Bellabeat Fitness Device Analysis - Data Cleaning

Clarissa McCarthy

2024-03-08

Introduction

This file documents my cleaning process for the data used in my Bellabeat Fitness Data Analysis project. This is not an exhaustive documentation of all the code I used to clean my data, but rather it is an outline of the general steps I took with important findings highlighted. For an in-depth look at my cleaning process, please check out my `cleaning_data.R` file.

Installing Libraries

For this analysis, I used the tidyverse library to clean my data.

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

Uploading Data Sets

My analysis consists of three data sets:

- FitBit Fitness Tracker Data by Mobius
- Fitness Consumer Survey Data by Harshita Aswani
- Fitness Analysis by Nithilaa

The FitBit Fitness Tracker Data consists of 18 CSV files, of which I uploaded the 7 files I intend to use in my analysis. The Fitness Consumer Survey Data and the Fitness Analysis data consist of 1 CSV file each.

```
activity <- read.csv("FitBit Fitness Tracker Data/dailyActivity_merged.csv")
calories <- read.csv("FitBit Fitness Tracker Data/dailyCalories_merged.csv")
intensities <- read.csv("FitBit Fitness Tracker Data/dailyIntensities_merged.csv")
steps <- read.csv("FitBit Fitness Tracker Data/dailySteps_merged.csv")
heartrate <- read.csv("FitBit Fitness Tracker Data/heartrate_seconds_merged.csv")
sleep <- read.csv("FitBit Fitness Tracker Data/sleepDay_merged.csv")
weight <- read.csv("FitBit Fitness Tracker Data/weightLogInfo_merged.csv")

device_survey <- read.csv("Fitness Consumer Survey Data/survey 605.csv")
exercise_survey <- read.csv("Fitness Analysis Data/fitness analysis.csv")
```

Identifying Missing Data

Prior to cleaning the data in R, I made initial SQL queries to evaluate the credibility of the sets. From these queries, I learned which files contained missing data. I confirmed the presence/absence of this missing data in R using the function `sum(is.na(data-frame))`.

FitBit Fitness Tracker Data - Missing Data I checked each data frame in this set for missing values. However, I have only included the code for checking the weight data frame, since it is the only data frame to contain missing data.

```
sum(is.na(weight))
```

```
## [1] 65
```

```
# Checking what columns in weight have nulls
names(weight)
```

```
## [1] "Id"             "Date"           "WeightKg"       "WeightPounds"
## [5] "Fat"            "BMI"            "IsManualReport" "LogId"
```

```
sum(is.na(weight$Fat))
```

```
## [1] 65
```

All of the null values were in the "Fat" column in the weight data frame. Since I did not intend to use this column in my analysis, I removed it.

```
weight <- subset(weight, select = -c(Fat))
names(weight)
```

```
## [1] "Id"             "Date"           "WeightKg"       "WeightPounds"
## [5] "BMI"            "IsManualReport" "LogId"
```

Fitness Consumer Survey Data & Fitness Analysis Data - Missing Data From my SQL queries, I knew there was no missing data in the Fitness Consumer Survey Data or the Fitness Analysis data. For completeness, I confirmed this in R.

```
sum(is.na(device_survey))
```

```
## [1] 0
```

```
sum(is.na(exercise_survey))
```

```
## [1] 0
```

Identifying Duplicate Data

In my initial SQL queries, I also identified which files contained duplicate data. I have therefore omitted checking every file for duplicates in R, and instead only included the files I already knew contained duplicates.

FitBit Fitness Tracker Data - Duplicates The only data from the FitBit Fitness Tracker data set that contained duplicates was the sleep data. I identified these duplicate rows by using the dplyr `duplicated()` function.

```
# Finding duplicate entries
dupe <- sleep[,c('Id','SleepDay','TotalSleepRecords','TotalMinutesAsleep','TotalTimeInBed')]
sleep_duplicates <- sleep[duplicated(dupe) | duplicated(dupe, fromLast = TRUE),]
head(sleep_duplicates)
```

```
##             Id              SleepDay TotalSleepRecords TotalMinutesAsleep
## 161 4388161847  5/5/2016 12:00:00 AM                 1                471
## 162 4388161847  5/5/2016 12:00:00 AM                 1                471
## 223 4702921684  5/7/2016 12:00:00 AM                 1                520
## 224 4702921684  5/7/2016 12:00:00 AM                 1                520
## 380 8378563200 4/25/2016 12:00:00 AM                 1                388
## 381 8378563200 4/25/2016 12:00:00 AM                 1                388
##     TotalTimeInBed
## 161            495
## 162            495
## 223            543
## 224            543
## 380            402
## 381            402
```

Since these entries are exact duplicates, I removed them from the data frame. I then confirmed that they were properly removed and no duplicates remained.

```
# Removing duplicate entries
sleep <- sleep %>% distinct()
```

```
# Confirming no duplicates remain
nrow(sleep[duplicated(sleep),])
```

```
## [1] 0
```

Fitness Consumer Survey Data - Duplicates From my initial SQL queries, I knew there were no duplicate entries in this data set.

Fitness Analysis Data - Duplicates The Fitness Analysis data did not contain exactly duplicate rows, largely due to the timestamp associated with each survey response. However, it did contain duplicate names. To determine whether or not two entries with the same name were submitted by the same person, I created a data frame containing these potentially duplicate entries and manually compared the data in each response. My criteria for deeming two entries as being from the same person was if the name, age, and gender values were the same and the values in the other columns were similar. If such a match was found, the older entry was removed from the data set and the newer one remained.

```
# Identifying duplicate entries in fitness analysis data
dupe <- exercise_survey[,c('Your.name.')]
exercise_duplicates <- exercise_survey[duplicated(dupe) | duplicated(dupe, fromLast = TRUE),]
head(exercise_duplicates[,1:4])
```

```
##                          Timestamp Your.name. Your.gender.    Your.age.
## 45   2019/07/04 8:49:20 AM GMT+5:30  Vaishnavi       Female     19 to 25
## 50   2019/07/04 8:52:30 AM GMT+5:30     Snehaa       Female     19 to 25
## 91   2019/07/04 9:39:57 AM GMT+5:30  Vaishnavi       Female 40 and above
## 94   2019/07/04 9:43:56 AM GMT+5:30   Bhargavi       Female     19 to 25
## 123 2019/07/04 10:39:09 AM GMT+5:30   Bhargavi       Female     19 to 25
## 141 2019/07/04 11:22:46 AM GMT+5:30    Rajesh          Male 40 and above
```

```
# Removing entries from same person
dupe_indices <- c(147, 267, 282, 330, 348, 389, 442, 472, 491, 505, 538, 539, 540)
exercise_original <- exercise_survey
exercise_survey <- exercise_survey %>% filter(!row_number() %in% dupe_indices)
```

To verify whether the duplicate entries had been successfully removed, I first calculated the number of rows in the original data frame minus the number of rows I wanted to remove. This was the number of rows I

expected to have in the final data frame. I then compared this number to the actual number of rows in the final data frame. Since these numbers were equal, I knew the duplicate removal was a success.

```
# Confirming no duplicates remain
expected_rows <- nrow(exercise_original) - length(dupe_indices) # expected number of remaining rows
actual_rows <- nrow(exercise_survey) # actual number of remaining rows
expected_rows
```

```
## [1] 532
```

```
actual_rows
```

```
## [1] 532
```

Removing Unnecessary Columns

I kept most of the columns in the data sets as I expected them to be useful in my analysis. I removed 4 columns across the FitBit Fitness Tracker Data set since their values were largely the same across all entries and thus wouldn't likely produce interesting trends. I also removed 5 columns from the Fitness Analysis data. Some of these columns dealt with dietary data, which I did not intend to analyze in this analysis. The other columns I removed also contained data that I felt would not produce the most useful analyses in this project.

```
activity <- subset(activity, select = -c(LoggedActivitiesDistance,SedentaryActiveDistance))
intensities <- subset(intensities, select = -c(SedentaryActiveDistance))
sleep <- subset(sleep, select = -c(TotalSleepRecords))
names(exercise_survey)
```

```
##  [1] "Timestamp"
##  [2] "Your.name."
##  [3] "Your.gender."
##  [4] "Your.age."
##  [5] "How.important.is.exercise.to.you.."
##  [6] "How.do.you.describe.your.current.level.of.fitness.."
##  [7] "How.often.do.you.exercise."
##  [8] "What.barriers..if.any..prevent.you.from.exercising.more.regularly.............Please.select.al
##  [9] "What.form.s..of.exercise.do.you.currently.participate.in.........................Please.sele
## [10] "Do.you.exercise._____________.."
## [11] "What.time.if.the.day.do.you.prefer.to.exercise."
## [12] "How.long.do.you.spend.exercising.per.day.."
## [13] "Would.you.say.you.eat.a.healthy.balanced.diet.."
## [14] "What.prevents.you.from.eating.a.healthy.balanced.diet..If.any.........................Please
## [15] "How.healthy.do.you.consider.yourself."
## [16] "Have.you.ever.recommended.your.friends.to.follow.a.fitness.routine."
## [17] "Have.you.ever.purchased.a.fitness.equipment."
## [18] "What.motivates.you.to.exercise...........Please.select.all.that.applies.."
```

```
exercise_survey <- exercise_survey[-c(10,11,13,14)]
exercise_survey <- exercise_survey[-c(12)]
```

Renaming Fitness Analysis Columns

The default naming convention for the columns in the Fitness Analysis data was lengthy and difficult to read. Thus, I renamed them to understand their purpose quicker and to make it easier to type them in R.

```r
exercise_survey <- exercise_survey %>%
  rename(
    Name = Your.name.,
    Gender = Your.gender.,
    Age = Your.age.,
    ExerciseImportance = How.important.is.exercise.to.you..,
    FitnessLevel = How.do.you.describe.your.current.level.of.fitness..,
    ExerciseFrequency = How.often.do.you.exercise.,
    ExerciseBarriers = What.barriers..if.any..prevent.you.from.exercising.more.regularly.....Pl
    ExerciseForms = What.form.s..of.exercise.do.you.currently.participate.in.....
    ExerciseLength = How.long.do.you.spend.exercising.per.day..,
    HealthLevel = How.healthy.do.you.consider.yourself.,
    EquipmentPurchase = Have.you.ever.purchased.a.fitness.equipment.,
    ExerciseMotivation = What.motivates.you.to.exercise.....Please.select.all.that.applies..
  )
```

Checking For Outliers

Lastly, I used the `min()` and `max()` functions to check for outliers in the numerical data. I coupled the use of these functions with manually sorting the columns and comparing the lowest/highest values with the next lowest/highest values. The only outlier I found was in the FitBit Fitness Tracker weight data. However, this value was still within the bounds of being a realistic weight, so I kept it in the data frame.
I have omitted the bulk of these commands here since they are long and had no affect on transforming the data. If you would like to check these commands, please check my `cleaning_data.R` file.

```r
min(exercise_survey$ExerciseImportance)
```

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

```r
max(exercise_survey$ExerciseImportance)
```

```
## [1] 5
```

```r
min(heartrate$Value)
```

```
## [1] 36
```

```r
max(heartrate$Value)
```

```
## [1] 203
```

```r
min(weight$WeightPounds)
```

```
## [1] 115.9631
```

```r
max(weight$WeightPounds)
```

```
## [1] 294.3171
```

```r
# etc...
```

Conclusion

Through this process, I effectively identified missing and duplicate values in my data, removed unnecessary columns in the data frames, renamed columns for easier use, and checked my data for outliers. After completing this process, all my data has been cleaned and is ready for analysis.