Cyclistic Analysis Code in R

Mital Doolab

2022-08-26

Set-up Working Environment

Install packages in R for Data Analysis

```
library(tidyverse) ## This package will be used to help us with data wrangling.
library(janitor)  ## This package will be used to help me with examining and cleaning data.
library(lubridate) ## This package will be used to convert data in our dataset into date and date-time formats.
library(tidyr) ## This package will be used for data cleaning.
library(readr) ## This package will be used to import data.
library(data.table) ## This package will be used to aggregate large sets of data.
library(dplyr) ## This package will be used for data manipulation tasks.
library(tibble) ## This package will be used to organize data.
library(skimr) ## This package will be used to create statistical summaries in data frames, tibbles, data tables, and vectors.
library(ggplot2)
library(rio) ## This package will be used to save summary tables that we may want to share in our presentation.
```

Before starting our analysis I will set-up a working directory so that I can easily access my datasets and save my work in one location. This set-up is being completed as I am using a desktop version of R.

```
getwd() ## I will use this function to determine my current working directory.
setwd("/Users/mitaldoolab/Desktop/Data Analytics/Bike Share/Cyclistic Bike Share analysis") ## I will use this function to set my working directory.
```

Importing Datasets For Data Analysis.

I will import all datasets needed for this data analysis. Also, I will assign each dataset a name which I will use to access the dataset.

```
df1 <- read.csv("2021-01-05.csv")
df2 <- read.csv("2021-02-04.csv")
df3 <- read.csv("2021-03-09.csv")
df4 <- read.csv("2021-04-08.csv")
df5 <- read.csv("2021-05-07.csv")
df6 <- read.csv("2021-06-11.csv")
df7 <- read.csv("2021-07-15.csv")
df8 <- read.csv("2021-08-14.csv")
df9 <- read.csv("2021-09-08.csv")
df10 <- read.csv("2021-10-04.csv")
df11 <- read.csv("2021-11-04.csv")
df12 <- read.csv("2021-12-08.csv")
```

To ensure that each dataset was uploaded correctly and there has been no loss of data integrity I will view each dataset using the "View()" function.

```
View(df1)
View(df2)
View(df3)
View(df4)
View(df5)
View(df6)
View(df7)
View(df8)
View(df9)
View(df10)
View(df11)
View(df12)
```

Data Cleaning

For this data analysis, I will need to blend all the datasets into one dataset. Before I start data blending I will check and confirm that each dataset has identical number of columns, column names, and data type. I will use the "compare_df_cols" function to compare all datasets.

```
compare_df_cols(df1, df2, df3, df4, df5, df6, df7, df8, df9, df10, df11, df12)
```

Now that I have confirmed that all datasets have identical number of columns, column names, and data type, I will use the "rbind" function for data blending. I will also name this new dataset "Cyclistic_data2021". Before moving to the next step, I will use the "View()" function to ensure that the data blending process was completed accurately and data integrity had been maintained.

```
Cyclistic_data2021 <- rbind(df1, df2, df3, df4, df5, df6, df7, df8, df9, df10, df11, df12)
View(Cyclistic_data2021)
```

Earlier, when I was comparing all the datasets I observed that the "ended_at" and "started_at" columns are a character data type. For my analysis, these two columns need to be in date-time format so I will change the data type from character to data-time using the "strptime" function.

```
Cyclistic_data2021$started_at = strptime(Cyclistic_data2021$started_at, "%Y-%m-%d %H:%M:%S")
Cyclistic_data2021$ended_at = strptime(Cyclistic_data2021$ended_at, "%Y-%m-%d %H:%M:%S")
```

Before moving forward with my analysis, I will confirm the "started-at" and "ended_at" columns have been correctly converted from character data type to date-time date type using the "str()" function.

```
str(Cyclistic_data2021)
```

For my data analysis I need to create a column for date, month, day of the week, and year for each ride.

```
Cyclistic_data2021$date <- as.Date(Cyclistic_data2021$started_at)
Cyclistic_data2021$month <- format(as.Date(Cyclistic_data2021$date),"%B")
Cyclistic_data2021$day <- format(as.Date(Cyclistic_data2021$date), "%d")
Cyclistic_data2021$year <- format(as.Date(Cyclistic_data2021$date), "%y")
Cyclistic_data2021$days_of_week <- format(as.Date(Cyclistic_data2021$date), "%A")
```

Before moving forward with my analysis, I will confirm that the data frame has been updated with the new columns that I added above using the "str()" function.

```
str(Cyclistic_data2021)
```

For my data analysis, I also need to create a column calculating ride_length for all trips in minutes.

```
Cyclistic_data2021$ride_length <- difftime(Cyclistic_data2021$ended_at, Cyclistic_data2021$started_at, units = "mins")
```

Before moving forward, I will confirm that my data frame has been updated with the new column "ride_length" and the data is in minutes using the "str()" function.

```
str(Cyclistic_data2021)
```

After examining the data structure I noticed that the "ride_length" column is a character data type as opposed to a numeric data type. Thus, I will convert the data type in this column from character data type to numeric data type.

```
Cyclistic_data2021$ride_length <- as.numeric(as.character(Cyclistic_data2021$ride_length))
```

Before moving forward with my analysis, I will check to confirm that the data frame has been updated and the "ride_length" is now a numeric data type using the "str()" function.

```
str(Cyclistic_data2021)
```

When examining the data I examined that there were negative "ride_lengths"; thus, I will find out how many ride_lengths are equal to, or less than 0.

```
nrow(subset(Cyclistic_data2021, ride_length <=0))
```

I will also check for any test rides that were made by the company for quality check. I will run 3 separate functions spelling "Test" 3 different ways to ensure I am checking all spelling versions for "Test" rides.

```
nrow(subset(Cyclistic_data2021, start_station_name %like% "TEST"))
nrow(subset(Cyclistic_data2021, start_station_name %like% "test"))
nrow(subset(Cyclistic_data2021, start_station_name %like% "Test"))
```

I will remove data that has a negative "ride_length" entry and data that is a "TEST" data to ensure this data does not affect my analysis. Since I am removing rows from the dataset I will create a new version of the dataset "Cyclic_data2021.v2" to maintain the data integrity of the previous dataset if I need to refer back to it.

```
Cyclistic_data2021.v2 <- Cyclistic_data2021 |>
  subset(!ride_length <= 0 &
           !start_station_name %like% "TEST")
```

Before moving forward with my analysis, I will confirm that all data with "ride_length <= 0" and "start_station_name %like%"TEST", "Test", "test"' has been removed from the new version "Cyclistic_data2021.v2".

```
nrow(subset(Cyclistic_data2021.v2, ride_length <=0))
nrow(subset(Cyclistic_data2021.v2, start_station_name %like% "TEST"))
nrow(subset(Cyclistic_data2021.v2, start_station_name %like% "test"))
nrow(subset(Cyclistic_data2021.v2, start_station_name %like% "Test"))
```

Before moving ahead with my analysis, I will confirm the new data frame has not lost its data integrity by using the "str" function and then cross checking using the "dim()" function.

```
str(Cyclistic_data2021.v2)
dim(Cyclistic_data2021.v2)
```

Data Analysis and Data Visualizations

I will start my data analysis by examining the total number of Casual Members and Annual Members. I will create a frequency table using the "table()" function and will call this table "member_type". After creating the table I will view the table using the "View()" function.

```
total_member_type <- table(Cyclistic_data2021$member_casual)
View(total_member_type)
```

Using the data from the "total_member_type" data analysis, I will create a data visualization to better understand the data.

```
total <- c(2489347, 2989749)
rider_type <- c("Casual Members", "Annual Members")
total_percent <- round(total/sum(total) * 100)
total_percent<- paste(total_percent, "%", sept = "") ## This function will add a percentage symbol after the number.
pie(total, labels = paste(total_percent), main = "Member Type", sub ="Percentage of user type", col = c("Orange", "Purple")) +
  legend("topright", cex = 0.8, legend = rider_type, fill = c("Orange", "Purple"))
```

I will examine which bike types (electric, classic, docked) were most used. I will create a frequency table using the "table()" function and will call this table "bike_type". After creating the table I will view the table using the "View()" function.

```
bike_type <- table(Cyclistic_data2021$rideable_type)
View(bike_type)
```

Using the data from the "bike_type" data analysis, I will create a data visualization to better understand the data.

```
barplot(bike_type, main = "Types of Bikes", xlab = "Type of Bike", ylab = "Number of Times Used", col = "Blue")
```

I will run the following functions below to gather descriptive analysis on "ride_length", all results will be in minutes.

```
mean(Cyclistic_data2021.v2$ride_length) ##Average ride_length in minutes.
median(Cyclistic_data2021.v2$ride_length) ## Midpoint value in the range of values in minutes.
max(Cyclistic_data2021.v2$ride_length) ## Maximum (longest) ride_length in minutes.
min(Cyclistic_data2021.v2$ride_length) ## Minimum (shortest) ride_length in minutes.
```

I will compare Casual Members and Annual Members MEAN ride length.

```
aggregate(Cyclistic_data2021.v2$ride_length ~ Cyclistic_data2021.v2$member_casual, FUN = mean)
```

I will compare Casual Members and Annual Members MEDIAN ride length.

```
aggregate(Cyclistic_data2021.v2$ride_length ~ Cyclistic_data2021.v2$member_casual, FUN = median)
```

I will compare Casual Members and Annual Member MAXIMUM ride length.

```
aggregate(Cyclistic_data2021.v2$ride_length ~ Cyclistic_data2021.v2$member_casual, FUN = max)
```

I will compare Casual Members and Annual Member MINIMUM ride length.

```
aggregate(Cyclistic_data2021.v2$ride_length ~ Cyclistic_data2021.v2$member_casual, FUN = min)
```

I will compare "member_casual" (Casual Members and Annual Members) by "ride_length". I will create a frequency table using the "table()" function and will call this table "ride_length_member_casual". After creating the table I will view the table using the "View()" function.

```
ride_length_member_casual <- Cyclistic_data2021.v2 |>
  group_by(member_casual) |> ## This function will group our results by member_casual.
  summarise(average_ride_length = mean(ride_length), median_ride_length = median(ride_length), max_ride_length = max(ride_length), min_ride_length = min(ride_length)) ## This function will calculate the mean, median, max.
View(ride_length_member_casual)
```

I will calculate the average "ride_length" for Casual Members and Annual Members per a day.

```
aggregate(Cyclistic_data2021.v2$ride_length ~ Cyclistic_data2021.v2$member_casual + Cyclistic_data2021.v2$days_of_week, FUN = mean)
```

Before I continue with my data analysis, I will order our days_of_week in order from Monday — Sunday for easy reading.

```
Cyclistic_data2021.v2$days_of_week <- ordered(Cyclistic_data2021.v2$days_of_week, levels=c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"))
```

Before moving forward I will calculate the average "ride_length" again to confirm that the"days_of_week" are in the correct order. I will also create a frequency table name "Average_ride_length" so I can refer back to this data when needed. After creating the table I will view the table using the "View()" function.

```
Average_ride_length <- aggregate(Cyclistic_data2021.v2$ride_length ~ Cyclistic_data2021.v2$member_casual + Cyclistic_data2021.v2$days_of_week, FUN = mean)
View(Average_ride_length)
```

I will also order the months in order from January — December for easy reading.

```
Cyclistic_data2021.v2$month <- ordered(Cyclistic_data2021.v2$month, levels=c("January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"))
```

I will calculate the "number_of_rides" and "average_duration" for "member_casual "(Casual Members and Annual Members) for each day of the week. After creating the table I will view the table using the"View()" function.

```
member_casual_day_of_week <- Cyclistic_data2021.v2 |>
  group_by(member_casual, days_of_week) |>  ## This function will group by member_casual and days_of_week.
  summarise(number_of_rides = n(),
            average_duration = mean (ride_length)) |> ## This function will calculate the number_of_rides and average_duration.
  arrange(member_casual, days_of_week)
View(member_casual_day_of_week)
```

Using the data from the "member_casual_day_of_week" data analysis, I will create a data visualization for "number_of_rides" and "day_of_week" filled by "member_casual" to better understand the data.

```
member_casual_day_of_week |>
  ggplot(aes(x = days_of_week,y = number_of_rides, fill = member_casual)) +
  geom_col(position = "dodge", width = 0.7) +
  scale_y_continuous(labels = scales::comma) + ## This function helps set the scaling for the y-axis and organize the numbers using commas to separate the thousands.
  labs(title = "Number of Rides Per Day", subtitle = "Number of rides taken by Casual Members or Members (Cyclistic Member) for each day of the week", x = "Day of the week", y = "Number of Rides", fill = "Member Type") +
  scale_fill_discrete(labels=c('Casual Members', 'Annual Members')) ## This function is used to change the names in the legend.
```

Using the data from the "member_casual_day_of_week" data analysis, I will create a data visualization for "average_duration" and "day_of_week" filled by "member_casual" to better understand the data.

```
member_casual_day_of_week |>
  ggplot(aes(x = days_of_week, y = average_duration, fill = member_casual)) +
  geom_col(position = "dodge", width = 0.7) +
  labs(title = "Average Duration of Rides Per Day", subtitle = "Average duration of rides taken by Casual Members or Members (Cyclistic Member) per day in minutes.", x = "Day of the week", y = "Average Duration (minutes)", fill = "Member Type") +
  scale_fill_discrete(labels=c('Casual Members', 'Annual Members')) ## This function is used to change the names in the legend.
```

I will calculate the "number_of_rides" and "average_duration" for "member_casual" (Casual Members and Annual Members) per month. After creating the table we will view the table using the "View()" function.

```
member_casual_month <- Cyclistic_data2021.v2 |>
  group_by(member_casual, month) |>  ## This function will group by member_casual and month.
  summarise(number_of_rides = n(),
            average_duration = mean(ride_length)) |> ## This function will calculate the number_of_rides and average_duration.
  arrange(member_casual, month)
View(member_casual_month)
```

Using the data from the "member_casual_month" data analysis, I will create a data visualization for "number_of_rides" and "month" filled by "member_casual" to better understand the data.

```
member_casual_month |>
  ggplot(aes(x = month,y = number_of_rides, fill = member_casual)) +
  geom_col(position = "dodge", width = 0.7) +
  scale_y_continuous(labels = scales::comma) + ## This function helps set the scaling for the y-axis and organize the numbers using commas to separate the thousands.
  labs(title = "Number of Rides Per Month", subtitle = "Number of rides taken by Casual Members and Annual Members each month.", x = "Month", y = "Number of Rides", fill = "Member Type") +
  scale_fill_discrete(labels=c('Casual Members', 'Annual Members')) ## This function is used to change the names in the legend.
```

Using the data from the "member_casual_month" data analysis, I will create a data visualization for "average_duration" and "month" filled by "member_casual" to better understand the data.

```
member_casual_month|>
  ggplot(aes(x = month, y = average_duration, fill = member_casual)) +
  geom_col(position = "dodge", width = 0.7) +
  labs(title = "Average Duration of Rides Per Month", subtitle = "Average duration of rides taken by Casual Members or Members (Cyclistic Member) each month in minutes.", x = "Month", y = "Average Duration (minutes)", fill = "Member Type") +
  scale_fill_discrete(labels=c('Casual Members', 'Annual Members')) ## This function is used to change the names in the legend.
```

I will calculate the "number_of_rides" and "average_duration" for "member_casual" (Casual Members and Annual Members) for each type of bike (electric, classic, and docked). After creating the table I will view the table using the "View()" function.

```
rideable_type_member_casual <- Cyclistic_data2021.v2 |>
  group_by(member_casual, rideable_type) |>  ## This function will group by member_casual and rideable_type.
  summarise(number_of_rides = n(),
            average_duration = mean(ride_length)) |> ## This function will calculate the number_of_rides and average_duration.
  arrange(member_casual, rideable_type)
View(rideable_type_member_casual)
```

Using the data from the "rideable_type_member_casual" data analysis, I will create a data visualization for "rideable_type" and "number_of_rides" filled by "member_casual" to better understand the data.

```
rideable_type_member_casual |>
  ggplot(aes(x = rideable_type, y = number_of_rides, fill = member_casual)) +
  geom_col(position = "dodge", width = 0.7) +
  scale_y_continuous(labels = scales::comma) + ## This function helps set the scaling for the y-axis and organize the numbers using commas to separate the thousands.
  labs(title = "Number of Rides Per Rideable Bike", subtitle = "Number of rides taken by Casual Members or Members (Cyclistic Member) per rideable bike.", x = "Bike Type", y = "Number of Rides", fill = "Member Type") +
  scale_fill_discrete(labels=c('Casual Members', 'Annual Members')) ## This function is used to change names in the legend.
```

Using the data from the "rideable_type_member_casual" data analysis, I will create a data visualization for "rideable_type" and "average_duration" filled by "member_casual" to better understand the data.

```
rideable_type_member_casual|>
  ggplot(aes(x = rideable_type, y = average_duration, fill = member_casual)) +
  geom_col(position = "dodge", width = 0.7) +
  labs(title = "Average Duration of Rides Per Bike Type", subtitle = "Average duration of rides taken by Casual Members or Members (Cyclistic Member) for each type of bike in minutes.", x = "Bike Type", y = "Average Duration (minutes)", fill = "Member Type") +
  scale_fill_discrete(labels=c('Casual Members', 'Annual Members')) ## This code is used to change the names in the legend.
```