Bellabeat Case Study

Pilar Alcazar

2022-09-14

Business Task: Determine how consumers use the smart devices they already have, and identify ways to apply this information to the marketing of Bellabeat products.

We install and library all necessary packages, then import the datasets.

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

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.2'
## (as 'lib' is unspecified)
```

```
install.packages("tidyr")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.2'
## (as 'lib' is unspecified)
```

```
install.packages("dplyr")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.2'
## (as 'lib' is unspecified)
```

```
install.packages("lubridate")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.2'
## (as 'lib' is unspecified)
```

```
install.packages("ggplot2")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.2'
## (as 'lib' is unspecified)
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
```

```
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.7      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.0
## v readr   2.1.2      v forcats 0.5.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tidyr)
library(dplyr)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

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

```
library(ggplot2)
dailyActivity_merged <- read.csv("Fitabase Data 4.12.16-5.12.16/dailyActivity_merged.csv")
```

Repeat the last line for all of the tables in the dataset.

Now that our data is loaded, we can begin to explore it. To get to know our data, we will use the `head()`, `glimpse()`, and `str()` functions.

Some things to note:

- Our data is stored in 18 tables in long format. There appears to be the presence of a primary key called "Id" that can be used to connect the various tables.
- Since this data is gathered from only 30 users, there is certainly concern regarding bias and credibility issues. As the sample size is so small, the data may be affected by a sampling bias and not be representative of the population as a whole. However this is the data we've been asked to analyze, so we will keep in mind its limitations when we get to our analysis. Our data is original and cited, however it is not current as it is from 2016. We also may have issues regarding its reliability and comprehensiveness as the pool of users is so small.
- The data is public and the users consented to the anonymous submission of their data, so we don't have to worry about licensing, privacy, or security. The data is readily accessible online to anyone who wishes to access it.
- We run tests to check certain parts of the data, such as verifying the amount of distinct user Id's or the period of time the data was collected over. This can be done in many ways, however I chose to use a mixture of pivot tables, SQL queries, and R programming.

Processing the data

I will use R for the data cleaning and processing. Let's save our datasets under new names to make our analysis easier, and to keep a copy of our original data:

```
activity <- read.csv("/cloud/project/Fitabase Data 4.12.16-5.12.16/dailyActivity_merged.csv")
calories <- read.csv("/cloud/project/Fitabase Data 4.12.16-5.12.16/hourlyCalories_merged.csv")
intensities <- read.csv("/cloud/project/Fitabase Data 4.12.16-5.12.16/hourlyIntensities_merged.csv")
sleep <- read.csv("/cloud/project/Fitabase Data 4.12.16-5.12.16/sleepDay_merged.csv")
weight <- read.csv("/cloud/project/Fitabase Data 4.12.16-5.12.16/weightLogInfo_merged.csv")
```

Now let's ensure our data imported correctly by using the `head()` and `str()` functions.

```
str(activity)
```

```
## 'data.frame':    940 obs. of  15 variables:
##  $ Id                       : num  1.5e+09 1.5e+09 1.5e+09 1.5e+09 1.5e+09 ...
##  $ ActivityDate             : chr  "4/12/2016" "4/13/2016" "4/14/2016" "4/15/2016" ...
##  $ TotalSteps               : int  13162 10735 10460 9762 12669 9705 13019 15506 10544 9819 ...
##  $ TotalDistance            : num  8.5 6.97 6.74 6.28 8.16 ...
##  $ TrackerDistance          : num  8.5 6.97 6.74 6.28 8.16 ...
##  $ LoggedActivitiesDistance: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ VeryActiveDistance       : num  1.88 1.57 2.44 2.14 2.71 ...
##  $ ModeratelyActiveDistance: num  0.55 0.69 0.4 1.26 0.41 ...
##  $ LightActiveDistance      : num  6.06 4.71 3.91 2.83 5.04 ...
##  $ SedentaryActiveDistance : num  0 0 0 0 0 0 0 0 0 0 ...
```

```
##  $ VeryActiveMinutes       : int  25 21 30 29 36 38 42 50 28 19 ...
##  $ FairlyActiveMinutes     : int  13 19 11 34 10 20 16 31 12 8 ...
##  $ LightlyActiveMinutes    : int  328 217 181 209 221 164 233 264 205 211 ...
##  $ SedentaryMinutes        : int  728 776 1218 726 773 539 1149 775 818 838 ...
##  $ Calories                : int  1985 1797 1776 1745 1863 1728 1921 2035 1786 1775 ...
```

```
head(activity)
```

```
##           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
```

Most of the time data is incorrectly recognized as string data. We can convert it to a date or date-time class using the following function:

```
activity$ActivityDate <- as.POSIXct(activity$ActivityDate, format="%m/%d/%Y", tz=Sys.timezone())
calories$ActivityHour <- as.POSIXct(calories$ActivityHour, format="%m/%d/%Y %H:%M:%S", tz=Sys.timezone()
intensities$ActivityHour <- as.POSIXct(intensities$ActivityHour, format="%m/%d/%Y %I:%M:%S %p", tz=Sys.t
sleep$SleepDay <- as.POSIXct(sleep$SleepDay, format="%m/%d/%Y %H:%M:%S", tz=Sys.timezone())
weight$Date <- as.POSIXct(weight$Date, format="%m/%d/%Y %H:%M:%S", tz=Sys.timezone())
```

Now we will split the data into date and time separately to make it easier to compare between different datasets.

```
activity$Date <- as.Date(activity$ActivityDate)
calories$Date <- as.Date(calories$ActivityHour)
calories$Time <- format(calories$ActivityHour, format="%H:%M:%S")
intensities$Date <- as.Date(intensities$ActivityHour)
intensities$Time <- format(intensities$ActivityHour, format="%H:%M:%S")
sleep$Date <- as.Date(sleep$SleepDay)
sleep$Time <- format(sleep$SleepDay, format="%H:%M:%S")
```

```
weight$Date <- as.Date(weight$Date)
```

Now that our data is formatted properly, we can begin our analysis.

Analyzing the data

Let's check to see if our user ID's and the range of time of the data matches.

```
n_distinct(activity$Id)
```

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

```
n_distinct(calories$Id)
```

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

```
n_distinct(intensities$Id)
```

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

```
n_distinct(sleep$Id)
```

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

```
n_distinct(weight$Id)
```

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

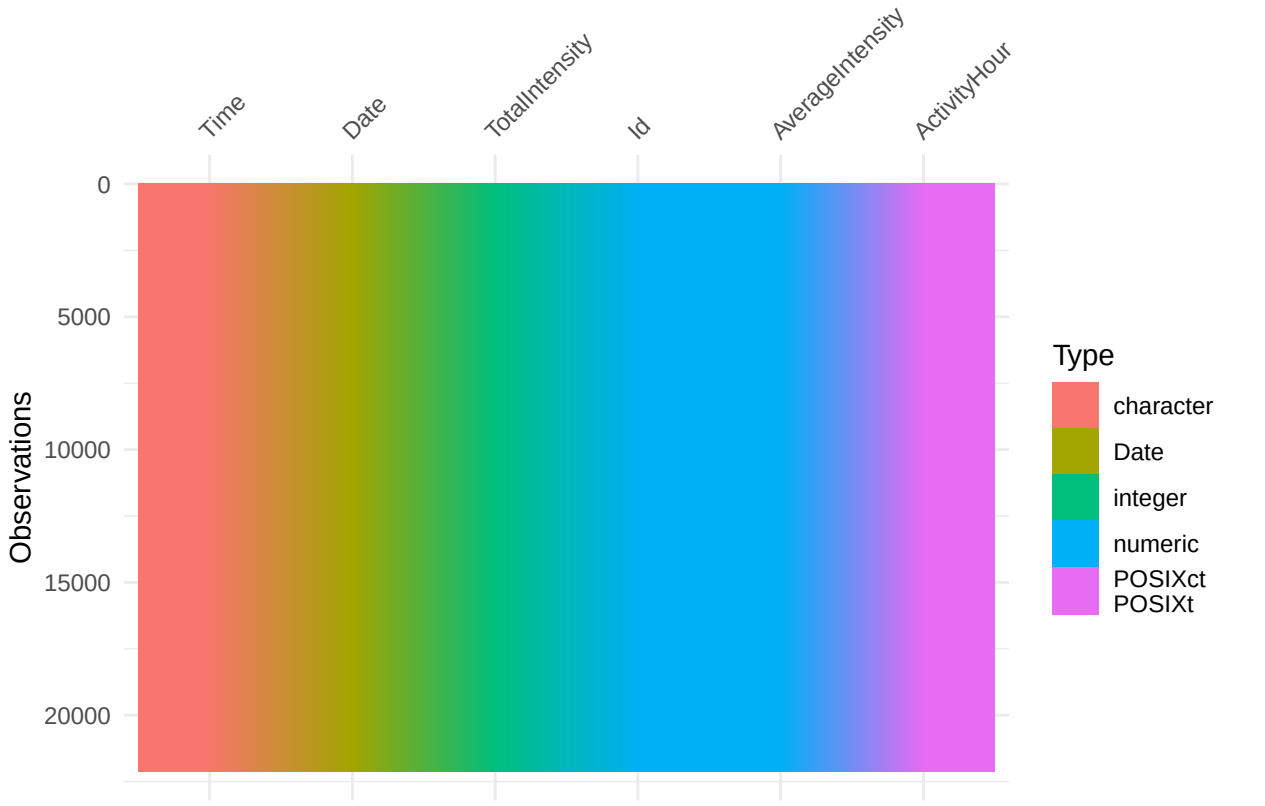
We see that most of our datasets contain 33 distinct user id's, however the sleep and weight datasets have fewer. Let's install another package to help us understand the data better. We can use the `visdat` packages to provide visualizations to show the amount of observations, data types, missing data, etc. This can help us with verifying the data's integrity.

```
install.packages("visdat")
```

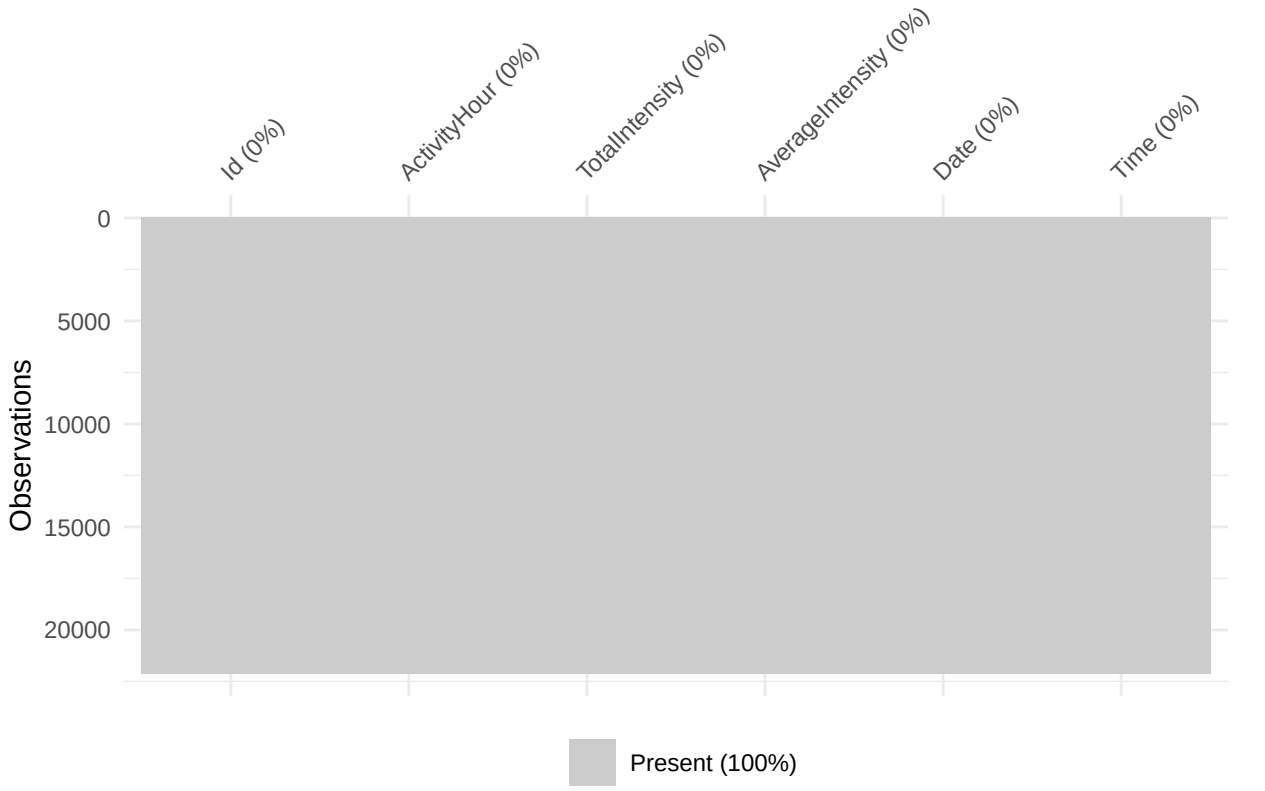
```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.2'
## (as 'lib' is unspecified)
```

```
library(visdat)
vis_dat(intensities)
```

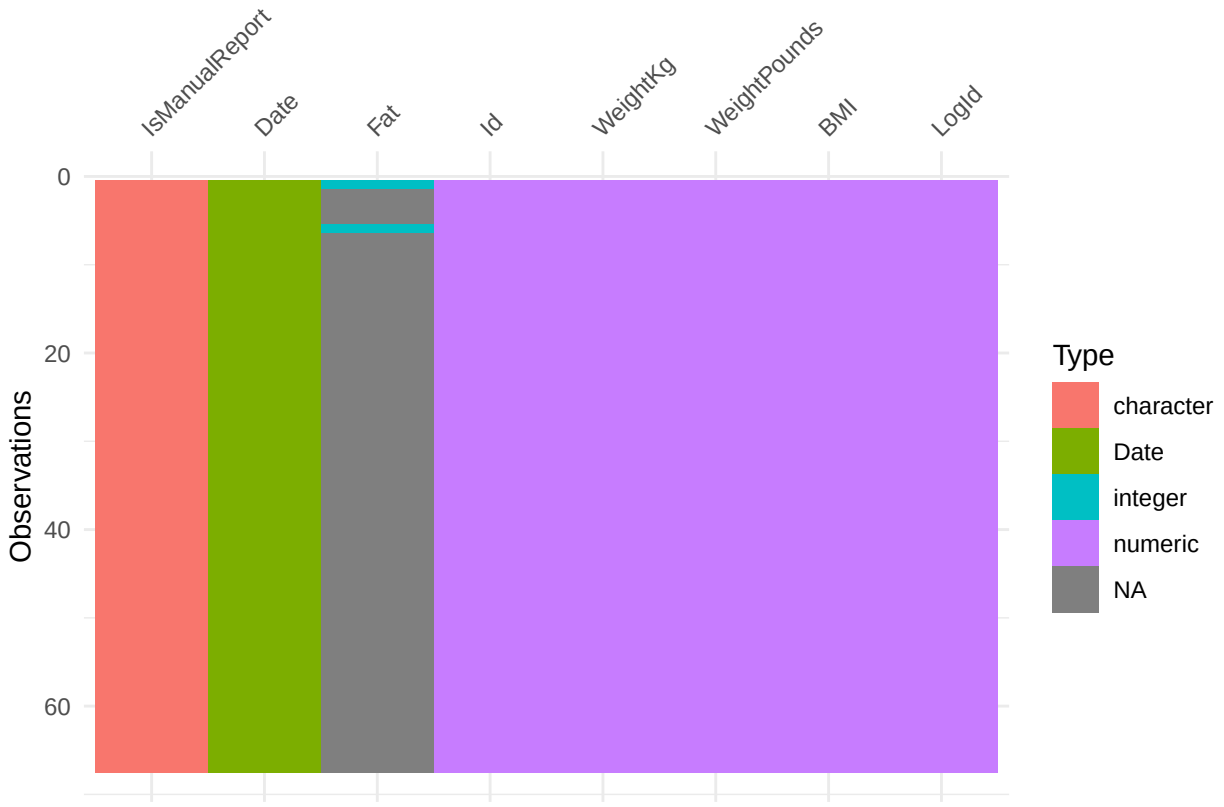
```
## Warning: `gather_()` was deprecated in tidyr 1.2.0.
## Please use `gather()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```



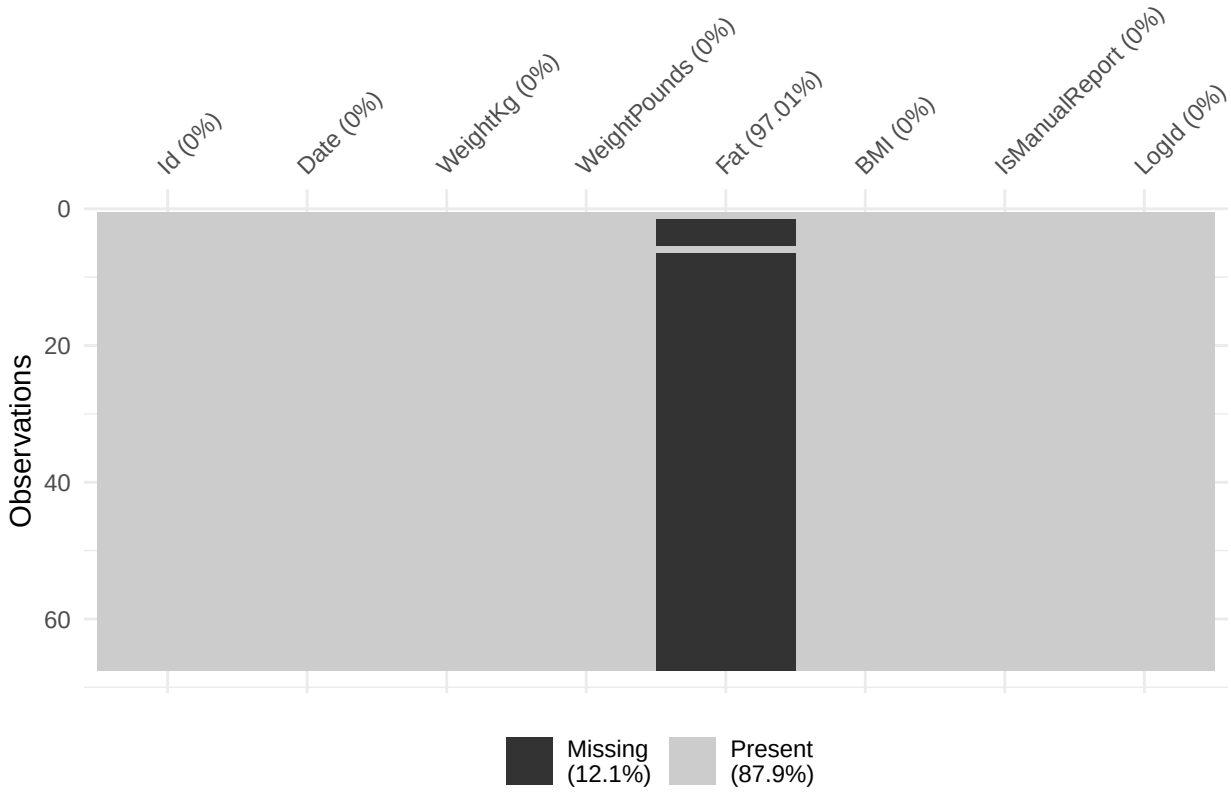

```
vis_miss(intensities)
```



```
vis_dat(weight)
```




```
vis_miss(weight)
```

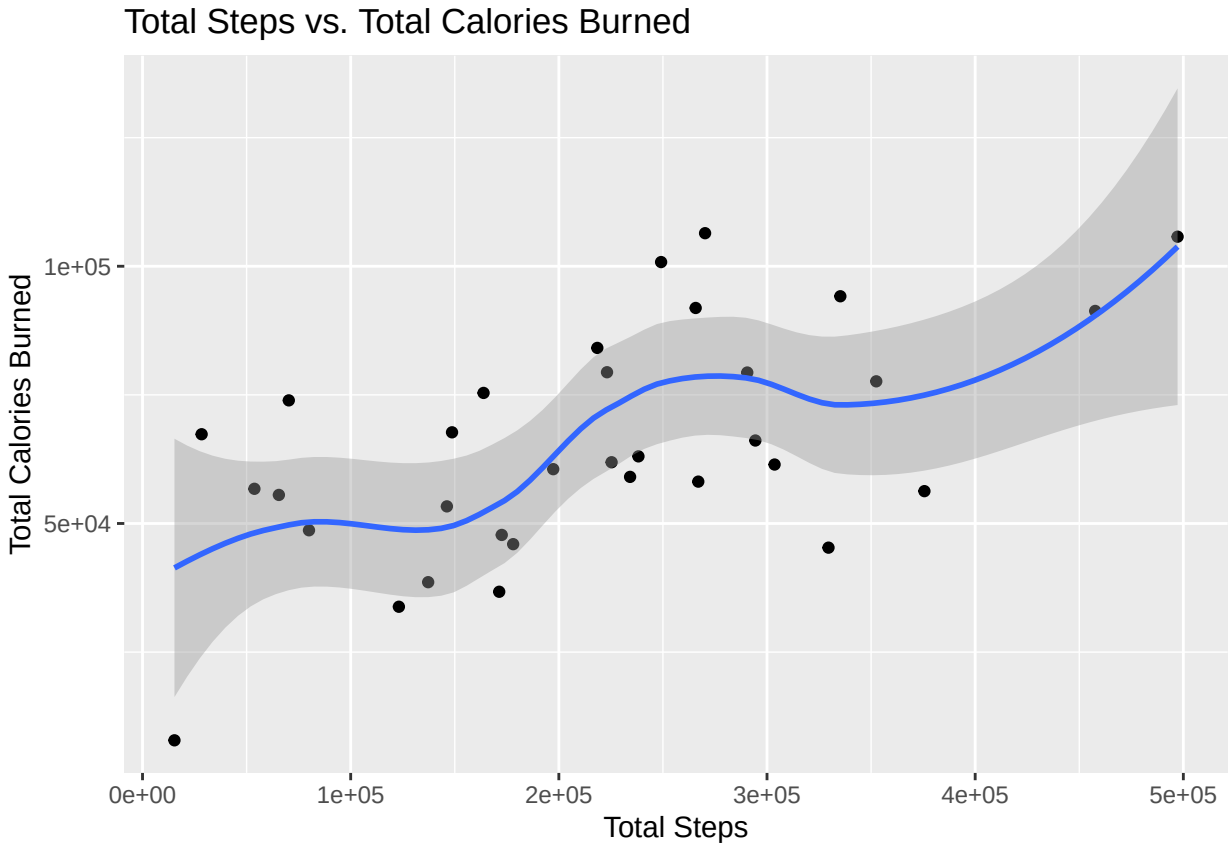


We can clearly see there is a big issue with the weight data, and since there are only 8 user Id's, we can say that data is not suitable for analysis.

We will merge some data in order to visualize it. Let's compare the amount of activity to the amount of calories burned.

```
activity_by_id <- setNames(aggregate(activity$TotalSteps, list(activity$Id), sum), c("Id Number","Total
calories_by_id <- setNames(aggregate(calories$Calories, list(calories$Id), sum), c("Id Number", "Total
activity_calories <- data.frame(x=activity_by_id$`Total Steps`, y=calories_by_id$`Total Calories Burned
ggplot(activity_calories, aes(x=x, y=y))+geom_point()+geom_smooth()+labs(x='Total Steps', y='Total Calo
```

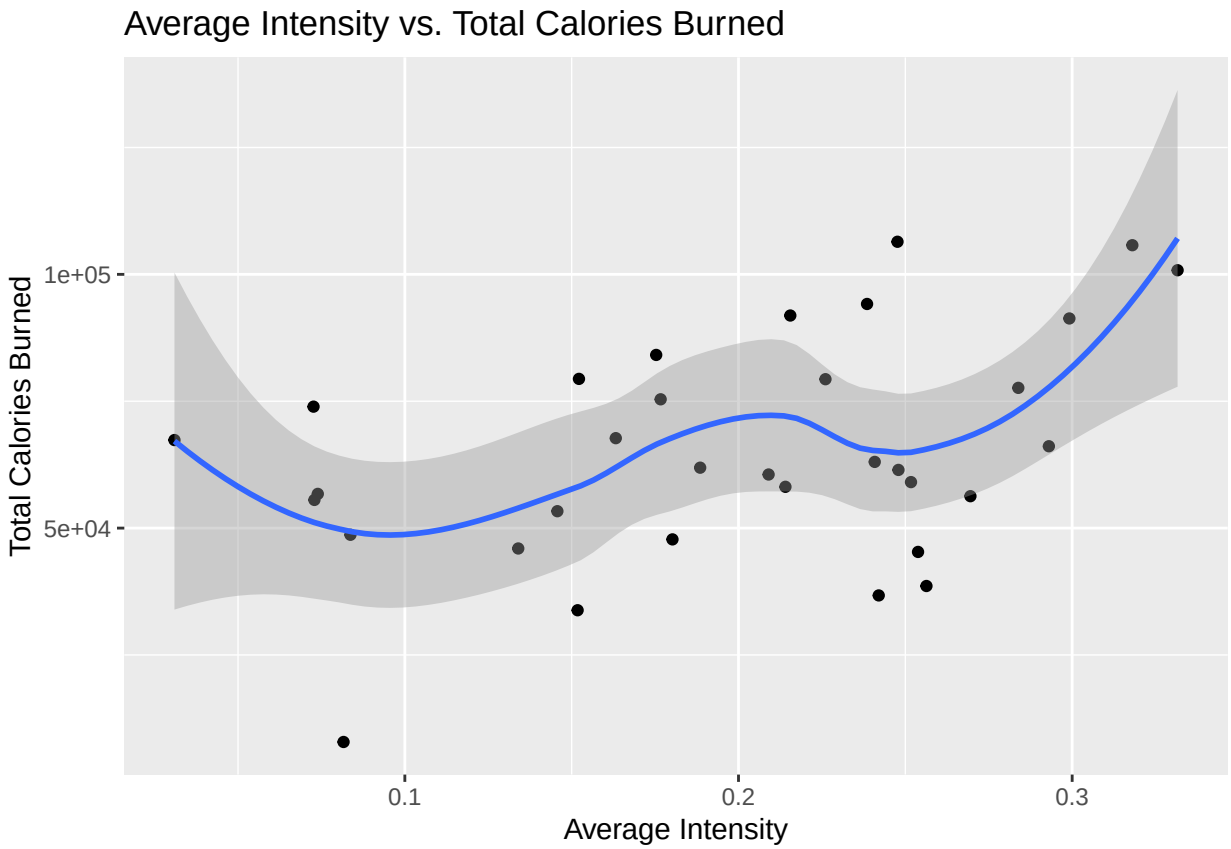
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



As expected, there is a positive correlation between total steps and total calories burned. Let's examine the intensities of the users' activity in relation to how many calories were burned.

```
intensity_by_id <- setNames(aggregate(intensities$AverageIntensity, list(intensities$Id), mean), c("Id
calories_by_id <- setNames(aggregate(calories$Calories, list(calories$Id), sum), c("Id Number", "Total
intensity_calories <- data.frame(x=intensity_by_id$`Average Intensity`, y=calories_by_id$`Total Calorie
ggplot(intensity_calories, aes(x=x, y=y))+geom_point()+geom_smooth()+labs(x='Average Intensity', y='Tota
```

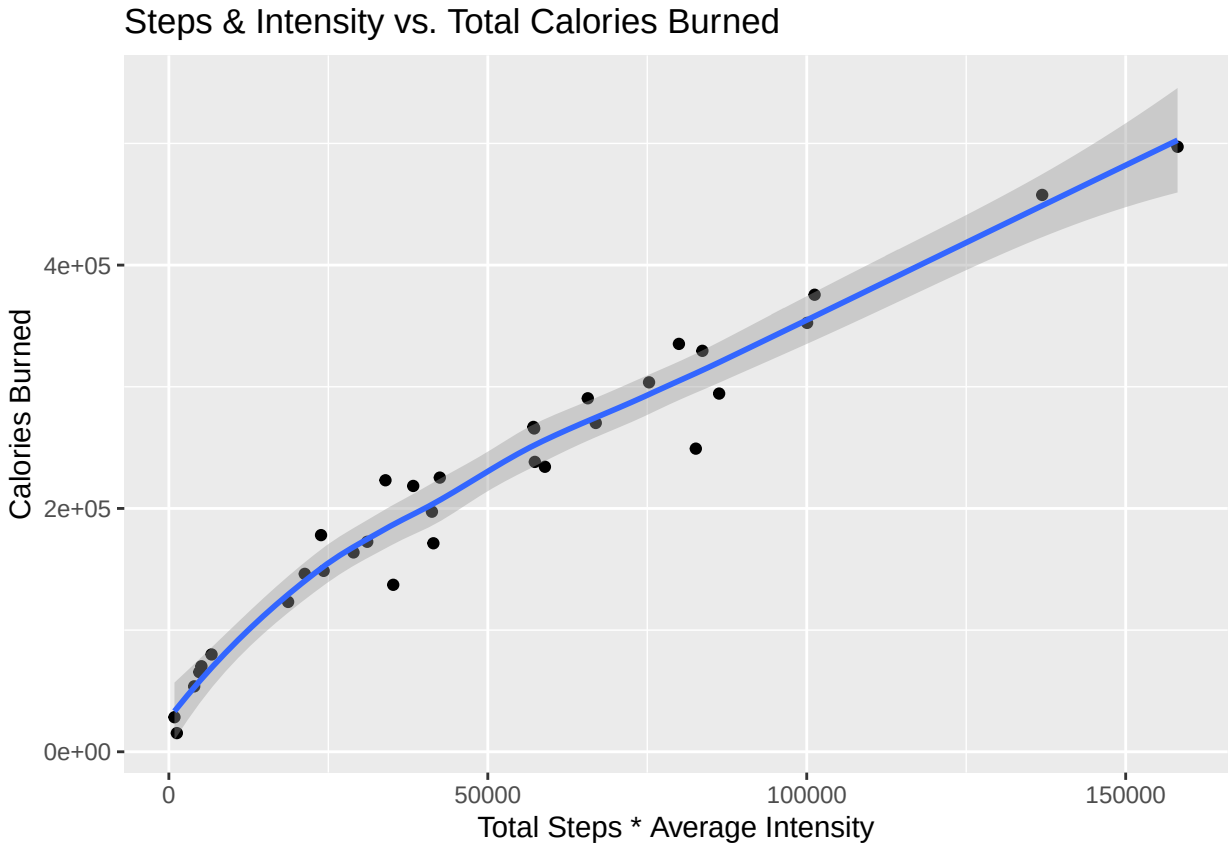
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Interestingly, we see a similar trend line as we did when comparing total steps to calories burned. Let's see what happens when we multiply the steps by intensity, then compare with total calories burned.

```
step_intensity <- c(intensity_by_id$`Average Intensity`*activity_by_id$`Total Steps`)
step_intensity_calories <- data.frame(x=step_intensity, y=activity_by_id$`Total Steps`)
ggplot(step_intensity_calories, aes(x=x,y=y))+geom_point()+geom_smooth()+labs(x='Total Steps * Average
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

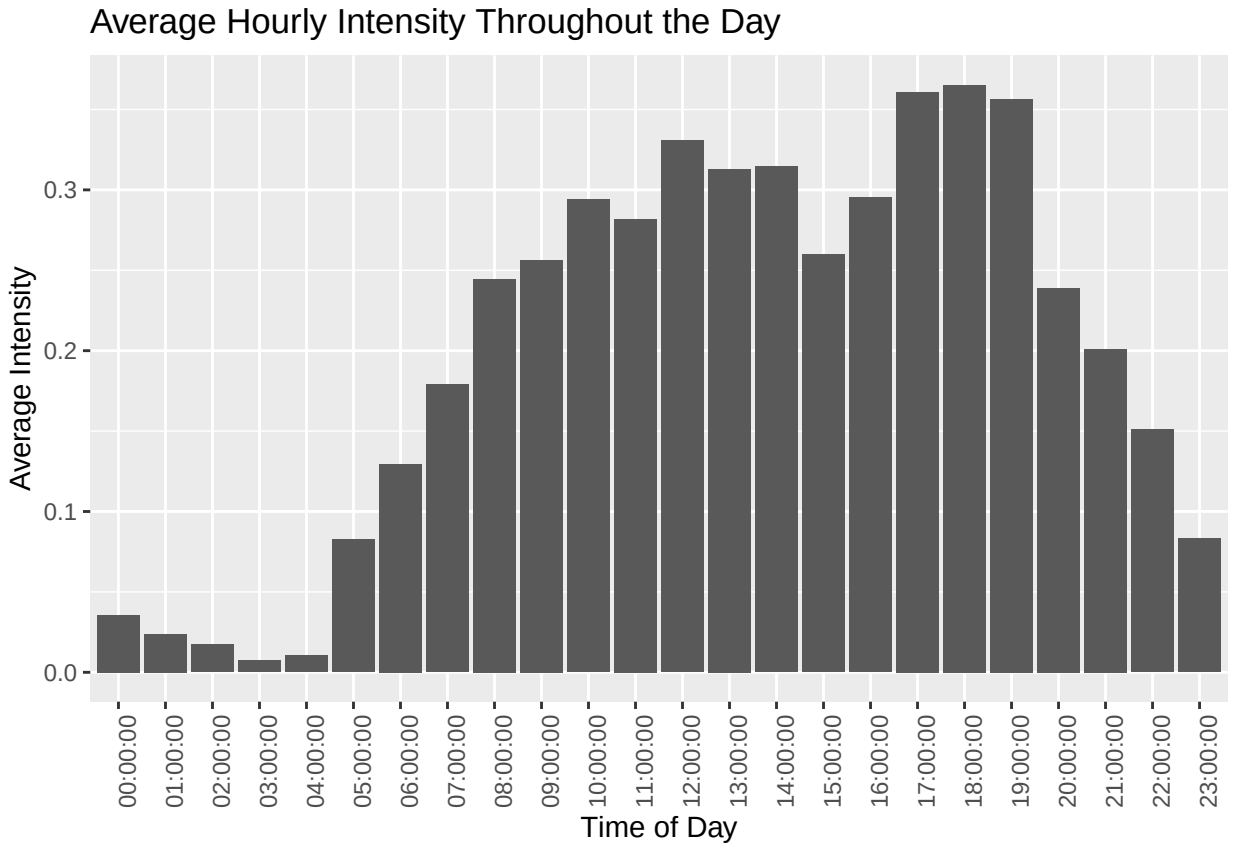


Here we observe a much more linear trend line than our previous visualizations, and it is clear that more calories are burned when there is more activity at a higher intensity.

Let's look at the hourly intensity data.

```
new_int <- intensities %>% group_by(Time) %>% drop_na() %>% summarise(avg_int = mean(AverageIntensity))
ggplot(data=new_int, aes(x=Time, y=avg_int))+geom_histogram(stat = "identity")+labs(x='Time of Day', y=
```

```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```

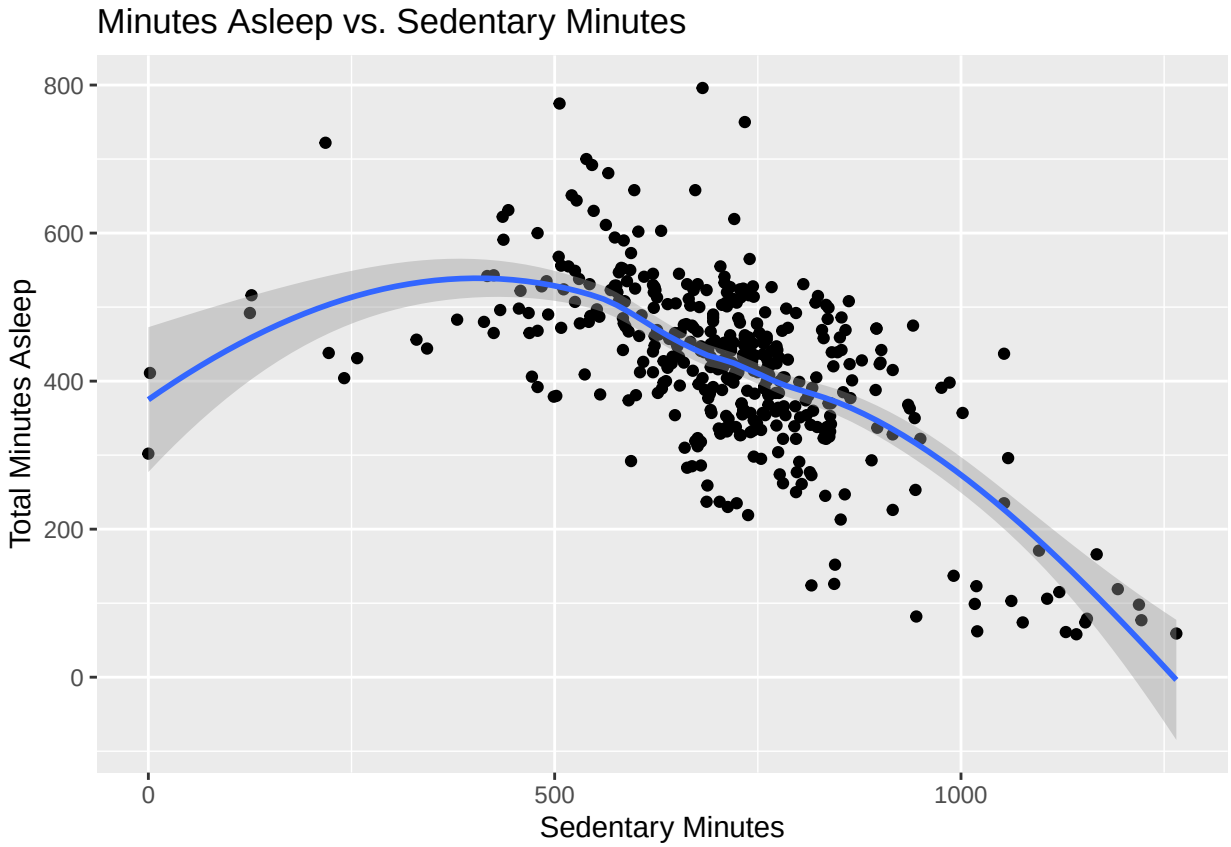


This histogram shows the activity intensities of all users averaged over a 24-hour time period. We see a spike between the hours of 5pm-7pm, which could possibly mean that people are working out when they get off of work.

Now let us explore the relationship between sleep and activity. To accomplish this, I will merge the sleep and activity datasets by Id and date.

```
sleep_and_activity <- merge(sleep, activity, by=c('Id', 'Date'))
ggplot(sleep_and_activity, aes(x=SedentaryMinutes, y=TotalMinutesAsleep))+geom_point()+geom_smooth()+lab
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



We can see a negative correlation here; that is, a more sedentary person spends fewer minutes asleep.

Recommendations

We've seen a few interesting trends in our data and can use these to provide recommendations for Bellabeat.

- As expected, we observed that more activity at higher intensities leads to more calories burned. Based on this, the Bellabeat app can send push notifications to encourage users to be more active throughout the day. For example, adding a daily goal feature may encourage users to complete tasks such as "maintain a heart rate of X bpm for X minutes" or "walk a total of 7,500 steps in a day."
- Bellabeat may wish to add a daily goal feature that uses push notifications and various incentives to encourage people to achieve their fitness goals. For example, if a user maintains a "streak" of meeting their goals every day for a certain number of days, they could be provided free trial of the Bellabeat membership.
- Since we observed a spike of user activity between the hours of 5pm-7pm, the Bellabeat app should send a push notification to remind users to workout at that time.
- If a user wishes to improve their sleep, there should be push notification reminders to engage in physical activity as it is shown that more sedentary time leads to fewer minutes asleep.
- Due to the activity spike between 5pm-7pm, an average sleep time of 7.6 hours, and an average step count of ~7,000 steps we can assume the typical user involved in this study works a typical 9-5 job and is moderately active. Bellabeat should market to this group by promoting an atmosphere of overall wellness as opposed to intense workouts and drastic weight loss. I recommend adjusting the app and its features to compliment this customer's lifestyle.