Bikeshare Analysis

Brandon Mathews

2024-06-12

#Combining Data Sets

Since all my files have similar names,using this expression pattern.

##1)Create an empty list

##2)Create a for loop and iterate through all 12 files and add to list

##3)Combine all data frames into a single data frame thus combining it all.

##4)This is where I declared the path for the new csv file.

##5)Created new csv with combined data from last 12 months.

This now gives me all my data in one dataset called “combined_dataset”.

#Cleaning the Data Make sure you are using the library tidyverse and dplyr

##1) Now we begin by removing any duplicates and Remove any empty rows/columns

```
combined_data <- read.csv("~/Capstone Bike-Share/Trip_Data_DEC_2023_to JUNE_2024/combined_datase
t.csv")
combined_data <- unique(combined_data)

combined_data <- na.omit(combined_data)
```

##2) Removing columns I don’t think are needed Such as: (Start_station_id, End_station_id, start_lat/lng,end_lat/lng and end_station_name) I could use long/lat to determine distance but I think trip_length will suffice for me.

```
dt_cleaned <- subset(combined_data, select=-c(start_station_id,
                                              end_station_id,
                                              start_lat,
                                              start_lng,
                                              end_lat,
                                              end_lng,
                                              end_station_name))
pathway <-"C:/Users/skarz/OneDrive/Documents/Capstone Bike-Share/Trip_Data_DEC_2023_to JUNE_202
4/"
output_file <- file.path(pathway, "cleaned_data.csv")
write.csv(dt_cleaned, file = output_file, row.names = FALSE)
```

**If your following along, you can clear your data window on Rstudio, we will grab the dataset again and it will help in case of any errors in our code we can retrieve the dataset again from this point called “cleaned_data.csv”

```
dt_cleaned <- read.csv("~/Capstone Bike-Share/Trip_Data_DEC_2023_to JUNE_2024/cleaned_data.csv")

#Now we will keep a tab of the dataset so we can track our changes.
View(dt_cleaned)
```

First let’s make sure our start and end columns are in the proper format

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

View(dt_cleaned)
```

Now we will create a column for just the Date, Time, Then Day of the Week

```
dt_cleaned$date <- as.Date(dt_cleaned$started_at)

# Create separate column for time
dt_cleaned$start_time <- format(dt_cleaned$started_at, format = "%H:%M:%S")
dt_cleaned$end_time <- format(dt_cleaned$ended_at, format = "%H:%M:%S")

# Grab Day of Week from 'started_at'
dt_cleaned$day_of_week <- weekdays(dt_cleaned$started_at)
```

##3) Calculate trip length in minutes

```
dt_cleaned$time_difference_minutes <- round(as.numeric(difftime(dt_cleaned$ended_at, dt_cleaned
$started_at, units = "mins")),2)
```

##4) Removing data based on invalid trip times (end_time is before start_time) if any

```
library(dplyr)
```

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

```
## The following objects are masked from 'package:data.table':
##
##     between, first, last
```

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

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

```
dt_cleaned <- dt_cleaned %>%
  filter(time_difference_minutes >= 5)
```

This Removed 1,222,938 amount of rows from the dataset ensuring all calculated trips will be valid and the analysis can be built off a solid foundation of actual trips.

##5) Filling empty Data - Add’s N/A to start_station_name if no data in that row for that column

```
dt_cleaned$start_station_name <- ifelse(dt_cleaned$start_station_name == "", "N/A", dt_cleaned$s
tart_station_name)
```

Note: I was contemplating removing rows without start_station but I felt for this analysis I was going to focus on member vs non-member so I will leave it in.

If it needs to me removed here is the following code to do so: dt_cleaned <- dt_Cleaned %>% filter(!is.na(start_station_name))

##6) Saving new dataset to the “combined_dataset.csv” file

```
write.csv(dt_cleaned, "C:/Users/skarz/OneDrive/Documents/Capstone Bike-Share/Trip_Data_DEC_2023_
to JUNE_2024/finished_dataset.csv", row.names=FALSE)
```

##7) DOUBLE CHECK YOUR WORK Now from here, we double check our dataset and make sure we have all relevant information we need to do our visualizations.

```
View(dt_cleaned)
```

If you find a mistake then correct it before moving on. An example of this for is I would prefer my time_difference column to be named trip_length_min to make it understandable for any audience.

```
library(dplyr)
dt_cleaned <- dt_cleaned %>%
  rename(trip_length_min = time_difference_minutes)
```

Then save again

```
write.csv(dt_cleaned, "C:/Users/skarz/OneDrive/Documents/Capstone Bike-Share/Trip_Data_DEC_2023_
to JUNE_2024/finished_dataset.csv", row.names=FALSE)
```

#Breaking down data and creating Visualizations LOAD DATASET:

```
dt <- read.csv("~/Capstone Bike-Share/Trip_Data_DEC_2023_to JUNE_2024/finished_dataset.csv")
library(ggplot2)
library(dplyr)
View(dt)
```

Converting Date, Start and End Time to proper data types and create name variable for naming my visualizations

```
dt$date <- as.Date(dt$date, format = "%Y-%m-%d")
dt$start_time <- as.POSIXct(dt$start_time, format = "%H:%M:%S")
dt$end_time <- as.POSIXct(dt$end_time, format = "%H:%M:%S")
str(dt)
```

```
## 'data.frame':    4512656 obs. of  11 variables:
##  $ ride_id           : chr  "6F1682AC40EB6F71" "3C88859D926253B4" "CF682EA7D0F961DB" "E68F437
84662A2D0" ...
##  $ rideable_type     : chr  "electric_bike" "electric_bike" "electric_bike" "electric_bike"
...
##  $ started_at        : chr  "2023-06-05 13:34:12" "2023-06-20 18:15:49" "2023-06-09 21:30:25"
"2023-06-02 22:27:35" ...
##  $ ended_at          : chr  "2023-06-05 14:31:56" "2023-06-20 18:32:05" "2023-06-09 21:49:52"
"2023-06-02 22:35:26" ...
##  $ start_station_name: chr  "N/A" "N/A" "N/A" "N/A" ...
##  $ member_casual     : chr  "member" "member" "member" "member" ...
##  $ date              : Date, format: "2023-06-05" "2023-06-20" ...
##  $ start_time        : POSIXct, format: "2024-06-14 13:34:12" "2024-06-14 18:15:49" ...
##  $ end_time          : POSIXct, format: "2024-06-14 14:31:56" "2024-06-14 18:32:05" ...
##  $ day_of_week       : chr  "Monday" "Tuesday" "Friday" "Friday" ...
##  $ trip_length_min   : num  57.73 16.27 19.45 7.85 248.8 ...
```

```
name <- "Brandon Mathews"
```

###1) Total rides by day and by member:

```
library(dplyr)
rides_by_day_member <- dt %>%
  group_by(day_of_week, member_casual) %>%
  summarise(ride_count = n())
```

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

```
View(rides_by_day_member)

rides_by_day <-dt %>%
  group_by(day_of_week) %>%
  summarise(ride_count=n())
View(rides_by_day)
```

Once you confirm both tables are correct, go ahead and save them

```
write.csv(rides_by_day_member,"~/Capstone Bike-Share/Created_Datasets_for_Visualization/rides_by
_day_member.csv")

write.csv(rides_by_day,"~/Capstone Bike-Share/Created_Datasets_for_Visualization/rides_by_day.cs
v")
```

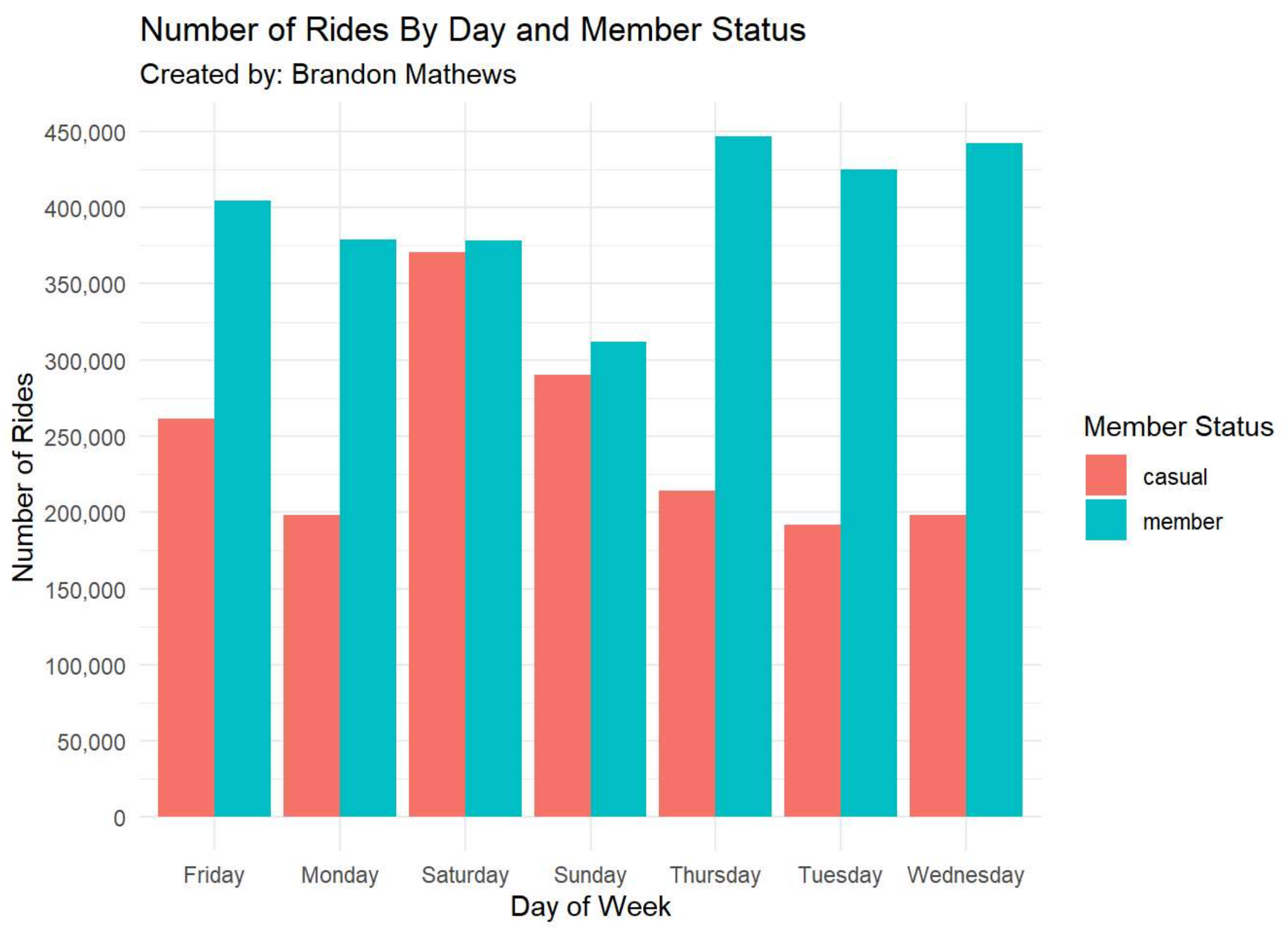
This gives me the summary for riders per day of the week for casual and members and then I use ggplot to visualize

For context this is how we find out how many rides were taken by non-members vs members:

```
rides_summary <- dt %>%
  group_by(member_casual) %>%
  summarize(total_rides = n())
View(rides_summary)
```

For context, My dataset at this point has: 1725537 non-members ("casual") & 2787119 members

```
library(ggplot2)
name <- "Brandon Mathews"
visual_plot <- ggplot(rides_by_day_member, aes(x=day_of_week, y=ride_count, fill=member_casual))
+
         geom_bar(stat="identity", position="dodge") +
         labs(title="Number of Rides By Day and Member Status",
              x="Day of Week",
              y="Number of Rides",
              fill="Member Status")+
         theme_minimal() +
  scale_y_continuous(labels = scales::comma, breaks = seq(0, 500000, by = 50000))

visual_plot + labs(subtitle=paste("Created by:", name))
```



###2) Total Rides per Month We will pull to create 2 new tables, both will showcase total rides per month and one in particular will go a bit further and showcase different types of bikes. We will convert some datatype to make sure we are staying on point.

```
dt$date <- as.Date(dt$date, format = "%Y-%m-%d")
dt$start_time <- format(dt$start_time, format = "%H:%M:%S")
dt$end_time <- format(dt$end_time, format = "%H:%M:%S")
```

```
dt <- dt %>%
  mutate(month = month(date),
         year = year(date))
dt <- dt %>%
  mutate(month_name = month.name[month])

monthly_totals <- dt %>%
  group_by(year, month, member_casual) %>%
  summarise(total_rides = n()) %>%
  mutate(month_name = month.name[month])
```

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

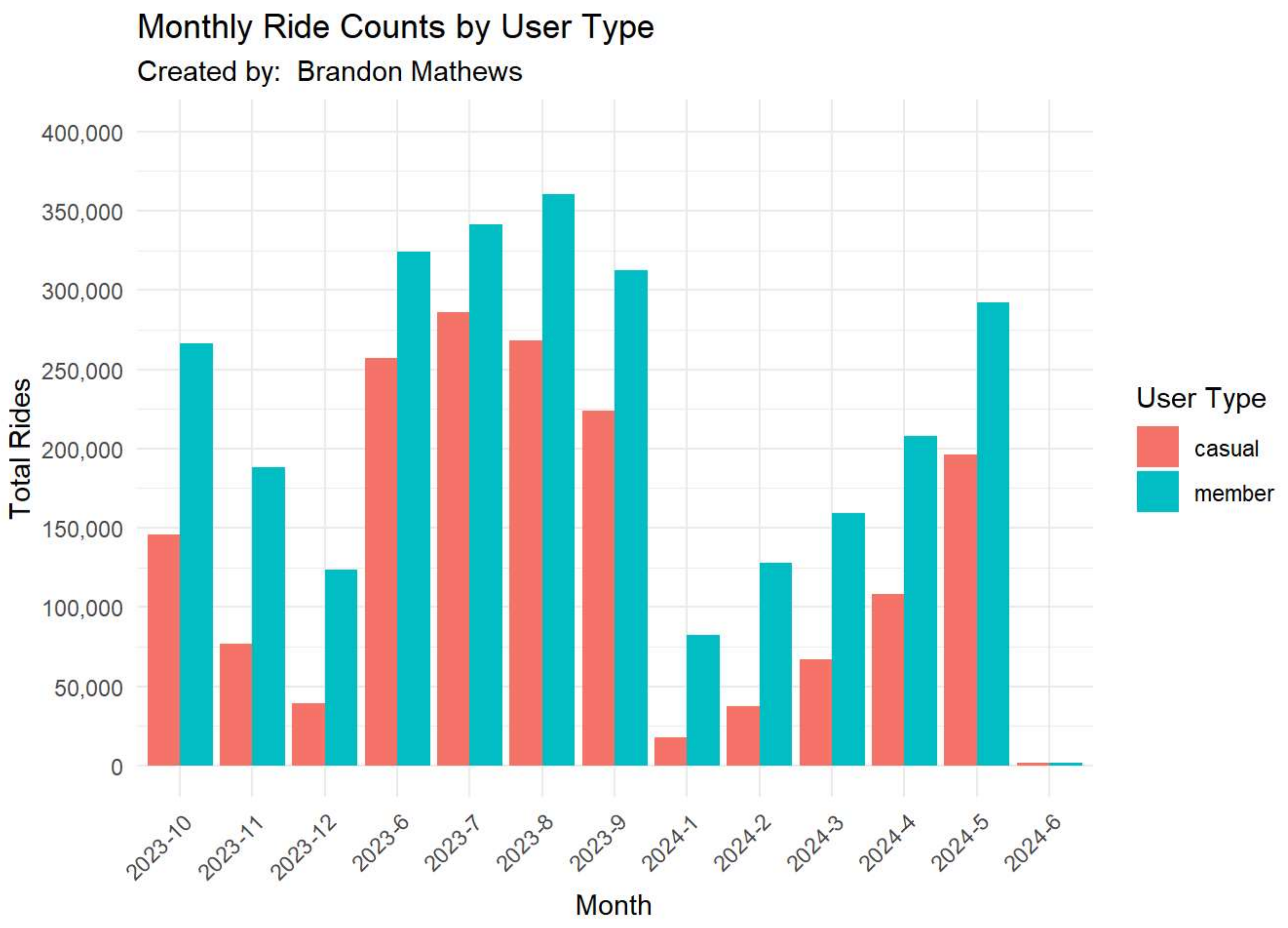
```
View(monthly_totals)



monthly_totals_bikes <- dt %>%
  group_by(year, month, member_casual, rideable_type) %>%
  summarise(total_rides = n()) %>%
  mutate(month_name = month.name[month])
```

```
## `summarise()` has grouped output by 'year', 'month', 'member_casual'. You can
## override using the `.groups` argument.
```

```
View(monthly_totals_bikes)



write.csv(monthly_totals,"~/Capstone Bike-Share/Created_Datasets_for_Visualization/monthly_total
s.csv")
write.csv(monthly_totals_bikes,"~/Capstone Bike-Share/Created_Datasets_for_Visualization/monthly
_totals_by_bikes_members.csv")
```

###3) Create Monthly Visualizations

```
monthly_totals$month_year <- paste(monthly_totals$year, monthly_totals$month, sep = "-")

ggplot(monthly_totals, aes(x = month_year, y = total_rides, fill = member_casual)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Monthly Ride Counts by User Type",
       x = "Month",
       y = "Total Rides",
       fill = "User Type") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +   scale_y_continuous(labels = scale
s::comma, limits = c(0, 400000), breaks = seq(0, 400000, by = 50000)) +
  labs(subtitle=paste("Created by: ",name))
```



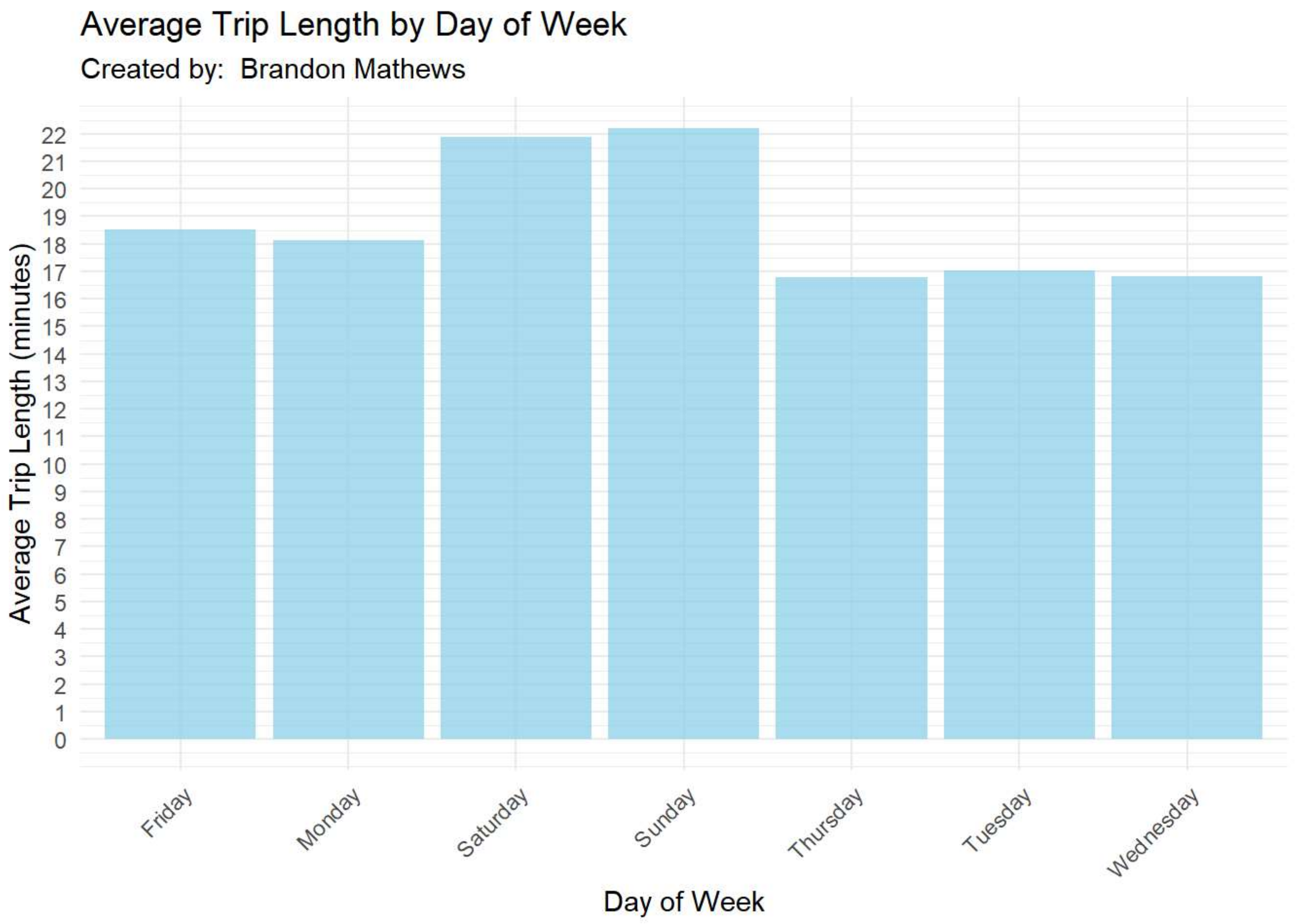
###4) DOUBLE CHECK YOUR VISUALIZATION SO FAR

###5) Now we need to load the month_dt and create dataset for trip_length on average by member/casual and then by day of week and then save our work

```
dt <- read.csv("~/Capstone Bike-Share/Trip_Data_DEC_2023_to JUNE_2024/finished_dataset.csv")

trip_length_avg_by_membership <- dt %>%
  group_by(member_casual) %>%
  summarise(avg_trip_length = round(mean(trip_length_min, na.rm = TRUE), 2))

trip_length_avg_by_day <- dt %>%
  group_by(day_of_week) %>%
  summarise(avg_trip_length = round(mean(trip_length_min, na.rm = TRUE), 2))

View(trip_length_avg_by_day)
View(trip_length_avg_by_membership)

write.csv(trip_length_avg_by_membership, file="~/Capstone Bike-Share/Created_Datasets_for_Visual
ization/trip_length_avg_by_membership.csv",row.names=FALSE)

write.csv(trip_length_avg_by_day, file="~/Capstone Bike-Share/Created_Datasets_for_Visualizatio
n/trip_length_avg_by_day.csv",row.names=FALSE)
```

VISUALIZE IT

```
name <-"Brandon Mathews"
ggplot(trip_length_avg_by_day, aes(x = day_of_week, y = avg_trip_length)) +
  geom_bar(stat = "identity", fill = "skyblue", alpha = 0.7) +
  labs(title = "Average Trip Length by Day of Week",
       x = "Day of Week",
       y = "Average Trip Length (minutes)") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
    scale_y_continuous(breaks = seq(0, max(trip_length_avg_by_day$avg_trip_length), by = 1))+
  labs(subtitle=paste("Created by: ", name))
```



###6) CREATING DATASET FOR TOTAL RIDES PER SEASON **For this one it was tricky so I took an approach I would normally do in like Python and with the help of documentation found this method to use a vector to create a list where numbers == months and then run it through a function that depending on the number it will return a char for the season in a new column.

```
month_dt <- read.csv("~/Capstone Bike-Share/Created_Datasets_for_Visualization/monthly_totals.cs
v")
```

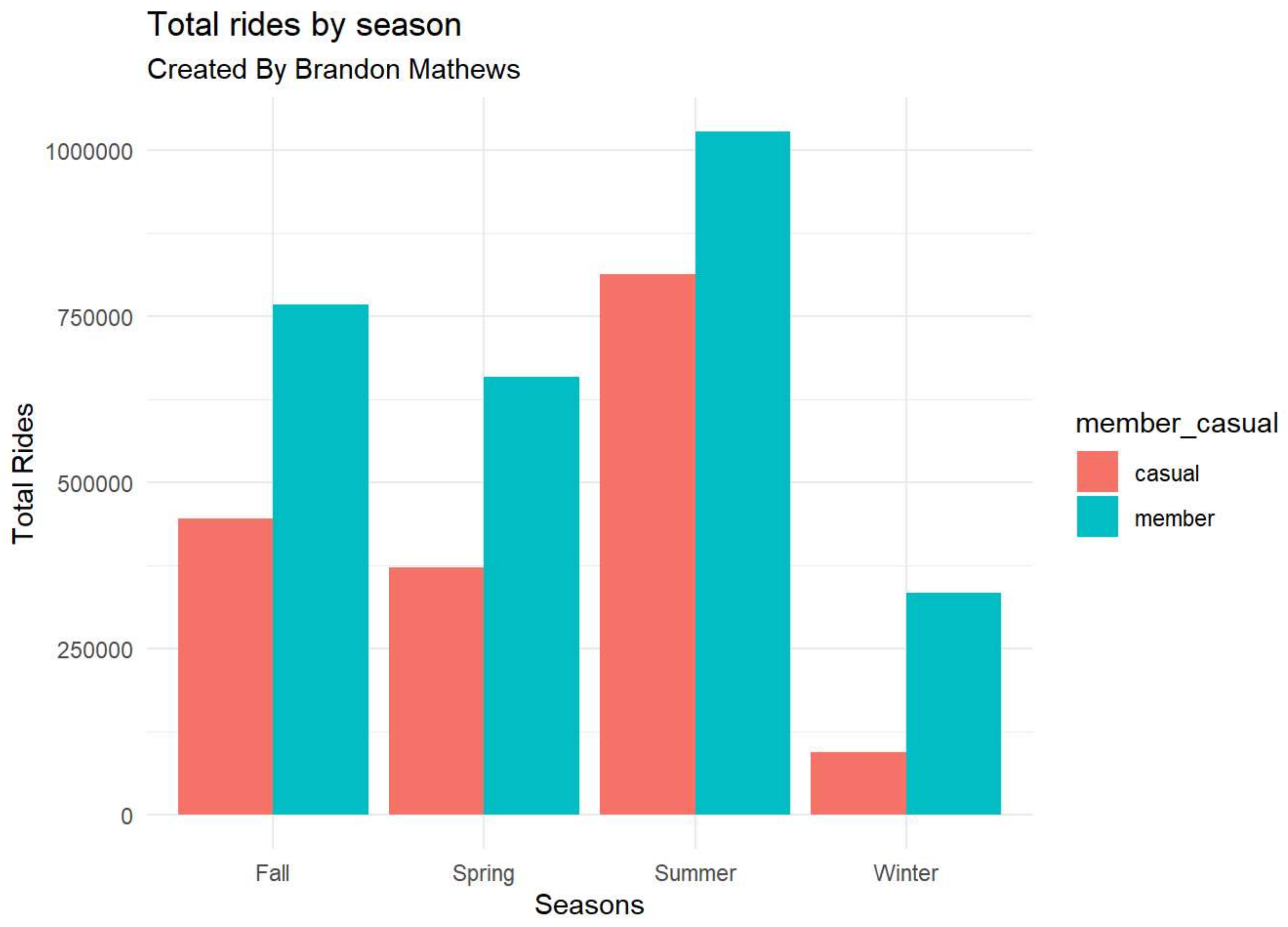
```
library(ggplot2)
library(dplyr)

get_season <- function(month) {
  if (month %in% c(12, 1, 2)) {
    return("Winter")
  } else if (month %in% c(3, 4, 5)) {
    return("Spring")
  } else if (month %in% c(6, 7, 8)) {
    return("Summer")
  } else if (month %in% c(9, 10, 11)) {
    return("Fall")
  } else {
    return("Unknown")  # Handle unexpected values
  }
}

# Apply the function to create a new column 'season'
month_dt$season <- sapply(month_dt$month, get_season)



seasonal_rides <- month_dt %>%
  group_by(season, member_casual) %>%
  summarise(total_rides = sum(total_rides))
```

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

```
write.csv(seasonal_rides, file="~/Capstone Bike-Share/Created_Datasets_for_Visualization/total_r
ides_by_season.csv",row.names=FALSE)
write.csv(month_dt,file="~/Capstone Bike-Share/Created_Datasets_for_Visualization/monthly_total
s.csv")
```

VISUALIZE IT

```
ggplot(seasonal_rides, aes(x= season, y=total_rides,fill=member_casual))+
  geom_bar(stat="identity",position="dodge") +
  labs(x="Seasons",y="Total Rides", title="Total rides by season", subtitle="Created By Brandon
Mathews") +
  theme_minimal()
```

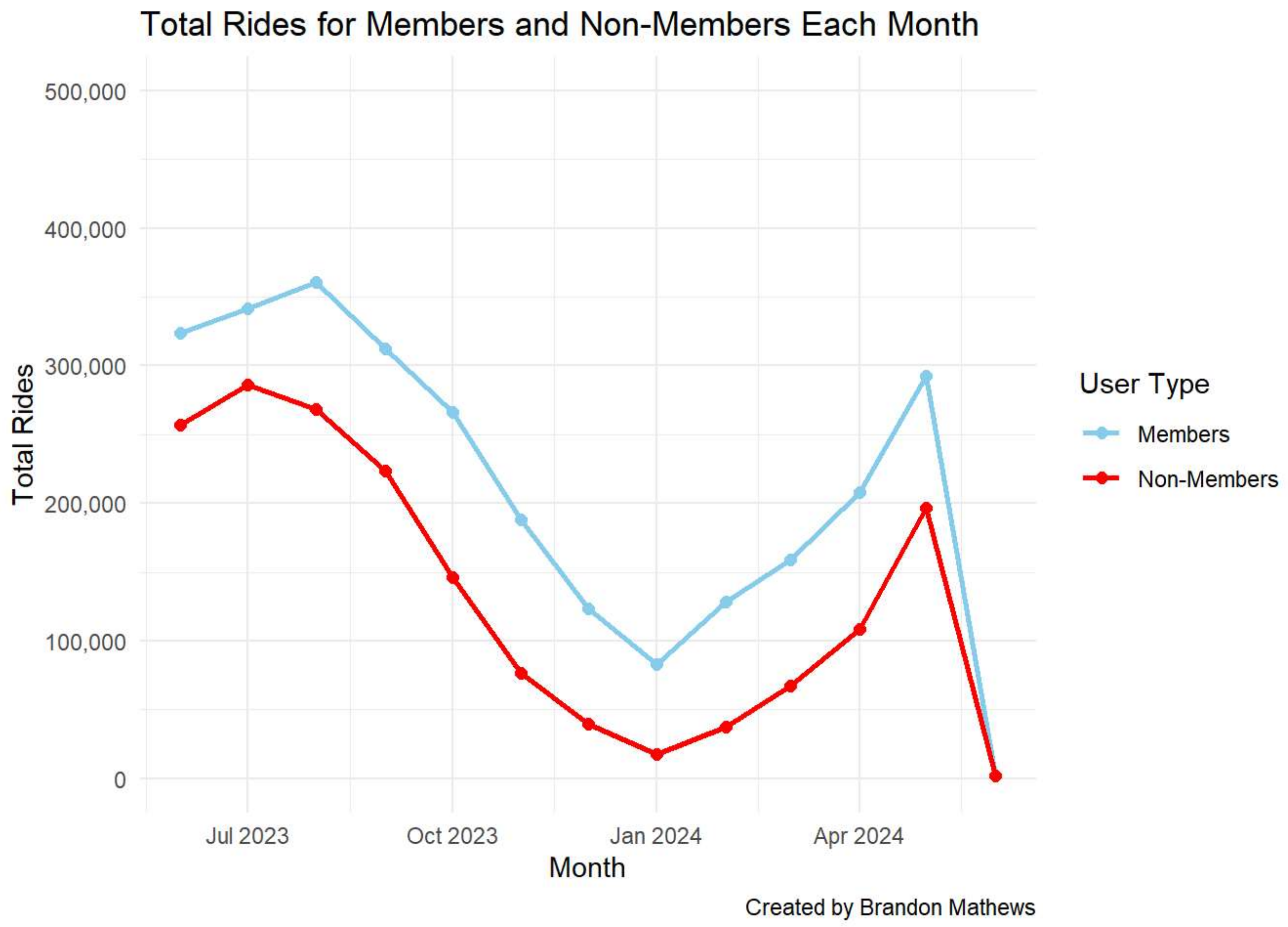


##7) Now we will create a line plot to show member and casual riders total rides over months

```
dt<- read.csv("~/Capstone Bike-Share/Created_Datasets_for_Visualization/monthly_totals.csv")

View(dt)
```

```
dt$month_date <- as.Date(paste(dt$year, dt$month, "01", sep = "-"))

# Sort the dataframe by month_date
dt <- dt[order(dt$month_date), ]

# Filter data for members and non-members
dt_members <- dt[dt$member_casual == "member", ]
dt_non_members <- dt[dt$member_casual == "casual", ]

# Load required libraries
library(ggplot2)
library(scales)  # For formatting numeric scales

# Create the line plot
ggplot() +
  geom_line(data = dt_members, aes(x = month_date, y = total_rides, color = "Members"), size =
1, linetype = "solid") +
  geom_point(data = dt_members, aes(x = month_date, y = total_rides, color = "Members"), size =
2) +
  geom_line(data = dt_non_members, aes(x = month_date, y = total_rides, color = "Non-Members"),
size = 1, linetype = "solid") +
  geom_point(data = dt_non_members, aes(x = month_date, y = total_rides, color = "Non-Members"),
size = 2) +
  scale_y_continuous(limits = c(0, 500000), breaks = seq(0, 500000, by = 100000),
                     labels = comma_format()) +
  labs(title = "Total Rides for Members and Non-Members Each Month",
       x = "Month", y = "Total Rides", color = "User Type", linetype = "User Type") +
  scale_color_manual(values = c("Members" = "skyblue", "Non-Members" = "red"),
                     labels = c("Members", "Non-Members")) +
  theme_minimal() +
  labs(caption = "Created by Brandon Mathews")
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



I will post additional visualizations on Tableau.

#Citation Lyft Bikes and Scooters, LLC. (2024). Divvy bicycle sharing system data [Data set]. City of Chicago. Retrieved from https://www.divvybikes.com/data (https://www.divvybikes.com/data)

Note: Use of this data is subject to the terms and conditions outlined in the Divvy License Agreement. Users must comply with all specified conditions including but not limited to non-commercial use and prohibition against using data mining or other extraction methods without authorization. Full agreement available at https://www.divvybikes.com/data (https://www.divvybikes.com/data).