Bellabeat Case Study – Google Data Analytics Course

Cameron Kemske

12/8/2021

This R Markdown file was created to showcase the R code that was used to perform the cleaning, analysis, and visualization process of this case study's data set. This file is complimentary to the "Bellabeat Presentation" that will be used to showcase additional and final recommendations for this case study's scenario. This R Markdown file should be seen as a notebook for the data project and give insights to my thought process and ability to execute different R and SQL functions.

Scenario

I am a Junior Data Analyst at the company, "Bellabeat." The company focuses on women's health technologies with similarities towards Apple Fitness and Fitbit. The CMO, Urska Srson, tasked me with analyzing FitBit data in order to find any recommendations for marketing efforts that can be used for Bellabeat's smartphone application.

Downloading All Needed Packages

To properly prepare for the cleaning and analysis process, several key packages have to be installed. Tidyverse is the basic package that allows for easy data cleaning and manipulation features. Skimr helps with highlighting and summarizing data. Janitor is for data cleaning. Sqldf allows R to use SQL syntax and command codes for easier manipulation of data. After all packages are downloaded, installation of those packages comes next. Now, you are set to begin cleaning the data sets.

```
install.packages("tidyverse")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.1'
## (as 'lib' is unspecified)
```

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

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.1'
## (as 'lib' is unspecified)
```

```
install.packages("janitor")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.1'
## (as 'lib' is unspecified)
```

```
install.packages("sqldf")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.1'
## (as 'lib' is unspecified)
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
```

```
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(skimr)
library(janitor)
```

```
##
## Attaching package: 'janitor'

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

```
library(sqldf)
```

```
## Loading required package: gsubfn

## Loading required package: proto

## Warning in fun(libname, pkgname): couldn't connect to display ":0"

## Loading required package: RSQLite
```

```
library(dplyr)
```

Cleaning Process

Using the read() function, you can import the CSV files that were used for this analysis. During this process, it is easy to rename the data sets to your specifications by using the assignment function "<-".

```
daily_activities <- read.csv("dailyActivity_merged.csv")
daily_calories <- read.csv("dailyCalories_merged.csv")
daily_intensities <- read.csv("dailyIntensities_merged.csv")
daily_sleep <- read.csv("sleepDay_merged.csv")
weight_log <- read.csv("weightLogInfo_merged.csv")
```

Once the files have been uploaded and renamed in the directory, you can use the head() function to see a preview of the variable names and the first six observations recorded for each of the variables.

```
head(daily_activities)
```

```
##           Id ActivityDate TotalSteps TotalDistance TrackerDistance
## 1 1503960366    4/12/2016      13162          8.50            8.50
## 2 1503960366    4/13/2016      10735          6.97            6.97
## 3 1503960366    4/14/2016      10460          6.74            6.74
## 4 1503960366    4/15/2016       9762          6.28            6.28
## 5 1503960366    4/16/2016      12669          8.16            8.16
## 6 1503960366    4/17/2016       9705          6.48            6.48
##   LoggedActivitiesDistance VeryActiveDistance ModeratelyActiveDistance
## 1                        0               1.88                     0.55
## 2                        0               1.57                     0.69
## 3                        0               2.44                     0.40
## 4                        0               2.14                     1.26
## 5                        0               2.71                     0.41
```

```
## 6                       0             3.19                     0.78
##   LightActiveDistance SedentaryActiveDistance VeryActiveMinutes
## 1                6.06                       0                25
## 2                4.71                       0                21
## 3                3.91                       0                30
## 4                2.83                       0                29
## 5                5.04                       0                36
## 6                2.51                       0                38
##   FairlyActiveMinutes LightlyActiveMinutes SedentaryMinutes Calories
## 1                  13                  328              728     1985
## 2                  19                  217              776     1797
## 3                  11                  181             1218     1776
## 4                  34                  209              726     1745
## 5                  10                  221              773     1863
## 6                  20                  164              539     1728
```

```
head(daily_calories)
```

```
##           Id ActivityDay Calories
## 1 1503960366   4/12/2016     1985
## 2 1503960366   4/13/2016     1797
## 3 1503960366   4/14/2016     1776
## 4 1503960366   4/15/2016     1745
## 5 1503960366   4/16/2016     1863
## 6 1503960366   4/17/2016     1728
```

The count() function was used to determine how many observations were in each of the data sets. This type of function acts as a way of confirming if there is the same number of observations between the three data sets that log the activities, calories and intensities of the users.

```
count(daily_activities)
```

```
##     n
## 1 940
```

```
count(daily_calories)
```

```
##     n
## 1 940
```

```
count(daily_intensities)
```

```
##     n
## 1 940
```

Creating a data frame that highlights the ID, ActivityDate, and Calories variables is created to perform a SQL check to see if the data sets have any of the same values for those specific variables.

```
daily_activities2 <- daily_activities %>%
  select(Id, ActivityDate, Calories)
```

Using the sqldf() function, you can use SQL syntax to perform a data query from inside R. Using SQL commands within R can help with the data cleaning process, as you can use multiple tools within one software, allowing for faster cleaning and analysis. The 'SELECT' function allows the returning of all values within the specified data set from the 'FROM' function. The 'INTERSECT' function joins the two specified data sets together that the query results can be compared against each other. The count() function confirms that the daily_calories data set contains the same observations as the daily_activities data set.

```
sql_check1 <- sqldf('SELECT *
                    FROM daily_activities2
                    INTERSECT
                    SELECT *
                    FROM daily_calories')
count(sql_check1)
```

```
##     n
## 1 940
```

Below is the second SQL check for the daily_intensities data set and the creation of the data frame to be used in the SQL query.

```
daily_activities3 <- daily_activities %>%
  select(Id,
         ActivityDate,
         SedentaryMinutes,
         LightlyActiveMinutes,
         FairlyActiveMinutes,
         VeryActiveMinutes,
         SedentaryActiveDistance,
         LightActiveDistance,
         ModeratelyActiveDistance,
         VeryActiveDistance)
```

Cross referencing the daily_activities data set with the daily_intensities data set to confirm that the data is consistent. The daily_activities data set contains all data from the daily_calories and daily_intensities data sets. We will be using the daily_activities data set for the remainder of the analysis.

```
sql_check2 <- sqldf('SELECT * FROM daily_activities3 INTERSECT SELECT * FROM daily_intensities')
count(sql_check2)
```

```
##     n
## 1 940
```

The final part of the cleaning/manipulation process is creating a new variable for analysis. Converting the dates of each recorded observation into their corresponding 'day of the week' can be used to highlight the concentration of different variables occurring on certain days. This new variable can derive insights as to which days were most active versus which days were most inactive. Marketing efforts can be recommended based off those insights.

```
daily_activities4 <- daily_activities
daily_activities4$dayofweek <- weekdays(daily_activities4$ActivityDate)
head(daily_activities4)
```

```
##           Id ActivityDate TotalSteps TotalDistance TrackerDistance
## 1 1503960366    4/12/2016      13162          8.50            8.50
## 2 1503960366    4/13/2016      10735          6.97            6.97
## 3 1503960366    4/14/2016      10460          6.74            6.74
## 4 1503960366    4/15/2016       9762          6.28            6.28
## 5 1503960366    4/16/2016      12669          8.16            8.16
## 6 1503960366    4/17/2016       9705          6.48            6.48
##   LoggedActivitiesDistance VeryActiveDistance ModeratelyActiveDistance
## 1                        0               1.88                     0.55
## 2                        0               1.57                     0.69
## 3                        0               2.44                     0.40
## 4                        0               2.14                     1.26
```

```
## 5                        0              2.71                     0.41
## 6                        0              3.19                     0.78
##   LightActiveDistance SedentaryActiveDistance VeryActiveMinutes
## 1                6.06                       0                25
## 2                4.71                       0                21
## 3                3.91                       0                30
## 4                2.83                       0                29
## 5                5.04                       0                36
## 6                2.51                       0                38
##   FairlyActiveMinutes LightlyActiveMinutes SedentaryMinutes Calories dayofweek
## 1                  13                  328              728     1985       Tue
## 2                  19                  217              776     1797       Wed
## 3                  11                  181             1218     1776       Thu
## 4                  34                  209              726     1745       Fri
## 5                  10                  221              773     1863       Sat
## 6                  20                  164              539     1728       Sun
```

Analysis Process

To begin, let's look into how many distinct IDs were recording their activity in the FitBit application and into the data sets of daily_activities4, weight_log, and sleep_log.

```
n_distinct(daily_activities4$Id)
```

```
## [1] 33
```

```
n_distinct(daily_sleep$Id)
```

```
## [1] 24
```

```
n_distinct(weight_log$Id)
```

```
## [1] 8
```

```
count(daily_activities4)
```

```
##     n
## 1 940
```

```
count(daily_sleep)
```

```
##     n
## 1 413
```

```
count(weight_log)
```

```
##    n
## 1 67
```

From the results of the n_distinct() functions for the data sets, it was established that there were progressively less unique IDs recording data starting from the daily_activities to the weight_log. To see the magnitude, you can find the percentage change from each data set to the next.

```
(24/33) - 1
```

```
## [1] -0.2727273
```

```
(8/24) - 1
```

```
## [1] -0.6666667
```

The results show a **27% decrease** of unique IDs recording data from the daily_activities data set to the daily_sleep data set. Then, a **67% decrease** in unique IDs recording data from the daily_sleep data set to the weight_log data set. The analysis' results were interesting to see the magnitude of dropping user participation from the recording activity. Finding a way to increase participation across all facets of the data logging process for the user would be useful to collect more data and make the application more engaging.

The next analysis performed was the basic statistical summary of several key variables within the daily_activities data set. This can give a brief overview into the key characteristics of the data and show you the high-level perspective of your data set.

```r
daily_activities4 %>%
  select(TotalSteps,
         TotalDistance,
         SedentaryMinutes,
         VeryActiveMinutes) %>%
  summary()
```

```
##    TotalSteps    TotalDistance    SedentaryMinutes VeryActiveMinutes
##  Min.   :    0   Min.   : 0.000   Min.   :   0.0   Min.   :  0.00
##  1st Qu.: 3790   1st Qu.: 2.620   1st Qu.: 729.8   1st Qu.:  0.00
##  Median : 7406   Median : 5.245   Median :1057.5   Median :  4.00
##  Mean   : 7638   Mean   : 5.490   Mean   : 991.2   Mean   : 21.16
##  3rd Qu.:10727   3rd Qu.: 7.713   3rd Qu.:1229.5   3rd Qu.: 32.00
##  Max.   :36019   Max.   :28.030   Max.   :1440.0   Max.   :210.00
```

The 'VeryActiveMinutes' variable statistics stood out because of the divergence of the mean and median values. We see the median as 4 minutes and the mean of 21.16 minutes. A hypothesize can form that several unique IDs are recording high activity levels which brings up the average of this variable. It would be inconclusive to say that the users are engaging in high-level physical exercise. This could lead to potential efforts to encourage users to partake in active exercise to increase the median values.

Final part of the analysis was to create a data frame that incorporates the mean values of "Calories" for each day of the week. This type of analysis would be helpful to see when users are most and least active on average. Marketing strategies could be put into place to encourage more activity on the least active day.

```r
day_of_week_mean <- sqldf('SELECT Id, Calories, dayofweek FROM daily_activities4')
```

```r
day_of_week_mean2 <- day_of_week_mean %>%
  group_by(dayofweek) %>%
    mutate(mean_by_day = mean(Calories))
```

The code above, created the data frame that outlined each day's average calories burned. The data frame can then be used to visualize the mean calories burned in a way that is sufficient for sharing to stakeholders.

Another form of analysis is to create a data frame to record the averages of each day of the week. The numeric values came from the data frame: "day_of_week2." they are assigned to a vector that then can be arranged visually in a chart or graph.

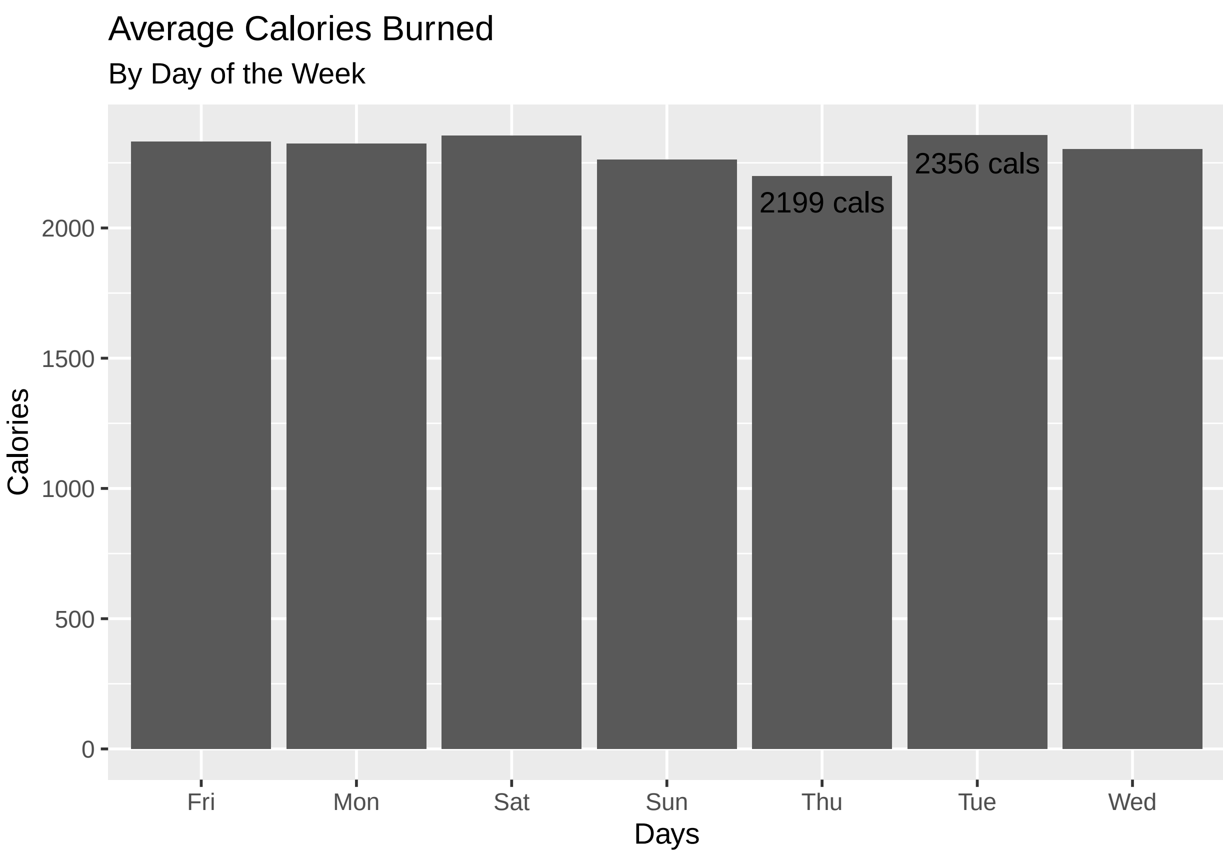
```r
Avg_cals_per_day <- data.frame (x = c(2199.571, 2263, 2302.62, 2324.208, 2331.786, 2354.968, 2356.073),
y = c("Thu", "Sun", "Wed", "Mon", "Fri", "Sat", "Tue"))
```

Data Visualization

In order to better represent the findings and insights found during analysis, you can create three relevant data visualizations. Each visualization reveals an interesting trend found and could be used for future marketing efforts for Bellabeat.

Bar Chart - Average Calories Burned Tuesday had the highest average calories burned for the users, while Sunday saw the lowest average calories burned. The difference between the two ends of the range was 93 calories which is almost negligible. However, this could be an opportunity to test new alerts or reminders for users on Sunday and even Tuesday. The data collected from this would see how much of an effect these alerts had on increasing calories burned.

```
ggplot(Avg_cals_per_day, aes(x = y , y = x)) +
  geom_bar(stat = 'identity') +
  labs(title = 'Average Calories Burned', subtitle = 'By Day of the Week', x = "Days", y = "Calories") +
  annotate('text', x = 'Tue', y = 2250, label = '2356 cals') +
  annotate('text', x = "Thu", y = 2100, label = '2199 cals')
```



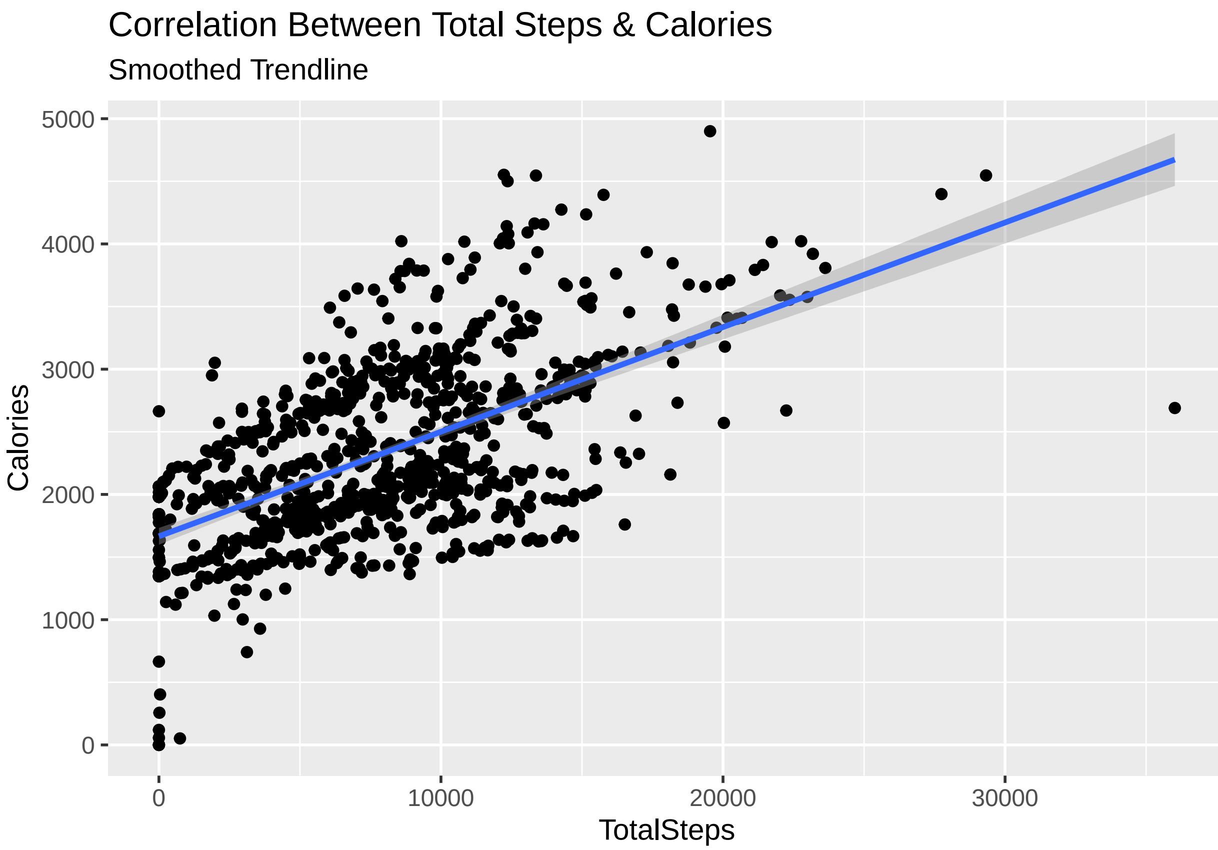
Scatterplot - Spread of Calories This scatter plot was created to see which of the days had the most variation of user records of calories burned with a similar purpose of the bar chart described above. Thursday saw the most variation in calories burned. Alerts or encouraging marketing materials could be sent to the application's users to encourage consistent exercise/activity on Thursday to promote higher burned calories amount.

```
ggplot(daily_activities4, aes(x = Calories, y = dayofweek)) +
  geom_point() +
  labs(title = 'Correlation Between Calories Burned & Days',
       subtitle = 'Each Logged "Calories" Value Displayed')
```

Trendline - Active Minutes to Calories The scatter plot was used in conjunction with a trend line to showcase the positive relationship between "TotalSteps" to "Calories" variables. As the user logged more total steps for the day, they saw an increase in their calories burned. The upward sloped trend line shows a positive relationship between the two variables. This cna conclude that the simple act of increasing your toal steps per day can lead to increases in total calories burned.

```
ggplot(daily_activities4, aes(x = TotalSteps, y = Calories)) +
  geom_point() +
  stat_smooth(method = lm) +
  labs(title = 'Correlation Between Total Steps & Calories',
       subtitle = 'Smoothed Trendline')
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Sources Jeremiah Hartsock made a similar R Markdown file that was used in this project to give inspiration and guidance during this data analysis process. Here is the link to their published R Markdown file: (https://rpubs.com/jerethar96/768783)