DA5020.Practicum3.Hsiao-Yu.Peng

Hsiao-Yu Peng

2023-12-05

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.

```
# load NYC Green Taxi Trip Record
# tripdata_df <- read_csv("~/Desktop/2023Fall_Syllabus/DA5020/week3/2018_Green_Taxi_Trip_Data.csv")
tripdata_df<-read.delim2("~/Desktop/2023Fall_Syllabus/DA5020/module3/2018_Green_Taxi_Trip_Data.csv", hea

# dataset overview
# glimpse(tripdata_df)
dim(tripdata_df)
```

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

```
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
##  Length:1048575     Length:1048575     Length:1048575     Length:1048575
##  Class :character   Class :character   Class :character   Class :character
```

```
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##   tip_amount        tolls_amount       ehail_fee      improvement_surcharge
##  Length:1048575     Length:1048575     Mode:logical   Length:1048575
##  Class :character   Class :character   NA's:1048575   Class :character
##  Mode  :character   Mode  :character                  Mode  :character
##
##
##
##
##  total_amount        payment_type     trip_type
##  Length:1048575     Min.   :1.000   Min.   :1.000
##  Class :character   1st Qu.:1.000   1st Qu.:1.000
##  Mode  :character   Median :1.000   Median :1.000
##                     Mean   :1.471   Mean   :1.018
##                     3rd Qu.:2.000   3rd Qu.:1.000
##                     Max.   :5.000   Max.   :2.000
##                                     NA's   :3
```

This dataset has dimension 1,048,575 x 19. Most variables do not have their corresponding data type, we will convert them as follows:

(a) Tips, amount, distance-related variables should be numerical, such as tip_amount, total_amount..., but they are character type. We'll convert them into numerical type.
(b) Categorical variables in the data set are shown as numeric type, we'll convert them to factor type.
(c) Date-related variables including "lpep_pickup_datetime" and "lpep_dropoff_datetime" are character types. We'll convert these character data types into datetime type.

The data set includes logical data for "ehai_fee," although "ehai_fee" has more than a 90% missing value. "total_amount" and "trip_type" have fewer missing values, and we will address these missing values later.

Tranform data type

```
## Convert numeric features from character to numerical
# Define numeric features
numeric_var<- c("trip_distance", "fare_amount", "extra", "mta_tax", "tip_amount", "tolls_amount", "impr

# Convert characters to numeric
tripdata_df[numeric_var] <- lapply(tripdata_df[numeric_var], as.numeric)
```

```
## Warning in lapply(tripdata_df[numeric_var], as.numeric): NAs introduced by
## coercion

## Warning in lapply(tripdata_df[numeric_var], as.numeric): NAs introduced by
## coercion
```

```r
## Convert categorical features to factor type
tripdata_df <- tripdata_df %>%
  mutate(store_and_fwd_flag = if_else(store_and_fwd_flag == "N", 0, 1))

# Define categorical variables
cat_var <- c("VendorID", "RatecodeID", "store_and_fwd_flag", "payment_type", "trip_type")

# Convert categorical variables to factors
tripdata_df[cat_var] <- lapply(tripdata_df[cat_var], as.factor)


# Convert pickup_date, dropoff_date to date.time type
tripdata_df <- tripdata_df %>%
  mutate(pickup = as.Date(lpep_pickup_datetime, format = "%m/%d/%Y"))

tripdata_df <- tripdata_df %>%
  mutate(dropoff = as.Date(lpep_dropoff_datetime, format = "%m/%d/%Y"))


summary(tripdata_df)
```

```
##  VendorID   lpep_pickup_datetime lpep_dropoff_datetime store_and_fwd_flag
##  1:181509   Length:1048575       Length:1048575        0:1046867
##  2:867066   Class :character     Class :character      1:   1708
##             Mode  :character     Mode  :character
##
##
##
##
##  RatecodeID    PULocationID  DOLocationID   passenger_count trip_distance
##  1 :1028080   Min.   :  1   Min.   :  1.0   Min.   :0.000   Min.   :  0.000
##  2 :   1715   1st Qu.: 49   1st Qu.: 61.0   1st Qu.:1.000   1st Qu.:  0.990
##  3 :    407   Median : 82   Median :129.0   Median :1.000   Median :  1.700
##  4 :    537   Mean   :110   Mean   :128.5   Mean   :1.359   Mean   :  2.662
##  5 :  17816   3rd Qu.:166   3rd Qu.:191.0   3rd Qu.:1.000   3rd Qu.:  3.260
##  6 :     17   Max.   :265   Max.   :265.0   Max.   :9.000   Max.   :140.620
##  99:      3
##   fare_amount          extra            mta_tax          tip_amount
##  Min.   :-183.00   Min.   :-4.5000   Min.   :-0.5000   Min.   : -2.720
##  1st Qu.:   6.00   1st Qu.: 0.0000   1st Qu.: 0.5000   1st Qu.:  0.000
##  Median :   9.00   Median : 0.0000   Median : 0.5000   Median :  0.000
##  Mean   :  11.72   Mean   : 0.3445   Mean   : 0.4882   Mean   :  1.037
##  3rd Qu.:  14.00   3rd Qu.: 0.5000   3rd Qu.: 0.5000   3rd Qu.:  1.760
##  Max.   : 999.99   Max.   : 4.5000   Max.   : 0.5000   Max.   :295.000
##  NA's   :5
##   tolls_amount      ehail_fee      improvement_surcharge  total_amount
##  Min.   :  0.0000   Mode:logical   Min.   :-0.3000       Min.   :-183.00
##  1st Qu.:  0.0000   NA's:1048575   1st Qu.: 0.3000       1st Qu.:   7.80
##  Median :  0.0000                  Median : 0.3000       Median :  10.80
##  Mean   :  0.0858                  Mean   : 0.2932       Mean   :  13.99
##  3rd Qu.:  0.0000                  3rd Qu.: 0.3000       3rd Qu.:  16.80
##  Max.   :557.5500                  Max.   : 0.3000       Max.   : 999.99
##                                                          NA's   :5
```

```
##  payment_type trip_type          pickup              dropoff
##  1:564347     1   :1030020   Min.   :2009-01-01   Min.   :2009-01-01
##  2:476995     2   :  18552   1st Qu.:2018-01-12   1st Qu.:2018-01-12
##  3:  5032     NA's:      3   Median :2018-01-21   Median :2018-01-21
##  4:  2133                    Mean   :2018-01-21   Mean   :2018-01-21
##  5:    68                    3rd Qu.:2018-01-31   3rd Qu.:2018-01-31
##                              Max.   :2018-04-05   Max.   :2018-04-05
##
```

```
# glimpse(tripdata_df)
```

After converting some variables to their data type, we found some issues in this data set.

(a) Since the data set contains numerical variables related to taxes, tips, charges, etc., the numbers should be positive or zero. However, some variables have minimum values that are negative, which should be unreasonable. These variables include "fare_amount," "extra," "mta_tax," "tip_amount," "improvement_surcharge," and "total_amount."

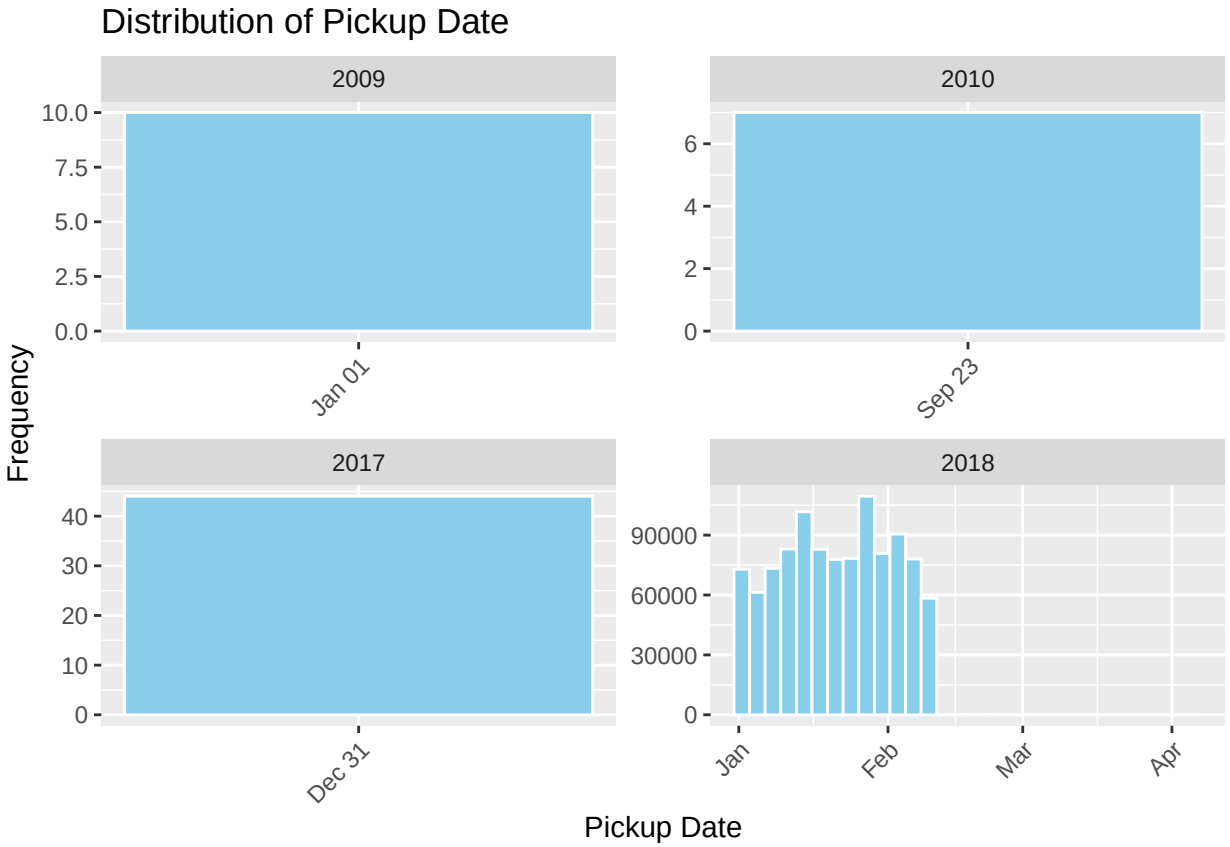
(b) RateCodeID is from 1 to 6, but it has maximum value 99. We should check what numbers are unique in this variable.

(c) This is 2018 dataset. But datetime variables, "lpep_pickup_datetime" and " lpep_dropoff_datetime", have information that was not in 2018, these are inconsistent data.

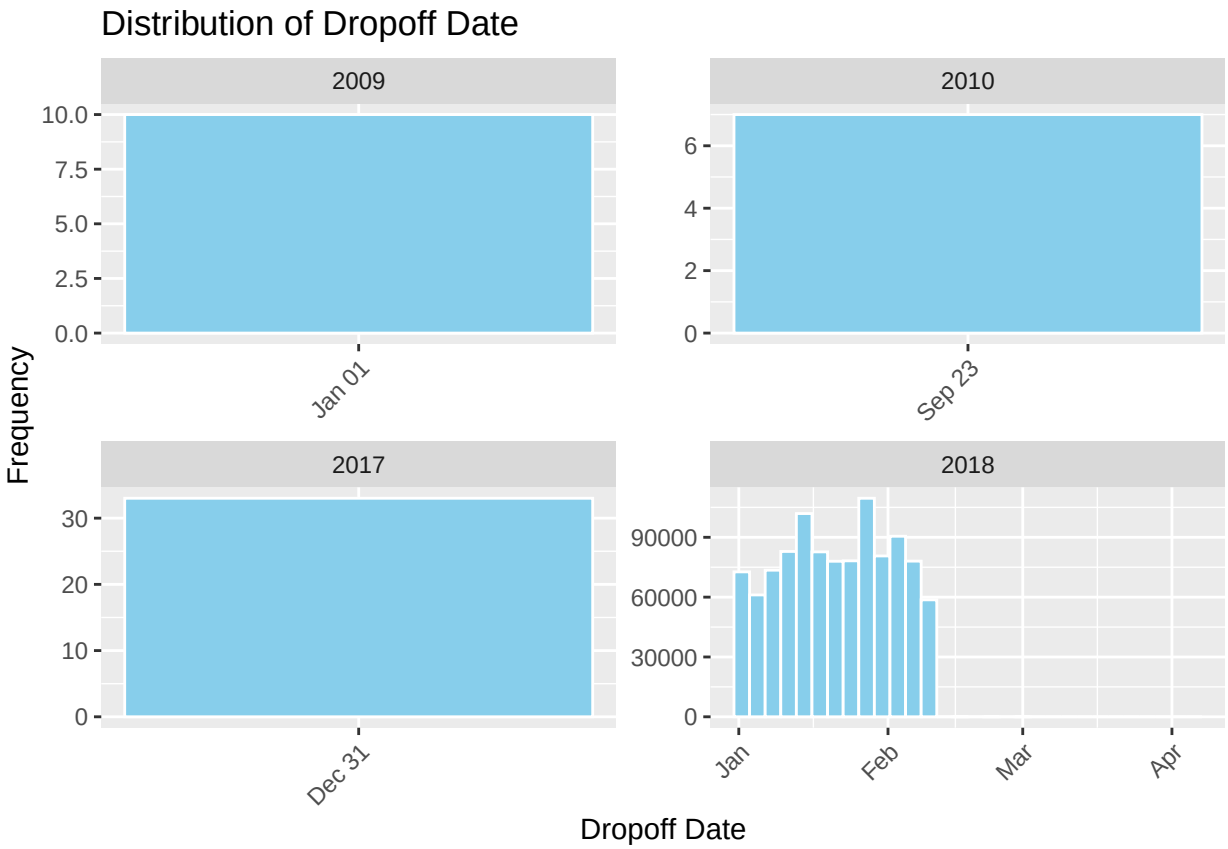
1.1 Date-Time Variable

1.1.1 Display date.time distribution

```
# Display pickup Date
ggplot(tripdata_df, aes(x = pickup)) +
  geom_histogram(bins = 30, fill = "skyblue", color = "white") +
  labs(x = "Pickup Date", y = "Frequency", title = "Distribution of Pickup Date") +
  facet_wrap(~year(pickup), scales = "free") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
# Display dropoff Date
ggplot(tripdata_df, aes(x = dropoff)) +
  geom_histogram(bins = 30, fill = "skyblue", color = "white") +
  labs(x = "Dropoff Date", y = "Frequency", title = "Distribution of Dropoff Date") +
  facet_wrap(~year(dropoff), scales = "free") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



We found it has other year information, we will remove the information except those in 2018.

1.1.2 Transform & clean date-time variable

We filter only 2018 data.

```
# Convert pickup date to weekday
tripdata_df$pickup_weekday <- weekdays(tripdata_df$pickup)

# Convert dropoff date to weekday
tripdata_df$dropoff_weekday <- weekdays(tripdata_df$dropoff)

# Filter data only in 2018
tripdata_2018 <- tripdata_df %>% filter(year(pickup) == 2018 & year(dropoff) == 2018)
```

1.1.3 Correlation analysis between date and tip_amount

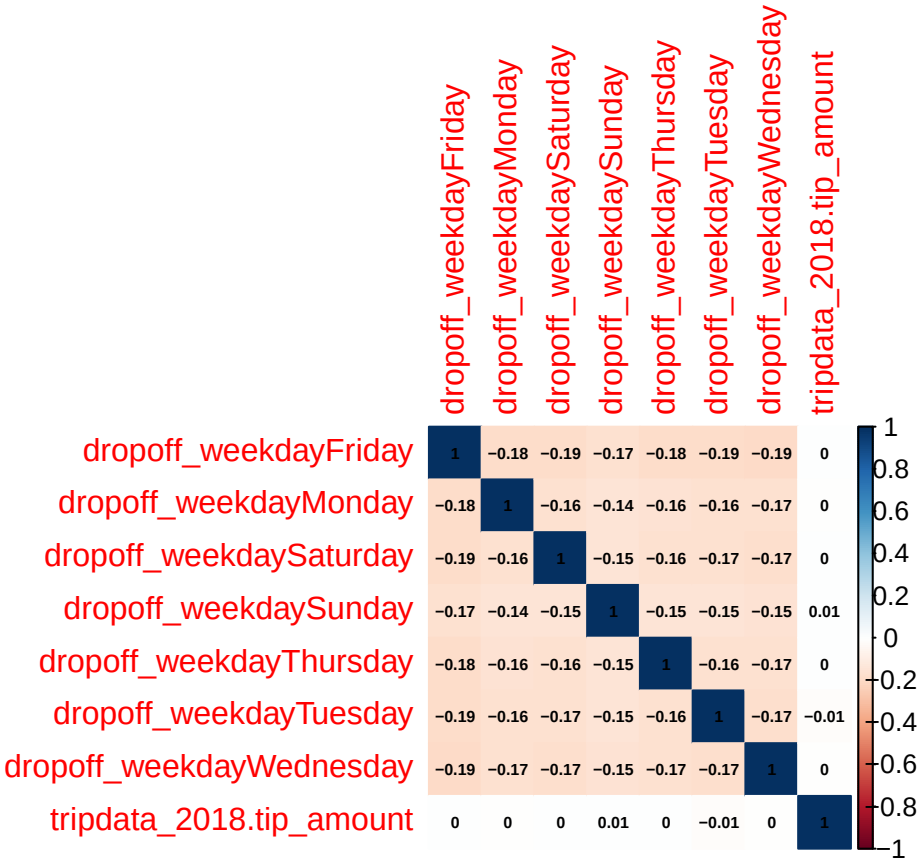
```
# pickup day tips
tripdata_2018 %>%
  group_by(pickup) %>%
  summarize(tip_total = sum(tip_amount)) %>%
  arrange(desc(tip_total))
```

```
## # A tibble: 66 x 2
##    pickup     tip_total
##    <date>         <dbl>
##  1 2018-02-02    34427.
##  2 2018-01-27    33624.
##  3 2018-01-26    33616.
##  4 2018-01-20    33011.
##  5 2018-02-03    32981.
##  6 2018-01-12    32784.
##  7 2018-01-13    32018.
##  8 2018-01-19    31553.
##  9 2018-02-08    30837.
## 10 2018-02-09    30613.
## # i 56 more rows
```

```
# Weekday tips
tripdata_2018 %>%
  group_by(pickup_weekday) %>%
  summarize(tip_total = sum(tip_amount)) %>%
  arrange(desc(tip_total))
```

```
## # A tibble: 7 x 2
##   pickup_weekday tip_total
##   <chr>              <dbl>
## 1 Friday           190143.
## 2 Wednesday        161629.
## 3 Saturday         159261.
## 4 Thursday         153202.
## 5 Tuesday          148256.
## 6 Monday           145419.
## 7 Sunday           129898.
```

```
dummy_weekday <- dummyVars("~dropoff_weekday", data= tripdata_2018) %>%
  predict(tripdata_2018)


combined_weekday_data <- data.frame(dummy_weekday, tripdata_2018$tip_amount)

# Get correlation matrix
cor_matrix <- cor(combined_weekday_data , use = "pairwise.complete.obs")

# Visualize correlation matrix
corrplot(cor_matrix, method = "color", addCoef.col = "black", number.cex = 0.5)
```

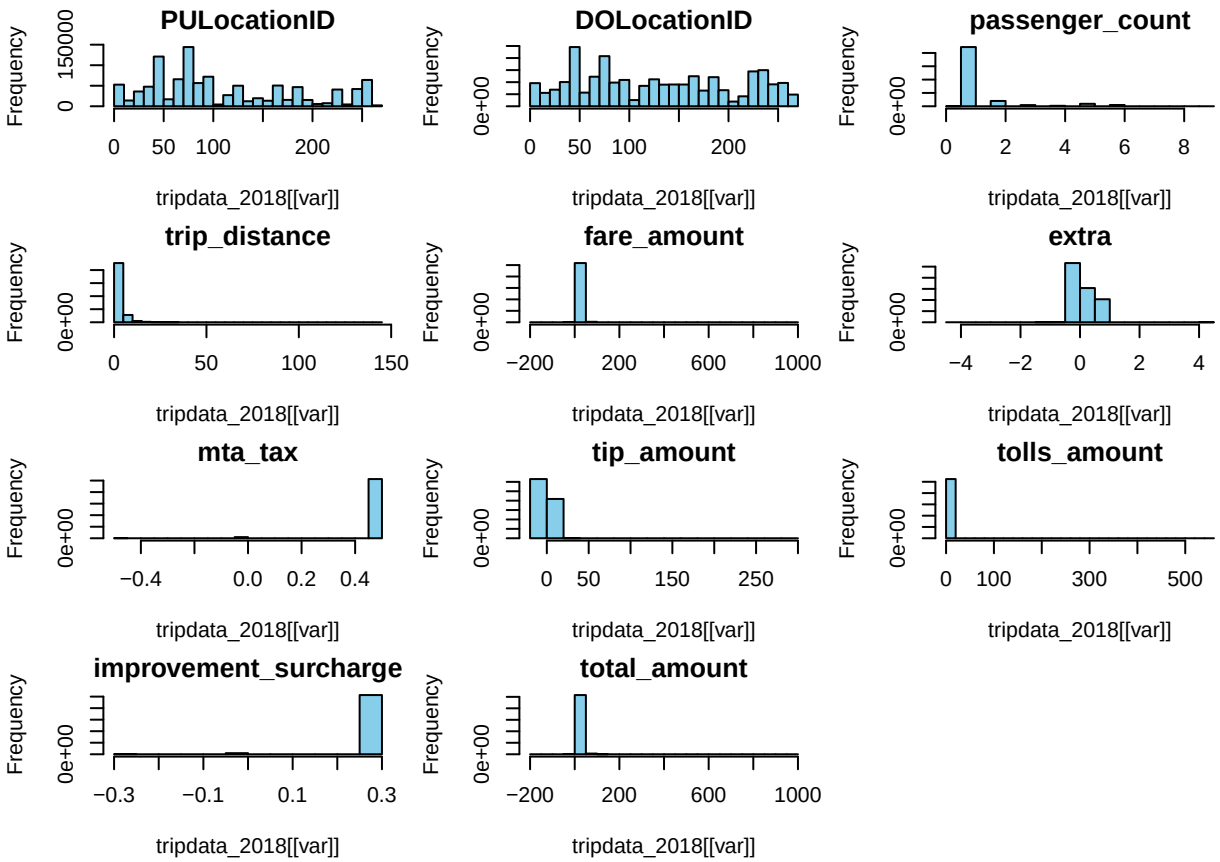


Weekday has no relationship between tip_amount.

1.2 Numeric Features Exploration

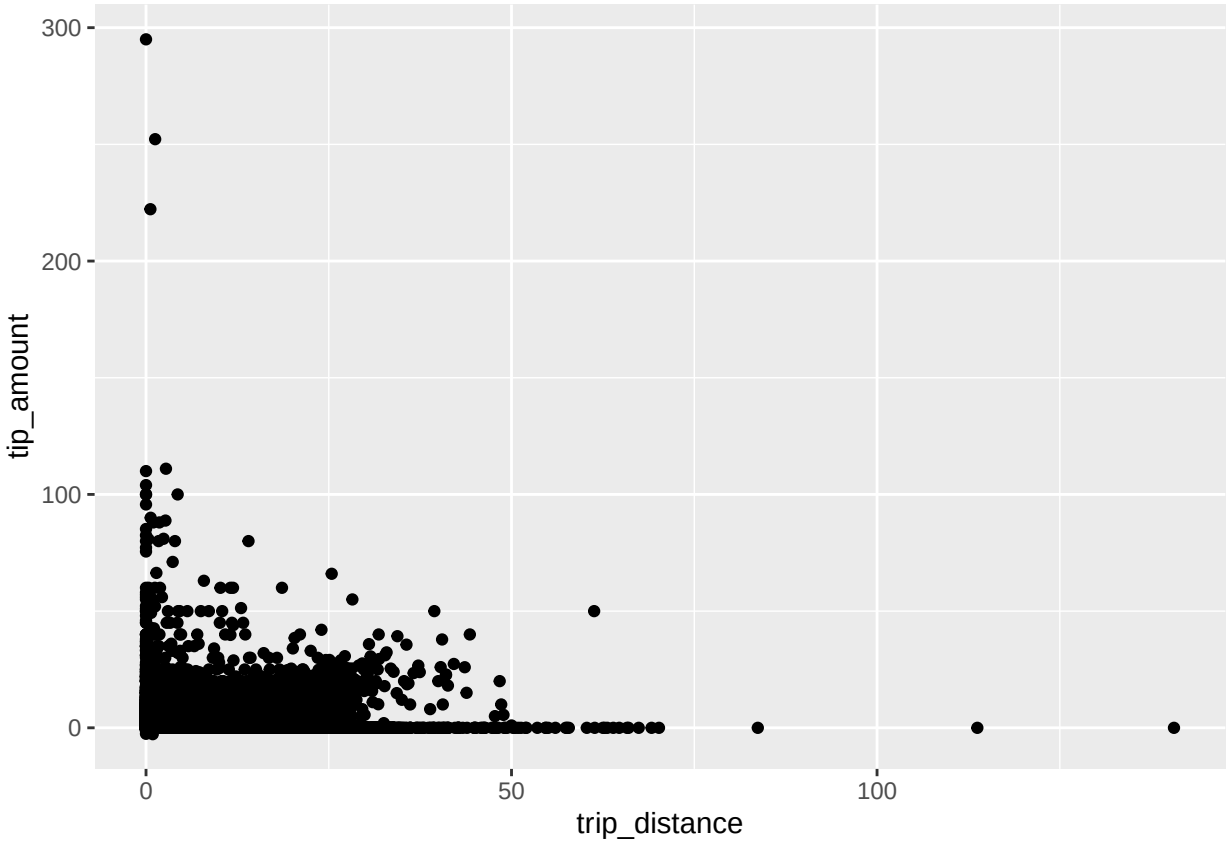
1.2.1 Numeric Features Distribution

```
num_var <- c("PULocationID", "DOLocationID", "passenger_count", "trip_distance", "fare_amount", "extra"

# Calculate the number of rows and columns for the grid
num_cols <- 3
num_rows <- ceiling(length(num_var) / num_cols)

# Create histograms for all numerical variables
par(mfrow=c(num_rows, num_cols), mar=c(4, 4, 2, 1))  # Set up a dynamic grid for subplots with adjusted
for (var in num_var) {
  hist(tripdata_2018[[var]], main = var, col = "skyblue")
}

par(mfrow=c(1, 1))  # Reset the layout to default
```



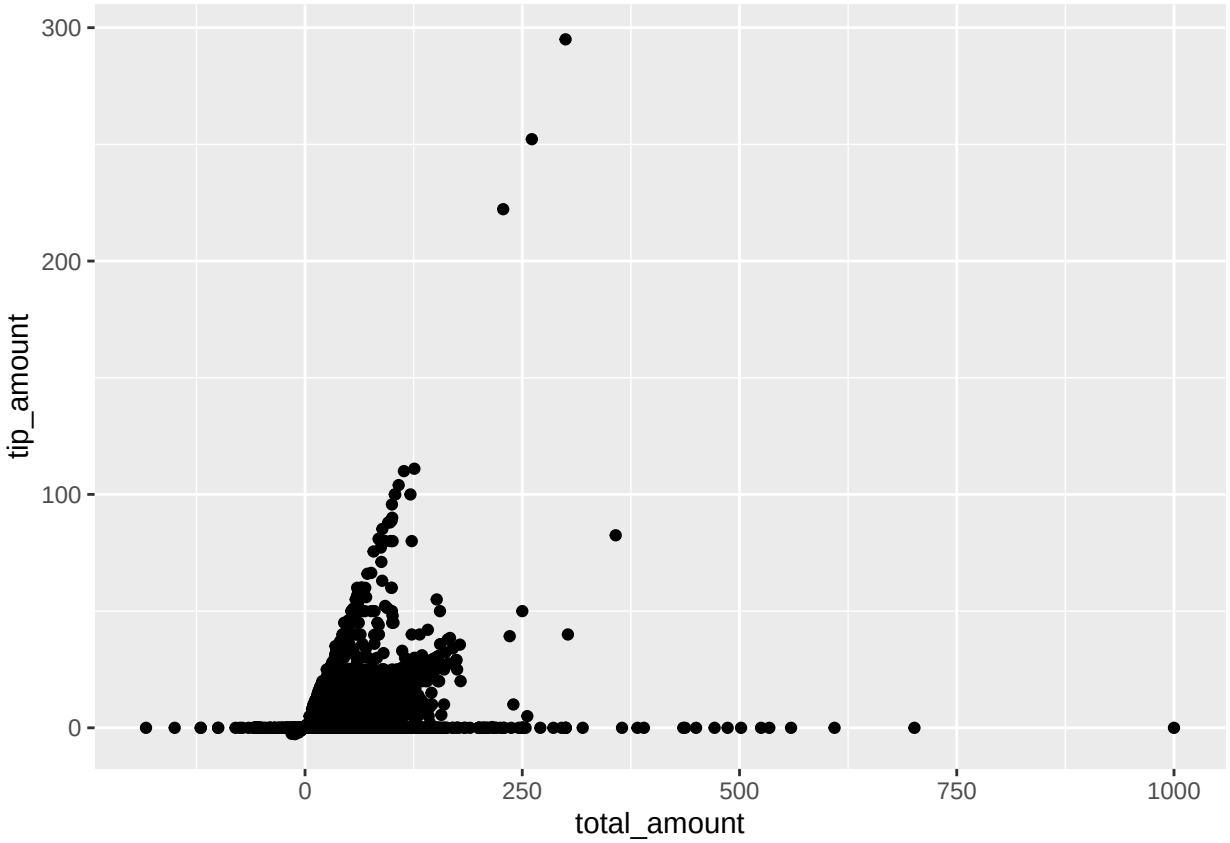
1.2.2 Numeric Features vs. Tip_amount (scatter plots)

```
ggplot(tripdata_2018, aes(x = trip_distance, y = tip_amount)) +
  geom_point()
```




```
ggplot(tripdata_2018, aes(x = total_amount, y = tip_amount)) +
  geom_point()
```

```
## Warning: Removed 5 rows containing missing values (‘geom_point()‘).
```



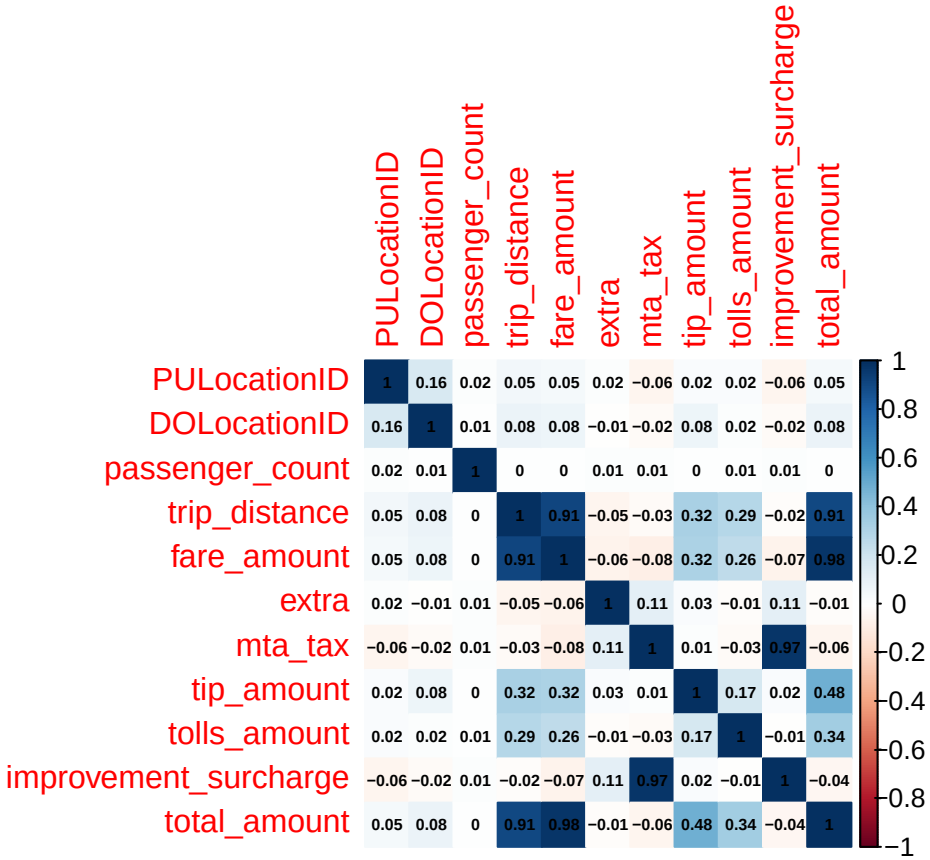
outliers: tip_amount > 200 trip_distance >75 total_amount > 500

Remove outliers above and then investigate the correlation with tip_amount.

1.2.3 Correlation Matrix

```
# select tip_amount < 200 & trip_distance < 75 & total_amount < 500
filtered_df <- tripdata_2018 %>%
  filter(tip_amount < 200 & trip_distance < 75 & total_amount < 500)

# Select numeric values > 0
filtered_df <- filtered_df %>%
  filter(fare_amount >= 0 &
           extra >= 0 &
           mta_tax >= 0 &
           tip_amount >= 0 &
           improvement_surcharge >= 0 &
           total_amount >= 0)

# Select numerical variables to create correlation matrix
numeric_vars <- filtered_df[num_var]

# correlation matrix
cor_matrix <- cor(numeric_vars , use = "pairwise.complete.obs") # ignore NA values
```

```r
# ggcorrplot(cor_matrix)
corrplot(cor_matrix, method = "color", addCoef.col = "black", number.cex = 0.5)
```



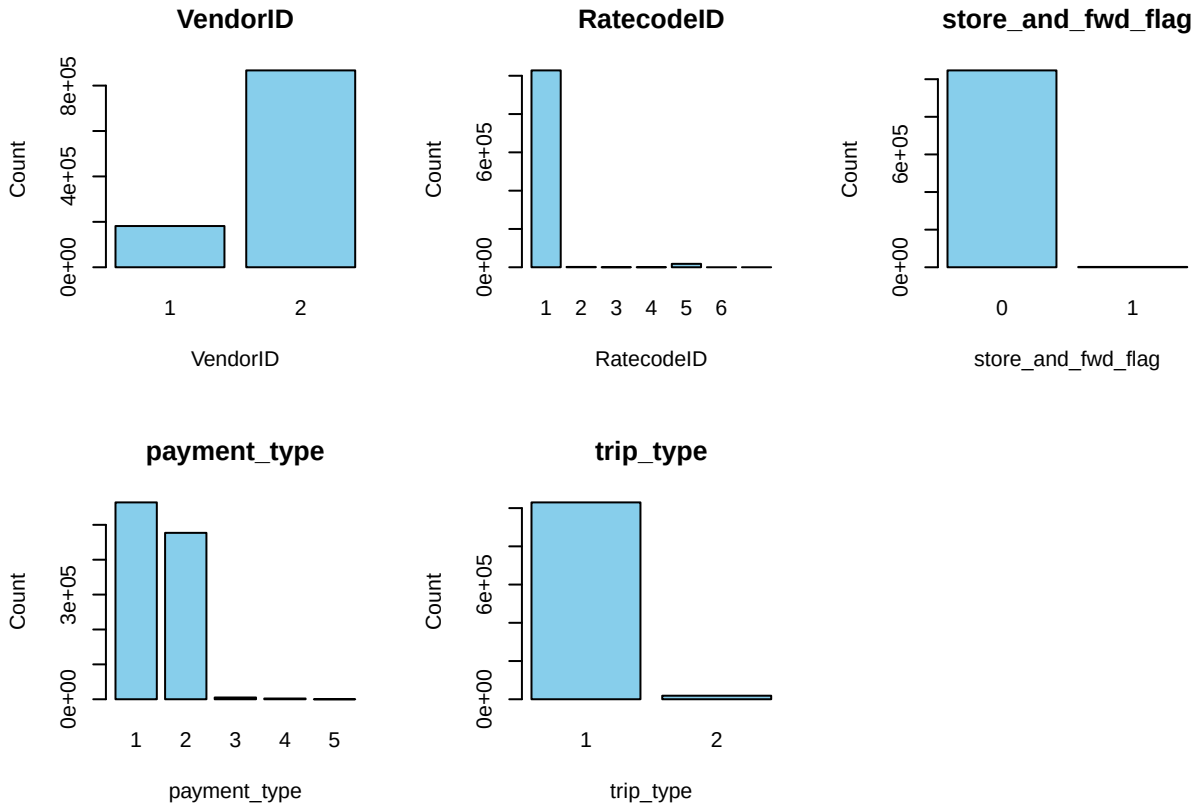
- **Trip_distance** and **total_amount** and **tolls_amount** are more related to tip_amount.

1.3 Categorical Features

1.3.1 Categorical Features Distribution

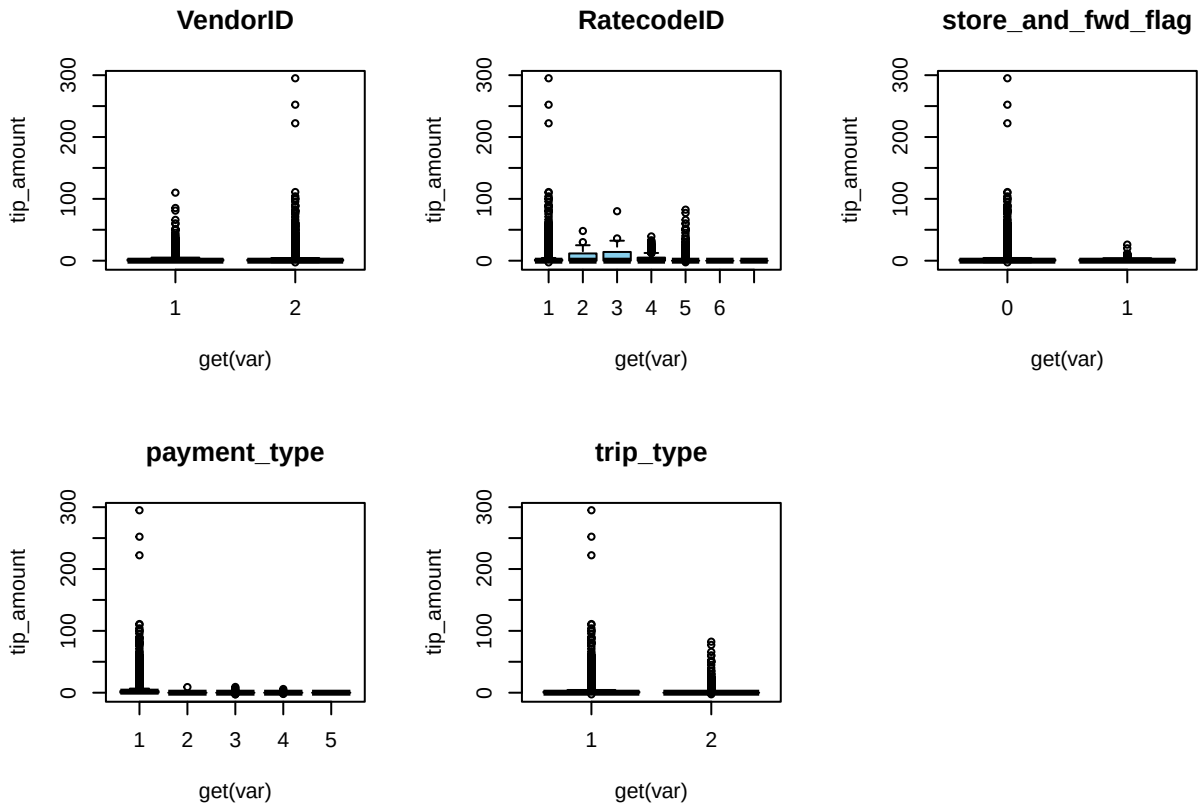
```r
# Define Categorical Variables
cat_var <- c("VendorID", "RatecodeID", "store_and_fwd_flag", "payment_type", "trip_type")

# Convert categorical variables to factors
tripdata_2018[cat_var] <- lapply(tripdata_2018[cat_var], as.factor)

# Set up the layout for subplots
par(mfrow = c(2, 3))

# Create bar plots for each factor variable
for (var in cat_var) {
  # Create a table of frequencies
  table_data <- table(tripdata_2018[[var]])
```

```r
  # Create a bar plot
  barplot(table_data, main = var, col = "skyblue", xlab = var, ylab = "Count")
}

# Reset the layout to default
par(mfrow = c(1, 1))
```



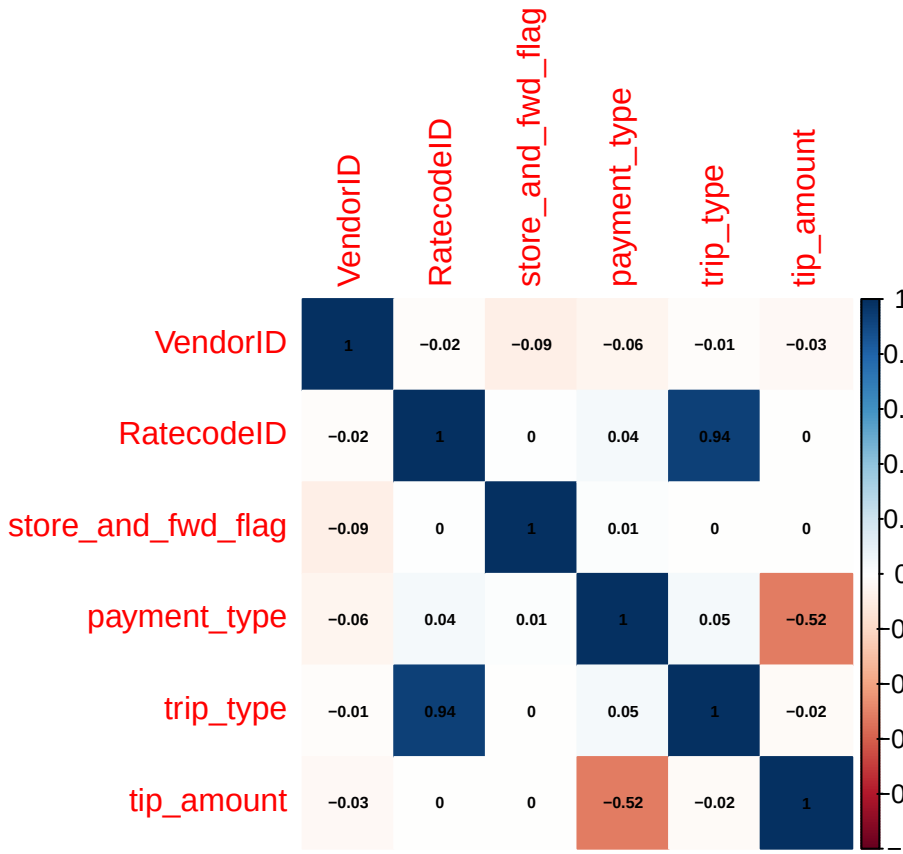
1.3.2 Display categorical features vs. tip_amout

```r
# Define categorical variables
cat_var <- c("VendorID", "RatecodeID", "store_and_fwd_flag", "payment_type", "trip_type")

# Set up the layout for subplots
par(mfrow = c(2, 3))

# Create box plots for each factor variable
for (var in cat_var) {
  boxplot(tip_amount ~ get(var), data = tripdata_2018, col = "skyblue", main = var)
}

# Reset the layout to default
par(mfrow = c(1, 1))
```



In original data set, tip_amount $>$ 200 is outlier.

```
# Select categorical variables data
cat_df <- filtered_df %>%
  select(VendorID, RatecodeID, store_and_fwd_flag, payment_type, trip_type)

# Convert factor variables to numerics
cat_2_num <- cat_df %>%
  mutate_all(~as.numeric(.))

# Combined cat_df and tip_amount
combined_data <- data.frame(cat_2_num, tip_amount = filtered_df$tip_amount)

# Exclude RatecodeID == 99
combined_data <- combined_data %>%
  filter(RatecodeID != 99)

# Get correlation matrix
cor_matrix <- cor(combined_data , use = "pairwise.complete.obs")

# Visualize correlation matrix
corrplot(cor_matrix, method = "color", addCoef.col = "black", number.cex = 0.5)
```



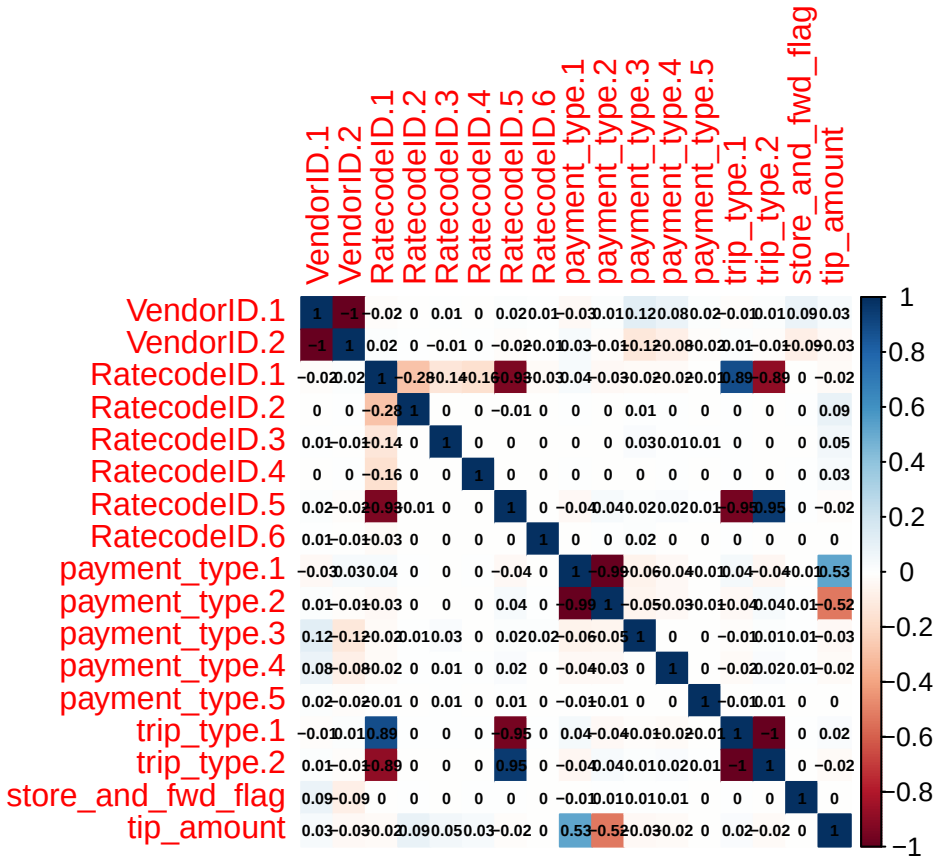
1.3.3. No one-hot encoding

Payment_type is more related

1.3.4 Catgorical var cleaning, Transform one-hot encoding

```
# One-hot encoding for VendorID, RatecodeID, payment_type, trip_type
dummy_data <- dummyVars(" ~ VendorID + RatecodeID + payment_type + trip_type", data = filtered_df)
transformed_data <- predict(dummy_data, newdata = filtered_df)

combined_data <- data.frame(transformed_data, store_and_fwd_flag = filtered_df$store_and_fwd_flag)


# Convert factor variables to numerics
cat_2_num <- combined_data %>%
  mutate_all(~as.numeric(.))

# Combined transformed_data and tip_amount
combined_data <- data.frame(cat_2_num, tip_amount = filtered_df$tip_amount)


# Remove RatecodeID.99 columns
combined_data <- combined_data %>% select(-RatecodeID.99)


# Get correlation matrix
cor_matrix <- cor(combined_data , use = "pairwise.complete.obs")
```

```r
# Visualize correlation matrix
corrplot(cor_matrix, method = "color", addCoef.col = "black", number.cex = 0.5)
```



Payment_type 1 and 2 are more related with tip_amount.

According to correlation coefficients result, we select **Trip_distance** and **total_amount**, **tolls_amount**, and **Payment_type**

```r
# Feature selection
selected_data <- filtered_df %>%
  select(trip_distance, total_amount, tolls_amount, payment_type, tip_amount)

# Transform payment_type to dummy variables
dummy_data <- dummyVars(" ~ payment_type", data = selected_data)
transformed_data <- predict(dummy_data, newdata = selected_data)

# Combined transformed dummy data and selected data
combined_data <- data.frame(transformed_data, selected_data)

# Remove payment_type
combined_data <- combined_data %>%
  select(-payment_type)
```

Q2 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: handle missing data and outliers, perform any suitable data transformation steps, etc. Also, ensure that you filter the data. The goal is to predict the tip amount, therefore you need to ensure that you extract the data that contains this information. Hint: read the data dictionary.

```
# Summarize all NA values in variables
colSums(is.na(combined_data))
```

```
## payment_type.1 payment_type.2 payment_type.3 payment_type.4 payment_type.5
##              0              0              0              0              0
##  trip_distance   total_amount   tolls_amount     tip_amount
##              0              0              0              0
```

There is no NA values in the selected data.

• **Normalize the data:** perform either max-min normalization or z-score standardization on the continuous variables/features.

```
# Normalize the numerical variables in the data set
# Define a function for min-max normalization
min_max_normalize <- function(x) { (x - min(x)) / (max(x) - min(x))
}
# min-max normalization
normalized_subset <- data.frame(lapply(combined_data[, 1:8], min_max_normalize))

normalized_df <- data.frame(normalized_subset, tip_amount = combined_data$tip_amount)

tail(normalized_df, 3)
```

```
##         payment_type.1 payment_type.2 payment_type.3 payment_type.4
## 1045926              1              0              0              0
## 1045927              1              0              0              0
## 1045928              1              0              0              0
##         payment_type.5 trip_distance total_amount tolls_amount tip_amount
## 1045926              0    0.02921476   0.02648571            0       2.58
## 1045927              0    0.01054582   0.01603948            0       2.00
## 1045928              0    0.02750463   0.02389472            0       2.32
```

• Prepare the data for modeling: shuffle the data and split it into training and test sets. The percent split between the training and test set is your decision. However, clearly indicate the reason.

```
# Setting seed for reproducibility
set.seed(123)

# Create an index for the training set (80%) and the test set (20%)
index <- createDataPartition(normalized_df$tip_amount, p = 0.8, list = FALSE) # Create training set

data_train <- normalized_df[index, ] # Create test set
data_test <- normalized_df[-index, ]
```

```r
# Display the dimensions of the training and test sets
cat("Training set dimensions:", dim(data_train), "\n")
```

```
## Training set dimensions: 836743 9
```

Question 3

In this step you will develop the k-nn regression model. Create a function with the following name and arguments: knn.predict(data_train, data_test, k); Perform the following logic inside the function: Implement the k-nn algorithm and use it to predict the tip amount for each observation in the test set i.e. data_test.
• Note: You are not required to implement the k-nn algorithm from scratch. Therefore, this step may only involve providing the training set, the test set, and the value of k to your chosen k-nn library.
• Calculate the mean squared error (MSE) between the predictions from the k-nn model and the actual tip amount in the test set.
• The knn-predict() function should return the MSE.

```r
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)

  return(mse)
}
```

```r
# knn.predict(data_train = data_train, data_test = data_test, k = 3)
```

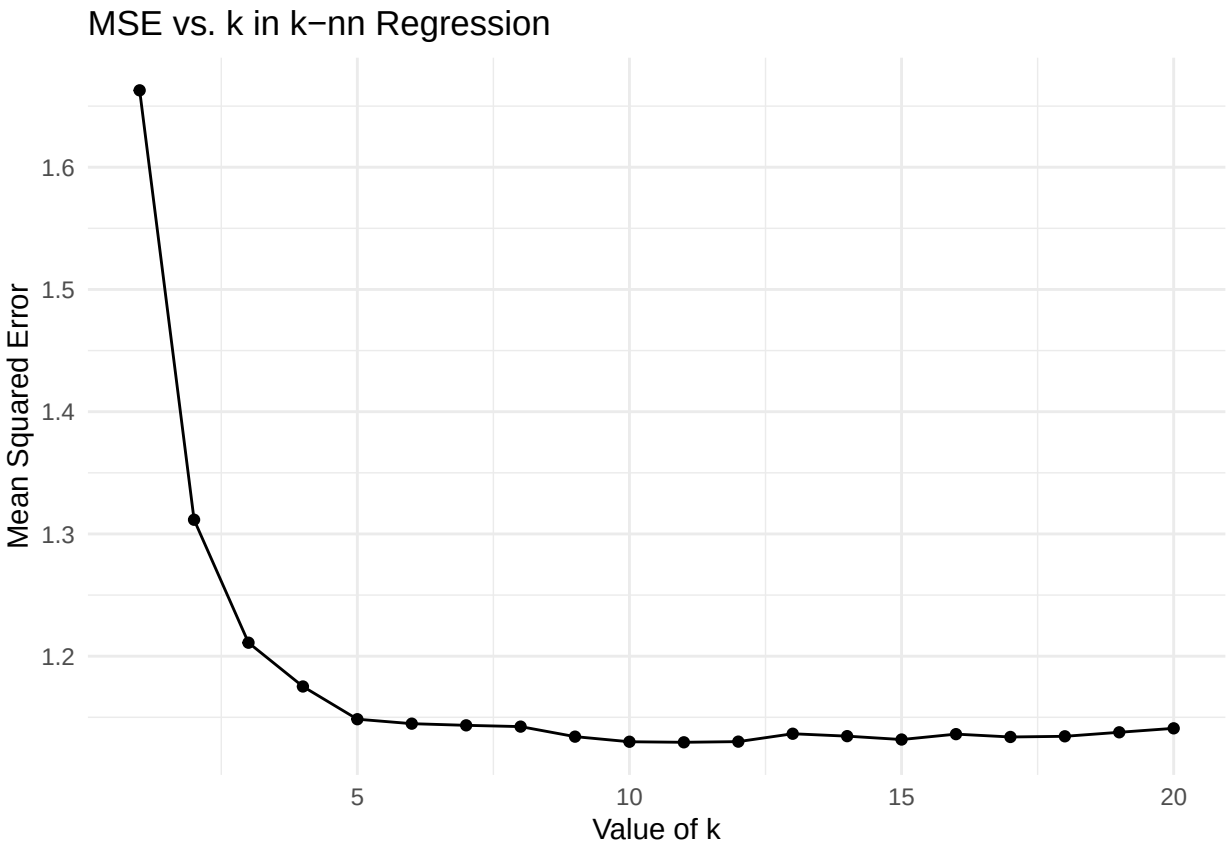
```r
# knn.predict(data_train = data_train, data_test = data_test, k = 10)
```

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.
• 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.
• What are your thoughts on the model that you developed and the accuracy of its predictions? Would you advocate for its use to predict the tip amount of future trips? Explain your answer.

```r
# Create a vector of different values of k
k_values <- seq(1, 20)

# Initialize an empty vector to store MSE values
mse_values <- numeric(length(k_values))

# Loop through each value of k
for (i in seq_along(k_values)) {
  mse_values[i] <- knn.predict(data_train, data_test, k_values[i])
}

# Create a data frame for plotting
mse_df <- data.frame(k = k_values, mse = mse_values)

# Plot the line chart
ggplot(mse_df, aes(x = k, y = mse)) +
  geom_line() +
  geom_point() +
  labs(title = "MSE vs. k in k-nn Regression",
       x = "Value of k",
       y = "Mean Squared Error") +
  theme_minimal()
```




```
# Find the k value with the lowest MSE
best_k <- mse_df$k[which.min(mse_df$mse)]
min_mse <- min(mse_df$mse)
cat("The value of k with the lowest MSE is:", best_k, "\n")
```

```
## The value of k with the lowest MSE is: 11
```

```
cat("The minimum MSE is:", min_mse,  "\n")
```

```
## The minimum MSE is: 1.129538
```

The line graph above shows that the minimum MSE value of 1.129 is associated with a k value of 11.

Question 5

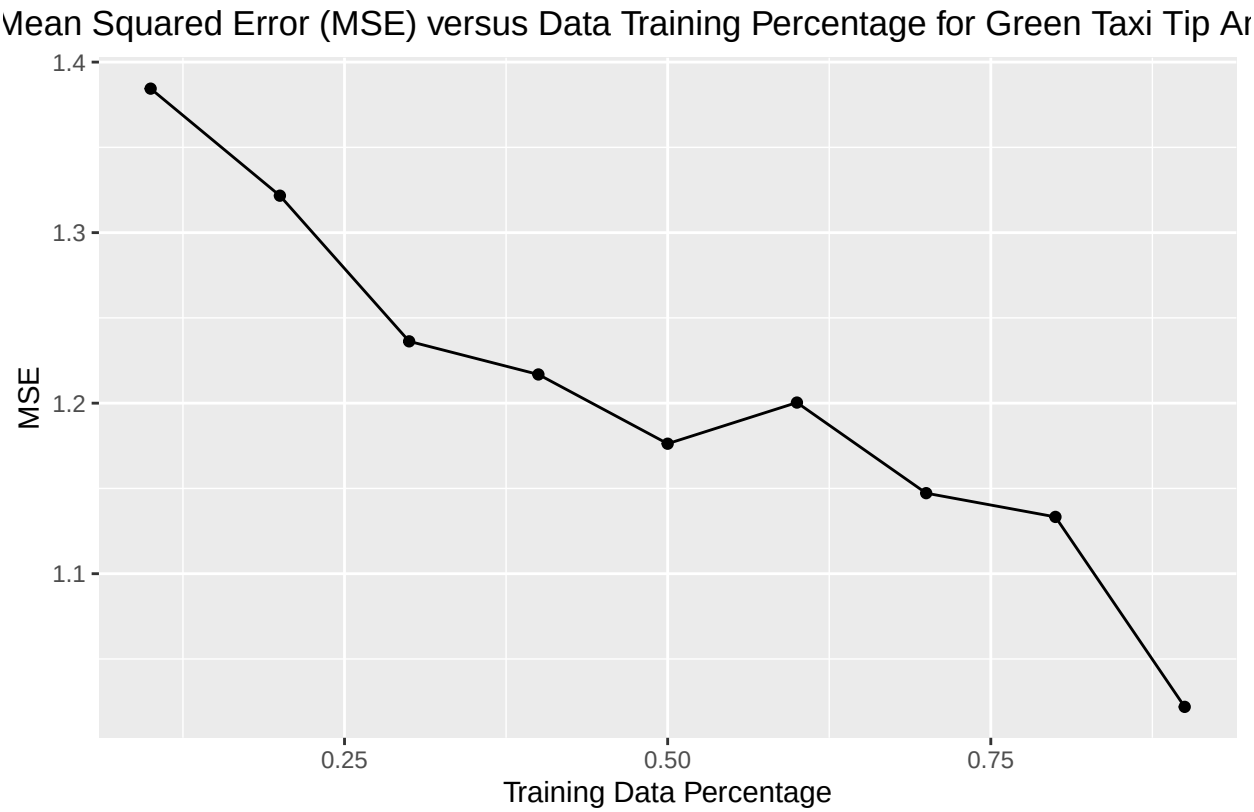
Evaluate the effect of the percentage split for the training and test sets and determine if a different split ratio improves your model's ability to make better predictions.

```
# creating vector of all training and testing data splits from 0.1 to 0.9 at an increment of 0.01
splits <- seq(0.1,0.9,0.1)


# create an empty data frame to store Data_Training_Percentage and MSE values
```

```r
df_split <- data.frame(matrix(ncol = 2, nrow = length(splits)))
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 constant
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 = 11)
}

# identifying the minimum MSE and associated data training and testing percentages from the data traini
min_split_MSE <- min(df_split$MSE)
min_split_index <- which.min(df_split$MSE)
min_split_train <- df_split$Data_Training_Percentage[min_split_index]

cat(sprintf("The minimum MSE is %f with an associated data training percentage of %.2f and an associate
            min_split_MSE, min_split_train, 1 - min_split_train))
```

```
## The minimum MSE is 1.021922 with an associated data training percentage of 0.90 and an associated tes
```

```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 Data Training Percentage 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))
```



ixed k value. The x−axis shows the data training percentage (data testing percentage = 1 − data training percentage)