Google Case Study

Nasra Salim

2021-10-15

Installing required libraries:

```
install.packages("tidyverse") #helps with data import and wrangling
library(tidyverse)
```

```
install.packages("lubridate") #helps with wranglig date attributes
library(lubridate)
```

```
install.packages("ggplot2") #helps with data visualization
library(ggplot2)
```

STEP 1: COLLECT DATA

Loading the twelve datasets:

```
trips_apr20 <- read_csv("202004-divvy-tripdata.csv")
trips_may20 <- read_csv("202005-divvy-tripdata.csv")
trips_jun20 <- read_csv("202006-divvy-tripdata.csv")
trips_jul20 <- read_csv("202007-divvy-tripdata.csv")
trips_aug20 <- read_csv("202008-divvy-tripdata.csv")
trips_sep20 <- read_csv("202009-divvy-tripdata.csv")
trips_oct20 <- read_csv("202010-divvy-tripdata.csv")
trips_nov20 <- read_csv("202011-divvy-tripdata.csv")
trips_dec20 <- read_csv("202012-divvy-tripdata.csv")
trips_jan21 <- read_csv("202101-divvy-tripdata.csv")
trips_feb21 <- read_csv("202102-divvy-tripdata.csv")
trips_mar21 <- read_csv("202103-divvy-tripdata.csv")
```

STEP 2: WRANGLE DATA AND COMBINE INTO A SINGLE FILE

Column names are compared to ensure the files can be joined, although they don't need to be in the same order:

```
colnames(trips_apr20)
colnames(trips_may20)
colnames(trips_jun20)
colnames(trips_jul20)
colnames(trips_aug20)
colnames(trips_sep20)
colnames(trips_oct20)
colnames(trips_nov20)
colnames(trips_dec20)
colnames(trips_jan21)
colnames(trips_feb21)
colnames(trips_mar21)
```

Inspect the dataframes and look for incongruencies, using str function to display the internal structure of the datasets:

```
str(trips_apr20)
str(trips_may20)
str(trips_jun20)
str(trips_jul20)
str(trips_aug20)
str(trips_sep20)
str(trips_oct20)
str(trips_nov20)
str(trips_dec20)
str(trips_jan21)
str(trips_feb21)
str(trips_mar21)
```

Convert end_station_id and start_station_id to character so that they can stack correctly, in the datasets where we saw this problem:

```
trips_apr20 <- mutate(trips_apr20, end_station_id = as.character
(end_station_id), start_station_id = as.character(start_station_id))

trips_may20 <- mutate(trips_may20, end_station_id = as.character
(end_station_id), start_station_id = as.character(start_station_id))

trips_jun20 <- mutate(trips_jun20, end_station_id = as.character
(end_station_id), start_station_id = as.character(start_station_id))

trips_jul20 <- mutate(trips_jul20, end_station_id = as.character
(end_station_id), start_station_id = as.character(start_station_id))

trips_aug20 <- mutate(trips_aug20, end_station_id = as.character
(end_station_id), start_station_id = as.character(start_station_id))

trips_sep20 <- mutate(trips_sep20, end_station_id = as.character
(end_station_id), start_station_id = as.character(start_station_id))

trips_oct20 <- mutate(trips_oct20, end_station_id = as.character
(end_station_id), start_station_id = as.character(start_station_id))

trips_nov20 <- mutate(trips_nov20, end_station_id = as.character
(end_station_id), start_station_id = as.character(start_station_id))
```

Stack all data frames into one big data frame:

```
all_trips <- bind_rows(trips_apr20, trips_may20, trips_jun20,
trips_jul20, trips_aug20, trips_sep20, trips_oct20, trips_nov20,
trips_dec20, trips_jan21, trips_feb21, trips_mar21)
```

Remove start_lng, start_lat, end_lng, end_lat. Columns that's not used:

```
all_trips <- all_trips %>%
  select(-c(start_lng, start_lat, end_lng, end_lat))
```

Rename columns:

```
all_trips <- all_trips %>%
  rename(trip_id = ride_id, vehicle_type = rideable_type,
  start_time = started_at, end_time = ended_at, customer_type = member_casual)
```

STEP 3: CLEAN UP AND ADD DATA TO PREPARE FOR ANALYSIS

Inspecting the new table that has been created

Showcases the column names:

```
colnames(all_trips)
```

Showcases the number of rows in data frame:

```
nrow(all_trips)
```

Showcases the dimensions of the data frame:

```
dim(all_trips)
```

Showcases the first 6 rows of data frame:

```
head(all_trips)
```

Showcases the list of columns and data types:

```
str(all_trips)
```

Showcases the statistical summary of the data. Mainly for numerics:

```
summary(all_trips)
```

Install "skimr" to be able to use skim function:

```
install.packages("skimr")
library(skimr)
```

Showcases summary of data:

```
skim(all_trips)
```

Add columns that list the date, month, day, and year of each ride. This will allow us to aggregate ride data for each month, day, or year. Before completing these operations we could only aggregate at the ride level:

```
all_trips$date <- as.Date(all_trips$start_time) #the default format: yyyy-mm-dd
all_trips$month <- format(as.Date(all_trips$date), "%m")
all_trips$day <- format(as.Date(all_trips$date), "%d")
all_trips$year <- format(as.Date(all_trips$date), "%Y")
all_trips$day_of_week <- format(as.Date(all_trips$date), "%A")
```

Add a "ride_length" calculation to all_trips (in seconds):

```
all_trips$ride_length <- difftime(all_trips$end_time,all_trips$start_time)
```

Inspect the structure of the columns:

```
str(all_trips)
```

Convert "ride_length" from Factor to numeric, so we can run calculations on the data:

```
is.factor(all_trips$ride_length)
all_trips$ride_length <- as.numeric(as.character(all_trips$ride_length))
is.numeric(all_trips$ride_length)
```

Remove "bad" data

We will create a new version of the dataframe (all_trips_cleaned) since data is being removed:

```
all_trips_cleaned <- all_trips[!(all_trips$ride_length<0),]
```

Doublecheck new dataframe:

```
skim(all_trips_cleaned)
```

STEP 4: CONDUCT DESCRIPTIVE ANALYSIS

Descriptive analysis on ride_length (all figures in seconds)

The summary() function showcases the mean, median, max, min of ride_length:

```
summary(all_trips_cleaned$ride_length)
```

Compare members and casual users:

```
aggregate(all_trips_cleaned$ride_length ~ all_trips_cleaned$customer_type, FUN = mean)
aggregate(all_trips_cleaned$ride_length ~ all_trips_cleaned$customer_type, FUN = max)
aggregate(all_trips_cleaned$ride_length ~ all_trips_cleaned$customer_type, FUN = median)
aggregate(all_trips_cleaned$ride_length ~ all_trips_cleaned$customer_type, FUN = min)
```

See the average ride time by each day for members vs casual users:

```
aggregate(all_trips_cleaned$ride_length ~ all_trips_cleaned$customer_type
+ all_trips_cleaned$day_of_week, FUN = mean)
```

Notice that the days of the week are out of order. Let's fix that:

```
all_trips_cleaned$day_of_week <- ordered(all_trips_cleaned$day_of_week,
levels=c("Sunday", "Monday", "Tuesday", "Wednesday",
"Thursday", "Friday", "Saturday"))
```

Now, let's run the average ride time by each day for members vs casual users:

```
aggregate(all_trips_cleaned$ride_length ~ all_trips_cleaned$customer_type
+ all_trips_cleaned$day_of_week, FUN = mean)
```

Analyze ridership data by Customer type and Weekday:

```
all_trips_cleaned %>%
  #creates weekday field using wday()
  mutate(weekday = wday(start_time, label = TRUE)) %>%
  #groups by customertype and weekday
  group_by(customer_type, weekday) %>%
  #calculates the number of rides and average duration
  summarise(number_of_rides = n()
             #calculates the average duration
            ,average_duration = mean(ride_length)) %>%
  #sorts
  arrange(customer_type, weekday)
```

Data Visualizations through R

Let's visualize the Number of Rides by Weekday and Customer type:

```
ggplot1 <- all_trips_cleaned %>%
  mutate(weekday = wday(start_time, label = TRUE)) %>%
  group_by(customer_type, weekday) %>%
  summarise(number_of_rides = n()
            ,average_duration = mean(ride_length)) %>%
  arrange(customer_type, weekday)  %>%
  ggplot(aes(x = weekday, y = number_of_rides, fill = customer_type)) +
  geom_col(position = "dodge")
```

Renaming variables, adding title and subtitles for ggplot1:

```
ggplot1 + ggtitle(label = "Number of Rides by Weekday and Customer Type",
            subtitle = "for better understanding") +
labs(x = "Weekday", y="Number of Rides")
```

Let's create a visualization for average duration:

```
ggplot2 <- all_trips_cleaned %>%
  mutate(weekday = wday(start_time, label = TRUE)) %>%
  group_by(customer_type, weekday) %>%
  summarise(number_of_rides = n()
            ,average_duration = mean(ride_length)) %>%
  arrange(customer_type, weekday)  %>%
  ggplot(aes(x = weekday, y = average_duration, fill = customer_type)) +
  geom_col(position = "dodge")
```

Renaming variables, adding title and subtitles for ggplot2:

```
ggplot1 + ggtitle(label = "Average duration by Weekday and Customer Type",
                  subtitle = "for better understanding") +
  labs(x = "Weekday", y="Average durations")
```

STEP 5: EXPORT FILE FOR FURTHER ANALYSIS

Export to use in Tableau:

```
write.csv(all_trips_cleaned, file = '~/Desktop/alltrips.csv')
```