Case_Study1_cleaned

2025-01-12

###Introduction Welcome to the Cyclistic bike-share analysis case study! In this case study, you work for a fictional company, Cyclistic, along with some key team members. In order to answer the business questions, follow the steps of the data analysis process: Ask, Prepare, Process, Analyze, Share, and Act. Along the way, the Case Study Roadmap tables — including guiding questions and key tasks — will help you stay on the right path. ###Scenario You are a junior data analyst working on the marketing analyst team at Cyclistic, a bike-share company in Chicago. The director of marketing believes the company's future success depends on maximizing the number of annual memberships. Therefore, your team wants to understand how casual riders and annual members use Cyclistic bikes differently. From these insights, your team will design a new marketing strategy to convert casual riders into annual members. But first, Cyclistic executives must approve your recommendations, so they must be backed up with compelling data insights and professional data visualizations.

Analyze

1. Combine Columns and Data

```
# Set working directory
setwd("/Users/giuliaribeiro/Documents/R_course/Case_Study1/")

# Import monthly datasets
library(dplyr)
```

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

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

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

```
library(readr)
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 4.3.3

##
## Attaching package: 'lubridate'

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

```
data_dir <- "./monthly_files/"
file_list <- list.files(path = data_dir, pattern = "*.csv", full.names = TRUE)
```

```r
# Combine datasets
data_combined <- file_list %>%
  lapply(read_csv) %>%
  bind_rows() %>%
  mutate(month = month(started_at),
         month_name = month(started_at, label = TRUE, abbr = FALSE))
```

```
## Rows: 144873 Columns: 13
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 223164 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 301687 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 415025 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 609493 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 710721 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 748962 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 755639 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 821276 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 616281 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 335075 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 178372 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# Preview combined data
glimpse(data_combined)
```

```
## Rows: 5,860,568
## Columns: 15
## $ ride_id            <chr> "C1D650626C8C899A", "EECD38BDB25BFCB0", "F4A9CE7806~
## $ rideable_type      <chr> "electric_bike", "electric_bike", "electric_bike", ~
## $ started_at         <dttm> 2024-01-12 15:30:27, 2024-01-08 15:45:46, 2024-01-~
## $ ended_at           <dttm> 2024-01-12 15:37:59, 2024-01-08 15:52:59, 2024-01-~
## $ start_station_name <chr> "Wells St & Elm St", "Wells St & Elm St", "Wells St~
## $ start_station_id   <chr> "KA1504000135", "KA1504000135", "KA1504000135", "TA~
## $ end_station_name   <chr> "Kingsbury St & Kinzie St", "Kingsbury St & Kinzie ~
## $ end_station_id     <chr> "KA1503000043", "KA1503000043", "KA1503000043", "13~
## $ start_lat          <dbl> 41.90327, 41.90294, 41.90295, 41.88430, 41.94880, 4~
## $ start_lng          <dbl> -87.63474, -87.63444, -87.63447, -87.63396, -87.675~
## $ end_lat            <dbl> 41.88918, 41.88918, 41.88918, 41.92182, 41.88918, 4~
## $ end_lng            <dbl> -87.63851, -87.63851, -87.63851, -87.64414, -87.638~
## $ member_casual      <chr> "member", "member", "member", "member", "member", "~
## $ month              <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ month_name         <ord> January, January, January, January, January, Januar~
```

2. Clean and Transform Data

I calculate the percentage of missing data on the columns to understand the strategy to be adopted. I saw that for station names 18% was missing data, so in that case to not miss a lot of data I just replaced NA for Unknown. In the case of end_lat and end_long it was only 0.1%, so In this case I dropped the rows with NA.

```r
# Calculate the total number of rows in the dataset
total_rows <- nrow(data_combined)

# Calculate the number of missing values for each column
missing_values <- colSums(is.na(data_combined))

# Calculate the percentage of missing values for each column
missing_percentage <- (missing_values / total_rows) * 100

# Combine the results into a data frame for better readability
missing_summary <- data.frame(
  Column = names(missing_values),
  Missing_Count = missing_values,
  Missing_Percentage = round(missing_percentage, 1) # Rounded to one decimal place
)

# Print the summary
print(missing_summary)
```

```
##                                 Column Missing_Count Missing_Percentage
## ride_id                       ride_id             0                0.0
## rideable_type           rideable_type             0                0.0
## started_at                 started_at             0                0.0
## ended_at                     ended_at             0                0.0
## start_station_name start_station_name       1073951               18.3
## start_station_id     start_station_id       1073951               18.3
## end_station_name     end_station_name       1104653               18.8
## end_station_id         end_station_id       1104653               18.8
## start_lat                   start_lat             0                0.0
## start_lng                   start_lng             0                0.0
## end_lat                       end_lat          7232                0.1
## end_lng                       end_lng          7232                0.1
## member_casual           member_casual             0                0.0
## month                           month             0                0.0
## month_name                 month_name             0                0.0
```

```
# Handle missing values
library(tidyr)
data_cleaned <- data_combined %>%
  mutate(
    start_station_name = replace_na(start_station_name, "Unknown"),
    start_station_id = replace_na(start_station_id, "Unknown"),
    end_station_name = replace_na(end_station_name, "Unknown"),
    end_station_id = replace_na(end_station_id, "Unknown"),
    ride_length = as.numeric(difftime(ended_at, started_at, units = "mins")),
    day_of_week = wday(started_at, label = TRUE)
  ) %>%
  drop_na(end_lat, end_lng) # Drop rows with critical missing values

# Preview cleaned data
glimpse(data_cleaned)
```

```
## Rows: 5,853,336
## Columns: 17
## $ ride_id            <chr> "C1D650626C8C899A", "EECD38BDB25BFCB0", "F4A9CE7806~
## $ rideable_type      <chr> "electric_bike", "electric_bike", "electric_bike", ~
## $ started_at         <dttm> 2024-01-12 15:30:27, 2024-01-08 15:45:46, 2024-01-~
## $ ended_at           <dttm> 2024-01-12 15:37:59, 2024-01-08 15:52:59, 2024-01-~
## $ start_station_name <chr> "Wells St & Elm St", "Wells St & Elm St", "Wells St~
## $ start_station_id   <chr> "KA1504000135", "KA1504000135", "KA1504000135", "TA~
## $ end_station_name   <chr> "Kingsbury St & Kinzie St", "Kingsbury St & Kinzie ~
## $ end_station_id     <chr> "KA1503000043", "KA1503000043", "KA1503000043", "13~
## $ start_lat          <dbl> 41.90327, 41.90294, 41.90295, 41.88430, 41.94880, 4~
## $ start_lng          <dbl> -87.63474, -87.63444, -87.63447, -87.63396, -87.675~
## $ end_lat            <dbl> 41.88918, 41.88918, 41.88918, 41.92182, 41.88918, 4~
## $ end_lng            <dbl> -87.63851, -87.63851, -87.63851, -87.64414, -87.638~
## $ member_casual      <chr> "member", "member", "member", "member", "member", "~
## $ month              <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ month_name         <ord> January, January, January, January, January, Januar~
## $ ride_length        <dbl> 7.533333, 7.216667, 8.000000, 29.816667, 26.200000,~
## $ day_of_week        <ord> Fri, Mon, Sat, Mon, Wed, Sun, Fri, Thu, Mon, Wed, W~
```

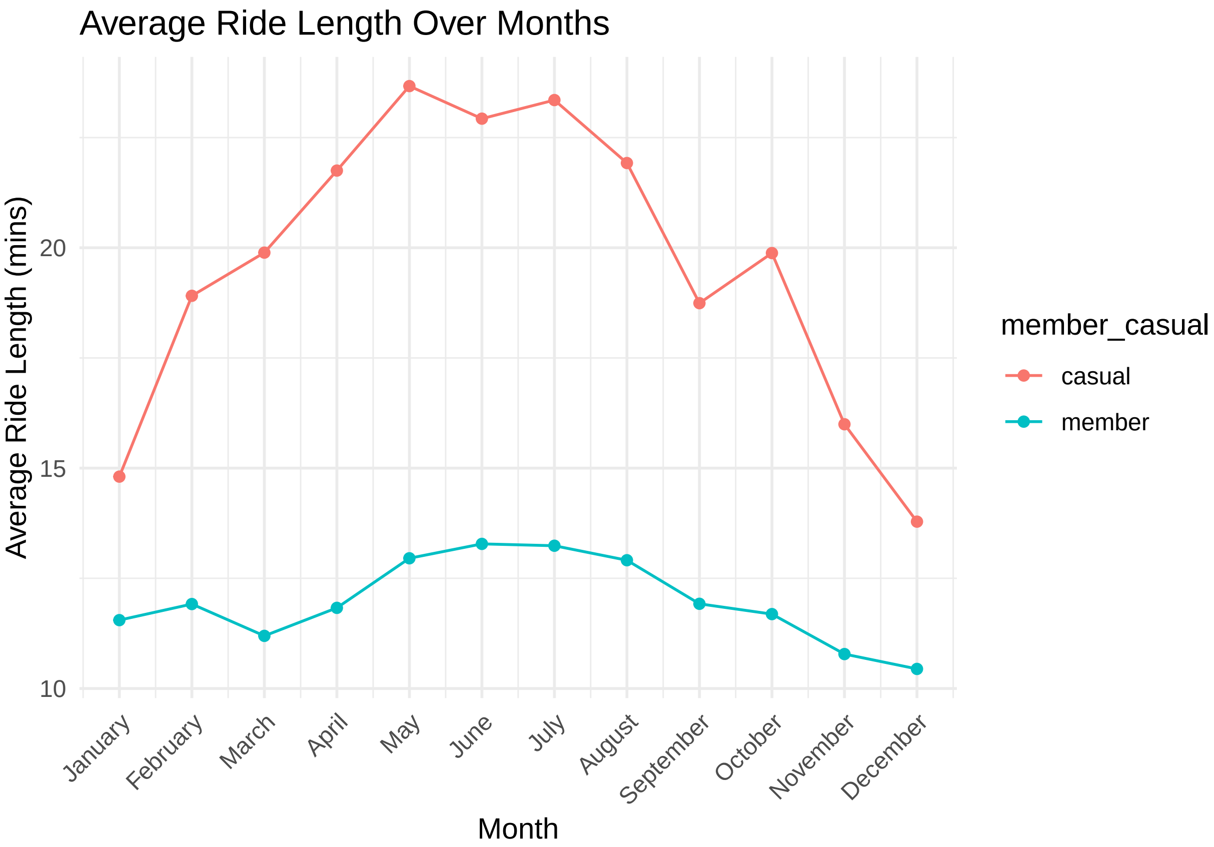
I added two new columns that I think will give usefull insights in this analysis: ride_length and day_of_week

3. Descriptive Analysis

```
# Analyze ride length (e.g., by month)
monthly_summary <- data_cleaned %>%
  group_by(month, member_casual) %>%
  summarize(
    avg_ride_length = mean(ride_length, na.rm = TRUE),
    total_rides = n(),
    .groups = "drop"
  )

# Print summary
print(monthly_summary)
```

```
## # A tibble: 24 x 4
##    month member_casual avg_ride_length total_rides
##    <dbl> <chr>                   <dbl>       <int>
##  1     1 casual                   14.8       24353
##  2     1 member                   11.6      120232
##  3     2 casual                   18.9       46963
##  4     2 member                   11.9      175883
##  5     3 casual                   19.9       82268
##  6     3 member                   11.2      219023
##  7     4 casual                   21.8      131431
##  8     4 member                   11.8      283115
##  9     5 casual                   23.7      230466
## 10     5 member                   13.0      378414
## # i 14 more rows
```

```
# Visualize ride length over the months
library(ggplot2)

ggplot(monthly_summary, aes(x = month, y = avg_ride_length, color = member_casual)) +
  geom_line() +
  geom_point() +
  labs(title = "Average Ride Length Over Months", x = "Month", y = "Average Ride Length (mins)") +
  scale_x_continuous(breaks = 1:12, labels = month.name) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



```
# Calculate mean and max ride length
mean_ride_length <- mean(data_cleaned$ride_length, na.rm = TRUE)
max_ride_length <- max(data_cleaned$ride_length, na.rm = TRUE)

# Convert day_of_week to character (remove ordering)
data_cleaned <- data_cleaned %>%
  mutate(day_of_week = as.character(day_of_week))

# Calculate mode of day_of_week
day_mode <- data_cleaned %>%
  group_by(day_of_week) %>%
  summarize(count = n(), .groups = "drop") %>%  # Ensure proper grouping behavior
  arrange(desc(count)) %>%
  slice_head(n = 1) %>%  # Select the first row explicitly
  pull(day_of_week)      # Extract the mode value

print(day_mode)
```

```
## [1] "Sat"
```

```
list(mean_ride_length = mean_ride_length, max_ride_length = max_ride_length, day_mode = day_mode)
```

```
## $mean_ride_length
## [1] 15.4839
##
## $max_ride_length
## [1] 1509.367
##
```

```
## $day_mode
## [1] "Sat"
```

4. Pivot Table Analysis

```
# Average ride length by member type
data_summary <- data_cleaned %>%
  group_by(member_casual) %>%
  summarize(avg_ride_length = mean(ride_length, na.rm = TRUE))

# Average ride length by day of week
data_by_day <- data_cleaned %>%
  group_by(day_of_week, member_casual) %>%
  summarize(avg_ride_length = mean(ride_length, na.rm = TRUE))
```

```
## `summarise()` has grouped output by 'day_of_week'. You can override using the
## `.groups` argument.
```

```
# Count rides by day of week
ride_count <- data_cleaned %>%
  group_by(day_of_week, member_casual) %>%
  summarize(total_rides = n())
```

```
## `summarise()` has grouped output by 'day_of_week'. You can override using the
## `.groups` argument.
```

```
list(summary = data_summary, by_day = data_by_day, ride_count = ride_count)
```

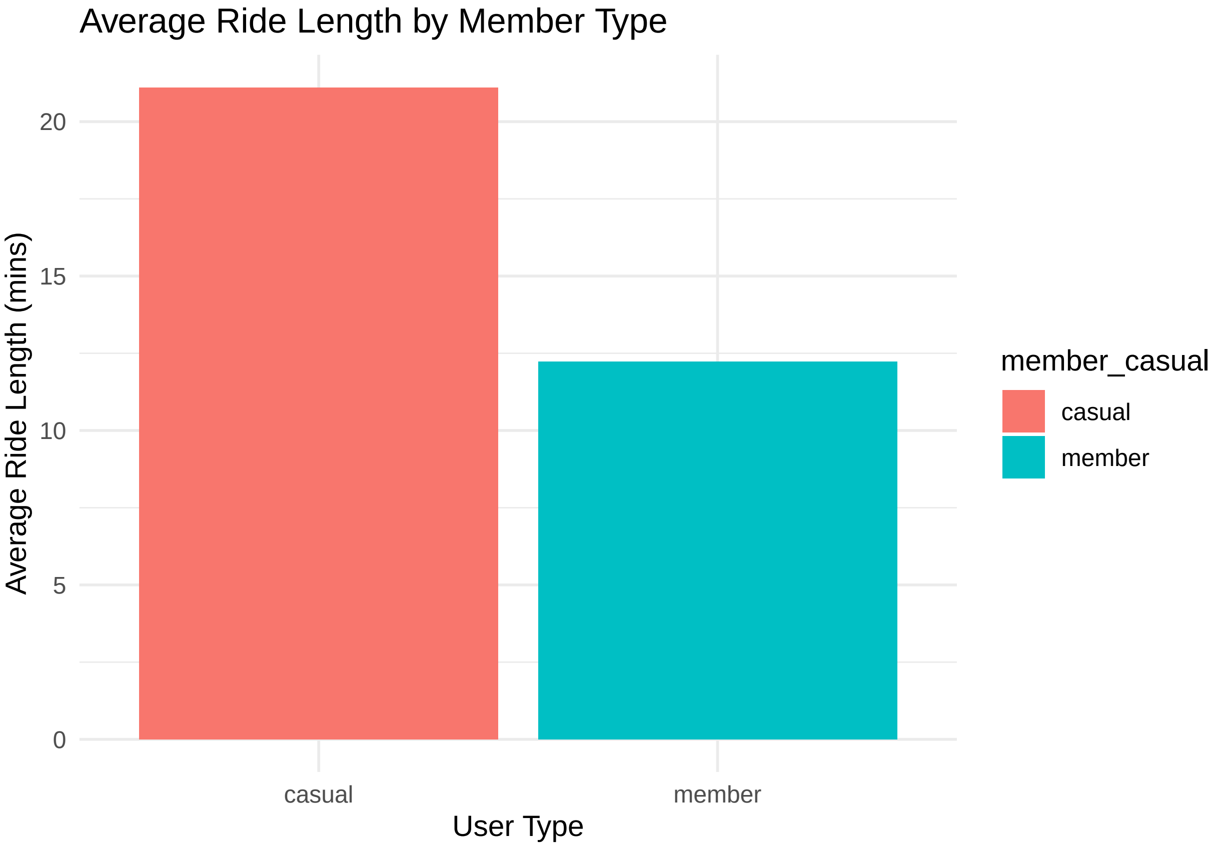
```
## $summary
## # A tibble: 2 x 2
##   member_casual avg_ride_length
##   <chr>                   <dbl>
## 1 casual                   21.1
## 2 member                   12.2
##
## $by_day
## # A tibble: 14 x 3
## # Groups:   day_of_week [7]
##    day_of_week member_casual avg_ride_length
##    <chr>       <chr>                   <dbl>
##  1 Fri         casual                   20.4
##  2 Fri         member                   11.9
##  3 Mon         casual                   20.4
##  4 Mon         member                   11.7
##  5 Sat         casual                   23.9
##  6 Sat         member                   13.6
##  7 Sun         casual                   24.4
##  8 Sun         member                   13.6
##  9 Thu         casual                   18.4
## 10 Thu         member                   11.7
## 11 Tue         casual                   18.2
## 12 Tue         member                   11.7
## 13 Wed         casual                   18.6
## 14 Wed         member                   11.9
##
## $ride_count
## # A tibble: 14 x 3
## # Groups:   day_of_week [7]
##    day_of_week member_casual total_rides
##    <chr>       <chr>               <int>
##  1 Fri         casual             314987
##  2 Fri         member             525647
##  3 Mon         casual             252963
##  4 Mon         member             534405
##  5 Sat         casual             444123
##  6 Sat         member             479520
##  7 Sun         casual             368682
##  8 Sun         member             417067
##  9 Thu         casual             264465
## 10 Thu         member             570393
## 11 Tue         casual             231865
## 12 Tue         member             570467
## 13 Wed         casual             268692
## 14 Wed         member             610060
```

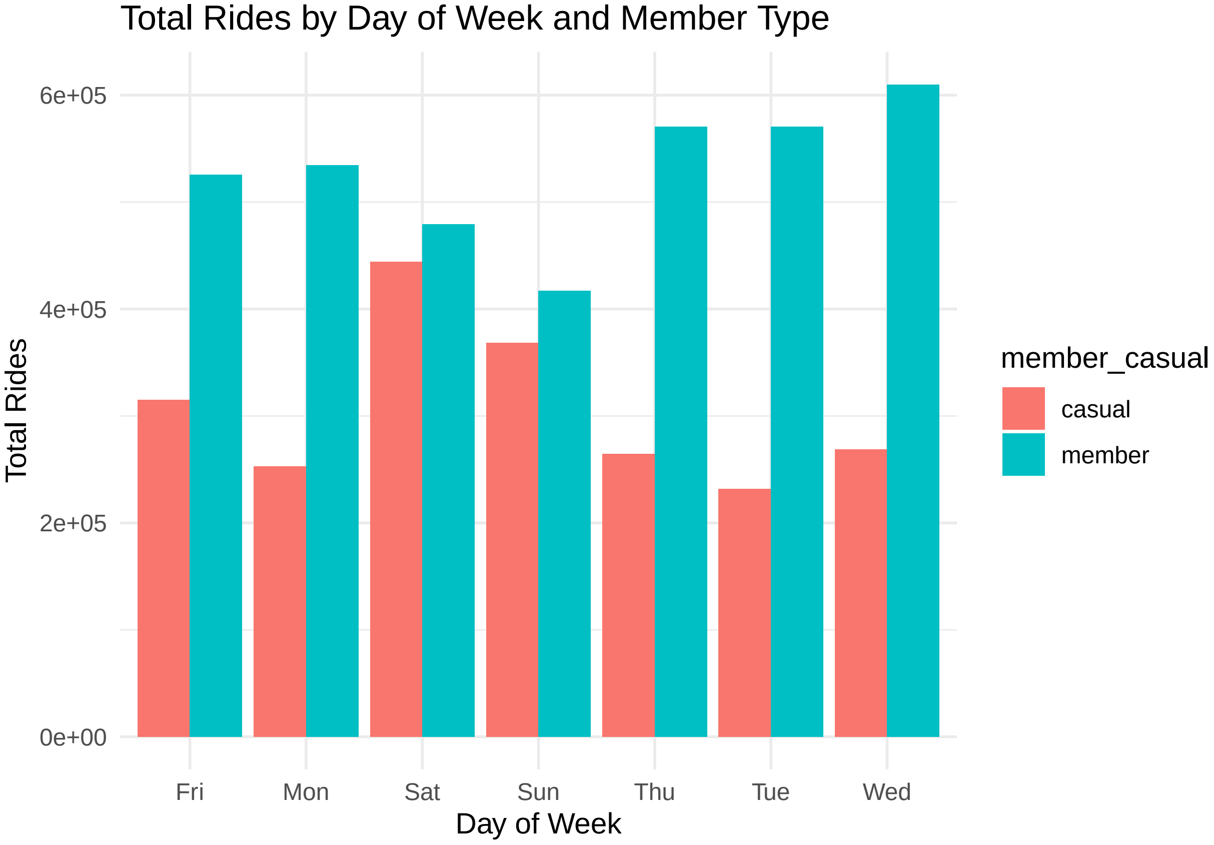
5. Seasonal Analysis

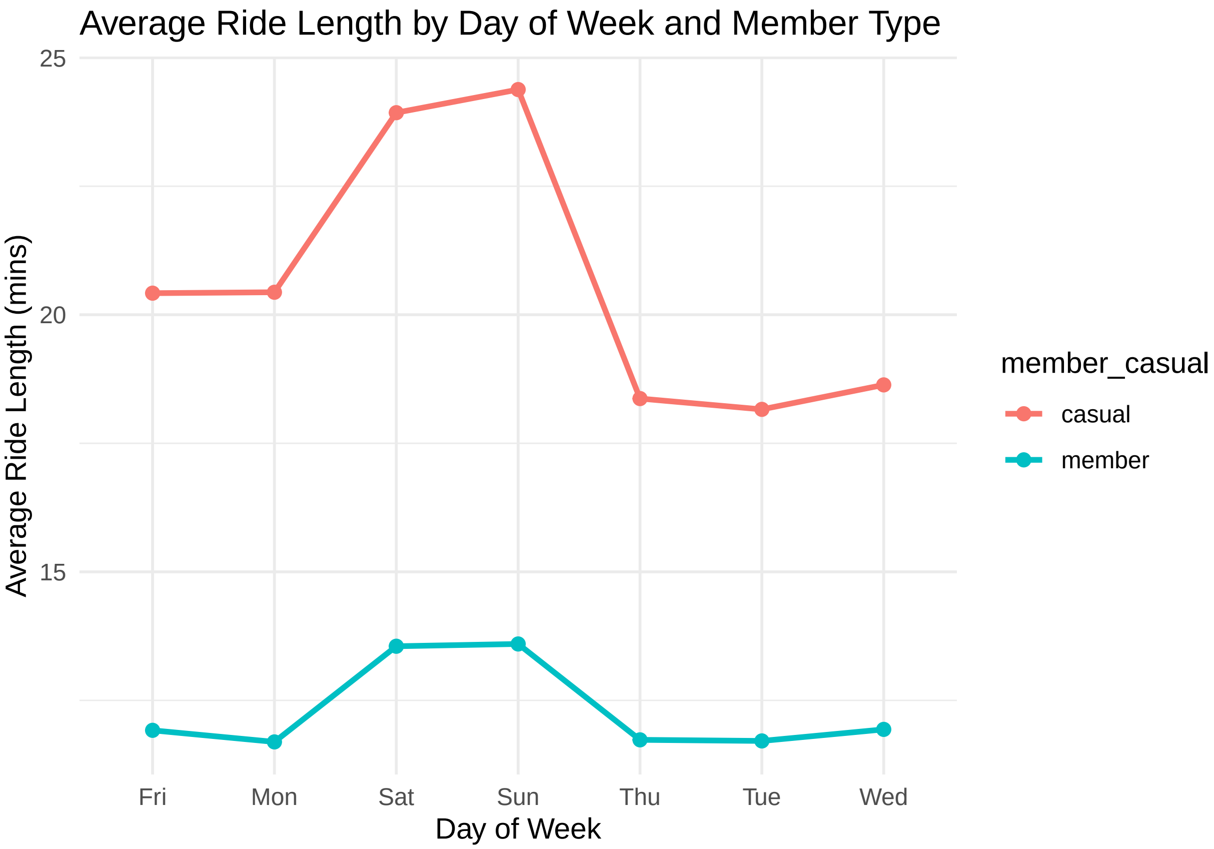
```
# Filter for summer months (June, July, August)
summer_data <- data_cleaned %>%
  filter(month(started_at) %in% c(6, 7, 8))

# Descriptive stats for summer
summer_stats <- summer_data %>%
  group_by(member_casual) %>%
  summarize(
    avg_ride_length = mean(ride_length, na.rm = TRUE),
    total_rides = n()
  )

summer_stats
```

```
## # A tibble: 2 x 3
##   member_casual avg_ride_length total_rides
##   <chr>                   <dbl>       <int>
## 1 casual                   22.7      937341
## 2 member                   13.1     1274958
```

6. Visualization


Average Ride Length by Member Type


Total Rides by Day of Week and Member Type



7. Export Summary File

```
# Export cleaned and summarized data
write_csv(data_cleaned, "cleaned_cyclistic_data.csv")
write_csv(data_summary, "ride_length_summary.csv")
write_csv(data_by_day, "ride_length_by_day.csv")
```

8. Summary Narrative

Insights from Analysis

1. **Key Trends:**
   - Casual riders consistently have longer average ride lengths compared to members, particularly during weekends and summer months.
   - Members primarily use bikes during weekdays, suggesting frequent utility-focused usage such as commuting.
2. **Seasonal Variations:**
   - Summer months show a significant spike in ridership, especially among casual users, indicating a strong link to leisure activities.
3. **Most Popular Days:**
   - Saturdays dominate as the most popular day for riding, particularly for casual users.

Implications for Marketing Strategy

- **Targeting Casual Riders:** Highlight benefits of membership during summer and weekends to encourage conversions. For example, offer discounts for summer memberships.
- **Member Retention:** Reinforce the utility and cost-effectiveness of membership for regular commuters.

- **Seasonal Campaigns:** Plan promotions during peak summer months to attract more casual users.