Project for ‘Cyclistic’: Analysis Phase

Igor Vysochanskyy

2024-04-30

This R Markdown analyzes differences between members & casual bike users.

Install R packages & their libraries to enable subsequent operations.

Depending on your RStudio version, packages may be pre-installed, or you might need to install them manually.

```
library("tidyverse")
```

```
library("skimr")
```

Upload 12 datasets for each month of 2023

```
for (i in 301:312) {
  load(paste0("RData_CleanBike\\CleanBike", i, ".RData"))
}
```

Bind 12 months dataset into a single data frame

```
data2023 <- bind_rows(data301, data302, data303, data304, data305, data306, data307, data308, da
ta309, data310, data311, data312)
```

Detailed combined dataset observation

```
skim_without_charts(data2023)
```

Data summary

Name	data2023
Number of rows	5380725
Number of columns	6
_____	
Column type frequency:	
character	4
numeric	1
POSIXct	1

Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
ride_id	0	1	16	16	0	5380725	0
rideable_type	0	1	12	13	0	2	0
member_casual	0	1	6	6	0	2	0
day_of_week	0	1	3	3	0	7	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
ride_length	0	1	14.83	20.24	2	5.9	9.9	17	719.4

Variable type: POSIXct

skim_variable	n_missing	complete_rate	min	max	median	n_unique
started_at	0	1	2023-01-01 00:02:06	2023-12-31 23:58:55	2023-07-21 16:24:33	4082465

Plot 1. Average trip duration for members vs casuals combined

Group by month, member_casual, rideable_type & calc. average trip duration

```
avg_dur1 <- data2023 %>%
  mutate(month = format(started_at, "%B")) %>%  # Extract month from started_at column
  group_by(month, member_casual) %>%
  summarise(avg_dur1 = mean(ride_length), .groups = 'drop')
```

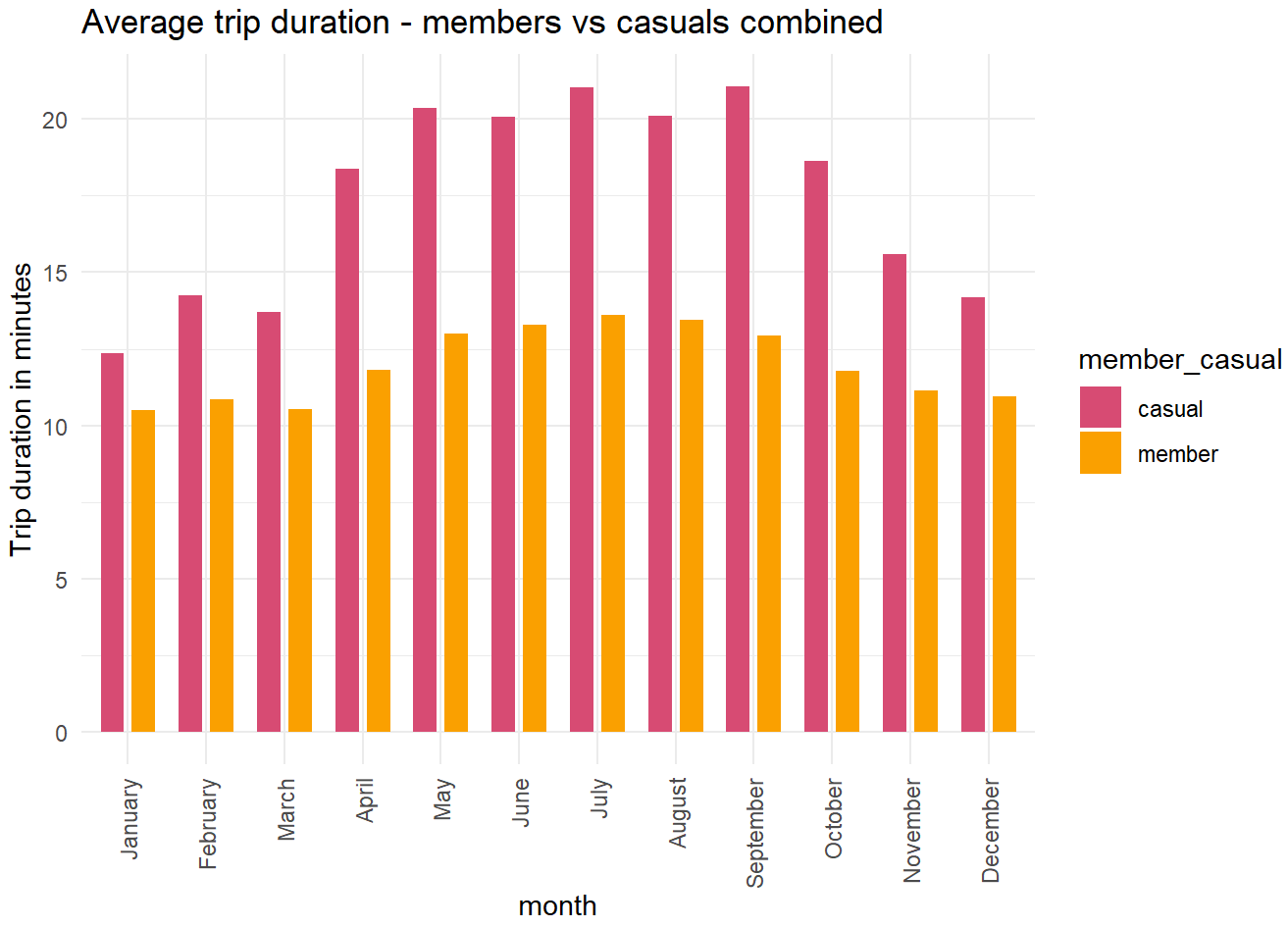
Define color palette 1

```
color_palette1 <- c("#D74B76", "#FAA300")
```

Convert ‘month’ to factor with custom levels in the desired order

```
avg_dur1$month <- factor(avg_dur1$month, levels = c(
  "January", "February", "March", "April", "May", "June",
  "July", "August", "September", "October", "November", "December"
))
```

```
ggplot(avg_dur1, aes(x = month, y = avg_dur1, fill = member_casual)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.8), width = 0.6) +
  scale_fill_manual(values = color_palette1) +
  labs(y = "Trip duration in minutes", title = "Average trip duration - members vs casuals combi
ned") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```



Plot2. Average trip duration & all types of bike users.

Group by the combination of all values from member_casual and rideable_type

```
avg_dur2 <- data2023 %>%
  mutate(month = format(started_at, "%B")) %>%
  group_by(month, type_of_users = paste(member_casual, rideable_type)) %>%
  summarise(avg_dur2 = mean(ride_length), .groups = 'drop')
```

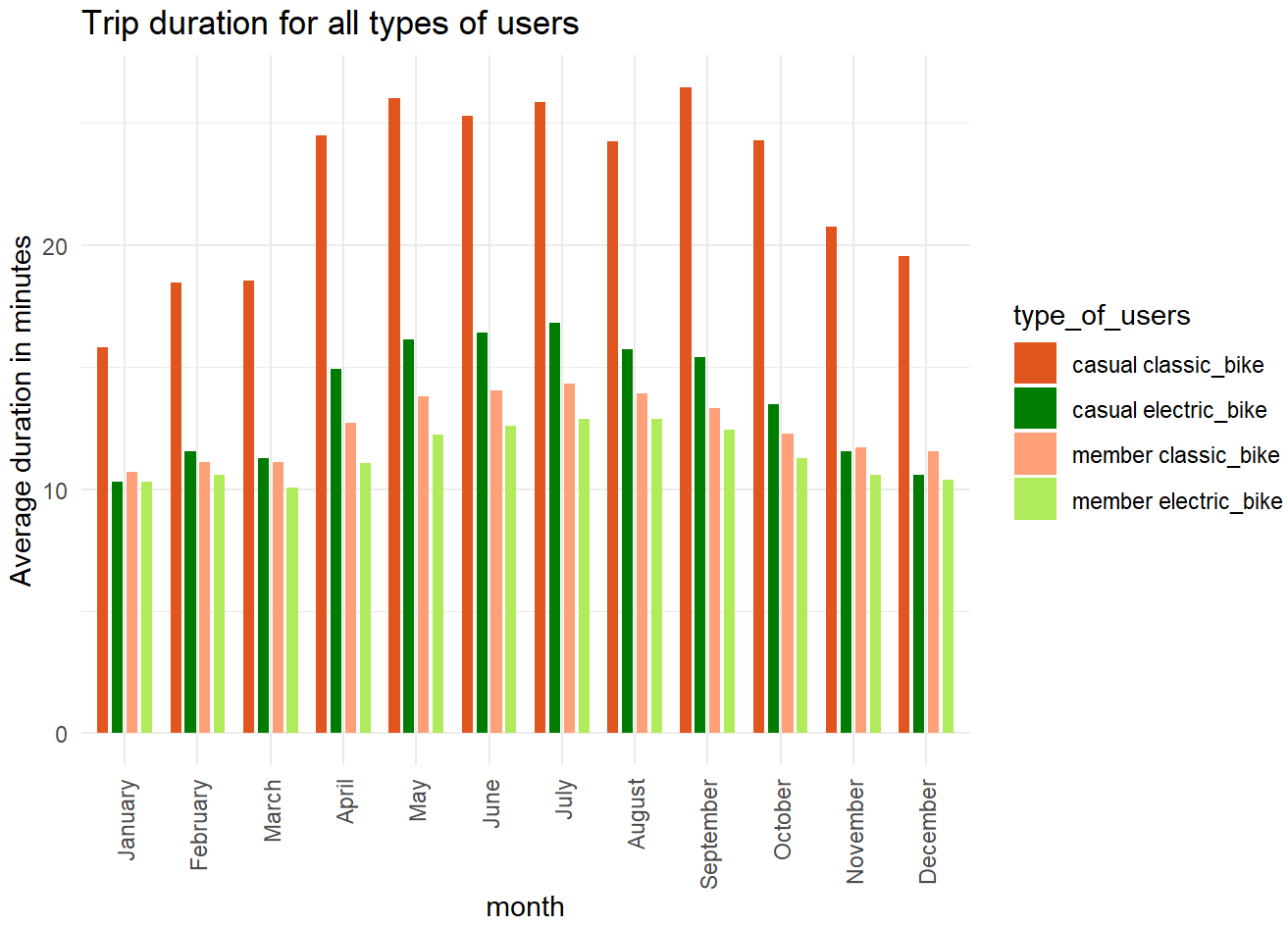
Define color palette 2 for each combination of user & bike type

```
color_palette2 <- c("#e25822", "#008000", "#ffa07a", "#b2ec5d")
```

Convert 'month' to factor with custom levels in the desired order

```
avg_dur2$month <- factor(avg_dur2$month, levels = c(
  "January", "February", "March", "April", "May", "June",
  "July", "August", "September", "October", "November", "December"))
```

```
ggplot(avg_dur2, aes(x = month, y = avg_dur2, fill = type_of_users)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.8), width = 0.6) +
  scale_fill_manual(values = color_palette2) +
  labs(y = "Average duration in minutes", title = "Trip duration for all types of users") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```



Plot3. Average trip duration for classic_bike between members & casuals.

Group by the combination of classic_bike with member_casual

```
avg_dur3 <- data2023 %>%
  mutate(month = format(started_at, "%B"),
         classic_bike_users = case_when(
           member_casual == "member" & rideable_type == "classic_bike" ~ "member_classic",
           member_casual == "casual" & rideable_type == "classic_bike" ~ "casual_classic")) %>%
  group_by(month, classic_bike_users) %>%
  summarise(avg_dur3 = mean(ride_length), .groups = 'drop') %>%
  filter(!is.na(classic_bike_users))
```

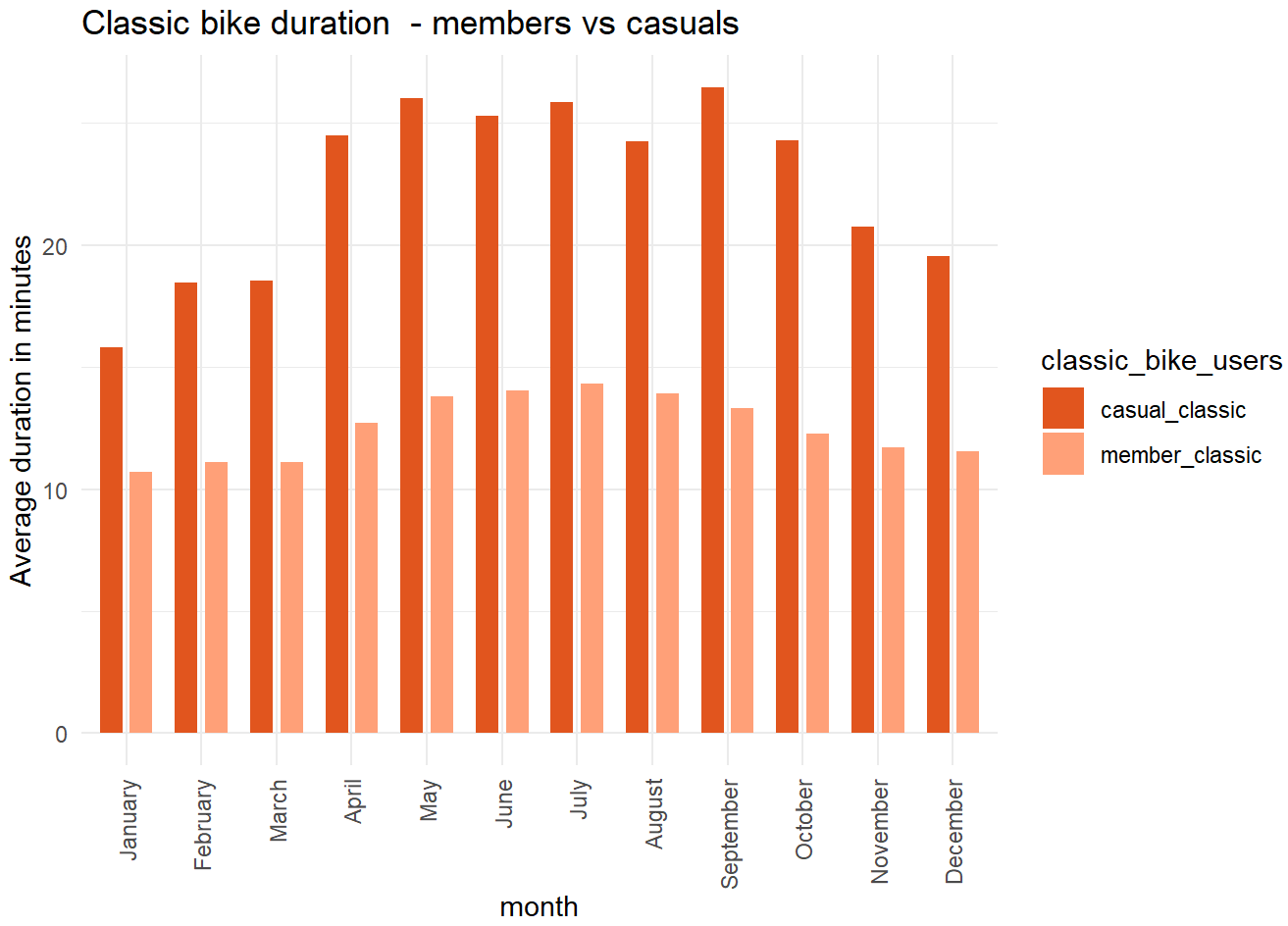
Define color palette 3

```
color_palette3 <- c("#e25822", "#ffa07a")
```

Convert ‘month’ to factor with custom levels in the desired order

```
avg_dur3$month <- factor(avg_dur3$month, levels = c(
  "January", "February", "March", "April", "May", "June",
  "July", "August", "September", "October", "November", "December"))
```

```
ggplot(avg_dur3, aes(x = month, y = avg_dur3, fill = classic_bike_users)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.8), width = 0.6) +
  scale_fill_manual(values = color_palette3) +
  labs(y = "Average duration in minutes", title = "Classic bike duration  - members vs casuals")
+
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```



Plot4. Average trip duration for electric_bike between members & casuals.

Group by the combination of electric_bike with member_casual

```
avg_dur4 <- data2023 %>%
  mutate(month = format(started_at, "%B"),
         electric_bike_users = case_when(
           member_casual == "member" & rideable_type == "electric_bike" ~ "member_electric",
           member_casual == "casual" & rideable_type == "electric_bike" ~ "casual_electric")) %
>%
  group_by(month, electric_bike_users) %>%
  summarise(avg_dur4 = mean(ride_length), .groups = 'drop') %>%
  filter(!is.na(electric_bike_users))
```

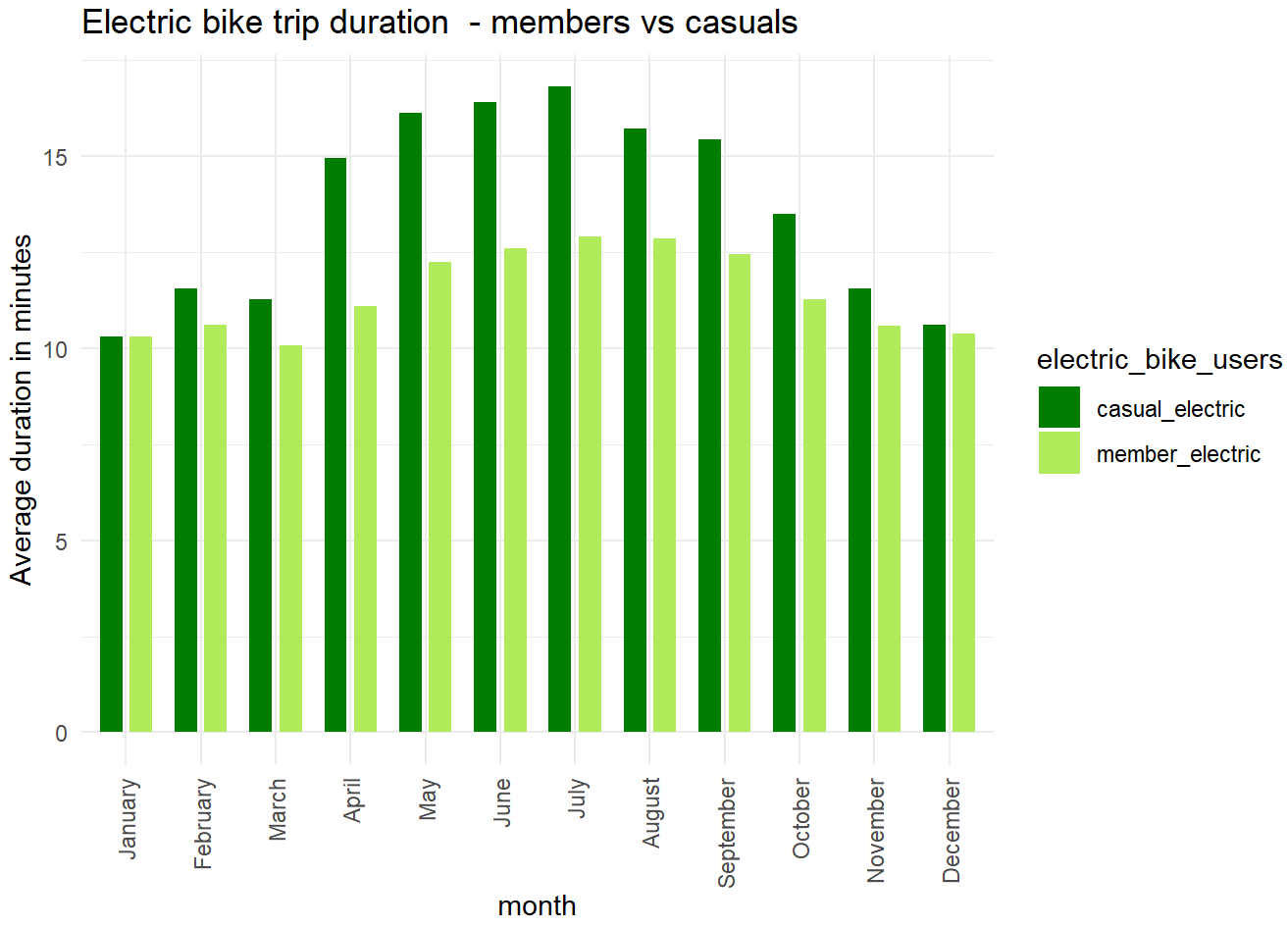
Define color palette 4

```
color_palette4 <- c("#008000", "#b2ec5d")
```

Convert 'month' to factor with custom levels in the desired order

```
avg_dur4$month <- factor(avg_dur4$month, levels = c(
  "January", "February", "March", "April", "May", "June",
  "July", "August", "September", "October", "November", "December"))
```

```
ggplot(avg_dur4, aes(x = month, y = avg_dur4, fill = electric_bike_users)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.8), width = 0.6) +
  scale_fill_manual(values = color_palette4) +
  labs(y = "Average duration in minutes", title = "Electric bike trip duration  - members vs cas
uals") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```



Plot5. Trip count for classic_bike between members & casuals.

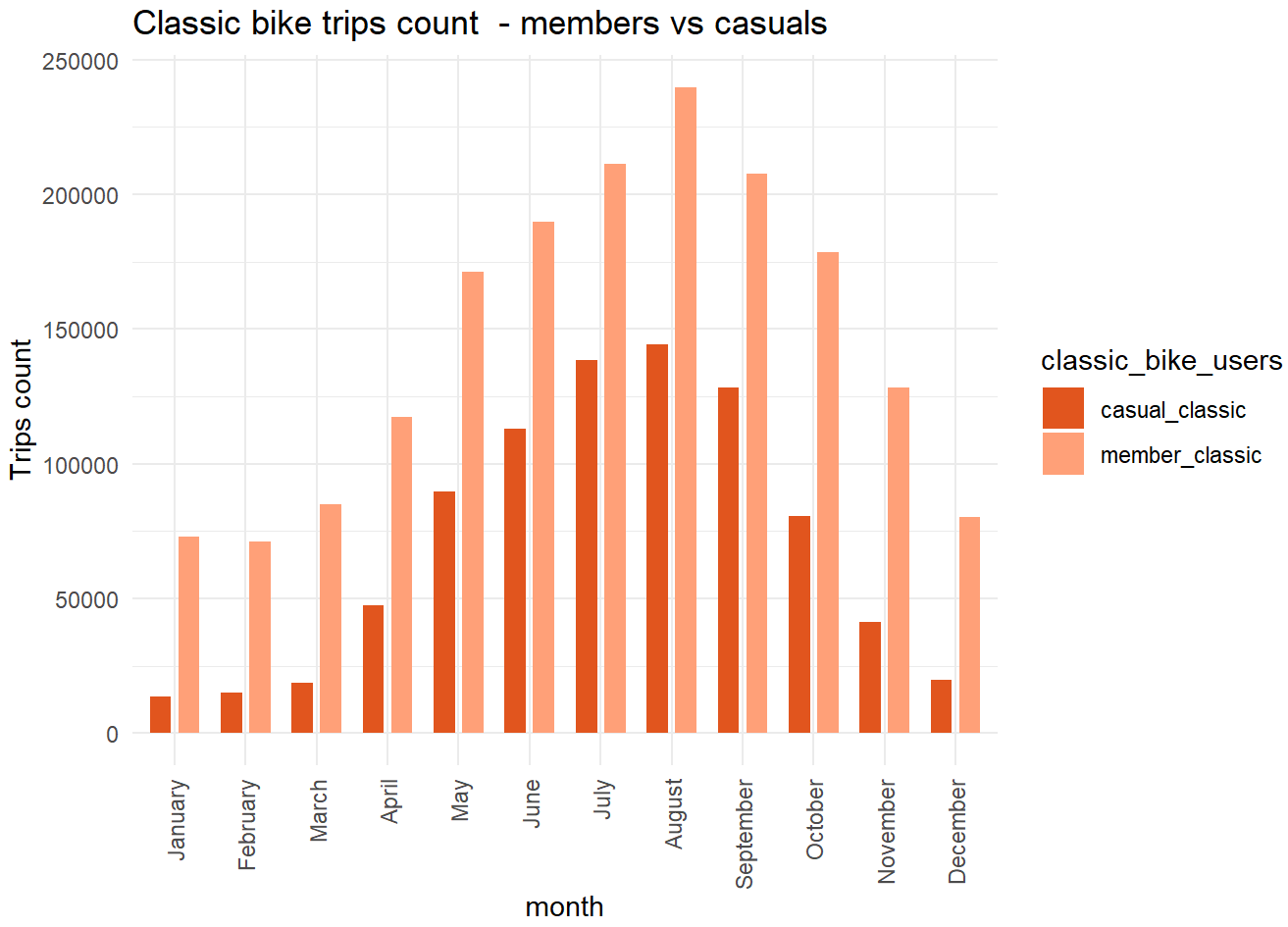
Group by the combination of classic_bike with member_casual

```
count_trips1 <- data2023 %>%
  mutate(month = format(started_at, "%B"),
         classic_bike_users = case_when(
           member_casual == "member" & rideable_type == "classic_bike" ~ "member_classic",
           member_casual == "casual" & rideable_type == "classic_bike" ~ "casual_classic")) %>%
  group_by(month, classic_bike_users) %>%
  summarise(count_trips1 = n(), .groups = 'drop') %>%
  filter(!is.na(classic_bike_users))
```

Convert 'month' to factor with custom levels in the desired order

```
count_trips1$month <- factor(count_trips1$month, levels = c(
  "January", "February", "March", "April", "May", "June",
  "July", "August", "September", "October", "November", "December"))
```

```
ggplot(count_trips1, aes(x = month, y = count_trips1,
                         fill = classic_bike_users)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.8),
  width = 0.6) +
  scale_fill_manual(values = color_palette3) +
  labs(title = "Classic bike trips count  - members vs casuals", y = "Trips count") +
  theme_minimal() +
   theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```



Plot6. Trip count for electric_bike between members & casuals.

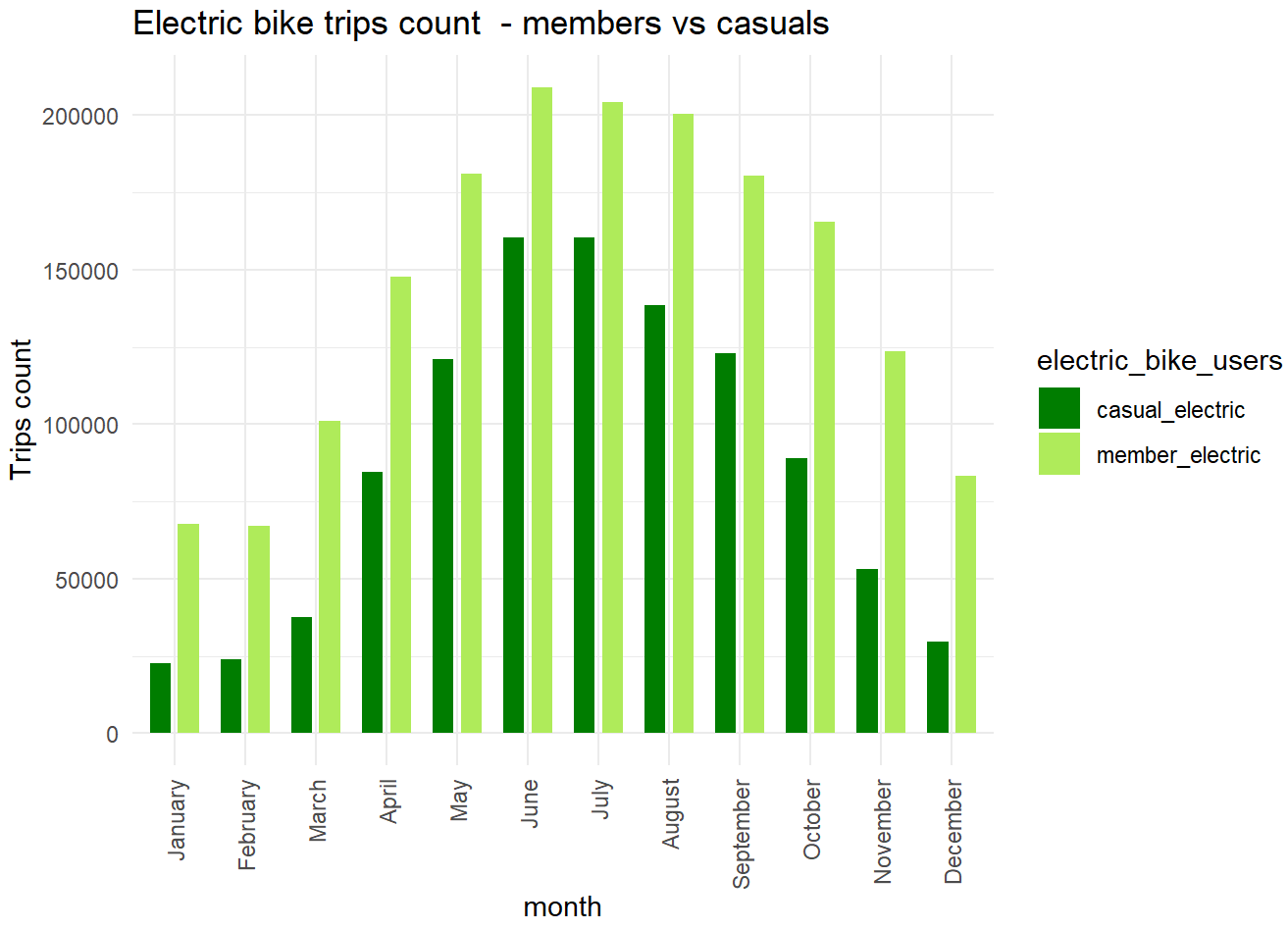
Group by the combination of electric_bike with member_casual

```
count_trips2 <- data2023 %>%
  mutate(month = format(started_at, "%B"),
         electric_bike_users = case_when(
           member_casual == "member" & rideable_type == "electric_bike" ~ "member_electric",
           member_casual == "casual" & rideable_type == "electric_bike" ~ "casual_electric")) %
>%
  group_by(month, electric_bike_users) %>%
  summarise(count_trips2 = n(), .groups = 'drop') %>%
  filter(!is.na(electric_bike_users))
```

Convert 'month' to factor with custom levels in the desired order

```
count_trips2$month <- factor(count_trips2$month, levels = c(
  "January", "February", "March", "April", "May", "June",
  "July", "August", "September", "October", "November", "December"))
```

```
ggplot(count_trips2, aes(x = month, y = count_trips2,
                         fill = electric_bike_users)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.8),
  width = 0.6) +
  scale_fill_manual(values = color_palette4) +
  labs(title = "Electric bike trips count  - members vs casuals", y = "Trips count") +
  theme_minimal() +
   theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```



Plot 7. Trip count for weekdays between members & casuals.

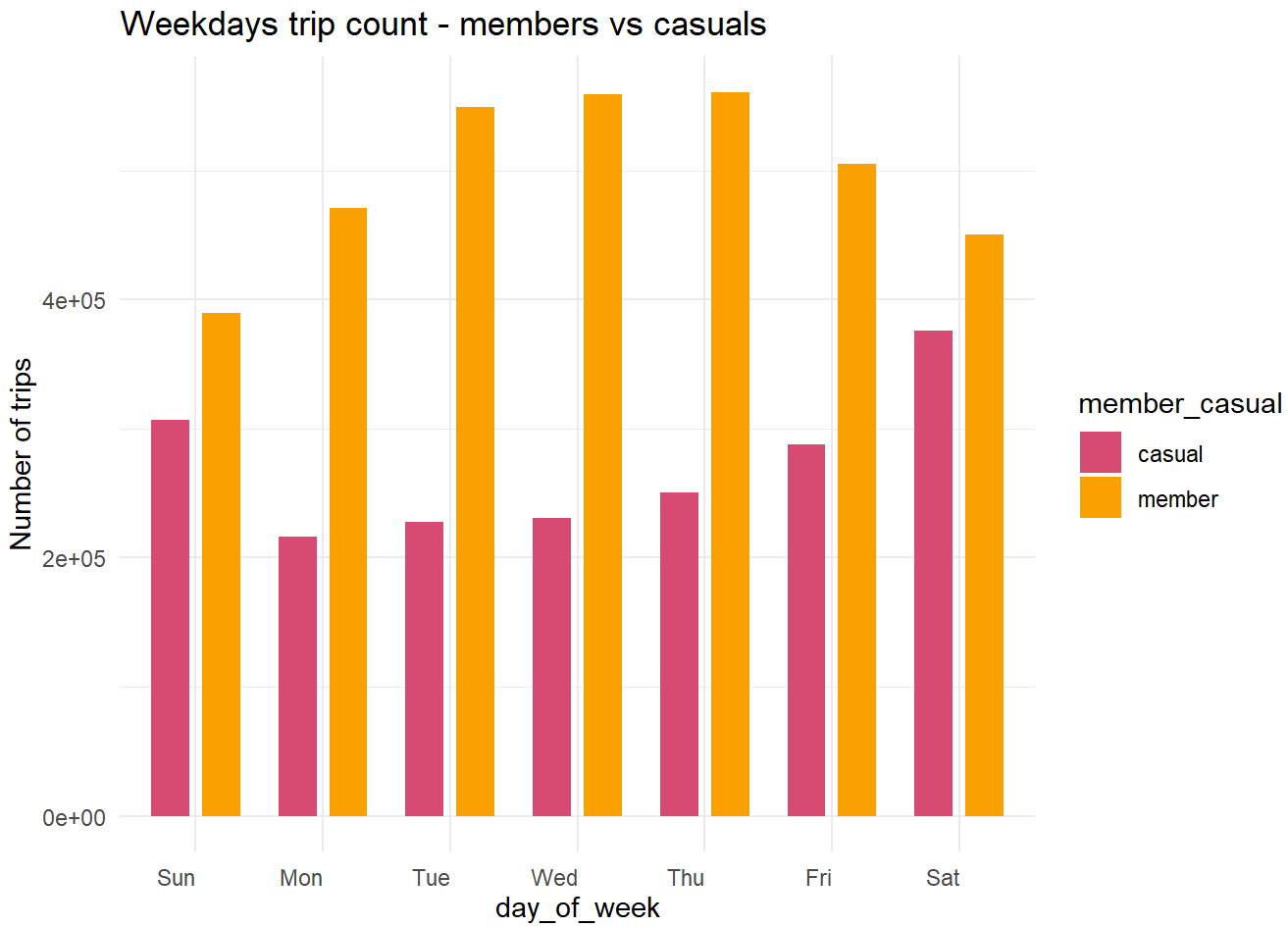
Group by the combination of day_of_week with member_casual

```
count_trips7 <- data2023 %>%
  group_by(day_of_week, member_casual) %>%
  summarise(count_trips7 = n(), .groups = 'drop') %>%
  filter(!is.na(member_casual))
```

Convert 'day_of_week' to factor with custom levels in the desired order

```
count_trips7$day_of_week <- factor(count_trips7$day_of_week, levels = c(
  "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"))
```

```
ggplot(count_trips7, aes(x = day_of_week, y = count_trips7,
                         fill = member_casual)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.8),
  width = 0.6) +
  scale_fill_manual(values = color_palette1) +
  labs(title = "Weekdays trip count - members vs casuals", y = "Number of trips") +
  theme_minimal() +
   theme(axis.text.x = element_text(angle = 0, vjust = 0.5, hjust=1))
```



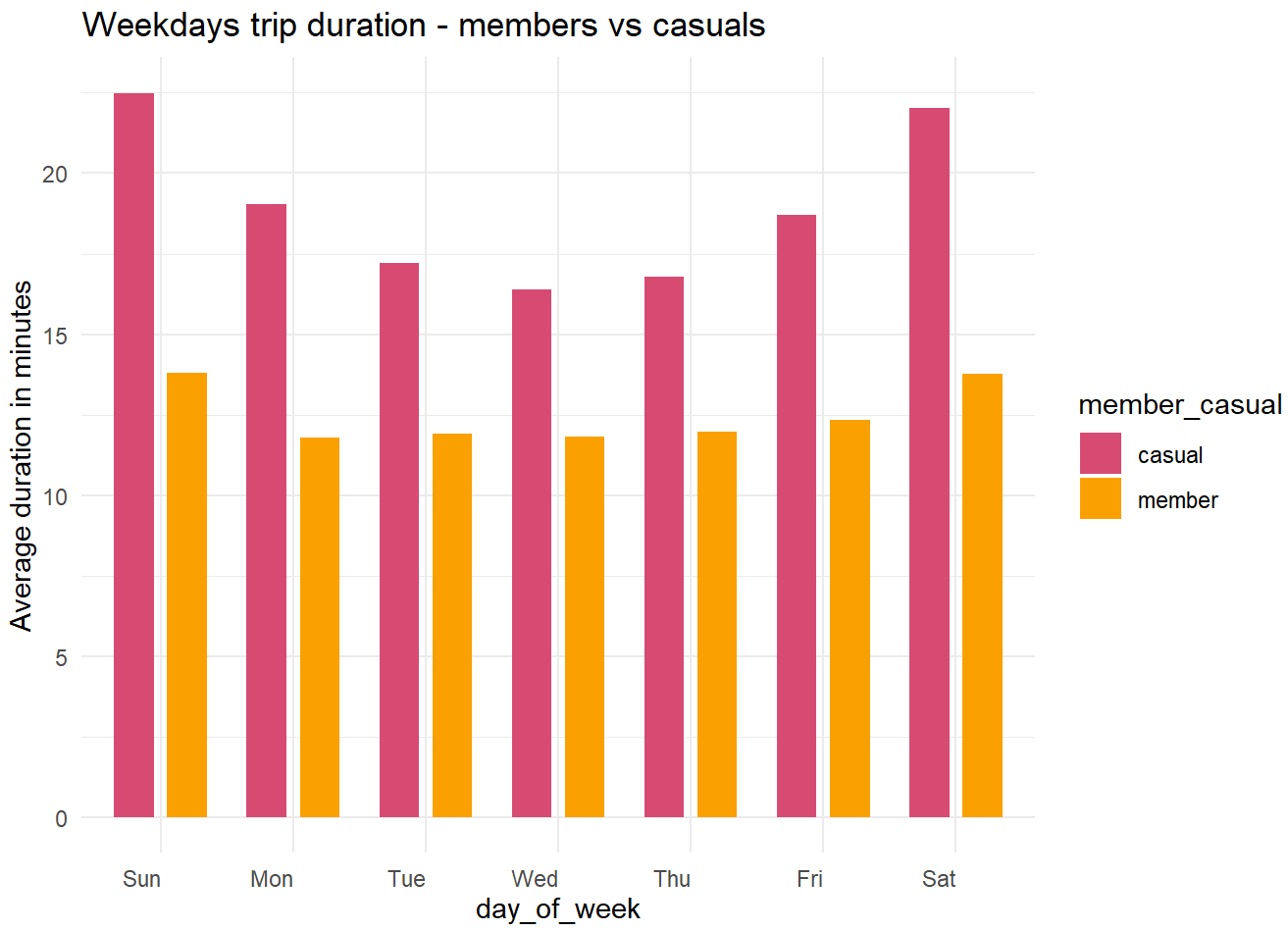
Plot 8. Duration for weekdays between members & casuals.

```
day_dur8 <- data2023 %>%
  group_by(day_of_week, member_casual) %>%
  summarise(day_dur8 = mean(ride_length), .groups = 'drop') %>%
  filter(!is.na(member_casual))
```

Convert ‘day_of_week’ to factor with custom levels in the desired order

```
day_dur8$day_of_week <- factor(day_dur8$day_of_week, levels = c(
  "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"))
```

```
ggplot(day_dur8, aes(x = day_of_week, y = day_dur8,
                         fill = member_casual)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.8),
  width = 0.6) +
  scale_fill_manual(values = color_palette1) +
  labs(title = "Weekdays trip duration - members vs casuals", y = "Average duration in minutes")
+
  theme_minimal() +
   theme(axis.text.x = element_text(angle = 0, vjust = 0.5, hjust=1))
```

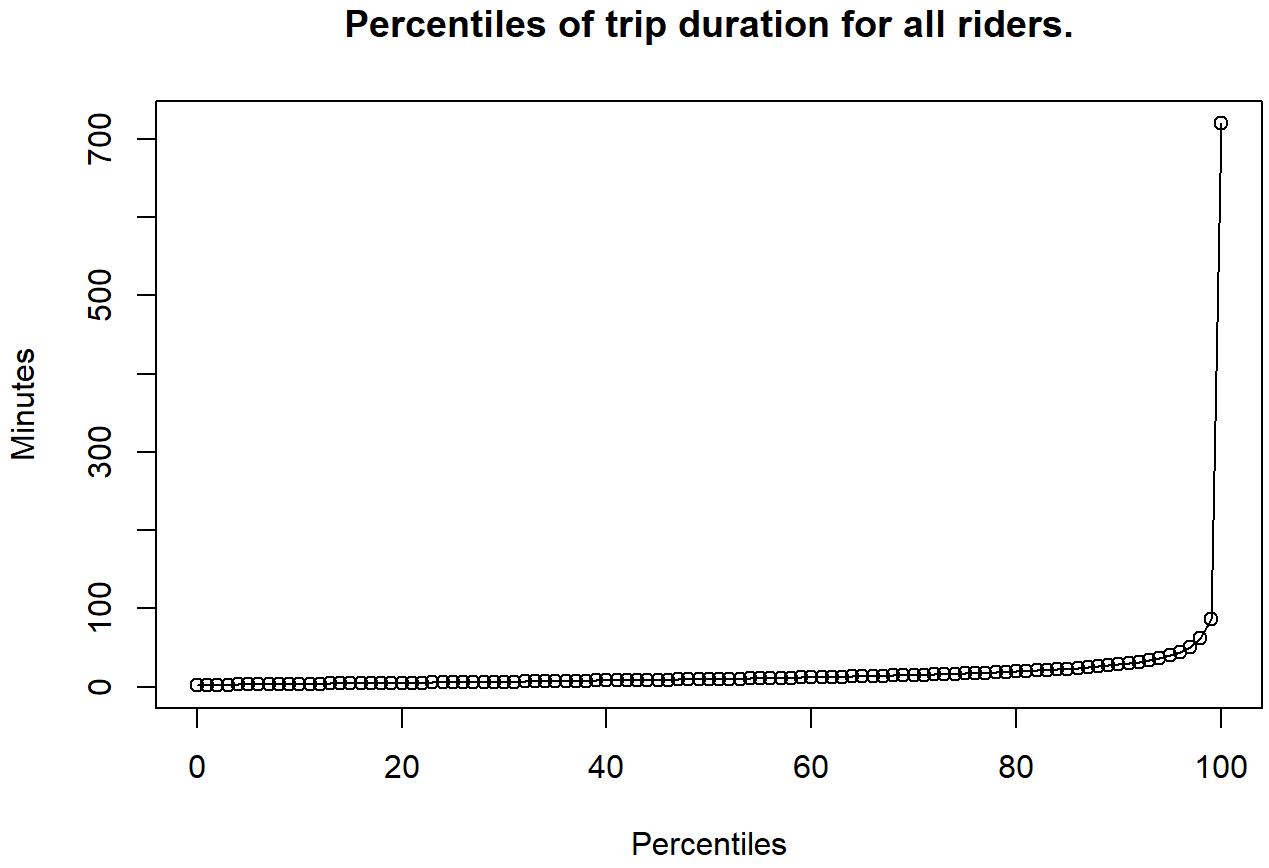


Plot 9. Percentiles of trip duration for all riders.

Define the percentiles with smaller increments

```
percentiles1 <- seq(0, 1, by = 0.01)
values1 <- quantile(data2023$ride_length, probs = percentiles1)
```

```
plot(percentiles1 * 100, values1, type = "o",
     xlab = "Percentiles", ylab = "Minutes",
     main = "Percentiles of trip duration for all riders.")
```



Actual percentiles of trip duration for casuals:

Define the percentiles with smaller increments

```
percentiles2 <- seq(0.97, 1, by = 0.005)
values2 <- quantile(subset(data2023, member_casual == "casual")$ride_length,
                    probs = percentiles2)
```

Create a data frame with percentiles and values

```
perc_casual2 <- data.frame(Percentile = percentiles2, Value = values2)
```

Print the data frame for casuals

```
print(perc_casual2)
```

```
##       Percentile Value
## 97%        0.970  75.1
## 97.5%      0.975  81.4
## 98%        0.980  89.7
## 98.5%      0.985 101.1
## 99%        0.990 118.5
## 99.5%      0.995 153.2
## 100%       1.000 719.4
```

Percentiles of trip duration for casual riders, from 60% to 100%.”

```
percentiles3 <- seq(0.6, 1, by = 0.05)
values3 <- quantile(subset(data2023, member_casual == "casual")$ride_length,
                    probs = percentiles3)
```

Create a data frame with percentiles and values

```
perc_casual3 <- data.frame(Percentile = percentiles3, Value = values3)
```

Print the data frame for casual riders

```
print(perc_casual3)
```

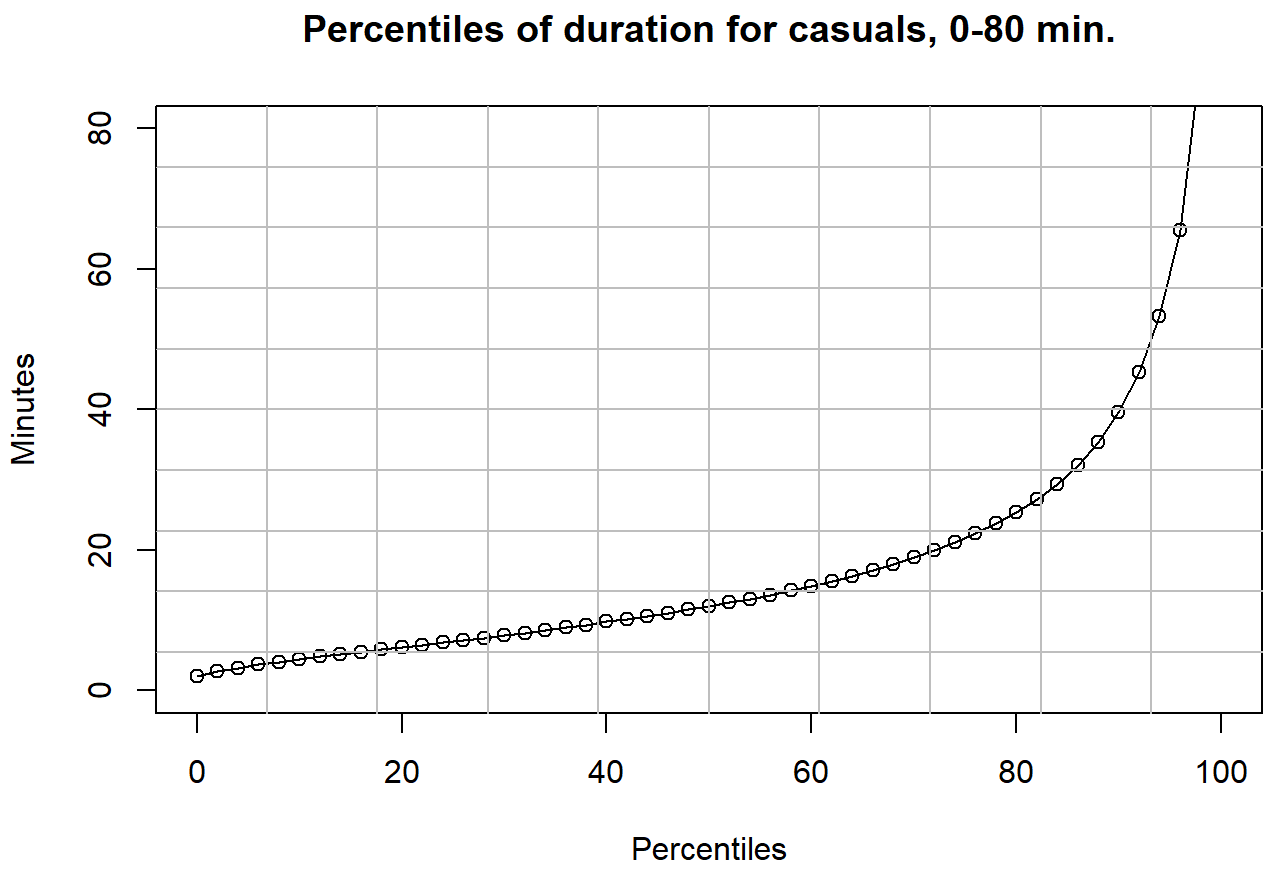
```
##      Percentile Value
## 60%        0.60  14.9
## 65%        0.65  16.7
## 70%        0.70  19.0
## 75%        0.75  21.8
## 80%        0.80  25.4
## 85%        0.85  30.7
## 90%        0.90  39.6
## 95%        0.95  58.6
## 100%       1.00 719.4
```

Plot 10. Trip duration 0-80 min. in percentiles for casuals

Define the percentiles with smaller increments

```
percentiles4 <- seq(0, 1, by = 0.02)
values4 <- quantile(subset(data2023, member_casual == "casual")$ride_length,
                    probs = percentiles4)
```

```
plot(percentiles4 * 100, values4, type = "o",
     xlab = "Percentiles", ylab = "Minutes",
     main = "Percentiles of duration for casuals, 0-80 min.",
     ylim = c(0, 80))
grid(nx = 10, ny = 10, lty = 1, col = "gray", lwd = 1)
```



Actual percentiles of trip duration for members:

Define the percentiles with smaller increments

```
percentiles5 <- seq(0.97, 1, by = 0.005)
values5 <- quantile(subset(data2023, member_casual == "member")$ride_length,
                    probs = percentiles5)
```

Create a data frame with percentiles and values

```
perc_member5 <- data.frame(Percentile = percentiles5, Value = values5)
```

Print the data frame for members

```
print(perc_member5)
```

```
##       Percentile Value
## 97%        0.970  37.5
## 97.5%      0.975  39.6
## 98%        0.980  42.0
## 98.5%      0.985  45.5
## 99%        0.990  52.1
## 99.5%      0.995  70.0
## 100%       1.000 718.1
```

Plot 11. Trip duration with data labels for all riders by quarters.

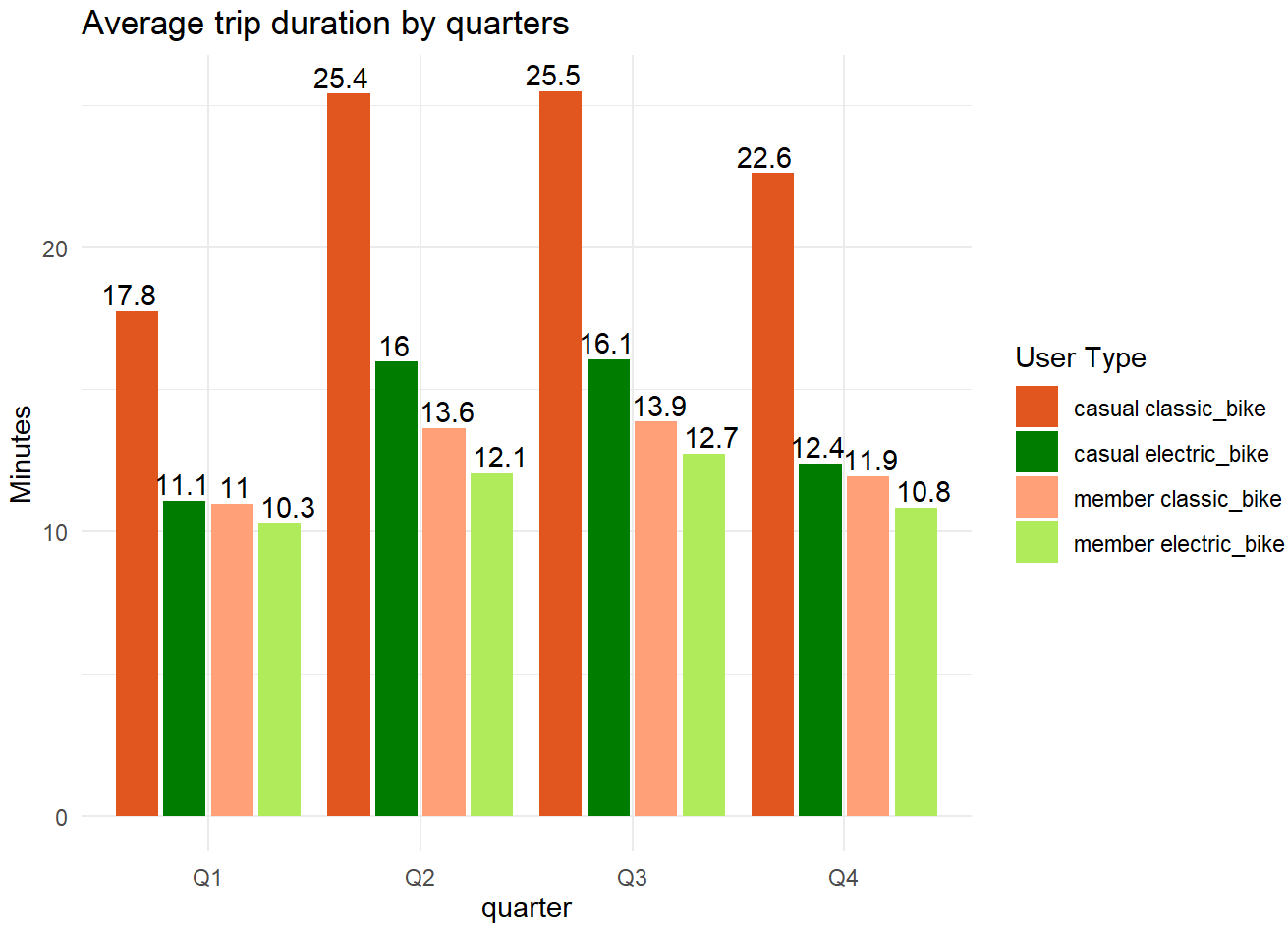
Define a function to convert month to quarter

```
month_to_quarter <- function(month) {
  case_when(
    month %in% c("January", "February", "March") ~ "Q1",
    month %in% c("April", "May", "June") ~ "Q2",
    month %in% c("July", "August", "September") ~ "Q3",
    TRUE ~ "Q4"
  )
}
```

Aggregate data by quarter

```
avg_dur_quarter <- data2023 %>%
  mutate(month = format(started_at, "%B")) %>%
  mutate(quarter = month_to_quarter(month)) %>%
  group_by(quarter, type_of_users = paste(member_casual, rideable_type)) %>%
  summarise(avg_dur_quarter = mean(ride_length), .groups = 'drop')
```

```
ggplot(avg_dur_quarter, aes(x = quarter, y = avg_dur_quarter, fill = type_of_users, label = roun
d(avg_dur_quarter, 1 ))) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.9), width = 0.8) +
  geom_text(position = position_dodge(width = 1), vjust = -0.3) +
  scale_fill_manual(values = color_palette2) +
  labs(y = "Minutes", title = "Average trip duration by quarters") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 0, vjust = 0.5)) +
  guides(fill = guide_legend(title = "User Type"))
```



Plot 12. Pie chart for trip number of members vs casuals in %.

Grouping & aggregation

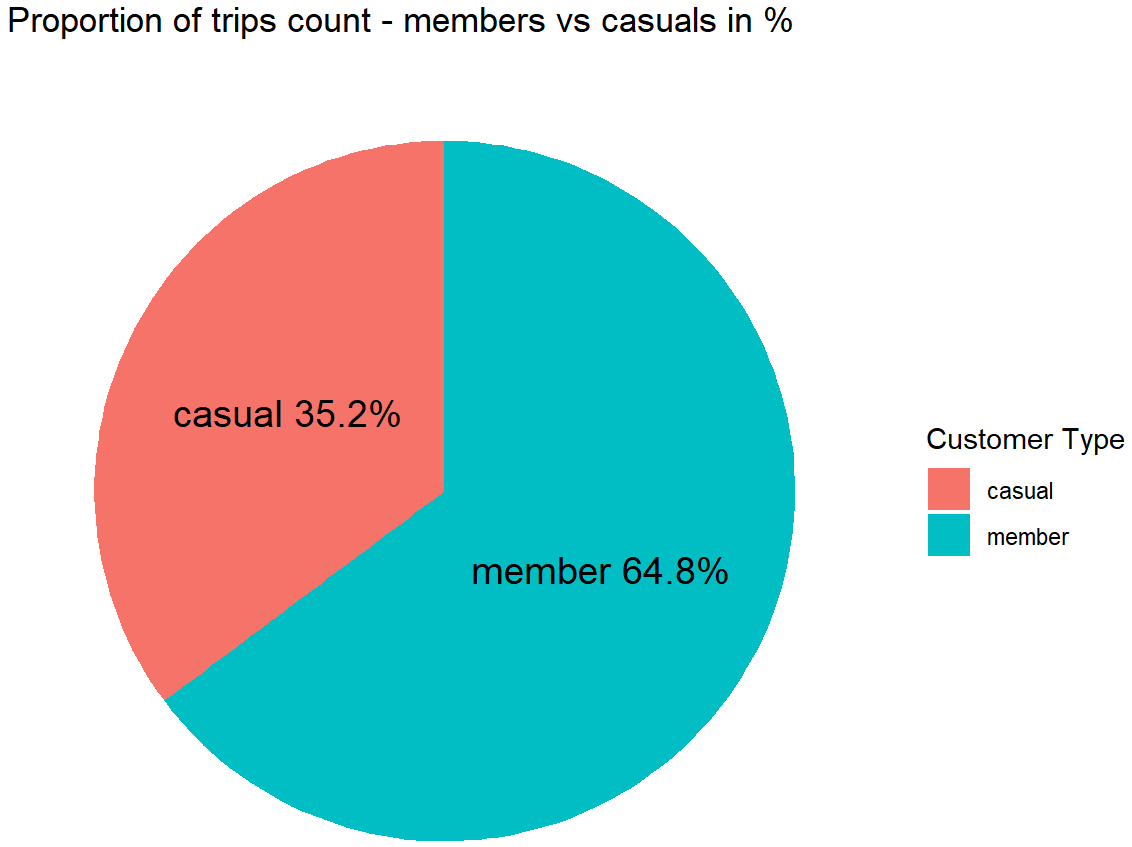
```
data2023_pie1 <- data2023 %>%
  group_by(member_casual) %>%
  summarise(count = n()) %>%
  mutate(percentage = count / sum(count) * 100)
```

Calculation of the ratio of casuals to members from the aggregated data

```
# Ensure data2023_pie1 has the correct format
casual_count <- data2023_pie1 %>% filter(member_casual == "casual") %>% pull(count)
member_count <- data2023_pie1 %>% filter(member_casual == "member") %>% pull(count)

# Calculate the ratio
unweighted_yearly_ratio <- casual_count / member_count
unweighted_yearly_ratio
```

```
## [1] 0.5436401
```

```
ggplot(data2023_pie1, aes(x = "", y = percentage, fill = member_casual)) +
  geom_bar(width = 1, stat = "identity") +
  geom_text(aes(label = paste(member_casual, sprintf("%.1f%%", percentage))), position = positio
n_stack(vjust = 0.5), size = 5) +
  coord_polar(theta = "y") +
  labs(fill = "Customer Type") +
  ggtitle("Proportion of trips count - members vs casuals in %") +
  theme_minimal() +
  theme(legend.position = "right",
        axis.line = element_blank(),
        axis.text = element_blank(),
        axis.title = element_blank(),
        panel.grid = element_blank())
```

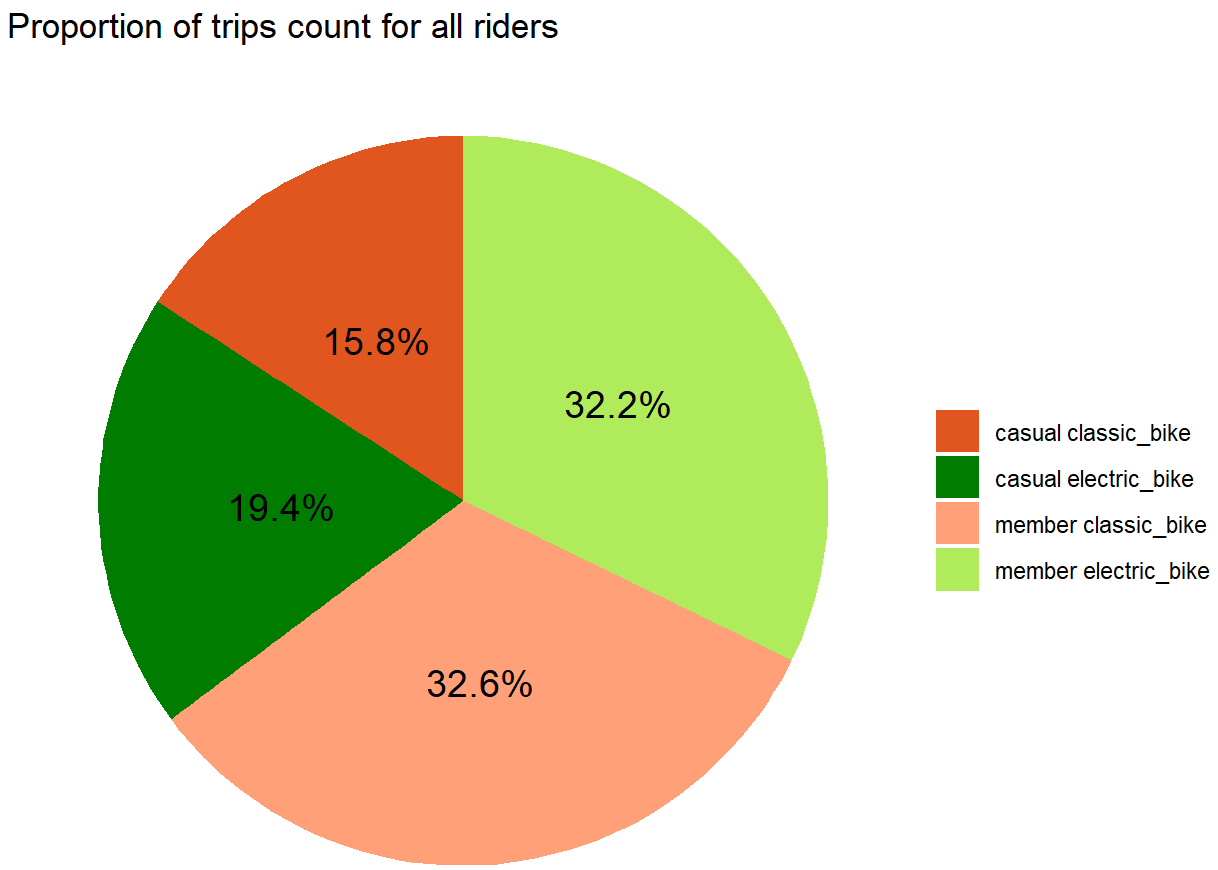


Plot 13. Pie chart for trip number of all riders in %.

Grouping & aggregation

```
trips_count <- data2023 %>%
  group_by(type_of_users = paste(member_casual, rideable_type)) %>%
  summarise(count = n(), .groups = 'drop')
```

```
ggplot(trips_count, aes(x = "", y = count, fill = type_of_users, label = sprintf("%.1f%%", coun
t/sum(count)*100))) +
  geom_bar(width = 1, stat = "identity") +
  coord_polar(theta = "y") +
  scale_fill_manual(values = color_palette2) +
  labs(fill = "User Type") +
  ggtitle("Proportion of trips count for all riders") +
  theme_void() +
  geom_text(position = position_stack(vjust = 0.5), size = 5) +
  theme(legend.position = "right",
        legend.title = element_blank())
```



Plot 14. Trip duration with data labels for all riders by seasons.

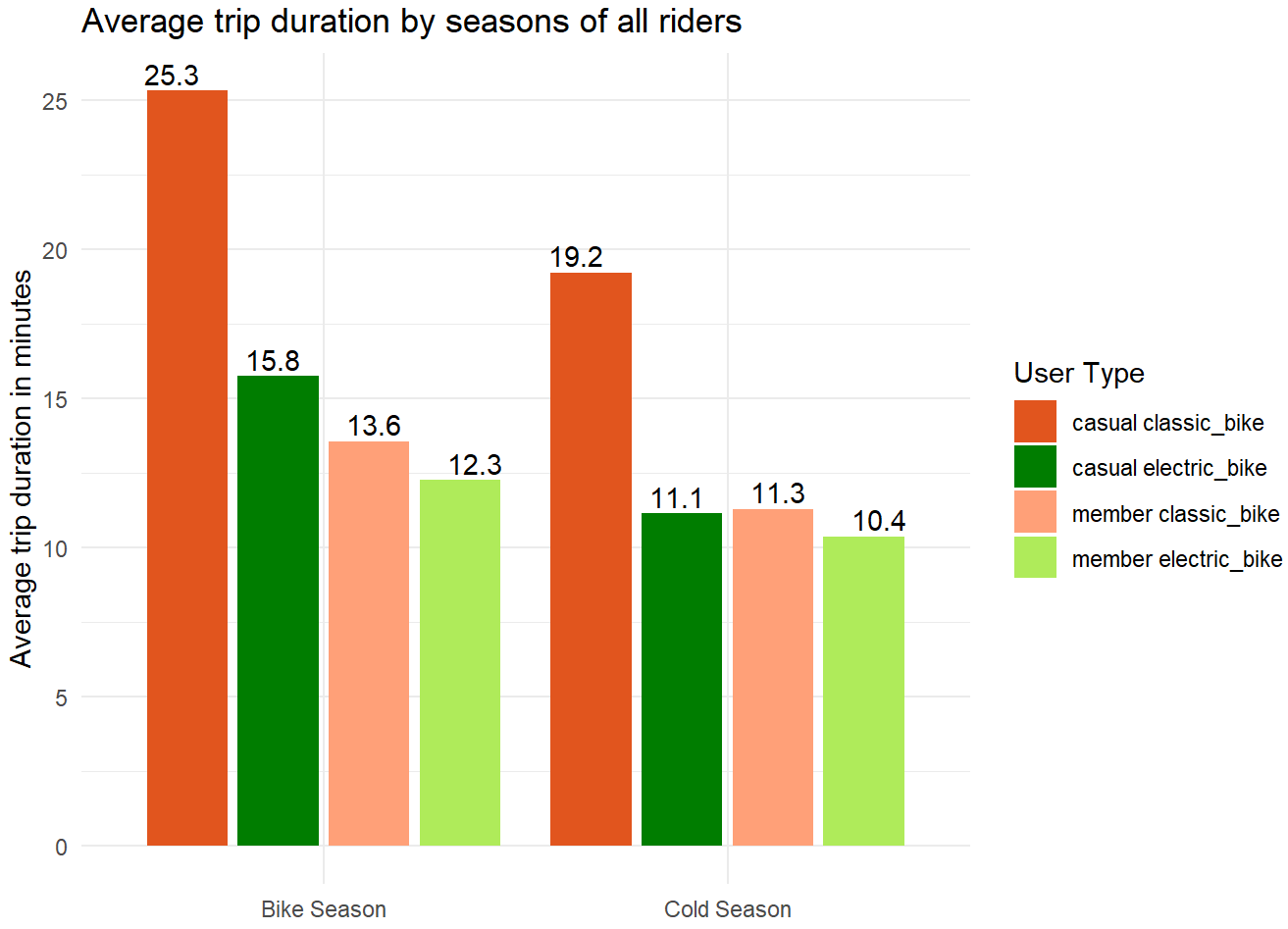
Define a function to convert month to season

```
month_to_season <- function(month) {
  case_when(
    month %in% c("January", "February", "March", "November", "December") ~ "Cold Season",
        TRUE ~ "Bike Season"
  )
}
```

Aggregate data by seasons

```
avg_dur_season <- data2023 %>%
  mutate(month = format(started_at, "%B")) %>%
  mutate(season = month_to_season(month)) %>%
  group_by(season, type_of_users = paste(member_casual, rideable_type)) %>%
  summarise(avg_dur_season = mean(ride_length), .groups = 'drop')
```

```
ggplot(avg_dur_season, aes(x = season, y = avg_dur_season, fill = type_of_users, label = round(a
vg_dur_season, 1 ))) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.9), width = 0.8) +
  geom_text(position = position_dodge(width = 1), vjust = -0.3) +
  scale_fill_manual(values = color_palette2) +
  labs(x = NULL, y = "Average trip duration in minutes", title = "Average trip duration by seaso
ns of all riders") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 0, vjust = 0.5)) +
  guides(fill = guide_legend(title = "User Type"))
```



Plot 15. Trip count with data labels for all riders by seasons.

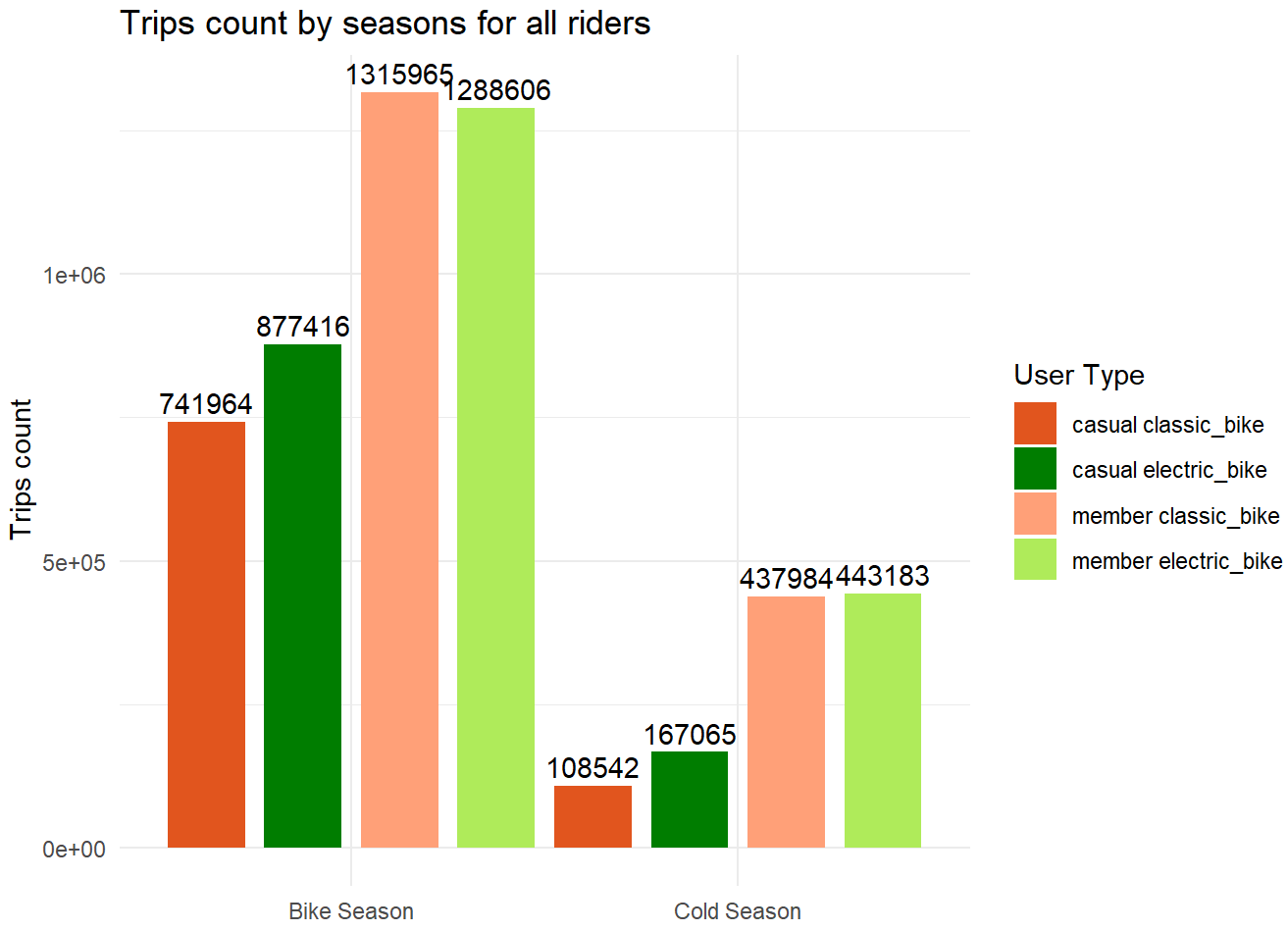
Define a function to convert month to season

```
month_to_season <- function(month) {
  case_when(
    month %in% c("January", "February", "March", "November", "December") ~ "Cold Season",
        TRUE ~ "Bike Season"
  )
}
```

Aggregate data by seasons

```
tr_count_season <- data2023 %>%
  mutate(month = format(started_at, "%B")) %>%
  mutate(season = month_to_season(month)) %>%
  group_by(season, type_of_users = paste(member_casual, rideable_type)) %>%
  summarise(tr_count_season = n(), .groups = 'drop')
```

```
ggplot(tr_count_season, aes(x = season, y = tr_count_season, fill = type_of_users, label = round
(tr_count_season, 0 ))) +
  geom_bar(stat = "identity", position = position_dodge(width = 1.0), width = 0.8) +
  geom_text(position = position_dodge(width = 1.0), vjust = -0.4) +
  scale_fill_manual(values = color_palette2) +
  labs(x = NULL, y = "Trips count", title = "Trips count by seasons for all riders") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 0, vjust = 0.5)) +
  guides(fill = guide_legend(title = "User Type"))
```



Plot 16. Weekly, Monthly, Yearly, Bike and Cold seasons ratios between Casuals and Members.

Create a week column and calculate the weekly ratio

```
weekly_counts <- data2023 %>%
  mutate(week = floor_date(started_at, unit = "week")) %>%  # Create 'week' column
  group_by(week, member_casual) %>%
  summarise(count = n(), .groups = "drop") %>%
  pivot_wider(names_from = member_casual, values_from = count, values_fill = 0) %>%
  mutate(ratio = casual / member)  # Calculate the ratio
```

Calculate monthly averages from weekly_counts

```
monthly_avg <- weekly_counts %>%
  mutate(month = floor_date(week, unit = "month")) %>%
  group_by(month) %>%
  summarise(monthly_ratio_avg = mean(ratio, na.rm = TRUE), .groups = "drop")
```

Calculate the Weighted yearly average based on weeks average

```
yearly_avg <- mean(weekly_counts$ratio, na.rm = TRUE)
yearly_avg
```

```
## [1] 0.480799
```

Ensure all date-related columns (week and month) are in Date format:

```
weekly_counts <- weekly_counts %>%
  mutate(
    week = as.Date(week)  # Ensure 'week' is a Date object
  )

monthly_avg <- monthly_avg %>%
  mutate(
    month = as.Date(month)  # Ensure 'month' is a Date object
  )
```

Group by the months in each range and calculate the averages for the ratio column:

```
# Averages for the specified month ranges
selected_avg <- monthly_avg %>%
  mutate(
    range_label = case_when(
      month >= as.Date("2023-04-01") & month <= as.Date("2023-10-31") ~ "Bike Season",  # April
to October
      month %in% c(as.Date("2023-01-01"), as.Date("2023-02-01"), as.Date("2023-03-01"),
                   as.Date("2023-11-01"), as.Date("2023-12-01")) ~ "Cold Season",  # Jan-Mar & N
ov-Dec
      TRUE ~ NA_character_
    )
  ) %>%
  group_by(range_label) %>%
  summarise(avg = mean(monthly_ratio_avg, na.rm = TRUE), .groups = "drop")

# Extract specific averages into separate objects
april_oct_avg <- selected_avg %>% filter(range_label == "Bike Season") %>% pull(avg)
jan_mar_nov_dec_avg <- selected_avg %>% filter(range_label == "Cold Season") %>% pull(avg)

# View averages
april_oct_avg
```

```
## [1] 0.6096428
```

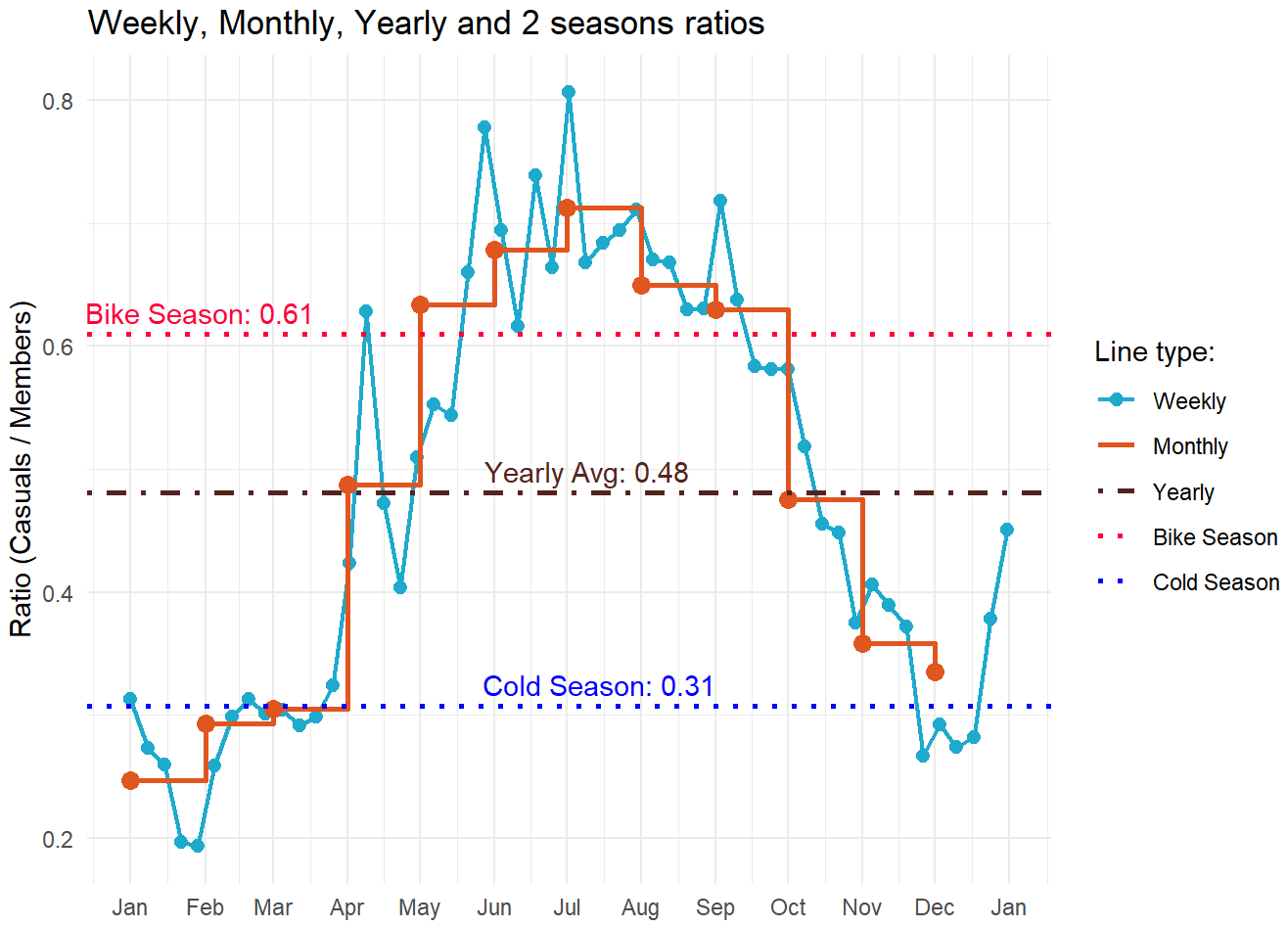
```
jan_mar_nov_dec_avg
```

```
## [1] 0.3077044
```

Plot for Weekly, Monthly, Yearly and 2 seasons ratios

```
ggplot(weekly_counts, aes(x = week, y = ratio, color = "Weekly")) +
  # Weekly ratio line
  geom_line(linewidth = 0.8) +
  geom_point(size = 2.2) +

  # Monthly average line
  geom_step(data = monthly_avg, aes(x = month, y = monthly_ratio_avg, color = "Monthly"),
            linewidth = 1) +
  geom_point(data = monthly_avg, aes(x = month, y = monthly_ratio_avg),
             color = "#e25822", size = 3) +  # Markers for monthly averages

  # Yearly average line
  geom_hline(aes(yintercept = yearly_avg, color = "Yearly"),
             linewidth = 1, linetype = "dotdash") +

  # Selected monthly averages
  geom_hline(data = selected_avg, aes(yintercept = avg, color = range_label),
             linetype = "dotted", linewidth = 1) +

  # Add annotations for averages
  annotate("text", x = as.Date("2023-12-01"), y = yearly_avg,
           label = paste0("Yearly Avg: ", round(yearly_avg, 2)),
           hjust = 2.2, vjust = -0.5, color = "#592720", fontface = "plain") +
  annotate("text", x = as.Date("2023-07-01"), y = april_oct_avg,
           label = paste0("Bike Season: ", round(april_oct_avg, 2)),
           hjust = 2.1, vjust = -0.5, color = "#ff033e", fontface = "plain") +
  annotate("text", x = as.Date("2023-03-01"), y = jan_mar_nov_dec_avg,
           label = paste0("Cold Season: ", round(jan_mar_nov_dec_avg, 2)),
           hjust = -0.9, vjust = -0.5, color = "#0000ff", fontface = "plain") +

  # Customize labels and legend #50c878
  labs(
    title = "Weekly, Monthly, Yearly and 2 seasons ratios",
        x = NULL,
        y = "Ratio (Casuals / Members)",
    color = "Line type:"
  ) +
  scale_color_manual(
    values = c(
      "Weekly" = "#21abcd",
      "Monthly" = "#e25822",
      "Yearly" = "#592720",
      "Bike Season" = "#ff033e",
      "Cold Season" = "#0000ff"
    ),
    breaks = c("Weekly", "Monthly", "Yearly", "Bike Season", "Cold Season"),
    labels = c("Weekly", "Monthly", "Yearly", "Bike Season", "Cold Season")
  ) +
  scale_x_date(
    date_labels = "%b",
    date_breaks = "1 month",
    limits = c(as.Date("2023-01-01"), as.Date("2023-12-31"))
```

```
  ) +
  theme_minimal()
```



Create a monthly1_counts dataset with the casual-to-member ratio for each month (Weighted yearly avr. based on month avr.):

```
monthly1_counts <- data2023 %>%
  mutate(month = floor_date(started_at, unit = "month")) %>%  # Create 'month' column
  group_by(month, member_casual) %>%  # Group by month and rider type
  summarise(count = n(), .groups = "drop") %>%  # Count rides for each group
  pivot_wider(names_from = member_casual, values_from = count, values_fill = 0) %>%  # Pivot to
wide format
  mutate(ratio = casual / member)  # Calculate the ratio for each month
```

Weighted (based on month average) yearly average from monthly1_counts (not used for chart)

```
yearly_avg_months <- mean(monthly1_counts$ratio, na.rm = TRUE)
# View averages
yearly_avg_months
```

```
## [1] 0.4836652
```

Print averages (Numeric value ‘unweighted_yearly_ratio’ is from the Pie chart aggregation, Plot 12)

```
# Create a data frame with renamed variables
averages_table <- data.frame(
  Category = c("Bike Season (April-Oct Avg)",
               "Cold Season (Jan-Mar & Nov-Dec Avg)",
               "Weighted Yearly (Weeks)",
               "Weighted Yearly (Months)",
               "Unweighted Yearly Ratio"),
  Average = c(april_oct_avg, jan_mar_nov_dec_avg, yearly_avg, yearly_avg_months, unweighted_year
ly_ratio)
)

# Print the table
print(averages_table)
```

```
##                              Category   Average
## 1         Bike Season (April-Oct Avg) 0.6096428
## 2 Cold Season (Jan-Mar & Nov-Dec Avg) 0.3077044
## 3             Weighted Yearly (Weeks) 0.4807990
## 4            Weighted Yearly (Months) 0.4836652
## 5             Unweighted Yearly Ratio 0.5436401
```

Retrieve the numeric values for Casuals only: 1) Number of total trips for each season; 2) Ratio Bike/Cold seasons; 3) Difference of Bike & Cold seasons. (The data frame ‘tr_count_season’ is from aggregation for Plot 15)

```
# 1. Filter for Casuals only and split by season
casual_season_counts <- tr_count_season %>%
  filter(str_detect(type_of_users, "casual")) %>%
  group_by(season) %>%
  summarise(total_trips = sum(tr_count_season), .groups = "drop")

# 2. Retrieve the total trips for Bike Season and Cold Season
bike_season_trips <- casual_season_counts %>%
  filter(season == "Bike Season") %>%
  pull(total_trips)

cold_season_trips <- casual_season_counts %>%
  filter(season == "Cold Season") %>%
  pull(total_trips)

# 3. Calculate the Ratio and Difference
season_ratio <- bike_season_trips / cold_season_trips  # Ratio Bike Season / Cold Season
season_difference <- bike_season_trips - cold_season_trips  # Difference Bike Season - Cold Seas
on

# Create a table of results
results_table <- tibble(
  Metric = c("Bike Season Trips", "Cold Season Trips", "Ratio (Bike/Cold)", "Difference (Bike -
Cold)"),
  Value = c(bike_season_trips, cold_season_trips, season_ratio, season_difference)
)

# Print the table
print(results_table)
```

```
## # A tibble: 4 × 2
##   Metric                        Value
##   <chr>                         <dbl>
## 1 Bike Season Trips        1619380
## 2 Cold Season Trips         275607
## 3 Ratio (Bike/Cold)              5.88
## 4 Difference (Bike - Cold) 1343773
```