BellaBeat Behavior Analysis

Durga Sundaram

Introduction and background

This R Markdown document sums up my user behavior analysis to help guide BellaBeat Marketing team to strategize on targeting the right consumer segment to boost their sales for one of their product, the smart watch called **Time**

Source file details

I am using the following open source dataset in the form of CSV files that can be found at the link below: https://www.kaggle.com/arashnic/fitbit * Attention: Although I find that the dataset is not ROCCC compliant, I am going ahead with the analysis for practice purposes.

The user fitness details being tracked by the FitBit application and tracker are as follows: * Activity level * Step count * Distance * Sleep measurement: Time Asleep and Time on Bed * Calories burnt * Heartrate

These CSV files can be categorized based on the following levels of granularity: * Daily * Hourly * Minutes * Seconds

For my analysis, I would like to rollup the daily data into week level. I feel the minutes level and seconds level information for different users is too fine to help my analysis. So I would be rolling up the minutes and seconds level to average it out to hourly data to find user behavior patterns.

Installing and loading the necessary packages and libraries

For our purpose, we are using the following packages in R: * tidyverse - Basic package for data analysis * lubridate - For date manipulation * skimr & janitor - For data cleaning * sqldf - For analysis using SQL commands on dataframe

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

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

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

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

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

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

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

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

```
install.packages("sqldf")

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

library(tidyverse)

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

## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     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(lubridate)

##
## Attaching package: 'lubridate'

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

library(skimr)
library(janitor)

##
## Attaching package: 'janitor'

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

library(dplyr)
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
```

Loading the source files into R structures

For my analysis I used spreadsheets to analyse Daily and Hourly data. Minutes and Seconds level data are too huge to process in spreadsheets. They have format issues to be loaded into SQL as well. I leveraged R to cleanup the minute level METs data. I used R solely for analysing the seconds level data about heartrate.

Lets dig in!

Cleaning the data

I am fixing the date format for this dataset to datetime format and saving it into heartrate_df_clean dataframe. Finally, I am converting ID from number to string field help plot axis values in graph easily

```
heartrate_df <- read_csv("heartrate_seconds_merged.csv")
```

```
## Rows: 2483658 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): Time
## dbl (2): Id, Value
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
colnames(heartrate_df)
```

```
## [1] "Id"    "Time"  "Value"
```

```
heartrate_df_clean <- mutate(heartrate_df, Time_date = mdy_hms(Time))
head(heartrate_df_clean,4)
```

```
## # A tibble: 4 x 4
##           Id Time                 Value Time_date
##        <dbl> <chr>                <dbl> <dttm>
## 1 2022484408 4/12/2016 7:21:00 AM    97 2016-04-12 07:21:00
## 2 2022484408 4/12/2016 7:21:05 AM   102 2016-04-12 07:21:05
## 3 2022484408 4/12/2016 7:21:10 AM   105 2016-04-12 07:21:10
## 4 2022484408 4/12/2016 7:21:20 AM   103 2016-04-12 07:21:20
```

```
heartrate_df_clean <- select(heartrate_df_clean, -Time)
head(heartrate_df_clean,4)
```

```
## # A tibble: 4 x 3
##           Id Value Time_date
##        <dbl> <dbl> <dttm>
## 1 2022484408    97 2016-04-12 07:21:00
## 2 2022484408   102 2016-04-12 07:21:05
## 3 2022484408   105 2016-04-12 07:21:10
## 4 2022484408   103 2016-04-12 07:21:20
```

```
heartrate_df_clean <- mutate(heartrate_df_clean, Id = as.character(Id))

skim_without_charts(heartrate_df_clean)
```

Table 1: Data summary

Name	heartrate_df_clean
Number of rows	2483658
Number of columns	3
Column type frequency:	
character	1
numeric	1
POSIXct	1
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
Id	0	1	10	10	0	14	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
Value	0	1	77.33	19.4	36	63	73	88	203

Variable type: POSIXct

skim_variable	n_missing	complete_rate	min	max	median	n_unique
Time_date	0	1	2016-04-12	2016-05-12 16:20:00	2016-04-26 20:28:50	961274

Once I check the summary statistics of the dataset, I proceed to check for data validity. I cross verify the user IDs to see if they match with the parent dataset dailyActivity_merged as available my spreadsheet. I loved experimenting the sql usage within the context of R using sqldf function. There are multiple ways to do this, which is the power of R!

```
sqldf("select count(distinct(ID)) from heartrate_df_clean")
```

```
##   count(distinct(ID))
## 1                  14
```

```
sqldf("select distinct(ID) from heartrate_df_clean")
```

```
##            Id
## 1  2022484408
## 2  2026352035
## 3  2347167796
## 4  4020332650
## 5  4388161847
## 6  4558609924
## 7  5553957443
## 8  5577150313
## 9  6117666160
## 10 6775888955
## 11 6962181067
## 12 7007744171
## 13 8792009665
## 14 8877689391
```

```
n_distinct(heartrate_df_clean$Id)
```

```
## [1] 14
```

Housekeeping

Given the dataset size, I proceed to cleanup the source dataframe to free up some memory for further analysis.

```
rm(heartrate_df)
```

Analysing the data

Splitting the time Here I am splitting the time into individual components of year, month, day, weekday, hour, minute and seconds. I'm sure there will be an efficient way to process this, but for beginner level, I find this good enough.

```
heartrate_df_clean <- mutate(heartrate_df_clean, Time_year = year(Time_date), Time_mon = month(Time_dat

glimpse(heartrate_df_clean)
```

```
## Rows: 2,483,658
## Columns: 10
## $ Id           <chr> "2022484408", "2022484408", "2022484408", "2022484408", "~
## $ Value        <dbl> 97, 102, 105, 103, 101, 95, 91, 93, 94, 93, 92, 89, 83, 6~
## $ Time_date    <dttm> 2016-04-12 07:21:00, 2016-04-12 07:21:05, 2016-04-12 07:~
## $ Time_year    <dbl> 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 201~
## $ Time_mon     <dbl> 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, ~
## $ Time_dateNum <int> 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 1~
## $ Time_week    <ord> Tue, Tue, Tue, Tue, Tue, Tue, Tue, Tue, Tue, Tue, Tue, Tu~
## $ Time_hour    <int> 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, ~
## $ Time_min     <int> 21, 21, 21, 21, 21, 22, 22, 22, 22, 22, 22, 22, 22, 22, 2~
## $ Time_sec     <dbl> 0, 5, 10, 20, 25, 5, 10, 15, 20, 25, 35, 40, 50, 55, 0, 1~
```

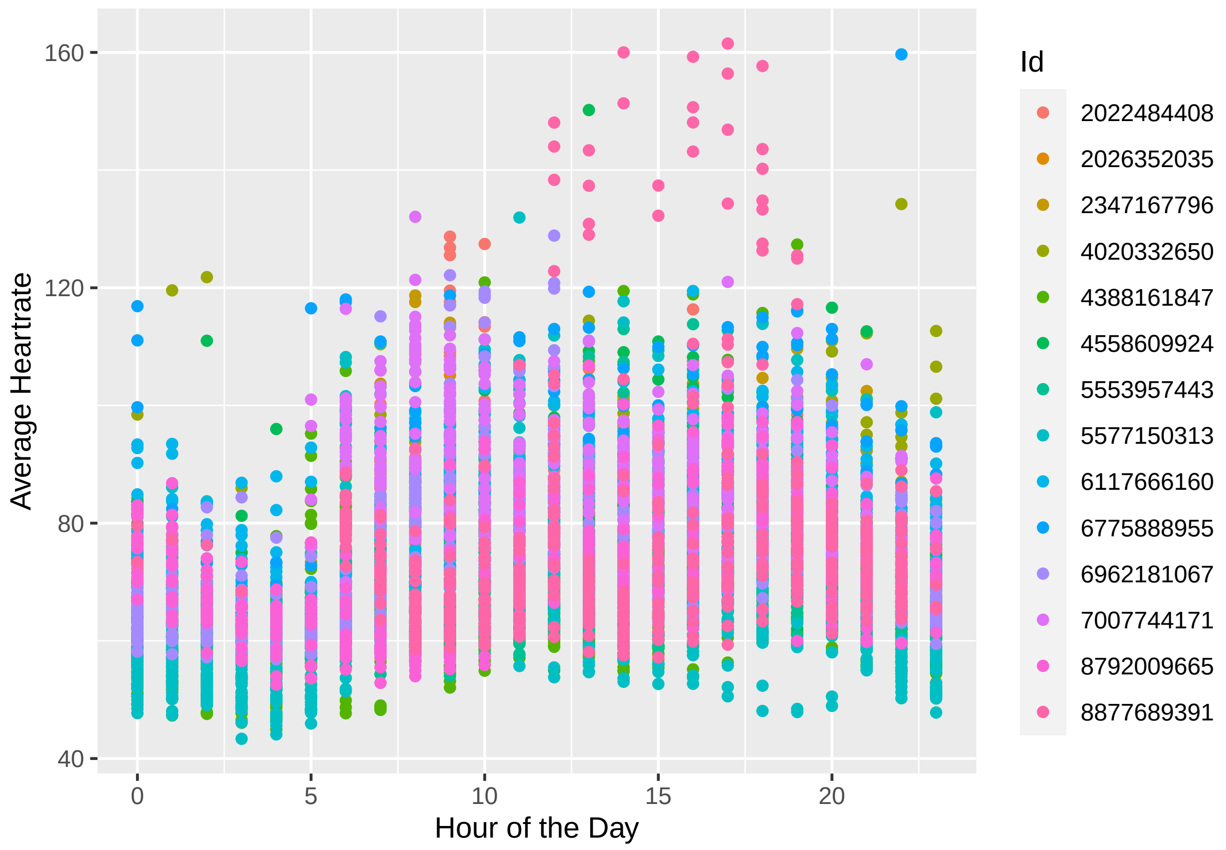
Visualize data

Now we are ready to summarize the average heartbeat by user, day, hour and so on. Heartbeat data gives a good indicator of the user's activity level.

I am creating a scatterplot with different users to show average heartrate per hour of the day.

```
heartrate_df_clean %>%
  group_by(Id, Time_mon, Time_dateNum, Time_hour) %>%
  summarise(AvgHeartRate = mean(Value)) %>%
  ggplot(aes(x=Time_hour, y=AvgHeartRate, group = Id, color = Id)) +
  geom_point() +
  labs(x= "Hour of the Day", y= "Average Heartrate")
```

```
## `summarise()` has grouped output by 'Id', 'Time_mon', 'Time_dateNum'. You can
## override using the `.groups` argument.
```

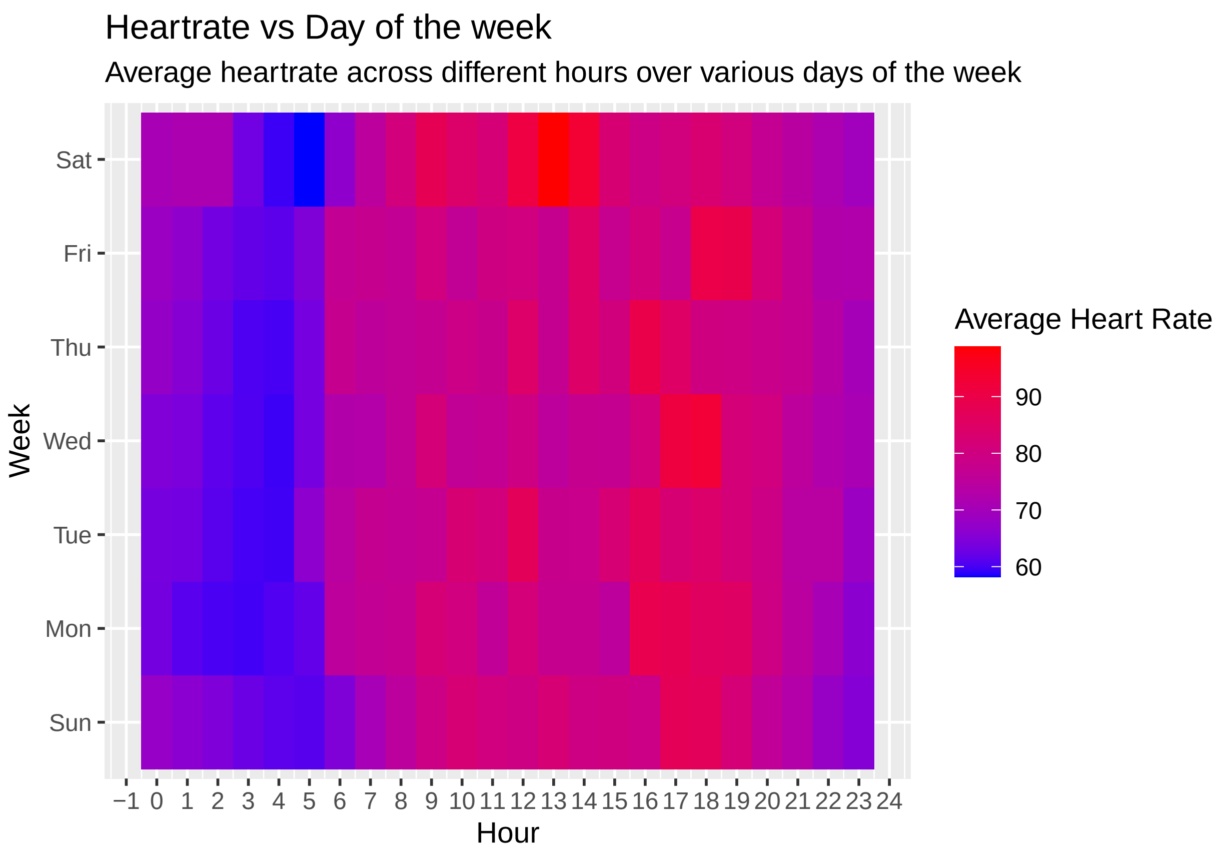


Analysis Result : This shows which user is more active and which one is less active by the hour of the day. After analysing the heartrate by users, we find the following: * Approximately 10-15% of the users are active with more than 140 bpm. * Those who are highly active during the later part of the day have a better quality of sleep signalled by lower heartrate below 60bpm. * Users who have recorded high intensity activity through heartrate of more than 120 bpm plan their workouts between noon and 6 PM, which is lunch break and right after office.

Next I want to take the users as a whole and want to see the spread of heartrate across each day of the week. This will tell us which days are more intense for the sample set and which hours are more intense on an average. I do this by creating a heatmap as below. I want to save this image as a pdf for later reference.

```
heartrate_df_clean %>%
  group_by( Time_week, Time_hour) %>%
  summarise(AvgHeartRate = mean(Value)) %>%
  ggplot(aes(x=Time_hour, y=Time_week, fill=AvgHeartRate)) +
  geom_tile() +
  scale_fill_gradient(low="blue", high="red")+
  scale_x_continuous(breaks = scales::pretty_breaks(n=23))+
  labs(x = "Hour", y= "Week",
       title="Heartrate vs Day of the week",
       subtitle ="Average heartrate across different hours over various days of the week",
       fill = "Average Heart Rate")
```

```
## `summarise()` has grouped output by 'Time_week'. You can override using the
## `.groups` argument.
```



```r
ggsave("Heartrate_Weekday.pdf", device = "pdf", width = 8, height = 10, units = "in" )
```

Analysis Result :

- The heartrate heatmap shows that the average user in the sample set are early risers with lowest heartrate between 12 -5AM
- Their week starts slowly with Sunday to Tuesday being lower activity as compared to the second half of the week. During the workweek, their favorite workout time is between 4-7PM
- On Tuesdays, they seem to push the high intensity workout during lunch break.
- On weekends they like to workout late on Friday between 5:30-8PM.
- On Saturday, they wake up late by and workout during noon, probably before a brunch.
- Sundays are relatively relaxed.

Conclusion

Based on the analysis, the user group seems to be office goers with desk jobs. They are highly motivated and follow a set routine in their life. They are committed to wellness and have the discipline to workout through out the week. * To target the population with these characteristics, the marketing team could use *social media* as a preferred medium to reach out to the segment of population who are single and have more control over their lives and buffer to prioritize their needs. * The selling point can be stress management and better quality of sleep as shown by the data to motivate them to improve their physical fitness which contributes to their looking and feeling good.