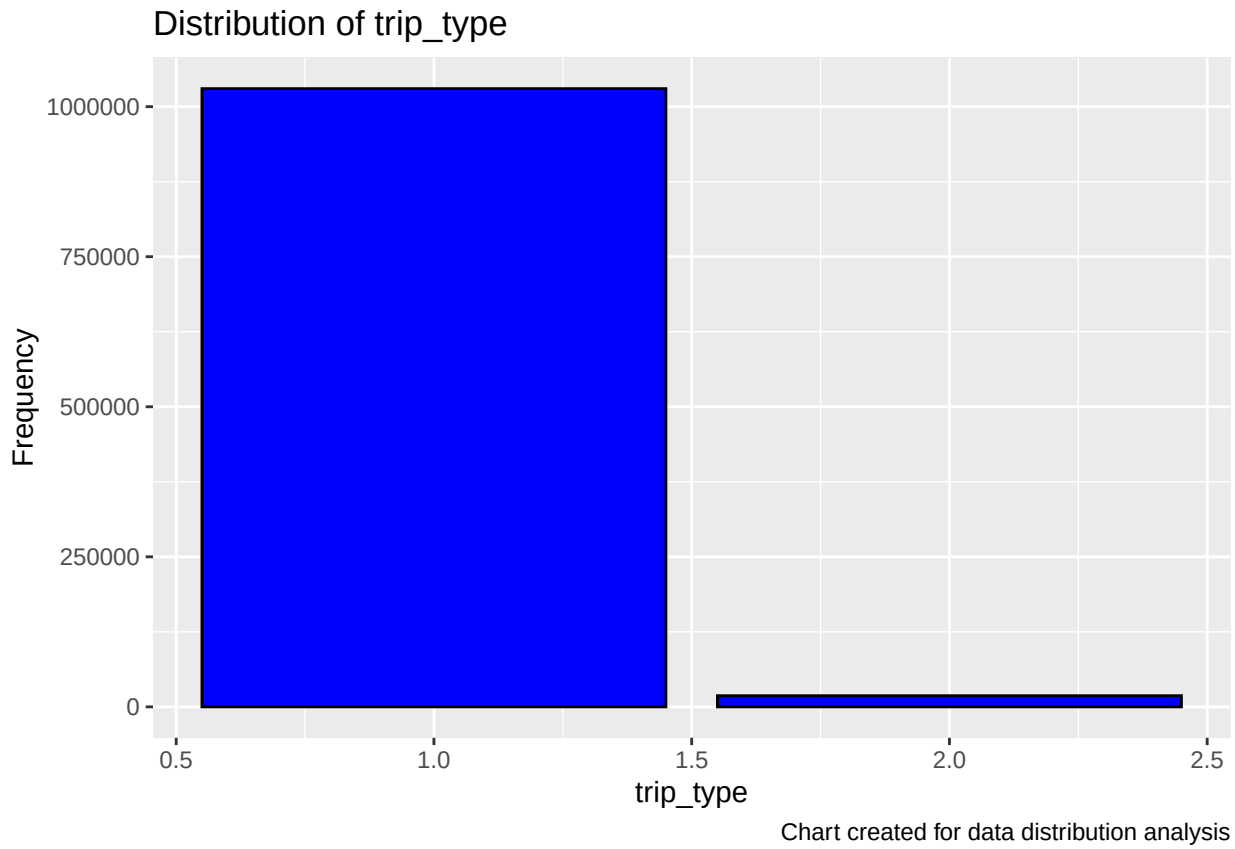
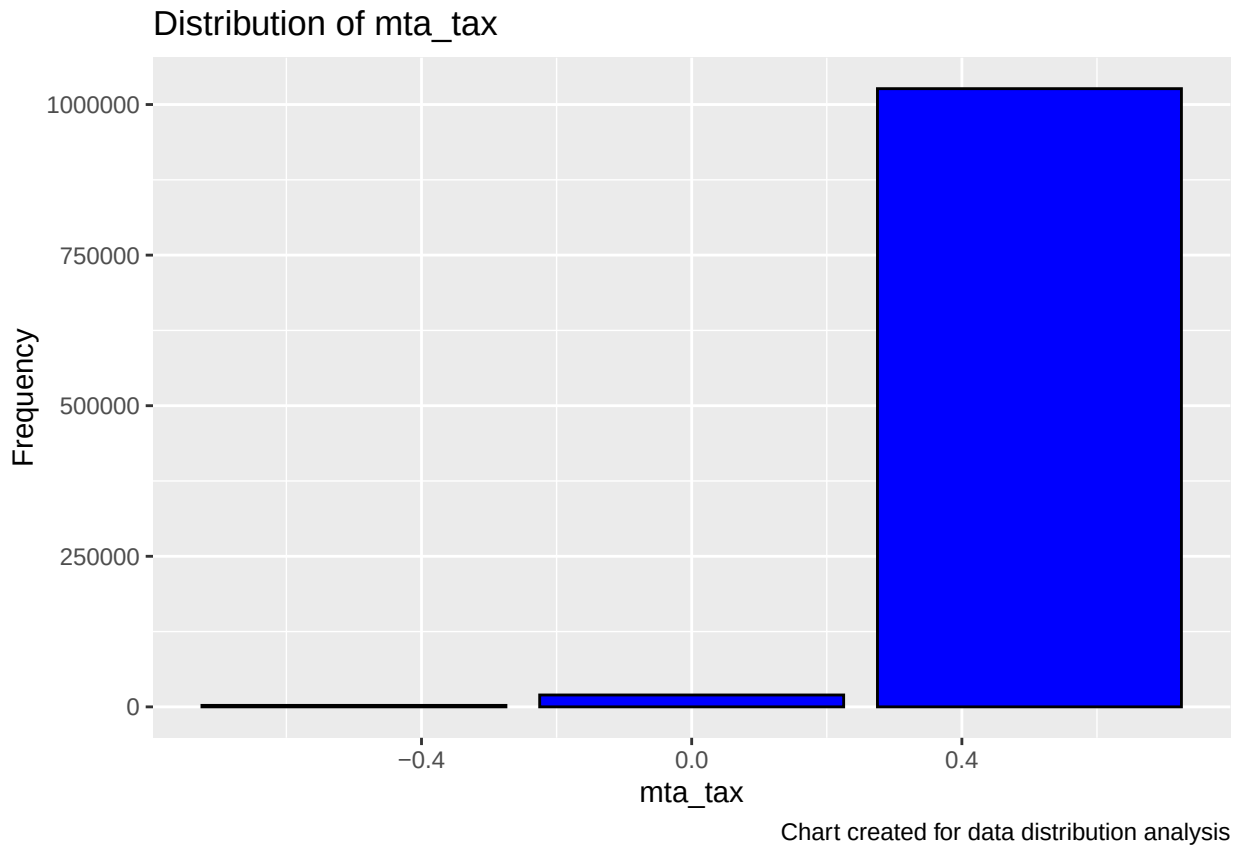


Chart created for data distribution analysis

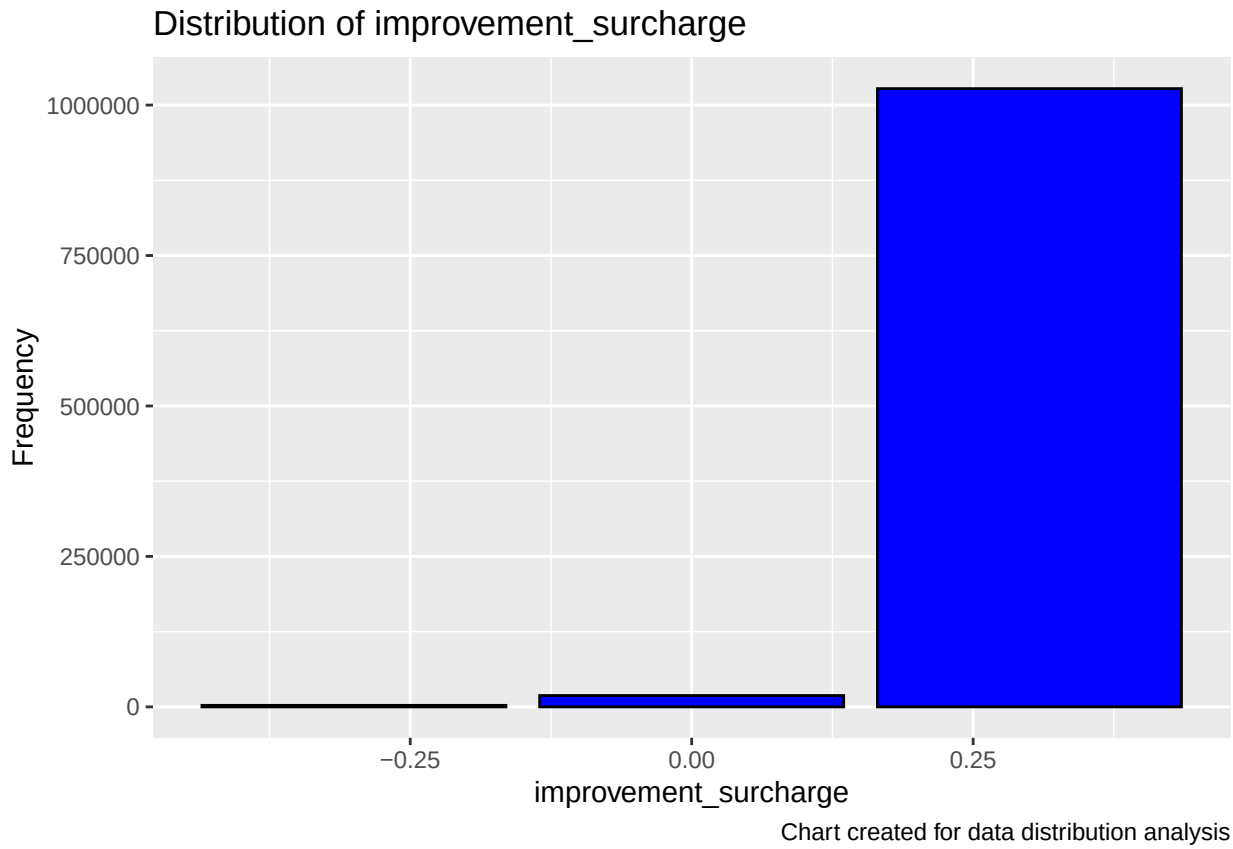
```
##
## [[5]]

## Warning: Removed 3 rows containing non-finite values (‘stat_count()‘).
```


Distribution of trip_type
1000000
750000
500000
250000
0
Frequency
0.5
1.0
1.5
2.0
2.5
trip_type
Chart created for data distribution analysis

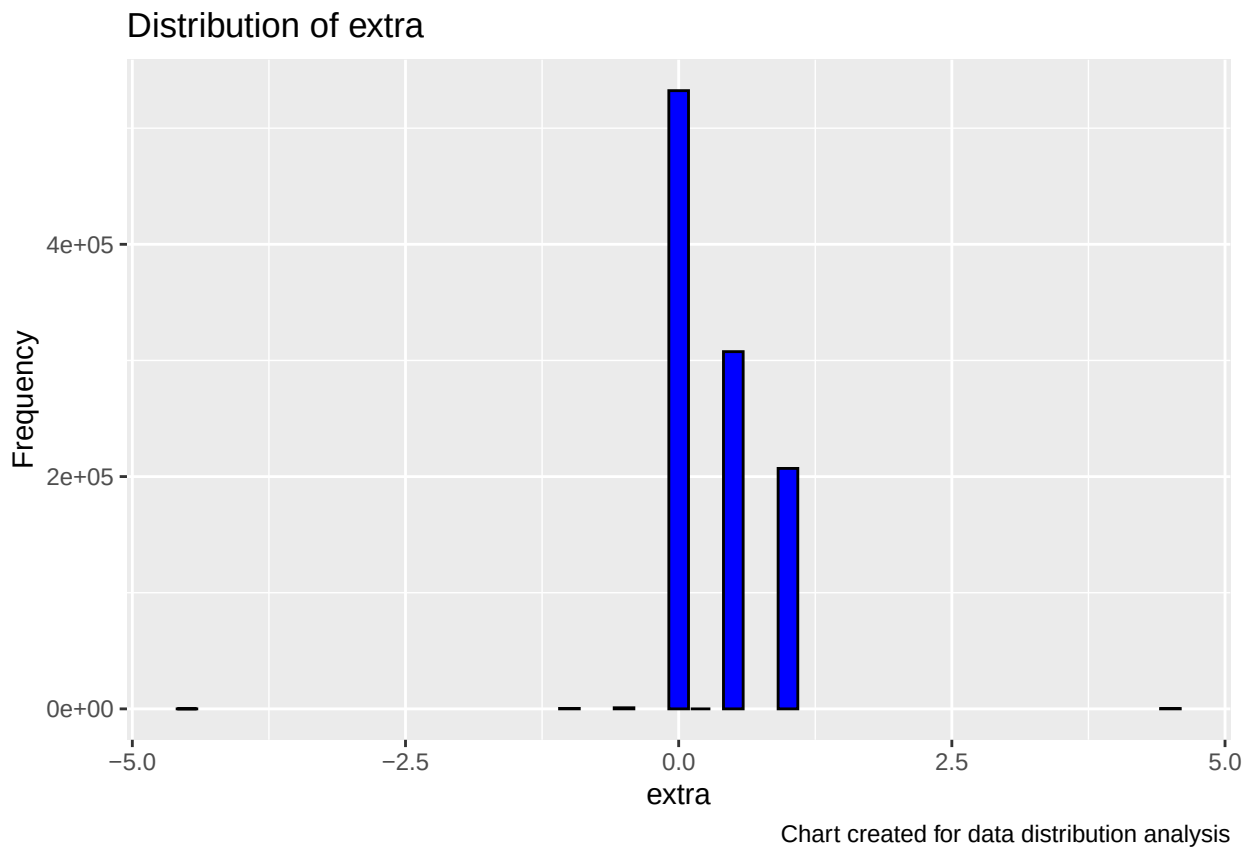
```
##
## [[6]]
```


Distribution of mta_tax
1000000
750000
500000
250000
0
Frequency
−0.4
0.0
0.4
mta_tax
Chart created for data distribution analysis

```
##
## [[7]]
```


Distribution of improvement_surcharge
1000000
750000
500000
250000
0
Frequency
−0.25
0.00
0.25
improvement_surcharge
Chart created for data distribution analysis

```
##
## [[8]]
```



According to bar plot above, we can see that VendID '2' (VeriFone Inc) has higher frequency than ID 1 - Creative Mobile Technologies, LLC.

The most prevalent Ratecode is 1 which is Standard rate.

From the store_and_fwd_flag barplot we can observe that most of the trip record were not stored before forwarding i.e type "N" was more than type "Y".

From payment_type bar plot we can observe that, among 5 types of payment payment type 1 i.e Credit card was the most common one and type 2 i.e Cash is second most common one. Least were categorized as unknown type i.e type 5.

Stree_hail(trip_type 1) was the most common one than dispatch(type 2) trip type.

Most of the trips were charged $0.4 as MTA tax and fewer were charged -$0.4 according to mta_tax barplot. Similarly, $0.25 improvement surcharge was the most common one accoring to improvement surcharge barplot.

Finally, $0 extra charge was the most common, most charges were lesser than $1.

Relationship between date and tip amount. We only first converted pickup date and dropoff date into date-time format using lubridate. And we also extracted hour of the day and only selected 2018 data as there were some 2009 data as well. Wethne grouped by the pickup and dropoff hour and claculated mean tip amount.

```
# select pickup, dropoff dates, pickup, dropoff locations and tip_amount and put in a new data frame

date_df <- tripdata_df %>%
  mutate(
```

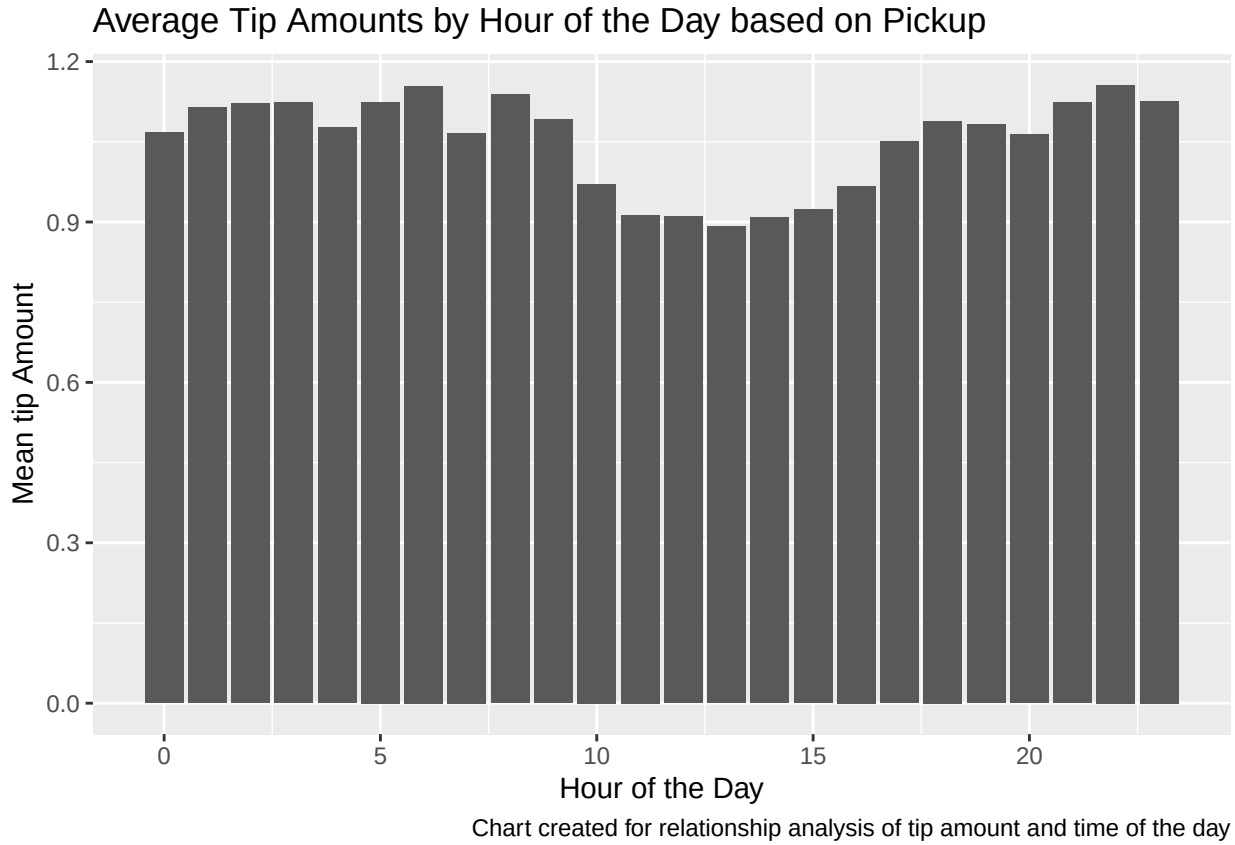
```
    lpep_pickup_datetime = mdy_hm(lpep_pickup_datetime),
    lpep_dropoff_datetime = mdy_hm(lpep_dropoff_datetime)
  ) %>%
  select(lpep_pickup_datetime,lpep_dropoff_datetime,tip_amount,PULocationID,DOLocationID)

summary(date_df)
```

```
##  lpep_pickup_datetime             lpep_dropoff_datetime
##  Min.   :2009-01-01 00:02:00.00   Min.   :2009-01-01 11:16:00.00
##  1st Qu.:2018-01-12 04:21:30.00   1st Qu.:2018-01-12 05:11:00.00
##  Median :2018-01-21 18:47:00.00   Median :2018-01-21 19:07:00.00
##  Mean   :2018-01-21 17:48:24.47   Mean   :2018-01-21 18:08:24.05
##  3rd Qu.:2018-01-31 18:58:00.00   3rd Qu.:2018-01-31 19:14:00.00
##  Max.   :2018-04-05 04:11:00.00   Max.   :2018-04-05 04:25:00.00
##    tip_amount       PULocationID  DOLocationID
##  Min.   : -2.720   Min.   :  1   Min.   :  1.0
##  1st Qu.:  0.000   1st Qu.: 49   1st Qu.: 61.0
##  Median :  0.000   Median : 82   Median :129.0
##  Mean   :  1.037   Mean   :110   Mean   :128.5
##  3rd Qu.:  1.760   3rd Qu.:166   3rd Qu.:191.0
##  Max.   :295.000   Max.   :265   Max.   :265.0
```

```
# select only data from 2018 because it had some data from 2009 as well
date_df <- date_df %>%
  mutate(hour_of_pickup = hour(lpep_pickup_datetime)) %>%
  mutate(hour_of_dropoff = hour(lpep_dropoff_datetime)) %>%
  mutate(year_of_pickup = year(lpep_pickup_datetime))  %>%
  filter(year_of_pickup == 2018)

summary(date_df)
```

```
##  lpep_pickup_datetime             lpep_dropoff_datetime
##  Min.   :2018-01-01 00:00:00.00   Min.   :2018-01-01 00:00:00.00
##  1st Qu.:2018-01-12 04:34:00.00   1st Qu.:2018-01-12 05:31:00.00
##  Median :2018-01-21 18:48:00.00   Median :2018-01-21 19:08:00.00
##  Mean   :2018-01-21 19:00:49.76   Mean   :2018-01-21 19:20:48.84
##  3rd Qu.:2018-01-31 18:59:00.00   3rd Qu.:2018-01-31 19:14:00.00
##  Max.   :2018-04-05 04:11:00.00   Max.   :2018-04-05 04:25:00.00
##    tip_amount       PULocationID  DOLocationID   hour_of_pickup
##  Min.   : -2.720   Min.   :  1   Min.   :  1.0   Min.   : 0.00
##  1st Qu.:  0.000   1st Qu.: 49   1st Qu.: 61.0   1st Qu.:10.00
##  Median :  0.000   Median : 82   Median :129.0   Median :15.00
##  Mean   :  1.037   Mean   :110   Mean   :128.5   Mean   :13.86
##  3rd Qu.:  1.760   3rd Qu.:166   3rd Qu.:191.0   3rd Qu.:19.00
##  Max.   :295.000   Max.   :265   Max.   :265.0   Max.   :23.00
##  hour_of_dropoff year_of_pickup
##  Min.   : 0.00   Min.   :2018
##  1st Qu.:10.00   1st Qu.:2018
##  Median :15.00   Median :2018
##  Mean   :13.91   Mean   :2018
##  3rd Qu.:19.00   3rd Qu.:2018
##  Max.   :23.00   Max.   :2018
```

```r
# Group by hour and calculate the mean tip_amount for each hour, for pickup
hourly_tip_mean <- date_df %>%
  group_by(hour_of_pickup) %>%
  summarise(mean_tip_amount = mean(tip_amount, na.rm = TRUE))

# Group by hour and calculate the mean tip_amount for each hour, for dropoff
hourly_tip_mean2 <- date_df %>%
  group_by(hour_of_dropoff) %>%
  summarise(mean_tip_amount = mean(tip_amount, na.rm = TRUE))


ggplot(hourly_tip_mean, aes(x = hour_of_pickup , y = mean_tip_amount)) +
  geom_bar(stat = "summary", fun = "mean") +
  labs(title = "Average Tip Amounts by Hour of the Day based on Pickup",
       x = "Hour of the Day",
       y = "Mean tip Amount",
       caption = "Chart created for relationship analysis of tip amount and time of the day")
```




```r
ggplot(hourly_tip_mean2, aes(x = hour_of_dropoff , y = mean_tip_amount)) +
  geom_bar(stat = "summary", fun = "mean") +
  labs(title = "Average Tip Amounts by Hour of the Day based on Dropoff",
       x = "Hour of the Day",
       y = "Mean tip Amount",
       caption = "Chart created for relationship analysis of tip amount and time of the day")
```

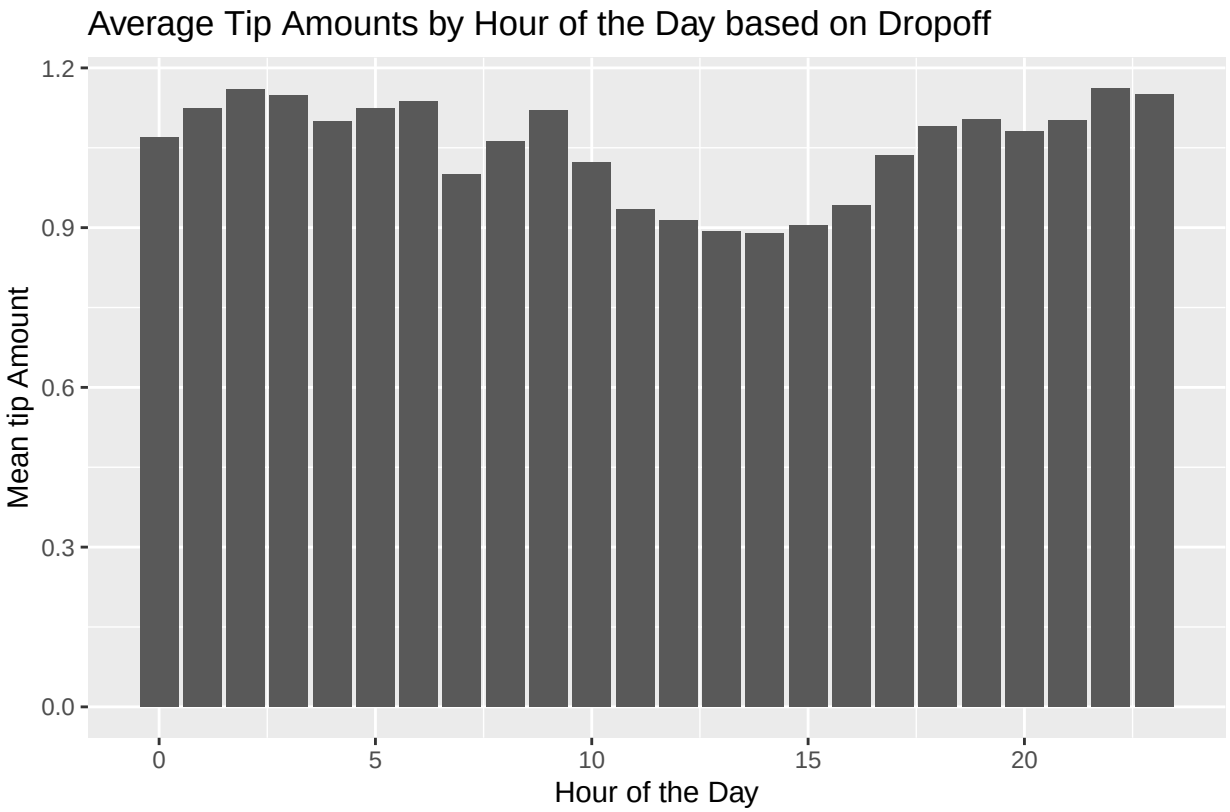


Chart created for relationship analysis of tip amount and time of the day

We found out that, tip amount varies based on hour of the day.For both pickup hour and drop off hour we got similar bar plots, showing higher tip amount in the early morning and late nights, and lower tip amount during the afternoons.

tip amount based on pickup and drop off location In order to see if tip amount varies based on location we also grouped our data based on PULocation and DOLocation.

```
date_df %>%
  group_by(PULocationID) %>%
  summarise(mean_tip_amount = mean(tip_amount, na.rm = TRUE)) %>%
  arrange(desc(mean_tip_amount)) %>%
  head(5)
```

```
## # A tibble: 5 x 2
##   PULocationID mean_tip_amount
##          <dbl>           <dbl>
## 1            1           16.4
## 2           99           12.8
## 3          214            7.33
## 4          132            7.28
## 5          138            5.96
```

```
date_df %>%
  group_by(DOLocationID) %>%
  summarise(mean_tip_amount = mean(tip_amount, na.rm = TRUE)) %>%
```

```
  arrange(desc(mean_tip_amount)) %>%
  head(5)
```

```
## # A tibble: 5 x 2
##   DOLocationID mean_tip_amount
##          <dbl>           <dbl>
## 1            1           12.9
## 2          204           12.1
## 3          214           10.2
## 4           84            9.08
## 5          245            7.50
```

The mean tip maount varies based on location of pickup and drop off. The top 5 pickup location with highest mean tip amount were, 1, 99, 214, 132 and 138. And top 5 dropoff location with the highest mean tip mount were 1, 204,214, 84 and 245. Compared to dropoff pickup, pickup location 1 has the highest mean tip, with $16.42. While highest tip at drop off location was $12.85 at location 1.

Correlation

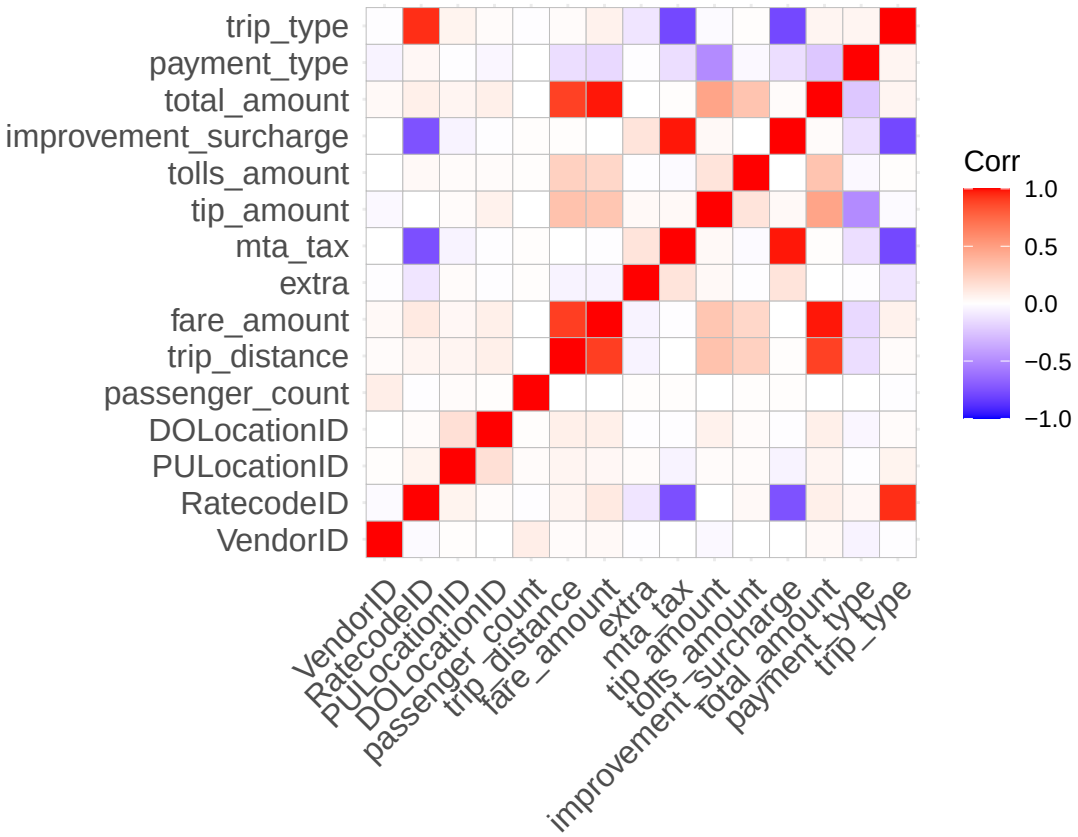
```
# select only the data with numeric type
all_num <- tripdata_df %>%
  select_if(is.numeric)
# correlation with numerical valiues
corr <- cor(all_num, use="complete.obs")

print(corr)
```

```
##                          VendorID   RatecodeID PULocationID DOLocationID
## VendorID              1.000000000 -0.016769911  0.008848187 -0.001157518
## RatecodeID           -0.016769911  1.000000000  0.058084953  0.023662357
## PULocationID          0.008848187  0.058084953  1.000000000  0.163456297
## DOLocationID         -0.001157518  0.023662357  0.163456297  1.000000000
## passenger_count       0.091670464 -0.008676655  0.016329169  0.006590991
## trip_distance         0.024861075  0.049055622  0.053266213  0.081718576
## fare_amount           0.033412715  0.107154022  0.044954761  0.075300582
## extra                -0.005761316 -0.109663412  0.016643010 -0.008900432
## mta_tax              -0.003136281 -0.762158159 -0.050177612 -0.012410496
## tip_amount           -0.030886848 -0.002638702  0.021813265  0.074675171
## tolls_amount          0.002137244  0.030404875  0.019243473  0.015309663
## improvement_surcharge -0.004289380 -0.746751523 -0.049735465 -0.013642596
## total_amount          0.025649524  0.084794462  0.046604755  0.082658063
## payment_type         -0.053905593  0.043255816 -0.007359088 -0.038432268
## trip_type            -0.010892528  0.936201851  0.055914769  0.019572261
##                      passenger_count trip_distance  fare_amount         extra
## VendorID                0.0916704643  0.0248610754  0.033412715 -0.0057613156
## RatecodeID             -0.0086766552  0.0490556218  0.107154022 -0.1096634116
## PULocationID            0.0163291685  0.0532662131  0.044954761  0.0166430097
## DOLocationID            0.0065909912  0.0817185758  0.075300582 -0.0089004315
## passenger_count         1.0000000000  0.0021171110  0.002497043  0.0099832264
## trip_distance           0.0021171110  1.0000000000  0.898969699 -0.0479061290
## fare_amount             0.0024970431  0.8989696987  1.000000000 -0.0497210166
```

```
## extra                   0.0099832264 -0.0479061290 -0.049721017  1.0000000000
## mta_tax                 0.0068479441  0.0007548377 -0.012995879  0.1430970856
## tip_amount              0.0029672505  0.3151344720  0.301978693  0.0293101773
## tolls_amount            0.0078247231  0.2407741170  0.211561337 -0.0119499184
## improvement_surcharge   0.0063969153  0.0076696244 -0.002281674  0.1425330897
## total_amount            0.0043155157  0.8911936853  0.976990348 -0.0004758442
## payment_type            0.0002877368 -0.1417985302 -0.161185298 -0.0130812686
## trip_type              -0.0104075631  0.0210684132  0.071478691 -0.1104047150
##                           mta_tax   tip_amount tolls_amount
## VendorID              -0.0031362814 -0.030886848  0.002137244
## RatecodeID            -0.7621581588 -0.002638702  0.030404875
## PULocationID          -0.0501776117  0.021813265  0.019243473
## DOLocationID          -0.0124104958  0.074675171  0.015309663
## passenger_count        0.0068479441  0.002967250  0.007824723
## trip_distance          0.0007548377  0.315134472  0.240774117
## fare_amount           -0.0129958788  0.301978693  0.211561337
## extra                  0.1430970856  0.029310177 -0.011949918
## mta_tax                1.0000000000  0.027686148 -0.016389408
## tip_amount             0.0276861480  1.000000000  0.138908909
## tolls_amount          -0.0163894079  0.138908909  1.000000000
## improvement_surcharge  0.9831822010  0.033952758 -0.002299583
## total_amount           0.0105295489  0.472833832  0.310702680
## payment_type          -0.1355514366 -0.499610071 -0.027987678
## trip_type             -0.7869271450 -0.020049681  0.006163309
##                       improvement_surcharge  total_amount  payment_type
## VendorID                       -0.004289380  0.0256495240 -0.0539055929
## RatecodeID                     -0.746751523  0.0847944624  0.0432558159
## PULocationID                   -0.049735465  0.0466047554 -0.0073590877
## DOLocationID                   -0.013642596  0.0826580628 -0.0384322680
## passenger_count                 0.006396915  0.0043155157  0.0002877368
## trip_distance                   0.007669624  0.8911936853 -0.1417985302
## fare_amount                    -0.002281674  0.9769903475 -0.1611852977
## extra                           0.142533090 -0.0004758442 -0.0130812686
## mta_tax                         0.983182201  0.0105295489 -0.1355514366
## tip_amount                      0.033952758  0.4728338320 -0.4996100713
## tolls_amount                   -0.002299583  0.3107026802 -0.0279876775
## improvement_surcharge           1.000000000  0.0226042282 -0.1368919786
## total_amount                    0.022604228  1.0000000000 -0.2441298721
## payment_type                   -0.136891979 -0.2441298721  1.0000000000
## trip_type                      -0.789865120  0.0466762205  0.0468749824
##                         trip_type
## VendorID              -0.010892528
## RatecodeID             0.936201851
## PULocationID           0.055914769
## DOLocationID           0.019572261
## passenger_count       -0.010407563
## trip_distance          0.021068413
## fare_amount            0.071478691
## extra                 -0.110404715
## mta_tax               -0.786927145
## tip_amount            -0.020049681
## tolls_amount           0.006163309
## improvement_surcharge -0.789865120
## total_amount           0.046676220
```

```
## payment_type            0.046874982
## trip_type               1.000000000
```

```
# visualize the correlation between variables
ggcorrplot(corr)
```



We did calculated correlation coefficient between all the numeric variables and also created a correlation matrix using ggcorrplot. First looking correlation between our response variable and other variables, we observed that - trip_distance and fare_amount have small positive correaltion toward tip_amount with the correlation coefficient around 0.30. The toll_amount minimal positive correlation with 0.14 with tip amount. However total_amount and payment_type had moderative correlation with the tip_amount - total_amount has positive correlation with 0.47 and payment_type has negative correaltion.

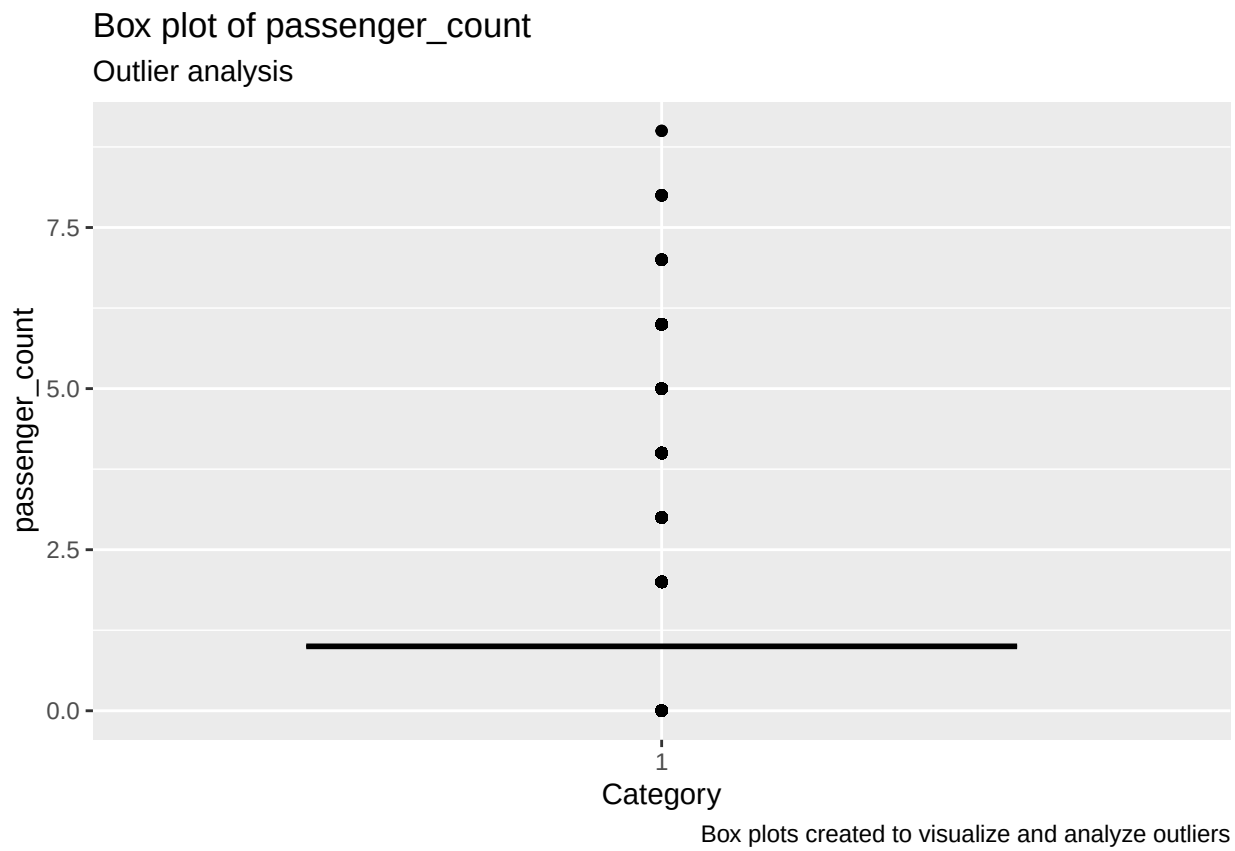
Besides the realtionship between response and other independent variables, some variables are highly correlated to each other. The correlation coeeficint between improvement_surcharge and mta_tax, and total_amount and fare_amount is 0.98 which indicates those are very strongly positively correlated. Similarly, value of RateCodeID and trip_type, and fare_amount and trip_distance is about 0.9 which also indicated strong positive correlation. The mta_tax and RateID has strong negative correaltion with -0.76, so is improvement_Surcharge and RateCodeID with the similar value. The trip_type and mta_tax, and trip_type and improvement_surcharge also has strong negative correaltion respectively with value of -0.79. The total_amount and trip_distance has also strong positive correaltion with 0.89. Lastly, tolls_amount and trip_distance, and tolls_amount and fare_amount are weakly positively correlated with values around 0.2.

Look for outliers

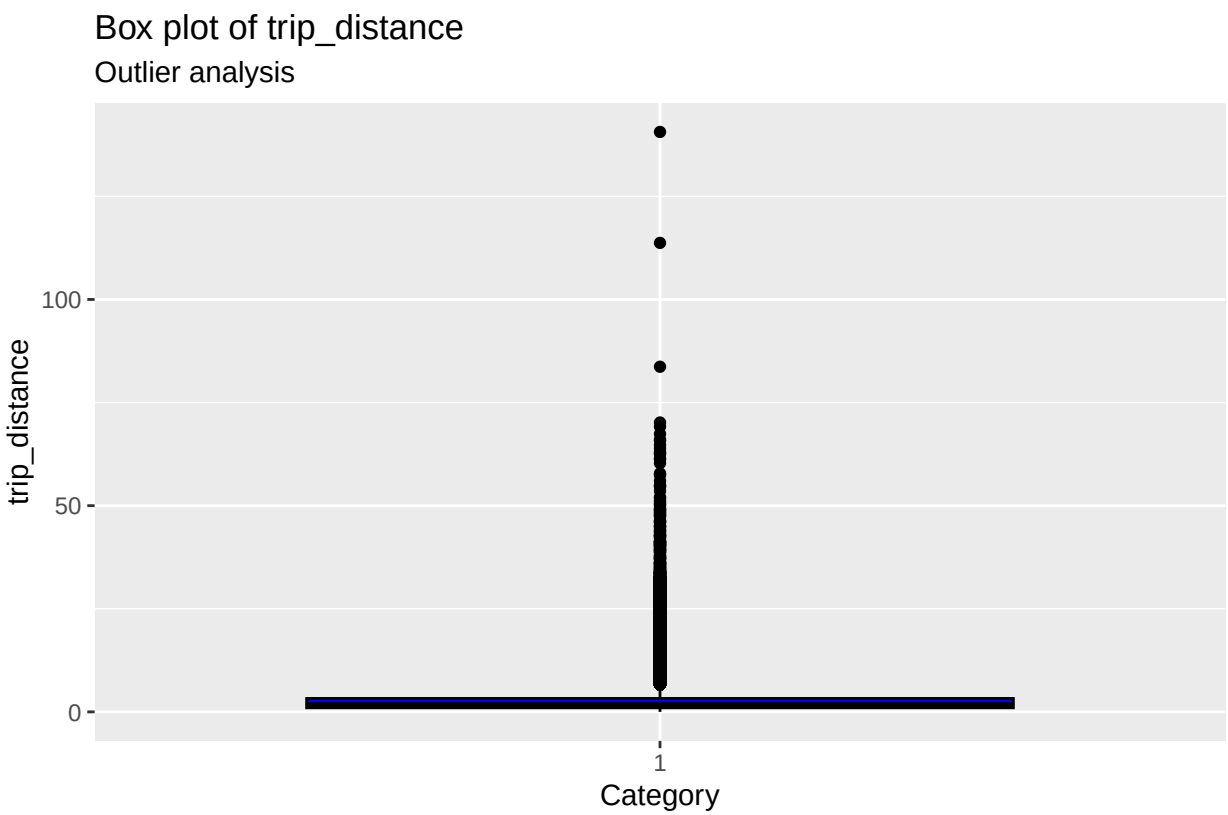
From the summary statistics above, we came to know that some of our variables have very extreme differences in minimum and maximum values, which shows an indication of presence of outliers. Hence, we visualize the outliers using box plots for all our numerical values.

```
# boxplots to look for outliers
boxplots <- lapply(1:ncol(numeric_df), function(i) {
  ggplot(data = numeric_df, aes(x = factor(1), y = numeric_df[[i]])) +
    geom_boxplot(fill = "blue", color = "black") +
    labs(
      title = paste("Box plot of", names(numeric_df)[i]),
      subtitle = "Outlier analysis",
      x = "Category",
      y = paste(names(numeric_df)[i]),
      caption = "Box plots created to visualize and analyze outliers"
    )
})

print(boxplots)
```

```
## [[1]]

## Warning: Use of `numeric_df[[i]]` is discouraged.
## i Use `.data[[i]]` instead.
```

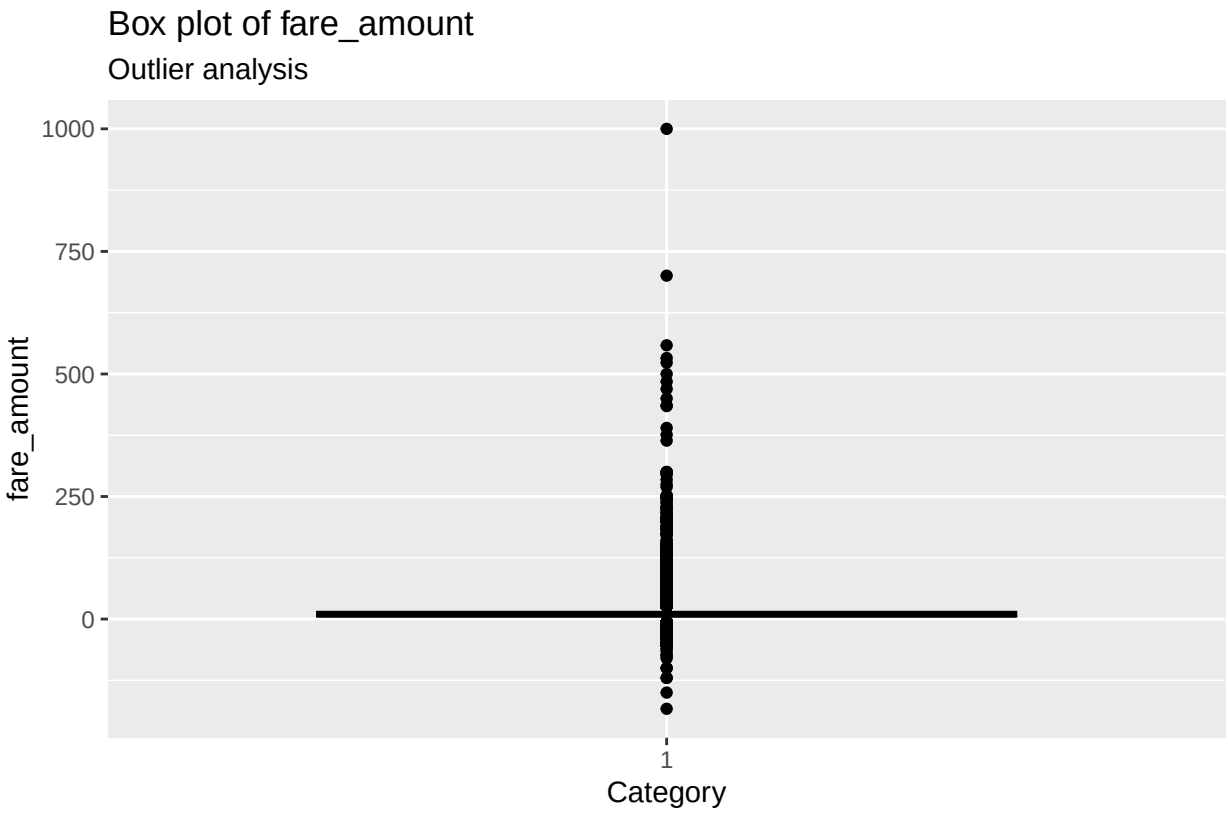


```
##
## [[2]]

## Warning: Use of `numeric_df[[i]]` is discouraged.
## i Use `.data[[i]]` instead.
```



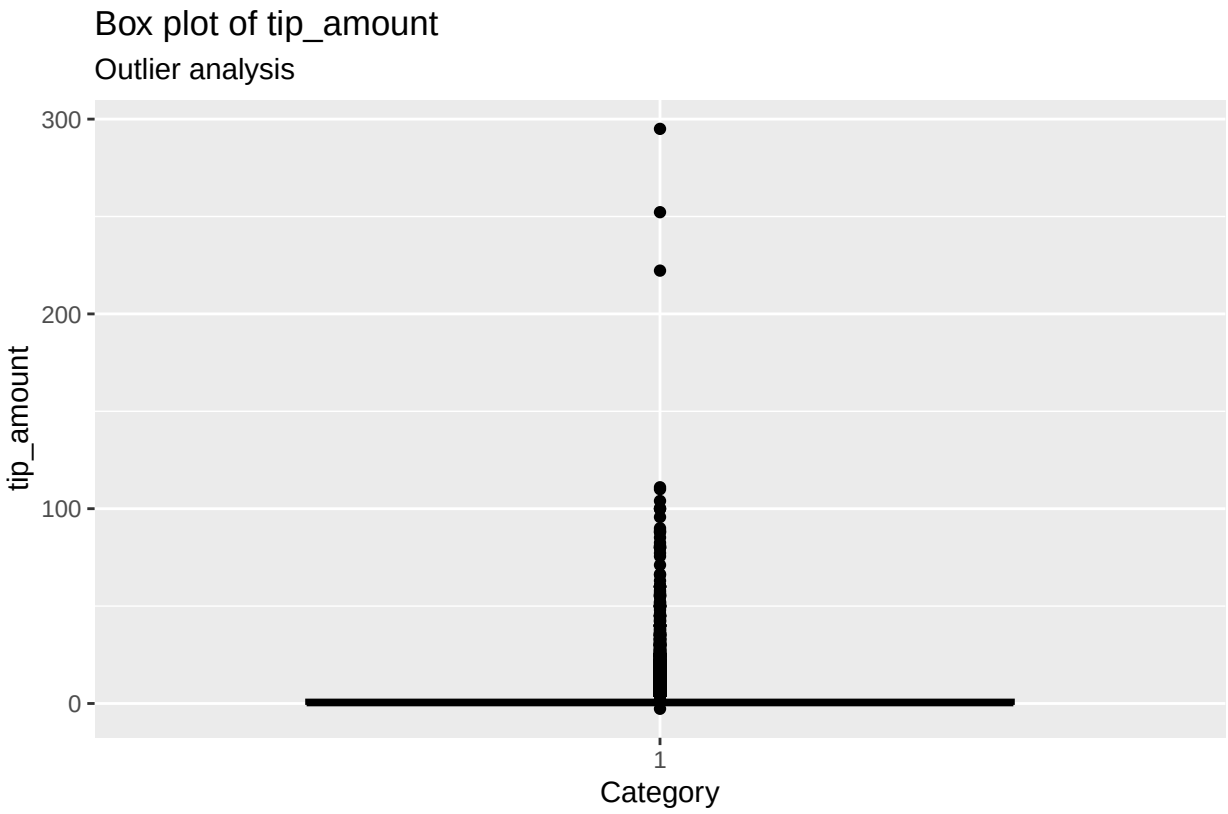
Box plots created to visualize and analyze outliers

```
##
## [[3]]

## Warning: Use of `numeric_df[[i]]` is discouraged.
## i Use `.data[[i]]` instead.

## Warning: Removed 5 rows containing non-finite values (`stat_boxplot()`).
```



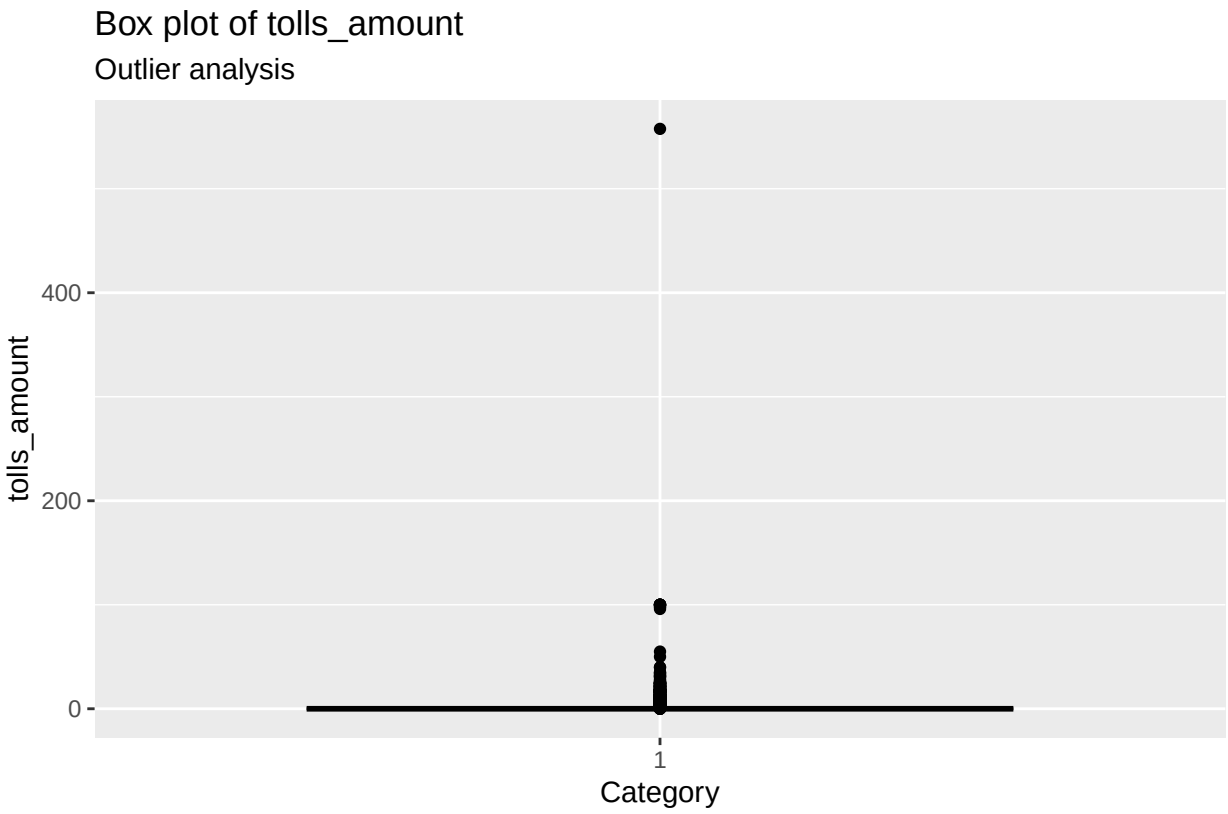
Box plots created to visualize and analyze outliers

```
##
## [[4]]

## Warning: Use of ‘numeric_df[[i]]‘ is discouraged.
## i Use ‘.data[[i]]‘ instead.
```



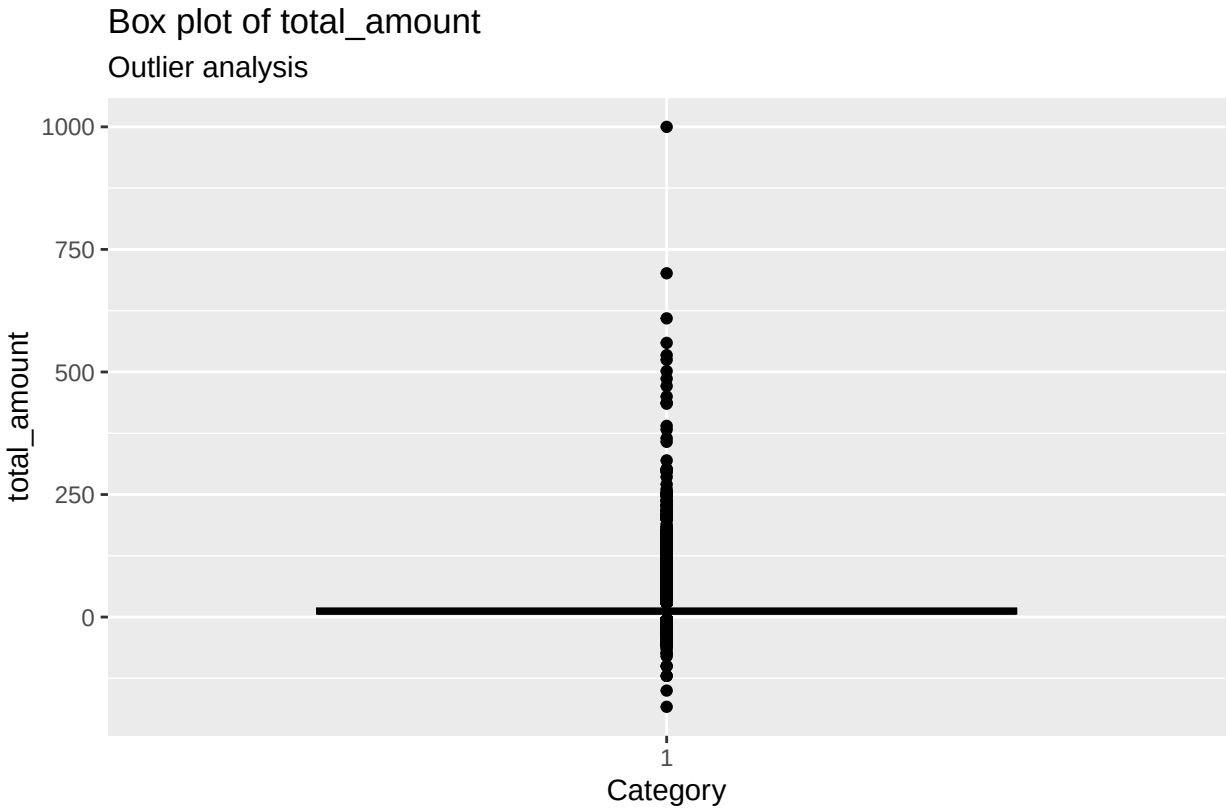
Box plots created to visualize and analyze outliers

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

## Warning: Use of ‘numeric_df[[i]]‘ is discouraged.
## i Use ‘.data[[i]]‘ instead.
```



Box plots created to visualize and analyze outliers

```
##
## [[6]]

## Warning: Use of ‘numeric_df[[i]]‘ is discouraged.
## i Use ‘.data[[i]]‘ instead.
## Removed 5 rows containing non-finite values (‘stat_boxplot()‘).
```



Box plots created to visualize and analyze outliers

From the box plot, we can see that all the plots have few or more outliers. passenger_count, trip_distance, fare_Amount, tip_Amount, tolls_amount and total_amount have highest numbers of outliers. Except for total_amount and fare_amount almost all outliers were on the higher side of the amount, but these two varaibles also have outliers in negative amount.

Feature selection

We only took data from 2018. And again did a correlation coeffient calculations and evaluate which varaibles have greater influence on tip_amount. To also look at the relation of pickup hour and tip amount, we created a new feature called "hour_of_pickup". While calculating the correlation coefficients we also included that new feature in the data frame.

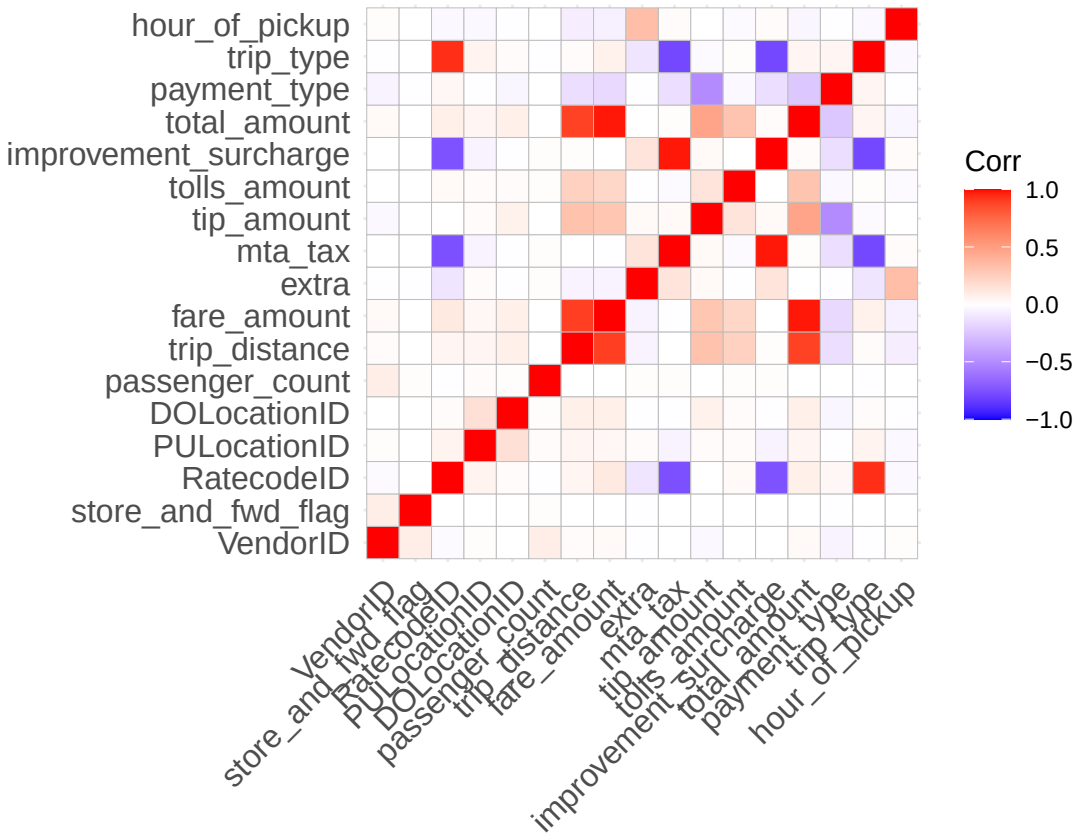
```
dist_data <- tripdata_df %>%
  mutate(
    lpep_pickup_datetime = mdy_hm(lpep_pickup_datetime),
    lpep_dropoff_datetime = mdy_hm(lpep_dropoff_datetime)) %>%
  mutate(year_of_pickup = year(lpep_pickup_datetime),
         hour_of_pickup = hour(lpep_pickup_datetime)) %>%

  filter(year_of_pickup == 2018) %>%
  select(-lpep_pickup_datetime, -lpep_dropoff_datetime, -year_of_pickup, -ehail_fee)

dist_data$store_and_fwd_flag <- as.integer(factor(dist_data$store_and_fwd_flag, levels = c("Y", "N")))

corr <- cor(dist_data, use="complete.obs")
```

```r
print(corr)
```

```
##                          VendorID store_and_fwd_flag    RatecodeID
## VendorID               1.000000000       0.0882311038 -0.0167701448
## store_and_fwd_flag     0.088231104       1.0000000000 -0.0004594771
## RatecodeID            -0.016770145      -0.0004594771  1.0000000000
## PULocationID           0.008840371       0.0001632264  0.0580988700
## DOLocationID          -0.001154950      -0.0024129940  0.0236723102
## passenger_count        0.091677658       0.0077241277 -0.0086761229
## trip_distance          0.024836876       0.0025948553  0.0490950670
## fare_amount            0.033395277       0.0027039762  0.1071929265
## extra                 -0.005762512      -0.0073278438 -0.1096612989
## mta_tax               -0.003137493      -0.0014380868 -0.7621488687
## tip_amount            -0.030879073       0.0011057015 -0.0026372416
## tolls_amount           0.002125755      -0.0003951322  0.0304114434
## improvement_surcharge -0.004290503      -0.0011121172 -0.7467412985
## total_amount           0.025631646       0.0024104382  0.0848235151
## payment_type          -0.053910121      -0.0105194445  0.0432655615
## trip_type             -0.010892626       0.0012894882  0.9361983883
## hour_of_pickup         0.012730159      -0.0020762689 -0.0304527466
##                        PULocationID DOLocationID passenger_count trip_distance
## VendorID               0.0088403707 -0.001154950    0.0916776576  0.0248368759
## store_and_fwd_flag     0.0001632264 -0.002412994    0.0077241277  0.0025948553
## RatecodeID             0.0580988700  0.023672310   -0.0086761229  0.0490950670
## PULocationID           1.0000000000  0.163458330    0.0163373672  0.0533174112
## DOLocationID           0.1634583305  1.000000000    0.0065919123  0.0817941697
## passenger_count        0.0163373672  0.006591912    1.0000000000  0.0021486077
## trip_distance          0.0533174112  0.081794170    0.0021486077  1.0000000000
## fare_amount            0.0449949633  0.075355772    0.0025227250  0.8989141015
## extra                  0.0166389821 -0.008891960    0.0099810973 -0.0479137801
## mta_tax               -0.0501889162 -0.012417696    0.0068478463  0.0007390202
## tip_amount             0.0218147848  0.074678559    0.0029647452  0.3153190138
## tolls_amount           0.0192447292  0.015315315    0.0078347260  0.2406960078
## improvement_surcharge -0.0497468067 -0.013649977    0.0063967433  0.0076570172
## total_amount           0.0466408217  0.082709425    0.0043394730  0.8911502871
## payment_type          -0.0073625635 -0.038448712    0.0002937441 -0.1417913331
## trip_type              0.0559284387  0.019581956   -0.0104070756  0.0210949313
## hour_of_pickup        -0.0337425138 -0.010241102    0.0005451293 -0.0670711497
##                        fare_amount         extra       mta_tax   tip_amount
## VendorID               0.033395277 -0.0057625122 -0.0031374927 -0.030879073
## store_and_fwd_flag     0.002703976 -0.0073278438 -0.0014380868  0.001105701
## RatecodeID             0.107192927 -0.1096612989 -0.7621488687 -0.002637242
## PULocationID           0.044994963  0.0166389821 -0.0501889162  0.021814785
## DOLocationID           0.075355772 -0.0088919601 -0.0124176960  0.074678559
## passenger_count        0.002522725  0.0099810973  0.0068478463  0.002964745
## trip_distance          0.898914101 -0.0479137801  0.0007390202  0.315319014
## fare_amount            1.000000000 -0.0497230249 -0.0130061362  0.302106052
## extra                 -0.049723025  1.0000000000  0.1430961967  0.029308593
## mta_tax               -0.013006136  0.1430961967  1.0000000000  0.027686230
## tip_amount             0.302106052  0.0293085933  0.0276862302  1.000000000
## tolls_amount           0.211476858 -0.0119517316 -0.0163946503  0.138932758
## improvement_surcharge -0.002288501  0.1425321879  0.9831816246  0.033952919
## total_amount           0.976980918 -0.0004658996  0.0105253394  0.472983807
```

```
## payment_type          -0.161176029 -0.0130818576 -0.1355634324 -0.499611768
## trip_type              0.071506302 -0.1104026416 -0.7869189587 -0.020048808
## hour_of_pickup        -0.061943304  0.3396790865  0.0246345086  0.003635207
##                        tolls_amount improvement_surcharge  total_amount
## VendorID               0.0021257545          -0.004290503  0.0256316462
## store_and_fwd_flag    -0.0003951322          -0.001112117  0.0024104382
## RatecodeID             0.0304114434          -0.746741298  0.0848235151
## PULocationID           0.0192447292          -0.049746807  0.0466408217
## DOLocationID           0.0153153153          -0.013649977  0.0827094254
## passenger_count        0.0078347260           0.006396743  0.0043394730
## trip_distance          0.2406960078           0.007657017  0.8911502871
## fare_amount            0.2114768579          -0.002288501  0.9769809177
## extra                 -0.0119517316           0.142532188 -0.0004658996
## mta_tax               -0.0163946503           0.983181625  0.0105253394
## tip_amount             0.1389327576           0.033952919  0.4729838066
## tolls_amount           1.0000000000          -0.002303774  0.3106453930
## improvement_surcharge -0.0023037737           1.000000000  0.0226032787
## total_amount           0.3106453930           0.022603279  1.0000000000
## payment_type          -0.0279835920          -0.136904153 -0.2441373866
## trip_type              0.0061678500          -0.789856834  0.0466949747
## hour_of_pickup        -0.0205737969           0.024500361 -0.0440736616
##                        payment_type    trip_type hour_of_pickup
## VendorID              -0.0539101211 -0.010892626   0.0127301587
## store_and_fwd_flag    -0.0105194445  0.001289488  -0.0020762689
## RatecodeID             0.0432655615  0.936198388  -0.0304527466
## PULocationID          -0.0073625635  0.055928439  -0.0337425138
## DOLocationID          -0.0384487122  0.019581956  -0.0102411019
## passenger_count        0.0002937441 -0.010407076   0.0005451293
## trip_distance         -0.1417913331  0.021094931  -0.0670711497
## fare_amount           -0.1611760287  0.071506302  -0.0619433044
## extra                 -0.0130818576 -0.110402642   0.3396790865
## mta_tax               -0.1355634324 -0.786918959   0.0246345086
## tip_amount            -0.4996117679 -0.020048808   0.0036352067
## tolls_amount          -0.0279835920  0.006167850  -0.0205737969
## improvement_surcharge -0.1369041529 -0.789856834   0.0245003611
## total_amount          -0.2441373866  0.046694975  -0.0440736616
## payment_type           1.0000000000  0.046884853  -0.0067987771
## trip_type              0.0468848531  1.000000000  -0.0307610658
## hour_of_pickup        -0.0067987771 -0.030761066   1.0000000000
```

```
# visualize the correlation between variables
ggcorrplot(corr)
```



By using the ggcorrplot and the correlation matrix, we can evaluate which features to select. We are considering features which have high correlation with the response variable tip_amount. We are selecting the variables which have at least > +/-0.1 correlation coefficient. So, we observed that trip_distance and fare_amount have slight positive correlation with tip_amount with the value of about 0.30. The tip_amount also has slight positive correlation with toll_amount with corrlation coefficient of 0.138. And moderate positive correlation with total_amount with 0.47 value, meanwhile tip_amount has moderate negative correlation with payment_type with the value of -0.5.

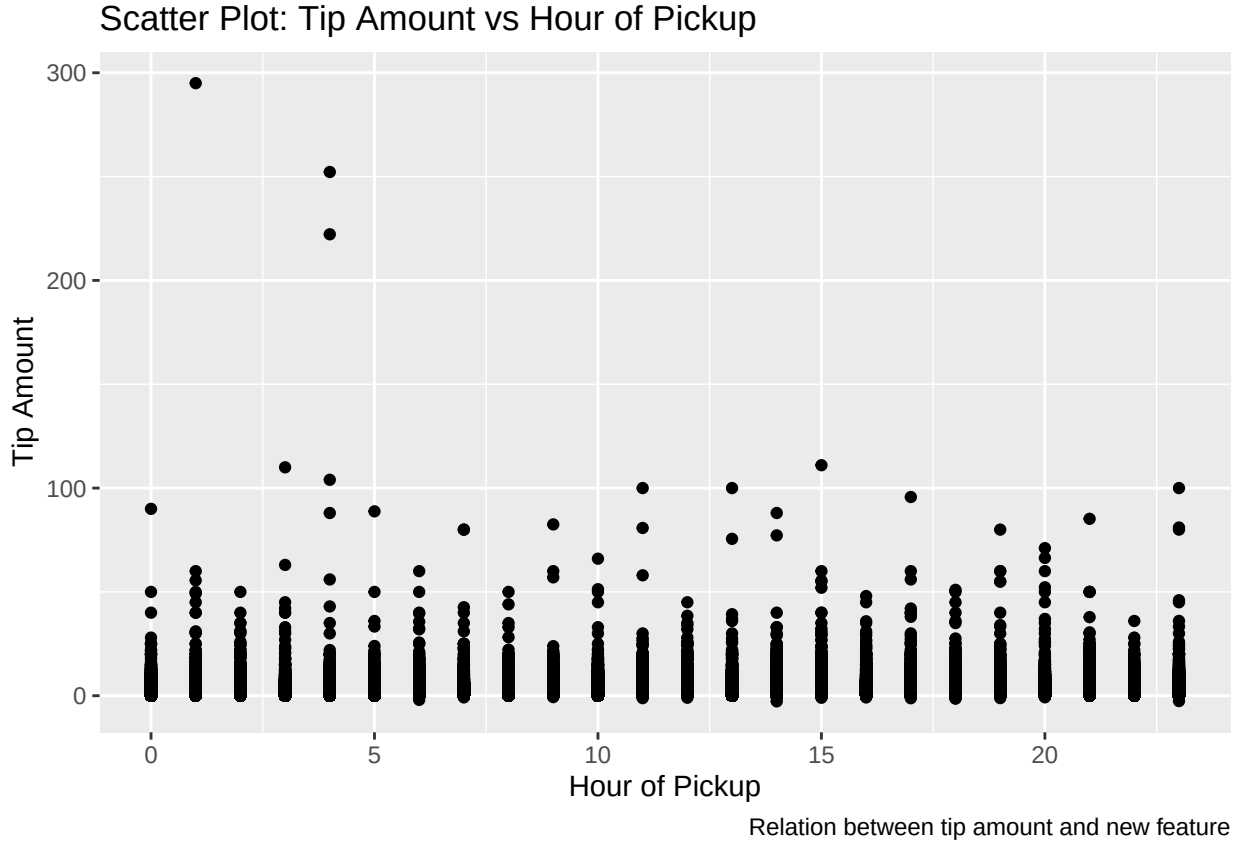
Hence, we are selecting trip_distance, fare_amount, toll_amount, total_amount and payment_type as our features.

Besides the relationship between response and other independent variables, some variables are highly correlated to each other. The correlation coefficient between improvement_surcharge and mta_tax, and total_amount and fare_amount is 0.98 which indicates those are very strongly positively correlated. Similarly, value of RateCodeID and trip_type, and fare_amount and trip_distance is about 0.9 which also indicated strong positive correlation. The mta_tax and RateID has strong negative correlation with -0.76, so is improvement_Surcharge and RateCodeID with the similar value. The trip_type and mta_tax, and trip_type and improvement_surcharge also has strong negative corrlation respectively with value of -0.79. The total_amount and trip_distance has also strong positive correlation with 0.89. Lastly, tolls_amount and trip_distance, and tolls_amount and fare_amount are weakly positively correlated with values around 0.2. This multicollinearity between different independent variables would lead to significant fluctuation on resultant model as change in one variable will also cause change in another variable. This will result in unstable model as there will be a lot of variability in it even made some small changes on the data or model. Moreover, those relation between response variable i.e. tip_amount and other independent variables is comapratively lower than the relationships between the independent variables. Hence, these variables which have strong correlation with each other should not be included as features.

Bonus (Feature engineering)

We created a new feature called "hour_of_pickup". While calculating the correlation coefficients we also included that new feature in the data frame in the above correlation matrix.

```
# create a scatter plot of tip_amount vs hour_of_pickup
ggplot(dist_data, aes(x = hour_of_pickup, y = tip_amount)) +
  geom_point() +
  labs(title = "Scatter Plot: Tip Amount vs Hour of Pickup",
       x = "Hour of Pickup",
       y = "Tip Amount",
      caption = "Relation between tip amount and new feature")
```



From the correltion matrix we found that the correlation coeffient between tip_amount and hour_of_pickup is really low with the value 0.0036352067. Also, extra and hour_of_pickup are also slightly positively correlated with correlation co-effient of 0.33, which is higher than the value it had with target variable - tip_amount. The scatter plot also shows poor correlation netween those two variables. Hence, we consider not to include that varaible as feature, as this variable has negligible influence on our target varaible.

Question 2

##CRISP-DM: Data Preparation • Prepare the data for the modeling phase and handle any issues that were identified during the exploratory data analysis.

Preprocess the data:

Based on the feature selection we are selecting, trip distance, toll amount, total amount, payment type and response variable tip amount for preprocessing the data for our model. As from our data exploration we came to know that one of our features have NAs in it. Hence we imputed the NAs with the mean of that value, this preserves the mean of the observed values.

And we again confirmed if there are any NAs left in our data and we didnot find any.

```
# select the features and response variable from our filtered data
model_df <- dist_data %>%
  select(trip_distance, tolls_amount, total_amount, payment_type, tip_amount)

# impute missing values with the mean amount
model_df <- model_df %>%
  mutate(total_amount = replace(total_amount, is.na(total_amount), mean(total_amount, na.rm = TRUE)))

# check if NA values left
sapply(model_df, function(x) sum(is.na(x)))
```

```
## trip_distance  tolls_amount  total_amount  payment_type    tip_amount
##             0             0             0             0             0
```

Based on the box-plot above for trip distance, toll amount, total amount, payment type and tip amount, we can see that there is presence of outliers. We consider z-score > 3 as outliers and we will filter out those values for the respective columns. First we create a function which calcualtes the z-score. Using that function we removed outliers from trip_distance, tolls_amount, tip_amount and total_amount one by one. We are excluding our payment_type variable as its a categorical variable.

```
# look for outliers and filter the data
# function for z_score calculation
z_score <- function(x) {
  x_mean <- mean(x,na.rm=TRUE)
  x_std <- sd(x,na.rm=TRUE)
  z <- abs(x_mean-x)/x_std
  return(z)
}


for_tip_amount <- model_df %>%
mutate(z_score_tip = z_score(tip_amount)) %>%
# filter outliers
filter(z_score_tip <= 3)

for_trip_dist <- for_tip_amount %>%
mutate(z_score_trip = z_score(trip_distance)) %>%
# filter outliers
filter(z_score_trip <= 3)

for_tolls_amount <- for_trip_dist %>%
mutate(z_score_toll = z_score(tolls_amount)) %>%
# filter outliers
filter(z_score_toll <= 3)
```

```r
filtered_df <- for_tolls_amount %>%
mutate(z_score_total = z_score(total_amount)) %>%
# filter outliers
filter(z_score_total <= 3)

# select only the required variables and remove the z-scores columns
filtered_df <- filtered_df %>%
select(trip_distance, tolls_amount, total_amount, payment_type, tip_amount)

dim(filtered_df)
```

```
## [1] 995322      5
```

```r
head(filtered_df)
```

```
## # A tibble: 6 x 5
##   trip_distance tolls_amount total_amount payment_type tip_amount
##           <dbl>        <dbl>        <dbl>        <dbl>      <dbl>
## 1          0.7             0          7.3            2          0
## 2          3.5             0         15.8            2          0
## 3          2.14            0         11.3            2          0
## 4          0.03            0         -4.3            3          0
## 5          0.03            0          4.3            2          0
## 6          5.63            0         22.3            2          0
```

```r
dim(model_df)
```

```
## [1] 1048514       5
```

```r
# number of outliers
total_outliers <- nrow(model_df) - nrow(filtered_df)
total_outliers
```

```
## [1] 53192
```

```r
str(filtered_df)
```

```
## tibble [995,322 x 5] (S3: tbl_df/tbl/data.frame)
##  $ trip_distance: num [1:995322] 0.7 3.5 2.14 0.03 0.03 5.63 1.71 3.45 1.61 1.87 ...
##  $ tolls_amount : num [1:995322] 0 0 0 0 0 0 0 0 0 0 ...
##  $ total_amount : num [1:995322] 7.3 15.8 11.3 -4.3 4.3 ...
##  $ payment_type : num [1:995322] 2 2 2 3 2 2 2 1 2 2 ...
##  $ tip_amount   : num [1:995322] 0 0 0 0 0 0 0 3.16 0 0 ...
```

The data which had 1048514 rows previously was decreased to 995322 after removal of outliers. There were 53192 outliers in our data frame for modeling.

Normalize the data:

Perform either max-min normalization or z-score standardization on the continuous variables/features.

```r
summary(filtered_df)
```

```
##  trip_distance     tolls_amount       total_amount    payment_type
##  Min.   : 0.000   Min.   :0.00e+00   Min.   :-9.30   Min.   :1.000
##  1st Qu.: 0.960   1st Qu.:0.00e+00   1st Qu.: 7.80   1st Qu.:1.000
##  Median : 1.600   Median :0.00e+00   Median :10.56   Median :1.000
##  Mean   : 2.238   Mean   :4.32e-05   Mean   :12.40   Mean   :1.483
##  3rd Qu.: 2.940   3rd Qu.:0.00e+00   3rd Qu.:15.36   3rd Qu.:2.000
##  Max.   :10.460   Max.   :1.70e+00   Max.   :35.10   Max.   :5.000
##    tip_amount
##  Min.   :-2.0000
##  1st Qu.: 0.0000
##  Median : 0.0000
##  Mean   : 0.8872
##  3rd Qu.: 1.6800
##  Max.   : 6.6600
```

```r
# select columns to normalize
# not selecting payment_type as its a categorical variable
# only selecting trip_distance, tolls_amount and total_amount
selected_columns<-c(1:3)
# used preProcess function from caret package
# max-min normalization
preproc1<- preProcess(filtered_df[,selected_columns], method=c("range"))

# normalization transformation
normalized_df <- predict(preproc1, filtered_df[,selected_columns])
tip_amount <- filtered_df$tip_amount
payment_type <- filtered_df$payment_type
normalized_df <- cbind(normalized_df, tip_amount, payment_type)
str(normalized_df)
```

```
## 'data.frame':    995322 obs. of  5 variables:
##  $ trip_distance: num  0.06692 0.33461 0.20459 0.00287 0.00287 ...
##  $ tolls_amount : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ total_amount : num  0.374 0.565 0.464 0.113 0.306 ...
##  $ tip_amount   : num  0 0 0 0 0 0 0 3.16 0 0 ...
##  $ payment_type : num  2 2 2 3 2 2 2 1 2 2 ...
```

We used functions from caret package to normalize the data and used min-max normalization. We only selected trip_distance, tolls_amount and total_amount, and left payment_type as its a categorical varaible. We then applied normalization tansforamtion to the slected columns using the predict() function. Created a new dataframe and added the tip_amount and payment_type to the new data frame.

Encode the data:

Convert to dummy code for our categorical variable - payment_type

```r
payment_type <- as.data.frame(dummy.code(normalized_df$payment_type))
normalized_df <- cbind(normalized_df, payment_type)
normalized_df <- normalized_df %>% select(-payment_type)
```

```r
col_names <- c("credit_card", "cash", "no_charge", "dispute", "unknown")
colnames(normalized_df)[5:9] <- col_names
head(normalized_df)
```

```
##   trip_distance tolls_amount total_amount tip_amount credit_card cash no_charge
## 1   0.066921606            0    0.3738739          0           0    1         0
## 2   0.334608031            0    0.5653153          0           0    1         0
## 3   0.204588910            0    0.4639640          0           0    1         0
## 4   0.002868069            0    0.1126126          0           0    0         1
## 5   0.002868069            0    0.3063063          0           0    1         0
## 6   0.538240918            0    0.7117117          0           0    1         0
##   dispute unknown
## 1       0       0
## 2       0       0
## 3       0       0
## 4       0       0
## 5       0       0
## 6       0       0
```

```r
str(normalized_df)
```

```
## 'data.frame':    995322 obs. of  9 variables:
##  $ trip_distance: num  0.06692 0.33461 0.20459 0.00287 0.00287 ...
##  $ tolls_amount : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ total_amount : num  0.374 0.565 0.464 0.113 0.306 ...
##  $ tip_amount   : num  0 0 0 0 0 0 0 3.16 0 0 ...
##  $ credit_card  : num  0 0 0 0 0 0 0 1 0 0 ...
##  $ cash         : num  1 1 1 0 1 1 1 0 1 1 ...
##  $ no_charge    : num  0 0 0 1 0 0 0 0 0 0 ...
##  $ dispute      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ unknown      : num  0 0 0 0 0 0 0 0 0 0 ...
```

Prepare the data for modeling

We used createDataPArtition() function from caret package to split our data into taining and testing set. This ensures maintinance of the distribution of target variable - tip_amount values in both traing and testing sets.

```r
set.seed(42)

# 'Outcome' is the response variable
indx <- createDataPartition(normalized_df$tip_amount, p = 0.8, list = FALSE)
trip_train <- normalized_df[indx, ]  # train data
trip_test <- normalized_df[-indx, ]  # test data
head(trip_train)
```

```
##   trip_distance tolls_amount total_amount tip_amount credit_card cash no_charge
## 2   0.334608031            0    0.5653153       0.00           0    1         0
## 4   0.002868069            0    0.1126126       0.00           0    0         1
## 5   0.002868069            0    0.3063063       0.00           0    1         0
## 6   0.538240918            0    0.7117117       0.00           0    1         0
```

```
## 8   0.329827916            0    0.6364865       3.16           1    0         0
## 9   0.153919694            0    0.4639640       0.00           0    1         0
##   dispute unknown
## 2       0       0
## 4       0       0
## 5       0       0
## 6       0       0
## 8       0       0
## 9       0       0
```

```
head(trip_test)
```

```
##    trip_distance tolls_amount total_amount tip_amount credit_card cash
## 1     0.06692161            0    0.3738739       0.00           0    1
## 3     0.20458891            0    0.4639640       0.00           0    1
## 7     0.16347992            0    0.4301802       0.00           0    1
## 15    0.57456979            0    0.7567568       4.00           1    0
## 17    0.41108987            0    0.6632883       3.35           1    0
## 22    0.17399618            0    0.4414414       0.00           1    0
##    no_charge dispute unknown
## 1          0       0       0
## 3          0       0       0
## 7          0       0       0
## 15         0       0       0
## 17         0       0       0
## 22         0       0       0
```

For our model, we chose to split our training and testing data into 80-20 ratio. Though this is a commmon choice, it achieves a balance between having a enough amount of data for training the model and another set for assessing how well it performs.

Question 3

CRISP-DM: Modeling

• In this step, develop the k-nn regression model. Create a function with the following name and arguments: knn.predict(data_train, data_test, k); • data_train represents the observations in the training set, • data_test represents the observations from the test set, and • k is the selected value of k (i.e. the number of neighbors).

```
knn.predict <- function(data_train, data_test, k) {

    # Extract predictors and response from training data
  predictors_train <- data_train %>% select(-tip_amount)
  response_train <- data_train$tip_amount

  # Extract predictors and true response from test data
  predictors_test <- data_test %>% select(-tip_amount)
  y_truth <- data_test$tip_amount

  # Train the KNN regression model
```

```
  knn_model <- knn.reg(train = predictors_train,
                       test = predictors_test,
                       y = response_train, k = k)

  # Extract predicted values
  y_pred <- knn_model$pred

  # Calculate Mean Squared Error (MSE)
  mse <- mean((y_truth - y_pred)^2)

  # Returning the MSE
  return(mse)
}
```

Question 4

CRISP-DM: Evaluation

• Determine the best value of k and visualize the MSE. This step requires selecting different values of k and evaluating which produced the lowest MSE. At a minimum, ensure that you perform the following: • Provide at least 20 different values of k to the knn.predict() function (along with the training set and the test set). • Create a line chart and plot each value of k on the x-axis and the corresponding MSE on the y-axis. Explain the chart and determine which value of k is more suitable and why.

```
# create 20 different values of k from 4 to 100 without duplicates
sample <- c(6,10,14,18,22,25,36,38,42,48,54,63,67,70,72,79,81,86,93,99)

# create an empty data frame to store k and MSE values
df_knn <- data.frame(matrix(ncol = 2, nrow = 20))
y <- c("k", "MSE")
colnames(df_knn) <- y

# using a loop to call knn.predict() for 20 values of k
for (i in 1:length(sample)){
    df_knn$k[i] <- sample[i]
    df_knn$MSE[i] <- knn.predict(data_train = trip_train, data_test = trip_test, k = sample[i])
}
df_knn
```

```
##     k       MSE
## 1   6 0.3534419
## 2  10 0.3616238
## 3  14 0.3732716
## 4  18 0.3847780
## 5  22 0.3953950
## 6  25 0.4032539
## 7  36 0.4273090
## 8  38 0.4316450
## 9  42 0.4390289
## 10 48 0.4491055
## 11 54 0.4581599
```

```
## 12 63 0.4704343
## 13 67 0.4754743
## 14 70 0.4788641
## 15 72 0.4809616
## 16 79 0.4881959
## 17 81 0.4901486
## 18 86 0.4941027
## 19 93 0.4994746
## 20 99 0.5037937
```

```r
# identifying the minimum MSE and associated k value from the 20 k values iterated through knn.predict
min_knn_MSE <- min(df_knn$MSE)
min_knn_k <- df_knn[df_knn$MSE == min_knn_MSE, "k"]
print(paste0("The minimmum MSE is ", min_knn_MSE, " with an associated k value of ", min_knn_k,"."))
```

```
## [1] "The minimmum MSE is 0.353441872728257 with an associated k value of 6."
```

```r
# creating a line chart to plot k and MSE values
ggplot(data = df_knn, aes(x = k, y = MSE)) +
  geom_line() +
  geom_point() +
  labs(title = "Mean Squared Error (MSE) versus k for Green Taxi Tip Amount",
       caption = "Determining the most suitable value of k by evaluting which k produces the lowest MSE
  theme(plot.title = element_text(hjust = 0.5))
```

Determining the most suitable value of k by evaluting which k produces the lowest MSE.

The above line chart shows the MSE value for each k value iterated through the knn.predict function. The 20 k values were randomly selected from a range of values between 4 and 100. The line chart shows that as the k value increases, the MSE value increases too. The line graph therefore shows that the minimum MSE value of 0.331 is associated with a k value of 6.

In assessing this knn model and the accuracy of its predictions, it's important to consider the guidelines for selecting a k value. The guidlines state that it is wise to start with a k value that is equivalent to the square root of the number of cases in the training data set. The number of cases in the training data set is 789,156 (0.8 multiplied by 986,445 total cases). The square root of 789,156 is approximately 888. As shown below, the knn.predict function was run for a k value of 888 and resulted in an MSE value of 0.600. This MSE value is greater than the 0.331 value associated with a k value of 6. Therefore, I am more likely to advocate for utilizing a k value of 6 for accurately predicting the tip amount of future trips over using a k value of 888. The smaller MSE value for a k value of 6 indicates that the data fits the k = 6 model better than k = 888 model.

```
knn_k <- knn.predict(data_train = trip_train, data_test = trip_test, k = 888)
print(paste0("The MSE value for a k value of 888 is ", knn_k,"."))
```

```
## [1] "The MSE value for a k value of 888 is 0.591421220528345."
```

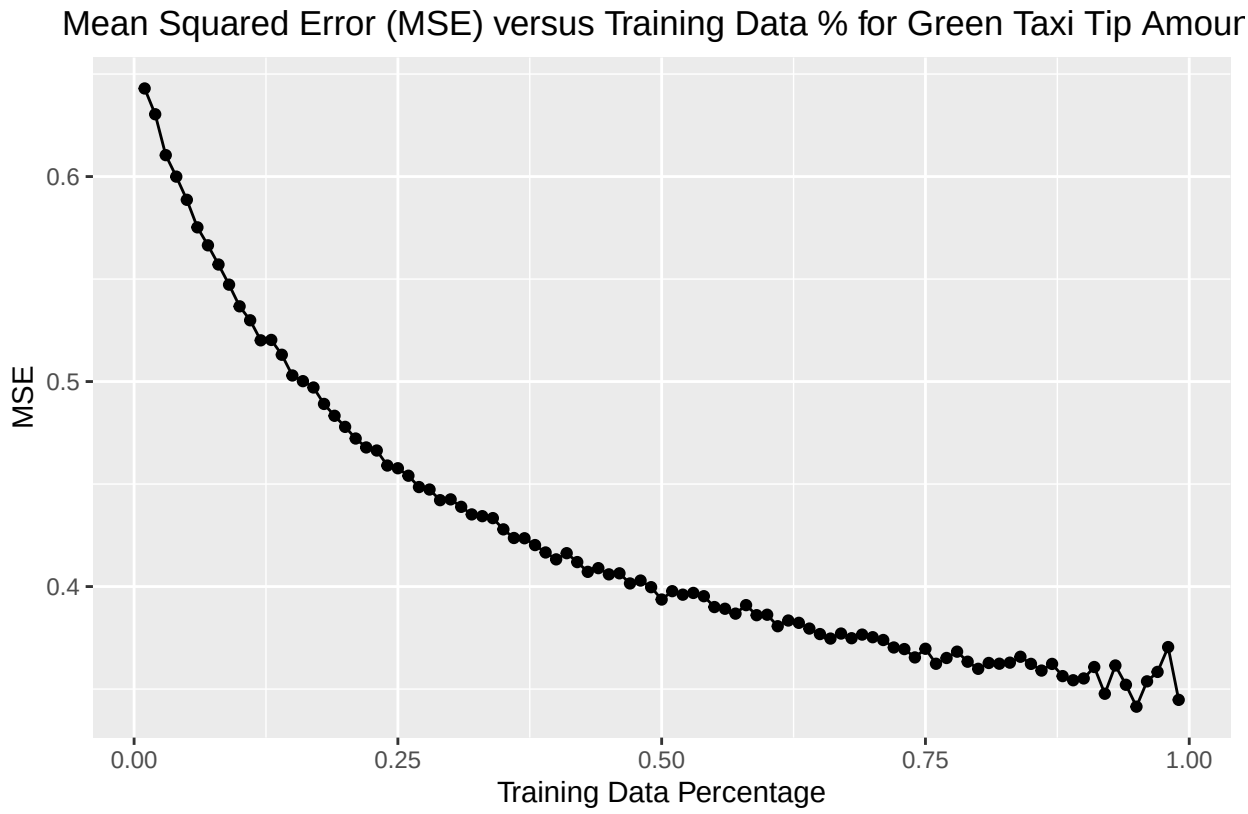
Question 5

Evaluate the effect of the percentage split for the training and test sets and determine if a different split ratio improves your model's ability to make better predictions.

```
# creating vector of all training and testing data splits from 0.01 to 0.99 at an increment of 0.01
splits <- seq(0.01,0.99,0.01)

# create an empty data frame to store Data_Training_Percentage and MSE values
df_split <- data.frame(matrix(ncol = 2, nrow = 99))
z <- c("Data_Training_Percentage", "MSE")
colnames(df_split) <- z

# make createDataPartition of training and test data reproducible by setting a seed
set.seed(18)

# using a loop to call knn.predict() for each training and testing data percentage split with a constan
for (i in 1:length(splits)){
    df_split$Data_Training_Percentage[i] <- splits[i]
    indexing <- createDataPartition(normalized_df$tip_amount, p = splits[i], list = FALSE)
    trip_train <- normalized_df[indexing, ]
    trip_test <- normalized_df[-indexing, ]
    df_split$MSE[i] <- knn.predict(data_train = trip_train, data_test = trip_test, k = 10)
}

# identifying the minimum MSE and associated data training and testing percentages from the data traini
min_split_MSE <- min(df_split$MSE)
min_split_train <- df_split[df_split$MSE == min_split_MSE, "Data_Training_Percentage"]
print(paste0("The minimmum MSE is ", min_split_MSE, " with an associated training data percentage of ",
```

```
## [1] "The minimmum MSE is 0.341410254455943 with an associated training data percentage of 0.95 and a
```

```r
# creating a line chart to plot Data Training Percentage (Data Testing Percentage = 1 - Data Training P
ggplot(data = df_split, aes(x = Data_Training_Percentage, y = MSE)) +
  geom_line() +
  geom_point() +
  labs(title = "Mean Squared Error (MSE) versus Training Data % for Green Taxi Tip Amount",
         x = "Training Data Percentage",
         y = "MSE",
       caption = "Optimizing the k-nn model by evaluating the effect of the training and testing data p
  theme(plot.title = element_text(hjust = 0.5))
```



k−nn model by evaluating the effect of the training and testing data percentage split on the MSE with a fixed k value.

Based on the above results, it appears that a greater training data percentage leads to a lower MSE. Therefore, as the training data percentage increases, the data better fits the knn-model. Logically, this makes sense for any machine learning algorithm because as the algorithm is supplied with more training data to teach the algorithm, the testing data is better prepared to validate the algorithm's training progress and the model is optimized for improved results. According to the results, a training data percentage of 0.95 and a testing data percentage of 0.05 lead to the minimum MSE value of 0.341.