Cyclistic bike-share

Silvester Cheptumo

2023-12-06

suppressWarnings({  
library(dplyr)  
library(tidyverse)  
library(ggplot2)  
library(lubridate)  
library(forcats)  
})

##   
## Attaching package: 'dplyr'

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

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

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ forcats 1.0.0 ✔ readr 2.1.4  
## ✔ ggplot2 3.4.2 ✔ stringr 1.5.0  
## ✔ lubridate 1.9.2 ✔ tibble 3.2.1  
## ✔ purrr 1.0.1 ✔ tidyr 1.3.0  
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

divvy4 <- read.csv("divvy-202004-tripdata.csv")  
divvy5 <- read.csv("divvy-202005-tripdata.csv")  
divvy6 <- read.csv("divvy-202006-tripdata.csv")  
divvy7 <- read.csv("divvy-202007-tripdata.csv")  
divvy8 <- read.csv("divvy-202008-tripdata.csv")  
divvy9 <- read.csv("divvy-202009-tripdata.csv")  
divvy10 <- read.csv("divvy-202010-tripdata.csv")  
divvy11 <- read.csv("divvy-202011-tripdata.csv")  
divvy12 <- read.csv("divvy-202012-tripdata.csv")  
divvy01 <- read.csv("divvy-202101-tripdata.csv")  
divvy02 <- read.csv("divvy-202102-tripdata.csv")  
divvy03 <- read.csv("divvy-202103-tripdata.csv")

### merging the six datasets

divvy\_merged <- rbind(divvy4,divvy5,divvy6,divvy7,divvy8,divvy9,divvy10, divvy11, divvy12, divvy01, divvy02, divvy03)  
str(divvy\_merged)

## 'data.frame': 3489748 obs. of 13 variables:  
## $ ride\_id : chr "A847FADBBC638E45" "5405B80E996FF60D" "5DD24A79A4E006F4" "2A59BBDF5CDBA725" ...  
## $ rideable\_type : chr "docked\_bike" "docked\_bike" "docked\_bike" "docked\_bike" ...  
## $ started\_at : chr "2020-04-26 17:45:14" "2020-04-17 17:08:54" "2020-04-01 17:54:13" "2020-04-07 12:50:19" ...  
## $ ended\_at : chr "2020-04-26 18:12:03" "2020-04-17 17:17:03" "2020-04-01 18:08:36" "2020-04-07 13:02:31" ...  
## $ start\_station\_name: chr "Eckhart Park" "Drake Ave & Fullerton Ave" "McClurg Ct & Erie St" "California Ave & Division St" ...  
## $ start\_station\_id : chr "86" "503" "142" "216" ...  
## $ end\_station\_name : chr "Lincoln Ave & Diversey Pkwy" "Kosciuszko Park" "Indiana Ave & Roosevelt Rd" "Wood St & Augusta Blvd" ...  
## $ end\_station\_id : chr "152" "499" "255" "657" ...  
## $ start\_lat : num 41.9 41.9 41.9 41.9 41.9 ...  
## $ start\_lng : num -87.7 -87.7 -87.6 -87.7 -87.6 ...  
## $ end\_lat : num 41.9 41.9 41.9 41.9 42 ...  
## $ end\_lng : num -87.7 -87.7 -87.6 -87.7 -87.7 ...  
## $ member\_casual : chr "member" "member" "member" "member" ...

The resulting dataset has 13 columns and more than 3 million rows. Of these columns, we will, only be needing a few of them.

#selecting the columns to use  
divvy\_df <- divvy\_merged %>%  
 select(rideable\_type, start\_station\_name, end\_station\_name, member\_casual, started\_at, ended\_at)

check for missing values

colSums(is.na(divvy\_df))

## rideable\_type start\_station\_name end\_station\_name member\_casual   
## 0 0 0 0   
## started\_at ended\_at   
## 0 0

Since there are no missing values, we proceed to create new columns from the started\_at and ended\_at columns

#converting dates from character to date format and finding the cycling duration  
divvy\_df$started\_at <- ymd\_hms(divvy\_df$started\_at) #ymd\_hms is a function in lubridate library  
divvy\_df$ended\_at <- ymd\_hms(divvy\_df$ended\_at)

Extract cycling\_time in minutes

divvy\_df <- divvy\_df %>%  
 mutate(cycling\_time = ended\_at - started\_at) %>%  
 mutate(cycling\_time\_minutes = as.numeric(round(cycling\_time/60, digits=2)))

Extract hours from start time

divvy\_df <- divvy\_df %>%  
 mutate(start\_hour = format(started\_at, "%H"))  
str(divvy\_df)

## 'data.frame': 3489748 obs. of 9 variables:  
## $ rideable\_type : chr "docked\_bike" "docked\_bike" "docked\_bike" "docked\_bike" ...  
## $ start\_station\_name : chr "Eckhart Park" "Drake Ave & Fullerton Ave" "McClurg Ct & Erie St" "California Ave & Division St" ...  
## $ end\_station\_name : chr "Lincoln Ave & Diversey Pkwy" "Kosciuszko Park" "Indiana Ave & Roosevelt Rd" "Wood St & Augusta Blvd" ...  
## $ member\_casual : chr "member" "member" "member" "member" ...  
## $ started\_at : POSIXct, format: "2020-04-26 17:45:14" "2020-04-17 17:08:54" ...  
## $ ended\_at : POSIXct, format: "2020-04-26 18:12:03" "2020-04-17 17:17:03" ...  
## $ cycling\_time : 'difftime' num 1609 489 863 732 ...  
## ..- attr(\*, "units")= chr "secs"  
## $ cycling\_time\_minutes: num 26.82 8.15 14.38 12.2 52.92 ...  
## $ start\_hour : chr "17" "17" "17" "12" ...

Extracting month-year from start time

# Extract the month and year from the start time  
divvy\_df <- divvy\_df %>%  
 mutate(Month\_year = format(started\_at, "%Y-%m"))

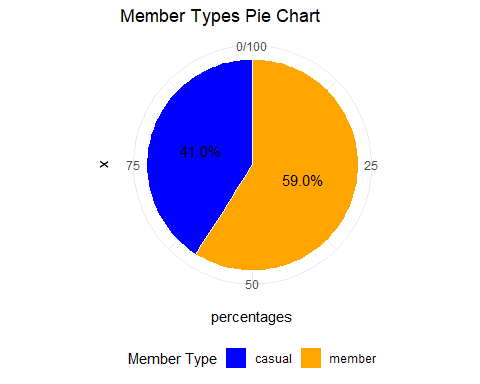
str(divvy\_df)

## 'data.frame': 3489748 obs. of 10 variables:  
## $ rideable\_type : chr "docked\_bike" "docked\_bike" "docked\_bike" "docked\_bike" ...  
## $ start\_station\_name : chr "Eckhart Park" "Drake Ave & Fullerton Ave" "McClurg Ct & Erie St" "California Ave & Division St" ...  
## $ end\_station\_name : chr "Lincoln Ave & Diversey Pkwy" "Kosciuszko Park" "Indiana Ave & Roosevelt Rd" "Wood St & Augusta Blvd" ...  
## $ member\_casual : chr "member" "member" "member" "member" ...  
## $ started\_at : POSIXct, format: "2020-04-26 17:45:14" "2020-04-17 17:08:54" ...  
## $ ended\_at : POSIXct, format: "2020-04-26 18:12:03" "2020-04-17 17:17:03" ...  
## $ cycling\_time : 'difftime' num 1609 489 863 732 ...  
## ..- attr(\*, "units")= chr "secs"  
## $ cycling\_time\_minutes: num 26.82 8.15 14.38 12.2 52.92 ...  
## $ start\_hour : chr "17" "17" "17" "12" ...  
## $ Month\_year : chr "2020-04" "2020-04" "2020-04" "2020-04" ...

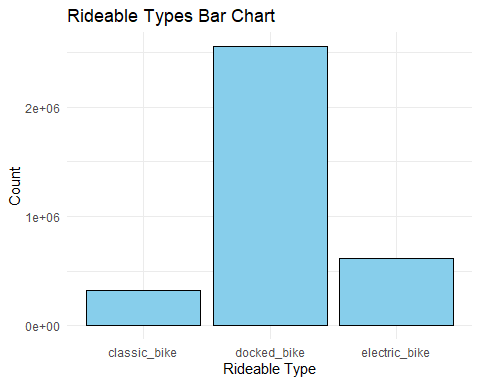
### Exploratory Data Analysis

member\_type <- table(divvy\_df$member\_casual)  
percentages <- prop.table(member\_type) \* 100  
  
# Convert data to a data frame  
pie\_data <- data.frame(member\_type = names(member\_type), percentage = percentages)  
  
# Create a pie chart using ggplot2  
ggplot(pie\_data, aes(x = "", y = percentages, fill = member\_type)) +  
 geom\_bar(stat = "identity", width = 1, color = "white") +  
 coord\_polar("y") +  
 labs(title = "Member Types Pie Chart", fill = "Member Type") +  
 theme\_minimal() +  
 theme(legend.position = "bottom") +  
 geom\_text(aes(label = sprintf("%.1f%%", percentages)), position = position\_stack(vjust = 0.5)) +  
 scale\_fill\_manual(values = c("blue", "orange"))

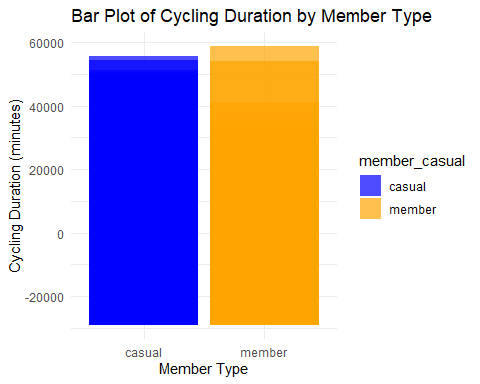
## Don't know how to automatically pick scale for object of type <table>.  
## Defaulting to continuous.

 The pie chart above shows that casuals constitute 41% while members constitute 59%.

rideable\_counts <- table(divvy\_df$rideable\_type)  
#creating a bar chart from ggplot for the rideable\_types  
ggplot(data = data.frame(rideable\_type = names(rideable\_counts), count = as.numeric(rideable\_counts)),  
 aes(x = rideable\_type, y = count)) +  
 geom\_bar(stat = "identity", fill = "skyblue", color = "black") +  
 labs(title = "Rideable Types Bar Chart", x = "Rideable Type", y = "Count") +  
 theme\_minimal()

 The barchart shows that the docked\_bikes, electric\_bikes and classic\_bikes are preferred in that order.

ggplot(data = divvy\_df, aes(x = member\_casual, y = cycling\_time\_minutes, fill = member\_casual)) +  
 geom\_bar(stat = "identity", position = "dodge", alpha = 0.7) +  
 labs(title = "Bar Plot of Cycling Duration by Member Type",  
 x = "Member Type", y = "Cycling Duration (minutes)") +  
 theme\_minimal() +  
 scale\_fill\_manual(values = c("blue", "orange"))

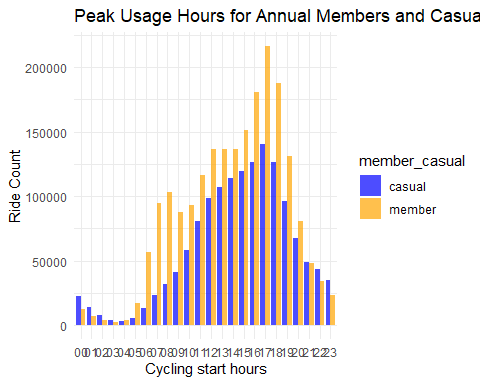


Let us now look at the peak hours that the member and casuals use the bikes

# Group by hour and ride category, count the number of rides  
peak\_hours <- divvy\_df %>%  
 group\_by(start\_hour, member\_casual) %>%  
 summarize(ride\_count = n())

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

# Create a plot to visualize the peak usage hours for each group  
ggplot(peak\_hours, aes(x = start\_hour, y = ride\_count, fill = member\_casual)) +  
 geom\_bar(stat = "identity", position = "dodge", alpha = 0.7) +  
 labs(title = "Peak Usage Hours for Annual Members and Casual Riders",  
 x = "Cycling start hours", y = "Ride Count") +  
 theme\_minimal() +  
 scale\_fill\_manual(values = c("blue", "orange"))

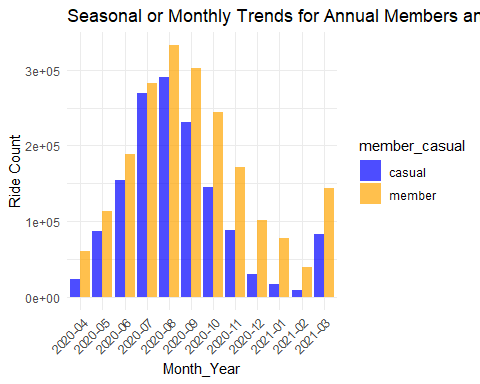


### seasonality/trends of the rides

# Group by month\_year, ride category, and count the number of rides  
monthly\_trends <- divvy\_df %>%  
 group\_by(Month\_year, member\_casual) %>%  
 summarize(ride\_count = n())

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

# Create a plot to visualize the seasonal or monthly trends for each group  
ggplot(monthly\_trends, aes(x = Month\_year, y = ride\_count, fill = member\_casual)) +  
 geom\_bar(stat = "identity", position = "dodge", alpha = 0.7) +  
 labs(title = "Seasonal or Monthly Trends for Annual Members and Casual Riders",  
 x = "Month\_Year", y = "Ride Count") +  
 theme\_minimal() +  
 scale\_x\_discrete(labels = scales::date\_format("%Y-%m")) +  
 scale\_fill\_manual(values = c("blue", "orange"))+  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))



Popular start stations by membership type

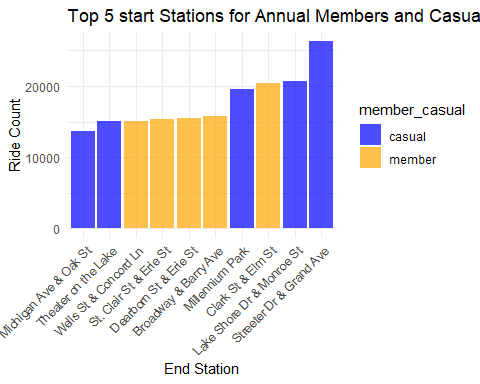
# Group by start and end stations, count the number of rides for each group  
popular\_start\_routes <- divvy\_df %>%  
 filter(!is.na(start\_station\_name) & start\_station\_name != "") %>% # remove rows with empty start\_station  
 group\_by(member\_casual, start\_station\_name) %>%  
 summarize(route\_count = n()) %>%  
 arrange(member\_casual, desc(route\_count)) %>%  
 top\_n(5, wt = route\_count)

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

popular\_start\_routes

## # A tibble: 10 × 3  
## # Groups: member\_casual [2]  
## member\_casual start\_station\_name route\_count  
## <chr> <chr> <int>  
## 1 casual Streeter Dr & Grand Ave 26263  
## 2 casual Lake Shore Dr & Monroe St 20742  
## 3 casual Millennium Park 19564  
## 4 casual Theater on the Lake 15095  
## 5 casual Michigan Ave & Oak St 13750  
## 6 member Clark St & Elm St 20397  
## 7 member Broadway & Barry Ave 15814  
## 8 member Dearborn St & Erie St 15465  
## 9 member St. Clair St & Erie St 15390  
## 10 member Wells St & Concord Ln 15147

# Create a bar plot to visualize the top 5 start stations for each group  
ggplot(popular\_start\_routes, aes(x = fct\_reorder(start\_station\_name, route\_count), y = route\_count, fill = member\_casual)) +  
 geom\_col(position = "dodge", alpha = 0.7) +  
 labs(title = "Top 5 start Stations for Annual Members and Casual Riders",  
 x = "End Station", y = "Ride Count") +  
 theme\_minimal() +  
 scale\_fill\_manual(values = c("blue", "orange")) +  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1)) # Rotate x-axis labels for better readability

 The popular start stations for casuals is Streeter Dr & Grand Ave while the popular start station for member is Clark St & Elm St.

popular end stations

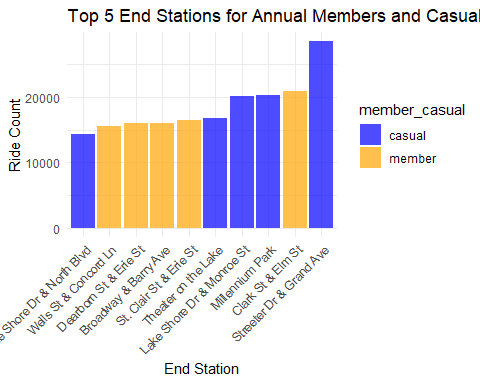
# Group by end stations, count the number of rides for each group  
popular\_end\_routes <- divvy\_df %>%  
 filter(!is.na(end\_station\_name) & end\_station\_name != "") %>% # remove rows with empty end station  
 group\_by(member\_casual, end\_station\_name) %>%  
 summarize(route\_count = n()) %>%  
 arrange(member\_casual, desc(route\_count)) %>%  
 top\_n(5, wt = route\_count)

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

popular\_end\_routes

## # A tibble: 10 × 3  
## # Groups: member\_casual [2]  
## member\_casual end\_station\_name route\_count  
## <chr> <chr> <int>  
## 1 casual Streeter Dr & Grand Ave 28471  
## 2 casual Millennium Park 20258  
## 3 casual Lake Shore Dr & Monroe St 20172  
## 4 casual Theater on the Lake 16850  
## 5 casual Lake Shore Dr & North Blvd 14321  
## 6 member Clark St & Elm St 20849  
## 7 member St. Clair St & Erie St 16540  
## 8 member Broadway & Barry Ave 16033  
## 9 member Dearborn St & Erie St 16013  
## 10 member Wells St & Concord Ln 15537

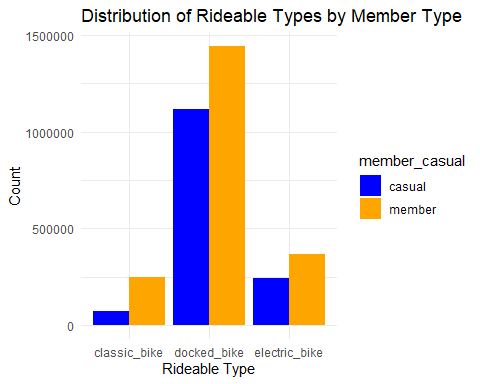
# Create a bar plot to visualize the top 5 end stations for each group  
ggplot(popular\_end\_routes, aes(x = fct\_reorder(end\_station\_name, route\_count), y = route\_count, fill = member\_casual)) +  
 geom\_col(position = "dodge", alpha = 0.7) +  
 labs(title = "Top 5 End Stations for Annual Members and Casual Riders",  
 x = "End Station", y = "Ride Count") +  
 theme\_minimal() +  
 scale\_fill\_manual(values = c("blue", "orange")) +  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1)) # Rotate x-axis labels for better readability



The bar plot shows that the top end station for casuals is Streeter Dr & Grand Ave while for members, their popular end station is Clark St & Elm St.

Checking how the various rideable types are popular among the customers

# Create a table of rideable types by member\_casual  
rideable\_by\_member <- as.data.frame(table(divvy\_df$rideable\_type, divvy\_df$member\_casual))  
  
# Rename the columns to 'rideable\_type', 'member\_casual', and 'count'  
colnames(rideable\_by\_member) <- c('rideable\_type', 'member\_casual', 'count')  
  
# Create a bar plot  
ggplot(rideable\_by\_member, aes(x = rideable\_type, y = count, fill = member\_casual)) +  
 geom\_bar(stat = "identity", position = "dodge") +  
 labs(title = "Distribution of Rideable Types by Member Type",  
 x = "Rideable Type", y = "Count") +  
 theme\_minimal() +  
 scale\_fill\_manual(values = c("blue", "orange"))

 The bar chart above shows that docked\_bike is the most popular bike among both membership types.