Cyclistic Bike-Share Analysis: Understanding Rider Behavior

Shreyash S Sahare

2025-02-20

Introduction

The report describes the steps of data cleaning, preparation, and processing toward an analysis of Cyclistic bike-share trip data. As a junior data analyst on the marketing analysis team at Cyclistic, I am tasked with understanding how casual riders and annual members use Cyclistic bikes differently. This analysis will provide valuable insights into a new marketing strategy designed to convert casual riders into annual members so as to maximize the company's annual membership base.

To the marketing director, Cyclistic's future success rests significantly on increasing annual memberships. Thus, it becomes imperative to supply Cyclistic executives with data-backed insights and professional data visualizations to approve our recommendations.

The central question driving this analysis is: **How do annual members and casual riders utilize Cyclistic bikes differently?**

The data for used for this analysis is available from divvy-tripdata. The dataset includes trip records for the year 2024, thus giving an all-inclusive view of the working of rider behavior.

The following sections expound on the steps involved in:

- **Loading and inspecting the raw data:** To understand the original structure and the original contents of the dataset.
- **Preparation of the data by cleansing:** This includes dealing with missing values, correcting data types, and creating useful features.
- **Perform explorative data analysis:** Detect patterns and trends with rider behavior.
- **Make some visualizations:** Share main insights effectively.

Loading and inspecting the raw data:

Loading necessary libraries:

```
library(tidyverse) # For data manipulation and visualization
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
```

```
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(skimr)   # For descriptive statistics
library(janitor)  # For cleaning column names
```

```
##
## Attaching package: 'janitor'
##
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
library(dplyr)    # For data manipulation (included in tidyverse, but good to call explicitly)
library(ggplot2) # For generating visuals
```

Load trip data CSV files for each month of 2024

We begin by loading the trip data for each month of 2024 as separate CSV files. This approach allows for individual file inspection if needed for troubleshooting or data validation.

```
td_202401 <- read.csv("202401-divvy-tripdata.csv")
td_202402 <- read.csv("202402-divvy-tripdata.csv")
td_202403<- read.csv("202403-divvy-tripdata.csv")
td_202404<- read.csv("202404-divvy-tripdata.csv")
td_202405<- read.csv("202405-divvy-tripdata.csv")
td_202406<- read.csv("202406-divvy-tripdata.csv")
td_202407<- read.csv("202407-divvy-tripdata.csv")
td_202408<- read.csv("202408-divvy-tripdata.csv")
td_202409<- read.csv("202409-divvy-tripdata.csv")
td_202410<- read.csv("202410-divvy-tripdata.csv")
td_202411<- read.csv("202411-divvy-tripdata.csv")
td_202412 <- read.csv("202412-divvy-tripdata.csv")
```

Inspect the structure of each monthly data frame

Before combining the monthly data, it's essential to understand the structure of each individual data frame. We use the `str()` function to display the column names, data types, and a preview of the data. This step is crucial for identifying any inconsistencies or issues across the monthly files.

```
str(td_202401)
```

```
## 'data.frame':    144873 obs. of  13 variables:
##  $ ride_id           : chr  "C1D650626C8C899A" "EECD38BDB25BFCB0" "F4A9CE78061F17F7" "0A0D9E15EE50B1
##  $ rideable_type     : chr  "electric_bike" "electric_bike" "electric_bike" "classic_bike" ...
##  $ started_at        : chr  "2024-01-12 15:30:27" "2024-01-08 15:45:46" "2024-01-27 12:27:19" "2024-
##  $ ended_at          : chr  "2024-01-12 15:37:59" "2024-01-08 15:52:59" "2024-01-27 12:35:19" "2024-
##  $ start_station_name: chr  "Wells St & Elm St" "Wells St & Elm St" "Wells St & Elm St" "Wells St & E
##  $ start_station_id  : chr  "KA1504000135" "KA1504000135" "KA1504000135" "TA1305000030" ...
##  $ end_station_name  : chr  "Kingsbury St & Kinzie St" "Kingsbury St & Kinzie St" "Kingsbury St & Kin
```

```
##  $ end_station_id    : chr  "KA1503000043" "KA1503000043" "KA1503000043" "13193" ...
##  $ start_lat         : num  41.9 41.9 41.9 41.9 41.9 ...
##  $ start_lng         : num  -87.6 -87.6 -87.6 -87.6 -87.7 ...
##  $ end_lat           : num  41.9 41.9 41.9 41.9 41.9 ...
##  $ end_lng           : num  -87.6 -87.6 -87.6 -87.6 -87.6 ...
##  $ member_casual     : chr  "member" "member" "member" "member" ...
```

```
str(td_202402)
```

```
## 'data.frame':    223164 obs. of  13 variables:
##  $ ride_id           : chr  "FCB05EB1758F85E8" "7FB986AD5D3DE9D6" "40CA13E15B5B470D" "D47A1660919E88
##  $ rideable_type     : chr  "classic_bike" "classic_bike" "electric_bike" "classic_bike" ...
##  $ started_at        : chr  "2024-02-03 14:14:18" "2024-02-05 21:10:06" "2024-02-05 15:10:44" "2024-
##  $ ended_at          : chr  "2024-02-03 14:21:00" "2024-02-05 21:15:44" "2024-02-05 15:12:32" "2024-
##  $ start_station_name: chr  "Clark St & Newport St" "Michigan Ave & Washington St" "Leavitt St & Arm
##  $ start_station_id  : chr  "632" "13001" "TA1309000029" "13235" ...
##  $ end_station_name  : chr  "Southport Ave & Waveland Ave" "Wabash Ave & Grand Ave" "Milwaukee Ave &
##  $ end_station_id    : chr  "13235" "TA1307000117" "13243" "13229" ...
##  $ start_lat         : num  41.9 41.9 41.9 41.9 41.8 ...
##  $ start_lng         : num  -87.7 -87.6 -87.7 -87.7 -87.6 ...
##  $ end_lat           : num  41.9 41.9 41.9 41.9 41.8 ...
##  $ end_lng           : num  -87.7 -87.6 -87.7 -87.7 -87.6 ...
##  $ member_casual     : chr  "member" "member" "member" "member" ...
```

```
str(td_202403)
```

```
## 'data.frame':    301687 obs. of  13 variables:
##  $ ride_id           : chr  "64FBE3BAED5F29E6" "9991629435C5E20E" "E5C9FECD5B71BEBD" "4CEA3EC8906DAE
##  $ rideable_type     : chr  "electric_bike" "electric_bike" "electric_bike" "electric_bike" ...
##  $ started_at        : chr  "2024-03-05 18:33:11" "2024-03-06 17:15:14" "2024-03-06 17:16:36" "2024-
##  $ ended_at          : chr  "2024-03-05 18:51:48" "2024-03-06 17:16:04" "2024-03-06 17:19:28" "2024-
##  $ start_station_name: chr  "" "" "" "" ...
##  $ start_station_id  : chr  "" "" "" "" ...
##  $ end_station_name  : chr  "" "" "" "" ...
##  $ end_station_id    : chr  "" "" "" "" ...
##  $ start_lat         : num  41.9 41.9 41.9 41.9 41.9 ...
##  $ start_lng         : num  -87.7 -87.6 -87.6 -87.6 -87.7 ...
##  $ end_lat           : num  42 41.9 41.9 41.9 41.9 ...
##  $ end_lng           : num  -87.7 -87.6 -87.6 -87.6 -87.7 ...
##  $ member_casual     : chr  "member" "member" "member" "member" ...
```

```
str(td_202404)
```

```
## 'data.frame':    415025 obs. of  13 variables:
##  $ ride_id           : chr  "743252713F32516B" "BE90D33D2240C614" "D47BBDDE7C40DD61" "6684E760BF9EA9
##  $ rideable_type     : chr  "classic_bike" "electric_bike" "classic_bike" "classic_bike" ...
##  $ started_at        : chr  "2024-04-22 19:08:21" "2024-04-11 06:19:24" "2024-04-20 11:13:13" "2024-
##  $ ended_at          : chr  "2024-04-22 19:12:56" "2024-04-11 06:22:21" "2024-04-20 11:29:31" "2024-
##  $ start_station_name: chr  "Aberdeen St & Jackson Blvd" "Aberdeen St & Jackson Blvd" "Sheridan Rd &
##  $ start_station_id  : chr  "13157" "13157" "TA1307000107" "13157" ...
##  $ end_station_name  : chr  "Desplaines St & Jackson Blvd" "Desplaines St & Jackson Blvd" "Ashland Av
##  $ end_station_id    : chr  "15539" "15539" "13249" "15539" ...
```

```
##  $ start_lat         : num  41.9 41.9 42 41.9 42 ...
##  $ start_lng         : num  -87.7 -87.7 -87.7 -87.7 -87.7 ...
##  $ end_lat           : num  41.9 41.9 42 41.9 41.9 ...
##  $ end_lng           : num  -87.6 -87.6 -87.7 -87.6 -87.6 ...
##  $ member_casual     : chr  "member" "member" "member" "member" ...
```

```
str(td_202405)
```

```
## 'data.frame':    609493 obs. of  13 variables:
##  $ ride_id           : chr  "7D9F0CE9EC2A1297" "02EC47687411416F" "101370FB2D3402BE" "E97E396331ED69
##  $ rideable_type     : chr  "classic_bike" "classic_bike" "classic_bike" "electric_bike" ...
##  $ started_at        : chr  "2024-05-25 15:52:42" "2024-05-14 15:11:51" "2024-05-30 17:46:04" "2024-0
##  $ ended_at          : chr  "2024-05-25 16:11:50" "2024-05-14 15:22:00" "2024-05-30 18:09:16" "2024-0
##  $ start_station_name: chr  "Streeter Dr & Grand Ave" "Sheridan Rd & Greenleaf Ave" "Streeter Dr & Gr
##  $ start_station_id  : chr  "13022" "KA1504000159" "13022" "13022" ...
##  $ end_station_name  : chr  "Clark St & Elm St" "Sheridan Rd & Loyola Ave" "Wabash Ave & 9th St" "Sho
##  $ end_station_id    : chr  "TA1307000039" "RP-009" "TA1309000010" "TA1307000052" ...
##  $ start_lat         : num  41.9 42 41.9 41.9 41.9 ...
##  $ start_lng         : num  -87.6 -87.7 -87.6 -87.6 -87.6 ...
##  $ end_lat           : num  41.9 42 41.9 41.9 41.9 ...
##  $ end_lng           : num  -87.6 -87.7 -87.6 -87.7 -87.6 ...
##  $ member_casual     : chr  "casual" "casual" "member" "member" ...
```

```
str(td_202406)
```

```
## 'data.frame':    710721 obs. of  13 variables:
##  $ ride_id           : chr  "CDE6023BE6B11D2F" "462B48CD292B6A18" "9CFB6A858D23ABF7" "6365EFEB642311
##  $ rideable_type     : chr  "electric_bike" "electric_bike" "electric_bike" "electric_bike" ...
##  $ started_at        : chr  "2024-06-11 17:20:06.289" "2024-06-11 17:19:21.567" "2024-06-11 17:25:27
##  $ ended_at          : chr  "2024-06-11 17:21:39.464" "2024-06-11 17:19:36.377" "2024-06-11 17:30:13
##  $ start_station_name: chr  "" "" "" "" ...
##  $ start_station_id  : chr  "" "" "" "" ...
##  $ end_station_name  : chr  "" "" "" "" ...
##  $ end_station_id    : chr  "" "" "" "" ...
##  $ start_lat         : num  41.9 41.9 41.9 41.9 41.9 ...
##  $ start_lng         : num  -87.7 -87.7 -87.7 -87.6 -87.6 ...
##  $ end_lat           : num  41.9 41.9 41.9 41.9 41.9 ...
##  $ end_lng           : num  -87.7 -87.7 -87.7 -87.6 -87.6 ...
##  $ member_casual     : chr  "casual" "casual" "casual" "casual" ...
```

```
str(td_202407)
```

```
## 'data.frame':    748962 obs. of  13 variables:
##  $ ride_id           : chr  "2658E319B13141F9" "B2176315168A47CE" "C2A9D33DF7EBB422" "8BFEA406DF01D8
##  $ rideable_type     : chr  "electric_bike" "electric_bike" "electric_bike" "electric_bike" ...
##  $ started_at        : chr  "2024-07-11 08:15:14.784" "2024-07-11 15:45:07.851" "2024-07-11 08:24:48
##  $ ended_at          : chr  "2024-07-11 08:17:56.335" "2024-07-11 16:06:04.243" "2024-07-11 08:28:05
##  $ start_station_name: chr  "" "" "" "" ...
##  $ start_station_id  : chr  "" "" "" "" ...
##  $ end_station_name  : chr  "" "" "" "" ...
##  $ end_station_id    : chr  "" "" "" "" ...
##  $ start_lat         : num  41.8 41.8 41.8 41.9 42 ...
```

```
##  $ start_lng         : num  -87.6 -87.6 -87.6 -87.6 -87.6 ...
##  $ end_lat           : num  41.8 41.8 41.8 41.9 41.9 ...
##  $ end_lng           : num  -87.6 -87.6 -87.6 -87.7 -87.6 ...
##  $ member_casual     : chr  "casual" "casual" "casual" "casual" ...
```

```
str(td_202408)
```

```
## 'data.frame':    755639 obs. of  13 variables:
##  $ ride_id           : chr  "BAA154388A869E64" "8752245932EFF67A" "44DDF9F57A9A161F" "44AAAF069B0C78
##  $ rideable_type     : chr  "classic_bike" "electric_bike" "classic_bike" "electric_bike" ...
##  $ started_at        : chr  "2024-08-02 13:35:14.403" "2024-08-02 15:33:13.965" "2024-08-16 15:44:06
##  $ ended_at          : chr  "2024-08-02 13:48:24.426" "2024-08-02 15:55:23.865" "2024-08-16 15:57:52
##  $ start_station_name: chr  "State St & Randolph St" "Franklin St & Monroe St" "Franklin St & Monroe
##  $ start_station_id  : chr  "TA1305000029" "TA1309000007" "TA1309000007" "TA1307000039" ...
##  $ end_station_name  : chr  "Wabash Ave & 9th St" "Damen Ave & Cortland St" "Clark St & Elm St" "McC
##  $ end_station_id    : chr  "TA1309000010" "13133" "TA1307000039" "TA1306000029" ...
##  $ start_lat         : num  41.9 41.9 41.9 41.9 42 ...
##  $ start_lng         : num  -87.6 -87.6 -87.6 -87.6 -87.7 ...
##  $ end_lat           : num  41.9 41.9 41.9 41.9 42 ...
##  $ end_lng           : num  -87.6 -87.7 -87.6 -87.6 -87.7 ...
##  $ member_casual     : chr  "member" "member" "member" "member" ...
```

```
str(td_202409)
```

```
## 'data.frame':    821276 obs. of  13 variables:
##  $ ride_id           : chr  "31D38723D5A8665A" "67CB39987F4E895B" "DA61204FD26EC681" "06F160D46AF235
##  $ rideable_type     : chr  "electric_bike" "electric_bike" "electric_bike" "electric_bike" ...
##  $ started_at        : chr  "2024-09-26 15:30:58.150" "2024-09-26 15:31:32.529" "2024-09-26 15:00:33
##  $ ended_at          : chr  "2024-09-26 15:30:59.437" "2024-09-26 15:53:13.501" "2024-09-26 15:02:25
##  $ start_station_name: chr  "" "" "" "" ...
##  $ start_station_id  : chr  "" "" "" "" ...
##  $ end_station_name  : chr  "" "" "" "" ...
##  $ end_station_id    : chr  "" "" "" "" ...
##  $ start_lat         : num  41.9 41.9 41.9 41.9 41.9 ...
##  $ start_lng         : num  -87.6 -87.6 -87.6 -87.6 -87.7 ...
##  $ end_lat           : num  41.9 41.9 41.9 41.9 41.9 ...
##  $ end_lng           : num  -87.6 -87.6 -87.6 -87.6 -87.6 ...
##  $ member_casual     : chr  "member" "member" "member" "member" ...
```

```
str(td_202410)
```

```
## 'data.frame':    616281 obs. of  13 variables:
##  $ ride_id           : chr  "4422E707103AA4FF" "19DB722B44CBE82F" "20AE2509FD68C939" "D0F17580AB9515
##  $ rideable_type     : chr  "electric_bike" "electric_bike" "electric_bike" "electric_bike" ...
##  $ started_at        : chr  "2024-10-14 03:26:04.083" "2024-10-13 19:33:38.926" "2024-10-13 23:40:48
##  $ ended_at          : chr  "2024-10-14 03:32:56.535" "2024-10-13 19:39:04.490" "2024-10-13 23:48:02
##  $ start_station_name: chr  "" "" "" "" ...
##  $ start_station_id  : chr  "" "" "" "" ...
##  $ end_station_name  : chr  "" "" "" "" ...
##  $ end_station_id    : chr  "" "" "" "" ...
##  $ start_lat         : num  42 42 42 42 42 ...
##  $ start_lng         : num  -87.7 -87.7 -87.7 -87.7 -87.7 ...
```

```
##  $ end_lat           : num  42 42 42 42 42 ...
##  $ end_lng           : num  -87.7 -87.7 -87.7 -87.7 -87.7 ...
##  $ member_casual     : chr  "member" "member" "member" "member" ...
```

```
str(td_202411)
```

```
## 'data.frame':    335075 obs. of  13 variables:
##  $ ride_id           : chr  "578DDD7CE1771FFA" "78B141C50102ABA6" "1E794CF36394E2D7" "E5DD2CAB58D73F
##  $ rideable_type     : chr  "classic_bike" "classic_bike" "classic_bike" "classic_bike" ...
##  $ started_at        : chr  "2024-11-07 19:21:58.206" "2024-11-22 14:49:00.431" "2024-11-08 09:24:00
##  $ ended_at          : chr  "2024-11-07 19:28:57.301" "2024-11-22 14:56:15.475" "2024-11-08 09:28:33
##  $ start_station_name: chr  "Walsh Park" "Walsh Park" "Walsh Park" "Clark St & Elm St" ...
##  $ start_station_id  : chr  "18067" "18067" "18067" "TA1307000039" ...
##  $ end_station_name  : chr  "Leavitt St & North Ave" "Leavitt St & Armitage Ave" "Damen Ave & Cortla
##  $ end_station_id    : chr  "TA1308000005" "TA1309000029" "13133" "TA1307000142" ...
##  $ start_lat         : num  41.9 41.9 41.9 41.9 41.9 ...
##  $ start_lng         : num  -87.7 -87.7 -87.7 -87.6 -87.6 ...
##  $ end_lat           : num  41.9 41.9 41.9 41.9 41.9 ...
##  $ end_lng           : num  -87.7 -87.7 -87.7 -87.6 -87.6 ...
##  $ member_casual     : chr  "member" "member" "member" "member" ...
```

```
str(td_202412)
```

```
## 'data.frame':    178372 obs. of  13 variables:
##  $ ride_id           : chr  "6C960DEB4F78854E" "C0913EEB2834E7A2" "848A37DD4723078A" "3FA09C762ECB48
##  $ rideable_type     : chr  "electric_bike" "classic_bike" "classic_bike" "electric_bike" ...
##  $ started_at        : chr  "2024-12-31 01:38:35.018" "2024-12-21 18:41:26.478" "2024-12-21 11:41:01
##  $ ended_at          : chr  "2024-12-31 01:48:45.775" "2024-12-21 18:47:33.871" "2024-12-21 11:52:45
##  $ start_station_name: chr  "Halsted St & Roscoe St" "Clark St & Wellington Ave" "Sheridan Rd & Monti
##  $ start_station_id  : chr  "TA1309000025" "TA1307000136" "TA1307000107" "13157" ...
##  $ end_station_name  : chr  "Clark St & Winnemac Ave" "Halsted St & Roscoe St" "Broadway & Barry Ave
##  $ end_station_id    : chr  "TA1309000035" "TA1309000025" "13137" "chargingstx3" ...
##  $ start_lat         : num  41.9 41.9 42 41.9 41.9 ...
##  $ start_lng         : num  -87.6 -87.6 -87.7 -87.7 -87.7 ...
##  $ end_lat           : num  42 41.9 41.9 41.9 41.9 ...
##  $ end_lng           : num  -87.7 -87.6 -87.6 -87.6 -87.7 ...
##  $ member_casual     : chr  "member" "member" "member" "member" ...
```

Combine all monthly data frames into a single data frame

We use the `bind_rows()` function to efficiently combine all twelve monthly data frames into a single data frame called `full_year`. This creates a comprehensive dataset for our analysis.

```
full_year <- bind_rows(td_202401, td_202402, td_202403, td_202404, td_202405, td_202406, td_202407, td_
```

Check the column names of the combined data frame

After combining the data, we double-check the column names of the `full_year` data frame to ensure that all columns from the monthly files are present and consistent.

```r
colnames(full_year)
```

```
##  [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"
```

Inspect the structure of the combined data frame

We use `str()` again to inspect the structure of the combined `full_year` data frame. This allows us to verify the data types of each column and confirm that the data has been combined correctly.

```r
str(full_year)
```

```
## 'data.frame':    5860568 obs. of  13 variables:
##  $ ride_id           : chr  "C1D650626C8C899A" "EECD38BDB25BFCB0" "F4A9CE78061F17F7" "0A0D9E15EE50B1
##  $ rideable_type     : chr  "electric_bike" "electric_bike" "electric_bike" "classic_bike" ...
##  $ started_at        : chr  "2024-01-12 15:30:27" "2024-01-08 15:45:46" "2024-01-27 12:27:19" "2024-
##  $ ended_at          : chr  "2024-01-12 15:37:59" "2024-01-08 15:52:59" "2024-01-27 12:35:19" "2024-
##  $ start_station_name: chr  "Wells St & Elm St" "Wells St & Elm St" "Wells St & Elm St" "Wells St & E
##  $ start_station_id  : chr  "KA1504000135" "KA1504000135" "KA1504000135" "TA1305000030" ...
##  $ end_station_name  : chr  "Kingsbury St & Kinzie St" "Kingsbury St & Kinzie St" "Kingsbury St & Kin
##  $ end_station_id    : chr  "KA1503000043" "KA1503000043" "KA1503000043" "13193" ...
##  $ start_lat         : num  41.9 41.9 41.9 41.9 41.9 ...
##  $ start_lng         : num  -87.6 -87.6 -87.6 -87.6 -87.7 ...
##  $ end_lat           : num  41.9 41.9 41.9 41.9 41.9 ...
##  $ end_lng           : num  -87.6 -87.6 -87.6 -87.6 -87.6 ...
##  $ member_casual     : chr  "member" "member" "member" "member" ...
```

View the first few rows of the data frame (optional)

The `View()` function opens the data in a spreadsheet-like viewer, allowing for a quick visual inspection of the data. This is useful for getting a sense of the actual values and format of the data.

Select relevant columns and create a cleaned data frame:

For our analysis, we only need a subset of the available columns. This line uses `select()` to choose the necessary columns and create a new data frame called `full_year_cleaned3`. Creating a new object for the cleaned data is a good practice as it preserves the original `full_year` data.

```r
full_year_cleaned3 <- full_year %>%
  select(-c(start_station_name, start_station_id, end_station_name, end_station_id))
```

Check the column names of the cleaned data frame

We verify the column names of the cleaned data frame to confirm that only the desired columns remain.

```r
colnames(full_year_cleaned3)
```

```
## [1] "ride_id"       "rideable_type" "started_at"    "ended_at"
## [5] "start_lat"     "start_lng"     "end_lat"       "end_lng"
## [9] "member_casual"
```

Check the number of rows and dimensions of the cleaned data frame

We use `nrow()` and `dim()` to check the number of rows and dimensions (rows and columns) of the cleaned data frame. This helps us ensure that no rows were lost during the column selection process.

```r
nrow(full_year_cleaned3)
```

```
## [1] 5860568
```

```r
dim(full_year_cleaned3)
```

```
## [1] 5860568       9
```

View the first few rows of the cleaned data frame

We use `head()` to view the first few rows of the cleaned data frame. This provides a quick look at the data after cleaning.

```r
head(full_year_cleaned3)
```

```
##            ride_id rideable_type          started_at            ended_at
## 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
##   start_lat start_lng  end_lat   end_lng member_casual
## 1  41.90327 -87.63474 41.88918 -87.63851        member
## 2  41.90294 -87.63444 41.88918 -87.63851        member
## 3  41.90295 -87.63447 41.88918 -87.63851        member
## 4  41.88430 -87.63396 41.92182 -87.64414        member
## 5  41.94880 -87.67528 41.88918 -87.63851        member
## 6  41.90322 -87.63432 41.88918 -87.63851        member
```

Generate descriptive statistics for the cleaned data frame

We use the `summary()` function to generate descriptive statistics for all columns in the cleaned data frame. This includes measures like mean, median, min, max, and quartiles for numeric data, and frequency counts for categorical data. This gives us a general overview of the data distribution.

```r
summary(full_year_cleaned3)
```

```
##    ride_id          rideable_type       started_at          ended_at
##  Length:5860568     Length:5860568     Length:5860568     Length:5860568
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##    start_lat       start_lng         end_lat         end_lng
##  Min.   :41.64   Min.   :-87.91   Min.   :16.06   Min.   :-144.05
##  1st Qu.:41.88   1st Qu.:-87.66   1st Qu.:41.88   1st Qu.: -87.66
##  Median :41.90   Median :-87.64   Median :41.90   Median : -87.64
##  Mean   :41.90   Mean   :-87.65   Mean   :41.90   Mean   : -87.65
##  3rd Qu.:41.93   3rd Qu.:-87.63   3rd Qu.:41.93   3rd Qu.: -87.63
##  Max.   :42.07   Max.   :-87.52   Max.   :87.96   Max.   : 152.53
##                                   NA's   :7232    NA's   :7232
##  member_casual
##  Length:5860568
##  Class :character
##  Mode  :character
##
##
##
##
```

Preparation of the data by cleansing:

This section performs several crucial data cleaning and transformation steps.

```r
full_year_cleaned_3.1 <- full_year_cleaned3 %>%
  filter(!is.na(ended_at)) %>% # Remove rows with missing end times.  Missing end times would make ride
  filter(!is.na(started_at)) %>% # Remove rows with missing start times. Similar to end times, missing
  filter(ended_at >= started_at) %>% # Ensure end time is not before start time. This is a logical chec
  # Convert character columns to proper datetime objects. The `ended_at` and `started_at` columns are l
  mutate(ended_at = as.POSIXct(ended_at, format = "%Y-%m-%d %H:%M:%S"),
         started_at = as.POSIXct(started_at, format = "%Y-%m-%d %H:%M:%S")) %>%
  # Calculate ride length in minutes. We calculate the ride length by subtracting the start time from t
  mutate(ride_length = as.numeric(difftime(ended_at, started_at, units = "mins"))) %>%
  # Extract the day of the week. We extract the day of the week from the `started_at` column using the
  mutate(weekday = wday(started_at, label = TRUE)) %>%
  # Determine the season based on the start month. We create a new `season` column based on the month o
  mutate(season = case_when(
    between(month(started_at), 3, 5) ~ "Spring",
    between(month(started_at), 6, 8) ~ "Summer",
    between(month(started_at), 9, 11) ~ "Autum", # Corrected spelling here
    TRUE ~ "Winter")) %>%
  # Determine the part of the day.  We create a new `part_of_day` column categorizing rides into Mornin
  mutate(part_of_day = case_when(
    hour(started_at) >= 6 & hour(started_at) < 12 ~ "Morning",
    hour(started_at) >= 12 & hour(started_at) < 17 ~ "Afternoon",
```

```
    hour(started_at) >= 17 & hour(started_at) < 22 ~ "Evening",
    TRUE ~ "Night"))

View(full_year_cleaned_3.1)
```

Perform explorative data analysis:

Summary table:-

Now that we have a cleaned and transformed dataset, we start creating summary tables for our analysis.

```
summary_stats2.2 <- full_year_cleaned_3.1 %>% # Corrected object name
  summarise(mean_ride_length = mean(ride_length, na.rm = TRUE),
            max_ride_length = max(ride_length, na.rm = TRUE),
            min_ridelngth = min(ride_length, na.rm = TRUE),
            total_rides = n())

View(summary_stats2.2)
```

Comparing Tables:-

```
comparing_stats2.2 <- full_year_cleaned_3.1 %>% group_by(member_casual) %>%
  summarise(mean_ride_length = mean(ride_length,na.rm = TRUE),
            max_ride_length= max(ride_length, na.rm = TRUE),
            lower_ride_length = quantile(ride_length,0.25,na.rm =TRUE),
            upper_ride_length = quantile(ride_length,0.75,na.rm =TRUE),
            min_ridelngth = min(ride_length, na.rm = TRUE),
            total_rides= n()
  )
```

Comparing Summary based on Members:-

```
comparing_stats_WD <- full_year_cleaned_3.1 %>%filter(!is.na(weekday))%>%
  group_by(member_casual,weekday) %>%
  summarise(mean_ride_length = mean(ride_length,na.rm = TRUE),
            max_ride_length= max(ride_length, na.rm = TRUE),
            min_ridelngth = min(ride_length, na.rm = TRUE),
            total_rides= n()
  )
```

Now comparing casual and members stats based on weekdays:-

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

```
View(comparing_stats_WD)
```

```
comparing_stats_WD_PD<- full_year_cleaned_3.1 %>%filter(!is.na(weekday))%>%
  filter(!is.na(part_of_day))%>%
  group_by(member_casual,weekday, part_of_day) %>%
  summarise(mean_ride_length = mean(ride_length,na.rm = TRUE),
            max_ride_length= max(ride_length, na.rm = TRUE),
            min_ridelngth = min(ride_length, na.rm = TRUE),
            total_rides= n()
  )
```

Now comparing casual and members stats based on Part of the days

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

```
View(comparing_stats_WD_PD)
```

```
comparing_stats_S<- full_year_cleaned_3.1 %>%filter(!is.na(weekday))%>%
  filter(!is.na(part_of_day))%>%
  filter(!is.na(season))%>%
  group_by(member_casual, season,weekday,part_of_day) %>%
  summarise(mean_ride_length = mean(ride_length,na.rm = TRUE),
            max_ride_length= max(ride_length, na.rm = TRUE),
            min_ridelngth = min(ride_length, na.rm = TRUE),
            total_rides= n()
  )
```

Now comparing casual and members stats based on Season

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

```
View(comparing_stats_S)
```

```
comparing_stats_A<- full_year_cleaned_3.1 %>%filter(!is.na(weekday))%>%
  filter(!is.na(part_of_day))%>%
  filter(!is.na(season))%>%
  group_by(member_casual, season,weekday,part_of_day) %>%
  summarise(mean_ride_length = mean(ride_length,na.rm = TRUE),
            max_ride_length= max(ride_length, na.rm = TRUE),
            min_ridelngth = min(ride_length, na.rm = TRUE),
            total_rides= n()
  )
```

Comparing casual and members based on Weekdays season and part of the day

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

```
View(comparing_stats_A)
```

```
comparing_stats_AA<- full_year_cleaned_3.1 %>%filter(!is.na(weekday))%>%
  filter(!is.na(part_of_day))%>%
  filter(!is.na(season))%>%
  filter(!is.na(rideable_type))%>%
  group_by(member_casual, season,weekday,part_of_day,rideable_type) %>%
  summarise(mean_ride_length = mean(ride_length,na.rm = TRUE),
            max_ride_length= max(ride_length, na.rm = TRUE),
            min_ridelngth = min(ride_length, na.rm = TRUE),
            total_rides= n()
  )
```

Comparing casual and members based on weekdays , season, part of day and ride type.

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

```
View(comparing_stats_AA)
```

Top 50 location

```
S_location<- full_year_cleaned_3.1 %>%filter(member_casual == "member")%>%
  group_by(member_casual, start_lat,start_lng) %>%
  summarise(total_rides= n())%>%arrange(desc(total_rides))
```

Start Station, Member = Annual

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

```
Top_50_S_location<-head(S_location,50)
View(Top_50_S_location)
```

```
SC_location<- full_year_cleaned_3.1 %>%filter(!is.na(weekday))%>%
  filter(!is.na(part_of_day))%>%
  filter(!is.na(season))%>%
  filter(!is.na(rideable_type))%>%
  filter(member_casual == "casual")%>%
  group_by(member_casual, start_lat,start_lng) %>%
  summarise(total_rides= n())%>%arrange(desc(total_rides))
```

Start Station, Member = Casual

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

```
Top_50_SC_location<-head(SC_location,50)
View(Top_50_SC_location)
```

```
EM_location<- full_year_cleaned_3.1 %>%filter(!is.na(weekday))%>%
  filter(!is.na(part_of_day))%>%
  filter(!is.na(season))%>%
  filter(!is.na(rideable_type))%>%
  filter(member_casual == "member")%>%
  group_by(member_casual, end_lat, end_lng) %>%
  summarise(total_rides= n())%>%arrange(desc(total_rides))
```

End Station, Member = Annual

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

```
Top_50_EM_location<-head(EM_location,50)
View(Top_50_EM_location)
```

```
EC_location<- full_year_cleaned_3.1 %>%filter(!is.na(weekday))%>%
  filter(!is.na(part_of_day))%>%
  filter(!is.na(season))%>%
  filter(!is.na(rideable_type))%>%
  filter(member_casual == "casual")%>%
  group_by(member_casual, end_lat, end_lng) %>%
  summarise(total_rides= n())%>%arrange(desc(total_rides))
```

END Station, Member = Casual

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

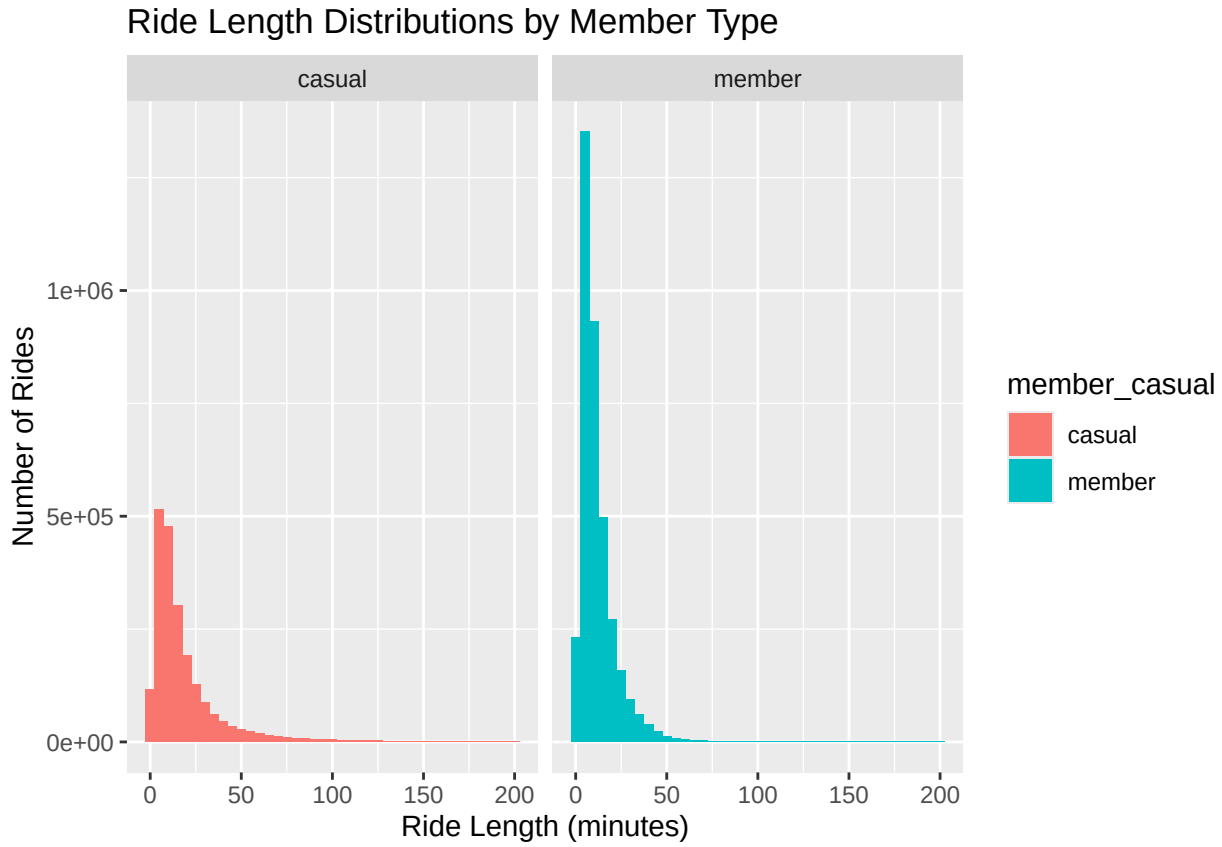
```
Top_50_EC_location<-head(EC_location,50)
View(Top_50_EC_location)
```

Make some visualizations:

This section focuses on visualizing the cleaned and processed data to gain insights into how annual members and casual riders use Cyclistic bikes differently. We'll utilize `ggplot2` to create various visualizations, including histograms, bar charts, and box plots, to explore ride length distributions, ride counts across weekdays and parts of the day, and average ride lengths.

Ride Length Distribution

```
histogram_1 <- full_year_cleaned_3.1 %>%
  filter(!is.na(weekday), !is.na(part_of_day), !is.na(ride_length), !is.na(season), ride_length < 200) %
  ggplot(aes(x = ride_length, fill = member_casual)) +
  geom_histogram(binwidth = 5, position = "dodge") +
  facet_wrap(~ member_casual) +
  labs(title = "Ride Length Distributions by Member Type",
       x = "Ride Length (minutes)",
       y = "Number of Rides")
histogram_1
```

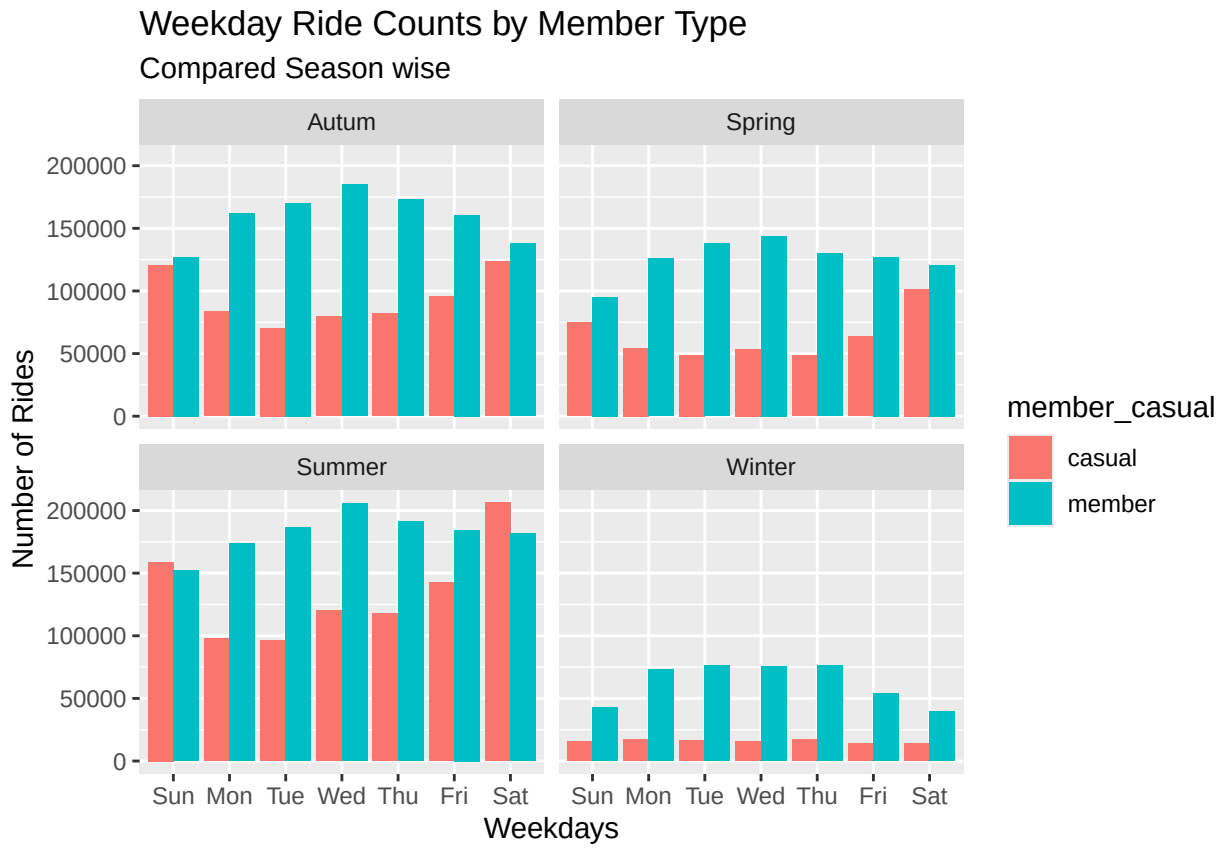


This histogram visualizes the distribution of ride lengths for both member types, with ride lengths truncated at 200 minutes to focus on typical ride durations. The facet_wrap(~ member_casual) function allows for a direct comparison between casual and member riders. The binwidth = 5 parameter groups rides into 5-minute intervals, and position = "dodge" ensures that the histograms for each member type are displayed side-by-side within each bin.

Ride Counts by Weekday and Season

```
weekday_ride_count <- full_year_cleaned_3.1 %>%
  filter(!is.na(weekday), !is.na(part_of_day), !is.na(season)) %>%
  ggplot(aes(x = weekday, fill = member_casual)) +
```

```
  geom_bar(position = "dodge") +
  facet_wrap(~ season) +
  labs(title = "Weekday Ride Counts by Member Type",
       subtitle = "Compared Season wise",
       x = "Weekdays",
       y = "Number of Rides")
weekday_ride_count
```

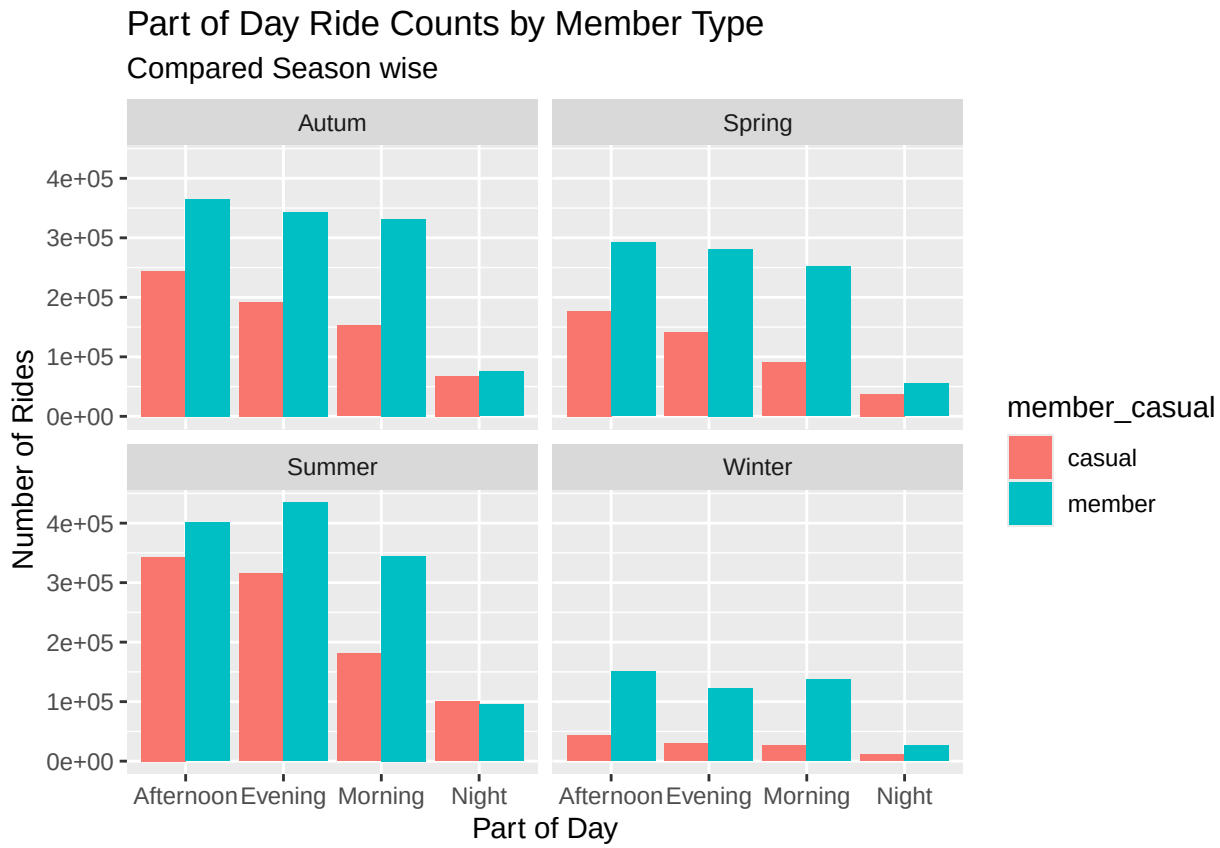


This bar chart displays the number of rides taken on each weekday, broken down by member type and season. The facet_wrap(~ season) function allows for a comparison of weekday patterns across different seasons. The position = "dodge" parameter ensures that the bars for each member type are displayed side-by-side for each weekday.

Ride Counts by Part of Day and Season

```
PD_ride_count <- full_year_cleaned_3.1 %>%
  filter(!is.na(weekday), !is.na(part_of_day), !is.na(season)) %>%
  ggplot(aes(x = part_of_day, fill = member_casual)) +
  geom_bar(position = "dodge") +
  facet_wrap(~ season) +
  labs(title = "Part of Day Ride Counts by Member Type",
       subtitle = "Compared Season wise",
       x = "Part of Day",
```

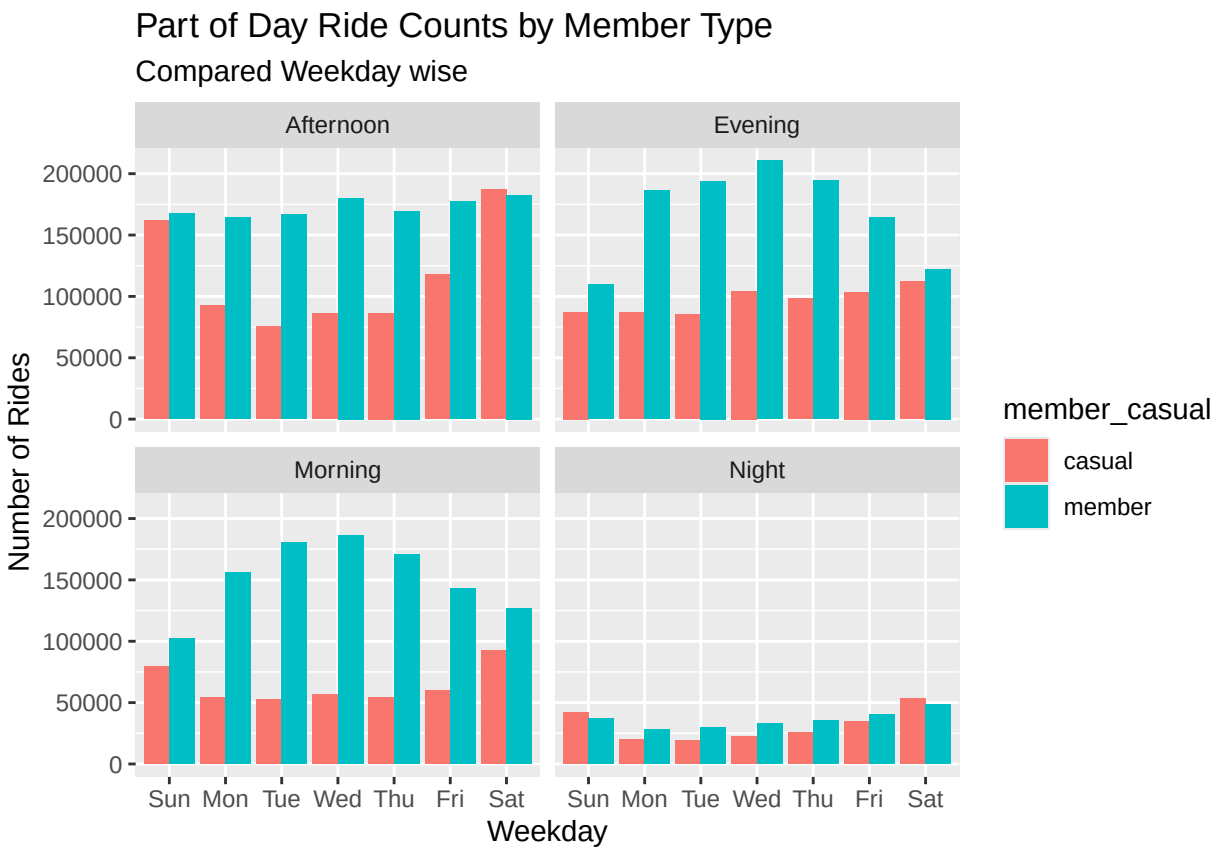
```
       y = "Number of Rides")
PD_ride_count
```



This chart shows the number of rides taken during different parts of the day (Morning, Afternoon, Evening, Night), segmented by member type and season. This helps understand daily usage patterns across different times of the year.

Ride Counts by Weekday and Part of Day

```
WD_PD_ride_count <- full_year_cleaned_3.1 %>%
  filter(!is.na(weekday), !is.na(part_of_day), !is.na(season)) %>%
  ggplot(aes(x = weekday, fill = member_casual)) +
  geom_bar(position = "dodge") +
  facet_wrap(~ part_of_day) +
  labs(title = "Part of Day Ride Counts by Member Type",
       subtitle = "Compared Weekday wise",
       x = "Weekday",
       y = "Number of Rides")
WD_PD_ride_count
```

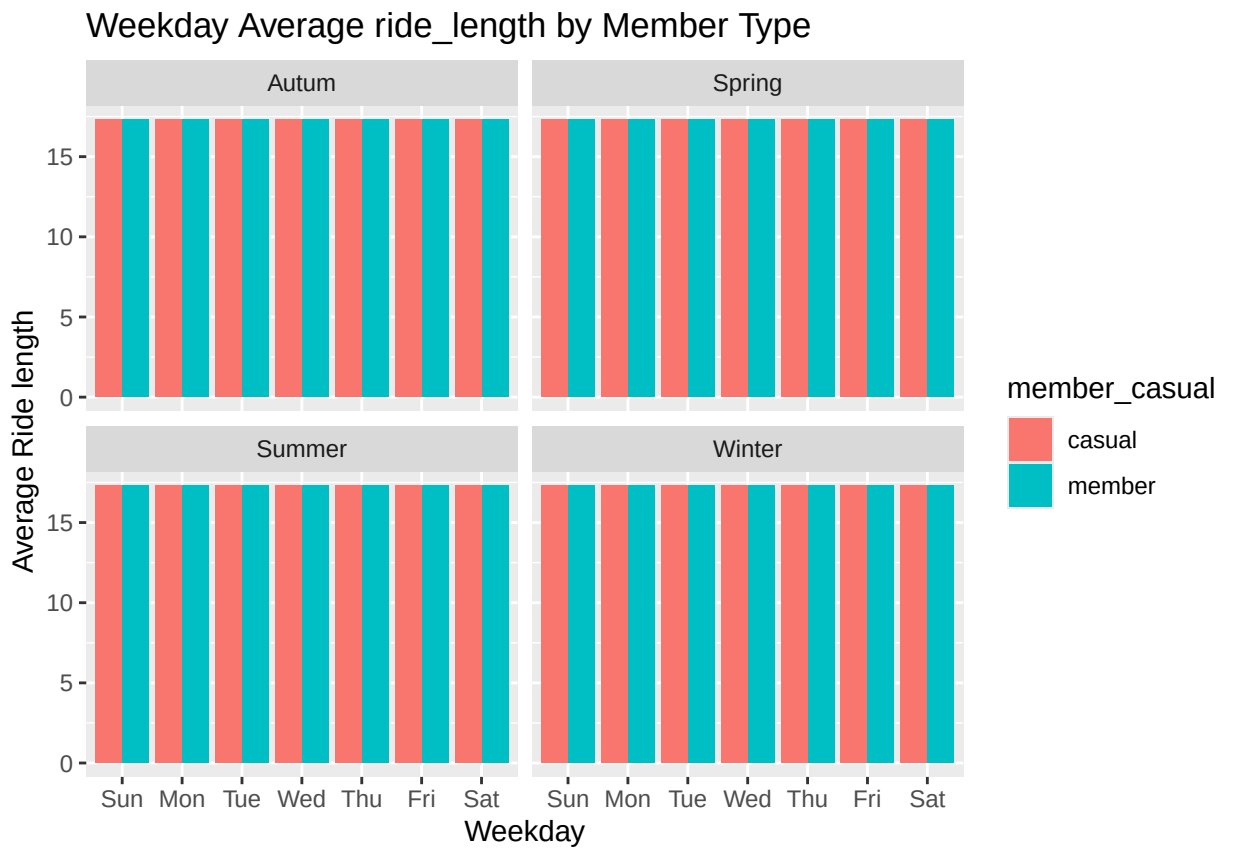


This visualization combines weekday and part-of-day information, showing ride counts across weekdays, further segmented by time of day. This provides a more granular view of usage patterns.

Average Ride Length by Weekday and Season

```
weekday_ride_length <- full_year_cleaned_3.1 %>%
  filter(!is.na(weekday), !is.na(part_of_day), !is.na(season), !is.na(ride_length)) %>%
  ggplot(aes(x = weekday, y = mean(ride_length), fill = member_casual)) +
  geom_bar(stat = "summary", position = "dodge") +
  facet_wrap(~ season) +
  labs(title = "Weekday Average ride_length by Member Type",
       x = "Weekday",
       y = "Average Ride length")
weekday_ride_length
```

```
## No summary function supplied, defaulting to `mean_se()`
## No summary function supplied, defaulting to `mean_se()`
## No summary function supplied, defaulting to `mean_se()`
## No summary function supplied, defaulting to `mean_se()`
```

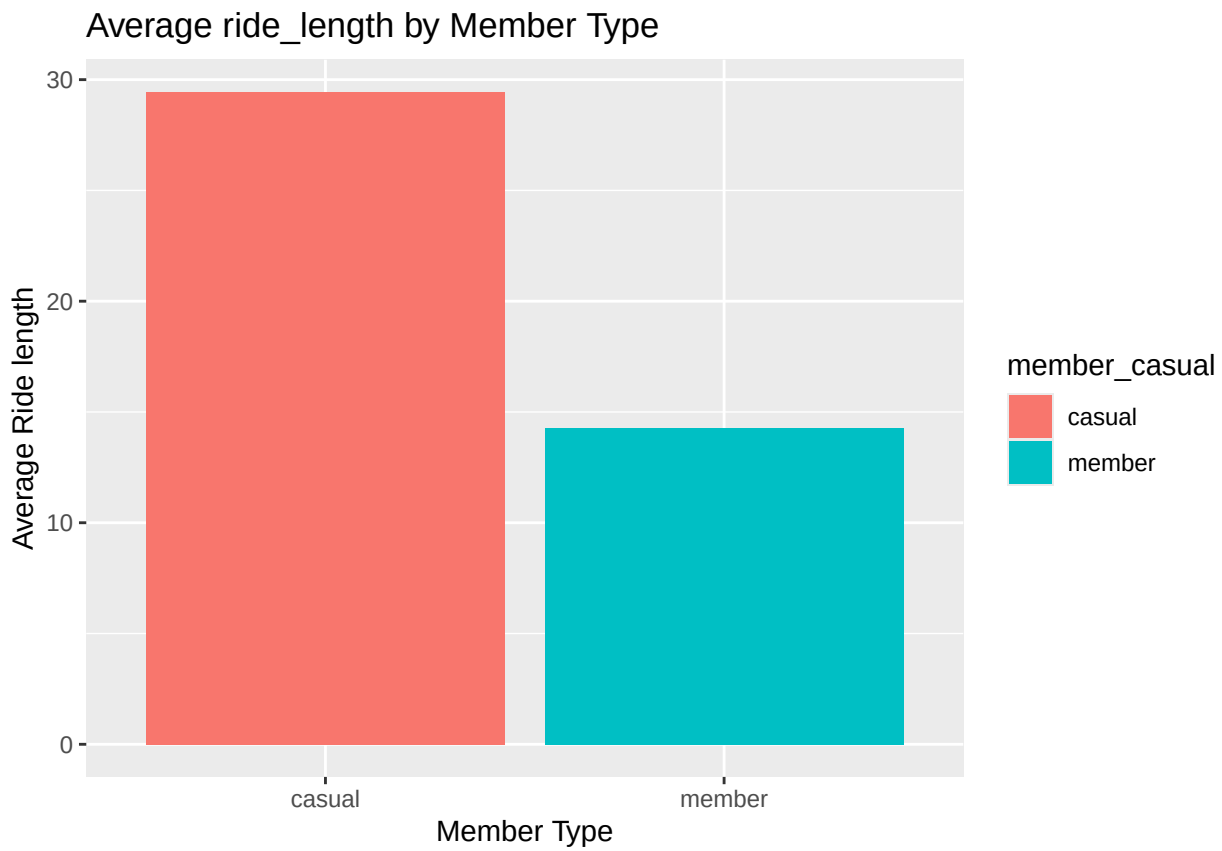


This chart displays the average ride length for each weekday, broken down by member type and season. The stat = "summary" parameter calculates the mean ride length for each group, and position = "dodge" ensures side-by-side comparison.

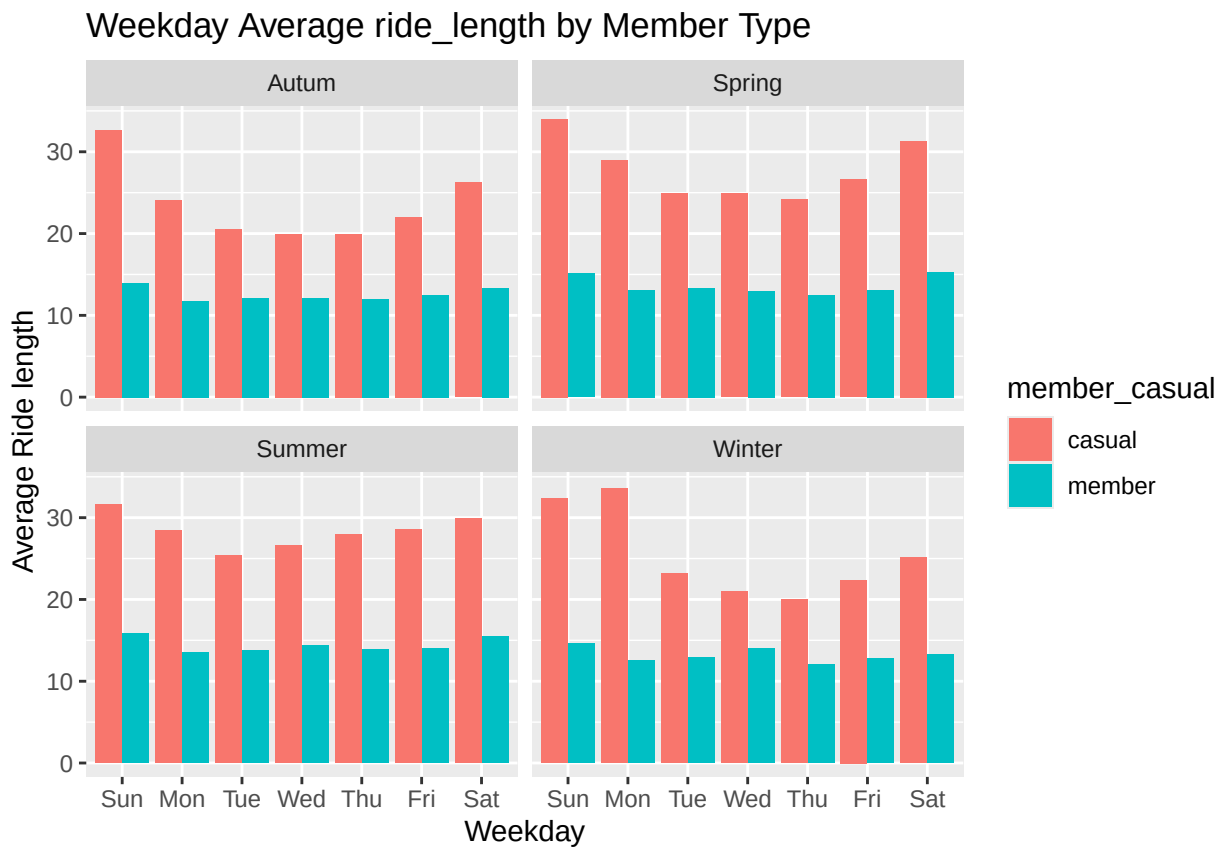
Alternative Average Ride Length Visualizations

The following code chunks explore alternative ways to visualize average ride lengths, including comparisons by member type, weekdays, seasons, and rideable types.

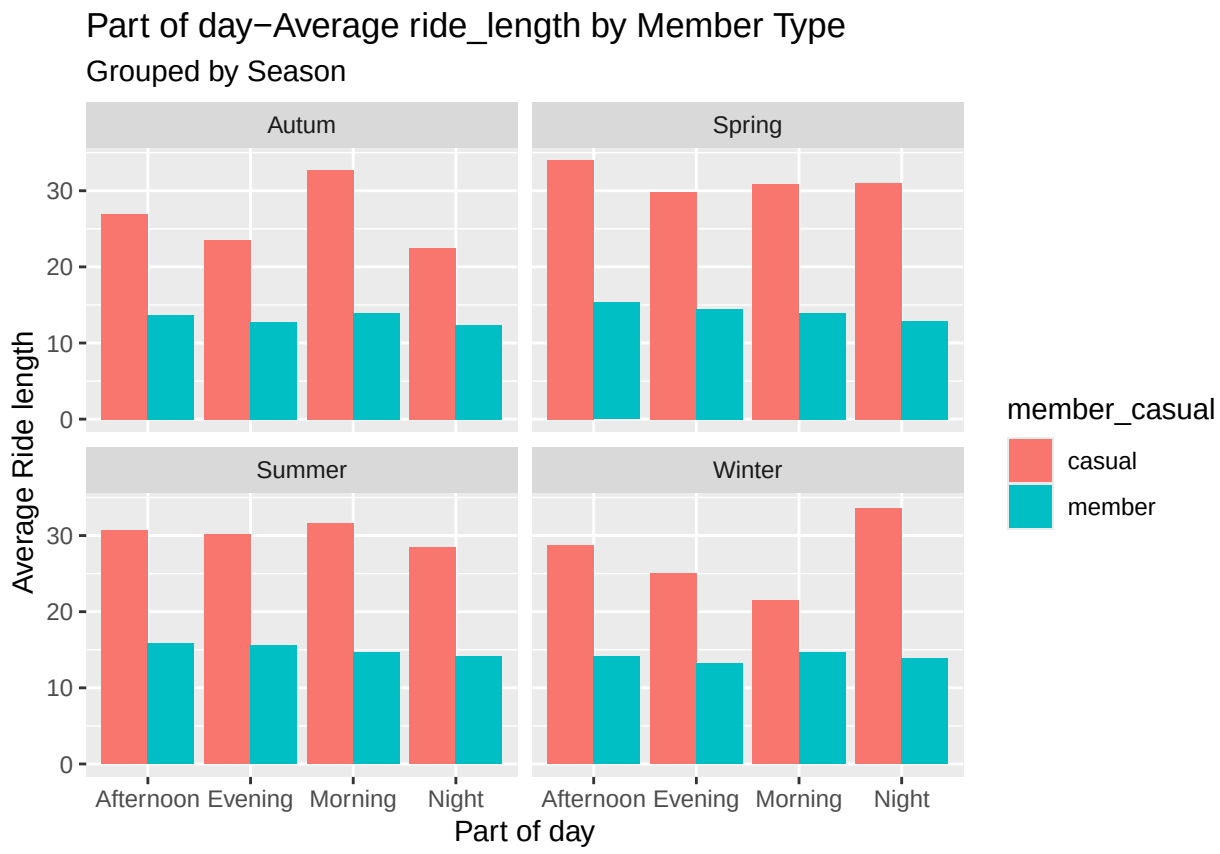
```
WD_avg_ridelength <- comparing_stats_WD %>%
  ggplot(aes(x = member_casual, y = mean_ride_length, fill = member_casual)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Average ride_length by Member Type",
       x = "Member Type",
       y = "Average Ride length")
WD_avg_ridelength
```



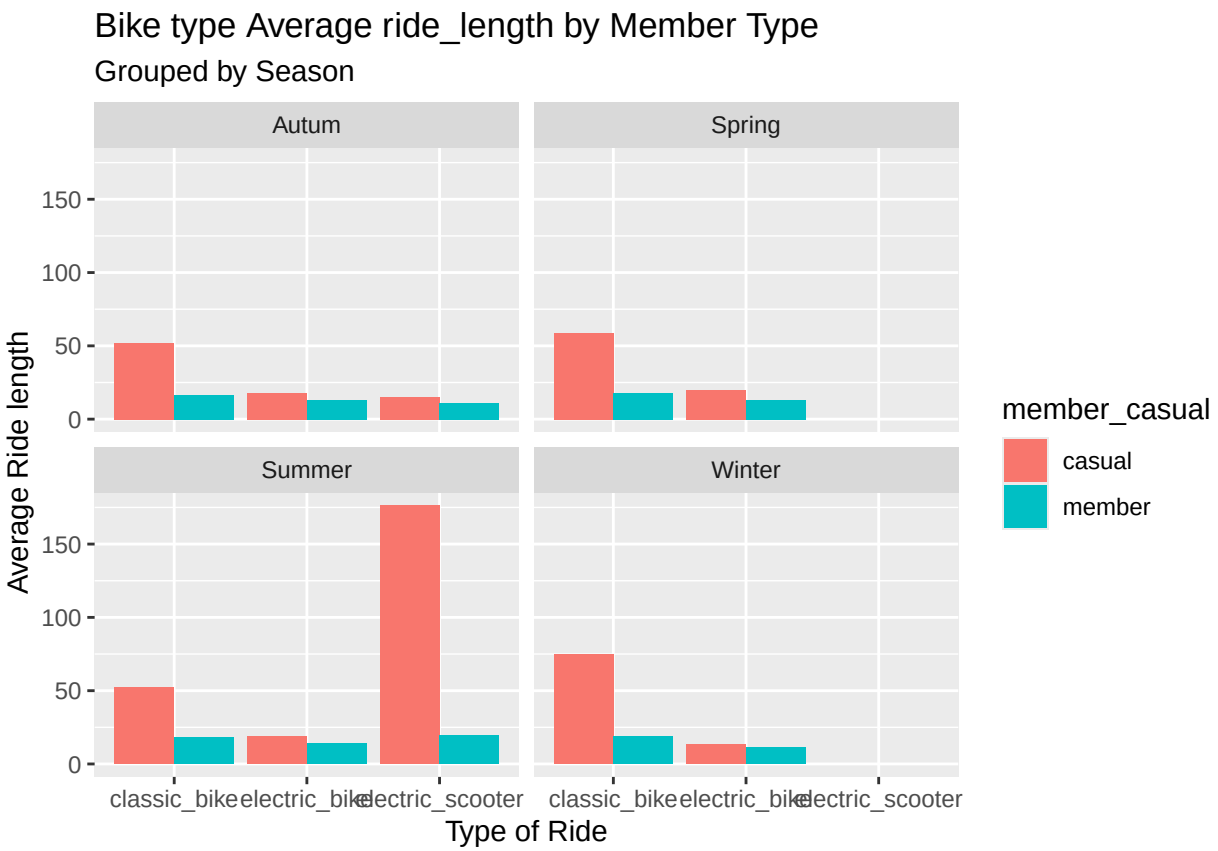

```
WD_S_avg_ridelength <- comparing_stats_A %>%
  ggplot(aes(x = weekday, y = mean_ride_length, fill = member_casual)) +
  geom_bar(stat = "identity", position = "dodge") +
  facet_wrap(~ season) +
  labs(title = "Weekday Average ride_length by Member Type",
       x = "Weekday",
       y = "Average Ride length")
WD_S_avg_ridelength
```




```
PD_S_avg_ridelength <- comparing_stats_A %>%
  ggplot(aes(x = part_of_day, y = mean_ride_length, fill = member_casual)) +
  geom_bar(stat = "identity", position = "dodge") +
  facet_wrap(~ season) +
  labs(title = "Part of day-Average ride_length by Member Type",
       subtitle = "Grouped by Season",
       x = "Part of day",
       y = "Average Ride length")
PD_S_avg_ridelength
```



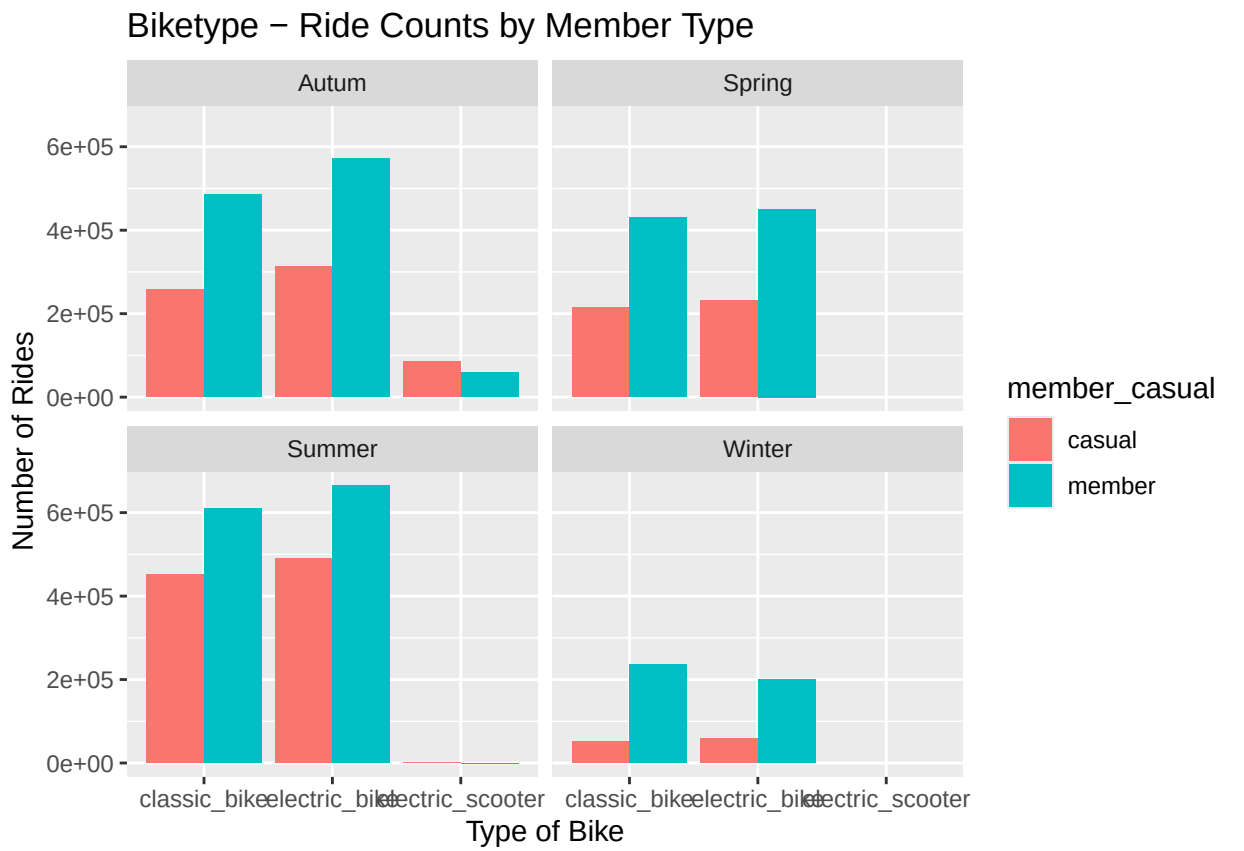

```
E_S_avg_ridelength <- comparing_stats_AA %>%
  ggplot(aes(x = rideable_type, y = mean_ride_length, fill = member_casual)) +
  geom_bar(stat = "identity", position = "dodge") +
  facet_wrap(~ season) +
  labs(title = "Bike type Average ride_length by Member Type",
       subtitle = "Grouped by Season",
       x = "Type of Ride",
       y = "Average Ride length")
E_S_avg_ridelength
```



These charts provide further insights into average ride lengths, segmented by various factors like member type, weekday, season, and bike type.

Ride Counts by Bike Type and Season

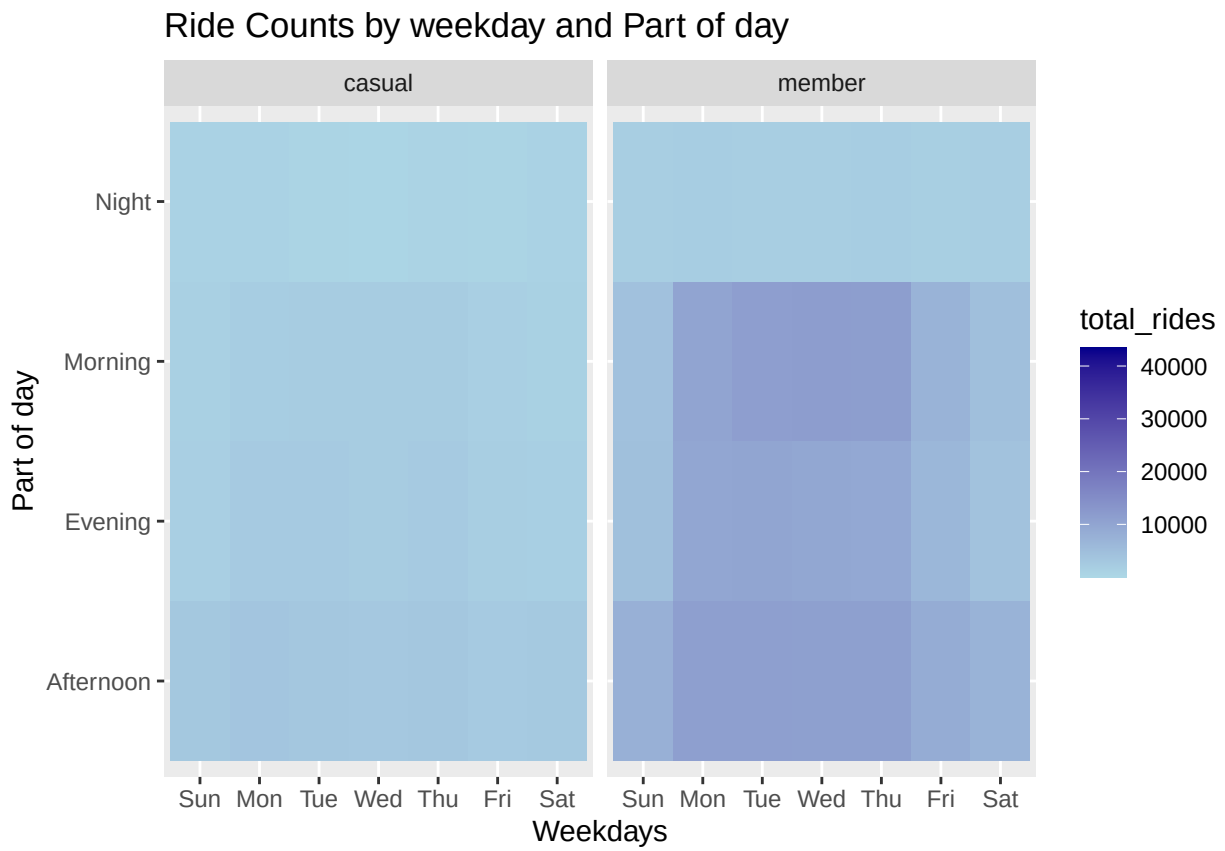
```
E_ride_count <- full_year_cleaned_3.1 %>%
  filter(!is.na(weekday), !is.na(part_of_day), !is.na(season)) %>%
  ggplot(aes(x = rideable_type, fill = member_casual)) +
  geom_bar(position = "dodge") +
  facet_wrap(~ season) +
  labs(title = "Biketype - Ride Counts by Member Type",
       x = "Type of Bike",
       y = "Number of Rides")
E_ride_count
```



This visualization shows the ride counts for different rideable types (e.g., classic bike, electric bike), segmented by member type and season.

###8. Heatmap of Ride Counts by Weekday and Part of Day

```
hm_weekday_ride_count <- comparing_stats_AA %>%
  ggplot(aes(x = weekday, y = part_of_day, fill = total_rides)) +
  geom_tile() +
  scale_fill_gradient(low = "lightblue", high = "darkblue") +
  facet_wrap(~ member_casual) +
  labs(title = "Ride Counts by weekday and Part of day",
       x = "Weekdays",
       y = "Part of day")
hm_weekday_ride_count
```



This heatmap visualizes ride counts using a color gradient, showing the distribution of rides across weekdays and parts of the day for each member type.

Saving Cleaned Data

```
Filnal_Cleaned_Avg_RL <- full_year_cleaned_3.1 %>%
  filter(!is.na(weekday), !is.na(part_of_day), !is.na(season), !is.na(ride_length), !is.na(start_lat),
```

This code filters the full_year_cleaned_3.1 data frame to remove rows with missing values in several key columns, including weekday, part_of_day, season, ride_length, start_lat, start_lng, end_lat, and end_lng.

```
write.csv(Filnal_Cleaned_Avg_RL, file = 'Final_cleaned_Ride_data_2024_1.csv')
```

This line saves the cleaned and processed data frame Filnal_Cleaned_Avg_RL to a CSV file named Final_cleaned_Ride_data_2024_1.csv.

Saving Top Location Data

```
write.csv(Top_50_SC_location, file = 'Top_50_SC_location.csv')
write.csv(Top_50_S_location, file = 'Top_50_SM_location.csv')
write.csv(Top_50_EC_location, file = 'Top_50_EC_location.csv')
write.csv(Top_50_EM_location, file = 'Top_50_EM_location.csv')
```

These lines save the data frames containing the top 50 start and end locations for casual and member riders to separate CSV files. This data can be used for further spatial analysis or visualization.