NYC CitiBike Demand Prediction and Optimization

This project leverages geospatial data, chronological patterns, and machine learning techniques to predict demand and optimize bike allocation across CitiBike stations in New York City. The goal is to create a framework for real-time demand forecasting, station-level performance insights, and an optimized bike reallocation strategy that improves operational efficiency, revenue and user experience.

1. Data Exploration and Preprocessing

All of the data for this project was provided from the CitiBikenyc.com/system-data. The data set includes ride details such as start and end coordinates, timestamps, and membership type. These features were first preprocessed to calculate the distance between stations using the Haversine formula, and key time-series variables such as the day of the week and hour of the day were extracted. This enables us to analyze the ride patterns based on geographical and time factors.

A major finding from the exploration was that the dataset contains 2,164 unique start stations spread across four boroughs: Brooklyn (700 stations), Manhattan (694), Queens (451), and the Bronx (319).

Here are some quick facts on the data processed from 2023:

1. Brooklyn had the longest average ride distances **(1,993 meters)**, while the Bronx had the shortest **(1,501 meters)**.
2. The stations with the longest average ride distances were concentrated in the Bronx and Queens, such as the station at 46 Rd & 11 St in Queens with an average ride distance of **11,245 meters**.
3. The analysis also revealed that all boroughs have peak ridership around 4-5 PM, with Manhattan exhibiting the highest peak
4. Over 80% of all the rides were from members and not casual riders.
5. Only 10% of members rode electric bikes compared to classic bike types

These findings suggest that certain areas experience longer trips, likely due to their location in relation to popular destinations or transportation hubs.

2. Station-Level Insights

The next step involved aggregating the data by station to compute key metrics such as total rides, average distance, casual versus member ride counts, and weekend activity. This station-level analysis helps identify trends in station performance and informs the clustering process for demand prediction.

Clustering stations based on geographic location using K-means helped group similar stations, which provides insights into station performance based on factors like proximity to high-demand areas and ridership behavior.

3. Demand Prediction with XGBoost

To predict future demand at each station, an XGBoost regression model was trained on the processed dataset. Given the size of the CitiBike dataset, which includes several station-level attributes and ride details, using an efficient algorithm like XGBoost ensures that the model can be trained quickly without compromising performance. The model utilized variables like average ride distance, membership type, and temporal features (hour of the day) to predict the total number of rides at each station. A random 80/20 split of the data was used for training and testing, ensuring that the model was robust and unbiased.

The model achieved a Root Mean Square Error (RMSE) of 14.93, which was a significant improvement over the baseline model RMSE of 563.11. This indicates that the XGBoost model effectively captures demand patterns, with a substantial reduction in prediction error compared to a simple mean-based prediction.

4. Bike Reallocation Strategy

An essential outcome of this project was the formulation of a bike reallocation strategy. By analyzing stations with excess or insufficient demand, the model can recommend a redistribution of bikes to ensure optimal availability. For instance, stations with underperforming demand could receive bikes from high-demand stations, maintaining a balanced bike supply across the network. This strategy is crucial for CitiBike operators to ensure bikes are available where they are most needed, especially during peak hours.

5. Conclusions and recommendations

Utilizing the predicted bike usage per station, I was able to deduce the top 5 stations that are projected to have the highest monthly revenue. Not surprisingly, all of the stations were located within Manhattan. However, the interesting part is all of the stations were located at or near Central Park. The station with the highest projected revenue was Central Park South and 6th Avenue with a projection little over $2,700 dollars which is the highest out of all the stations looked at in the data.

Looking at a borough level, Manhattan as a whole has the highest average projected monthly revenue. Brooklyn is actually in second which would make a good argument to incorporate more stations within Manhattan. Instead of incorporating more stations in Brooklyn, you could reallocate electric bikes to the outer boroughs as they have higher average distances.

Lastly, I would recommend adding more electric bikes to the Central Park Stations. On average in Manhattan, casual riders use an electric bike 70% of the time, at Central Park South and 6th Ave casual riders use electric bikes only 61% of the time. This leads me to believe there are not enough electric bikes at that station and a possible loss of revenue as electric bikes generate more revenue.

This project provides a data-driven approach to enhancing CitiBike station performance through demand prediction and resource optimization. The insights gained from the station-level analysis and demand forecasting can be used to inform real-time operational decisions, improving the user experience and operational efficiency. The bike reallocation strategy, based on predicted demand, is a key step towards achieving a more responsive and efficient bike-sharing system.

Below is the full code used for this project as well as visuals to help explain the data.

CitiBike-Machine-Learning-Project.R

wills

2024-11-30

```r
knitr::opts_chunk$set(warning = FALSE, message = FALSE)
```

NYC Citibike Demand Project

This project focuses on analyzing CitiBike station performance using geospatial data, temporal patterns, and machine learning techniques to predict demand and optimize bike allocation. The main objectives include:

1. Data Processing:

The dataset includes ride details such as start and end coordinates, timestamps, and membership type. Distance between stations is calculated using the Haversine formula, day of the week and hour of the day are extracted.

2. Station Level Insights:

Key metrics, including total rides, average distance, casual vs. member ride counts, and weekend activity, are computed for each station. Stations are grouped by clusters based on geographic location using K-means clustering.

3. Demand Prediction:

Using an XGBoost model, predicted demand at each station is calculated. These predictions are compared within clusters to identify underperforming and high-demand stations.

4. Bike Reallocation Strategy:

Stations with excess or insufficient demand are identified, and a reallocation strategy is proposed by calculating the number of bikes to move or receive, ensuring a balance in bike availability.

5. Decision:

The project delivers a framework for real-time demand prediction and station optimization, empowering CitiBike operators to enhance user experience through efficient resource management.

```r
library(geosphere)
library(ggplot2)
library(caret)
library(xgboost)
library(lubridate)
library(sf)
library(scales)
library(stringr)
library(dplyr)
```

1-2. Data Exploration and Cleansing

```r
# Load the sample Citibike data for 2023
DF <- read.csv('sample_citibike_2023.csv')

# Calculate the distance between start and end stations using the Haversine formula
DF <- DF %>% mutate(
  distance = distHaversine(
    matrix(c(start_lng, start_lat), ncol = 2),  # Starting coordinates
    matrix(c(end_lng, end_lat), ncol = 2)      # Ending coordinates
  )
) %>%
  mutate(started_at = as.POSIXct(started_at, format = '%m/%d/%Y %H:%M'),
         Hour = hour(started_at)

  )

# Process and summarize the Citibike data to create station-level insights
station_data <- DF %>%
  # Convert the start time to a POSIXct datetime format for easier manipulation
  mutate(started_at = as.POSIXct(started_at, format = '%m/%d/%Y %H:%M')) %>%
  # Extract the day of the week and hour of the day from the start time
  mutate(
    day_of_week = weekdays(started_at), # Day of the week
    hour_of_day = hour(started_at)      # Hour of the day (0-23)
  ) %>%
  # Group the data by the start station ID to compute station-level statistics
  group_by(start_station_id) %>%
  # Summarize data for each station
  summarise(
    total_rides = n(),                            # Total # of rides at start station
    avg_distance = mean(distance, na.rm = TRUE),  # Average dist. of rides starting here
    casual_rides = sum(member_casual == "casual"), # Total rides by casual users
    member_rides = sum(member_casual == "member"), # Total rides by members
    peak_hour = which.max(table(hour_of_day)),    # The hour of day with the most rides
    rides_weekend = sum(day_of_week %in% c("Saturday", "Sunday")), # Rides on weekends
    start_lat = mean(start_lat, na.rm = TRUE),    # Average latitude of start station
    start_lng = mean(start_lng, na.rm = TRUE)     # Average longitude of start station
  )

## Total Stations
DF %>% distinct(start_station_id) %>% nrow()
```

```
## [1] 2164
```

```r
## Stations by Borough
DF %>% distinct(start_station_id, boro_name) %>% group_by(boro_name) %>% count(boro_name)
```

```
## # A tibble: 4 x 2
## # Groups:   boro_name [4]
##   boro_name     n
##   <chr>     <int>
## 1 Bronx       319
## 2 Brooklyn    700
## 3 Manhattan   694
## 4 Queens      451
```

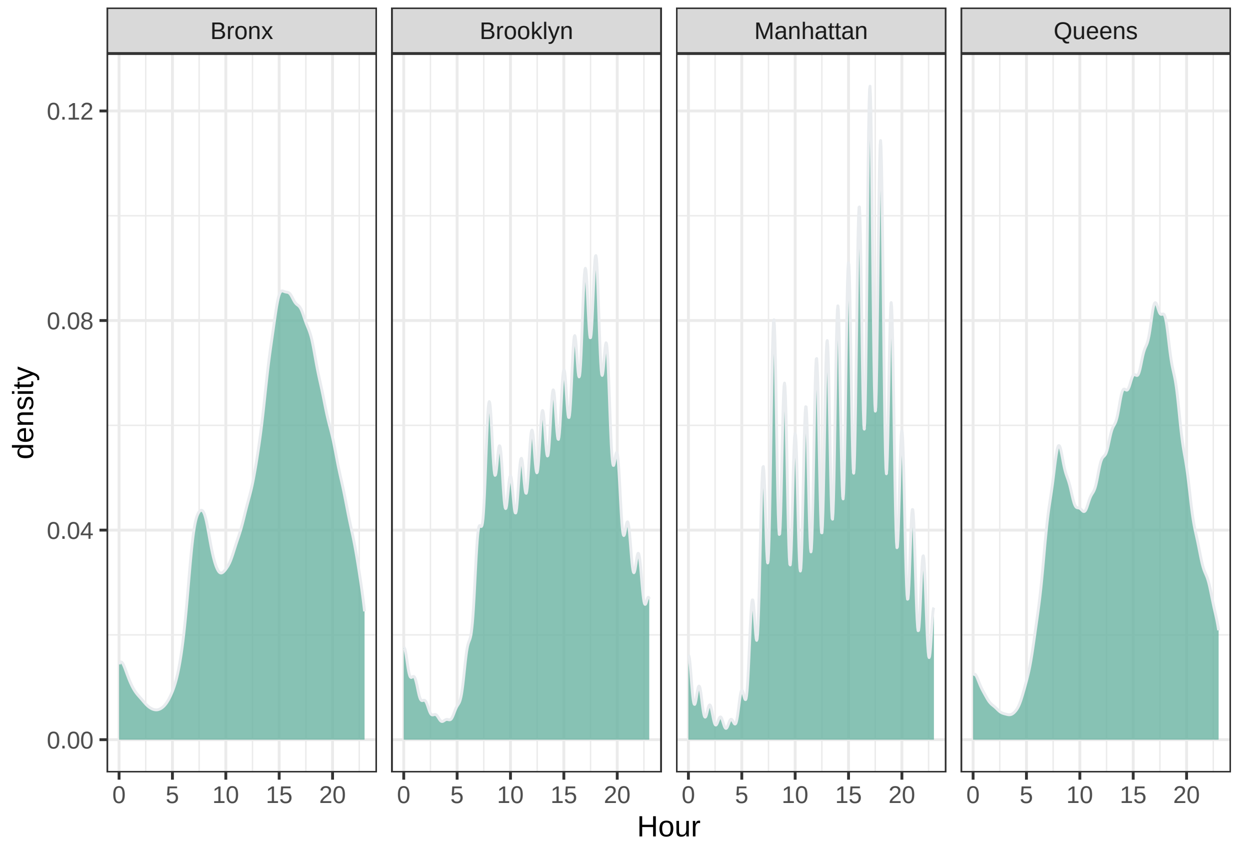
```
# Average ride distance by Borough
DF %>% group_by(boro_name) %>% summarize(distance = mean(distance, na.rm = T))
```

```
## # A tibble: 4 x 2
##   boro_name distance
##   <chr>        <dbl>
## 1 Bronx        1501.
## 2 Brooklyn     1993.
## 3 Manhattan    1853.
## 4 Queens       1799.
```

```
# Top 5 average distance by station
T5 <- DF %>% group_by(start_station_id) %>%
  summarize(distance = mean(distance, na.rm = T)) %>% ungroup() %>%
  arrange(desc(distance)) %>% slice(., 1:5)

DF %>% distinct(start_station_id, start_station_name, boro_name) %>%
  filter(., start_station_id %in% c(T5$start_station_id)) %>%
  left_join(., T5, by = c('start_station_id'))
```

```
##   start_station_id     start_station_name boro_name  distance
## 1          8485.01   E 188 St & Hughes Ave     Bronx  5701.524
## 2          8897.05 W 238 St & Tibbett Ave     Bronx  6045.693
## 3          5995.07 56 Ave & Junction Blvd    Queens  5052.838
## 4          6245.08           46 Rd & 11 St    Queens 11245.359
## 5          8315.03   3 Ave & E Tremont Ave     Bronx  5871.726
```

```
ggplot(DF, aes(x = Hour))+
  geom_density(fill = "#69b3a2", color="#e9ecef", alpha=0.8)+
  facet_grid(~boro_name)+
  theme_bw()
```



```
# All peak mainly around 4:00 to 5:00 PM
# Manhattan has highest peak of all
```

Data Exploration Findings: The dataset contains a total of 2,164 unique start stations. These stations are spread across four boroughs, with Brooklyn having the most stations (700), followed by Manhattan (694), Queens (451), and the Bronx (319). When examining the average ride distance by borough, Brooklyn has the longest average ride distance at 1,993 meters, followed by Manhattan at 1,853 meters, Queens at 1,799 meters, and the Bronx with the shortest average at 1,501 meters.

Additionally, the top five stations with the longest average ride distances are concentrated in the Bronx and Queens. The station at 46 Rd & 11 St in Queens has the longest average distance at 11,245 meters, followed by W 238 St & Tibbett Ave in the Bronx with 6,046 meters, and E 188 St & Hughes Ave in the Bronx at 5,702 meters. Other notable stations include 3 Ave & E Tremont Ave in the Bronx (5,872 meters) and 56 Ave & Junction Blvd in Queens (5,053 meters). These stations stand out with significantly higher average ride distances compared to other stations in the dataset.

3. Training the Model

```
# Set a random seed for reproducibility
set.seed(123)

# Split the data into training (80%) and testing (20%) sets using random sampling
train_idx <- sample(1:nrow(station_data), size = 0.8 * nrow(station_data))
train_data <- station_data[train_idx, ] %>% ungroup()
test_data <- station_data[-train_idx, ] %>% ungroup()

# Prepare the data for the model by converting the data frames into matrices
```

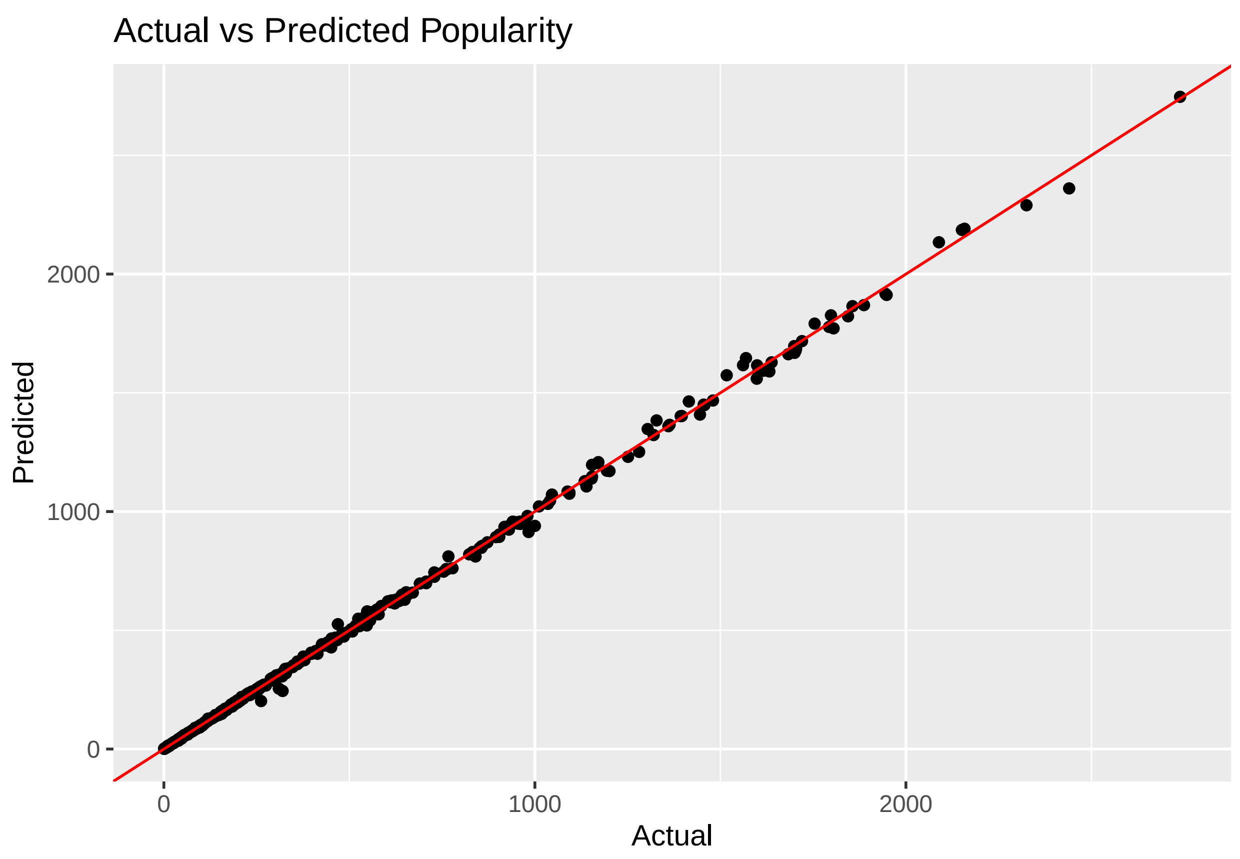
```r
train_matrix <- as.matrix(train_data %>% select(-start_station_id, -total_rides,
                                                -start_lat, -start_lng))
test_matrix <- as.matrix(test_data %>% select(-start_station_id, -total_rides,
                                              -start_lat, -start_lng))

# Convert the training and testing matrices into the format required by XGBoost
dtrain <- xgb.DMatrix(data = train_matrix, label = train_data$total_rides)
dtest <- xgb.DMatrix(data = test_matrix, label = test_data$total_rides)

# Train the model
xgb_model <- xgboost(data = dtrain, max_depth = 6, eta = 0.3, nrounds = 100,
                     objective = "reg:squarederror", verbose = 0)

# Makes the predictions from the testing matrix
y_pred <- predict(xgb_model, dtest)

# Calculates the RMSE
rmse <- sqrt(mean((y_pred - test_data$total_rides)^2))
print(paste("RMSE:", rmse))
```

```
## [1] "RMSE: 14.9271616506604"
```

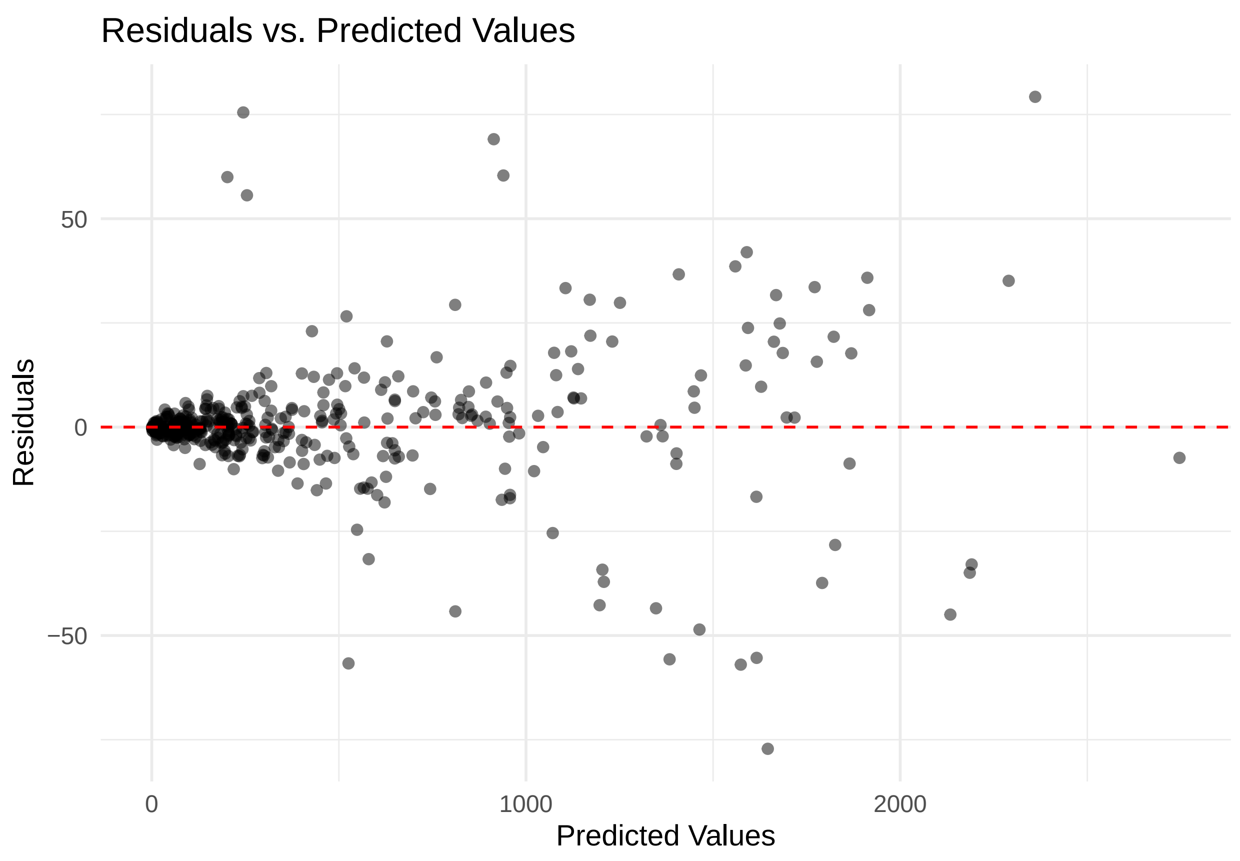
```r
# Calculate the mean of the target variable (total rides)
baseline_mean <- mean(station_data$total_rides, na.rm = TRUE)

# Calculate RMSE for the baseline model
baseline_rmse <- sqrt(mean((station_data$total_rides - baseline_mean)^2,
                           na.rm = TRUE))
print(paste("Baseline RMSE:", round(baseline_rmse, 2)))
```

```
## [1] "Baseline RMSE: 563.11"
```

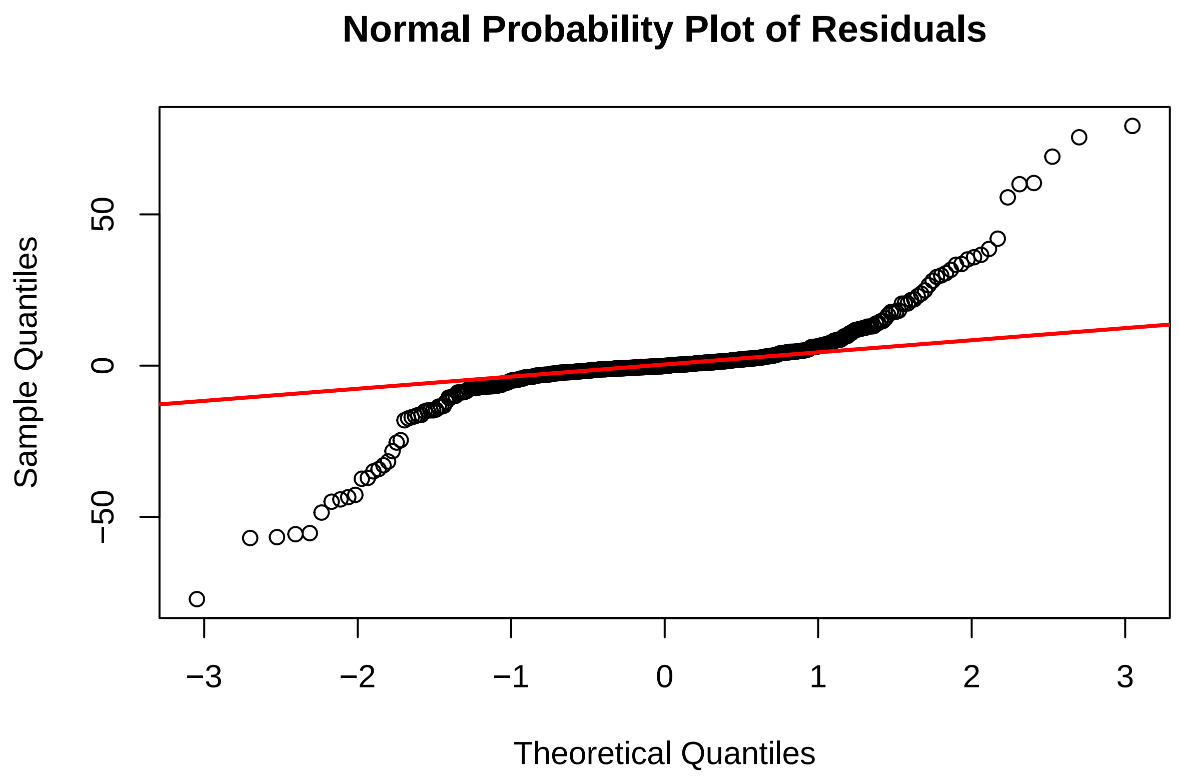
```r
ggplot(data.frame(actual = test_data$total_rides, predicted = y_pred),
       aes(x = actual, y = predicted)) +
  geom_point() +
  geom_abline(color = "red") +
  labs(title = "Actual vs Predicted Popularity", x = "Actual",
       y = "Predicted")
```



```
# Residual calculation
residuals <- test_data$total_rides - y_pred

# Residual plot
ggplot(data = test_data, aes(x = y_pred, y = residuals)) +
  geom_point(alpha = 0.5) +
  geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
  labs(title = "Residuals vs. Predicted Values",
       x = "Predicted Values",
       y = "Residuals") +
  theme_minimal()
```



```r
qqnorm(residuals, main = "Normal Probability Plot of Residuals")
qqline(residuals, col = "red", lwd = 2)
```



```r
# Residuals vs Fitted Values and NPP display some assumptions may be violated
# trying to remove the outliers to see if this fixes the problem

# Calculate the mean and standard deviation of the average distance
mean_distance <- mean(station_data$avg_distance)
sd_distance <- sd(station_data$avg_distance)

# Calculate Z-scores to identify outliers
station_data$z_score <- (station_data$avg_distance - mean_distance) / sd_distance

# Filter the data to exclude outliers
clean_data <- station_data %>%
  filter(abs(z_score) <= 3)

# Perform K-means clustering on cleaned data based on latitude and longitude
kmeans_result <- kmeans(
  clean_data %>% select(start_lat, start_lng), # Features for clustering
  centers = 5                                 # Number of clusters
)

# Assign cluster labels to the data
clean_data$cluster <- kmeans_result$cluster

# Split the data into training and testing sets (80-20 split)
train_idx <- sample(1:nrow(clean_data), size = 0.8 * nrow(clean_data))
train_data <- clean_data[train_idx, ] %>% ungroup() # Training data
```

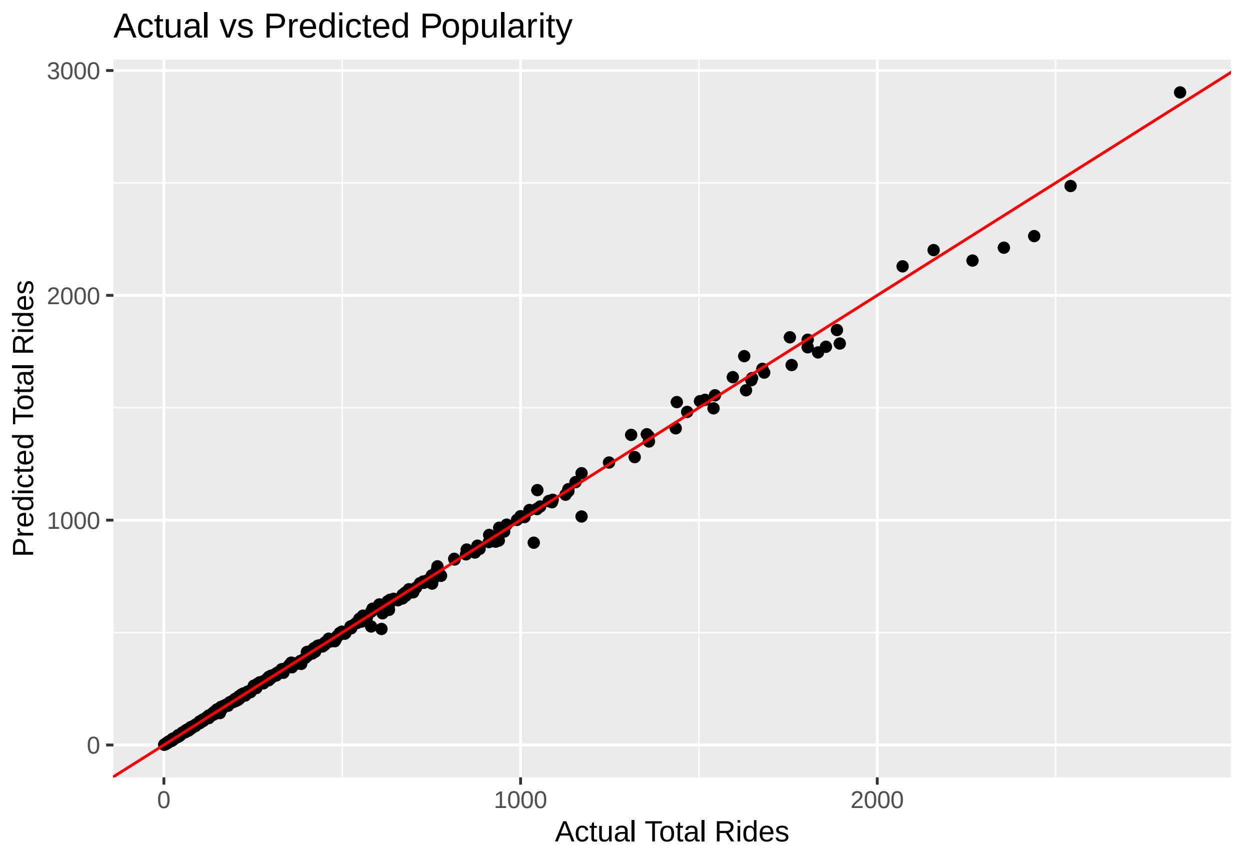
```r
test_data <- clean_data[-train_idx, ] %>% ungroup() # Testing data

# Prepare training and testing matrices, excluding non-relevant columns
train_matrix <- as.matrix(train_data %>%
  select(-start_station_id, -total_rides, -start_lat, -start_lng, -z_score))
test_matrix <- as.matrix(test_data %>%
  select(-start_station_id, -total_rides, -start_lat, -start_lng, -z_score))

# Convert training and testing data into DMatrix format for XGBoost
dtrain <- xgb.DMatrix(data = train_matrix, label = train_data$total_rides)
dtest <- xgb.DMatrix(data = test_matrix, label = test_data$total_rides)

# Train an XGBoost regression model
xgb_model_2 <- xgboost(
  data = dtrain,
  max_depth = 6,
  eta = 0.3,
  nrounds = 100,
  objective = "reg:squarederror",
  verbose = 0
)

# Generate predictions for the testing set
y_pred <- predict(xgb_model_2, dtest)

# Calculate the Root Mean Square Error for the model
rmse <- sqrt(mean((y_pred - test_data$total_rides)^2))
print(paste("RMSE:", rmse))
```

```
## [1] "RMSE: 23.4553553474725"
```

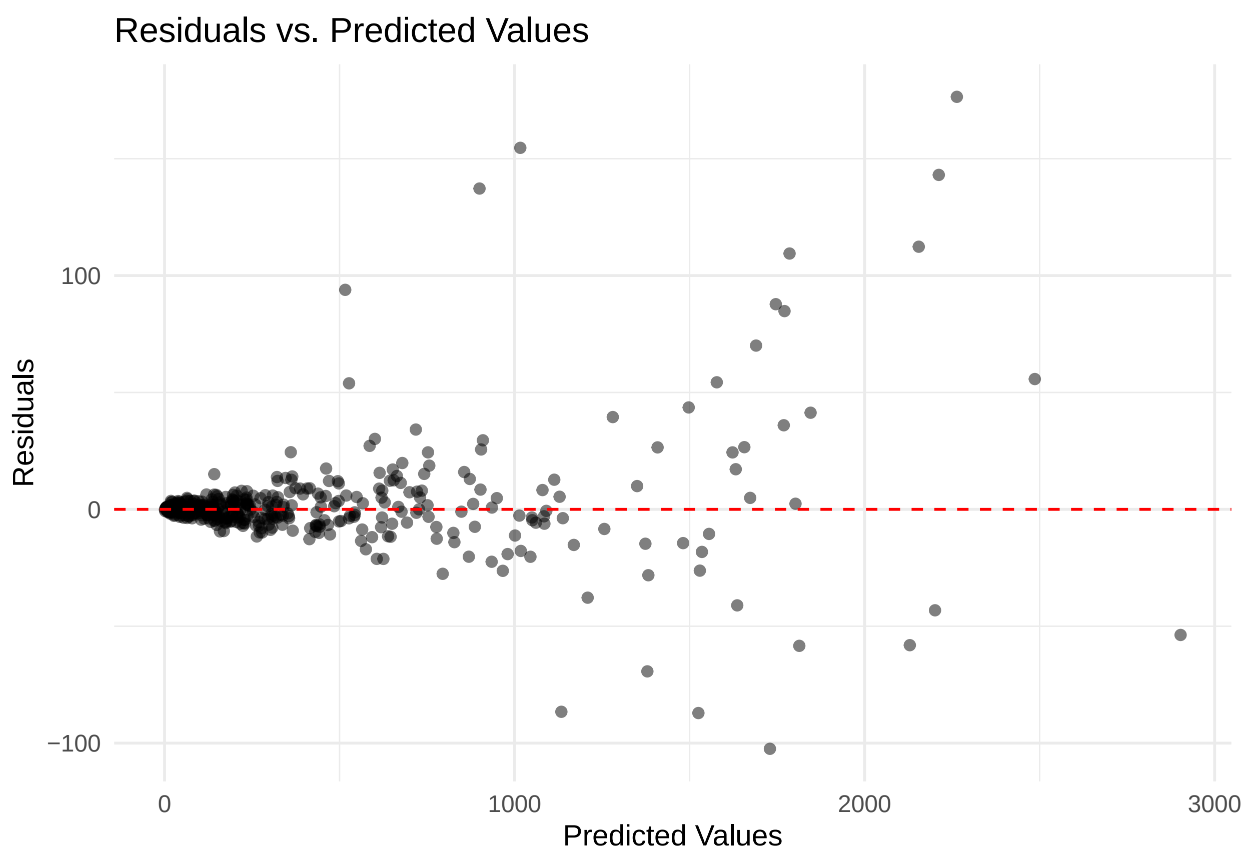
```r
# Calculate the baseline RMSE using the mean as a simple prediction
baseline_mean <- mean(station_data$total_rides, na.rm = TRUE)
baseline_rmse <- sqrt(mean((station_data$total_rides - baseline_mean)^2,
                           na.rm = TRUE))
print(paste("Baseline RMSE:", round(baseline_rmse, 2)))
```

```
## [1] "Baseline RMSE: 563.11"
```

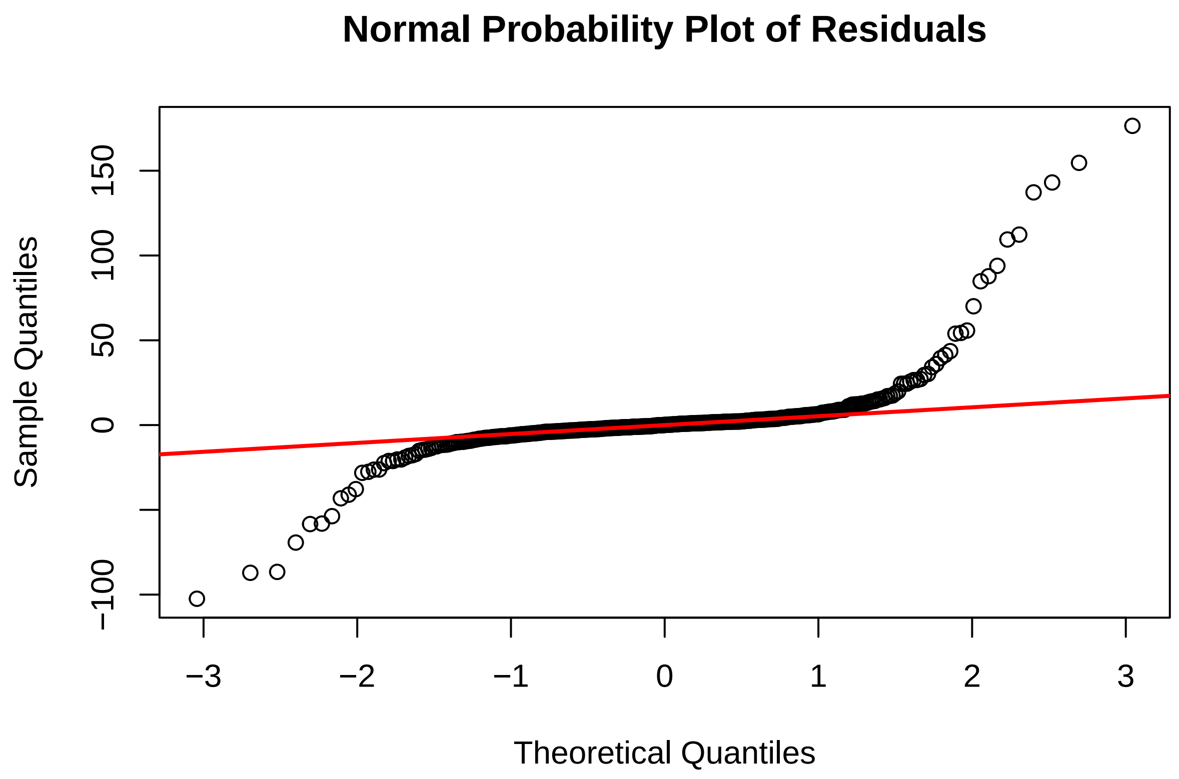
```r
# Scatter plot of actual vs. predicted values
ggplot(data.frame(actual = test_data$total_rides, predicted = y_pred),
       aes(x = actual, y = predicted)) +
  geom_point() +
  geom_abline(color = "red") +
  labs(
    title = "Actual vs Predicted Popularity",
    x = "Actual Total Rides",
    y = "Predicted Total Rides"
  )
```



```r
# Calculate residuals (differences between actual and predicted values)
residuals <- test_data$total_rides - y_pred

# Plot residuals vs. predicted values
ggplot(data = test_data, aes(x = y_pred, y = residuals)) +
  geom_point(alpha = 0.5) +
  geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
  labs(
    title = "Residuals vs. Predicted Values",
    x = "Predicted Values",
    y = "Residuals"
  ) +
  theme_minimal()
```



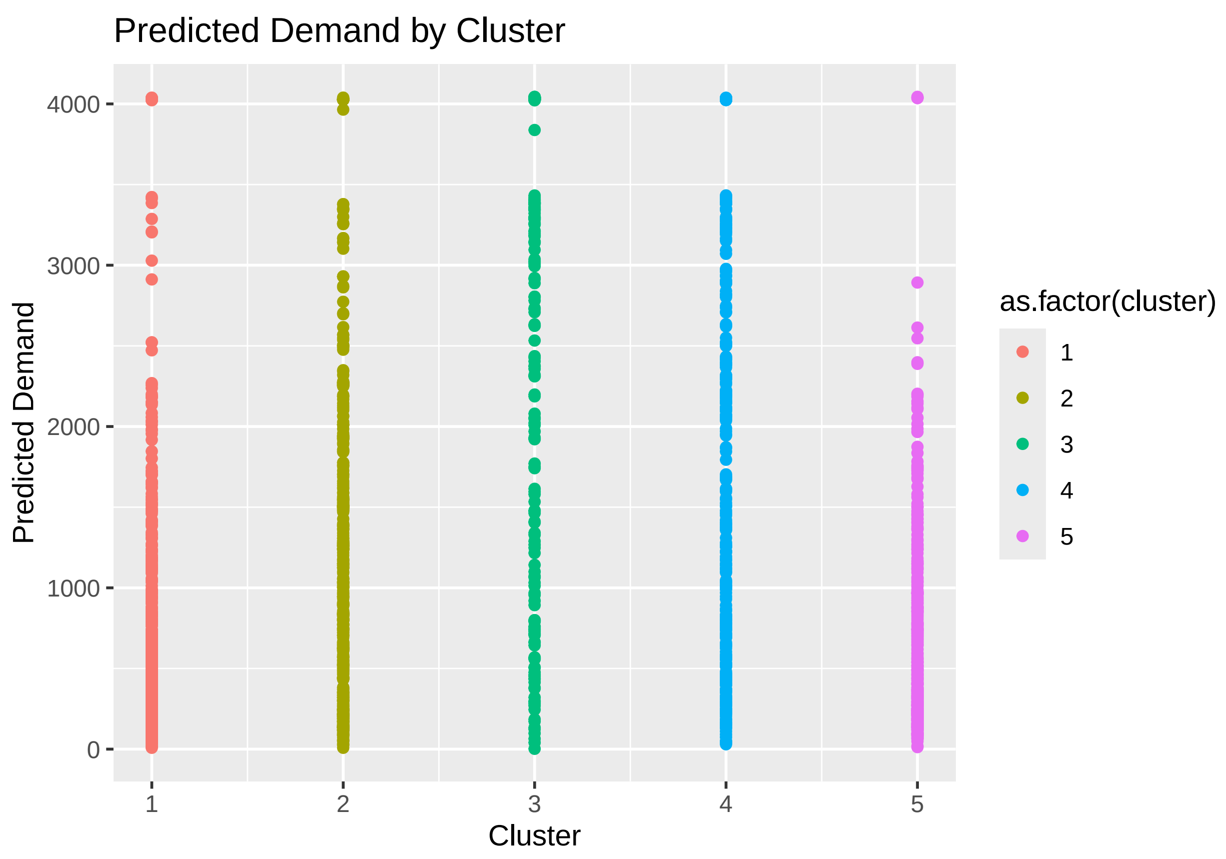
```r
# Normal probability plot of residuals
qqnorm(residuals, main = "Normal Probability Plot of Residuals")
qqline(residuals, col = "red", lwd = 2)
```



```r
##--New Data for Predicting 2024 Demand
DF2 <- read.csv('CitiBike_NewData.csv')

# Calculate distance between start and end stations
DF2 <- DF2 %>%
  mutate(distance = distHaversine(matrix(c(start_lng, start_lat), ncol = 2),
                                  matrix(c(end_lng, end_lat), ncol = 2)))

# Read borough boundaries
borough_geojson <- st_read("Borough Boundaries.geojson", quiet = TRUE)

# Assign borough to stations
stations_boros <- DF2 %>%
  group_by(start_station_id) %>%
  summarise(longitude = mean(start_lng, na.rm = TRUE),
            latitude = mean(start_lat, na.rm = TRUE))

stations_sf <- st_as_sf(stations_boros, coords = c("longitude", "latitude"),
                        crs = 4326)
stations_with_borough <- st_join(stations_sf, borough_geojson) %>%
  select(start_station_id, boro_name)

DF2 <- DF2 %>%
  left_join(stations_with_borough, by = 'start_station_id')

station_data <- DF2 %>%
  mutate(started_at = as.POSIXct(started_at, format = '%Y-%M-%d %H:%M:%OS')) %>%
  mutate(day_of_week = weekdays(started_at), hour_of_day = hour(started_at)) %>%
  group_by(start_station_id, start_station_name, boro_name) %>%
  summarise(
    total_rides = n(),
    avg_distance = mean(distance, na.rm = TRUE),
    casual_rides = sum(member_casual == "casual"),
    member_rides = sum(member_casual == "member"),
    peak_hour = which.max(table(hour_of_day)),
    rides_weekend = sum(day_of_week %in% c("Saturday", "Sunday")),
    start_lat = mean(start_lat, na.rm = TRUE),
    start_lng = mean(start_lng, na.rm = TRUE)
  ) %>% ungroup()
```

4. Bike Reallocation

```r
# Perform K-means clustering
kmeans_result <- kmeans(station_data %>% select(start_lat, start_lng),
                        centers = 5)
station_data$cluster <- kmeans_result$cluster

# Prepare data matrix for prediction
new_matrix <- as.matrix(station_data %>% select(-start_station_id, -total_rides,
                                                -start_lat, -start_lng,
                                                -start_station_name,
                                                -boro_name))
dtest_new <- xgb.DMatrix(data = new_matrix, label = station_data$total_rides)

# Predict demand using pre-trained model
predicted_demand <- predict(xgb_model_2, dtest_new)
station_data$predicted_demand <- predicted_demand

##-Post Prediction Analysis

# Visualize demand by cluster
ggplot(station_data, aes(x = cluster, y = predicted_demand,
                         color = as.factor(cluster))) +
  geom_point() +
  labs(title = "Predicted Demand by Cluster", x = "Cluster", y = "Predicted Demand")
```



```r
# Calculate average demand by cluster
cluster_avg_demand <- station_data %>%
  group_by(cluster) %>%
  summarise(cluster_avg = mean(predicted_demand))

# Identify underperforming and high-demand stations
underperforming_stations <- station_data %>%
  left_join(cluster_avg_demand, by = 'cluster') %>%
  filter(predicted_demand < cluster_avg) %>%
  mutate(bike_to_move = cluster_avg - predicted_demand)

# Underperforming stations by borough
underperforming_stations %>% group_by(boro_name) %>%
  count(boro_name)
```

```
## # A tibble: 4 x 2
## # Groups:   boro_name [4]
##   boro_name     n
##   <chr>     <int>
## 1 Bronx       278
## 2 Brooklyn    379
## 3 Manhattan   203
## 4 Queens      314
```

```r
# Top 5 underperforming stations
underperforming_stations %>% arrange(predicted_demand) %>%
  slice(1:5) %>% select(start_station_name,
                         predicted_demand)
```

```
## # A tibble: 5 x 2
##   start_station_name        predicted_demand
##   <chr>                                <dbl>
## 1 Pier 40 X2                            2.04
## 2 E 34 St & Linden Blvd                 8.37
## 3 W 211 St & 10 Ave                     8.38
## 4 Gillmore St & 27 Ave                 12.5
## 5 Dawson St & Intervale Ave            13.1
```

```
high_demand_stations <- station_data %>%
  left_join(cluster_avg_demand, by = 'cluster') %>%
  filter(predicted_demand > cluster_avg) %>%
  mutate(bike_to_receive = predicted_demand - cluster_avg)

# High demand stations by borough
high_demand_stations %>% group_by(boro_name) %>%
  count(boro_name)
```

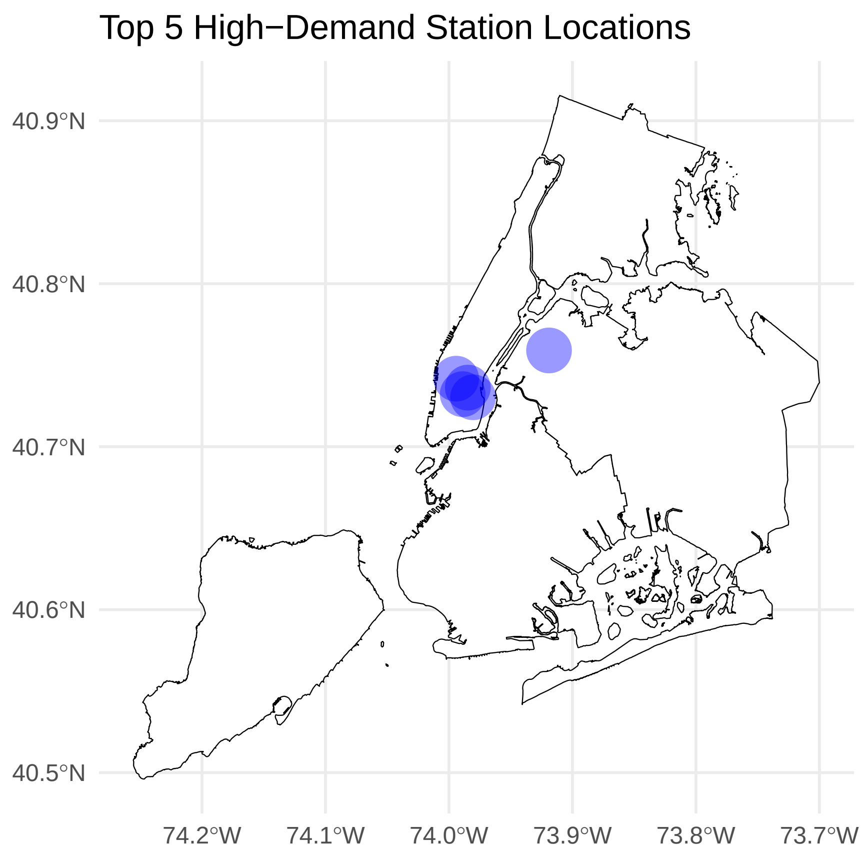
```
## # A tibble: 4 x 2
## # Groups:   boro_name [4]
##   boro_name     n
##   <chr>     <int>
## 1 Bronx        55
## 2 Brooklyn    348
## 3 Manhattan   508
## 4 Queens      149
```

```
# Top 5 high-demand stations
top_5_predicted_stations <- high_demand_stations %>%
  arrange(desc(predicted_demand)) %>%
  slice(1:5)

top_5_predicted_stations %>% select(start_station_name, boro_name,
                                    predicted_demand)
```

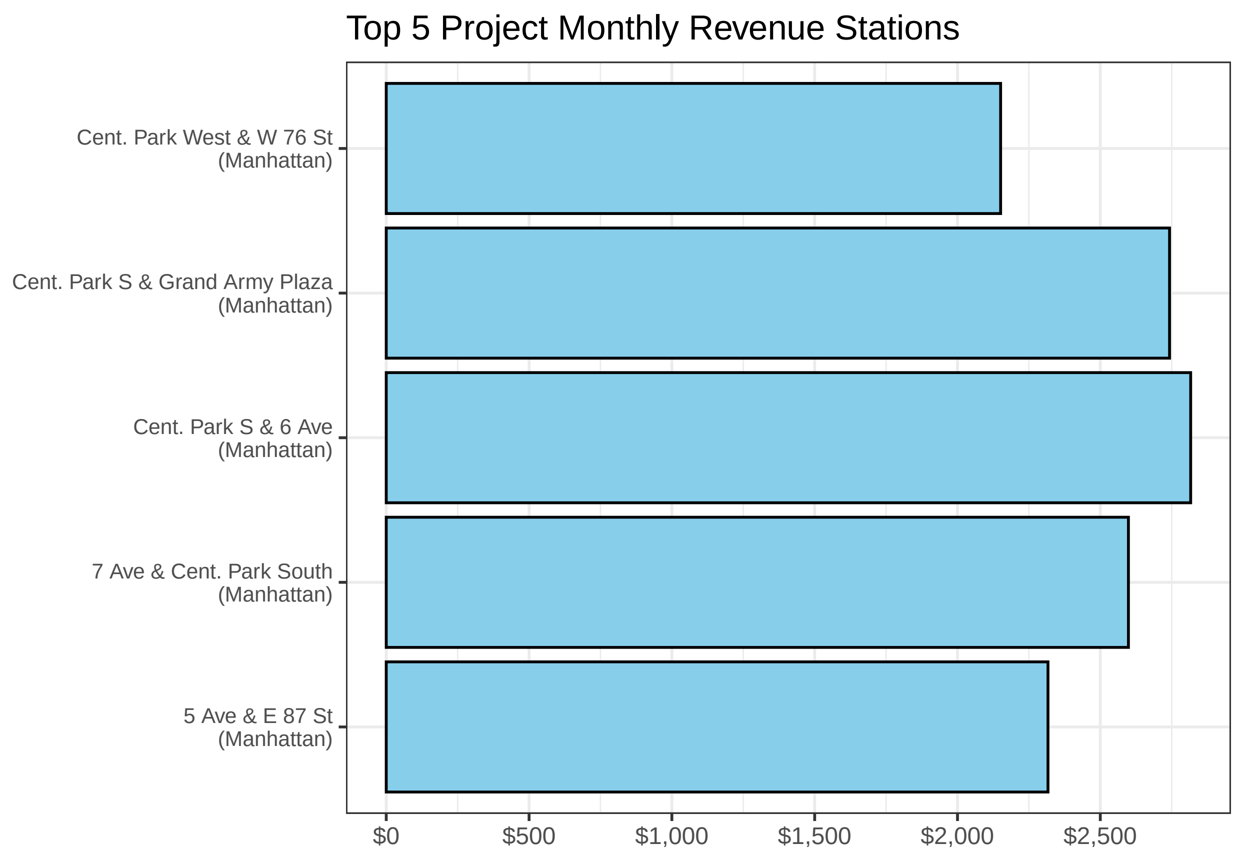
```
## # A tibble: 5 x 3
##   start_station_name     boro_name predicted_demand
##   <chr>                  <chr>                <dbl>
## 1 Steinway St & Broadway Queens               4044.
## 2 E 19 St & 3 Ave        Manhattan            4044.
## 3 W 21 St & 6 Ave        Manhattan            4043.
## 4 E 12 St & 3 Ave        Manhattan            4043.
## 5 Ave A & E 14 St        Manhattan            4043.
```

```
high_demand_locations <- DF2 %>%
  filter(., start_station_id %in% top_5_predicted_stations$start_station_id &
           start_lat %in% c(top_5_predicted_stations$start_lat)) %>%
  distinct(start_station_id, start_lng, start_lat)

stations_sf <- st_as_sf(high_demand_locations,
                        coords = c("start_lng", "start_lat"), crs = 4326)
stations_sf$density <- sapply(stations_sf$geometry, function(point) {
  sum(st_distance(st_sfc(point, crs = st_crs(stations_sf)),
                  stations_sf$geometry) < units::set_units(500, "meters"))
```

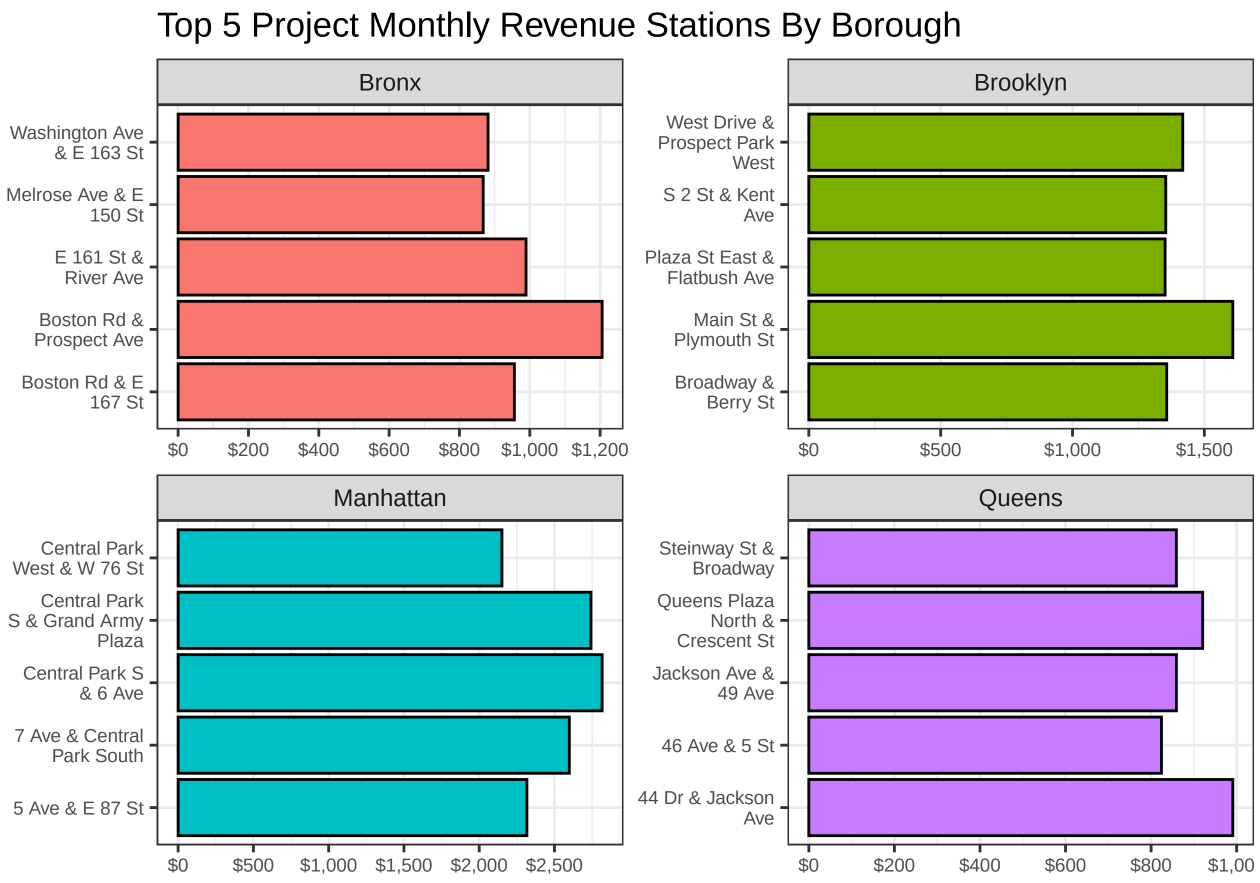
```r
})

# Plot top 5 stations on map
ggplot() +
  geom_sf(data = borough_geojson, fill = 'white', color = "black") +
  geom_sf(data = stations_sf, aes(size = density), color = "blue", alpha = 0.4) +
  scale_size_continuous(range = c(1, 10)) +
  theme_minimal() +
  labs(title = "Top 5 High-Demand Station Locations")+
  theme(legend.position = 'none')
```



```r
# Cost Analysis
costs <- data.frame(member_casual = c('casual','casual', 'member', 'member'),
         rideable_type = c('electric_bike', 'classic_bike','electric_bike',
                           'classic_bike'),
         bike_unlocks = c(4.79, 4.79,0,0),
         cost_per_min = c(.36, 0, .24, 0))

DF2 <- DF2 %>% left_join(., costs, by = c('member_casual', 'rideable_type')) %>%
  mutate(ended_at = as.POSIXct(ended_at, format = '%Y-%m-%d %H:%M:%OS')) %>%
  mutate(total_time = as.numeric(difftime(ended_at, started_at, units = "mins"))) %>%
  mutate(cost = bike_unlocks+(cost_per_min*total_time))

predicted_revenue <- DF2 %>% group_by(start_station_id) %>%
  summarize(avg_cost = mean(cost, na.rm = T)) %>% ungroup() %>%
  left_join(station_data, ., by = c('start_station_id')) %>%
  mutate(predicted_revenue = predicted_demand*avg_cost)

predicted_revenue %>% arrange(desc(predicted_revenue)) %>%
  filter(., start_station_name != '') %>%
  slice(1:5) %>%
  mutate(start_station_name = str_replace(start_station_name, 'Central', 'Cent.')) %>%
  mutate(station_label = paste0(start_station_name, '\n', '(', boro_name,')')) %>%
  mutate(predicted_revenue_monthly = predicted_revenue/12) %>%
  ggplot()+
  geom_bar(aes(x = station_label, y = predicted_revenue_monthly),
           stat = 'identity', fill = 'skyblue', color = 'black')+
  scale_y_continuous(breaks = pretty_breaks(),
                     labels = label_dollar(prefix = "$", accuracy = 1))+
  theme_bw()+
  theme(axis.title = element_blank(),
        axis.text.y = element_text(size = 8))+
  coord_flip()+
  ggtitle('Top 5 Project Monthly Revenue Stations')
```



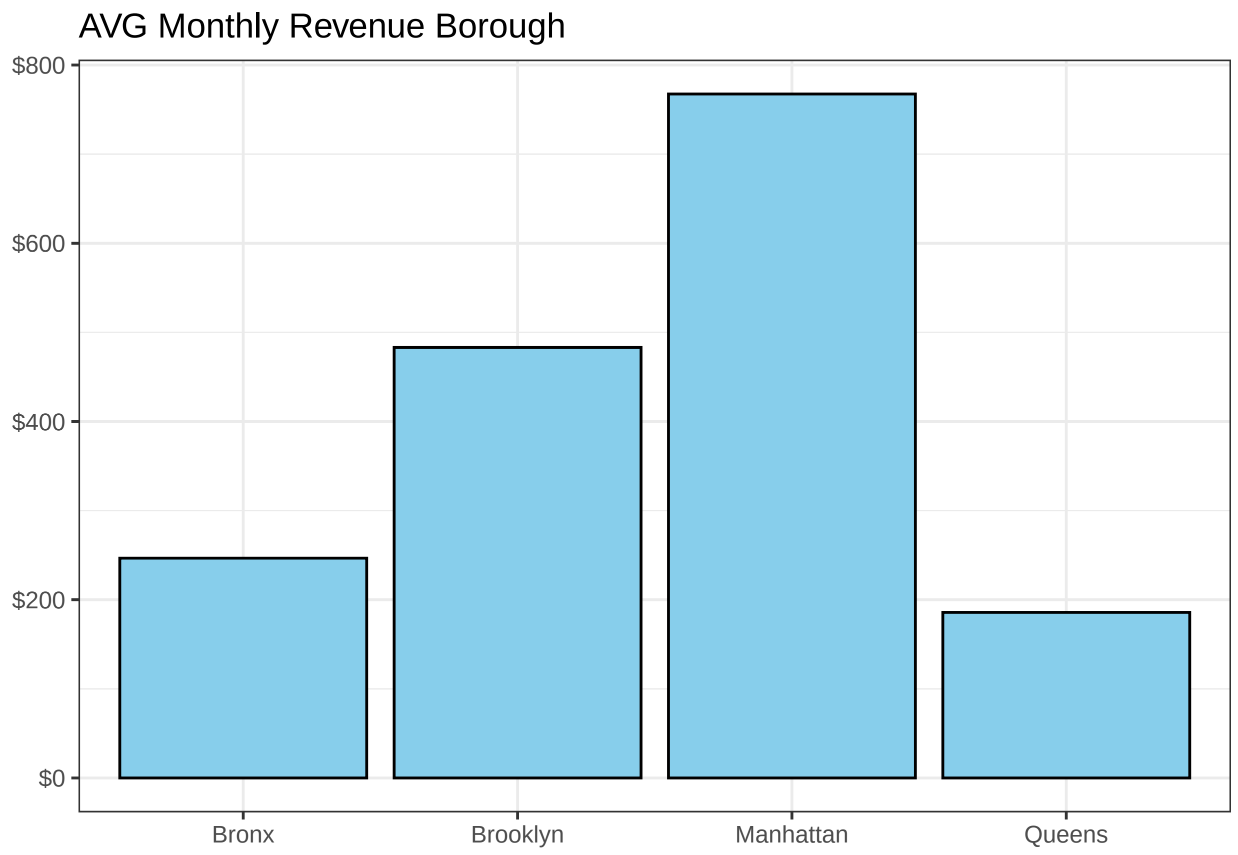
```
predicted_revenue %>% arrange(desc(predicted_revenue)) %>%
  filter(., start_station_name != '') %>%
  group_by(boro_name) %>%
  slice(1:5) %>%
  mutate(predicted_revenue_monthly = predicted_revenue/12) %>%
  mutate(start_station_name = str_wrap(start_station_name, width = 15)) %>%
  ggplot()+
  geom_bar(aes(x = start_station_name,
               y = predicted_revenue_monthly, fill = boro_name),
           stat = 'identity', color = 'black')+
  scale_y_continuous(breaks = pretty_breaks(),
                     labels = label_dollar(prefix = "$", accuracy = 1))+
  facet_wrap(~boro_name, ncol = 2, scales = "free")+
  theme_bw()+
  theme(axis.title = element_blank(), legend.position = 'none',
        axis.text = element_text(size = 7),
        plot.margin = margin(0,0,0,0, "cm"))+
  coord_flip()+
  ggtitle('Top 5 Project Monthly Revenue Stations By Borough')
```



```
predicted_revenue %>% arrange(desc(predicted_revenue)) %>%
  filter(., start_station_name != '') %>%
  group_by(boro_name) %>%
  slice(1:5) %>%
  mutate(predicted_revenue_monthly = predicted_revenue/12,
         predicted_revenue_monthly = label_dollar()(predicted_revenue_monthly)) %>%
  select(Station = start_station_name, Borough = boro_name,
         Monthly_Revenue_Dollars = predicted_revenue_monthly)
```

```
## # A tibble: 20 x 3
## # Groups:   Borough [4]
##    Station                                  Borough   Monthly_Revenue_Dollars
##    <chr>                                    <chr>     <chr>
##  1 Boston Rd & Prospect Ave          Bronx     $1,205.72
##  2 E 161 St & River Ave              Bronx     $989.40
##  3 Boston Rd & E 167 St              Bronx     $956.29
##  4 Washington Ave & E 163 St         Bronx     $881.42
##  5 Melrose Ave & E 150 St            Bronx     $867.62
##  6 Main St & Plymouth St             Brooklyn  $1,607.76
##  7 West Drive & Prospect Park West   Brooklyn  $1,418.01
##  8 Broadway & Berry St               Brooklyn  $1,357.18
##  9 S 2 St & Kent Ave                 Brooklyn  $1,353.53
## 10 Plaza St East & Flatbush Ave      Brooklyn  $1,351.09
## 11 Central Park S & 6 Ave            Manhattan $2,816.40
## 12 Central Park S & Grand Army Plaza Manhattan $2,742.45
## 13 7 Ave & Central Park South        Manhattan $2,598.55
## 14 5 Ave & E 87 St                   Manhattan $2,316.88
## 15 Central Park West & W 76 St       Manhattan $2,151.05
## 16 44 Dr & Jackson Ave               Queens    $990.96
## 17 Queens Plaza North & Crescent St  Queens    $920.52
## 18 Jackson Ave & 49 Ave              Queens    $859.18
## 19 Steinway St & Broadway            Queens    $859.09
## 20 46 Ave & 5 St                     Queens    $824.21
```

```
predicted_revenue %>% arrange(desc(predicted_revenue)) %>%
  filter(., start_station_name != '') %>%
  group_by(boro_name) %>%
  summarize(predicted_revenue_monthly = mean(predicted_revenue/12, na.rm = T)) %>%
  ungroup() %>%
  ggplot()+
  geom_bar(aes(x = boro_name, y = predicted_revenue_monthly),
           stat = 'identity', fill = 'skyblue', color = 'black')+
  scale_y_continuous(breaks = pretty_breaks(),
                     labels = label_dollar(prefix = "$", accuracy = 1))+
  theme_bw()+
  theme(axis.title = element_blank(), legend.position = 'none')+
  ggtitle('AVG Monthly Revenue Borough')
```



```
# Highest Predicted Avg Monthly Revenue Station
DF2 %>% filter(., start_station_name == 'Central Park S & 6 Ave') %>%
  group_by(member_casual, rideable_type) %>% count(rideable_type)
```

```
## # A tibble: 4 x 3
## # Groups:   member_casual, rideable_type [4]
##   member_casual rideable_type     n
##   <chr>         <chr>         <int>
## 1 casual        classic_bike   2834
## 2 casual        electric_bike  4535
## 3 member        classic_bike   2042
## 4 member        electric_bike  3618
```

```
# AVG Manhattan Station in October 2024
DF2 %>% filter(., boro_name == 'Manhattan') %>%
  group_by(start_station_id, member_casual, rideable_type) %>%
  count(rideable_type) %>% ungroup(start_station_id) %>%
  summarize(AVG = mean(n,na.rm = T))
```

```
## # A tibble: 4 x 3
## # Groups:   member_casual [2]
##   member_casual rideable_type   AVG
##   <chr>         <chr>         <dbl>
## 1 casual        classic_bike   249.
## 2 casual        electric_bike  597.
## 3 member        classic_bike  1375.
## 4 member        electric_bike 2379.
```