CaseStudy\_Bellabeat

Piyusha

2022-07-20

# Bellabeat Case Study

### Scenario :

I am a junior data analyst working on the marketing analyst team at Bellabeat, a high-tech manufacturer of health-focused products for women. Urška Sršen, co founder and Chief Creative Officer of Bellabeat, believes that analyzing smart device fitness data could help unlock new growth opportunities for the company. You have been asked to focus on one of Bellabeat’s products and analyze smart device data to gain insight into how consumers are using their smart devices. The insights you discover will then help guide marketing strategy for the company.

### Phases of Data Analysis

1. Ask
2. Prepare
3. Process
4. Analyse
5. Share
6. Act

## Ask Phase

### Business Task

Find new growth opportunities for Bellabeat in the smart wellness devices market by analyzing smart device data to gain insights on the consumer’s usage of smart wellness devices.

### Key Stakeholders

* Urška Sršen – Cofounder
* Sando Mur – executive team member and cofounder

## Prepare Phase

Key objectives of the prepare phase of Data Analysis :

* Understand how data is generated and collected.
* Identify and use different data formats, types, and structures.
* Make sure data is unbiased and credible.
* Organize and protect data.

### About the dataset

FitBit Fitness Tracker Data (CC0: Public Domain, dataset made available through Mobius): This Kaggle data set contains personal fitness tracker from thirty fitbit users. Thirty eligible Fitbit users consented to the submission of personal tracker data, including minute-level output for physical activity, heart rate, and sleep monitoring. It includes information about daily activity, steps, and heart rate that can be used to explore users’ habits.

### Exploring data to find limitations

#Installing and loading the necessary packages   
#install.packages("tidyverse")  
#install.packages("sqldf") # I will be using a bit of SQL to query data   
  
#install.packages("proto") #this and the next two packages are for using sql  
#install.packages("RSQLite")  
#install.packages("gsubfun")  
  
library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──  
## ✔ ggplot2 3.3.6 ✔ purrr 0.3.4  
## ✔ tibble 3.1.7 ✔ dplyr 1.0.9  
## ✔ tidyr 1.2.0 ✔ stringr 1.4.0  
## ✔ readr 2.1.2 ✔ forcats 0.5.1  
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()

library(readr)  
library(sqldf)

## Loading required package: gsubfn  
## Loading required package: proto  
## Loading required package: RSQLite

library(dplyr)  
  
#Importing the data   
  
  
activity<- read.csv("dailyActivity\_merged.csv")  
  
steps <- read.csv("dailySteps\_merged.csv")  
  
intensity <- read.csv("dailyIntensities\_merged.csv")  
  
weight <- read.csv("weightLogInfo\_merged.csv")  
  
calories <- read.csv("dailyCalories\_merged.csv")  
  
sleep <- read.csv("sleepDay\_merged.csv")  
  
#Checking the structure of imported data   
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 ...

str(steps)

## 'data.frame': 940 obs. of 3 variables:  
## $ Id : num 1.5e+09 1.5e+09 1.5e+09 1.5e+09 1.5e+09 ...  
## $ ActivityDay: chr "4/12/2016" "4/13/2016" "4/14/2016" "4/15/2016" ...  
## $ StepTotal : int 13162 10735 10460 9762 12669 9705 13019 15506 10544 9819 ...

str(intensity)

## 'data.frame': 940 obs. of 10 variables:  
## $ Id : num 1.5e+09 1.5e+09 1.5e+09 1.5e+09 1.5e+09 ...  
## $ ActivityDay : chr "4/12/2016" "4/13/2016" "4/14/2016" "4/15/2016" ...  
## $ SedentaryMinutes : int 728 776 1218 726 773 539 1149 775 818 838 ...  
## $ LightlyActiveMinutes : int 328 217 181 209 221 164 233 264 205 211 ...  
## $ FairlyActiveMinutes : int 13 19 11 34 10 20 16 31 12 8 ...  
## $ VeryActiveMinutes : int 25 21 30 29 36 38 42 50 28 19 ...  
## $ SedentaryActiveDistance : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ LightActiveDistance : num 6.06 4.71 3.91 2.83 5.04 ...  
## $ ModeratelyActiveDistance: num 0.55 0.69 0.4 1.26 0.41 ...  
## $ VeryActiveDistance : num 1.88 1.57 2.44 2.14 2.71 ...

str(sleep)

## 'data.frame': 413 obs. of 5 variables:  
## $ Id : num 1.5e+09 1.5e+09 1.5e+09 1.5e+09 1.5e+09 ...  
## $ SleepDay : chr "4/12/2016 12:00:00 AM" "4/13/2016 12:00:00 AM" "4/15/2016 12:00:00 AM" "4/16/2016 12:00:00 AM" ...  
## $ TotalSleepRecords : int 1 2 1 2 1 1 1 1 1 1 ...  
## $ TotalMinutesAsleep: int 327 384 412 340 700 304 360 325 361 430 ...  
## $ TotalTimeInBed : int 346 407 442 367 712 320 377 364 384 449 ...

library(dplyr)  
  
# n\_distinct() function from dplyr package can also be used to find distinct IDs in dataframes  
  
n\_distinct(activity$Id)

## [1] 33

n\_distinct(steps$Id)

## [1] 33

n\_distinct(intensity$Id)

## [1] 33

n\_distinct(weight$Id)

## [1] 8

n\_distinct(sleep$Id)

## [1] 24

From this output we find that all the dates are in char format. We must convert them all to ‘date’ format and have same name for column containing date for all tables to make the data consistent. That will be our first step while cleaning data in the next phase.

From n\_distinct() output we can infer the following about consumer usage :

1. Steps and Intensity data is available for all customers in the dataset.
2. Only 24 % users provided weight data.
3. 27.28 % users did not record sleep data.

So we decide not to use weight data for our analysis.

# On viewing all imported data frames, we observe that the data frames "calories" and "intensity" may be subsets of activity. We use SQL to find out if that is true.   
  
#Following code performs leftjoin of calories and intensities table on activity table and returns the number of rows in the result.   
  
sqldf("SELECT COUNT()  
 FROM activity   
 LEFT JOIN calories ON   
 activity.Id = calories.Id AND   
 activity.ActivityDate = calories.ActivityDay AND   
 activity.Calories = calories.Calories")

## COUNT()  
## 1 940

sqldf("SELECT COUNT()  
 FROM activity   
 LEFT JOIN steps ON   
 activity.Id = steps.Id AND   
 activity.ActivityDate = steps.ActivityDay AND   
 activity.Totalsteps = steps.StepTotal")

## COUNT()  
## 1 940

sqldf("SELECT COUNT()  
 FROM activity   
 LEFT JOIN intensity ON   
 activity.Id = intensity.Id AND   
 activity.ActivityDate = intensity.ActivityDay AND   
 activity.SedentaryMinutes = intensity.SedentaryMinutes AND  
 activity.LightlyActiveMinutes = intensity.LightlyActiveMinutes AND  
 activity.FairlyActiveMinutes = intensity.FairlyActiveMinutes AND  
 activity.VeryActiveMinutes = intensity.VeryActiveMinutes AND  
 activity.SedentaryActiveDistance = intensity.SedentaryActiveDistance AND  
 activity.LightActiveDistance = intensity.LightActiveDistance AND  
 activity.ModeratelyActiveDistance = intensity.ModeratelyActiveDistance AND  
 activity.VeryActiveDistance = intensity.VeryActiveDistance")

## COUNT()  
## 1 940

In the result output, we find that number of rows for all joins is the same i.e 940. So we remove these joins now so that we can use “activity” as a single dataframe.

rm(calories,intensity,steps)  
  
# Just checking to make sure they've been removed   
colnames(activity)

## [1] "Id" "ActivityDate"   
## [3] "TotalSteps" "TotalDistance"   
## [5] "TrackerDistance" "LoggedActivitiesDistance"  
## [7] "VeryActiveDistance" "ModeratelyActiveDistance"  
## [9] "LightActiveDistance" "SedentaryActiveDistance"   
## [11] "VeryActiveMinutes" "FairlyActiveMinutes"   
## [13] "LightlyActiveMinutes" "SedentaryMinutes"   
## [15] "Calories"

### Limitations of this data

As mentioned in the case study statement, there are limitations to this data. Following are some of the most critical limitations :

1. No demographic information present. This is a womens brand so wome’s data must be used for analysis.
2. This data can be biased because it has information about only 33 users.
3. The data seems outdated as it is from 2016 i.e five years ago.
4. Data is also from only two months – April and May. It is possible that different trends are observed in different seasons / times of the year.
5. All the data is not available and all the dates are not consistent.

## Process Phase

### Data cleaning

As mentioned before, the first step in cleaning data will be changing the inconsistent data into consistent data.

#Importing the data   
  
activity<- read.csv("dailyActivity\_merged.csv")  
  
steps <- read.csv("dailySteps\_merged.csv")  
  
intensity <- read.csv("dailyIntensities\_merged.csv")  
  
weight <- read.csv("weightLogInfo\_merged.csv")  
  
calories <- read.csv("dailyCalories\_merged.csv")  
  
sleep <- read.csv("sleepDay\_merged.csv")  
  
  
  
  
  
  
  
  
# We begin by renaming columns and chaning the format of date from char to 'date'   
  
library(lubridate)

##   
## Attaching package: 'lubridate'

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

activity <- activity %>%   
 rename(date= ActivityDate) %>%  
 mutate(date= mdy(date)) # as\_date() must be used with mutate to convert char to date format   
  
sleep <- sleep %>%  
 rename(date= SleepDay) %>%  
 mutate(date= as\_date(date, format= "%m/%d/%Y %I:%M:%S %p"))  
  
intensity <- intensity %>%   
 rename(date= ActivityDay) %>%   
 mutate(date = mdy(date))  
  
# Finding the number of duplicates and removing if they exist  
  
sum(duplicated(activity))

## [1] 0

sum(duplicated(sleep))

## [1] 3

sum(duplicated(intensity))

## [1] 0

So there are duplicates in sleep data frame. We must remove them.

sleep <- distinct(sleep)  
  
# Checking if all duplicates are removed   
  
sum(duplicated(sleep))

## [1] 0

Now our data is consistent and free of duplicate rows. Further we will perform analysis on our data. We will use only ‘activity’ data frame to perform our analysis and plot it. For that, we must merge the ‘sleep’ data frame with ‘activity’.

final\_activity <- merge(activity,sleep, by = c("Id","date"), all.x = TRUE) # all.x = TRUE is used because we know that sleep has data for fewer users than total.

#Analyse Phase

In this phase we will summarize data to make data driven decisions to provide recommendations.

summary(final\_activity)

## Id date TotalSteps TotalDistance   
## Min. :1.504e+09 Min. :2016-04-12 Min. : 0 Min. : 0.000   
## 1st Qu.:2.320e+09 1st Qu.:2016-04-19 1st Qu.: 3790 1st Qu.: 2.620   
## Median :4.445e+09 Median :2016-04-26 Median : 7406 Median : 5.245   
## Mean :4.855e+09 Mean :2016-04-26 Mean : 7638 Mean : 5.490   
## 3rd Qu.:6.962e+09 3rd Qu.:2016-05-04 3rd Qu.:10727 3rd Qu.: 7.713   
## Max. :8.878e+09 Max. :2016-05-12 Max. :36019 Max. :28.030   
##   
## TrackerDistance LoggedActivitiesDistance VeryActiveDistance  
## Min. : 0.000 Min. :0.0000 Min. : 0.000   
## 1st Qu.: 2.620 1st Qu.:0.0000 1st Qu.: 0.000   
## Median : 5.245 Median :0.0000 Median : 0.210   
## Mean : 5.475 Mean :0.1082 Mean : 1.503   
## 3rd Qu.: 7.710 3rd Qu.:0.0000 3rd Qu.: 2.053   
## Max. :28.030 Max. :4.9421 Max. :21.920   
##   
## ModeratelyActiveDistance LightActiveDistance SedentaryActiveDistance  
## Min. :0.0000 Min. : 0.000 Min. :0.000000   
## 1st Qu.:0.0000 1st Qu.: 1.945 1st Qu.:0.000000   
## Median :0.2400 Median : 3.365 Median :0.000000   
## Mean :0.5675 Mean : 3.341 Mean :0.001606   
## 3rd Qu.:0.8000 3rd Qu.: 4.782 3rd Qu.:0.000000   
## Max. :6.4800 Max. :10.710 Max. :0.110000   
##   
## VeryActiveMinutes FairlyActiveMinutes LightlyActiveMinutes SedentaryMinutes  
## Min. : 0.00 Min. : 0.00 Min. : 0.0 Min. : 0.0   
## 1st Qu.: 0.00 1st Qu.: 0.00 1st Qu.:127.0 1st Qu.: 729.8   
## Median : 4.00 Median : 6.00 Median :199.0 Median :1057.5   
## Mean : 21.16 Mean : 13.56 Mean :192.8 Mean : 991.2   
## 3rd Qu.: 32.00 3rd Qu.: 19.00 3rd Qu.:264.0 3rd Qu.:1229.5   
## Max. :210.00 Max. :143.00 Max. :518.0 Max. :1440.0   
##   
## Calories TotalSleepRecords TotalMinutesAsleep TotalTimeInBed   
## Min. : 0 Min. :1.000 Min. : 58.0 Min. : 61.0   
## 1st Qu.:1828 1st Qu.:1.000 1st Qu.:361.0 1st Qu.:403.8   
## Median :2134 Median :1.000 Median :432.5 Median :463.0   
## Mean :2304 Mean :1.119 Mean :419.2 Mean :458.5   
## 3rd Qu.:2793 3rd Qu.:1.000 3rd Qu.:490.0 3rd Qu.:526.0   
## Max. :4900 Max. :3.000 Max. :796.0 Max. :961.0   
## NA's :530 NA's :530 NA's :530

Next, we’ll see what percentage of the users whose data is available are active users. We well give a usage\_type to each unique user ID using SQL

usage\_df <- sqldf("select Id, LightlyActiveMinutes,SedentaryMinutes, FairlyActiveMinutes,VeryActiveMinutes  
 from activity ")  
  
  
usage\_df <- usage\_df %>%   
 mutate(total\_active\_mins =LightlyActiveMinutes+FairlyActiveMinutes+VeryActiveMinutes,   
 total\_mins= total\_active\_mins + SedentaryMinutes  
 )  
  
usage\_df <- sqldf("select Id,   
 avg(total\_active\_mins) as avg\_total\_mins  
 from usage\_df   
 group by Id ")  
  
 mean(usage\_df$avg\_total\_mins) # = 225.090

## [1] 225.0903

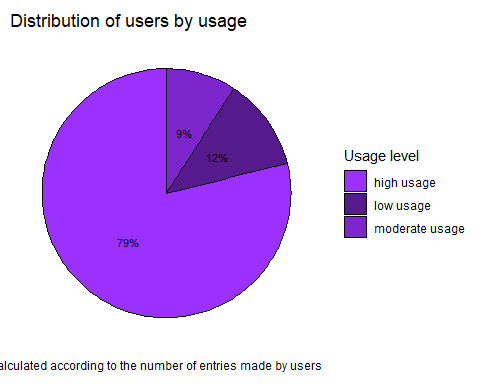
quantile(usage\_df$avg\_total\_mins, probs = 0.9) #=312.6194

## 90%   
## 312.6194

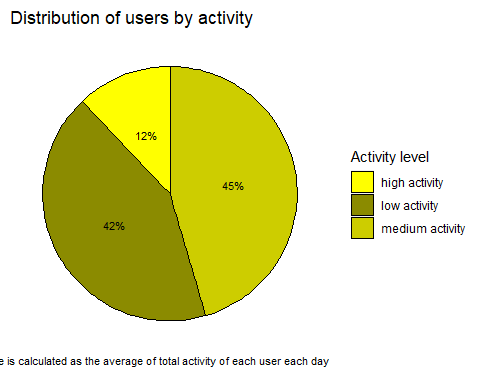
# We use quantile() function to find the 90th percentile of avg\_total\_mins i.e average of active minutes for data available for each user.   
  
  
# User data by active minutes. We will later use this calculation to plot a pie chart to visualise the findings for the stakeholders.   
   
user\_activity\_type <- sqldf("select Id, avg\_total\_mins,   
  
 case  
   
 when avg\_total\_mins < 225.090 then 'low activity'  
   
 when avg\_total\_mins > 225.090 and avg\_total\_mins < 312.6194 then 'medium activity'  
   
 when avg\_total\_mins > 312.6194 then 'high activity'  
 end as user\_activity\_type   
   
 from usage\_df   
   
 group by Id  
 order by user\_activity\_type")  
  
  
# Now we will find user data by logging activity. We will find how many entries were made by each user in the given time period . This will enable us to see how much the customers use the device - we will sort users by usage\_type ( fairly active, not active, very active ).  
  
usage\_type <- activity %>%   
 select(Id) %>%  
 group\_by(Id) %>%   
 summarise(num\_of\_entries = n()) %>%  
 mutate(usage\_type = case\_when(  
 num\_of\_entries >= 1 & num\_of\_entries <= 21 ~ "low usage",  
 num\_of\_entries >= 22 & num\_of\_entries <= 28 ~ "moderate usage",   
 num\_of\_entries >= 28 ~ "high usage")) %>%   
 arrange(usage\_type)  
  
# value of 10th percentile of num\_of\_entries data is 21.20 and average of values is 28.48 Hence the choice of num\_of\_entries to segregate users.   
  
# Further, we can create pie charts to illustrate the distribution of users by activity and usage.   
  
# Chart to show the distribution of users by usage.  
  
## First we'll have to create a usage\_dist data frame to store the usage data in a format that is appropriate for creation of pie charts.   
  
usage\_dist <- usage\_type %>%   
 group\_by(usage\_type) %>%  
 summarise(users = n\_distinct(Id)) %>%   
 mutate(percent\_dist = (users/sum(users))\*100) %>%   
 arrange(percent\_dist)  
  
## Now create a plot  
  
install.packages("ggplot2")

## Warning: package 'ggplot2' is in use and will not be installed

library(ggplot2)  
  
ypos = cumsum(usage\_dist$percent\_dist) - 0.5 \* (usage\_dist$percent\_dist)  
  
  
usage\_pie <- ggplot( data = usage\_dist, aes(x = "", y = percent\_dist, fill = usage\_type)) +   
 geom\_bar(stat="identity", width=1, color="black") +  
 coord\_polar("y", start= 0) +  
 geom\_text(aes(x = "", y = ypos, label = paste0(round(percent\_dist), "%")), size = 3)+  
 labs(title="Distribution of users by usage", caption = " Usage is calculated according to the number of entries made by users") +   
 theme(  
 panel.background = element\_blank(),  
 axis.line = element\_blank(),  
 axis.text = element\_blank(),  
 axis.title = element\_blank(),  
 axis.ticks =element\_blank())+  
 guides(fill = guide\_legend(title = "Usage level")) +  
 scale\_fill\_manual(values=c("purple1", "purple4", "purple3"))  
   
usage\_pie

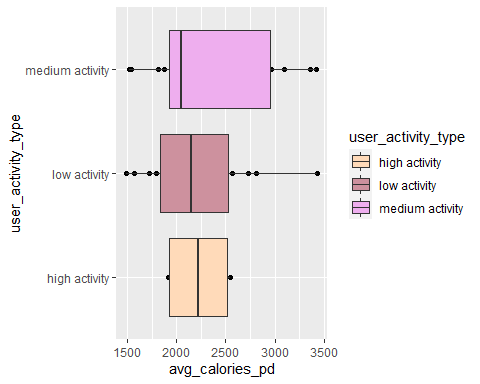
 Now we’ll plot a pie chart to show distribution of users by activity.

# Transform activity data into a form from which pie chart can be created.  
activity\_dist <- user\_activity\_type %>%   
 group\_by(user\_activity\_type) %>%  
 summarise(users = n\_distinct(Id)) %>%   
 mutate(activity\_dist = (users/sum(users))\*100) %>%   
 arrange(activity\_dist)  
  
ypos2 = cumsum(activity\_dist$activity\_dist) - 0.5\*(activity\_dist$activity\_dist)  
  
ypos2 = 100- ypos2  
  
activity\_pie <- ggplot( data = activity\_dist, aes(x = "", y = activity\_dist, fill = user\_activity\_type)) +   
 geom\_bar(stat="identity", width=1, color="black") +  
 coord\_polar("y", start= 0) +  
 geom\_text(aes(x = "", y = ypos2, label = paste0(round(activity\_dist), "%")), size = 3)+  
 labs(title="Distribution of users by activity", caption = "Activity type is calculated as the average of total activity of each user each day") +   
 theme(  
 panel.background = element\_blank(),  
 plot.caption = element\_text(angle = 0, vjust = 5, size = 8),  
 axis.line = element\_blank(),  
 axis.text = element\_blank(),  
 axis.title = element\_blank(),  
 axis.ticks =element\_blank())+  
 guides(fill = guide\_legend(title = "Activity level")) +  
 scale\_fill\_manual(values=c("yellow1", "yellow4", "yellow3"))  
   
activity\_pie

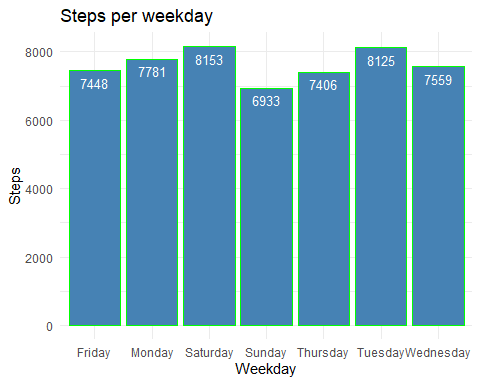


Now we wish to find daily trends in activity, intensity and sleep. For this, we will create a new data frame for all the daily data.

daily\_data <- left\_join(activity,sleep, by = c("Id", "date"))  
  
calories\_per\_user <- sqldf("select Id,   
 avg(calories) as avg\_calories\_pd  
 from calories   
 group by Id")  
  
activity\_calories <- left\_join(user\_activity\_type,calories\_per\_user, by = c("Id"))  
  
  
  
# Now we will use the above data frame to plot calories burnt according to usertype.   
  
activity\_calories\_plot<- ggplot(data = activity\_calories, aes(avg\_calories\_pd,user\_activity\_type, fill = user\_activity\_type))+  
 geom\_point(stat = "identity")+  
 geom\_boxplot(stat = "boxplot")+  
 scale\_fill\_manual(values=c("peachpuff1", "pink3", "plum2"))  
   
  
activity\_calories\_plot

 Now we will plots the steps taken by users per week day. We will also plot average of active minutes per weekday to get an idea of which week days are the most active.

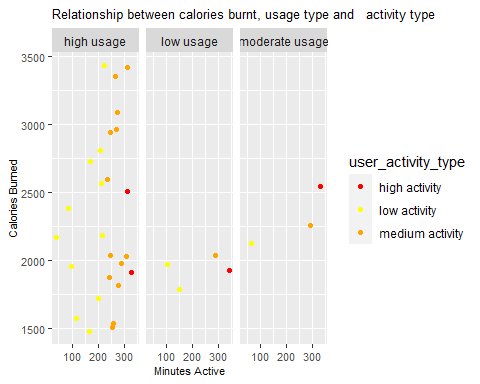
#plotting steps taken by users pper weekday.   
# We will use the daily\_data table to create another daily\_data\_weekdays table for this analysis.   
  
daily\_data\_weekdays <- daily\_data %>%   
 mutate(weekday = weekdays(date)) %>%  
 group\_by(weekday) %>%   
 summarize(average\_steps= mean(TotalSteps))  
  
# Now we use ggplot2 to plot weekdays Vs steps   
  
weekdays\_steps <- ggplot(daily\_data\_weekdays, aes(x=weekday,y=average\_steps))+ geom\_bar(stat = "identity", color= "green", fill = "steelblue") + theme\_minimal() + geom\_text(aes(label= round(average\_steps)), vjust=1.6, color="white", size=3.5)+  
labs(title="Steps per weekday", x="Weekday", y = "Steps")  
  
weekdays\_steps

 The plot shows that most steps were taken on the weekend.

Now we find the relationship between activity, calories burnt and usertype. For this, we must first create a data frame in which all the required data is present.

We join activity\_calories table with usage\_table table using sql query.

activity\_cal\_usage <- left\_join(activity\_calories,usage\_type, by = "Id")  
  
# Now we isolate only the required columns from activity\_cal\_usage using sql   
  
activity\_cal\_usage <- sqldf("select id, avg\_total\_mins, usage\_type,user\_activity\_type, avg\_calories\_pd  
 from activity\_cal\_usage")  
  
# Now we can plot to find the relationship between relationship between activity, calories burned, and user type  
  
  
  
activity\_cal\_usage\_plot<- ggplot(activity\_cal\_usage , aes(avg\_total\_mins , avg\_calories\_pd,colour = user\_activity\_type))+geom\_point()+  
 facet\_wrap(~usage\_type)+   
 theme(plot.title = element\_text(size = 10),  
 plot.subtitle = element\_text(size = 8),  
 plot.caption = element\_text(size = 8),  
 axis.title = element\_text(size = 8),  
 axis.text.x = element\_text(size = 8),  
 axis.text.y = element\_text(size = 8))+  
 xlab("Minutes Active")+  
 ylab("Calories Burned")+  
 labs(titles = "Relationship between calories burnt, usage type and activity type ") +   
 scale\_color\_manual(values=c("red","yellow", "orange"))  
   
activity\_cal\_usage\_plot

 Inference : From this plot, we can say that : 1) In the case of high usage members, highly active members burn less calories than morderately active members. This could mean that although their activity level is high, their intensity could be low. Thus, they burn less calories. Thus, more data must be collected to evaluate this trend further. 2) We find an obvious trend in morderate users of the device. In this facet, we can see that calories burnt increase with their activity type.

However, an important point to consider is that the data in consideration here is only for 33 people. Hence, these trends may be based on biased data.

## Act phase

The following reccomendations can be made to stakeholders based on our analysis :

1. Only 24 % users provided weight data so providing weight data must be made mandatory to all users because it also affects the analysis of other health- related data for the benefit of user. Information about weight can enable Bellabeat app to recommend personilised fitness paths to follow to its users.
2. Bellabeat can invest in R&D for sleep tracking in particular because a vast majority (73%) of users show an interest in sleep tracking.They can also motivate users to track their sleep and provide them more insights on their sleep Vs activity level.
3. Include more default tracking metrics in its line of devices to gather more, useful data about user fitness.
4. We find that most of the users in the study were moderate users i.e the number of entries available for them was moderate as compared to other users. Thus bellabeat app must prompt and motivate users to track their usage more for better health.
5. Bellabeat could group their users according to their activity-level by providing badges and recommend fitness data accordingly. They could also track consider the intensity of their activity to make the recommendations more personalized and accurate.
6. According to our analysis, on an average, users take the least steps on Sundays. Thus, Bellabeat can prompt users to engage in more activity on Sundays. Further, the app can also prompt users when their daily step total is less than the recommended 10,000 steps.