Cyclistic Case Study

Joseph Brockly-Anderson

2023-08-16

Project overview

Cyclistic is a (fictional) bike-sharing service operating out of New York City that successfully launched in 2016. Since their opening, their fleet has grown to 5824 bicycles, all of which are geotracked and connected to a network of nearly 700 stations across the city. From one station, bikes can be unlocked for a small fee and returned to any other station within the network.

People Wanting to use the service can either buy single-ride passes, full-day passes, or annual memberships. Those who buy single ride passes or full day passes are referred to as casual riders, Whereas those who purchase annual memberships are referred to as Cyclistic members. This flexibility of Cyclistic's pricing plans gave it a competitive edge when building the general awareness of the brand and marketing to a broad demographic of potential customers.

Financial analysts at Cyclistic have demonstrated successfully that annual members are much more profitable than casual riders, but why do some people buy single-ride and full-day passes while others purchase annual memberships? Does the length of rides have anything to do with this decision? How can analyzing the past twelve month's worth of ride-share data help Cyclistic create a more targeted campaign ad to convert casual riders into Cyclistic members?

By asking the right questions, keeping our sights focused on the business task, gathering and preparing relevant data for processing, analyzing it to look for any trends or helpful insights, and using simple maps and charts to visualize our analysis, I hope to tell a story that can inform Cyclistic's marketing team of potential next steps in their journey to activate annual members.

In order to provide Cyclistic, their marketing team, executive team, and other relevant stakeholders with valuable, accurate insights and data-driven suggestions to convert casual riders into annual members in pursuit of fiscal growth, I'll be following a programmatic process that includes the following steps:

$$Ask - Prepare - Process - Analyze - Share - Act$$

Each process will iteratively build on the former, though there will be times we go back to a certain step and repeat the process again and again for more clarity. We've already begun asking solid questions: why do some people buy single-ride and full-day passes while others purchase annual memberships? Does the length of rides have anything to do with this decision? How can analyzing the past twelve month's worth of ride-share data help Cyclistic create a more targeted campaign ad to convert casual riders into Cyclistic members?

Data dictionary

Variable	Meaning
ride_id	A signifier used to uniquely identify each bike ride
rideable_type	Whether the bike ridden was electric, classic, or docked

Variable	Meaning
started_at	The date and time the trip was started
ended_at	The date and time the trip was ended
start_station_name	The name of the station where the trip began
start_station_id	A unique identifier given to the station where the trip began
end_station_name	The name of the station where the trip ended
end_station_id	A unique identifier given to the station where the trip ended
start_lat	The latitude of the starting location
start_lng	The longitude of the starting location
end_lat	The latitude of the ending location
end_lng	The longitude of the ending location
member_casual	Whether a rider is casual or an annual member
ride_length	A calculated column containing the ride length in minutes
is_weekday	Whether the ride started on a weekday (TRUE) or weekend (FALSE)
computed_distance	The Haversine distance between trip start and end locations
cluster	What cluster a starting point belongs to

Preparing our data

Calling up the required packages

First, we're going to call in the `tidyverse` collection for it's `readr`, `dplyr`, `tidyr`, and `ggplot2` packages. We'll also call up several other packages that will helps us at varying moments throughout our analysis.

```r
library(tidyverse) # an avowed collection of data analytics packages such as as `readr`
# which helps us pull in data, `dplyr` which helps us manipulate data, `tidyr` which helps us clean dat
library(lubridate) # helps us systematize dates and times
library(data.table) # helps us stack our data
library(validate) # this helps us validate the data
library(assertr) # helps us validate the data
library(knitr)# helps us create aesthetically pleasing tables
library(ggmap) # helps us pull in maps for visualization
library(dbscan) # enables us to perform density-based spatial cluster
```

Pulling in the data

This data was collected and made available by **Motivate LLC** (formerly **Alta Bicycle Share** and also **Motivate International Inc.**). The original collection of data sets contains a total of 13 columns, 5723606 rows, and details data from the months of July 2022 to August 2023, but through our data processing we'll be shaving this down to a data frame of 666 observations.

Combining the datasets into one manageable data frame

In order to clean and analyze this data, we first need to combine all of the rows in each of the 12 data sets into one data frame with 13 columns that correspond to the original data sets' variable types. We can do this with the `list()` and `rbindlist()` functions.

```r
# combining each month's data source into a list
trips_df <-
```

```
  list(df202208, df202209, df202210,
       df202211, df202212, df202301,
       df202302, df202303, df202304,
       df202305, df202306, df202307)

# stacking the rows into a single data frame
df_dirty <- rbindlist(trips_df)

## The code below is specifically for cleaning up our environment once we've pulled all
## the data in.
rm(list = ls(pattern = "^df202")) # this removes all data objects beginning with "df202"
rm(trips_df) # keeping our environment clean
```

Obtaining a sample

In order to make the process of cleaning this data more efficient and less computationally burdensome, we will take a sample size of 666 observations, determined using the sample size calculator at this website. Based on our population, this sample size will give us a confidence level of 99% with a confidence interval of 5.

```
# setting a seed for reproducibility
set.seed(888)

sample_size <- 666

df_sample_dirty <- df_dirty[sample(nrow(df_dirty), sample_size), ]
```

Getting acquainted with the data

Now that we've successfully gathered and sampled our data, this section will help us understand its structure, how many rows and columns our data frame contains, the names of our variables, what they measure, and how these data types are codified. The data frame contains 666 observations (trips taken on a Cyclistic bicycle within the period between August 2022 and July 2023) with 13 separate attributes. As seen below, there are seven string (character) variables, four float (double) variables, and two date/time (POSIXct) variables. A table of summary statistics can be seen below as well.

```
# pulling up the column names
colnames(df_sample_dirty)
```

```
##  [1] "ride_id"            "rideable_type"      "started_at"
##  [4] "ended_at"           "start_station_name" "start_station_id"
##  [7] "end_station_name"   "end_station_id"     "start_lat"
## [10] "start_lng"          "end_lat"            "end_lng"
## [13] "member_casual"
```

```
# glancing at the data types and beginning values
glimpse(df_sample_dirty)
```

```
## Rows: 666
## Columns: 13
## $ ride_id            <chr> "6DBE185CA213103F", "EA4D25B1A74DFDCA", "6720C8EADE~
## $ rideable_type      <chr> "electric_bike", "electric_bike", "electric_bike", ~
## $ started_at         <dttm> 2022-08-14 13:46:03, 2023-03-23 06:53:25, 2022-08-~
## $ ended_at           <dttm> 2022-08-14 13:46:04, 2023-03-23 07:09:06, 2022-08-~
## $ start_station_name <chr> "Lincoln Park Conservatory", "Broadway & Barry Ave"~
## $ start_station_id   <chr> "LP-", "13137", "13300", "TA1309000002", "623", "TA~
## $ end_station_name   <chr> NA, "Elizabeth St & Randolph St", "Burnham Harbor",~
## $ end_station_id     <chr> NA, "23001", "15545", "TA1307000130", "TA1307000128~
## $ start_lat          <dbl> 41.92415, 41.93758, 41.88099, 41.87785, 41.87277, 4~
## $ start_lng          <dbl> -87.63591, -87.64400, -87.61673, -87.62408, -87.623~
## $ end_lat            <dbl> 41.92000, 41.88000, 41.85741, 41.87174, 41.89897, 4~
## $ end_lng            <dbl> -87.64000, -87.66000, -87.61379, -87.65103, -87.629~
## $ member_casual      <chr> "casual", "member", "casual", "casual", "member", "~
```

```
# viewing the first rows
kable(head(df_sample_dirty, 5),
      col.names = c("ID",
                    "Type",
                    "Start Time",
                    "End Time",
                    "Start Station",
                    "Start ID",
                    "End Station",
                    "End ID",
                    "Start Lat",
                    "Start Lng",
                    "End Lat",
                    "End Lng",
                    "Customer")
      )
```

ID	Type	Start Time	End Time	Start Station	Start ID	End Station	End ID	Start Lat	Start Lng	End Lat	End Lng	Customer
6DBE185CA213D3EE	electric_bike	2022-08-14 13:46:03	2022-08-14 13:46:04	Lincoln Park Conservatory	LP-	NA	NA	41.92415	-87.63591	41.92000	-87.64000	casual
EA4D25B1A74D2FCA	electric_bike	2023-03-23 06:53:25	2023-03-23 07:09:06	Broadway & Barry Ave	13137	Elizabeth St & Randolph St	23001	41.93758	-87.64400	41.88000	-87.66000	member
6720C8EADEFFDBE0	electric_bike	2022-08-27 14:32:47	2022-08-27 14:51:05	DuSable Lake Shore Dr & Monroe St	13300	Burnham Harbor	15545	41.88099	-87.61673	41.85741	-87.61379	casual
F2F65886BB45C2D2	classic_bike	2022-08-06 20:46:03	2022-08-06 21:15:34	Michigan Ave & Jackson Blvd	TA1309000002	Morgan St & Polk St	TA1307000130	41.87785	-87.62408	41.87174	-87.65103	casual
8CE93693980C2772	classic_bike	2022-11-30 17:28:01	2022-11-30 17:51:47	Michigan Ave & 8th St	623	Dearborn Pkwy & Delaware Pl	TA1307000128	41.87277	-87.62398	41.89897	-87.62991	member

```
# getting summary statistics
summary(df_sample_dirty)
```

```
##    ride_id          rideable_type         started_at
##  Length:666         Length:666          Min.   :2022-08-01 18:09:52.00
##  Class :character   Class :character    1st Qu.:2022-09-24 12:13:38.00
##  Mode  :character   Mode  :character   Median :2023-01-20 19:12:55.00
##                                        Mean   :2023-01-28 18:03:51.58
##                                        3rd Qu.:2023-06-02 21:54:57.00
##                                        Max.   :2023-07-31 19:07:31.00
##
##     ended_at                        start_station_name start_station_id
##  Min.   :2022-08-01 18:14:39.00   Length:666         Length:666
##  1st Qu.:2022-09-24 12:22:14.75   Class :character   Class :character
##  Median :2023-01-20 19:22:21.50   Mode  :character   Mode  :character
##  Mean   :2023-01-28 18:23:46.73
##  3rd Qu.:2023-06-02 21:57:46.25
##  Max.   :2023-07-31 19:26:18.00
##
##  end_station_name   end_station_id       start_lat       start_lng
##  Length:666         Length:666         Min.   :41.75   Min.   :-87.78
##  Class :character   Class :character   1st Qu.:41.88   1st Qu.:-87.66
##  Mode  :character   Mode  :character   Median :41.90   Median :-87.64
##                                        Mean   :41.90   Mean   :-87.65
##                                        3rd Qu.:41.93   3rd Qu.:-87.63
##                                        Max.   :42.06   Max.   :-87.55
##
##     end_lat         end_lng       member_casual
##  Min.   :41.71   Min.   :-87.77   Length:666
##  1st Qu.:41.88   1st Qu.:-87.66   Class :character
##  Median :41.90   Median :-87.65   Mode  :character
##  Mean   :41.90   Mean   :-87.65
##  3rd Qu.:41.93   3rd Qu.:-87.63
##  Max.   :42.05   Max.   :-87.55
##  NA's   :1       NA's   :1
```

Cleaning the data

Checking for duplicates

In order to perform an accurate analysis, we need to process the data to make sure it contains no incorrect, duplicate, or unaccounted for observations. We also need to manage any gaps in the information and make sure the formatting and data types remain consistent across our data sources. To do this, we'll first check for duplicate observations with the `duplicated()` function.

```r
# every observation should be unique
sum(duplicated(df_dirty))
```

```
## [1] 0
```

```r
# there should be three types of bike
n_distinct(df_sample_dirty$rideable_type)
```

```
## [1] 3
```

```r
# there should be two types of rider
uniqueN(df_sample_dirty$member_casual)
```

```
## [1] 2
```

```r
# there should be as many unique ride IDs as there are total observations
n_distinct(df_sample_dirty$ride_id)
```

```
## [1] 666
```

Checking for null values

Let’s now check for missing values, which we can deal with by thinking critically about their context and making use of professional outreach and statistical modeling techniques (depending on the type of data that we may need to find). When we execute these checks, we discover this data set has quite a large amount of null values (a total of), 99.5% of which occur in just 4 columns related to station names and IDs. Another small amount of null values occurs in the `end_lat` and `end_lng` columns.

```r
sum(is.na(df_sample_dirty))
```

```
## [1] 402
```

```r
sum(is.na(df_sample_dirty$ride_id))
```

```
## [1] 0
```

```r
sum(is.na(df_sample_dirty$rideable_type))
```

```
## [1] 0
```

```r
sum(is.na(df_sample_dirty$started_at))
```

```
## [1] 0
```

```r
sum(is.na(df_sample_dirty$ended_at))
```

```
## [1] 0
```

```r
sum(is.na(df_sample_dirty$start_station_name))
```

```
## [1] 99
```

```r
sum(is.na(df_sample_dirty$start_station_id))
```

```
## [1] 99
```

```r
sum(is.na(df_sample_dirty$end_station_name))
```

```
## [1] 101
```

```r
sum(is.na(df_sample_dirty$end_station_id))
```

```
## [1] 101
```

```r
sum(is.na(df_sample_dirty$start_lat))
```

```
## [1] 0
```

```r
sum(is.na(df_sample_dirty$start_lng))
```

```
## [1] 0
```

```r
sum(is.na(df_sample_dirty$end_lat))
```

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

```r
sum(is.na(df_sample_dirty$end_lng))
```

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

```r
sum(is.na(df_sample_dirty$member_casual))
```

```
## [1] 0
```

What could be causing this lack of data? Will these null values jeopardize our analysis or are they sufficiently random? After further investigation, it has come to light that Motivate LLC 's bike fleet is outfitted with on-board GPS trackers to pinpoint their location from within the entire service area. The fleet is technically designated as a "dockless" one (though they do manage a small sub-fleet of "docked" bikes. From this information, we can conclude why null values appear in the start station name, start station ID, end station name, and end station ID columns: neither casual riders nor annual members need to pick up or drop off bikes at designated stations. This makes dealing with these null values significantly simpler in our analysis, but what about the missing value in bike trips' ending locations?

It seems reasonable to believe that this missing values could be coming from:

- human error on part of the database gathering systems in use at Cyclistic
- technical issues with GPS tracking caused by electronic malfunctions
- user-related error emerging from a failure to properly conclude bike trips

We've gone ahead and checked the starting and ending coordinates to check if they are correlated for non-missing values. There is a moderate correlation between starting latitude and ending latitude, but nothing has pointed us to a belief this null value was systematically incorporated into the data. From this, we're going to assume that this null value's presence is sufficiently random enough to avoid complications in our analysis, though further investigation may reveal the necessity to impute the value in the future. With this in mind, we've gone ahead and ignored the observation which accounts for ~0.15% of our observation total.

Removing Leading/Trailing Spaces

Leading and trailing spaces in a character string can frustrate our attempts as analysts to capture that data and manipulate it in our analysis. We need some way to find and remove leading and trailing spaces within the data frame that we're working with. This is a solid practice to safeguard the quality of our data before analysis, streamline our troubleshooting process, and avoid any setbacks later on. To do this, we can use the `stringr` package. NOTE: To prevent the lengthening of our final document when knitted to PDF, I've gone ahead and hidden the bulk of this code, though my process can be identified within the first chunk.

```r
# the `str_trim()` function only works on string data
df_sample_dirty$ride_id <-
  str_trim(df_sample_dirty$ride_id,
           "left")
df_sample_dirty$ride_id <-
  str_trim(df_sample_dirty$ride_id,
           "right")
```

```r
df_sample_dirty$rideable_type <-
  str_trim(df_sample_dirty$rideable_type,
           "left")
df_sample_dirty$rideable_type <-
  str_trim(df_sample_dirty$rideable_type,
           "right")

df_sample_dirty$start_station_name <-
  str_trim(df_sample_dirty$start_station_name,
           "left")
df_sample_dirty$start_station_name <-
  str_trim(df_sample_dirty$start_station_name,
           "right")

df_sample_dirty$start_station_id <-
  str_trim(df_sample_dirty$start_station_id,
           "left")
df_sample_dirty$start_station_id <-
  str_trim(df_sample_dirty$start_station_id,
           "right")

df_sample_dirty$end_station_name <-
  str_trim(df_sample_dirty$end_station_name,
           "left")
df_sample_dirty$end_station_name <-
  str_trim(df_sample_dirty$end_station_name,
           "right")

df_sample_dirty$end_station_id <-
  str_trim(df_sample_dirty$end_station_id,
           "left")
df_sample_dirty$end_station_id <-
  str_trim(df_sample_dirty$end_station_id,
           "right")

df_sample_dirty$member_casual <-
  str_trim(df_sample_dirty$member_casual,
           "left")
df_sample_dirty$member_casual <-
  str_trim(df_sample_dirty$member_casual,
           "right")
```

Ensuring proper formats

As a further measure of data processing, we'll make sure our columns are structured correctly. This is an invaluable part of keeping an organized project and will save us time troubleshooting should we encounter bugs in our code during analysis and visualization.

Date/time

```r
## the POSIXct stores date/time values
is.POSIXct(df_sample_dirty$started_at)
```

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

```r
is.POSIXct(df_sample_dirty$ended_at)
```

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

Strings

```r
is.character(df_sample_dirty$ride_id)
```

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

```r
is.character(df_sample_dirty$rideable_type)
```

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

```r
is.character(df_sample_dirty$start_station_name)
```

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

```
is.character(df_sample_dirty$start_station_id)
```

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

```
is.character(df_sample_dirty$end_station_name)
```

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

```
is.character(df_sample_dirty$end_station_id)
```

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

```
is.character(df_sample_dirty$member_casual)
```

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

Numeric

```
# a "double" atomic vector is commonly referred to as a "float" in other languages and programs
is.double(df_sample_dirty$start_lat)
```

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

```
is.double(df_sample_dirty$start_lng)
```

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

```
is.double(df_sample_dirty$end_lat)
```

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

```
is.double(df_sample_dirty$end_lng)
```

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

Data Validation

We need to validate each column to ensure that it contains the range of values that we assume actually exists within the data set.

```
## validating `ride_id`
# setting the number of characters to `v_ride_id`
valid_ride_id <- nchar(
  df_sample_dirty$ride_id
  )

sum(
  valid_ride_id == 16
  ) # the number of TRUEs should match the number of total observations
```

```
## [1] 666
```

```
## validating `rideable_type` and `member_casual`
# creating vectors of allowed values
valid_rideable <- c("electric_bike", "classic_bike", "docked_bike")

valid_m_c <- c("member", "casual")


# returning an error if any cell doesn't contain appropriate values
```

```r
stopifnot(all(df_sample_dirty$rideable_type %in% valid_rideable))

stopifnot(all(df_sample_dirty$member_casual %in% valid_m_c))


## validating `started_at` and `ended_at`
# determining minimum and maximum values
max(df_sample_dirty$started_at)
```

```
## [1] "2023-07-31 19:07:31 UTC"
```

```r
min(df_sample_dirty$started_at)
```

```
## [1] "2022-08-01 18:09:52 UTC"
```

```r
max(df_sample_dirty$ended_at)
```

```
## [1] "2023-07-31 19:26:18 UTC"
```

```r
min(df_sample_dirty$started_at)
```

```
## [1] "2022-08-01 18:09:52 UTC"
```

```r
sum(df_sample_dirty$started_at < "2022-07-01 00:00:00")
```

```
## [1] 0
```

```r
sum(df_sample_dirty$ended_at > "2023-08-31 11:59:59")
```

```
## [1] 0
```

```r
# defining the date/time range
start_date <- as.POSIXct("2022-07-01 00:00:00")
end_date <- as.POSIXct("2023-08-31 11:59:59")

# filtering out unnecessary observations
df_sample_dirty <- subset(df_sample_dirty,
                      started_at >= start_date &
                        started_at <= end_date &
                        ended_at >= start_date &
                        ended_at <= end_date)
```

Presenting the cleaned data

So far, we have called up the required packages, pulled in the data relevant to providing insights in favor of our business task with Cyclistic, merged the data sets together into one data frame, obtained a subset of that data based on an appropriate sample size, made sure that no duplicates exist, explored the context of present null values, removed leading and trailing spaces in character columns, ensured that the attributes are properly formatted, and verified that our data exists within the boundaries we expect. Our data is now clean and ready for a thorough analysis. We've printed a small table of the clean data below to get a feel for what we're now working with, but we've left out the ID attributes for ease of readability.

```r
# renaming our data frame from ease of use
df_clean <- df_sample_dirty


# cleaning up our environment
rm(df_dirty, df_sample_dirty)
```

```
# selecting desired columns to present in a summary table
df_clean_selected <- df_clean %>%
  select("rideable_type",
         "started_at",
         "ended_at",
         "start_station_name",
         "end_station_name",
         "start_lat",
         "start_lng",
         "end_lat",
         "end_lng",
         "member_casual")

# shortening "Type" values for readability
df_clean_selected$rideable_type <- recode(df_clean_selected$rideable_type,
       "electric_bike" = "Electric",
       "docked_bike" = "Docked",
       "classic_bike" = "Classic")

kable(head(df_clean_selected, 5),
      col.names = c("Type",
                    "Start Time",
                    "End Time",
                    "Start Station",
                    "End Station",
                    "Start Lat",
                    "Start Lng",
                    "End Lat",
                    "End Lng",
                    "Customer"),
      caption = "Cyclistic Bike Trip Data",
      digits = 2)
```

Table 3: Cyclistic Bike Trip Data

Type	Start Time	End Time	Start Station	End Station	Start Lat	Start Lng	End Lat	End Lng	Customer
Electric	2022-08-14 13:46:03	2022-08-14 13:46:04	Lincoln Park Conservatory	NA	41.92	-87.64	41.92	-87.64	casual
Electric	2023-03-23 06:53:25	2023-03-23 07:09:06	Broadway & Barry Ave	Elizabeth St & Randolph St	41.94	-87.64	41.88	-87.66	member
Electric	2022-08-27 14:32:47	2022-08-27 14:51:05	DuSable Lake Shore Dr & Monroe St	Burnham Harbor	41.88	-87.62	41.86	-87.61	casual
Classic	2022-08-06 20:46:03	2022-08-06 21:15:34	Michigan Ave & Jackson Blvd	Morgan St & Polk St	41.88	-87.62	41.87	-87.65	casual

Classic	2022-11-30 17:28:01	2022-11-30 17:51:47	Michigan Ave & 8th St	Dearborn Pkwy & Delaware Pl	41.87	-87.62	41.90	-87.63	member

Analyzing and visualizing the data

Now that the data has been thoroughly cleaned, we can use it to generate data driven insights that will help Cyclistic's marketing team convert casual riders into annual members. Why do some people prefer buying day passes and single ride passes while others prefer purchasing annual memberships?

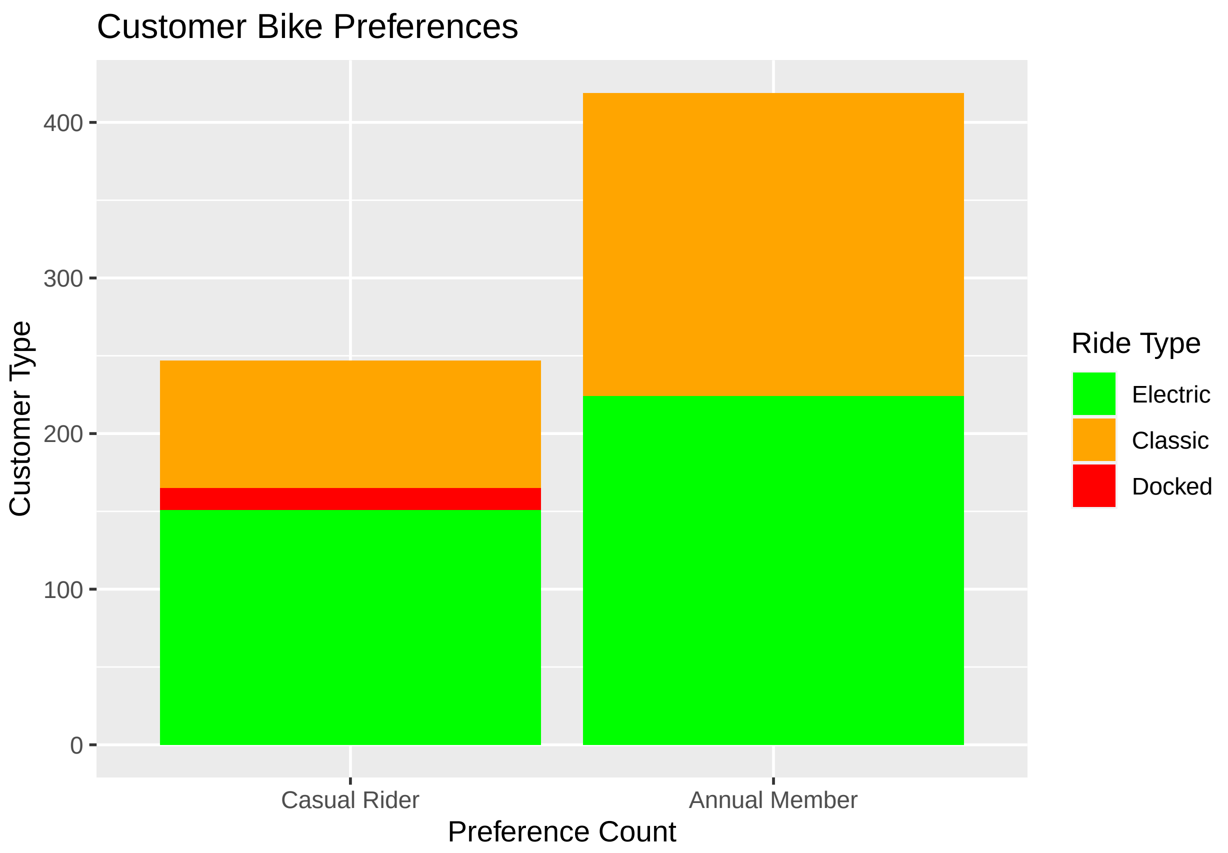
Do they prefer a specific type of bike?

Casual riders and annual members both prefer to ride electric bikes, but annual members do still enjoy riding classic bicycles. The one thing they cannot stand is the restrictive nature of docked bikes. **We know with that annual Cyclistic members never ride docked bikes, so the marketing should avoid featuring them in their advertising campaigns**

```
bike_pref <- df_clean %>%
  group_by(member_casual, rideable_type) %>%
  summarise(pref_count = n()) %>%
  arrange(member_casual, desc(pref_count)) %>%
  slice(1) %>%
  ungroup()

kable(head(bike_pref),
      col.names = c("Customer Type", "Preferred Bike", "Preference Count"))
```

Customer Type	Preferred Bike	Preference Count
casual	electric_bike	151
member	electric_bike	224

```
# visualizing preferences
ggplot(df_clean, aes(x = member_casual, fill = rideable_type)) +
  geom_bar() +
  labs(title = "Customer Bike Preferences",
       x = "Preference Count",
       y = "Customer Type",
       fill = "Ride Type") +
  scale_x_discrete(labels = c(casual = "Casual Rider",
                              member = "Annual Member")) +
  scale_fill_manual(values = c("green",
                               "orange",
                               "red"),
                    breaks = c("electric_bike",
                               "classic_bike",
                               "docked_bike"),
                    labels = c("Electric",
                               "Classic",
                               "Docked"))
```



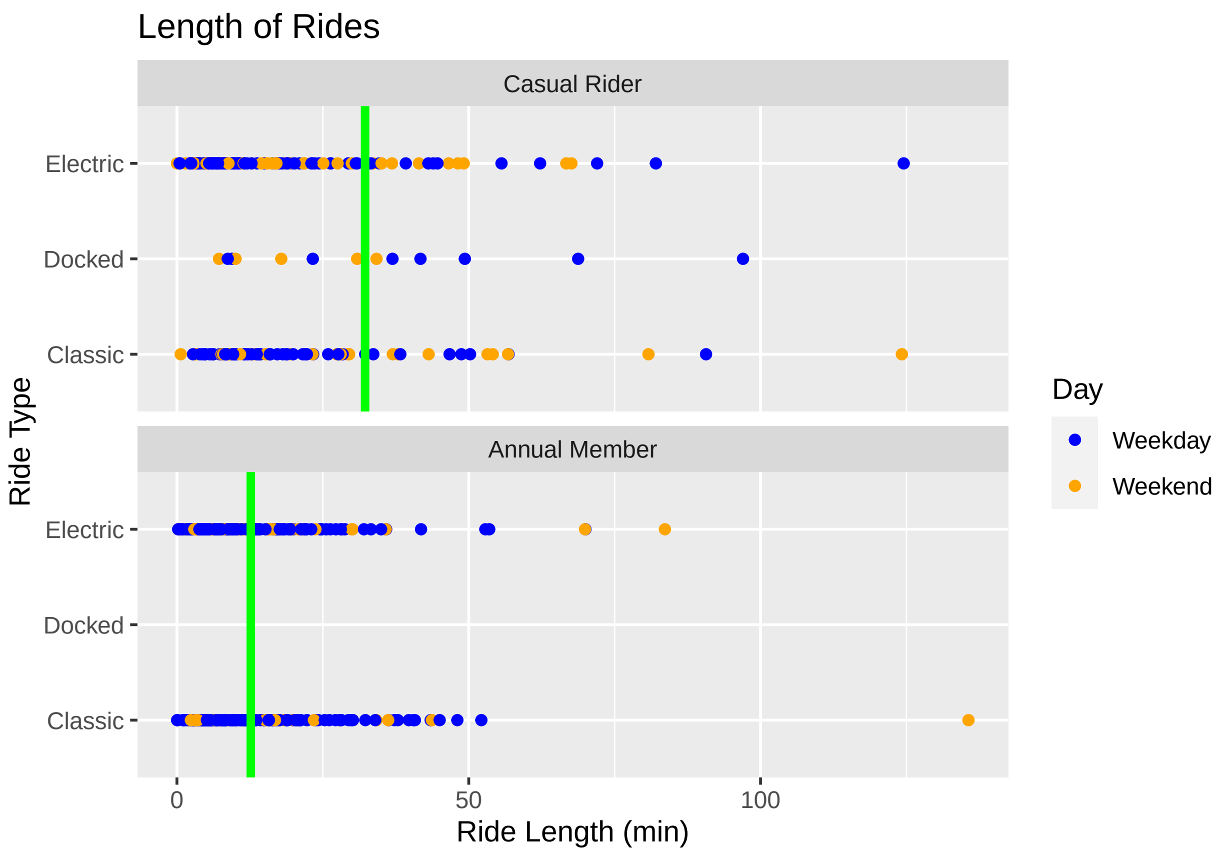
Trip duration analysis

Does the length of bike rides have anything to do with this decision? Casual members take bike trips that are nearly three times longer than those taken by annual members. **Annual members take shorter bike trips.** Is this because annual members aren't using Cyclistic's services for scenic bike rides? Do they take shorter bike rides because they need to take more trips? A further analysis will help us speculate potential answers.

```r
# calculating ride length in minutes
df_clean <- df_clean %>%
  mutate(ride_length = as.numeric(difftime(ended_at, started_at, units = "mins")))

# grouping data by customer type
average_ride_length <- df_clean %>%
  group_by(member_casual) %>%
  summarise(avg_ride_length = mean(ride_length, na.rm = TRUE))

kable(head(average_ride_length),
      col.names = c("Customer Type", "Average Length (min)"),
      digits = 1)
```

Customer Type	Average Length (min)
casual	32.2
member	12.7

```r
# adding a day of the week column
df_clean <- df_clean %>%
  mutate(is_weekday = weekdays(started_at) %in%
           c("Monday",
             "Tuesday",
             "Wednesday",
             "Thursday",
             "Friday"))

# visualizing ride length
df_clean %>%
  filter(ride_length < 500) %>%
  ggplot(aes(x = ride_length,
             y = rideable_type,
             color = (is_weekday))) +
  geom_point() +
  geom_vline(data = subset(
    average_ride_length, member_casual == "casual"),
             aes(xintercept = avg_ride_length),
             linetype = "solid",
             color = "green",
             linewidth = 1.5) +
  geom_vline(data = subset(average_ride_length, member_casual == "member"),
             aes(xintercept = avg_ride_length),
             linetype = "solid",
             color="green",
             linewidth = 1.5) +
  facet_wrap(~ member_casual,
             nrow = 2,
             labeller =
               labeller(member_casual =
                                   c("casual" = "Casual Rider", "member" = "Annual Member"))) +
  labs(title = "Length of Rides",
       x = "Ride Length (min)",
       y = "Ride Type",
       color = "Day") +
  scale_y_discrete(labels = c("electric_bike" = "Electric",
                              "classic_bike" = "Classic",
                              "docked_bike" = "Docked")) +
  scale_color_manual(values = c("blue", "orange"),
                     breaks = c("TRUE", "FALSE"),
                     labels = c("Weekday", "Weekend"))
```



Trip frequency analysis

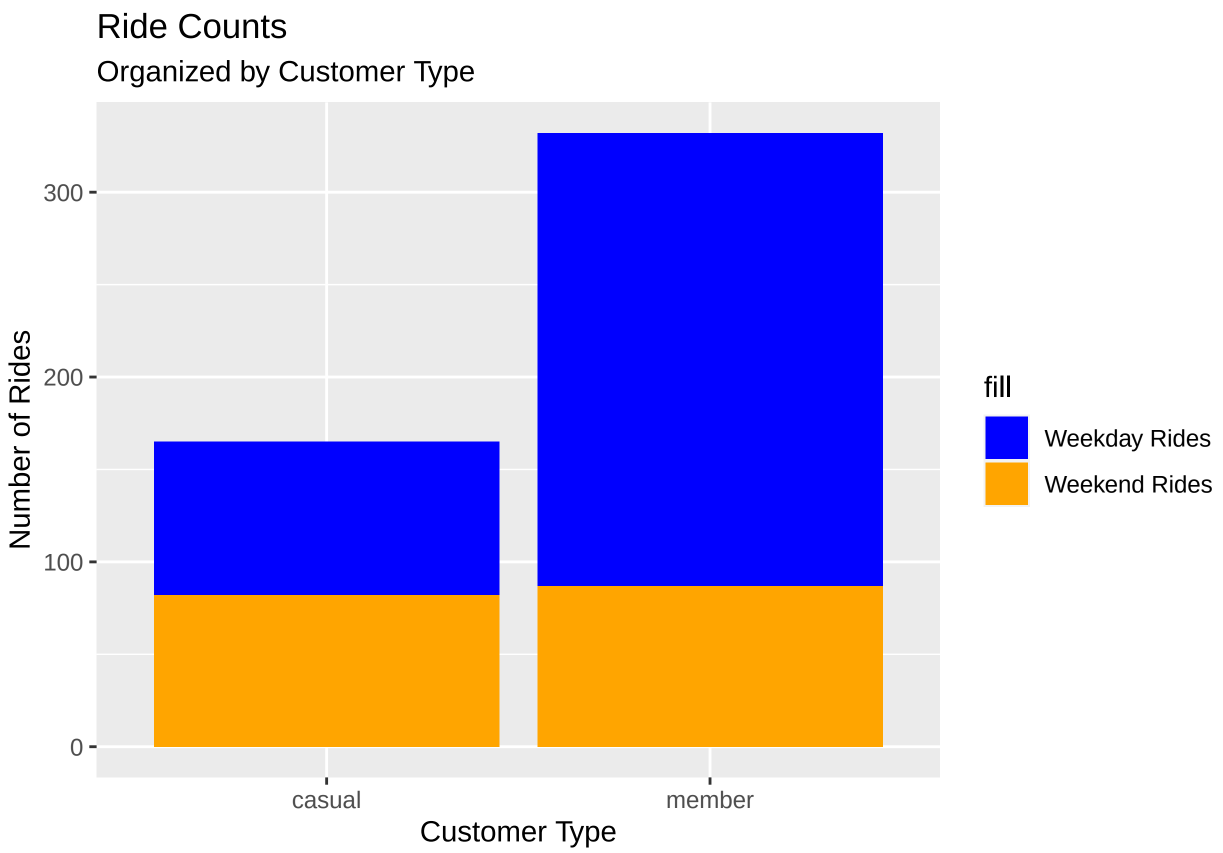
Do annual members ride Cyclistic bikes more often and, if so, by how much? A simple analysis proves that members ride Cyclistic bicycles nearly twice as often as casual riders. Annual members tend to ride the bicycles more often on weekdays.

```
# calculating the number of rides taken by each customer type
ride_counts <- df_clean %>%
  group_by(member_casual) %>%
  summarise(ride_count = n()) %>%
  mutate(average_rides_perm = ride_count / as.numeric(((max(df_clean$started_at) - min(df_clean$started_

# rides taken by customer by weekday/weekend
weekday_rides <- df_clean %>%
  filter(is_weekday) %>%
  group_by(member_casual) %>%
  summarise(weekday_ride_count = n())

weekend_rides <- df_clean %>%
  filter(!is_weekday) %>%
  group_by(member_casual) %>%
  summarise(weekend_ride_count = n())

# merging the new data frames
ride_counts <- ride_counts %>%
  left_join(weekend_rides, by = "member_casual") %>%
  left_join(weekday_rides, by = "member_casual")

# calculating weekday trip percentages
weekday_ride_percentage <- ride_counts %>%
  group_by(member_casual) %>%
  summarise(
    weekday_percentage = sum(weekday_ride_count) / sum(weekday_ride_count + weekend_ride_count) * 100
  )

# joining weekday trip percentages
ride_counts <- ride_counts %>%
  left_join(weekday_ride_percentage,
            by = "member_casual")

# making a table with bike counts
kable(head(ride_counts),
      col.names = c("Customer Type", "Rides Taken", "Rides Taken Per Month", "Weekday Trips", "Weekend T
      caption = "Trip Frequency",
      digits = 1)
```

Table 6: Trip Frequency

Customer Type	Rides Taken	Rides Taken Per Month	Weekday Trips	Weekend Trips	Weekday %
casual	247	20.3	82	165	66.8
member	419	34.4	87	332	79.2

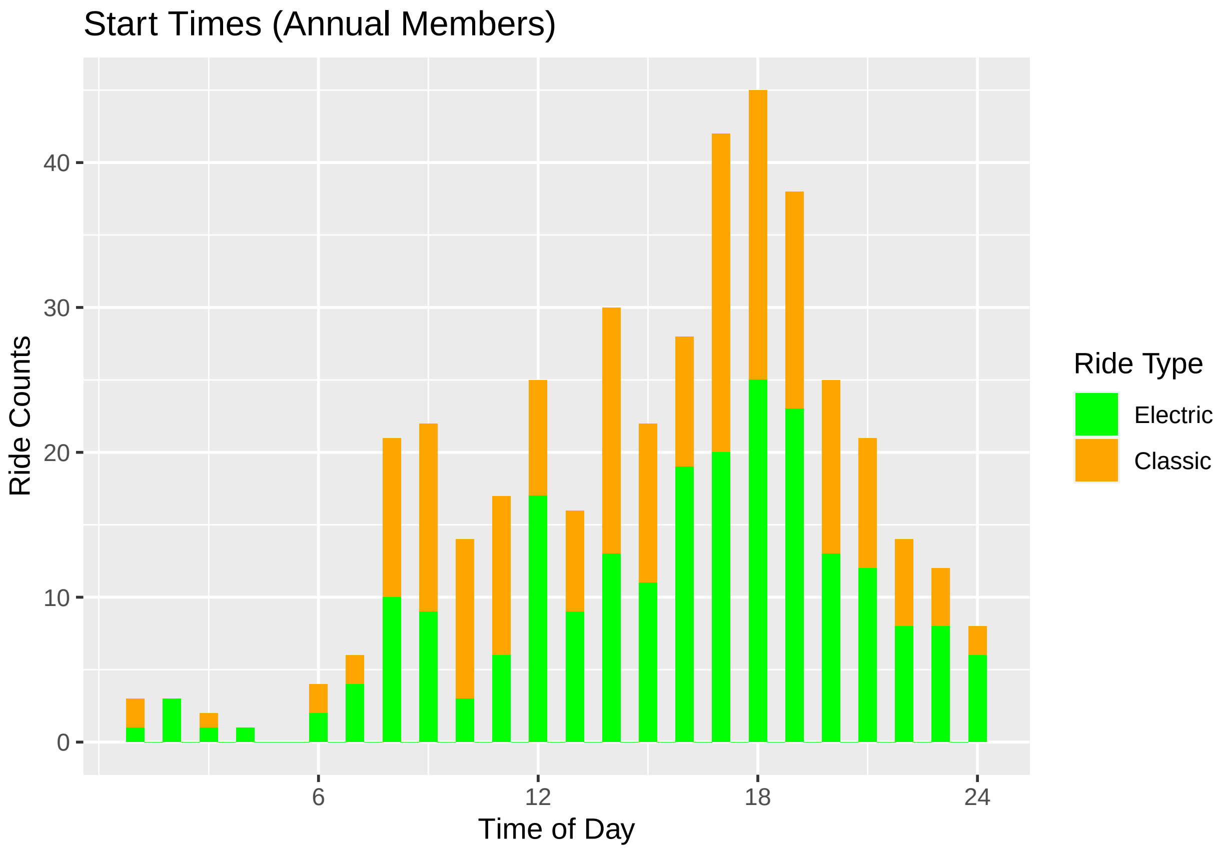
```
# plotting the data
ggplot(ride_counts, aes(x = member_casual)) +
  geom_bar(aes(y = weekday_ride_count,
               fill = "Weekday Rides"),
           stat = "identity") +
  geom_bar(aes(y = weekend_ride_count,
               fill = "Weekend Rides"),
           stat = "identity",
           position = "dodge") +
  labs(title = "Ride Counts",
       subtitle = "Organized by Customer Type",
       x = "Customer Type",
       y = "Number of Rides") +
  scale_fill_manual(
    values = c("Weekday Rides" = "blue",
               "Weekend Rides" = "orange"))
```



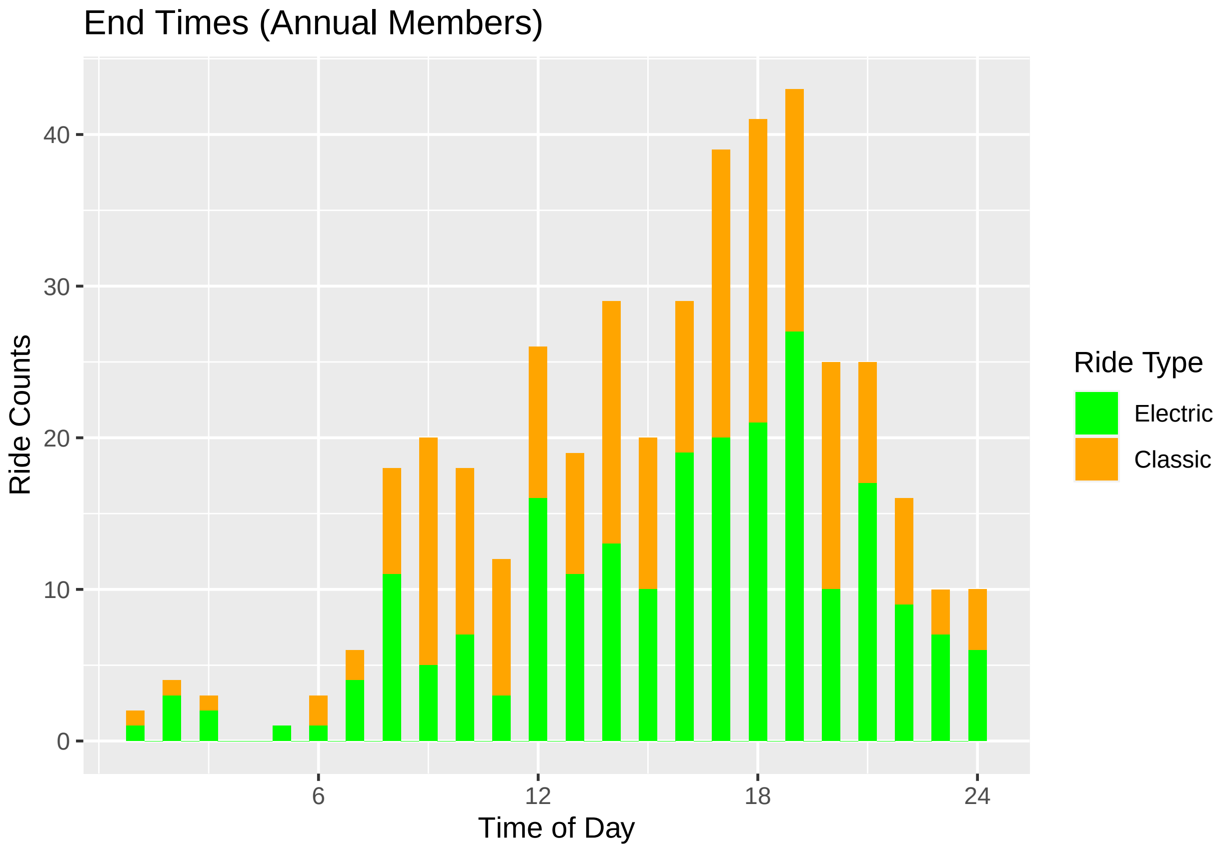
Trip time analysis

When do these two groups start their bike rides? When do they end them? Are there any seasonal trends we can look at to help us answer these questions?

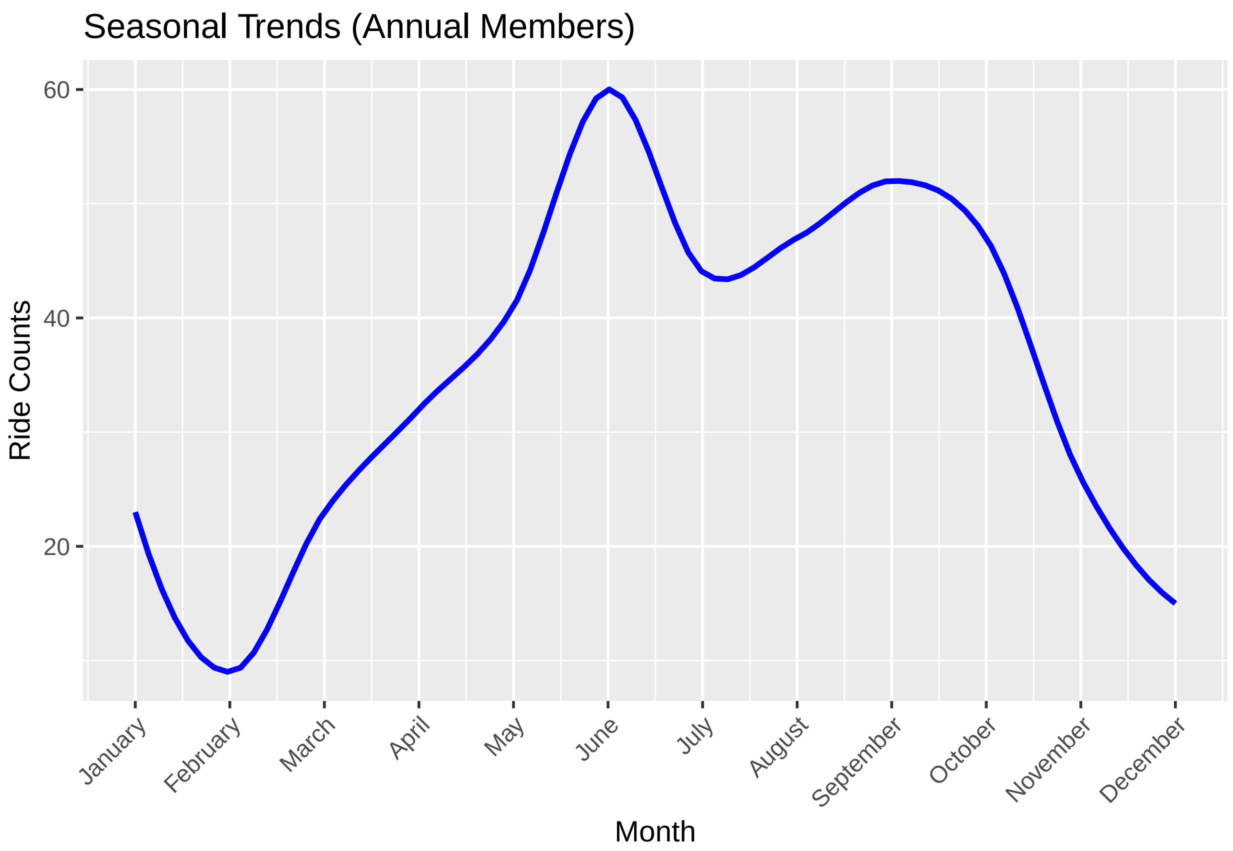
```
## annual members
member_df <- df_clean %>%
  filter(member_casual == "member")

# start time histogram
ggplot(member_df,
       aes(x = hour(started_at),
           fill = rideable_type)) +
  geom_histogram(binwidth = 0.5) +
  labs(
    title = "Start Times (Annual Members)",
    x = "Time of Day",
    y = "Ride Counts",
    fill = "Ride Type") +
  scale_fill_manual(values = c("green",
                               "orange",
                               "red"),
                    breaks = c("electric_bike",
                               "classic_bike",
                               "docked_bike"),
                    labels = c("Electric",
                               "Classic",
                               "Docked")) +
  scale_x_continuous(breaks = seq(5, 23, by = 6),
                   labels = seq(6, 24, by = 6)
                   )
```



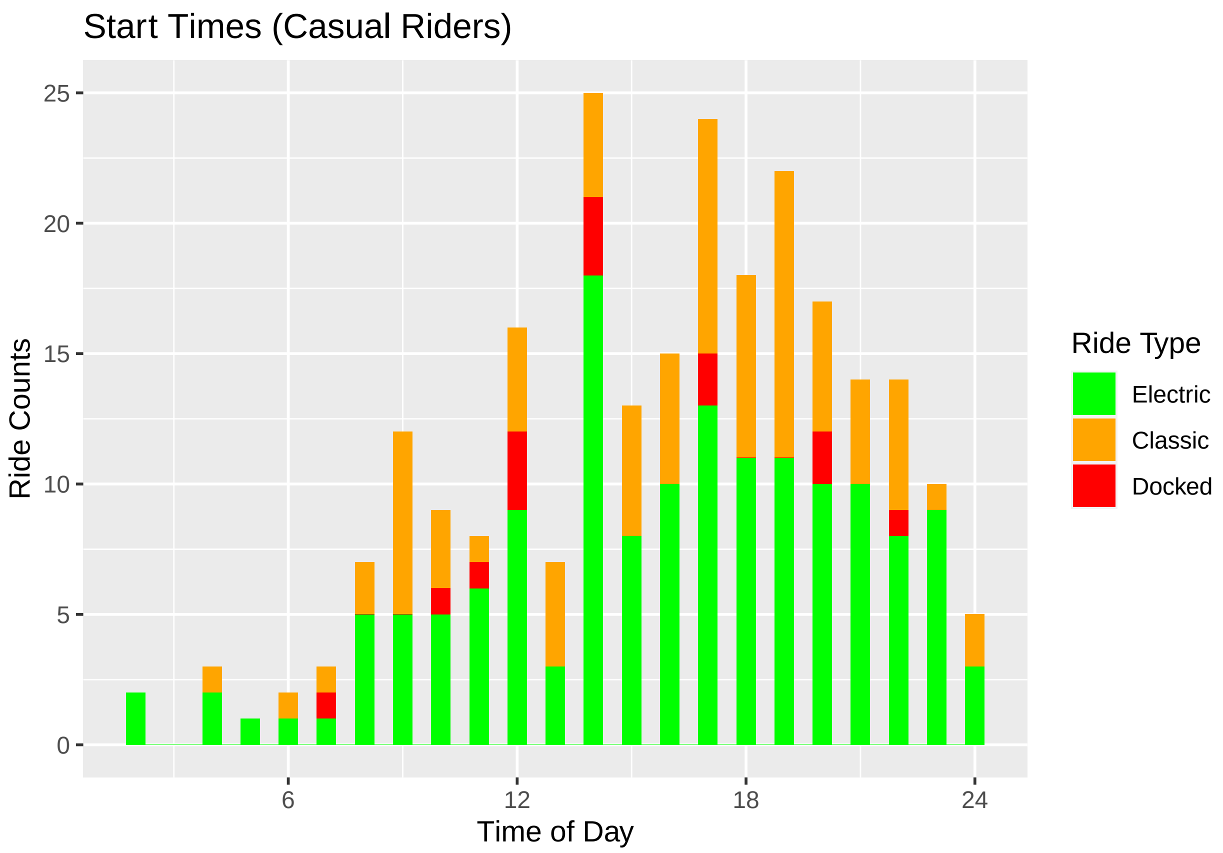
```
# end time histogram
ggplot(member_df,
       aes(x = hour(ended_at),
           fill = rideable_type)) +
  geom_histogram(binwidth = 0.5) +
  labs(
    title = "End Times (Annual Members)",
    x = "Time of Day",
    y = "Ride Counts",
    fill = "Ride Type") +
  scale_fill_manual(values = c("green",
                               "orange",
                               "red"),
                    breaks = c("electric_bike",
                               "classic_bike",
                               "docked_bike"),
                    labels = c("Electric",
                               "Classic",
                               "Docked")) +
  scale_x_continuous(breaks = seq(5, 23, by = 6),
                   labels = seq(6, 24, by = 6)
                   )
```



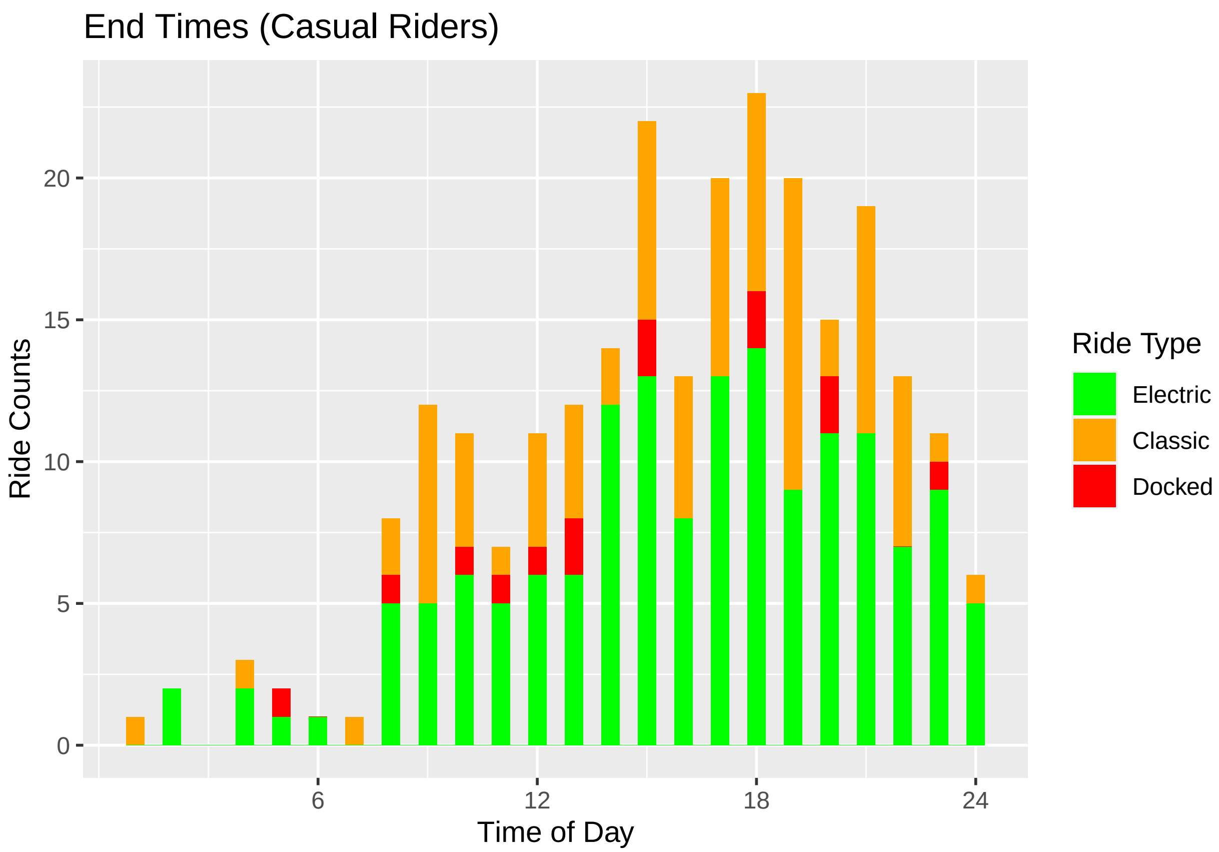
```
# seasonal trends
member_df %>%
  mutate(month = month(started_at)) %>%
  group_by(month) %>%
  summarise(ride_count = n()) %>%
  ggplot(aes(x = month, y = ride_count)) +
  geom_smooth(method = "loess",
              span = 0.35,
              se = FALSE,
              color = "blue",
              linewidth = 1) +
  labs(title = "Seasonal Trends (Annual Members)",
       x = "Month",
       y = "Ride Counts") +
  scale_x_continuous(breaks = c(1, 2, 3, 4, 5, 6,
                                7, 8, 9, 10, 11, 12),
                     labels = c("January","February","March",
                                "April","May","June","July",
                                "August","September","October",
                                "November","December")) +
  theme(axis.text.x = element_text(angle = 45,
                                   hjust = 1)
        )
```



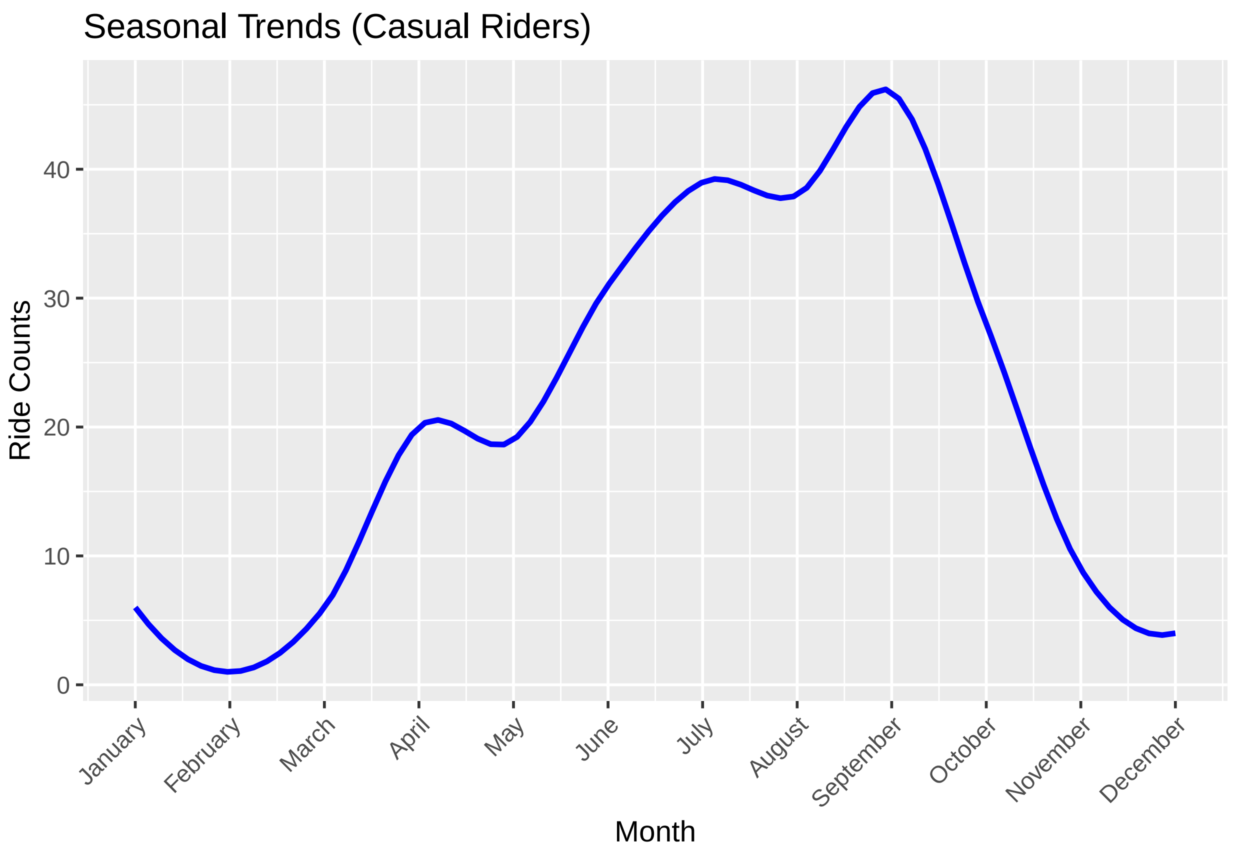
```r
## casual riders
casual_df <- df_clean %>%
  filter(member_casual == "casual")

# start time histogram
ggplot(casual_df,
       aes(x = hour(started_at),
           fill = rideable_type)) +
  geom_histogram(binwidth = 0.5) +
  labs(
    title = "Start Times (Casual Riders)",
    x = "Time of Day",
    y = "Ride Counts",
    fill = "Ride Type") +
  scale_fill_manual(values = c("green",
                               "orange",
                               "red"),
                    breaks = c("electric_bike",
                               "classic_bike",
                               "docked_bike"),
                    labels = c("Electric",
                               "Classic",
                               "Docked")) +
  scale_x_continuous(breaks = seq(5, 23, by = 6),
                   labels = seq(6, 24, by = 6)
                   )
```



```
# end time histogram
ggplot(casual_df,
       aes(x = as.POSIXct(hour(ended_at)),
           fill = rideable_type)) +
  geom_histogram(binwidth = 0.5) +
  labs(
    title = "End Times (Casual Riders)",
    x = "Time of Day",
    y = "Ride Counts",
    fill = "Ride Type") +
  scale_fill_manual(values = c("green",
                               "orange",
                               "red"),
                    breaks = c("electric_bike",
                               "classic_bike",
                               "docked_bike"),
                    labels = c("Electric",
                               "Classic",
                               "Docked")) +
  scale_x_continuous(breaks = seq(5, 23, by = 6),
                   labels = seq(6, 24, by = 6)
                   )
```



```
# seasonal trends
casual_df %>%
  mutate(month = month(started_at)) %>%
  group_by(month) %>%
  summarise(ride_count = n()) %>%
  ggplot(aes(x = month, y = ride_count)) +
  geom_smooth(method = "loess",
              span = 0.35,
              se = FALSE,
              color = "blue",
              linewidth = 1) +
  labs(title = "Seasonal Trends (Casual Riders)",
       x = "Month",
       y = "Ride Counts") +
  scale_x_continuous(breaks = c(1, 2, 3, 4, 5, 6,
                                7, 8, 9, 10, 11, 12),
                     labels = c("January","February","March",
                                "April","May","June","July",
                                "August","September","October",
                                "November","December")) +
  theme(axis.text.x = element_text(angle = 45,
                                   hjust = 1)
        )
```



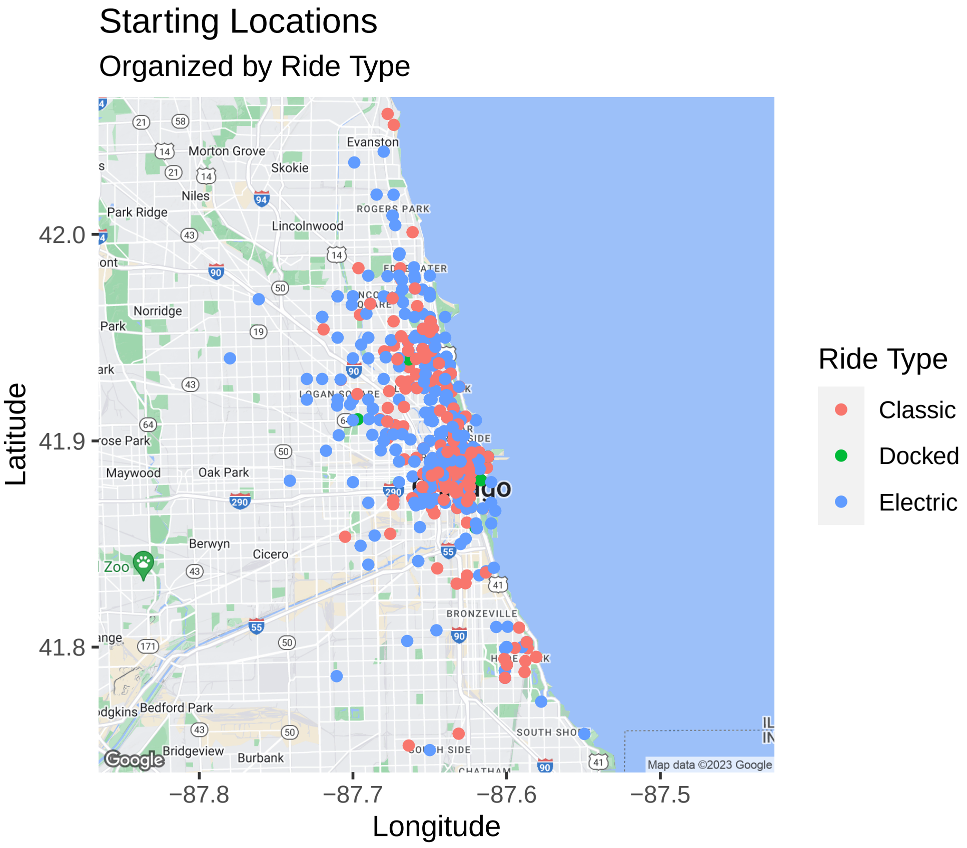
Trip location analysis

Our data shows that the top 10 most popular starting stations are all in and around Hyde Park, Chicago. There is also a dense concentration of starting locations for both casual riders and annual members along the Central Lakefront.

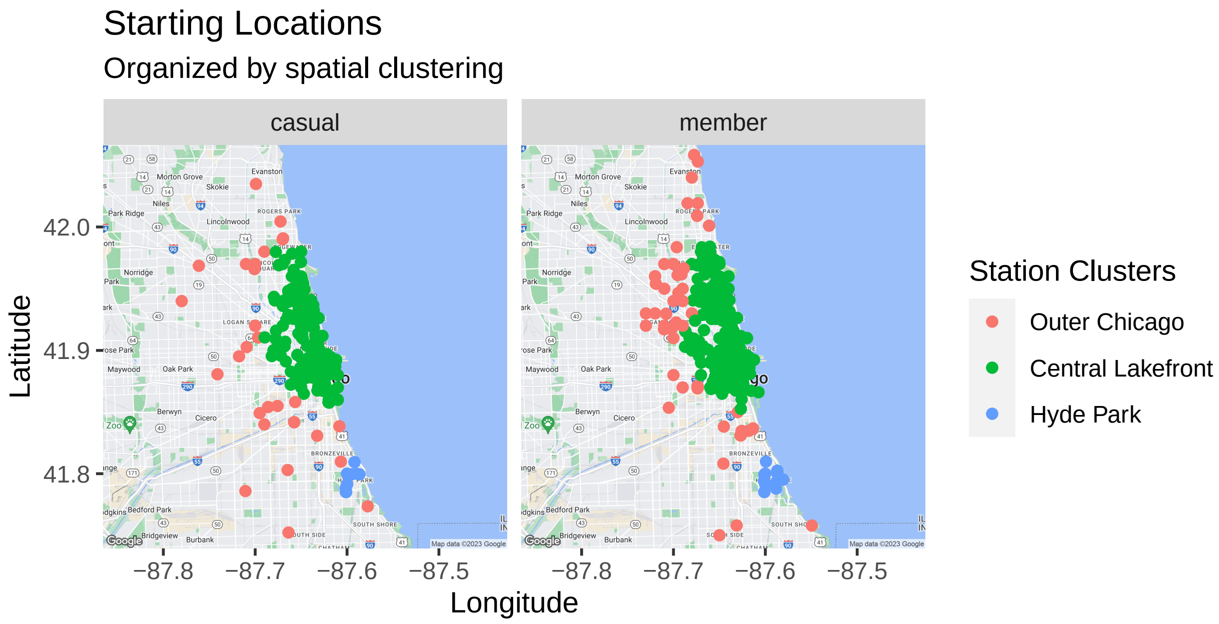
```
## building a function to calculate Haversine distance
haversine_distance <- function(lat1, lon1, lat2, lon2){
  # converting degrees to radians
  lat1 <- as.numeric(lat1) * pi / 180
  lon1 <- as.numeric(lon1) * pi / 180
  lat2 <- as.numeric(lat2) * pi / 180
  lon2 <- as.numeric(lon2) * pi / 180

  # Haversine formula
  a <- sin((lat2 - lat1)/2)^2 + cos(lat1) * cos(lat2) *   sin((lon2 - lon1)/2)^2
  distance <- 2 * 3958.756 * asin(sqrt(a)) # Earth's radius in miles

  return(distance)
}

# applying the function
df_clean$computed_distance <-
  mapply(haversine_distance,
         df_clean$start_lat,
         df_clean$start_lng,
```

```r
         df_clean$end_lat,
         df_clean$end_lng)

# average distance by customer type
average_trip_distance <- df_clean %>%
  filter(!is.na(computed_distance)) %>%
  group_by(member_casual) %>%
  summarise(avg_distance = mean(computed_distance))


## finding popular stations
popular_starts <- df_clean %>%
  filter(!is.na(start_station_name)) %>%
  group_by(start_station_name) %>%
  summarise(
    count = n(),
    start_lat = mean(start_lat),
    start_lng = mean(start_lng)
  ) %>%
  arrange(-count) %>%
  top_n(10)

popular_ends <- df_clean %>%
  filter(!is.na(end_station_name)) %>%
  group_by(end_station_name) %>%
  summarise(count = n()) %>%
  arrange(-count) %>%
  top_n(10)


## finding popular routes
popular_routes <- df_clean %>%
  filter(
    !is.na(start_station_name) & !is.na(end_station_name)
    ) %>%
  group_by(start_station_name,
  end_station_name) %>%
  summarise(count = n()) %>%
  arrange(-count) %>%
  top_n(10)


## mapping start stations
# finding the center of our map
total_mean_lat <- mean(df_clean$start_lat)
total_mean_lng <- mean(df_clean$start_lng)

chicago_map <- get_map(
  location = c(
    lon = total_mean_lng,
    lat = total_mean_lat), zoom = 11)

# plotting our map
```

```
ggmap(chicago_map) +
  geom_point(
    data = df_clean,
    aes(
      x = start_lng,
      y = start_lat,
      color = rideable_type)) +
  labs(title = "Starting Locations",
       subtitle = "Organized by Ride Type",
       x = "Longitude",
       y = "Latitude") +
  scale_color_discrete(name = "Ride Type",
                       labels = c("Classic",
                                  "Docked",
                                  "Electric"))
```

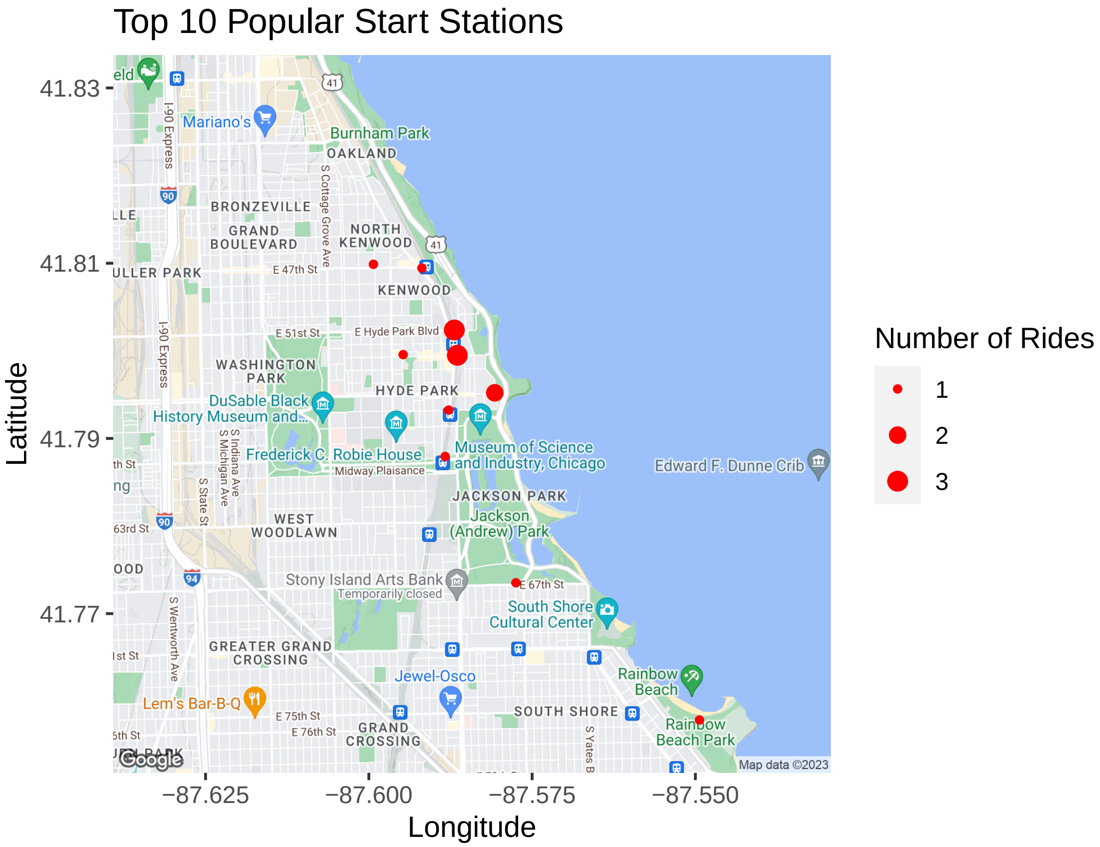


```
## density-based spatial clustering
coordinates <- df_clean[, c("start_lng", "start_lat")]

clusters <- dbscan(coordinates, eps = 0.01, minPts = 10)

df_clean$cluster <- clusters$cluster

df_clean$member_casual <- as.factor(df_clean$member_casual)

cluster_labels <- c("Outer Chicago",
                    "Central Lakefront",
                    "Hyde Park")

# plotting clusters by color and customer
ggmap(chicago_map) +
  geom_point(
    data = df_clean,
    aes(
      x = start_lng,
      y = start_lat,
      color = as.factor(cluster))) +
  labs(title = "Starting Locations",
       subtitle = "Organized by spatial clustering",
       x = "Longitude",
       y = "Latitude") +
  scale_color_discrete(name = "Station Clusters",
                       labels = cluster_labels) +
  facet_wrap(~ member_casual)
```



```
## mapping popular start stations
# finding the center of our map
mean_lat <- mean(popular_starts$start_lat)
mean_lng <- mean(popular_starts$start_lng)

s_chicago_map <- get_map(location = c(lon = mean_lng, lat = mean_lat), zoom = 13)
```

```
# plotting our map
ggmap(s_chicago_map) +
  geom_point(
    data = popular_starts,
    aes(
      x = start_lng,
      y = start_lat,
      size = count
    ),
    color = "red") +
  scale_size(
    name = "Number of Rides",
    range = c(1, 3),
    breaks = 1:3
  ) +
  labs(title = "Top 10 Popular Start Stations",
       x = "Longitude",
       y = "Latitude")
```



Final analysis, marketing suggestions, and future research

Through our analysis and visualization of Cyclistic's bike share data, we have shown that annual members have a slight preference for electric bikes but also ride casual bikes at nearly the same frequency. They absolutely avoid docked bikes. They like the freedom of being able to pick up a bike from wherever they are and drop it off wherever they need to go, and the data suggests that either electric or classic bikes will do the job. by designing ad campaigns that emphasize the plurality of choices offered to those in need, Cyclistic can appeal to those casual riders who may prefer similar benefits.

Annual members ride much more often on weekdays than do casual riders and they both ride about the same amount on weekends. The time of day that annual members choose to start their rides are also indicative of typical commuting hours which, along with their disinclination towards either type of bike offered by the company, suggests that they're using the bikes for commuting purposes.

Members typically ride the most leading up to and peaking in June. presumably, as temperatures rise in July Cyclistic members ride slightly less frequently, though things pick back up between the middle of August and the end of September. Running advertising campaigns during these peak times may capture the attention of casual riders seeking to bike more often.

Both annual members and casual riders alike typically begin their journeys along the Central Lakefront, which is a region in Chicago that contains some of the most populated and popular destinations. Moreover, The top 10 most popular starting locations are all grouped in and around Hyde Park. These two locations represent prime real estate for advertising campaigns focused on converting casual riders into annual members.

Based on the information that we've gathered through our analysis, here are five simple recommendations to the Cyclistic marketing team:

- Focus on promotional content that emphasizes the appeal of electric and classic bikes
- Develop a marketing campaign that demonstrates the cost savings of an annual membership for those who ride frequently by comparing the cost of taxi rides, gas station visits, etc.
- Highlight the convenience of being able to freely hop on and off a bike whenever you're in a rush to work or on your way to a meeting
- Launch seasonal advertising that targets casual riders during June and between August and October
- Increase the availability of Cyclistic's bike fleet along Central Lakefront and around Hyde Park
- Reach out to local businesses in Hyde Park with cross-promotion offers

Taking these suggestions seriously will increase the conversion rate between casual riders and annual members and open up a doorway towards increasing sales and inflating membership subscriptions. To further analyze how the marketing team can drive these conversions, Cyclistic could look towards gathering more types of data about their customers. A simple way to do this would be through customer surveys, which would enable Cyclistic to understand rider motivations and offer specialized feedback, or capturing demographic information, the usage of secondary services, payment methods, website interaction, weather conditions, and more. A treasure trove of data analysis could take place if we aggregated all of this information together.

Citations

Dowle M, Srinivasan A (2023). *data.table: Extension of `data.frame`.* R package version 1.14.8, https://CRAN.R-project.org/package=data.table.

Fischetti T (2022). *assertr: Assertive Programming for R Analysis Pipelines.* R package version 3.0.0, https://CRAN.R-project.org/package=assertr.

Garrett Grolemund, Hadley Wickham (2011). Dates and Times Made Easy with lubridate. Journal of Statistical Software, 40(3), 1-25. URL https://www.jstatsoft.org/v40/i03/.

D. Kahle and H. Wickham. ggmap: Spatial Visualization with ggplot2. The R Journal, 5(1), 144-161. URL http://journal.r-project.org/archive/2013-1/kahle-wickham.pdf

Mark P. J. van der Loo, Edwin de Jonge (2021). Data Validation Infrastructure for R. Journal of Statistical Software, 97(10), 1-31. doi:10.18637/jss.v097.i10

Wickham H, Averick M, Bryan J, Chang W, McGowan LD, François R, Grolemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen TL, Miller E, Bache SM, Müller K, Ooms J, Robinson D, Seidel DP, Spinu V, Takahashi K, Vaughan D, Wilke C, Woo K, Yutani H (2019). "Welcome to the tidyverse." *Journal of Open Source Software*, *4*(43), 1686. doi:10.21105/joss.01686 https://doi.org/10.21105/joss.01686.

Xie Y (2023). *knitr: A General-Purpose Package for Dynamic Report Generation in R*. R package version 1.43, https://yihui.org/knitr/.

Yihui Xie (2015) Dynamic Documents with R and knitr. 2nd edition. Chapman and Hall/CRC. ISBN 978-1498716963

Yihui Xie (2014) knitr: A Comprehensive Tool for Reproducible Research in R. In Victoria Stodden, Friedrich Leisch and Roger D. Peng, editors, Implementing Reproducible Computational Research. Chapman and Hall/CRC. ISBN 978-1466561595