Case Study: Key Differences in use of Divvy Bikes by Casual Riders and Member Riders for the month of May, 2023

2023-07-1

Setting Up The Data

```r
# Loading required libraries
install.packages("ggmap")
```

Installing The Proper Libraries

```
## Warning: package 'ggmap' is in use and will not be installed
```

```r
library(ggmap)   # For accurate maps
library(tidyr)   # Because tidy-verse
```

```
## Warning: package 'tidyr' was built under R version 4.3.1
```

```r
library(dplyr)   # For data manipulation
```

```
## Warning: package 'dplyr' was built under R version 4.3.1

##
## 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(stringr) # For string ops
```

```
## Warning: package 'stringr' was built under R version 4.3.1
```

```r
library(ggplot2) # For data viz
library(png)     # For data viz
library(RColorBrewer)
```

```r
# Upload the data
# Original data source. See appendix for citation.

divvy_data_clean <- read.csv("202305-divvy-tripdata.csv")

# Used to acquire a consistent geo-location for the bikes when docked.
# See appendix for citation.
```

```r
divvy_bicycle_station_coords <- read.csv("Divvy_Bicycle_Stations.csv")
```

Uploading The Data Sets

```r
chicago_map <- get_map(location = "Chicago", zoom = 12)
ggsave("chicago_map.png", plot = ggmap(chicago_map), width = 8, height = 6, dpi = 300)
```

Retreiving Map For Visuals

Saving Addresses For Later Reference

```r
# New Data Frame to save a legible version of the Bike Station Locations

address <- divvy_bicycle_station_coords[c("Station.Name")]

# Colnames to lower case. Changing '.' to '_', and trimming ws in column names

colnames(address) <- tolower(gsub("\\.", "_", colnames(address)))
colnames(address) <- trimws(colnames(address))

# Creating stripped version of addresses for unique identifier

address$start_station_name <- address$station_name

# replace special chars with no-space

address <- address %>%
  mutate(start_station_name = gsub("[^[:alnum:]]", "", start_station_name))

# ensuring data type consistency, making lower case and trimming ws

address$start_station_name <- as.character(address$start_station_name)
address$start_station_name <- trimws(tolower(address$start_station_name))

# A copy is made for later use with ending stations

address$end_station_name <- address$start_station_name

# Removing '*' from the end of station_name

address$station_name <- gsub("\\*$", "", address$station_name)
```

It is important to note that the stripped down version of the address is used as a unique identifier for the data references later. This is because there was an inconsistency between the "ID" column for Divvy_Bicycle_Stations and the "start/end_station_id" columns for the 202305-divvy-tripdata. It is also important to note that for purposes of frequencies, the ride_id was used to create each unique instance for the measured frequencies in the analysis.

Cleaning The Data Sets

```r
# Display the number of empty values for each column

empty_counts <- colSums(divvy_data_clean == "")
empty_counts
```

We first note that there are cells with entries " " in the starting_station_name. As we use that column to perform analysis, this must be cleaned. A small portion of the analysis uses end_station_name, as well, so this will be cleaned as well. The start_station_id, and end_station_id are not used to perform analysis, so these columns will be removed. And finally, after visiting the website and seeing that their were no options for "docked bike" as a bike choice, I am left to assume this was some sort of error. Because of this, the rows containing docked_bike under column titled rideable_type have been removed. Addtionally, in retrospect, we also note that the divvy_bicycle_station_coords data frame does not have any " " entries, so we should not need to fix the address data frame.

```
##            ride_id      rideable_type         started_at           ended_at
##                  0                  0                  0                  0
## start_station_name   start_station_id   end_station_name     end_station_id
##              89240              89240              95267              95267
##          start_lat          start_lng            end_lat            end_lng
##                  0                  0                 NA                 NA
##      member_casual
##                  0
```

```r
# Code is recycled for second data frame

empty_counts <- colSums(divvy_bicycle_station_coords == "")
empty_counts
```

```
##               ID     Station.Name      Total.Docks Docks.in.Service
##                0                0                0                0
##           Status         Latitude        Longitude         Location
##                0                0                0                0
```

```r
# Removing start_station_id and end_station_id from data frame divvy_data_clean

columns_to_remove <- c("start_station_id", "end_station_id")
divvy_data_clean <- divvy_data_clean[, -which(names(divvy_data_clean) %in% columns_to_remove)]

# Removing the offending rows with "" under columns start/end_station_name

divvy_data_clean <- divvy_data_clean[trimws(divvy_data_clean$start_station_name) != "", ]
divvy_data_clean <- divvy_data_clean[trimws(divvy_data_clean$end_station_name) != "", ]

# Removing the offending rows with docked_bike under column rideable_type

divvy_data_clean <- divvy_data_clean[divvy_data_clean$rideable_type != "docked_bike", ]

# Removing id from data frame divvy_bicycle_station_coords

columns_to_remove <- c("ID")
divvy_bicycle_station_coords <- divvy_bicycle_station_coords[, -which(names(divvy_bicycle_station_coords

# Cleaning the remaining Column Names:
# Trimming whitespace / changing column names to lower case and changing '.' to
```

```r
# '_' for consistency in both data sets.

colnames(divvy_data_clean) <- tolower(trimws(colnames(divvy_data_clean)))

colnames(divvy_bicycle_station_coords) <- tolower(gsub("\\.", "_", colnames(divvy_bicycle_station_coords
colnames(divvy_bicycle_station_coords) <- trimws(colnames(divvy_bicycle_station_coords))


# changing Column Names for Consistency
# changed to match divvy_data_clean

colnames(divvy_bicycle_station_coords)[colnames(divvy_bicycle_station_coords) == "station_name"] <- "sta
```

Cleaning The Rows

```r

# Changing special characters to no-space.
# For divvy_bicycle_station_coords

divvy_bicycle_station_coords <- divvy_bicycle_station_coords %>%
  mutate(start_station_name = gsub("[^[:alnum:]]", "", start_station_name))
divvy_bicycle_station_coords$start_station_name <- as.character(divvy_bicycle_station_coords$start_stati

# For divvy_data_clean - Start

divvy_data_clean <- divvy_data_clean %>%
  mutate(start_station_name = gsub("[^[:alnum:]]", "", start_station_name))
divvy_data_clean$start_station_name <- as.character(divvy_data_clean$start_station_name)

# For divvy_data_clean - End

divvy_data_clean <- divvy_data_clean %>%
  mutate(end_station_name = gsub("[^[:alnum:]]", "", end_station_name))
divvy_data_clean$end_station_name <- as.character(divvy_data_clean$end_station_name)

# Changing rows to lower case and trimming row whitespace for specific Columns

divvy_data_clean$start_station_name <- tolower(divvy_data_clean$start_station_name)
divvy_data_clean$end_station_name <- tolower(divvy_data_clean$end_station_name)

# Trimming the whitespace in the rest of the data set - divvy_data_clean

divvy_data_clean <- data.frame(lapply(divvy_data_clean, trimws))

# Trimming the whitespace in the rest of the data set - divvy_bicycle_station_coords

divvy_bicycle_station_coords$start_station_name <- tolower(divvy_bicycle_station_coords$start_station_na
divvy_bicycle_station_coords <- data.frame(lapply(divvy_bicycle_station_coords, trimws)) # Trimming whi

#Changing address for consistency

divvy_data_clean$start_station_name <- gsub("kedzieave24thsttemp", "kedzieave24thst", divvy_data_clean$s
```

Here we will be removing Special Characters from addresses in rows abd replacing them with no-space. Following that, the letters are all moved to lower case and this is what will help create the unique identifier for the adresses. That is to say: we will use the start_station_name column created below to connect the address in its stripped down version to the address that is readable, then later use these identifiers to link to the legible address for our visuals so our stakeholders can identify the locations quickly.

Setting Up To Merge And Simplify Data Frames

```
# end_station_name is used later as a unique identifier for addresses used in viz

divvy_bicycle_station_coords$end_station_name <- divvy_bicycle_station_coords$start_station_name

# Kept ride_id in the df to create a unique instance for each row.

columns_to_remove <- c("start_lat", "start_lng", "end_lat", "end_lng", "started_at", "ended_at")
divvy_data_clean_temp <- divvy_data_clean[, -which(names(divvy_data_clean) %in% columns_to_remove)]

# Temporarily removing unused columns for data frame merger with the address data frame.

columns_to_remove <- c("end_station_name", "location", "status", "docks_in_service", "id", "total_docks
divvy_bicycle_station_coords_temp <- divvy_bicycle_station_coords[, -which(names(divvy_bicycle_station_c
columns_to_remove <- c("end_station_name")
address_temp <- address[, -which(names(address) %in% columns_to_remove)]

# Merging the data frames.

merged_df <- distinct(merge(x = divvy_bicycle_station_coords_temp, y = divvy_data_clean_temp, by = "sta
merged_df <- distinct(merge(x = merged_df, y = address_temp, by = "start_station_name", all = FALSE))

# Creating separate data frames for the casual rider and member rider.

merged_df_casual <- filter(merged_df, member_casual == "casual")
merged_df_member <- filter(merged_df, member_casual == "member")

#Removing temps

rm(divvy_bicycle_station_coords_temp)
rm(divvy_data_clean_temp)
rm(address_temp)

# Creating the 'simple' data frames for later mergers.
# Reducing Clutter in divvy_bicycle_station

columns_to_remove <- c("location", "total_docks", "status", "docks_in_service", "start_station_id")
divvy_bicycle_station_simple <- divvy_bicycle_station_coords[, -which(names(divvy_bicycle_station_coords

# Reducing Clutter in divvy_data_clean

columns_to_remove <- c("location", "total_docks", "status", "docks_in_service", "start_station_id", "ri
divvy_data_clean_simple <- divvy_data_clean[, -which(names(divvy_data_clean) %in% columns_to_remove)]
```

Here we are creating our data frames for the visuals. We use ‘divvy_bicycle_station_coords$start_station_n and ‘divvy_bicycle_station_coords$end_station_name’ to link visuals to their legible addresses within the legend. The ride_id was left in the data frame to create a unique instance of each observation.

Frequency Data Frames

```
# Starting location frequencies - combined

start_combined_freq <- as.data.frame(sort(table(merged_df$start_station_name), decreasing = TRUE))
start_combined_freq <- setNames(start_combined_freq, c("start_station_name", "frequency"))

# Ending location frequencies - combined

end_combined_freq <- as.data.frame(sort(table(merged_df$end_station_name), decreasing = TRUE))
end_combined_freq <- setNames(end_combined_freq, c("end_station_name", "frequency"))

# Starting location frequencies - casual

start_casual_freq <- as.data.frame(sort(table(merged_df_casual$start_station_name), decreasing = TRUE))
start_casual_freq <- setNames(start_casual_freq, c("start_station_name", "frequency"))

# ending location frequencies - casual

end_casual_freq <- as.data.frame(sort(table(merged_df_casual$end_station_name), decreasing = TRUE))
end_casual_freq <- setNames(end_casual_freq, c("end_station_name", "frequency"))

# Starting location frequencies - member

start_member_freq <- as.data.frame(sort(table(merged_df_member$start_station_name), decreasing = TRUE))
start_member_freq <- setNames(start_member_freq, c("start_station_name", "frequency"))

# Ending location frequencies - member

end_member_freq <- as.data.frame(sort(table(merged_df_member$end_station_name), decreasing = TRUE))
end_member_freq <- setNames(end_member_freq, c("end_station_name", "frequency"))
```

These frequency data frames will later be used to merge with the merged_df_casual/member to identify the frequencies with the locations, accounting for end location as well as the start location. From here we can freely drop the ride_id attribute since we now have start and end frequencies tied to their starting and ending locations.

```
# Casual df first.
# Eliminating the end_station_name since this is for the starting locations

columns_to_remove <- c("end_station_name")
divvy_bicycle_station_temp <- divvy_bicycle_station_simple[, -which(names(divvy_bicycle_station_simple)
address_temp <- address[, -which(names(address) %in% columns_to_remove)]

# Merging in the lng/lat of the starting locations to frequencies

casual_df_start_loc <- merge(x = start_casual_freq, y = divvy_bicycle_station_temp, by = "start_station_
```

```r
# Merging in the addresses of starting locations

casual_df_start_loc <- merge(x = casual_df_start_loc, y = address_temp, by = "start_station_name", all =

# Eliminating repeats, reordering based on frequency, and ensuring data type.

casual_df_start_loc <- distinct(casual_df_start_loc)
casual_df_start_loc <- casual_df_start_loc[order(casual_df_start_loc$frequency, decreasing = TRUE), ]
casual_df_start_loc$latitude <- as.numeric(casual_df_start_loc$latitude)
casual_df_start_loc$longitude <- as.numeric(casual_df_start_loc$longitude)

# Member df Second.
# Merging in the lng/lat of the starting locations to frequencies

member_df_start_loc <- merge(x = start_member_freq, y = divvy_bicycle_station_temp, by = "start_station_

# Merging in the addresses of starting locations

member_df_start_loc <- merge(x = member_df_start_loc, y = address_temp, by = "start_station_name", all =

# Eliminating repeats, reordering based on frequency, and ensuring data type.

member_df_start_loc <- distinct(member_df_start_loc)
member_df_start_loc <- member_df_start_loc[order(member_df_start_loc$frequency, decreasing = TRUE), ]
member_df_start_loc$latitude <- as.numeric(member_df_start_loc$latitude)
member_df_start_loc$longitude <- as.numeric(member_df_start_loc$longitude)

# Removing Temps

rm(divvy_bicycle_station_temp)
rm(address_temp)
```

Creating Starting Location Data Frames For Visuals

```r
# Casual df first.
# Eliminating the start_station_name since this is for the ending locations

columns_to_remove <- c("start_station_name")
divvy_bicycle_station_temp <- divvy_bicycle_station_simple[, -which(names(divvy_bicycle_station_simple)
address_temp <- address[, -which(names(address) %in% columns_to_remove)]

# Merging in the lng/lat of the starting locations to frequencies

casual_df_end_loc <- merge(x = end_casual_freq, y = divvy_bicycle_station_temp, by = "end_station_name"

# Merging in the addresses of starting locations

casual_df_end_loc <- merge(x = casual_df_end_loc, y = address_temp, by = "end_station_name", all = FALSE

# Eliminating repeats, reordering based on frequency, and ensuring data type.

casual_df_end_loc <- distinct(casual_df_end_loc)
```

```r
casual_df_end_loc <- casual_df_end_loc[order(casual_df_end_loc$frequency, decreasing = TRUE), ]
casual_df_end_loc$latitude <- as.numeric(casual_df_end_loc$latitude)
casual_df_end_loc$longitude <- as.numeric(casual_df_end_loc$longitude)

# Member df second.
# Merging in the lng/lat of the starting locations to frequencies

member_df_end_loc <- merge(x = end_member_freq, y = divvy_bicycle_station_temp, by = "end_station_name"

# Merging in the addresses of starting locations

member_df_end_loc <- merge(x = member_df_end_loc, y = address_temp, by = "end_station_name", all = FALSE

# Eliminating repeats, reordering based on frequency, and ensuring data type.

member_df_end_loc <- distinct(member_df_end_loc)
member_df_end_loc <- member_df_end_loc[order(member_df_end_loc$frequency, decreasing = TRUE), ]
member_df_end_loc$latitude <- as.numeric(member_df_end_loc$latitude)
member_df_end_loc$longitude <- as.numeric(member_df_end_loc$longitude)

# Removing Temps

rm(divvy_bicycle_station_temp)
rm(address_temp)
```

Creating Ending Location Data Frames For Visuals

Creating The Most Popular Day & Time Data Frame

```r
columns_to_remove <- c("ride_id", "rideable_type", "start_station_name", "start_station_id", "end_statio
divvy_data_clean_temp <- divvy_data_clean[, -which(names(divvy_data_clean) %in% columns_to_remove)]

# Splitting the DateTime column for starting date/time into separate Date and Time columns

day_pop <- separate(divvy_data_clean_temp, "started_at", into = c("starting_date", "starting_time"), sep
day_pop$starting_date <- as.Date(day_pop$starting_date, "%m/%d/%y")
day_pop$starting_timehr <- substr(day_pop$starting_time, start = 1, stop = 2)

# Splitting the DateTime column for ending date/time into separate Date and Time columns

day_pop <- separate(day_pop, "ended_at", into = c("ending_date", "ending_time"), sep = " ")
day_pop$ending_date <- as.Date(day_pop$ending_date, "%m/%d/%y")
day_pop$ending_timehr <- substr(day_pop$ending_time, start = 1, stop = 2)

# replace special chars in the time with no-space and then add 0 to the
# beginning of the string if there is only one character present

day_pop <- day_pop %>%
  mutate(starting_timehr = gsub("[^[:alnum:]]", "", starting_timehr))
day_pop$starting_timehr <- ifelse(nchar(day_pop$starting_timehr) == 1, paste0("0",day_pop$starting_time
day_pop$weekday <- weekdays(day_pop$starting_date)
```

```r
# Removing Temps

rm(divvy_data_clean_temp)
```

It is important to note here that the hour for the time was not working with the separate() and as.POSIXct and were only returning NA.The fix was the extract the substring and then fix that result by adding a '0' in front of any entry that had only one digit, namely the cases that started in the first half of the day.

```r
day_pop_temp <- subset(day_pop, select = c("member_casual", "starting_timehr", "weekday"))

# For Casual Riders -
# Filter the data for Wednesday and Thursday - See day_freqc for frequencies

time_pop <- subset(day_pop_temp, weekday %in% c("Wednesday", "Thursday"))
time_popw <-filter(time_pop, member_casual == "casual", weekday == "Wednesday")
time_popt <-filter(time_pop, member_casual == "casual", weekday == "Thursday")

time_pop_freqw <- as.data.frame(table(time_popw$starting_timehr))
time_pop_freqw <- setNames(time_pop_freqw, c("starting_hr", "frequency"))
time_pop_freqw <- time_pop_freqw %>% arrange(starting_hr)

time_pop_freqt <- as.data.frame(table(time_popt$starting_timehr))
time_pop_freqt <- setNames(time_pop_freqt, c("starting_hr", "frequency"))
time_pop_freqt <- time_pop_freqt %>% arrange(starting_hr)

# For Member Riders -
# Filter the data for Saturday and Sunday - see day_freqm for frequencies

time_pop <- subset(day_pop_temp, weekday %in% c("Saturday", "Sunday"))
time_popsa <-filter(time_pop, member_casual == "member", weekday == "Saturday")
time_popsu <-filter(time_pop, member_casual == "member", weekday == "Sunday")

time_pop_freqsa <- as.data.frame(table(time_popsa$starting_timehr))
time_pop_freqsa <- setNames(time_pop_freqsa, c("starting_hr", "frequency"))
time_pop_freqsa <- time_pop_freqsa %>% arrange(starting_hr)

time_pop_freqsu <- as.data.frame(table(time_popsu$starting_timehr))
time_pop_freqsu <- setNames(time_pop_freqsu, c("starting_hr", "frequency"))
time_pop_freqsu <- time_pop_freqsu %>% arrange(starting_hr)
```

Creating The Most Popular Time Data Frames

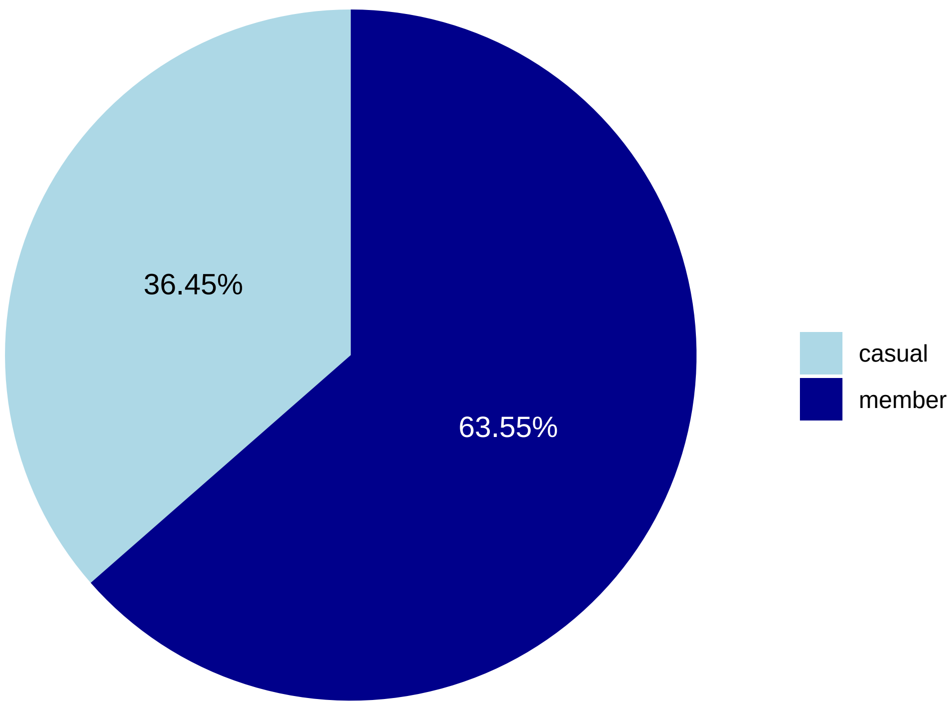
Visuals

Comparison - Member Vs Casual - Count

```r
# Calculate the counts
category_counts <- table(merged_df$member_casual)

# Create a data frame with the category labels and counts
label_freq <- data.frame(category = names(category_counts), count = category_counts)
```

```r
label_freq$count <- label_freq$count.Freq

# Calculate the relative frequency
label_freq$relative_frequency <- label_freq$count / sum(label_freq$count)

# Define the color scheme
colors <- c("lightblue", "darkblue")

# Create the pie chart
ggplot(data = label_freq, aes(x = "", y = relative_frequency, fill = factor(category))) +
  geom_bar(width = 1, stat = "identity") +
  scale_fill_manual(name = "", values = colors) +
  geom_text(aes(label = paste0(round(relative_frequency * 100, 2), "%")),
            position = position_stack(vjust = 0.5), color = ifelse(label_freq$category == "member", "whi
  coord_polar("y", start = 0) +
  theme_void()
```

Our first comparison between the two rider types, member and casual, looks at the percentage of rides by members versus the percentage of rides by casual riders for the month of May, 2023. Here we see the majority of rides taken were by the member riders.



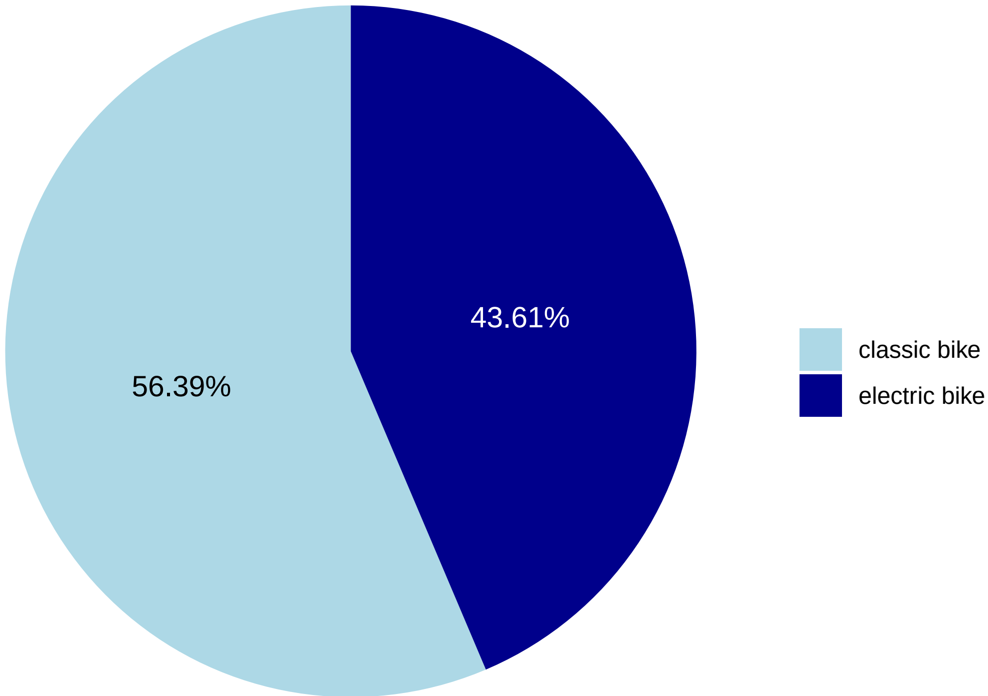
Preferred Type Of Bike

Preference - Casual - Electric Vs Classic

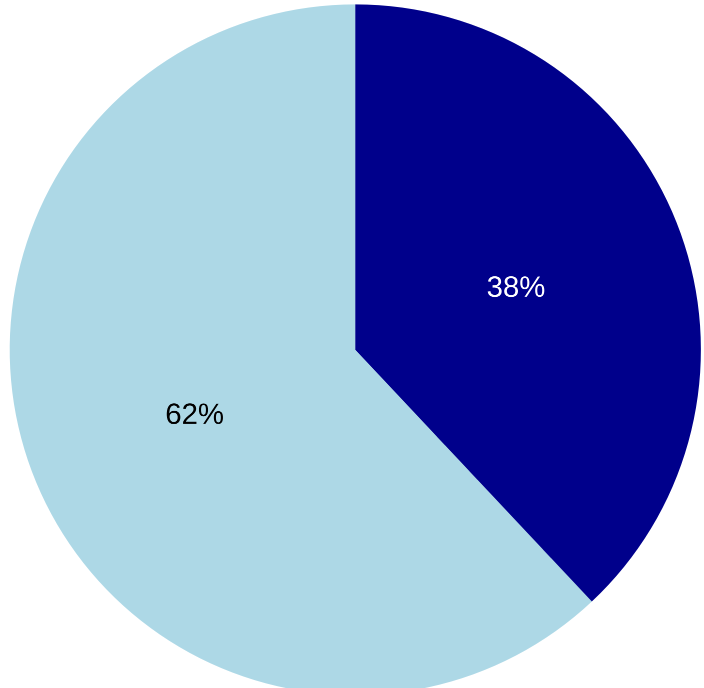
Next we consider the preference of electric bike versus the classic bike. First we consider the casual riders which, based on the pie chart, shows there is a preference towards using the classic bike.

```r
# Calculate the counts
category_counts <- table(merged_df_casual$rideable_type)

# Create a data frame with the category labels and counts
label_freq <- data.frame(category = names(category_counts), count = category_counts)
label_freq$count <- label_freq$count.Freq
label_freq$category <- gsub("_", " ", label_freq$category)
custom_labels <- label_freq$category

# Calculate the relative frequency
label_freq$relative_frequency <- label_freq$count / sum(label_freq$count)

# Define the color scheme
colors <- c("lightblue", "darkblue")

# Create the pie chart
ggplot(data = label_freq, aes(x = "", y = relative_frequency, fill = factor(category))) +
  geom_bar(width = 1, stat = "identity") +
  scale_fill_manual(name = "", values = colors, labels = custom_labels) +
  geom_text(aes(label = paste0(round(relative_frequency * 100, 2), "%")),
            position = position_stack(vjust = 0.5), color = ifelse(label_freq$category == "electric bik
  coord_polar("y", start = 0) +
  theme_void()
```

For the member riders, we see a similar preference with member riders preferring the classic bike. However, note that the preference for members to choose the classic bike is stronger than the preference for casual riders to choose the classic bike.




```
# Calculate the counts
category_counts <- table(merged_df_member$rideable_type)

# Create a data frame with the category labels and counts
label_freq <- data.frame(category = names(category_counts), count = category_counts)
label_freq$count <- label_freq$count.Freq
label_freq$category <- gsub("_", " ", label_freq$category)
custom_labels <- label_freq$category

# Calculate the relative frequency
label_freq$relative_frequency <- label_freq$count / sum(label_freq$count)

# Define the color scheme
colors <- c("lightblue", "darkblue")

# Create the pie chart
ggplot(data = label_freq, aes(x = "", y = relative_frequency, fill = factor(category))) +
  geom_bar(width = 1, stat = "identity") +
  scale_fill_manual(name = "", values = colors, labels = custom_labels) +
  geom_text(aes(label = paste0(round(relative_frequency * 100, 2), "%")),
            position = position_stack(vjust = 0.5), color = ifelse(label_freq$category == "electric bik
  coord_polar("y", start = 0) +
  theme_void()
```



Preference - Member - Electric Vs Classic

Preferred Starting Location

We next consider the preffered starting location for both the members and the casual riders. Given that there are well over one thousand sites being tracked, we selected the top 6 locations based on the strength of their frequency compared to others.

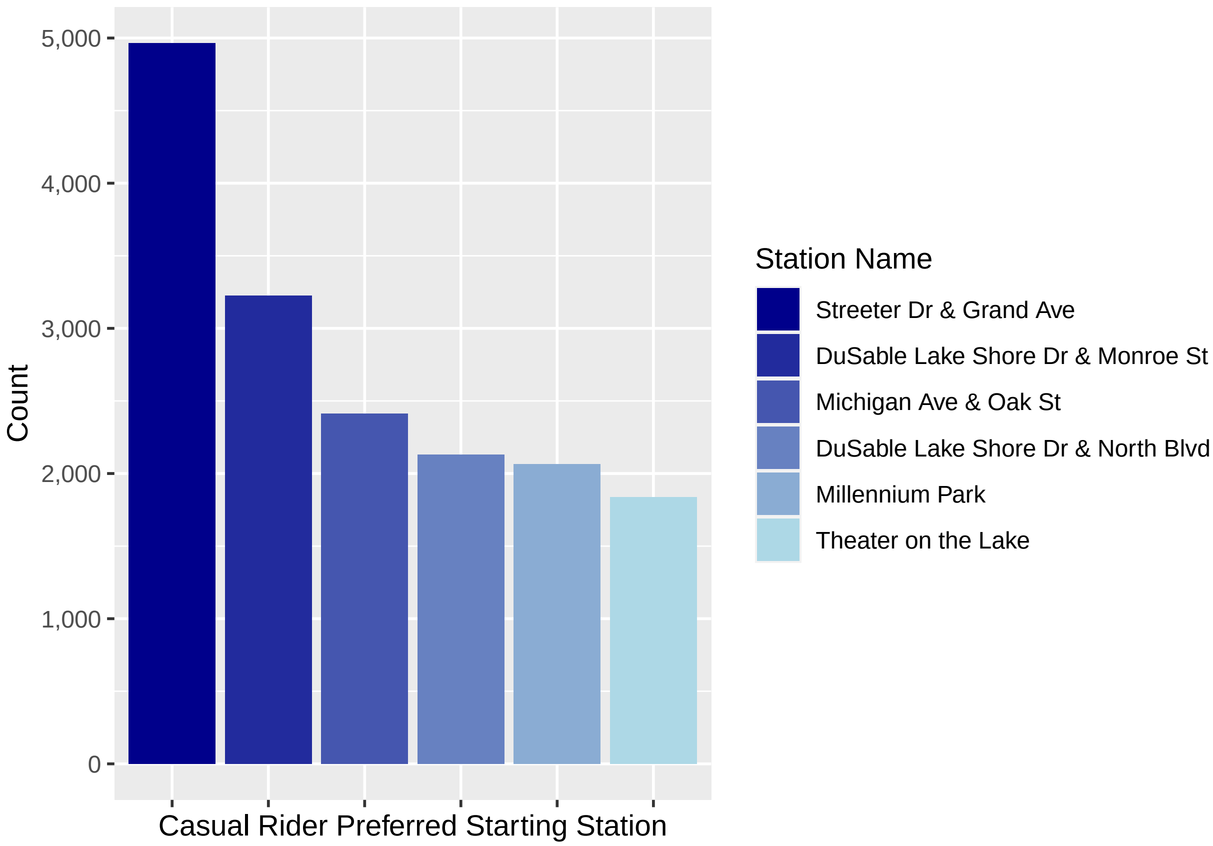
For casual riders, the preferred starting location from greatest to least is: Streeter Dr. & Grand Ave., DuSable Lake Shore Dr. & Monroe St., Michigan Ave & Oak St., DuSable Lake Shore Dr. & North Blvd., Millennium Park, and Theater on the Lake. When we look at the Top 6 Starting Locations for Casual Riders map, we see these 6 locations are along the coastline of Lake Michigan.

For the member riders, the preferred starting location from greatest to least is: Clinton St. & Washington Blvd., Kingsbury St. & Kinzie St., Clark St. & Elm St., University Ave. & 57th St., Wells St. & Concord Ln., Streeter Dr. & Grand Ave. When we look at the Top 6 Starting Locations for Member Riders map, we see that these locations are more focused on the downtown regions of the city of Chicago.

Preference - Casual - Starting Location

```
top_startc <- casual_df_start_loc[1:6, c("station_name", "frequency")]
top_startc <- top_startc[order(top_startc$frequency, decreasing = TRUE), ]
top_startc$station_name <- factor(top_startc$station_name, levels = top_startc$station_name)
```

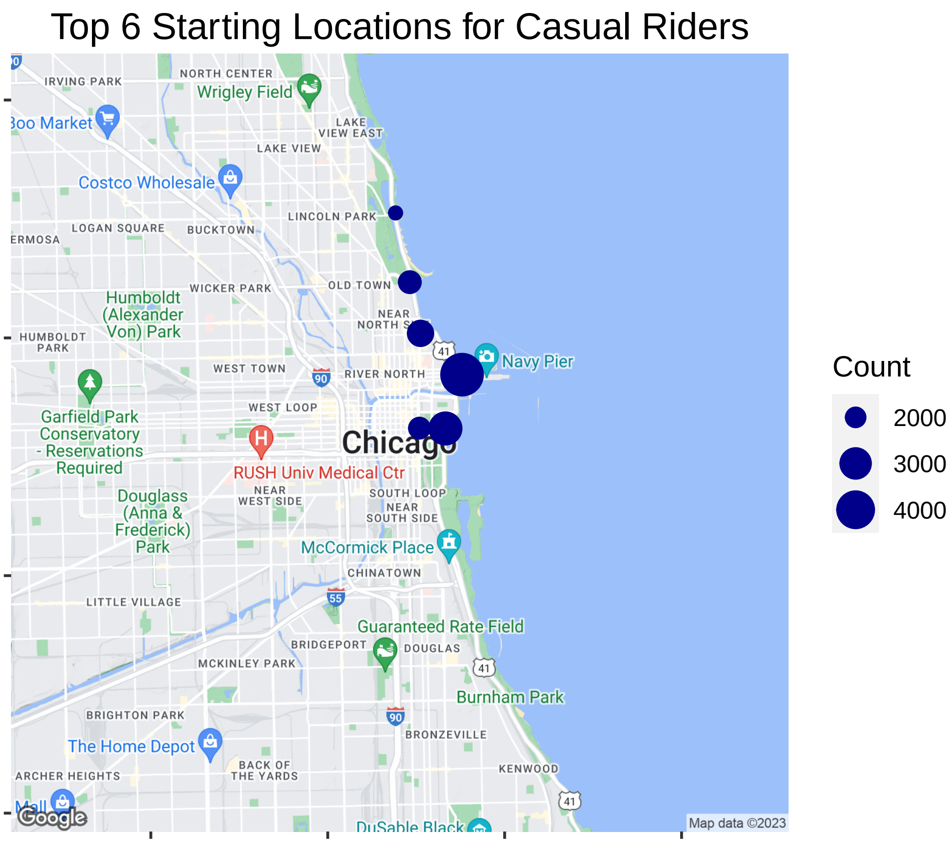
```r
custom_labels <- top_startc$station_name
colors <- colorRampPalette(c("darkblue", "lightblue"))(length(custom_labels))

ggplot(top_startc, aes(x = station_name, y = frequency, fill = station_name)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = colors, labels = custom_labels) +
  labs(x = "Casual Rider Preferred Starting Station", y = "Count", fill = "Station Name") +
  scale_y_continuous(labels = scales::comma) +
  theme(axis.text.x = element_blank())
```



Barchart

```r
top_startc_geo <- casual_df_start_loc[1:6, c("station_name", "frequency", "longitude", "latitude")]
top_startc_geo <- top_startc_geo[order(top_startc_geo$frequency, decreasing = TRUE), ]
top_startc_geo$station_name <- factor(top_startc_geo$station_name, levels = top_startc_geo$station_name

chicago_map <- get_map(location = "Chicago", zoom = 12)

member_map <- ggmap(chicago_map) +
  geom_point(data = top_startc_geo, aes(x = longitude, y = latitude, size = frequency), color = "darkbl
  labs(title = "Top 6 Starting Locations for Casual Riders", x = NULL, y = NULL, size = "Count") +
  scale_size(range = c(2, 7)) +
  theme(axis.text.x = element_blank(), axis.title.x = element_blank(), axis.text.y = element_blank(), a

# Save the map as a PNG file
ggsave("casual_start_map.png", plot = member_map, width = 8, height = 6, dpi = 300)
```

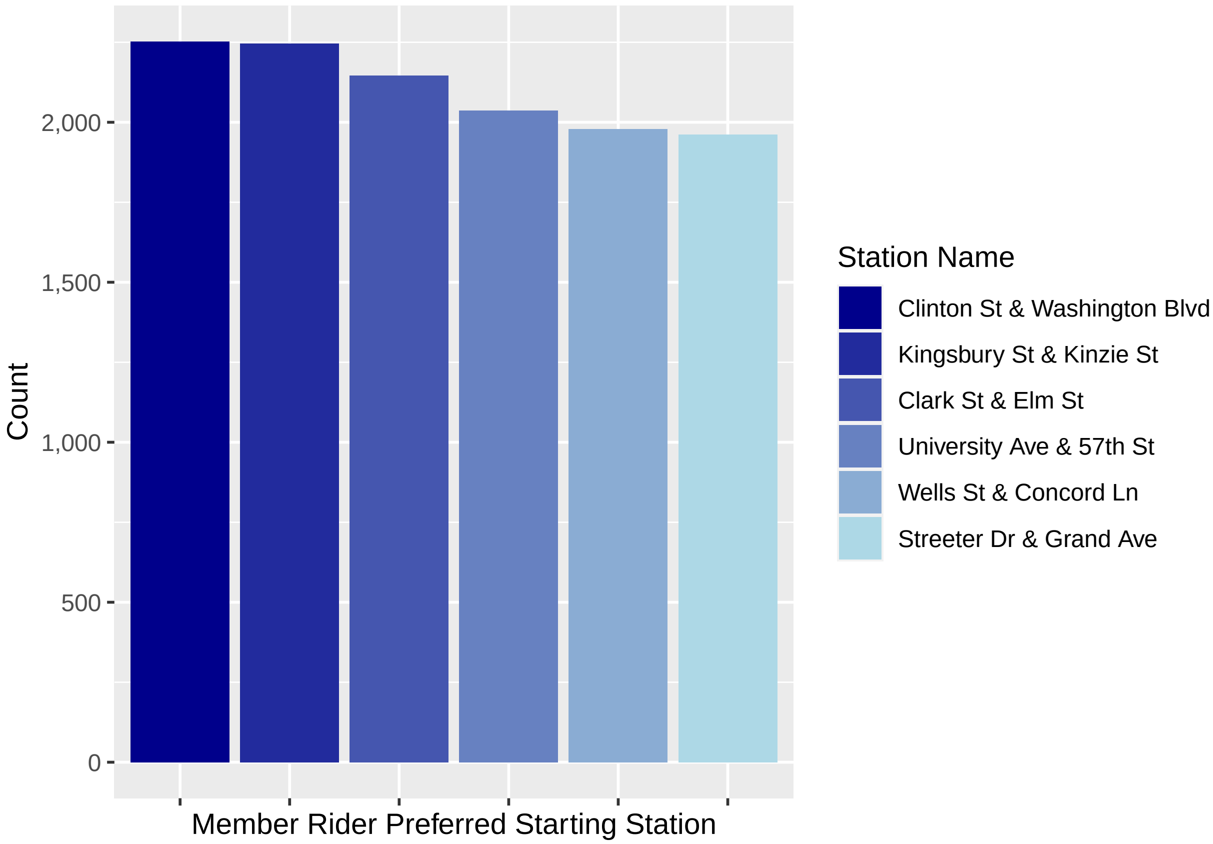
```
plot(member_map)
```



Map

Preference - Member - Starting Location

```
top_startm <- member_df_start_loc[1:6, c("station_name", "frequency")]
top_startm <- top_startm[order(top_startm$frequency, decreasing = TRUE), ]
top_startm$station_name <- factor(top_startm$station_name, levels = top_startm$station_name)

custom_labels <- top_startm$station_name
colors <- colorRampPalette(c("darkblue", "lightblue"))(length(custom_labels))

ggplot(top_startm, aes(x = station_name, y = frequency, fill = station_name)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = colors, labels = custom_labels) +
  labs(x = "Member Rider Preferred Starting Station", y = "Count", fill = "Station Name") +
  scale_y_continuous(labels = scales::comma) +
  theme(axis.text.x = element_blank())
```

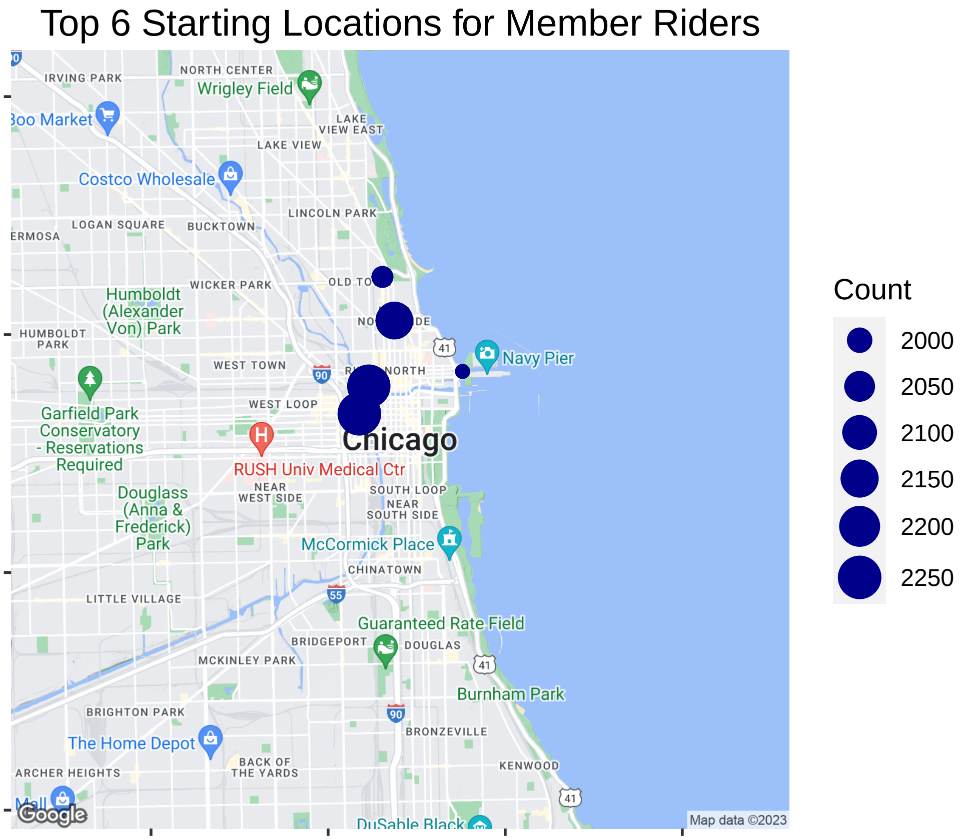


Barchart

```
top_startm_geo <- member_df_start_loc[1:6, c("station_name", "frequency", "longitude", "latitude")]
top_startm_geo <- top_startm_geo[order(top_startm_geo$frequency, decreasing = TRUE), ]
top_startm_geo$station_name <- factor(top_startm_geo$station_name, levels = top_startm_geo$station_name

chicago_map <- get_map(location = "Chicago", zoom = 12)

member_map <- ggmap(chicago_map) +
  geom_point(data = top_startm_geo, aes(x = longitude, y = latitude, size = frequency), color = "darkblu
  labs(title = "Top 6 Starting Locations for Member Riders", x = NULL, y = NULL, size = "Count") +
  scale_size(range = c(2, 7)) +
  theme(axis.text.x = element_blank(), axis.title.x = element_blank(), axis.text.y = element_blank(), a

# Save the map as a PNG file
#ggsave("member_start_map.png", plot = member_map, width = 8, height = 6, dpi = 300)
member_map
```

Map

```
## Warning: Removed 1 rows containing missing values (`geom_point()`).
```



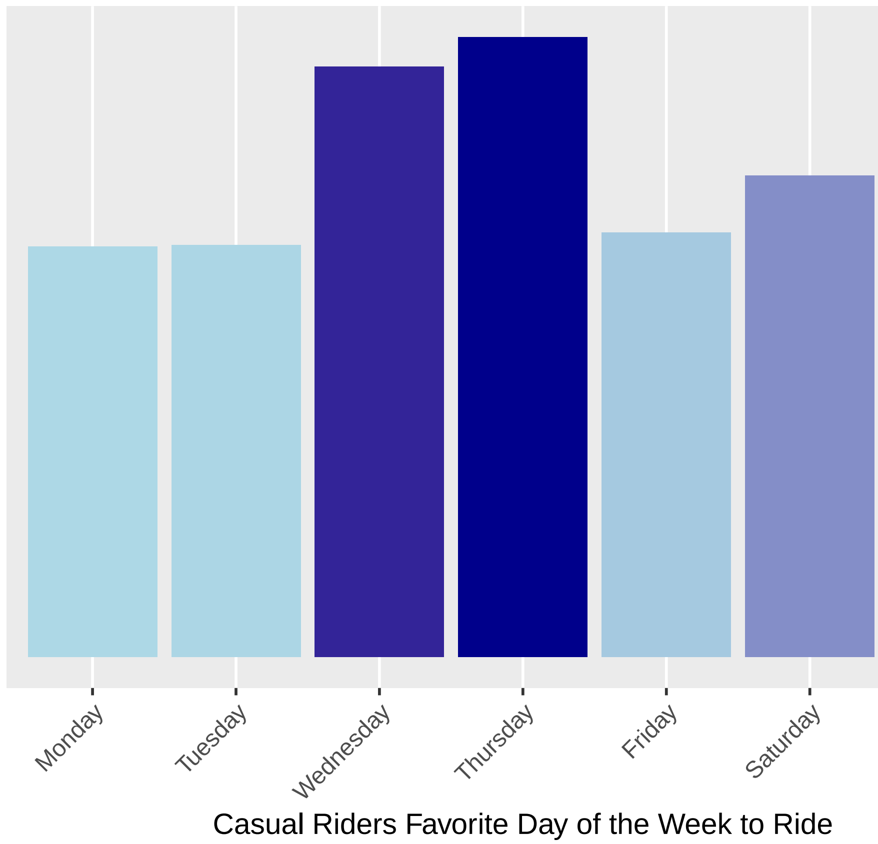
Preferred Day And Time

Now that we know the preferred starting locations for the riders, we now consider their preferred day or days to ride. FOr the casual rider, the preferred days appear to be Wednesday and Thursday, whereas for the member rider, the preferred days appear to be Saturday and Sunday. Additionally, the average travel time for the casual rider is around 22 minutes and 30 seconds where the member rider spends an average of 12 minutes and 53 seconds on travel time.

We next consider their preferred time to ride. For the casual riders the preference for starting is strongest from 1:00 PM to 5:00 PM on Wednesday and 12:00 PM to 4:00 PM on Thursdays. Whereas for member riders the preference for starting is strongest at 5:00 PM on both Saturday and Sunday. It is important to note that the distribution for the Member riders is bimodal with the other strong preference occuring at around 8:00 AM.

```
#Creating the frequency table for the most popular day for casual riders.

day_freqc <-filter(day_pop, member_casual == "casual")
day_freqc <- as.data.frame(table(day_freqc$weekday))
day_freqc <- setNames(day_freqc, c("weekday", "frequency"))

# Setting the order in which the bars are displayed.

order_temp <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday")
weekdays_ordered <- factor(day_freqc$weekday, levels = order_temp)
```

```r
ggplot(data = day_freqc, aes(x = weekdays_ordered, y = frequency, fill = frequency)) +
  geom_bar(stat = "identity") +
  scale_fill_gradient(name = "Count", low = "lightblue", high = "darkblue") +
  labs(x = "Casual Riders Favorite Day of the Week to Ride", y = NULL) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_y_continuous(breaks = NULL) +
  theme(axis.title.y = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

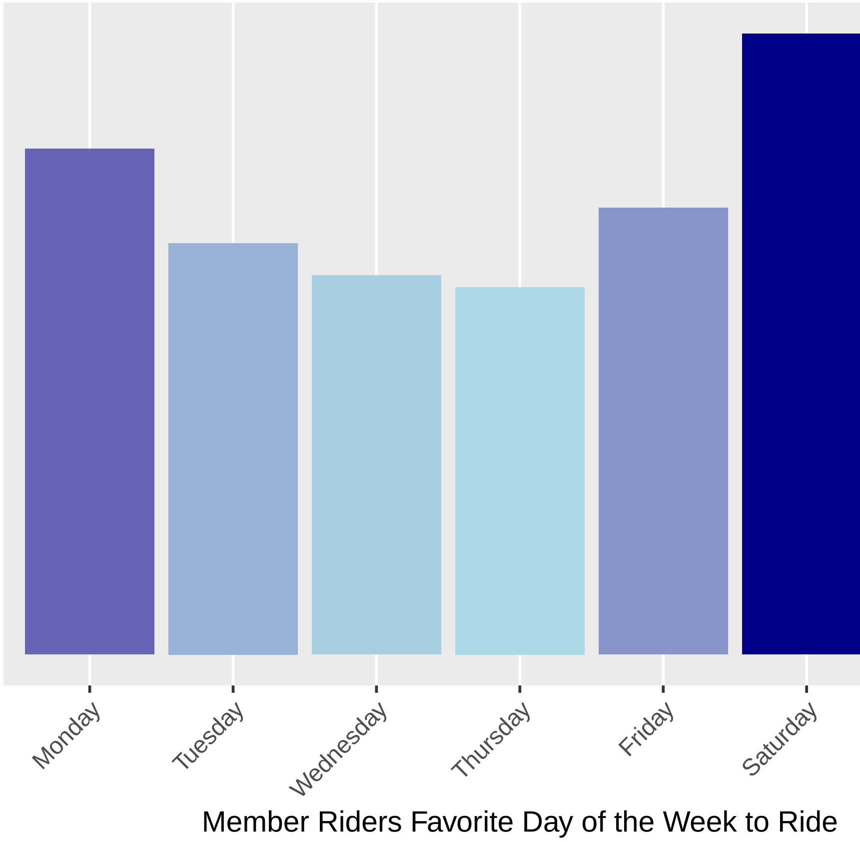


Preference - Casual - Most Popular Day

```r
rm(order_temp)
```

```r
#Creating the frequency table for the most popular day for member riders.

day_freqm <-filter(day_pop, member_casual == "member")
day_freqm <- as.data.frame(table(day_freqm$weekday))
day_freqm <- setNames(day_freqm, c("weekday", "frequency"))

# Setting the order in which the bars are displayed.

order_temp <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday")
weekdays_ordered <- factor(day_freqm$weekday, levels = order_temp)

ggplot(data = day_freqm, aes(x = weekdays_ordered, y = frequency, fill = frequency)) +
```

```r
  geom_bar(stat = "identity") +
  scale_fill_gradient(name = "Count", low = "lightblue", high = "darkblue") +
  labs(x = "Member Riders Favorite Day of the Week to Ride", y = NULL) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_y_continuous(breaks = NULL) +
  theme(axis.title.y = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```



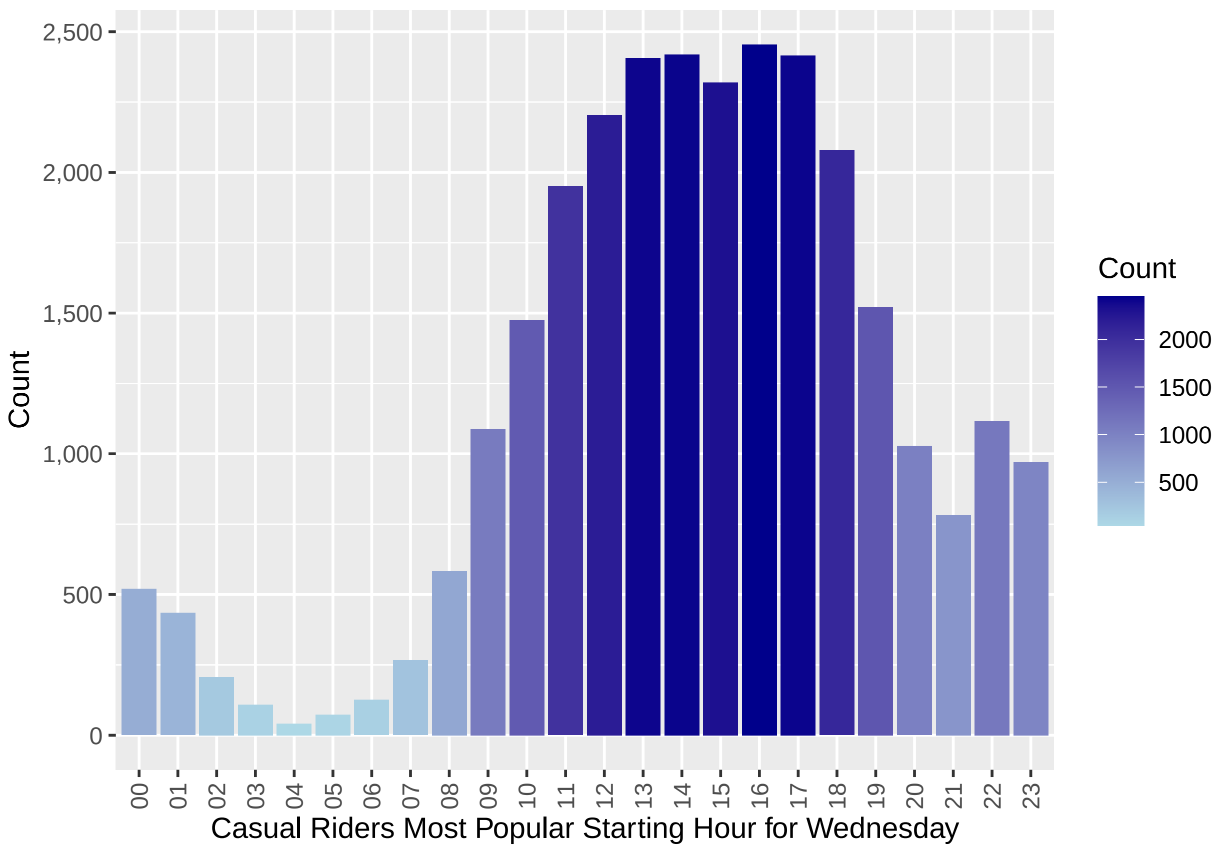
Preference - Member - Most Popular Day

```r
rm(order_temp)
```

Preferred Starting Time

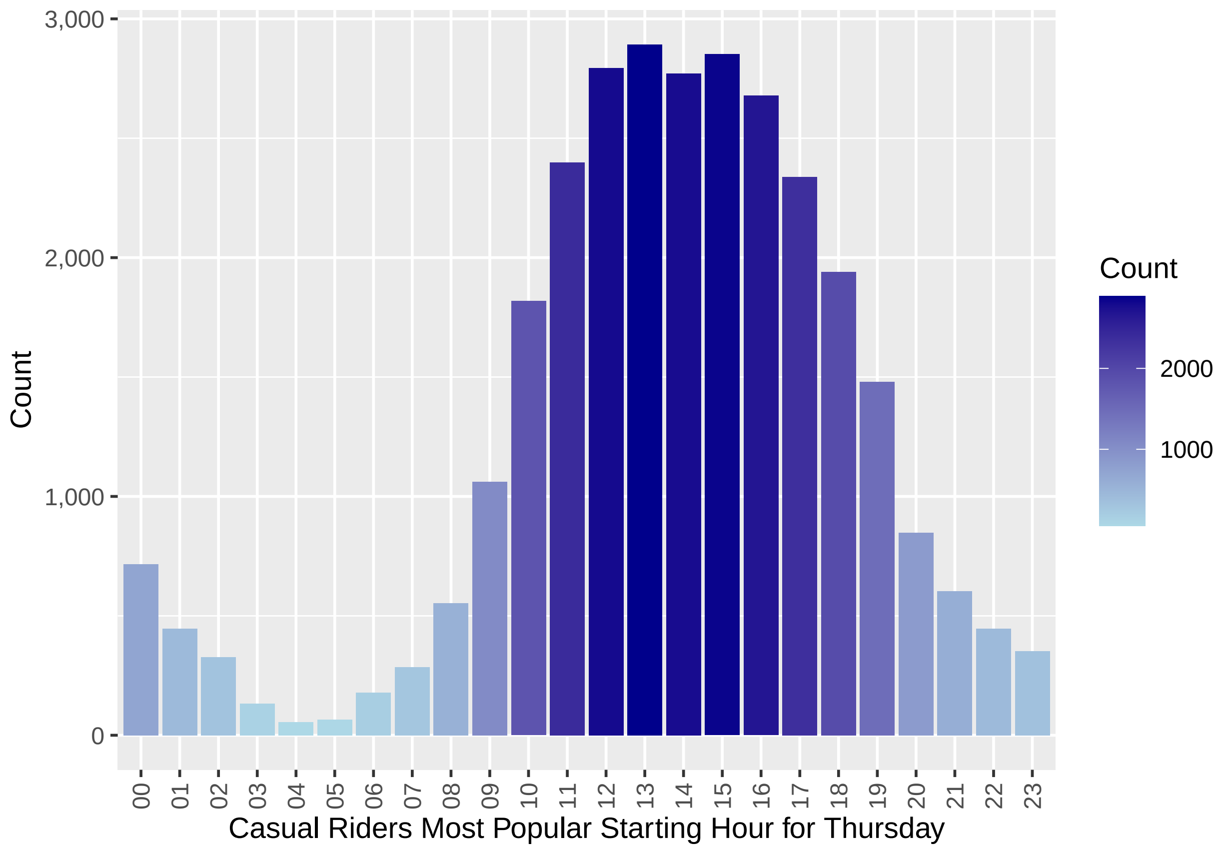
Preference - Casual - Most Popular Time

```r
ggplot(data = time_pop_freqw, aes(x = starting_hr, y = frequency, fill = frequency)) +
  geom_bar(stat = "identity") +
  scale_fill_gradient(name = "Count", low = "lightblue", high = "darkblue") +
  labs(x = "Casual Riders Most Popular Starting Hour for Wednesday", y = "Count") +
  scale_x_discrete(labels = unique(time_pop_freqw$starting_hr), limits = unique(time_pop_freqw$starting_
  scale_y_continuous(labels = scales::comma) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```



Wednesday

```
ggplot(data = time_pop_freqt, aes(x = starting_hr, y = frequency, fill = frequency)) +
  geom_bar(stat = "identity") +
  scale_fill_gradient(name = "Count", low = "lightblue", high = "darkblue") +
  labs(x = "Casual Riders Most Popular Starting Hour for Thursday", y = "Count") +
  scale_x_discrete(labels = unique(time_pop_freqt$starting_hr), limits = unique(time_pop_freqt$starting_
  scale_y_continuous(labels = scales::comma) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```



Thursday

```
columns_to_remove <- c("rideable_type", "start_station_name", "end_station_name")
day_pop_temp <-filter(divvy_data_clean_simple, member_casual == "casual")
day_pop_temp <- day_pop_temp[, -which(names(day_pop_temp) %in% columns_to_remove)]

day_pop_temp$started_at <- as.POSIXct(day_pop_temp$started_at, format = "%m/%d/%Y %H:%M")
day_pop_temp$ended_at <- as.POSIXct(day_pop_temp$ended_at, format = "%m/%d/%Y %H:%M")

day_pop_temp$time_trav <- as.numeric(difftime(day_pop_temp$ended_at, day_pop_temp$started_at, units = "

avg_ride <- mean(day_pop_temp$time_trav[day_pop_temp$time_trav != 0])

print(avg_ride)
```

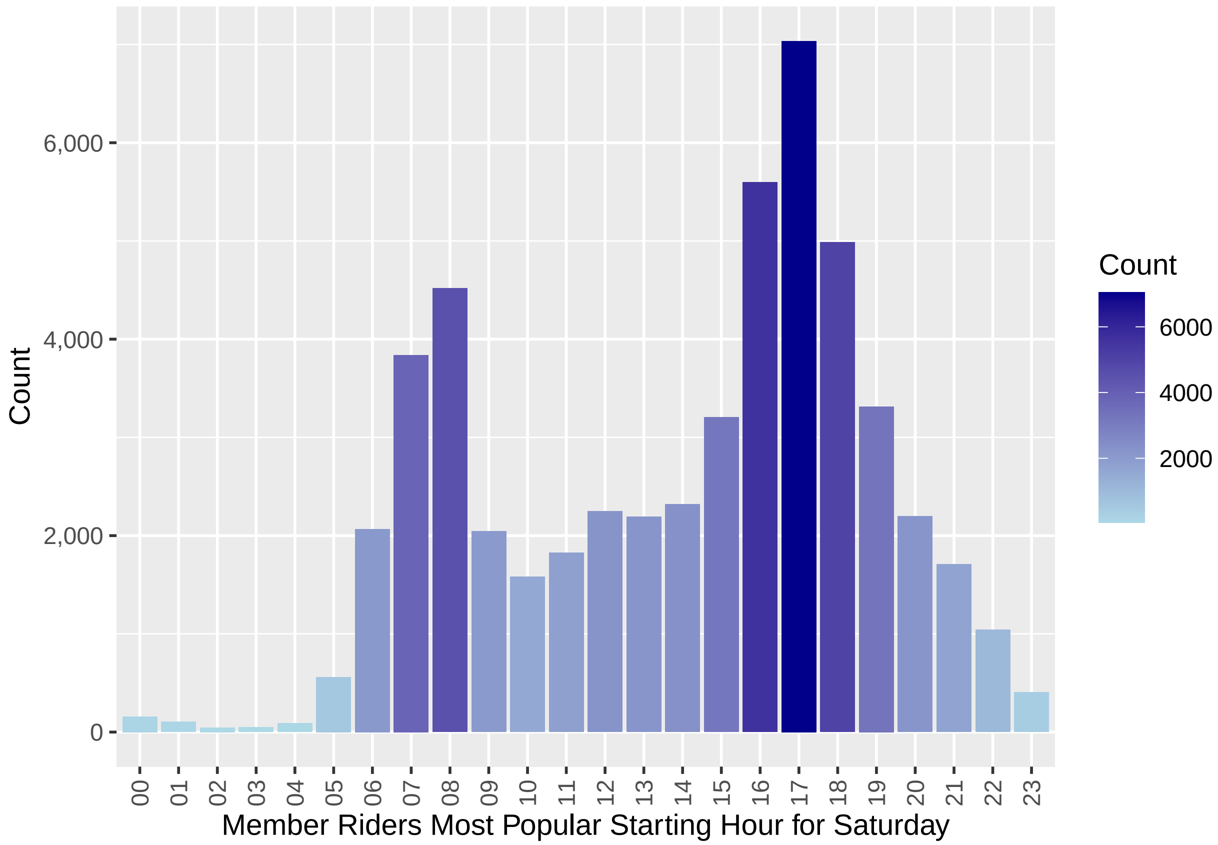
Average Travel Time - Casual Riders

```
## [1] 22.50132
```

Preference - Member - Most Popular Time

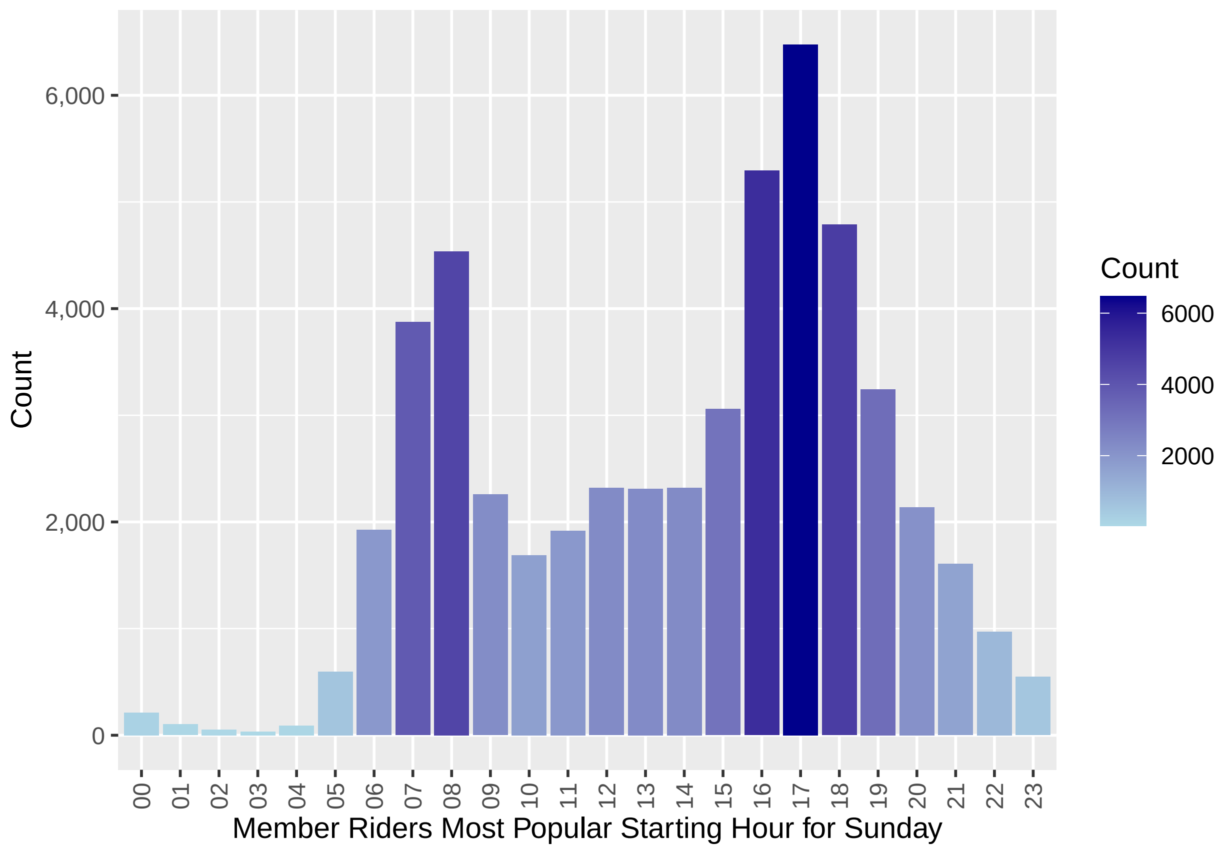
```
ggplot(data = time_pop_freqsa, aes(x = starting_hr, y = frequency, fill = frequency)) +
  geom_bar(stat = "identity") +
  scale_fill_gradient(name = "Count", low = "lightblue", high = "darkblue") +
  labs(x = "Member Riders Most Popular Starting Hour for Saturday", y = "Count") +
  scale_x_discrete(labels = unique(time_pop_freqsa$starting_hr), limits = unique(time_pop_freqsa$startin
```

```
  scale_y_continuous(labels = scales::comma) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```



Saturday

```
ggplot(data = time_pop_freqsu, aes(x = starting_hr, y = frequency, fill = frequency)) +
  geom_bar(stat = "identity") +
  scale_fill_gradient(name = "Count", low = "lightblue", high = "darkblue") +
  labs(x = "Member Riders Most Popular Starting Hour for Sunday", y = "Count") +
  scale_x_discrete(labels = unique(time_pop_freqsu$starting_hr), limits = unique(time_pop_freqsu$startin
  scale_y_continuous(labels = scales::comma) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```



Sunday

```
columns_to_remove <- c("rideable_type", "start_station_name", "end_station_name")
day_pop_temp <-filter(divvy_data_clean_simple, member_casual == "member")
day_pop_temp <- day_pop_temp[, -which(names(day_pop_temp) %in% columns_to_remove)]

day_pop_temp$started_at <- as.POSIXct(day_pop_temp$started_at, format = "%m/%d/%Y %H:%M")
day_pop_temp$ended_at <- as.POSIXct(day_pop_temp$ended_at, format = "%m/%d/%Y %H:%M")

day_pop_temp$time_trav <- as.numeric(difftime(day_pop_temp$ended_at, day_pop_temp$started_at, units = "

avg_ride <- mean(day_pop_temp$time_trav[day_pop_temp$time_trav != 0])

print(avg_ride)
```

Average Travel Time - Member Riders

```
## [1] 12.88419
```

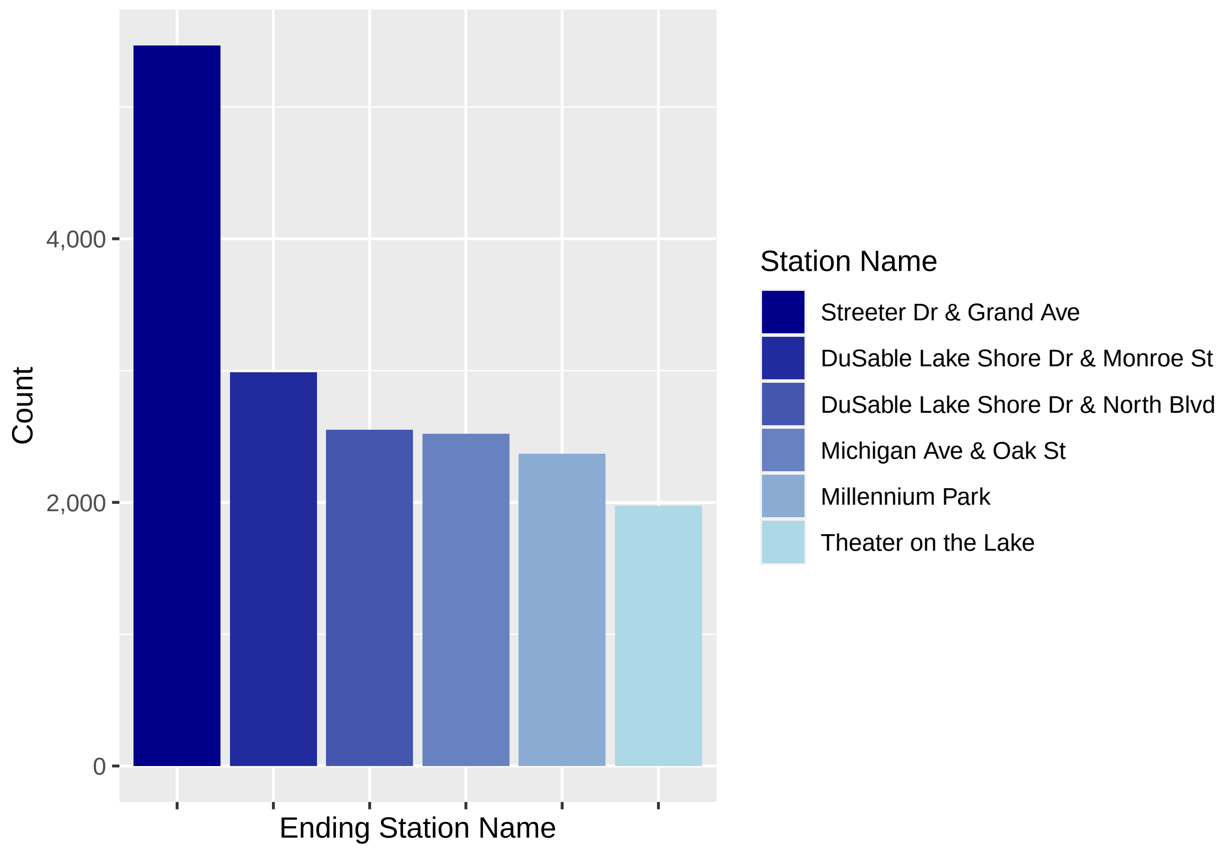
Preferred Ending Location

We finally consider the preffered ending location for both the members and the casual riders. For casual riders, the preferred ending location from greatest to least is: Streeter Dr. & Grand Ave., DuSable Lake Shore Dr. & Monroe St., DuSable Lake Shore Dr. & North Blvd., Michigan Ave & Oak St., Millennium Park, and Theater on the Lake. When we look at the Top 6 Ending Locations for Casual Riders map, we see these 6 locations are, again, along the coastline of Lake Michigan.For the member riders, the preferred ending location from greatest to least is: Clinton St. & Washington Blvd., Kingsbury St. & Kinzie St., Clark St. & Elm St.,

University Ave. & 57th St., Wells St. & Concord Ln., and Ellis Ave & 60th St. When we look at the Top 6 Ending Locations for Member Riders map, we see that these locations are, again, more focused on the downtown regions of the city of Chicago.

Preference - Casual - Ending Location

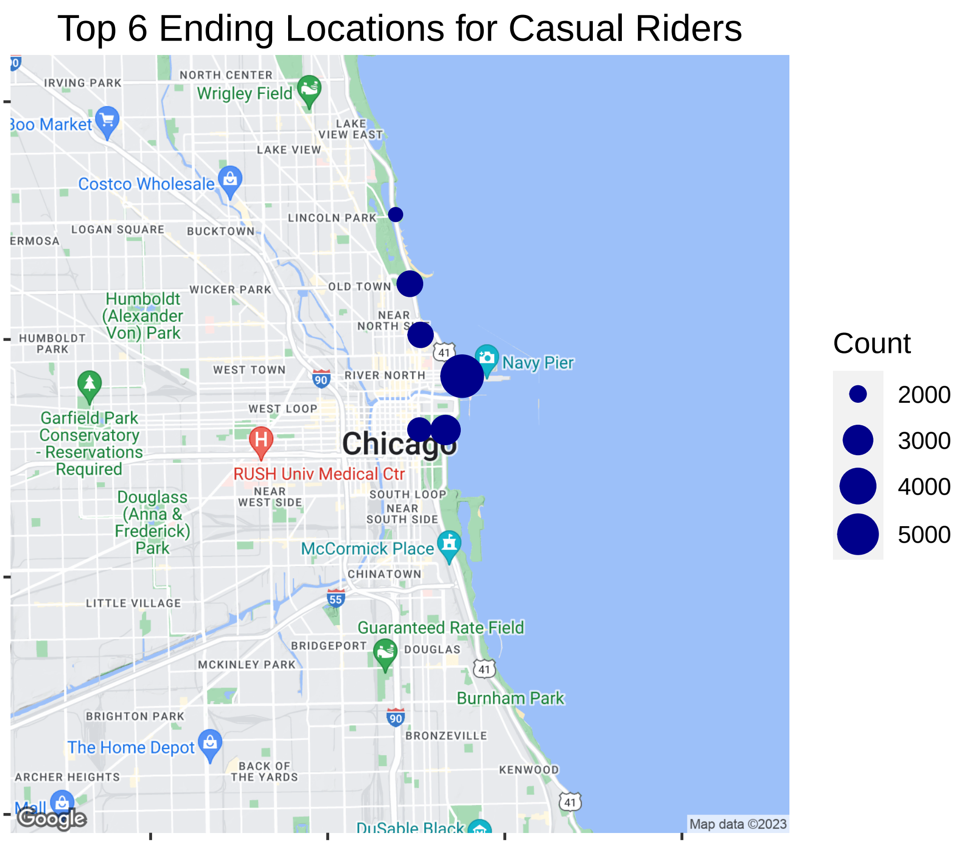
```
top_endc <- casual_df_end_loc[1:6, c("station_name", "frequency")]
top_endc <- top_endc[order(top_endc$frequency, decreasing = TRUE), ]
top_endc$station_name <- factor(top_endc$station_name, levels = top_endc$station_name)

custom_labels <- top_endc$station_name
colors <- colorRampPalette(c("darkblue", "lightblue"))(length(custom_labels))

ggplot(data = top_endc, aes(x = station_name, y = frequency, fill = station_name)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = colors, labels = custom_labels) +
  labs(x = "Ending Station Name", y = "Count", fill = "Station Name") +
  scale_y_continuous(labels = scales::comma) +
  theme(axis.text.x = element_blank())
```



Barchart

```
top_endc_geo <- casual_df_end_loc[1:6, c("station_name", "frequency", "longitude", "latitude")]
top_endc_geo <- top_endc_geo[order(top_endc_geo$frequency, decreasing = TRUE), ]
top_endc_geo$station_name <- factor(top_endc_geo$station_name, levels = top_endc_geo$station_name)
```

```
chicago_map <- get_map(location = "Chicago", zoom = 12)

member_map <- ggmap(chicago_map) +
  geom_point(data = top_endc_geo, aes(x = longitude, y = latitude, size = frequency), color = "darkblue
  labs(title = "Top 6 Ending Locations for Casual Riders", x = NULL, y = NULL, size = "Count") +
  scale_size(range = c(2, 7)) +
  theme(axis.text.x = element_blank(), axis.title.x = element_blank(), axis.text.y = element_blank(), a

# Save the map as a PNG file
ggsave("casual_end_map.png", plot = member_map, width = 8, height = 6, dpi = 300)

plot(member_map)
```

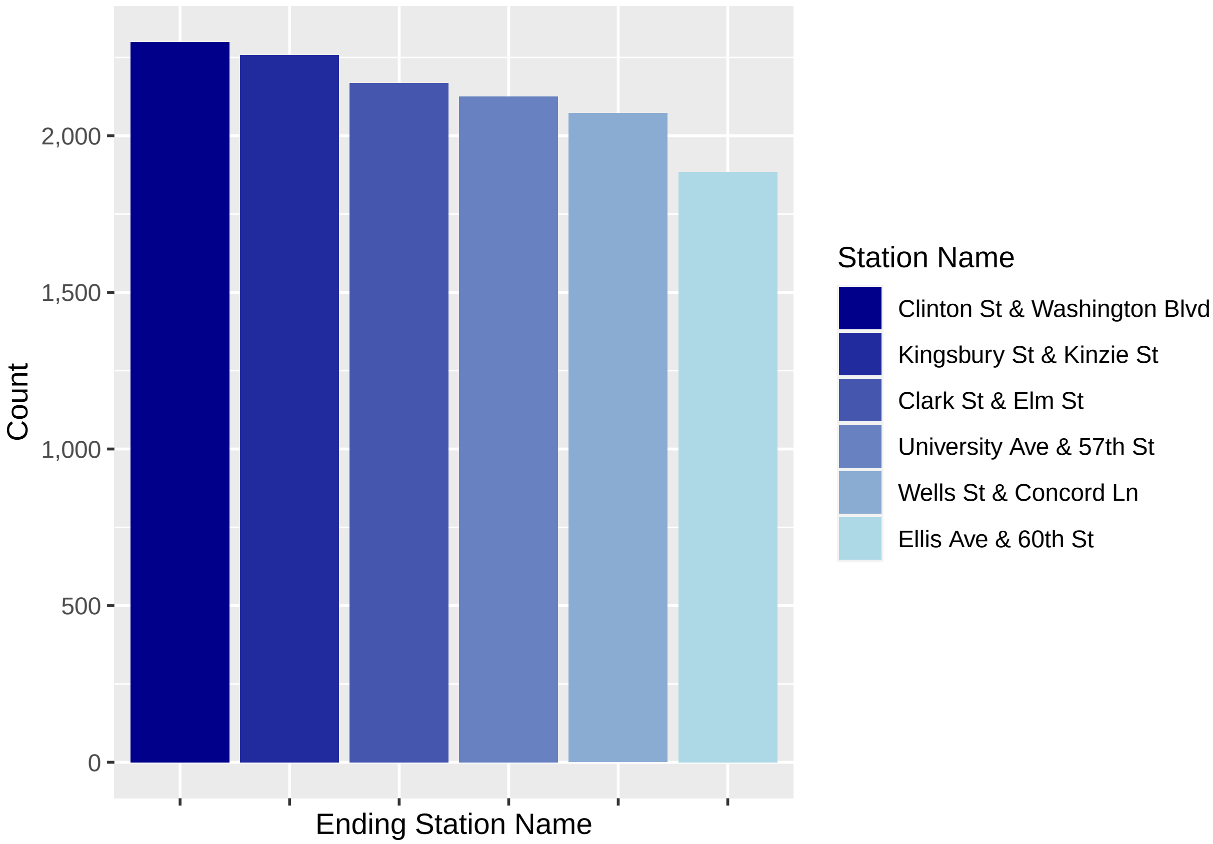


Map

Preference - Casual - Ending Location

```
top_endm <- member_df_end_loc[1:6, c("station_name", "frequency")]
top_endm <- top_endm[order(top_endm$frequency, decreasing = TRUE), ]
top_endm$station_name <- factor(top_endm$station_name, levels = top_endm$station_name)

custom_labels <- top_endm$station_name
colors <- colorRampPalette(c("darkblue", "lightblue"))(length(custom_labels))

ggplot(data = top_endm, aes(x = station_name, y = frequency, fill = station_name)) +
  geom_bar(stat = "identity") +
```

```
  scale_fill_manual(values = colors, labels = custom_labels) +
  labs(x = "Ending Station Name", y = "Count", fill = "Station Name") +
  scale_y_continuous(labels = scales::comma) +
  theme(axis.text.x = element_blank())
```



Barchart

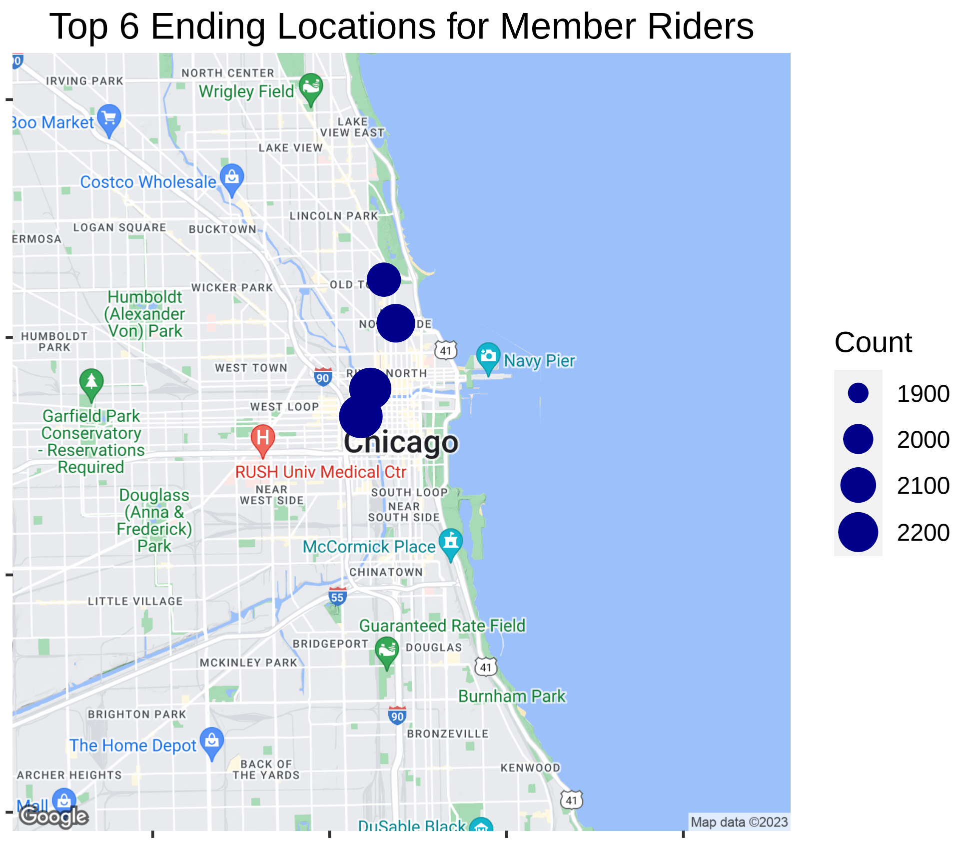
```
top_endm_geo <- member_df_end_loc[1:6, c("station_name", "frequency", "longitude", "latitude")]
top_endm_geo <- top_endm_geo[order(top_endm_geo$frequency, decreasing = TRUE), ]
top_endm_geo$station_name <- factor(top_endm_geo$station_name, levels = top_endm_geo$station_name)

chicago_map <- get_map(location = "Chicago", zoom = 12)

member_map <- ggmap(chicago_map) +
  geom_point(data = top_endm_geo, aes(x = longitude, y = latitude, size = frequency), color = "darkblue
  labs(title = "Top 6 Ending Locations for Member Riders", x = NULL, y = NULL, size = "Count") +
  scale_size(range = c(2, 7)) +
  theme(axis.text.x = element_blank(), axis.title.x = element_blank(), axis.text.y = element_blank(), a

# Save the map as a PNG file
ggsave("member_end_map.png", plot = member_map, width = 8, height = 6, dpi = 300)
```

Map

```
## Warning: Removed 2 rows containing missing values (`geom_point()`).
```

```
plot(member_map)
```

```
## Warning: Removed 2 rows containing missing values (`geom_point()`).
```



Analysis

Our task at hand is to determine the following: How do members and casual riders use Cyclistic bikes differently? Our focus will be on the most recent data set published for the month of May, 2023. We begin our analysis by looking at the differences between our two groups within the following categories:

- Total Ride Percentages
- Preferred Starting Locations
- Preferred Days
- Preferred Starting Times
    - Average Time Traveled
- Preferred Ending Locations

Summary

In summary we begin by noting that relative to the total trips taken for the month of May, we have the about 63.5% of the rides were taken by riders with annual membership. Digging deeper we begin to look at the preferences for our two categories of riders: the casual rider, and the member rider. Our first notable preference is that with both categories of riders, the classic bike is preferred over the electric bike with the members having a stronger preference for the classic bike than the casual rider.

Starting And Ending Locations

For the casual riders we see a that the preferred starting and stopping locations are well aligned, matching in the top six starting and ending locations. This is almost true with our member riders. While they are consistent in their top 5 starting and ending locations, there is a slight difference of preference for the 6th preferred location. Additionally, it is important to note that though the preferred starting and stopping locations are the same, we cannot assume that the frequency of use at each of these stations is defined by the same riders. Another thing to note is the difference in the strength of preference for starting locations as well as the ending locations. For the starting locations, the casual riders' preference strongly leans towards one location: Streeter Dr & Grand Ave., while with the member riders, there is no lean in preference towards one location, specifically. With the ending locations we see the same pattern emerge, where the casual riders strongly prefer one ending location and the top six ending locations for members is pretty uniform.When we take a look at the maps of these locations, we see that the casual riders are preferring starting and ending locations along the lakeshore, whereas our member riders are preferring locations closer to downtown, Chicago.

Days, Time, And Average Travel Time

In regards to the days, time, and average travel time, we begin by looking at the average travel time for our casual riders for the month of May, versus the average travel time for our member riders for that same month. For our casual riders, the average amount of time spent on travel was about 22 minutes and 30 seconds, where the average amount of time spent on travel for our member riders was about 12 minutes and 53 seconds.We then consider the preferred day and preferred time of day. For our casual riders the preferred days were on Wednesday and Thursday with the strongest preference for starting time in the range from 1:00 PM to 5:00 PM. For our member riders the preferred days were on Saturday and Sunday with a bimodal distribution for each, indicating that specific preferred starting time at around 8:00 AM and 5:00 PM, with the stronger preference pointing to 5:00 PM starting time.

Recommendations

Based on these findings my recommendation are as follows

- Consider reducing rate or an eBike-specific membership at a reduced cost to see if there is boosted interest in use of eBikes.
- Consider developing the app to allow customers to track their own time, distance, and favored locations. Additionally, things like Co_2 offset from number of miles rode on a bike instead of driving could help to promote the use of the bike system.
- Offer rewards to long time members as well as new members to promote switching to a membership and then keeping it.

For follow up analysis:

- Consider the rate of change for the bike density at popular locations to see if increasing the availability of bikes can be useful.
- Continue to survey customers on usefulness of bikes and the mobile app in their everyday lives.

Appendix

Citations *Chicago Data Portal (2023). Divvy Bicycle Stations [Data Set]. Open Data Repository. https://data.cityofchicago.org/Transportation/Divvy-Bicycle-Stations/bbyy-e7gq/data* Divvy (2023). divvy-tripdata [Data Set]. Open Data Repository. https://divvy-tripdata.s3.amazonaws.com/index.html