Cyclistic_Bike_Share

JF

2024-08-29

Set up the data

```r
# Set the path to the directory containing the files
path <- "/Users/jellyfish/Documents/Untitled/My-repo/bike"

# List all files containing "tripdata" in their name
files <- list.files(path = path, pattern = "tripdata", full.names = TRUE)

# Read all files and combine them into one data frame
all_data <- lapply(files, read_csv) %>%
  bind_rows()
```

```
## 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.
```

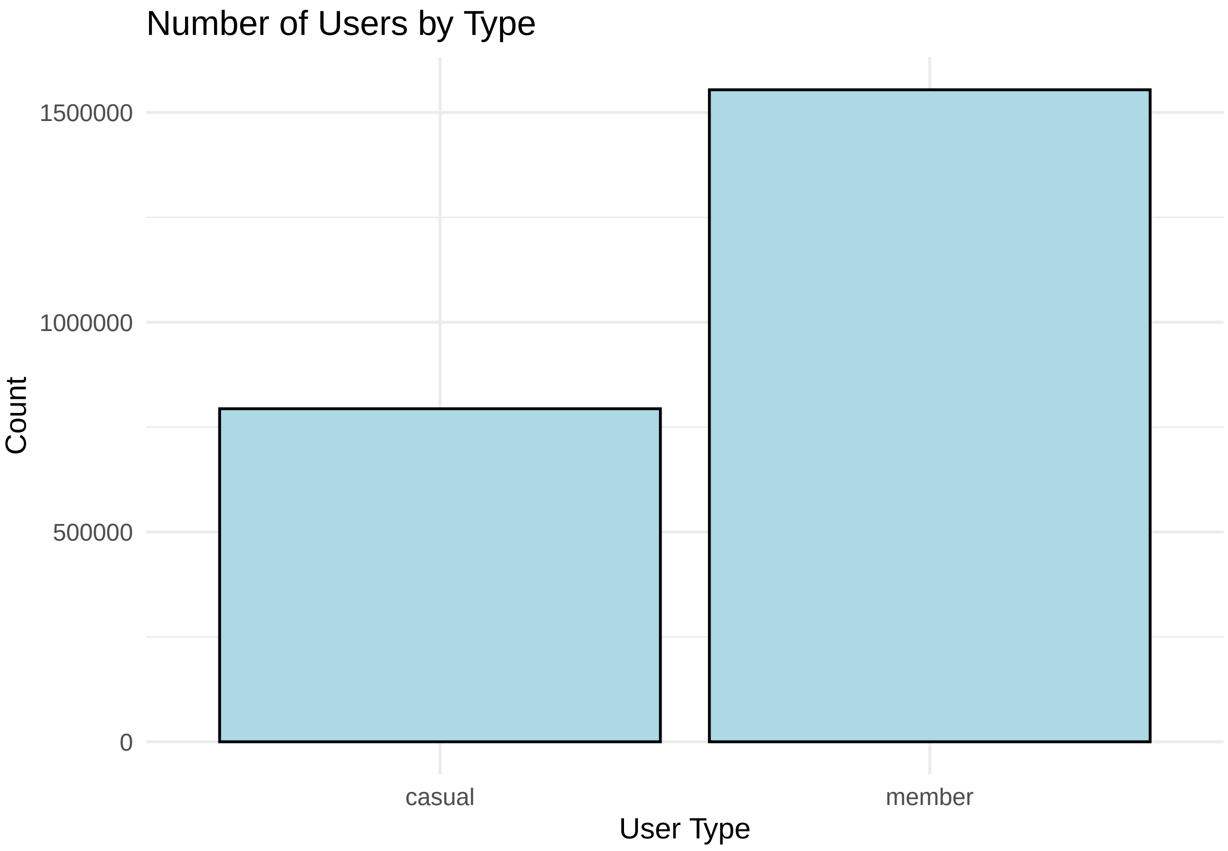
```r
# Preview the combined data
head(all_data)
```

```
## # A tibble: 6 x 13
##   ride_id          rideable_type started_at          ended_at
##   <chr>            <chr>         <dttm>              <dttm>
## 1 C1D650626C8C899A electric_bike 2024-01-12 15:30:27 2024-01-12 15:37:59
## 2 EECD38BDB25BFCB0 electric_bike 2024-01-08 15:45:46 2024-01-08 15:52:59
## 3 F4A9CE78061F17F7 electric_bike 2024-01-27 12:27:19 2024-01-27 12:35:19
## 4 0A0D9E15EE50B171 classic_bike  2024-01-29 16:26:17 2024-01-29 16:56:06
## 5 33FFC9805E3EFF9A classic_bike  2024-01-31 05:43:23 2024-01-31 06:09:35
## 6 C96080812CD285C5 classic_bike  2024-01-07 11:21:24 2024-01-07 11:30:03
## # i 9 more variables: start_station_name <chr>, start_station_id <chr>,
## #   end_station_name <chr>, end_station_id <chr>, start_lat <dbl>,
## #   start_lng <dbl>, end_lat <dbl>, end_lng <dbl>, member_casual <chr>
```

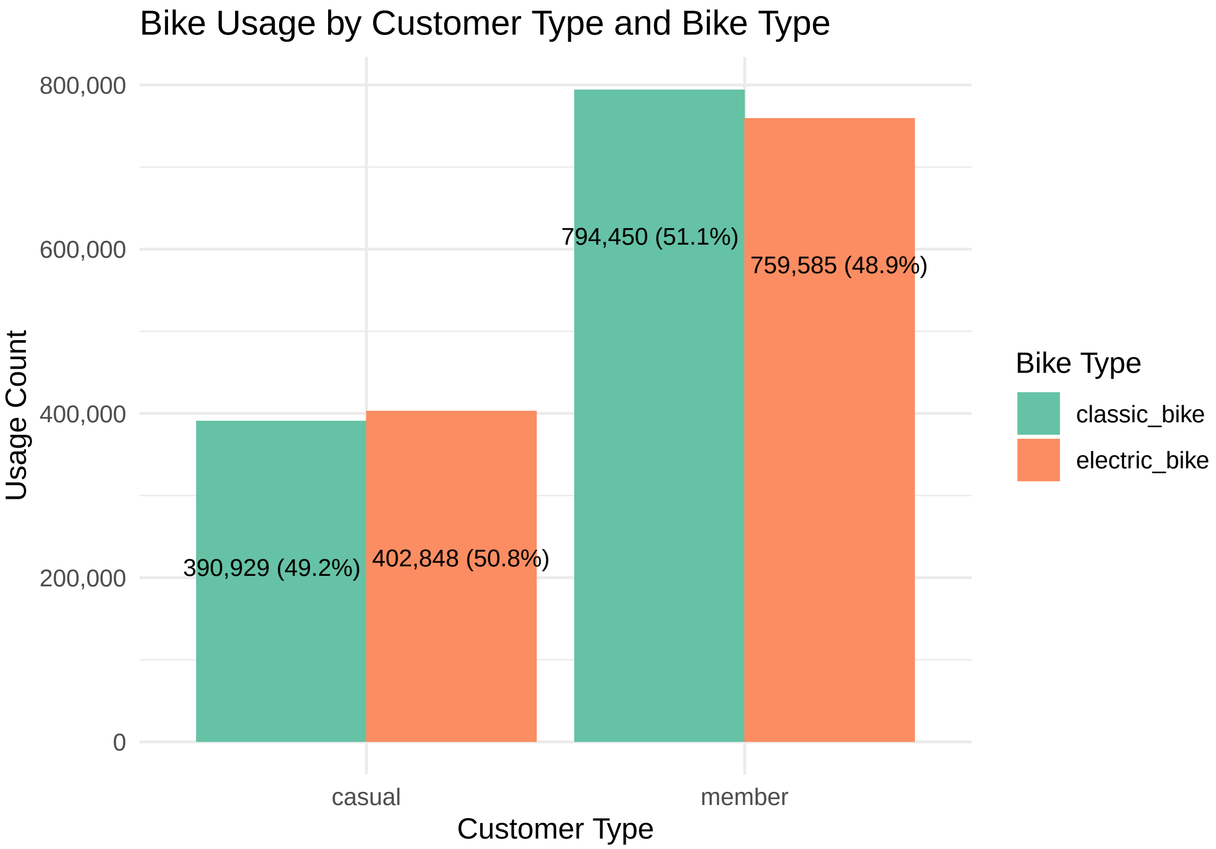
```r
# Calculate the duration
all_data$duration <- all_data$ended_at - all_data$started_at

# Convert duration to different units (e.g., minutes)
all_data$duration <- as.numeric(all_data$duration, units = "mins")
all_data$duration <- round(all_data$duration, 1)

# Add new columns for month, year, and weekday
all_data$month <- format(all_data$started_at, "%m")
all_data$year <- format(all_data$started_at, "%Y")
all_data$weekday <- weekdays(all_data$started_at)

# Drop rows where duration is less than 1 mins
all_data <- all_data[all_data$duration >= 1, ]
```

Create a bar plot showing the number of users by type

```r
# Create a bar plot showing the number of users by type
ggplot(all_data, aes(x = member_casual)) +
    geom_bar(fill = "lightblue", color = "black") +
    labs(title = "Number of Users by Type",
        x = "User Type",
        y = "Count") +
    theme_minimal()
```

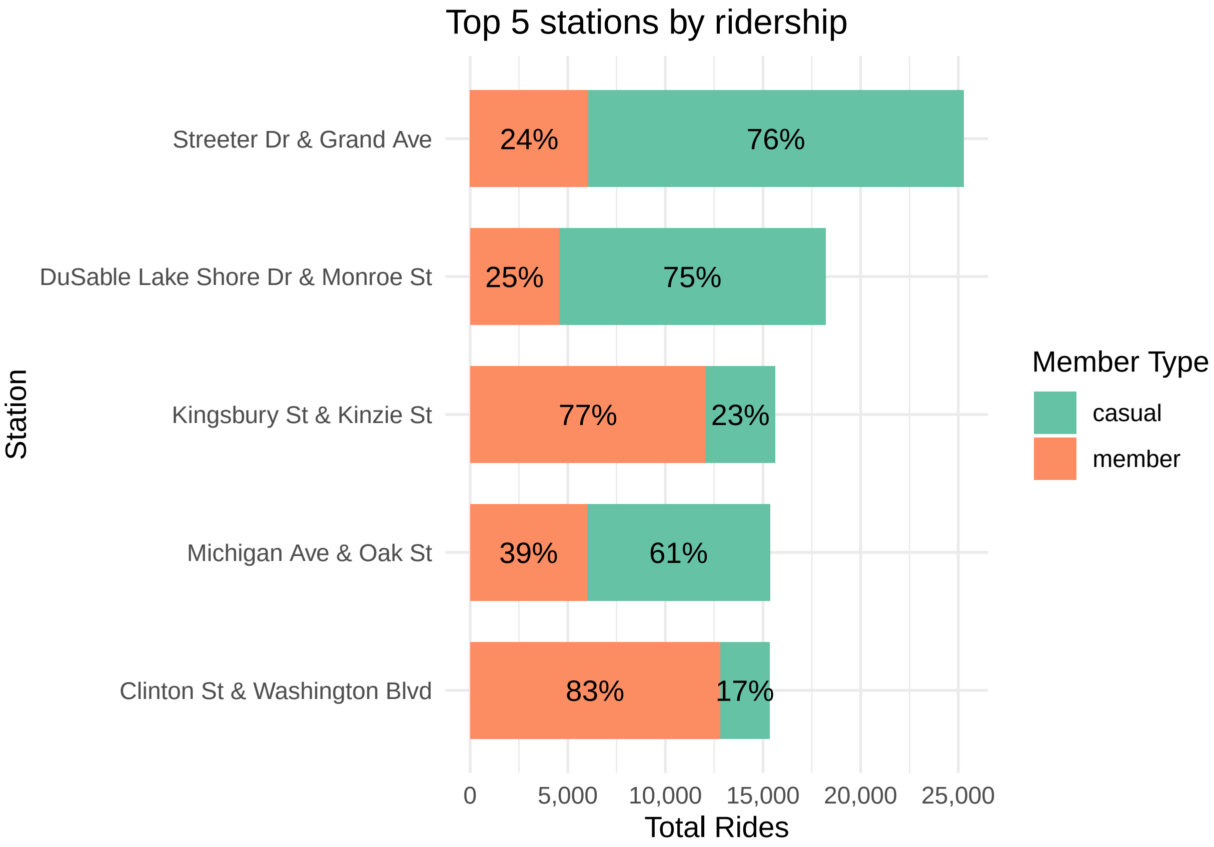


Bar chart of bike usage by customer type and bike type



Top five stations by rideship

```r
# Determine Top 5 stations by ridership
top_stations <- all_data %>%
    filter(!is.na(start_station_name)) %>%
    group_by(start_station_name, member_casual) %>%
    summarise(number_rides = n(), .groups="drop_last") %>%
    group_by(start_station_name) %>%
    mutate(total_rides = sum(number_rides)) %>%
    slice_max(order_by = total_rides, n = 1) %>%
    ungroup() %>%
    arrange(desc(total_rides)) %>%
    mutate(percent = number_rides/total_rides*100) %>%
    top_n(10, total_rides)

# Plot Top 5 stations by ridership
ggplot(top_stations, aes(x = reorder(start_station_name, total_rides), y = number_rides, fill = member_
    geom_bar(stat = "identity", width = 0.7) +
    geom_text(aes(label=paste0(round(percent), "%")),
            position = position_stack(vjust = 0.5)) +
    coord_flip() +
    labs(title = "Top 5 stations by ridership",
        x = "Station",
        y = "Total Rides",
        fill = "Member Type") +
    theme_minimal() +
    scale_y_continuous(labels = scales::comma) +
    scale_fill_brewer(palette = "Set2")
```



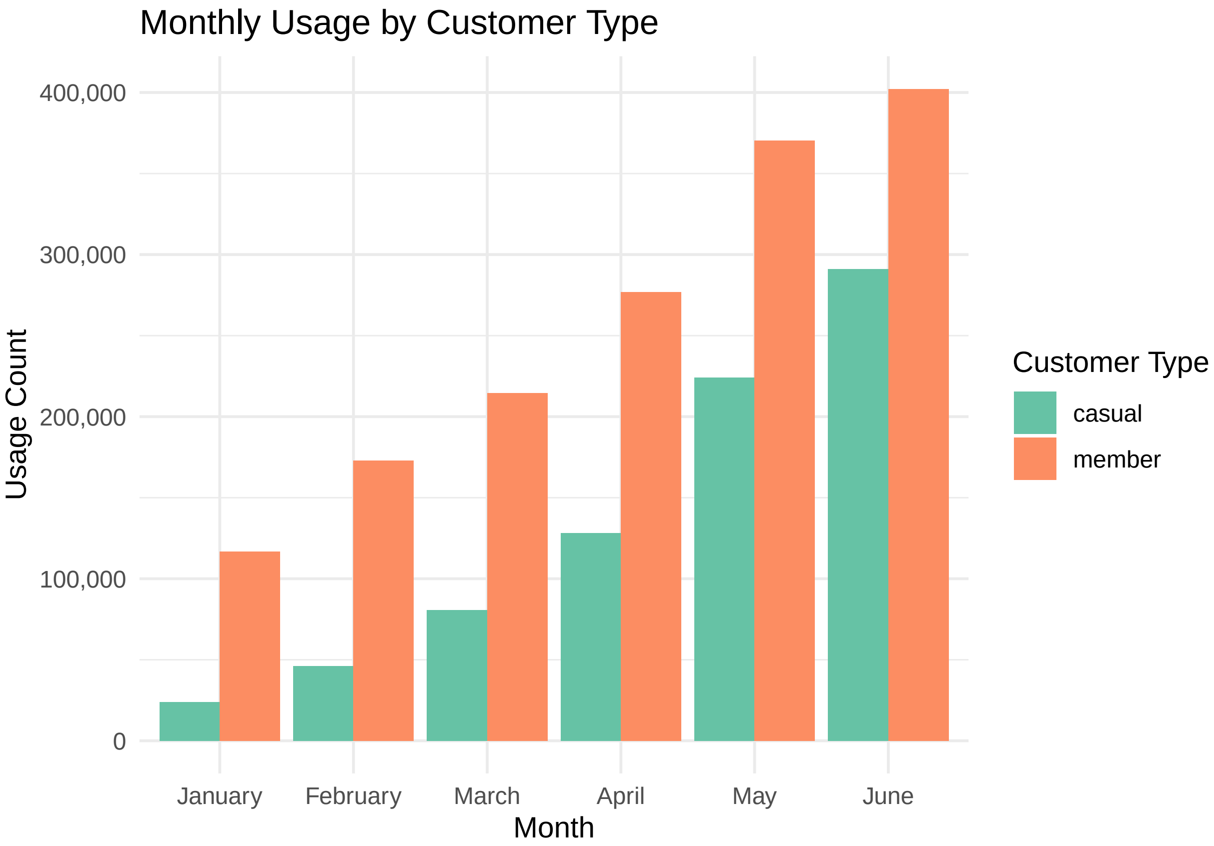
Modify the monthly order in the dataset for later use

```
# Define the month names
month_names <- c("January", "February", "March", "April", "May", "June",
                  "July", "August", "September", "October", "November", "December")
# Convert numeric month to month names
all_data$month <- month_names[as.numeric(all_data$month)]

# Convert month to ordered factor
all_data$month <- ordered(all_data$month, levels = month_names)
```

Monthly usage bu customer type

```
# Summarize the monthly data
month_summary <- all_data %>%
  group_by(month, member_casual) %>%
  summarise(count = n(), .groups = "drop")%>%
  arrange(month, member_casual)
```

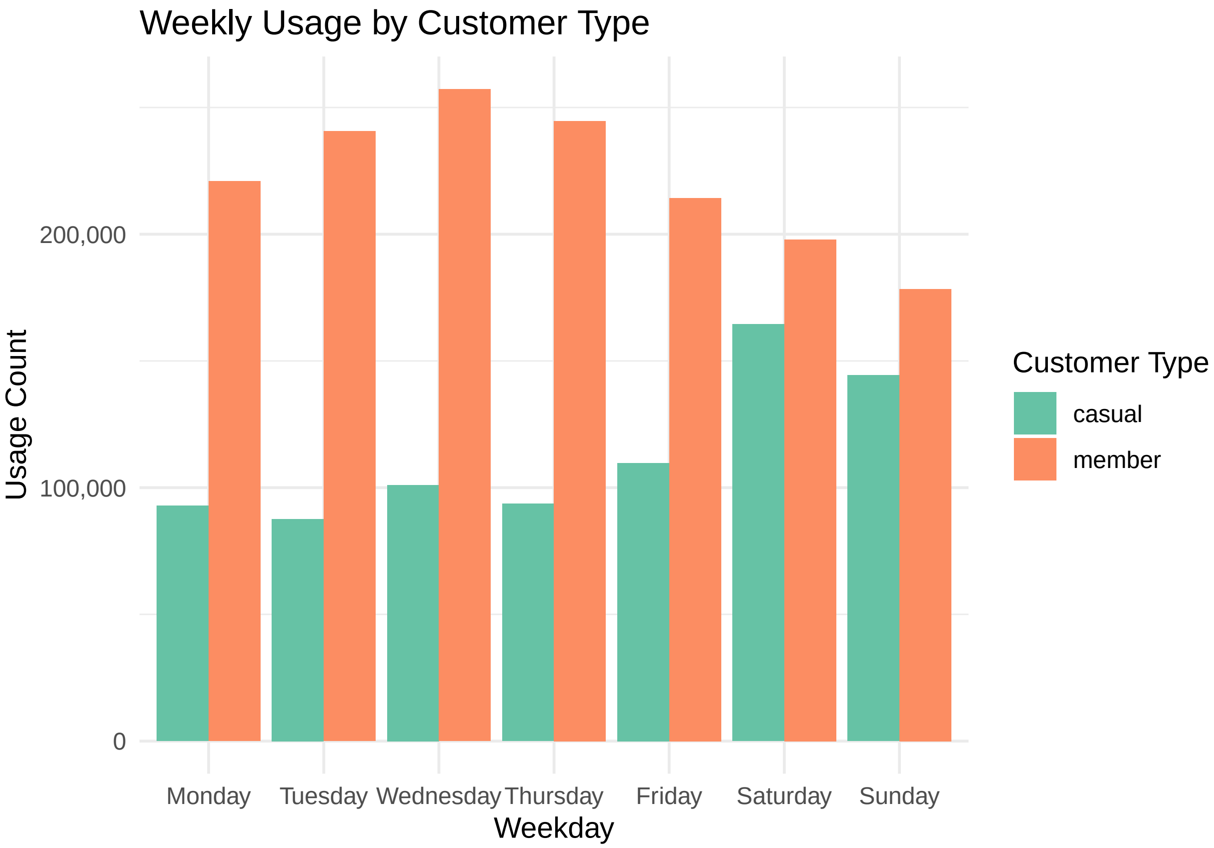
```
# Plot monthly data
ggplot(month_summary, aes(x = month, y = count, fill = member_casual)) +
    geom_bar(stat = "identity", position = "dodge") +
    labs(title = "Monthly Usage by Customer Type",
        x = "Month",
        y = "Usage Count",
        fill = "Customer Type") +
    scale_x_discrete(labels = month_names) +
    scale_y_continuous(labels = scales::comma) +
    scale_fill_brewer(palette = "Set2") +
    theme(axis.text.x = element_text(angle = 45, hjust = 1),
        legend.position = "bottom") +
    theme_minimal()
```



#

Weekly usage by customer type

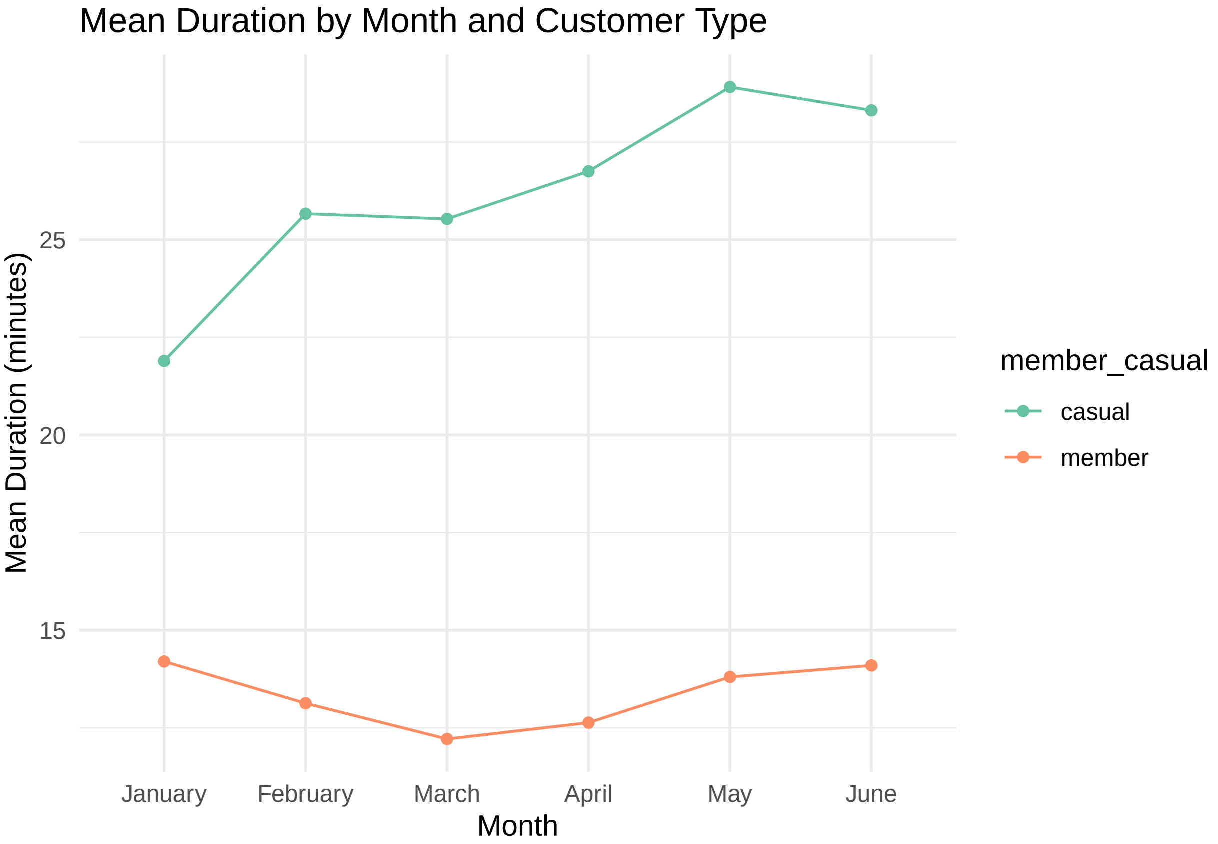
```
# Convert weekday to ordered factor
weekday_names <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday")
all_data$weekday <- ordered(all_data$weekday, levels = weekday_names)

# Summarize the weekly data
weekday_summary <- all_data %>%
  group_by(weekday, member_casual) %>%
  summarise(count = n(), .groups = "drop")%>%
  arrange(weekday, member_casual)

# Plot weekly data
ggplot(weekday_summary, aes(x = weekday, y = count, fill = member_casual)) +
    geom_bar(stat = "identity", position = "dodge") +
    labs(title = "Weekly Usage by Customer Type",
        x = "Weekday",
        y = "Usage Count",
        fill = "Customer Type") +
    scale_x_discrete(labels = weekday_names) +
    scale_y_continuous(labels = scales::comma) +
    scale_fill_brewer(palette = "Set2") +
    theme(axis.text.x = element_text(angle = 45, hjust = 1),
        legend.position = "bottom") +
    theme_minimal()
```



#

Mean duration by month and customer type

```
# Calculate mean duration by month and rider type
mean_duration_summary <- all_data %>%
  group_by(month, member_casual) %>%
  summarise(mean_duration = mean(duration), .groups = "drop")%>%
  ungroup

# Plot mean duration by month and rider type by line chart
ggplot(mean_duration_summary, aes(x = month, y = mean_duration, color = member_casual, group = member_ca
  geom_line() +
  labs(title = "Mean Duration by Month and Customer Type",
       x = "Month",
       y = "Mean Duration (minutes)") +
  scale_x_discrete(labels = month_names) +
  geom_point() +
  scale_color_brewer(palette = "Set2") +
  theme_minimal()
```



Base on the observations above, here are some suggestions:

Since casual users typically take longer rides and visit popular stations more frequently, it is likely that they are satisfied with the service. Create targeted campaigns offering discounts on memberships, highlighting the cost benefits of switching from casual rides to a membership plan.

Introduce loyalty programs that reward casual riders for repeated usage, encouraging them to consider the benefits of a membership. For instance, offer a free ride after a certain number of casual rides or discounts on future rides.

Offer limited-time trial memberships to casual riders. This allows them to experience the benefits of being a member, potentially leading to a conversion. Implement dynamic pricing where casual riders are charged more during peak hours or at popular stations. This can either increase revenue directly or encourage them to switch to a membership, which offers more predictable pricing.

Offer ride bundles for casual customers (e.g., buy 10 rides at a discounted rate), which can lock in future revenue while providing an incentive to ride more frequently.

Since casual customers use specific stations more frequently, consider enhancing the experience at these top stations. This could include better amenities, signage promoting memberships, or even staff assistance during peak times to encourage them to consider a membership.

Analyze why these top stations are popular and consider placing more stations in similar areas or enhancing the service in less-served areas to increase overall usage.

Offer promotions that cater to longer rides, such as scenic routes, guided tours, or partnerships with local attractions. Casual customers who already enjoy longer rides might be encouraged to take more trips if there's an added experience. (Reference PokemonGo)