Cyclistic Bike Share Case Study

Carson Hartje

2024-11-13

#Introduction My name is Carson Hartje and I recently completed the Google Data Analytics case study. My goal with this project was to further develop my skills in R and further improve my overall understanding and capability of analyzing data. I hope to soon become a full-time data analyst, as understanding data has always been an area of interest for me, both personally and professionally.

#Background In 2016, Cyclistic launched a successful bike-share offering. They have since seen exponential growth, expanding to a fleet of 5,824 bicycles that are geotracked and locked into a network of 692 stations across Chicago. Cyclistic users have three options when using the service: single-ride passes, full-day passes, and annual memberships. Single-ride passes and full-day passes are used by "casual riders", while those who purchase annual memberships are referred to as "Members." Financial analysts at Cyclystic have concluded that members are much more profitable than casual riders. The purpose of this study is to find out how to get more casual riders to convert to members and answer the question "How do annual members and casual riders use Cyclistic bikes differently?"

#Preparation The data used in this specific study is collected from 12 consecutive months, June 2021 - May 2022. The link to all available data sets regarding this capstone can be found here.

I began my project by loading the appropriate libraries.

```
library(tidyverse) #clean data
```

```
## -- 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 1.9.3     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(lubridate) #adjust format of dates
library(data.table) #for exporting the data
```

```
##
## Attaching package: 'data.table'
##
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
```

```
##
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
##
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
library(hms)#adjust format of time
```

```
##
## Attaching package: 'hms'
##
## The following object is masked from 'package:lubridate':
##
##     hms
```

```
library(janitor)#help examine and clean data
```

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

I then load the csv files and create names for them

```
bike_2022_05 <- read.csv('2022 05 bike data.csv')
bike_2022_04 <- read.csv('2022 04 bike data.csv')
bike_2022_03 <- read.csv('2022 03 bike data.csv')
bike_2022_02 <- read.csv('2022 02 bike data.csv')
bike_2022_01 <- read.csv('2022 01 bike data.csv')
bike_2021_12 <- read.csv('2021 12 bike data.csv')
bike_2021_11 <- read.csv('2021 11 bike data.csv')
bike_2021_10 <- read.csv('2021 10 bike data.csv')
bike_2021_09 <- read.csv('2021 09 bike data.csv')
bike_2021_08 <- read.csv('2021 08 bike data.csv')
bike_2021_07 <- read.csv('2021 07 bike data.csv')
bike_2021_06 <- read.csv('2021 06 bike data.csv')
```

Combine the 12 months of data into one data frame using rbind(). Make a copy of the data frame in case something goes wrong at some point.

```
#merging all the files into one (bike_12_months)
bike_12_months <- rbind(bike_2022_05, bike_2022_04, bike_2022_03, bike_2022_02, bike_2022_01, bike_2021

  #Remove the monthly files from my environment
remove(bike_2022_05, bike_2022_04, bike_2022_03, bike_2022_02, bike_2022_01, bike_2021_12, bike_2021_11

  #Create a copy of the data frame for reassurance
bike_12_months_copy <- bike_12_months
```

Check to make sure the data is correctly merged

```r
view(bike_12_months_copy)

head(bike_12_months_copy) #view the rows
```

```
##            ride_id rideable_type          started_at            ended_at
## 1 EC2DE40644C6B0F4  classic_bike 2022-05-23 23:06:58 2022-05-23 23:40:19
## 2 1C31AD03897EE385  classic_bike 2022-05-11 08:53:28 2022-05-11 09:31:22
## 3 1542FBEC830415CF  classic_bike 2022-05-26 18:36:28 2022-05-26 18:58:18
## 4 6FF59852924528F8  classic_bike 2022-05-10 07:30:07 2022-05-10 07:38:49
## 5 483C52CAAE12E3AC  classic_bike 2022-05-10 17:31:56 2022-05-10 17:36:57
## 6 C0A3AA5A614DCE01  classic_bike 2022-05-04 14:48:55 2022-05-04 14:56:04
##                 start_station_name start_station_id
## 1            Wabash Ave & Grand Ave     TA1307000117
## 2 DuSable Lake Shore Dr & Monroe St            13300
## 3           Clinton St & Madison St     TA1305000032
## 4           Clinton St & Madison St     TA1305000032
## 5           Clinton St & Madison St     TA1305000032
## 6           Carpenter St & Huron St            13196
##                end_station_name end_station_id start_lat start_lng  end_lat
## 1        Halsted St & Roscoe St   TA1309000025  41.89147 -87.62676 41.94367
## 2   Field Blvd & South Water St          15534  41.88096 -87.61674 41.88635
## 3       Wood St & Milwaukee Ave          13221  41.88224 -87.64107 41.90765
## 4        Clark St & Randolph St   TA1305000030  41.88224 -87.64107 41.88458
## 5           Morgan St & Lake St   TA1306000015  41.88224 -87.64107 41.88578
## 6 Sangamon St & Washington Blvd          13409  41.89456 -87.65345 41.88316
##     end_lng member_casual
## 1 -87.64895        member
## 2 -87.61752        member
## 3 -87.67255        member
## 4 -87.63189        member
## 5 -87.65102        member
## 6 -87.65110        member
```

```r
colnames(bike_12_months_copy) #view the column names
```

```
##  [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"
```

```r
nrow(bike_12_months_copy) #the number of rows in our data frame
```

```
## [1] 5860776
```

```r
summary(bike_12_months_copy) #view the dimensions of the data
```

```
##    ride_id          rideable_type       started_at          ended_at
##  Length:5860776     Length:5860776     Length:5860776     Length:5860776
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##  start_station_name start_station_id   end_station_name   end_station_id
##  Length:5860776     Length:5860776     Length:5860776     Length:5860776
##  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.   :-87.84   Min.   :41.39   Min.   :-88.97
##  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.   :45.64   Max.   :-73.80   Max.   :42.17   Max.   :-87.49
##                                   NA's   :5036    NA's   :5036
##  member_casual
##  Length:5860776
##  Class :character
##  Mode  :character
##
##
##
##
```

Convert the started_at and ended_at columns to a standard format to avoid "character string is not in a standard unambiguous format" message.

```
bike_12_months_copy$started_at <- as.POSIXct(bike_12_months_copy$started_at, format="%Y-%m-%d %H:%M:%S"

bike_12_months_copy$ended_at <- as.POSIXct(bike_12_months_copy$ended_at, format="%Y-%m-%d %H:%M:%S", tz
```

Now I can remove columns I know I will not use for this analysis

```
bike_12_months_copy <- bike_12_months_copy %>%
  select(-c(start_lat, start_lng, end_lat, end_lng, start_station_id,end_station_id, end_station_name))
```

#Process Next I create a column that displays each ride length in minutes for as much accuracy as possible. I use difftime() to convert the time from the original table. The new column will be called ride_length.

```
bike_12_months_copy$ride_length <- difftime(bike_12_months_copy$ended_at, bike_12_months_copy$started_a

  #Column for the day of the week. I use wday() for the precise day of the week. The new column will be
bike_12_months_copy$day_of_week <-wday(bike_12_months_copy$started_at)
```

Now create columns that will show the date, day of the week, month, day, and year. Finally, a column for the corresponding season will be made.

```r
bike_12_months_copy$date <- as.Date(bike_12_months_copy$started_at)
bike_12_months_copy$day_of_week <- format(as.Date(bike_12_months_copy$date), "%A")
bike_12_months_copy$month <- format(as.Date(bike_12_months_copy$date), "%m")
bike_12_months_copy$day <- format(as.Date(bike_12_months_copy$date), "%d")
bike_12_months_copy$year <- format(as.Date(bike_12_months_copy$date), "%Y")

  #column showing the season that corresponds to the month
bike_12_months_copy <- bike_12_months_copy %>% mutate(season =
  case_when(month == "03" ~ "Spring",
            month == "04" ~ "Spring",
            month == "05" ~ "Spring",
            month == "06" ~ "Summer",
            month == "07" ~ "Summer",
            month == "08" ~ "Summer",
            month == "09" ~ "Fall",
            month == "10" ~ "Fall",
            month == "11" ~ "Fall",
            month == "12" ~ "Winter",
            month == "01" ~ "Winter",
            month == "02" ~ "Winter")
)
```

Now I remove duplicate rows, rows with null values, and rows that show ride length is <= 0. (1,552,503 rows were removed)

```r
bike_12_months_copy <- distinct(bike_12_months_copy)
#remove rows with null values
bike_12_months_copy <- na.omit(bike_12_months_copy)
#remove rows where ride_length is <= to zero
bike_12_months_copy <- bike_12_months_copy[!(bike_12_months_copy$ride_length <=0),]
```

Then convert the ride_length column to numeric because it says ‘mins’ after each number in the column

```r
bike_12_months_copy$ride_length <- as.numeric(as.character(bike_12_months_copy$ride_length))
is.numeric(bike_12_months_copy$ride_length)
```

```
## [1] TRUE
```

The row titled “Member_casual” contains the customer types, casual and member. I felt there could be a more straightforward label for this row so I will rename it customer_type.

```r
bike_12_months_copy <- bike_12_months_copy %>%
  rename(customer_type = member_casual)
```

#Analyzation Because the datat set is so large, axes on the graphs will default to scientific notation. The next code chunk will overwrite this.

```r
options(scipen = 999)
```

Ride totals - there were around 4.3 million rides taken from the data that was observed after cleaning.

```
nrow(bike_12_months_copy) #Total number of rides
```

```
## [1] 4308273
```

```
round(nrow(bike_12_months_copy), digits = -5) #Number of rides rounded
```

```
## [1] 4300000
```

Now count how many of each member type there are (Casual vs. Member). There are 1,746,854 casual riders compared to 2,561,419 Members.

```
bike_12_months_copy %>% count(customer_type)
```

```
##   customer_type       n
## 1        casual 1746854
## 2        member 2561419
```

Now the number of times each type of bike was used. Classic bikes saw the most use at 2,275,632 rides Electric bikes saw the second most use at 1,867,614 rides Docked bikes saw significantly less use at 165,027 rides

```
bike_12_months_copy %>% count(rideable_type)
```

```
##   rideable_type       n
## 1  classic_bike 2275632
## 2   docked_bike  165027
## 3 electric_bike 1867614
```

Next, I look at the type of bike and how many times they were used according to each customer type. Classic bikes are the most popular with both casual riders and members.

```
bike_12_months_copy %>% count(customer_type, rideable_type)
```

```
##   customer_type rideable_type       n
## 1        casual  classic_bike  806723
## 2        casual   docked_bike  165027
## 3        casual electric_bike  775104
## 4        member  classic_bike 1468909
## 5        member electric_bike 1092510
```

Then I look at how many trips were taken each day.

Saturday is the busiest day of the week, with 702,052 rides. Friday is the least busy day, with 582,922 rides.

```
bike_12_months_copy %>% count(day_of_week)
```

```
##   day_of_week      n
## 1      Friday 582922
## 2      Monday 594383
## 3    Saturday 702052
## 4      Sunday 641529
## 5    Thursday 598309
## 6     Tuesday 604373
## 7   Wednesday 584705
```

We can explore this further by looking at the most popular day of the week between customer types. The busiest days for members are Tuesdays, with 410,851. The least busy day for members is Sunday, with 310,083 rides. The busiest day for casual riders is Saturday, with 367,145. The least busy day for casual riders is Wednesday, with 190,522 rides.

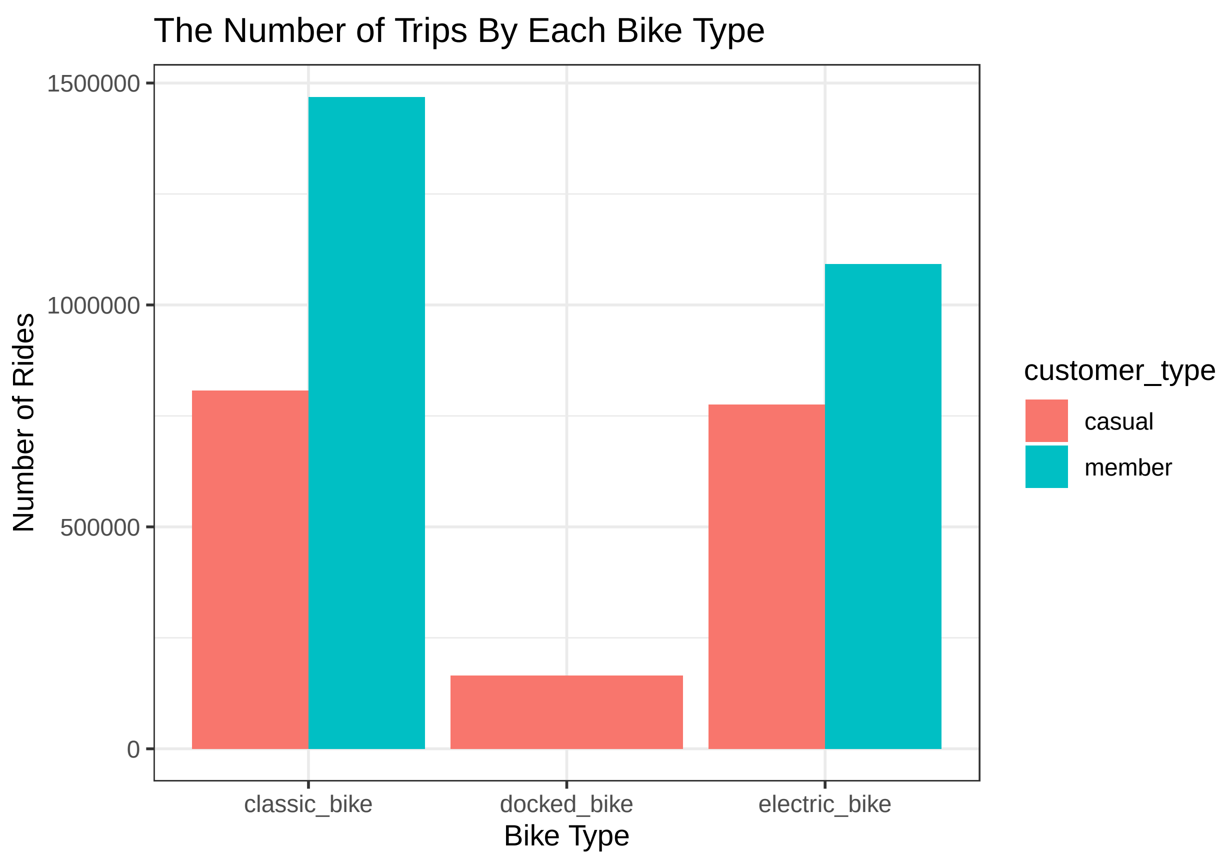
```r
bike_12_months_copy %>% count(day_of_week, customer_type)
```

```
##    day_of_week customer_type      n
## 1       Friday        casual 235235
## 2       Friday        member 347687
## 3       Monday        casual 218656
## 4       Monday        member 375727
## 5     Saturday        casual 367145
## 6     Saturday        member 334907
## 7       Sunday        casual 331446
## 8       Sunday        member 310083
## 9     Thursday        casual 210328
## 10    Thursday        member 387981
## 11     Tuesday        casual 193522
## 12     Tuesday        member 410851
## 13   Wednesday        casual 190522
## 14   Wednesday        member 394183
```

#Graphing

```r
bike_12_months_copy %>%
  group_by(rideable_type, customer_type) %>%
  dplyr:: summarize(count_trips = n()) %>%
  ggplot(aes(x = rideable_type, y = count_trips, fill = customer_type))+
  geom_bar(stat = 'identity', position = 'dodge') +
  theme_bw()+
labs(title = 'The Number of Trips By Each Bike Type', x = 'Bike Type', y = 'Number of Rides')
```

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



The graph above displays the number of trips taken on each bike. Classic bikes see the most use overall, with casual riders taking 806,723 trips, and members taking 1,468,909 trips. Electric bikes are the second most popular, with casual riders taking 775,104 trips and members taking 1,092,510 trips. Casual riders are the only ones to utilize docked bikes, with only 165,027 trips taken.

#Mean and Median Ride Times The average ride time was 19.11 minutes, the median was 10.67 minutes. The maximum ride time was 41,629 minutes, or roughly 29 days. This is likely an error within the data as it does not make much sense as to why a ride would last this long.

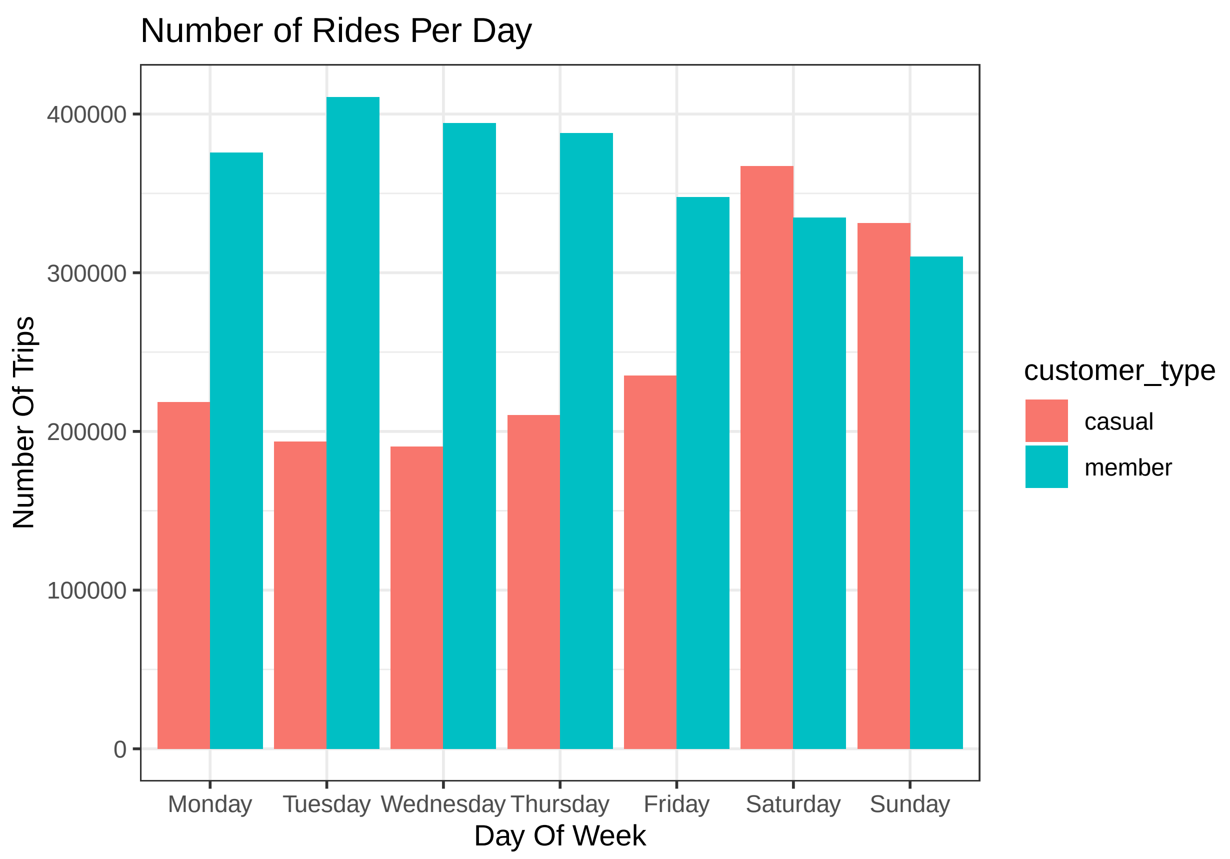
```
summary(bike_12_months_copy$ride_length) #max/min/median/mean length of rides
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##     0.02     6.00    10.67    19.11    19.38 41629.17
```

```
#put the days of the week in standard order
bike_12_months_copy$day_of_week <- ordered(bike_12_months_copy$day_of_week, levels = c('Monday', 'Tuesda

bike_12_months_copy %>%
  group_by(customer_type, day_of_week) %>%
  dplyr::summarize(count_trips = n()) %>%
  ggplot(aes(x = day_of_week, y = count_trips, fill = customer_type)) +
  geom_bar(stat = 'identity', position = 'dodge') +
  theme_bw() +
  labs(title = 'Number of Rides Per Day', x = 'Day Of Week', y = 'Number Of Trips')
```

```
## ‘summarise()‘ has grouped output by ’customer_type’. You can override using the
## ‘.groups‘ argument.
```

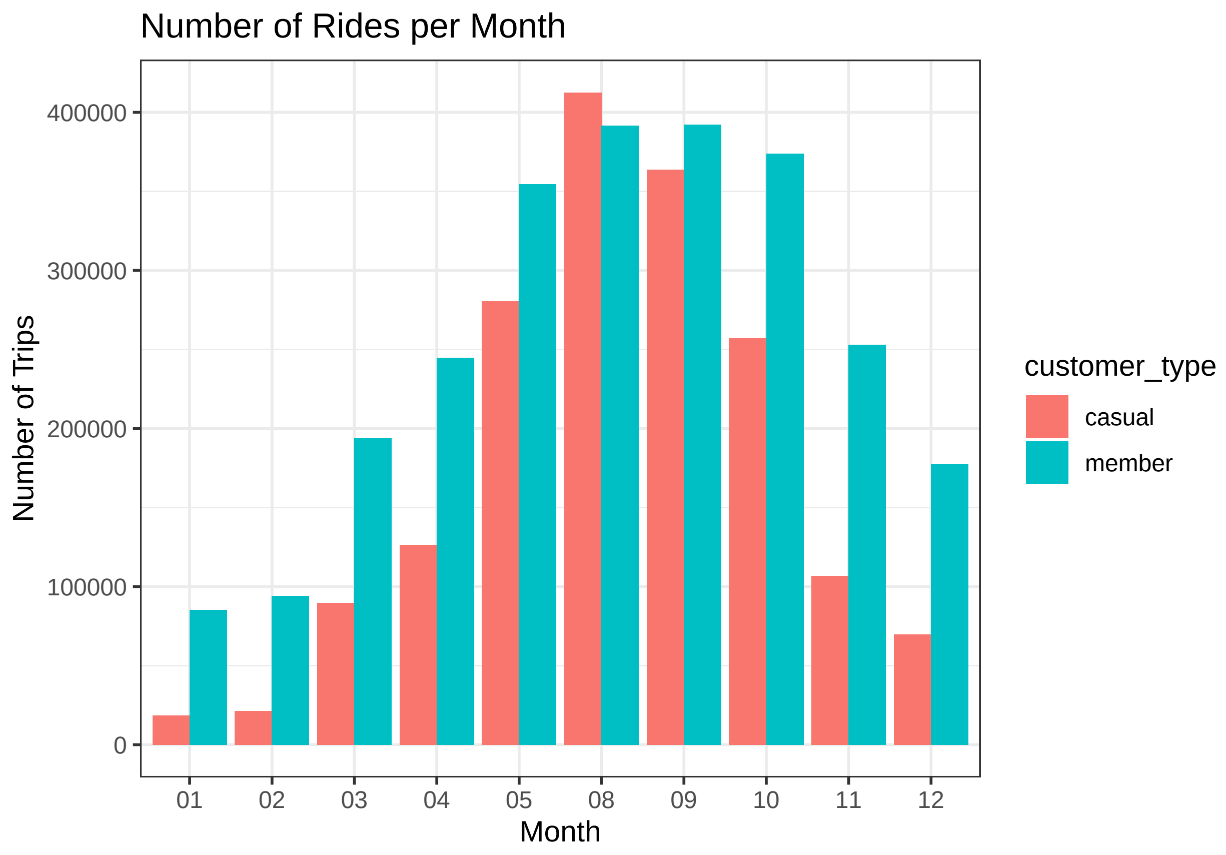


The graph above shows the number of rides by day of the week. Members heavily dominate the usage of bikes throughout the week. Weekends, however, see casual riders using the bikes more. Tuesdays are the busiest days for members, while Saturdays are the busiest for casual riders.

```
#arranges months in order
bike_12_months_copy$month <- ordered(bike_12_months_copy$month, levels=c("01", "02", "03", "04", "05",

#plot number of rides per month
bike_12_months_copy%>%
  group_by(customer_type, month) %>%
  dplyr::summarize(count_trips = n()) %>%
  ggplot(aes(x= month, y=count_trips, fill=customer_type)) +
  geom_bar(stat='identity', position = 'dodge') +
  theme_bw() +
  labs(title ="Number of Rides per Month", x = "Month", y = "Number of Trips")
```

```
## ‘summarise()‘ has grouped output by ’customer_type’. You can override using the
## ‘.groups‘ argument.
```



Above is a table for number of rides per month The summer months are, unsurprisingly, the busiest. The number of casual riders exceeds the number of member riders in the summer months. Overall, it is clear that member use is still dominant for the majority of the months in the year.

Next, I look at the most popular starting points between each customer type. I start with members.

```
bike_12_months_copy %>%
  group_by(customer_type,start_station_name) %>%
  dplyr::summarise(number_of_ride = n()) %>%
  filter(start_station_name != "", "member" == customer_type) %>%
  arrange(-number_of_ride) %>%
  head(n=5) %>%
  select(-customer_type)
```

```
## ‘summarise()‘ has grouped output by ’customer_type’. You can override using the
## ‘.groups‘ argument.
## Adding missing grouping variables: ‘customer_type‘

## # A tibble: 5 x 3
## # Groups:   customer_type [1]
##   customer_type start_station_name       number_of_ride
##   <chr>         <chr>                             <int>
## 1 member        Kingsbury St & Kinzie St          20527
## 2 member        Clark St & Elm St                 18419
## 3 member        Ellis Ave & 60th St               18413
## 4 member        Wells St & Concord Ln             17871
## 5 member        University Ave & 57th St          17156
```

And now the casual riders.

```
bike_12_months_copy %>%
  group_by(customer_type,start_station_name) %>%
  dplyr::summarise(number_of_ride = n()) %>%
  filter(start_station_name != "", "casual" == customer_type) %>%
  arrange(-number_of_ride) %>%
  head(n=5) %>%
  select(-customer_type)
```

```
## `summarise()` has grouped output by 'customer_type'. You can override using the
## `.groups` argument.
## Adding missing grouping variables: `customer_type`
```

```
## # A tibble: 5 x 3
## # Groups:   customer_type [1]
##   customer_type start_station_name                 number_of_ride
##   <chr>         <chr>                                       <int>
## 1 casual        Streeter Dr & Grand Ave                     41572
## 2 casual        DuSable Lake Shore Dr & Monroe St           21899
## 3 casual        Millennium Park                             20521
## 4 casual        Michigan Ave & Oak St                       18194
## 5 casual        DuSable Lake Shore Dr & North Blvd          16930
```

When comparing the two chunks of code above, it seems the casual riders are likely starting from points of interest because the top start station for casual riders sees more than double the riders compared to the top start station for members.

#Recomendations and Conclusion

Key takeaways

Classic Bikes are the most popular bike in the fleet, docked bikes are used by an extremely small percentage of users Summer months are much more popular with casual riders Casual riders see much more activity on the weekends Recommendations

This study was aimed at finding out how members and casual riders use the bikes of cyclistic differently. After reviewing all the information from above, I can offer a few ideas on how to attain more members.

Create a marketing campaign that raises more awareness about the perks of becoming a member in early spring or late winter so the casual riders will have time to get the message and purchase a membership in time for their peak use. Run a promotion at the most popular start stations for casual riders, further incentivizing them to upgrade to member. Get rid of docked bikes as only a small percentage of the users actually utilize these. Classic bikes are the most used bike between both groups, so more of those can likely be purchased. Thanks For Reading!