ETE3-1.R

Pranab Rai-2447137

2025-01-01

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 4.4.2
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
library(ggplot2)


 # Load your data
  df <-
read.csv("C:\\Users\\prana\\OneDrive\\Desktop\\2trimester\\R\\ETE3\\test1.csv
")

  # Convert pickup datetime
  df$tpep_pickup_datetime <- ymd_hms(df$tpep_pickup_datetime)

  # Extract date and hour
  df <- df %>%
    mutate(
      pickup_date = as.Date(tpep_pickup_datetime),
      pickup_hour = hour(tpep_pickup_datetime)
    )

  # Basic Summary
  print("Basic Summary:")
```

```
## [1] "Basic Summary:"
```

```
  print(summary(df))
```

```
##   pickup_date          pickup_hour    VendorID   
##  Min.   :2001-09-21   Min.   :0    Min.   :1.00  
##  1st Qu.:2008-09-21   1st Qu.:0    1st Qu.:1.75  
##  Median :2016-03-22   Median :0    Median :2.00  
##  Mean   :2016-03-22   Mean   :0    Mean   :1.75  
##  3rd Qu.:2023-09-21   3rd Qu.:0    3rd Qu.:2.00  
##  Max.   :2030-09-21   Max.   :0    Max.   :2.00  
##  tpep_pickup_datetime             tpep_dropoff_datetime passenger_count
##  Min.   :2001-09-21 00:00:15.00   Length:720            Min.   :0.000  
##  1st Qu.:2008-09-21 00:12:05.50   Class :character      1st Qu.:1.000  
##  Median :2016-03-22 00:11:48.00   Mode  :character      Median :1.000  
##  Mean   :2016-03-22 00:11:59.49                         Mean   :1.274  
##  3rd Qu.:2023-09-21 00:12:09.50                         3rd Qu.:1.000  
##  Max.   :2030-09-21 00:23:06.00                         Max.   :6.000  
##  trip_distance      RatecodeID     store_and_fwd_flag  PULocationID  
##  Min.   : 0.000   Min.   : 1.000   Length:720         Min.   :  4.0  
##  1st Qu.: 1.070   1st Qu.: 1.000   Class :character   1st Qu.:132.0  
##  Median : 1.890   Median : 1.000   Mode  :character   Median :161.0  
##  Mean   : 3.616   Mean   : 2.224                      Mean   :164.7  
##  3rd Qu.: 3.595   3rd Qu.: 1.000                      3rd Qu.:233.0  
##  Max.   :22.550   Max.   :99.000                      Max.   :265.0  
##   DOLocationID    payment_type    fare_amount          extra       
##  Min.   :  1.0   Min.   :1.000   Min.   :-107.30   Min.   :-5.000  
##  1st Qu.:106.8   1st Qu.:1.000   1st Qu.:   9.30   1st Qu.: 0.000  
##  Median :161.0   Median :1.000   Median :  13.50   Median : 1.000  
##  Mean   :156.7   Mean   :1.228   Mean   :  19.93   Mean   : 1.417  
##  3rd Qu.:231.0   3rd Qu.:1.000   3rd Qu.:  23.30   3rd Qu.: 2.500  
##  Max.   :265.0   Max.   :4.000   Max.   : 130.40   Max.   : 9.250  
##     mta_tax           tip_amount       tolls_amount      
improvement_surcharge
##  Min.   :-0.5000   Min.   : 0.000   Min.   :-13.3800   Min.   :-1.0000  
##  1st Qu.: 0.5000   1st Qu.: 1.000   1st Qu.:  0.0000   1st Qu.: 1.0000  
##  Median : 0.5000   Median : 2.860   Median :  0.0000   Median : 1.0000  
##  Mean   : 0.4736   Mean   : 3.738   Mean   :  0.5872   Mean   : 0.9542  
##  3rd Qu.: 0.5000   3rd Qu.: 4.400   3rd Qu.:  0.0000   3rd Qu.: 1.0000  
##  Max.   : 0.5000   Max.   :40.050   Max.   : 21.3800   Max.   : 1.0000  
##   total_amount     congestion_surcharge  Airport_fee    
##  Min.   :-121.68   Min.   :-2.500       Min.   :-1.7500  
##  1st Qu.:  15.86   1st Qu.: 2.500       1st Qu.: 0.0000  
##  Median :  21.68   Median : 2.500       Median : 0.0000  
##  Mean   :  28.85   Mean   : 2.205       Mean   : 0.1434  
##  3rd Qu.:  31.32   3rd Qu.: 2.500       3rd Qu.: 0.0000  
##  Max.   : 175.30   Max.   : 2.500       Max.   : 1.7500
```

```
  # More Detailed Descriptive Statistics
  # Numerical Columns
  numerical_cols <- c("trip_distance", "fare_amount", "total_amount",
"passenger_count") # Add more as needed
  for (col in numerical_cols) {
    if (col %in% names(df)) {
      print(paste("n", col, "Statistics:"))
      print(paste("Mean", col, ":", mean(df[[col]], na.rm = TRUE)))
      print(paste("Median", col, ":", median(df[[col]], na.rm = TRUE)))
      print(paste("Standard Deviation of", col, ":", sd(df[[col]], na.rm =
TRUE)))
      print(paste("Range of", col, ":", paste(range(df[[col]], na.rm=TRUE),
collapse = " - ")))
      print(paste("Interquartile Range (IQR) of", col, ":", IQR(df[[col]],
na.rm = TRUE)))
      print("Quantiles")
      print(quantile(df[[col]], probs = c(0.05,0.25, 0.5, 0.75,0.95),
na.rm=TRUE))
    }
  }
```

```
## [1] "n trip_distance Statistics:"
## [1] "Mean trip_distance : 3.61588888888889"
## [1] "Median trip_distance : 1.89"
## [1] "Standard Deviation of trip_distance : 4.5533884784586"
## [1] "Range of trip_distance : 0 - 22.55"
## [1] "Interquartile Range (IQR) of trip_distance : 2.525"
## [1] "Quantiles"
##      5%     25%     50%     75%     95% 
##  0.4895  1.0700  1.8900  3.5950 16.3160 
## [1] "n fare_amount Statistics:"
## [1] "Mean fare_amount : 19.9270277777778"
## [1] "Median fare_amount : 13.5"
## [1] "Standard Deviation of fare_amount : 20.0215538600284"
## [1] "Range of fare_amount : -107.3 - 130.4"
## [1] "Interquartile Range (IQR) of fare_amount : 14"
## [1] "Quantiles"
##   5%  25%  50%  75%  95% 
##  5.1  9.3 13.5 23.3 70.0 
## [1] "n total_amount Statistics:"
## [1] "Mean total_amount : 28.8541388888889"
## [1] "Median total_amount : 21.675"
## [1] "Standard Deviation of total_amount : 25.4034276449892"
## [1] "Range of total_amount : -121.68 - 175.3"
## [1] "Interquartile Range (IQR) of total_amount : 15.465"
## [1] "Quantiles"
##      5%     25%     50%     75%     95% 
## 11.2760 15.8600 21.6750 31.3250 88.8505 
## [1] "n passenger_count Statistics:"
## [1] "Mean passenger_count : 1.27361111111111"
## [1] "Median passenger_count : 1"
## [1] "Standard Deviation of passenger_count : 0.718422986186287"
## [1] "Range of passenger_count : 0 - 6"
## [1] "Interquartile Range (IQR) of passenger_count : 0"
## [1] "Quantiles"
##  5% 25% 50% 75% 95% 
##   1   1   1   1   3
```

```
  # Categorical Columns
  categorical_cols <- c("VendorID", "payment_type") # Add more as needed
  for (col in categorical_cols) {
    if (col %in% names(df)) {
      print(paste( col, "Frequencies:"))
      print(table(df[[col]]))
      print("Proportions")
      print(prop.table(table(df[[col]])))
    }
  }
```

```
## [1] "VendorID Frequencies:"
## 
##   1   2 
## 180 540 
## [1] "Proportions"
## 
##    1    2 
## 0.25 0.75 
## [1] "payment_type Frequencies:"
## 
##   1   2   3   4 
## 596 102   4  18 
## [1] "Proportions"
## 
##           1           2           3           4 
## 0.827777778 0.141666667 0.005555556 0.025000000
```

```
  # Combined Statistics (Example: Average Fare Amount by Hour of Day)
  if ("fare_amount" %in% names(df) & "pickup_hour" %in% names(df)){
    print("Average Fare Amount by Hour of Day:")
    print(aggregate(fare_amount ~ pickup_hour, data = df, FUN = mean,
na.rm=TRUE))
  }
```

```
## [1] "Average Fare Amount by Hour of Day:"
##   pickup_hour fare_amount
## 1           0    19.92703
```

```
  #Checking for missing values
  print("Missing Values per column")
```

```
## [1] "Missing Values per column"
```

```r
print(colSums(is.na(df)))
```

```
##           pickup_date           pickup_hour              VendorID 
##                     0                     0                     0 
##  tpep_pickup_datetime tpep_dropoff_datetime       passenger_count 
##                     0                     0                     0 
##         trip_distance            RatecodeID    store_and_fwd_flag 
##                     0                     0                     0 
##          PULocationID          DOLocationID          payment_type 
##                     0                     0                     0 
##           fare_amount                 extra               mta_tax 
##                     0                     0                     0 
##            tip_amount          tolls_amount improvement_surcharge 
##                     0                     0                     0 
##          total_amount  congestion_surcharge           Airport_fee 
##                     0                     0                     0
```

```r
#Data Type of each column
print("Data Type of each column")
```

```
## [1] "Data Type of each column"
```

```r
print(sapply(df, class))
```

```
## $pickup_date
## [1] "Date"
## 
## $pickup_hour
## [1] "integer"
## 
## $VendorID
## [1] "integer"
## 
## $tpep_pickup_datetime
## [1] "POSIXct" "POSIXt" 
## 
## $tpep_dropoff_datetime
## [1] "character"
## 
## $passenger_count
## [1] "integer"
## 
## $trip_distance
## [1] "numeric"
## 
## $RatecodeID
## [1] "integer"
## 
## $store_and_fwd_flag
## [1] "character"
## 
```

```
## $PULocationID
## [1] "integer"
## 
## $DOLocationID
## [1] "integer"
## 
## $payment_type
## [1] "integer"
## 
## $fare_amount
## [1] "numeric"
## 
## $extra
## [1] "numeric"
## 
## $mta_tax
## [1] "numeric"
## 
## $tip_amount
## [1] "numeric"
## 
## $tolls_amount
## [1] "numeric"
## 
## $improvement_surcharge
## [1] "integer"
## 
## $total_amount
## [1] "numeric"
## 
## $congestion_surcharge
## [1] "numeric"
## 
## $Airport_fee
## [1] "numeric"
```

```
  # Number of rows and columns
  print(paste("Number of rows:", nrow(df)))
```

```
## [1] "Number of rows: 720"
```

```
  print(paste("Number of columns:", ncol(df)))
```

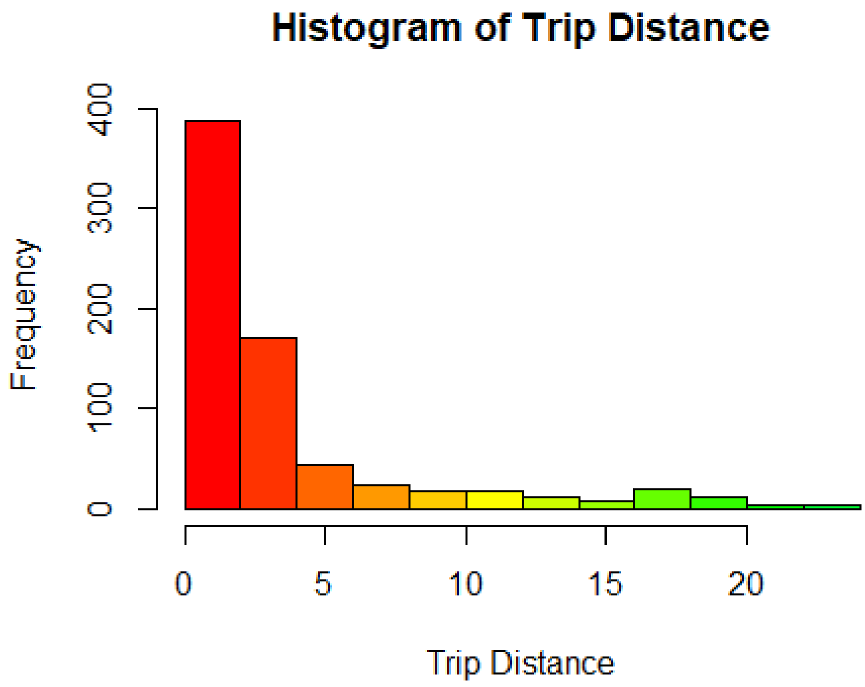
```
## [1] "Number of columns: 21"
```

```
#########

  # Extract date and hour
  df <- df %>%
    mutate(
      pickup_date = as.Date(tpep_pickup_datetime),
      pickup_hour = hour(tpep_pickup_datetime),
      day_of_week = wday(tpep_pickup_datetime, label = TRUE) #Add day of week
    )

  # --- Data Visualization ---
  print("Data Visualizations:")
```

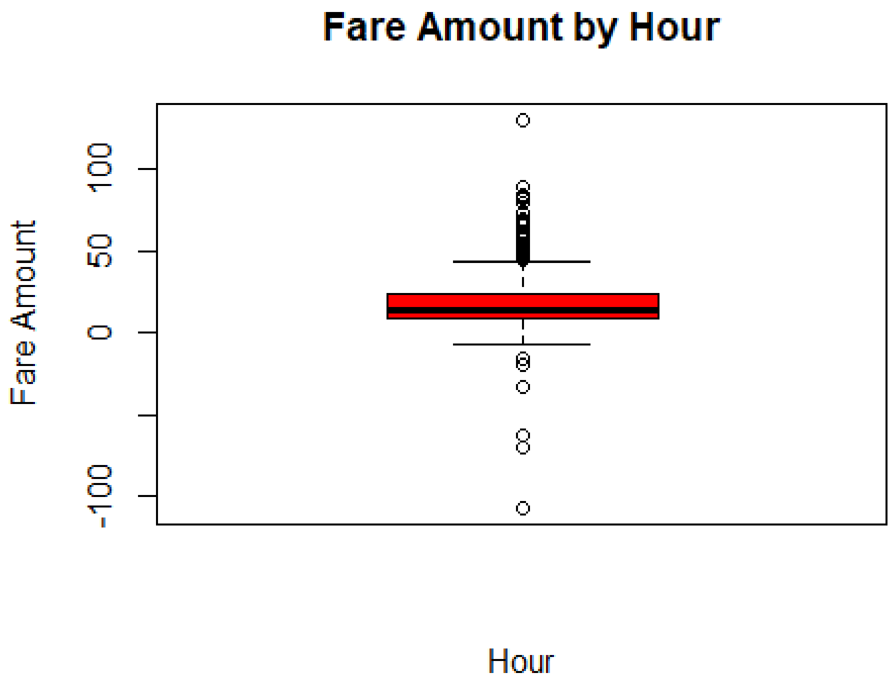
```
## [1] "Data Visualizations:"
```

```
  # 1. Histogram of Trip Distance
hist(df$trip_distance, main = "Histogram of Trip Distance", xlab = "Trip
Distance", na.rm=TRUE, col = rainbow(30)) # Rainbow color palette
```

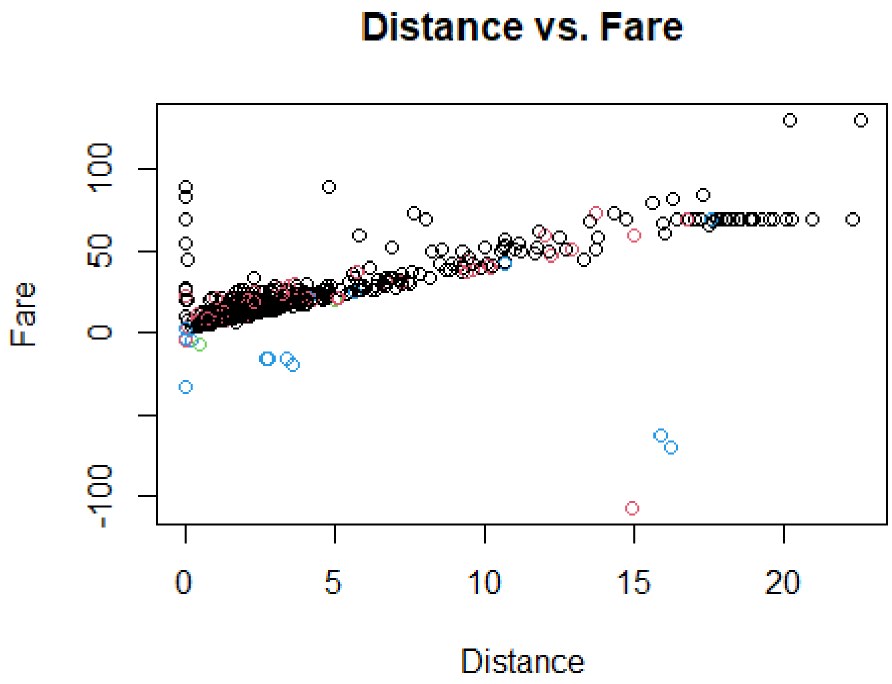
```
## Warning in plot.window(xlim, ylim, "", ...): "na.rm" is not a graphical
## parameter

## Warning in title(main = main, sub = sub, xlab = xlab, ylab = ylab, ...):
## "na.rm" is not a graphical parameter

## Warning in axis(1, ...): "na.rm" is not a graphical parameter

## Warning in axis(2, at = yt, ...): "na.rm" is not a graphical parameter
```



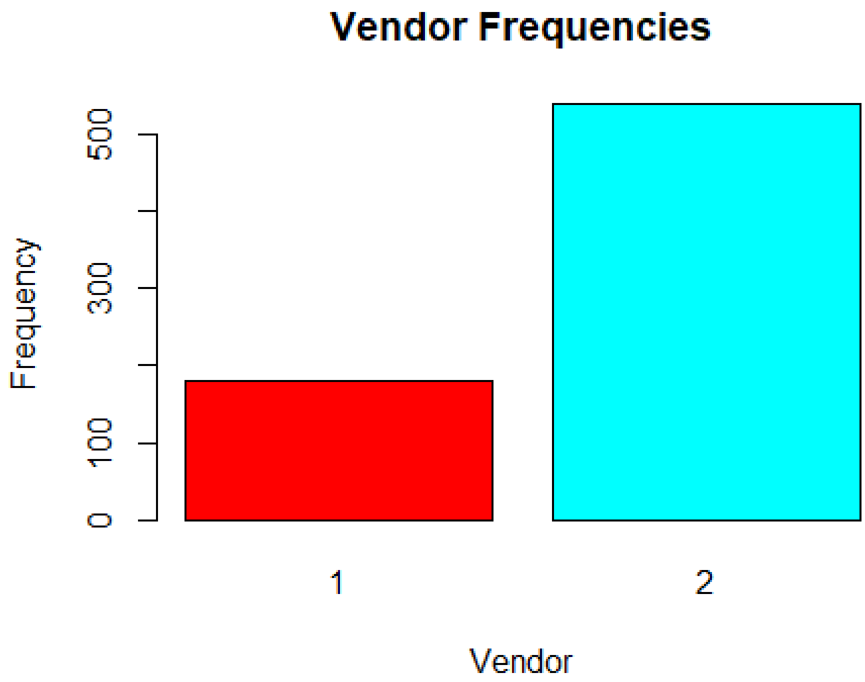
```
  # 2. Boxplot of Fare Amount by Hour of Day
boxplot(fare_amount ~ pickup_hour, data = df, main = "Fare Amount by Hour",
xlab = "Hour", ylab = "Fare Amount", na.rm=TRUE, col = rainbow(24)) # Rainbow
color palette for each hour
```



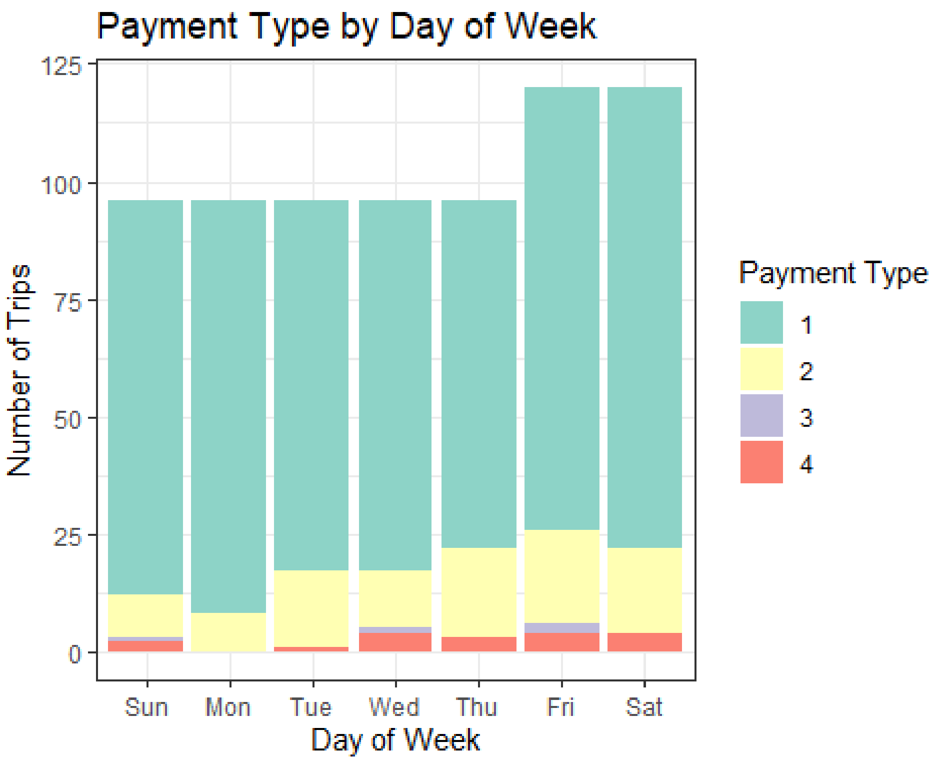
```
  # 3. Scatterplot of Trip Distance vs. Fare Amount with color by payment
type
plot(df$trip_distance, df$fare_amount, main = "Distance vs. Fare", xlab =
"Distance", ylab = "Fare", na.rm=TRUE, col = factor(df$payment_type)) # Color
by payment type

## Warning in plot.window(...): "na.rm" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "na.rm" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "na.rm" is
not a
## graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "na.rm" is
not a
## graphical parameter

## Warning in box(...): "na.rm" is not a graphical parameter

## Warning in title(...): "na.rm" is not a graphical parameter
```



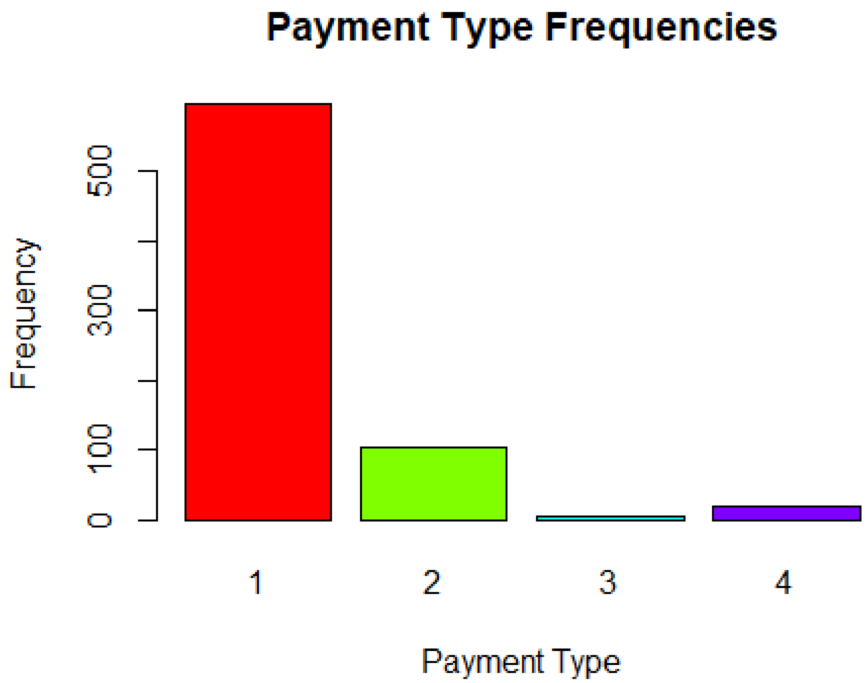
```
  # 4. Bar Chart of Vendor ID
barplot(table(df$VendorID), main = "Vendor Frequencies", xlab = "Vendor",
ylab = "Frequency", col = rainbow(nlevels(factor(df$VendorID)))) # Rainbow
color palette for each vendor
```



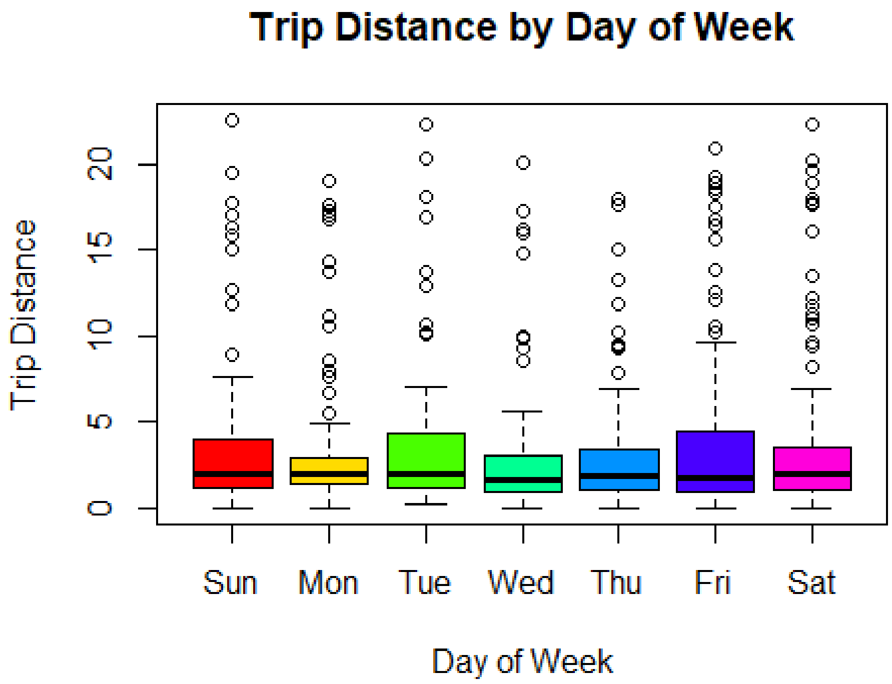
```
  # 5.Payment Type by Day of Week
ggplot(df, aes(x = day_of_week, fill = factor(payment_type))) +
          geom_bar(position = "stack") +
          labs(title = "Payment Type by Day of Week", x = "Day of Week", y =
"Number of Trips", fill = "Payment Type") +
          theme_bw() +
          scale_fill_brewer(palette="Set3")
```



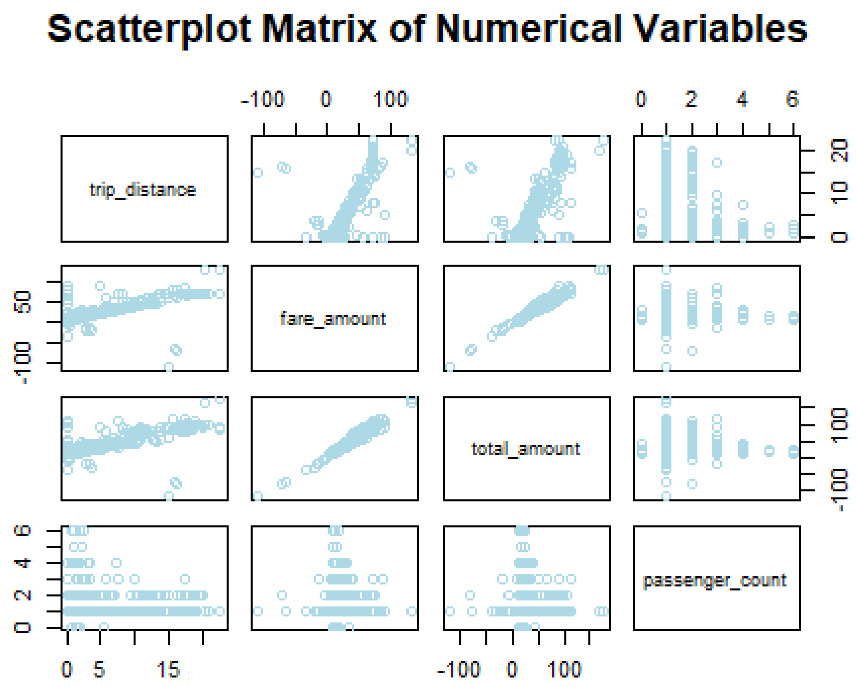
```
  # 6. Bar chart of payment type
barplot(table(df$payment_type), main = "Payment Type Frequencies", xlab =
"Payment Type", ylab = "Frequency", col =
rainbow(nlevels(factor(df$payment_type)))) # Rainbow color palette for each
payment type
```



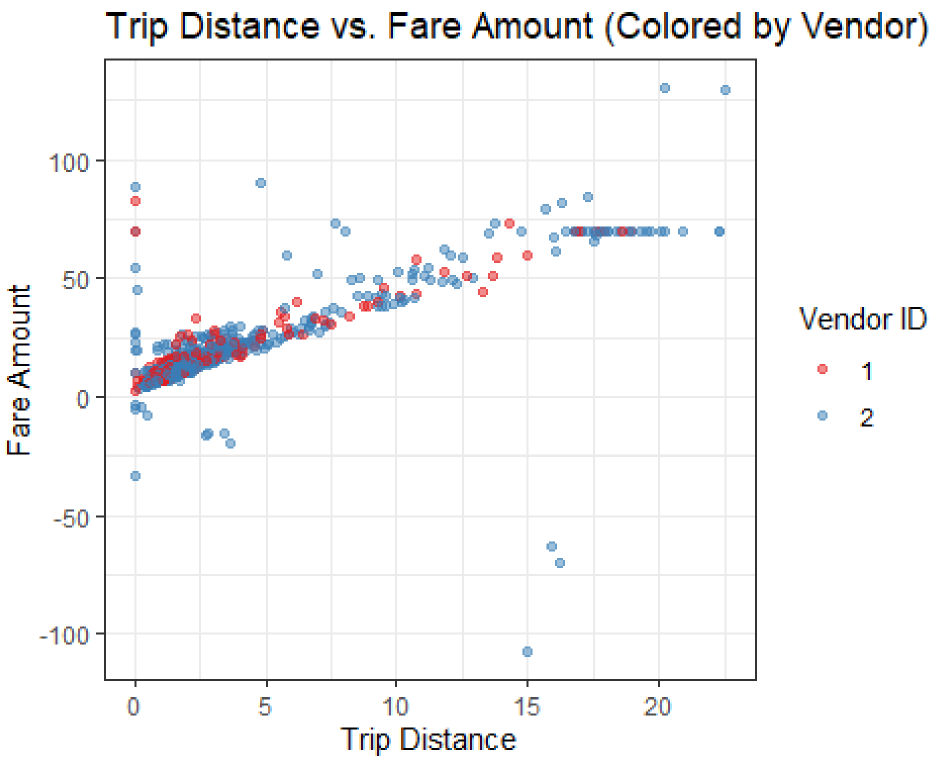
```
  # 7. Boxplot of trip distance by day of the week
boxplot(trip_distance ~ day_of_week, data = df, main = "Trip Distance by Day
of Week", xlab = "Day of Week", ylab = "Trip Distance", na.rm=TRUE, col =
rainbow(7)) # Rainbow color palette for each day of the week
```



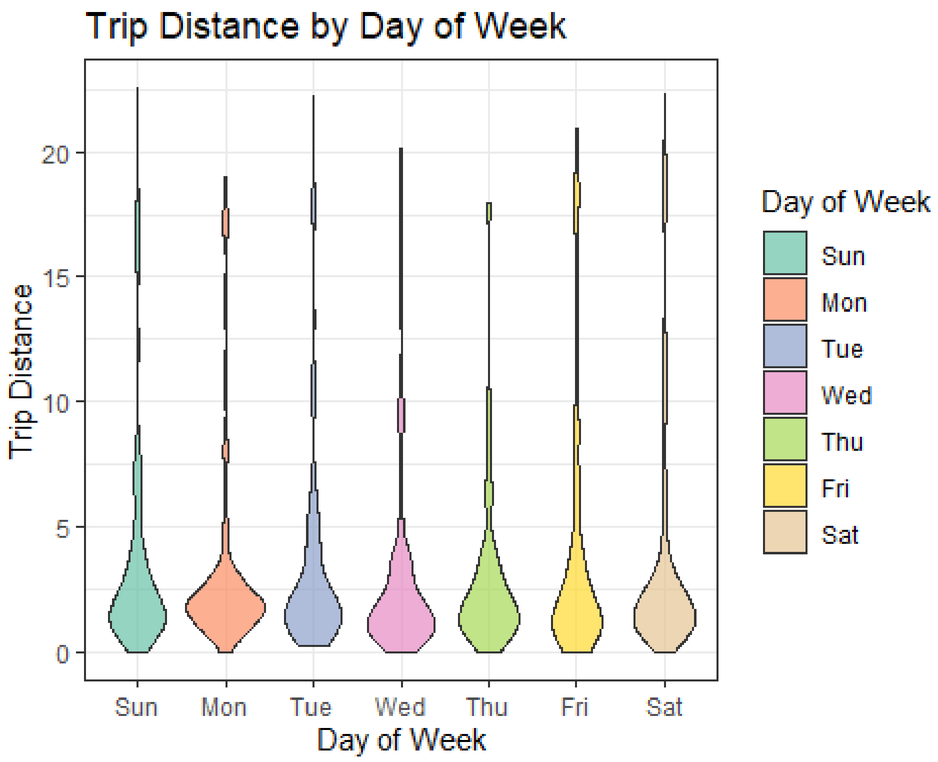
```
  # 8. Scatterplot matrix for numerical variables (if more than 2)
  numerical_cols <- c("trip_distance", "fare_amount", "total_amount",
"passenger_count")
  numerical_data <- df[, numerical_cols[numerical_cols %in% names(df)]] #
Select only available numerical cols
  if (ncol(numerical_data) >= 2) {
    pairs(numerical_data, main = "Scatterplot Matrix of Numerical Variables",
col = "lightblue") # Single color for scatterplot matrix
  }
```



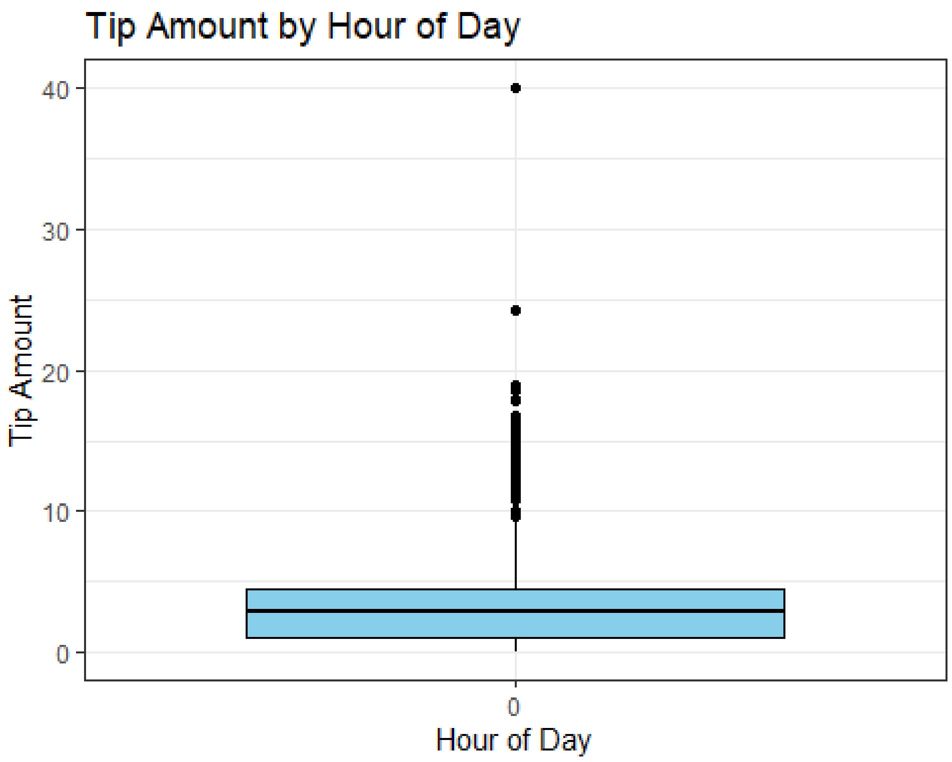
```
  # 9. Using ggplot trip vs fare amount
ggplot(df, aes(x = trip_distance, y = fare_amount, color =
as.factor(VendorID))) +
            geom_point(alpha = 0.5) + # Added alpha for transparency
            labs(title = "Trip Distance vs. Fare Amount (Colored by Vendor)",
x = "Trip Distance", y = "Fare Amount", color = "Vendor ID") +
            theme_bw() +
            scale_color_brewer(palette="Set1")
```



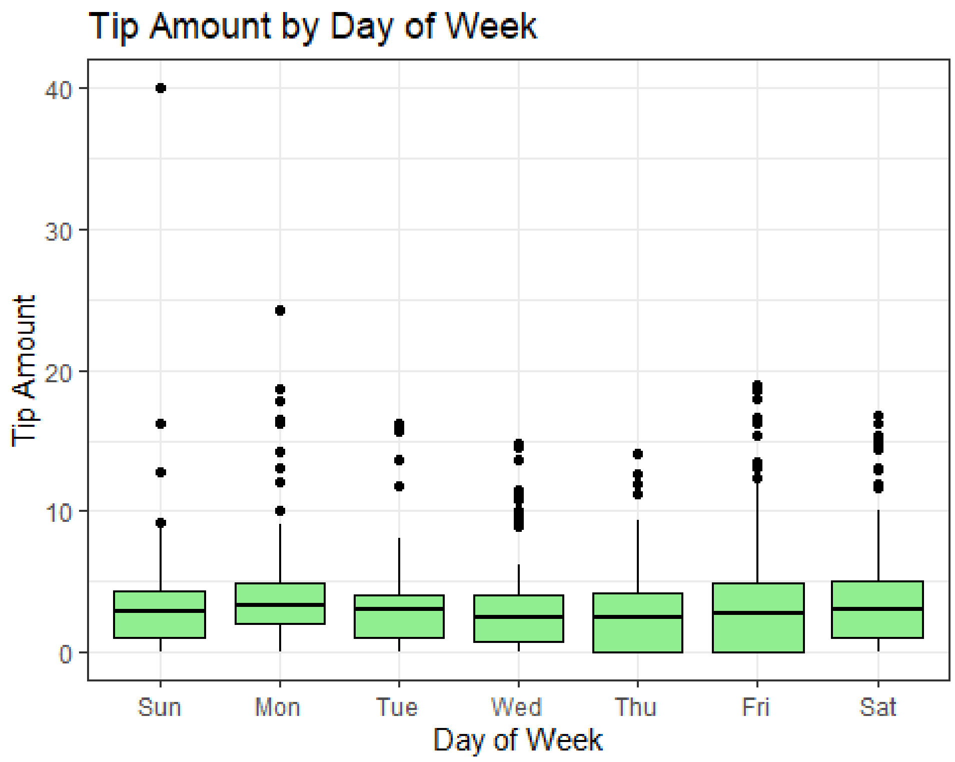
```
  # 10.Violin plot for days in week
ggplot(df, aes(x=day_of_week, y=trip_distance, fill = day_of_week)) + # Fill
by day of week
            geom_violin(alpha = 0.7) +
            labs(title="Trip Distance by Day of Week", x="Day of Week",
y="Trip Distance", fill = "Day of Week") +
            theme_bw() +
            scale_fill_brewer(palette="Set2")
```



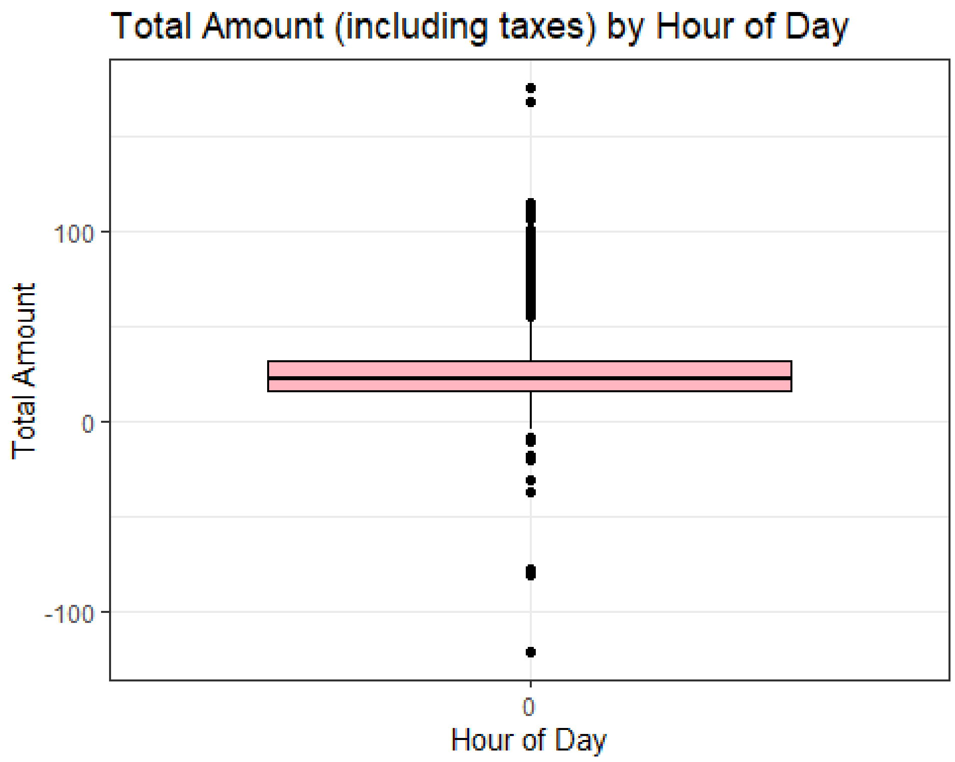
```
# 11. Boxplot of Tip Amount by Hour of Day
ggplot(df, aes(x = factor(pickup_hour), y = tip_amount)) + # factor to treat
hour as categorical
          geom_boxplot(fill = "skyblue", color = "black") +
          labs(title = "Tip Amount by Hour of Day", x = "Hour of Day", y =
"Tip Amount") +
          theme_bw()
```



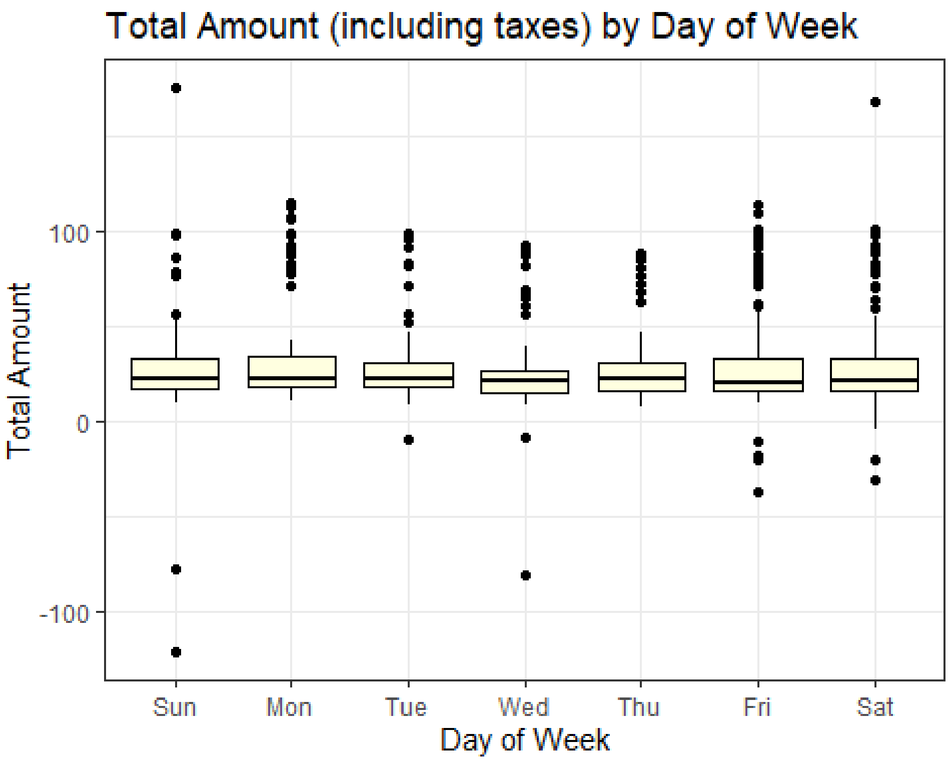
```r
# 12. Boxplot of Tip Amount by Day of Week
ggplot(df, aes(x = day_of_week, y = tip_amount)) +
          geom_boxplot(fill = "lightgreen", color = "black") +
          labs(title = "Tip Amount by Day of Week", x = "Day of Week", y =
"Tip Amount") +
          theme_bw()
```



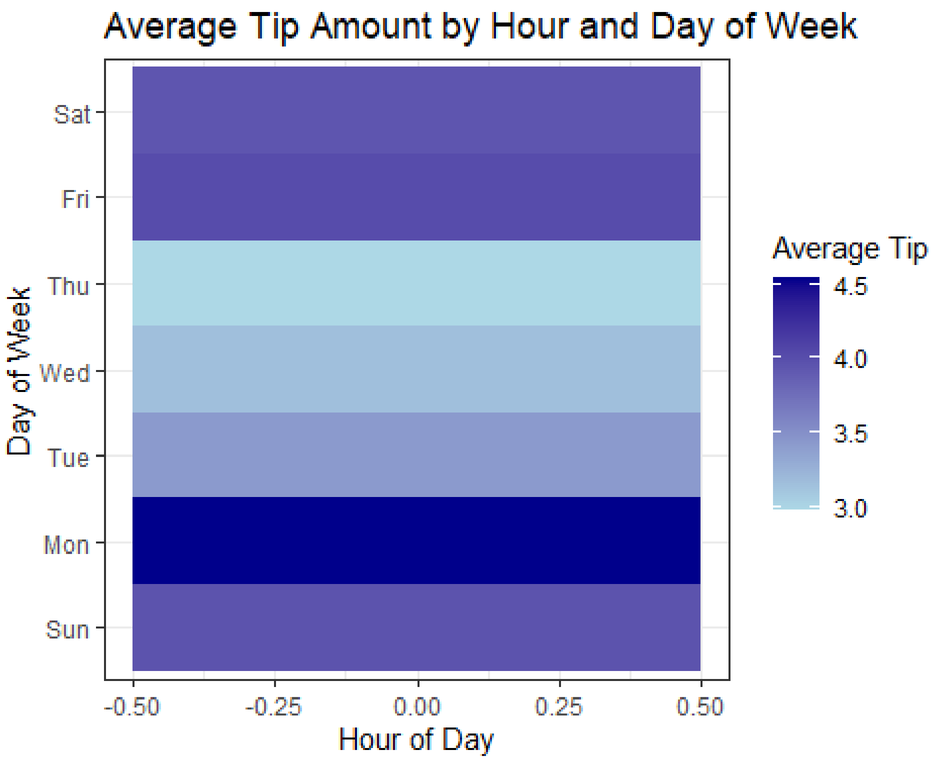
```r
# 13. Boxplot of Total Amount (including taxes) by Hour of Day
ggplot(df, aes(x = factor(pickup_hour), y = total_amount)) +
          geom_boxplot(fill = "lightpink", color = "black") +
          labs(title = "Total Amount (including taxes) by Hour of Day", x =
"Hour of Day", y = "Total Amount") +
          theme_bw()
```



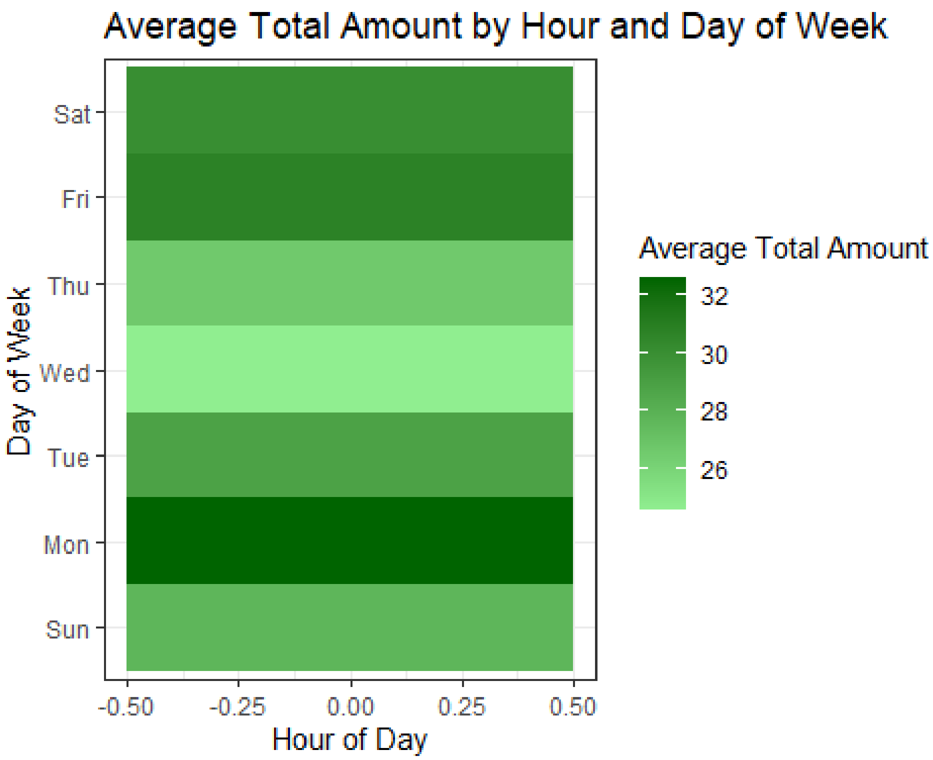
```
# 14. Boxplot of Total Amount (including taxes) by Day of Week
ggplot(df, aes(x = day_of_week, y = total_amount)) +
          geom_boxplot(fill = "lightyellow", color = "black") +
          labs(title = "Total Amount (including taxes) by Day of Week", x =
"Day of Week", y = "Total Amount") +
          theme_bw()
```



```
# 15. Heatmap of Average Tip Amount by Hour and Day of Week
tip_heatmap_data <- df %>%
    group_by(day_of_week, pickup_hour) %>%
    summarize(avg_tip = mean(tip_amount, na.rm = TRUE), .groups = "drop")

ggplot(tip_heatmap_data, aes(x = pickup_hour, y = day_of_week, fill =
avg_tip)) +
          geom_tile() +
          scale_fill_gradient(low = "lightblue", high = "darkblue", na.value
= "lightgrey") +

          labs(title = "Average Tip Amount by Hour and Day of Week", x =
"Hour of Day", y = "Day of Week", fill = "Average Tip") +
          theme_bw()
```



```r
# 16. Heatmap of Average Total Amount by Hour and Day of Week
total_heatmap_data <- df %>%
    group_by(day_of_week, pickup_hour) %>%
    summarize(avg_total = mean(total_amount, na.rm = TRUE), .groups = "drop")

ggplot(total_heatmap_data, aes(x = pickup_hour, y = day_of_week, fill =
avg_total)) +
          geom_tile() +
          scale_fill_gradient(low = "lightgreen", high = "darkgreen",
na.value = "lightgrey") +
          labs(title = "Average Total Amount by Hour and Day of Week", x =
"Hour of Day", y = "Day of Week", fill = "Average Total Amount") +
          theme_bw()
```


Average Total Amount by Hour and Day of Week
Sat
Fri
Thu
Wed
Tue
Mon
Sun
Day of Week
-0.50
-0.25
0.00
0.25
0.50
Hour of Day
Average Total Amount
32
30
28
26