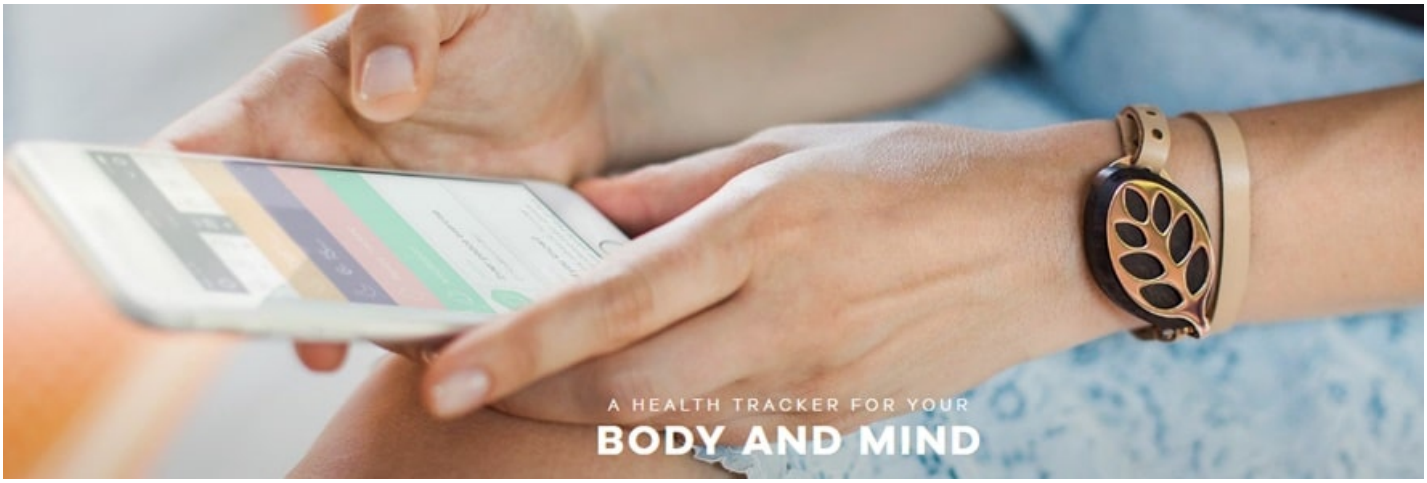
Capstone Project | Bellabeat Data Analysis

Code

Alicia Alkire

09 May, 2023



Bellabeatis a successful small business who specializes in health-focused smart devices for women. With their innovative smart device design, they have potential to become a larger player in the global market.

1 Ask

1.1 Business task

The stakeholders at Bellabeat would like to identify new opportunities for growth. Given this task, I will be conducting an analysis based on the information of its users and how they are utilizing the products offered by the company. I then will make useful recommendations to unlock new marketing strategies based on trends within the data.

1.2 Key stakeholders

- *Urška Sršen*: Bellabeat's co-founder and Chief Creative Officer.
- *Sando Mur*: Mathematician and Bellabeat's co-founder.
- *Bellabeat marketing analytics team.*

---

2 Prepare

##About the data:

The data used for this analysis is public and accessible by everyone. The source of the data is reliable as it was provided by FitBit Fitness Tracker, which is a well known entity.

To download the data, click here (https://www.kaggle.com/datasets/arashnic/fitbit).

Limitations: + The data is from 2016. FitBit devices have likely changed over time to deliver more accurate results. + This data was collected through survey which may indicate that the results are not accurate, as surveys rely on participants providing honest answers. + The sample size is relatively small, which can lead to potential sampling bias. Furthermore, since the target audience is specifically women, we should factor this unreliability into any business decision as a result from this analysis.

2.1 Installing packages 2.2 Loading libraries 2.3 Daily activity 2.4 Daily sleep

2.5 Hourly steps 2.6 Weight

To begin, start by installing the necessary packages and loading them into the library for easy analysis.

Hide

```
# installing packages.

install.packages("tidyverse")
install.packages("kableExtra")
install.packages("highcharter")
install.packages("RColorBrewer")
install.packages("scales")
install.packages("cowplot")
install.packages("plotly")
install.packages("dplyr")
```

3 Process

3.1 Viewing the datasets

3.1.1 Daily activity 3.1.2 Daily sleep 3.1.3 Hourly steps 3.1.4 Weight

Hide

```
# Viewing the datasets using the "kableExtra" package

kbl(daily_activity[1:5, ], 'html', caption = 'Table 1: Daily activity') %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed")) %>%
  scroll_box(width = "100%")  # Adding a scroll bar
```

Table 1: Daily activity

Id	ActivityDate	TotalSteps	TotalDistance	TrackerDistance	LoggedActivitiesDistance	V
1503960366	4/12/2016	13162	8.50	8.50	0	
1503960366	4/13/2016	10735	6.97	6.97	0	
1503960366	4/14/2016	10460	6.74	6.74	0	
1503960366	4/15/2016	9762	6.28	6.28	0	

Id	ActivityDate	TotalSteps	TotalDistance	TrackerDistance	LoggedActivitiesDistance	V
1503960366	4/16/2016	12669	8.16	8.16	0	

3.2 Cleaning data

While further reviewing the datasets, we can observe that some columns won't be needed. We will remove these variables that aren't useful for our analysis.

Hide

```
# removing unnecessary columns

daily_activity <- daily_activity %>% select(-c(TrackerDistance))
weight <- weight %>% select(-c(WeightKg, Fat, IsManualReport, LogId))
```

I will start by checking unique Id's, removing duplicates, renaming columns for consistency, and splitting the datetime into two columns. I'll also clean up the date columns to make them all match the correct format.

Hide

```
# Checking the number of unique Id's

n_distinct(daily_activity$Id)
```

```
[1] 33
```

Hide

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

```
[1] 24
```

Hide

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

```
[1] 8
```

Hide

```
n_distinct(hourly_steps$Id)
```

```
[1] 33
```

Hide

```
# Checking for duplicates

sum(duplicated(daily_activity))
```

```
[1] 0
```

Hide

```
sum(duplicated(daily_sleep))
```

```
[1] 3
```

Hide

```
sum(duplicated(weight))
```

```
[1] 0
```

Hide

```
sum(duplicated(hourly_steps))
```

```
[1] 0
```

Hide

```
# Remove duplicates and drop null values

daily_sleep <- daily_sleep[!duplicated(daily_sleep),]
```

Hide

```
# Confirming duplicates in sleep table were removed

sum(duplicated(daily_sleep))
```

```
[1] 0
```

Here, I'll focus on formatting, consistent column names, and splitting "Date" and "Time" into their own separate columns.

Hide

```
# split datetime into its own columns "Date", "Time" and rename columns for consistency.

weight <- separate(data = weight, col = Date, into  = c('Date', 'Time'), sep = ' ')

daily_sleep <- separate(data = daily_sleep, col = SleepDay, into  = c('Date', 'Time'), se
         p = ' ')

hourly_steps <- separate(data = hourly_steps, col = ActivityHour, into  = c('Date', 'Tim
         e'), sep = ' ')

colnames(daily_activity)[2]="Date"


# formatting variables.

daily_activity <- daily_activity %>%
  mutate(Date = as_datetime(Date, format = "%m/%d/%Y"))

daily_sleep <- daily_sleep %>%
  mutate(Date = as_datetime(Date,format ="%m/%d/%Y"))

hourly_steps<- hourly_steps %>%
  mutate(Date = as_datetime(Date,format ="%m/%d/%Y"))

weight <- weight %>%
  mutate(Date = as_datetime(Date,format ="%m/%d/%Y"))
```

3.3 Checking variables

Hide

```
# Checking variables
str(daily_activity)
```

```
'data.frame':   940 obs. of  14 variables:
 $ Id                       : num  1.5e+09 1.5e+09 1.5e+09 1.5e+09 1.5e+09 ...
 $ Date                     : POSIXct, format: "2016-04-12" "2016-04-13" "2016-04-14" "201
6-04-15" ...
 $ TotalSteps               : int  13162 10735 10460 9762 12669 9705 13019 15506 10544 981
9 ...
 $ TotalDistance            : 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 ...
```

4 Analyze

I'll set up a common theme for plots in the analysis here before I start. This makes our visuals more streamline and consistent.

Hide

```
# Creating common theme for plots

custom_theme <- function() {
  theme(
    panel.border = element_rect(colour = "black",
                                fill = NA,
                                linetype = 1),
    panel.background = element_rect(fill = "white",
                                    color = 'grey50'),
    panel.grid.minor.y = element_blank(),
    axis.text = element_text(colour = "black",
                             face = "italic",
                             family = "Helvetica"),
    axis.title = element_text(colour = "black",
                              family = "Helvetica"),
    axis.ticks = element_line(colour = "black"),
    plot.title = element_text(size=23,
                              hjust = 0.5,
                              family = "Helvetica"),
    plot.subtitle=element_text(size=16,
                               hjust = 0.5),
    plot.caption = element_text(colour = "black",
                             face = "italic",
                             family = "Helvetica")
  )
}
```

4.1 Smart Device Usage

To really dig into consumer behaviors, we'll need to understand how the devices are being used and for what purposes. We will do this by grouping the datasets daily_activity, daily_sleep, and weight. This will allow us to find the distinct number of participants, the frequencies they're using smart devices, as well as the distribution.

4.1.1 Grouping users

Hide

```
# Grouping daily activity for daily averages of users

daily_activity_user_summary <- daily_activity %>%
 group_by(Id) %>%
 summarise(n = n(), steps = round(mean(TotalSteps)), calories = round(mean(Calories)),
           VeryActiveMinutes = round(mean(VeryActiveMinutes)),
           FairlyActiveMinutes = round(mean(FairlyActiveMinutes)),
           LightlyActiveMinutes = round(mean(LightlyActiveMinutes)),
           SedentaryMinutes = round(mean(SedentaryMinutes)))



# Grouping sleep log by user
daily_sleep_user_summary <- daily_sleep %>%
  group_by(Id) %>%
  summarise(n=n(), averageTimeAsleep = mean(TotalMinutesAsleep), averageTimeInBed = mean
         (TotalTimeInBed))

# Grouping weight log by user
weight_user_summary <- weight %>%
  group_by(Id) %>%
  summarise(n=n(), weight_lbs = mean(WeightPounds), max_weight = max(WeightPounds), min_w
         eight = min(WeightPounds), averageBMI = mean(BMI))

n_distinct(daily_activity_user_summary$Id)
```

```
[1] 33
```

Hide

```
n_distinct(daily_sleep_user_summary$Id)
```

```
[1] 24
```

Hide

```
n_distinct(weight_user_summary$Id)
```

```
[1] 8
```

Participant numbers in this activity are 33 for activity, 24 for sleep, and 8 for weight datasets.

If we categorize the users based on how often they log per day, then group the data based on the categories, we will be able see the total number, and percentage of users in each category. Which will give us a better picture of how the devices are being used.

To do this, I'll start by categorize them into "low", "medium" and "regular" for usage of 0-10, 11-20, and 21-31 days.

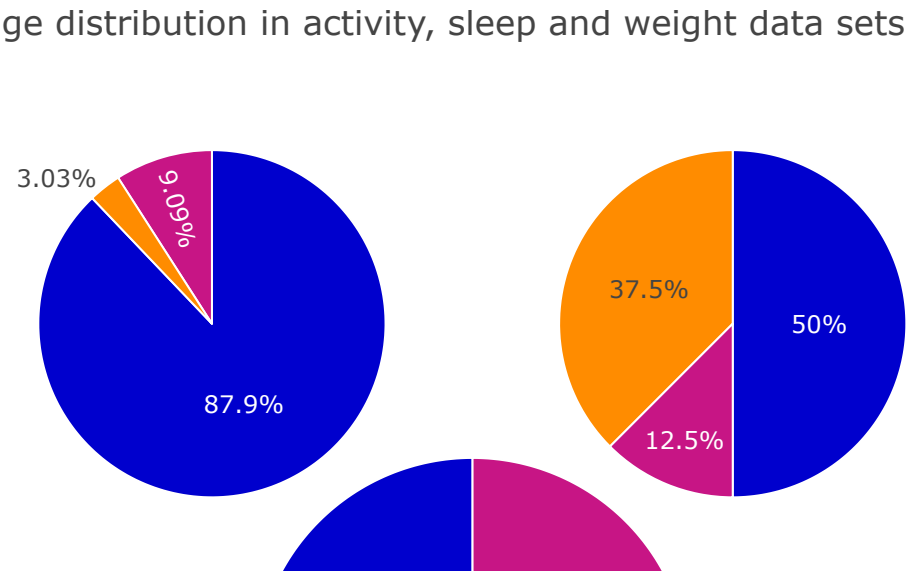
After grouping the data, we'll pull the total number and percentage so we can create a visualization of the distribution by usage for all three types of health data.

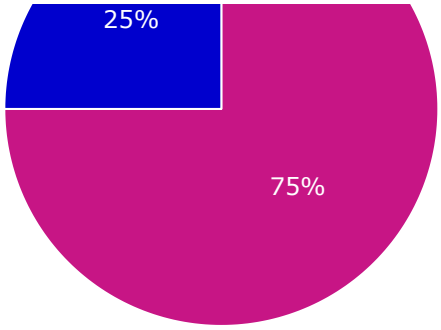
4.1.2 Assigning categories, grouping, sorting and viewing the usage distribution

Hide

```
# Assigning categories to users based on number of days logged

daily_activity_user_summary$Usage <- ifelse(daily_activity_user_summary$n<=10,"Low",
                ifelse(daily_activity_user_summary$n>10 & daily_activity_user_summary$n<2
         1, "Medium",
                       "Regular"))

daily_sleep_user_summary$Usage <- ifelse(daily_sleep_user_summary$n<=10,"Low",
                ifelse(daily_sleep_user_summary$n>10 & daily_sleep_user_summary$n<21, "Me
         dium", "Regular"))

weight_user_summary$Usage <- ifelse(weight_user_summary$n<=10,"Low",
                ifelse(weight_user_summary$n>10 & weight_user_summary$n<21, "Medium", "Re
         gular"))



# Grouping users based on number of days logged

daily_activity_usage <- daily_activity_user_summary %>%
  group_by(Usage) %>%
  summarise(userCount=n(), average_usage= round(mean(n),0))

daily_sleep_usage <- daily_sleep_user_summary %>%
  group_by(Usage) %>%
  summarise(userCount=n(), average_usage= round(mean(n),0))

weight_usage <- weight_user_summary %>%
  group_by(Usage) %>%
  summarise(userCount=n(), average_usage= round(mean(n),0))



# Sorting categories based on the number of users in each category

daily_activity_usage <- daily_activity_usage[order(-daily_activity_usage$average_usage),]
daily_sleep_usage <- daily_sleep_usage[order(-daily_sleep_usage$average_usage),]
weight_usage <- weight_usage[order(-weight_usage$average_usage),]



# Adding percentage column for the number of users in each category

daily_activity_usage$perc <-scales::percent(
  daily_activity_usage$userCount/sum(daily_activity_usage$userCount))
daily_sleep_usage$perc <- scales::percent(
  daily_sleep_usage$userCount/sum(daily_sleep_usage$userCount))
weight_usage$perc <- scales::percent(
```

```
  weight_usage$userCount/sum(weight_usage$userCount))



# Visualizing the representation of usage

colors <- c("mediumblue" , "mediumvioletred","darkorange" )
colors1 <- c("mediumblue", "mediumvioletred","darkorange" )
fig_usage <- plot_ly()
fig_usage <- fig_usage %>% add_pie(data = daily_activity_usage, labels = ~Usage, values =
         ~userCount,
                       name = "Activity", domain = list(x = c(0, 0.4), y = c(0.4, 1)),
                        marker = list(colors = colors, line = list(color = '#FFFFFF', wid
         th = 1)))

fig_usage <- fig_usage %>% add_pie(data = daily_sleep_usage, labels = ~Usage, values = ~u
         serCount,
                       name = "Sleep", domain = list(x = c(0.6, 1), y = c(0.4, 1)),
                       marker = list(colors = colors, line = list(color = '#FFFFFF', widt
         h = 1)),
                       showlegend = TRUE)

fig_usage <- fig_usage %>% add_pie(data = weight_usage, labels = ~Usage, values = ~userCo
         unt,
                       name = "Weight", domain = list(x = c(0.25, 0.75), y = c(0, 0.6)),
                      marker = list(colors = colors1, line = list(color = '#FFFFFF', widt
         h = 1)))

fig_usage <- fig_usage %>% layout(title = "Usage distribution in activity, sleep and weig
         ht data sets", showlegend = F,
         xaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
         yaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE))

fig_usage
```





Hide

```
NA
NA
NA
```

Here we can see that 87.9% of users are regularly logging when it comes to daily activity. We have 9.09% medium, and 3.03% low frequency users.

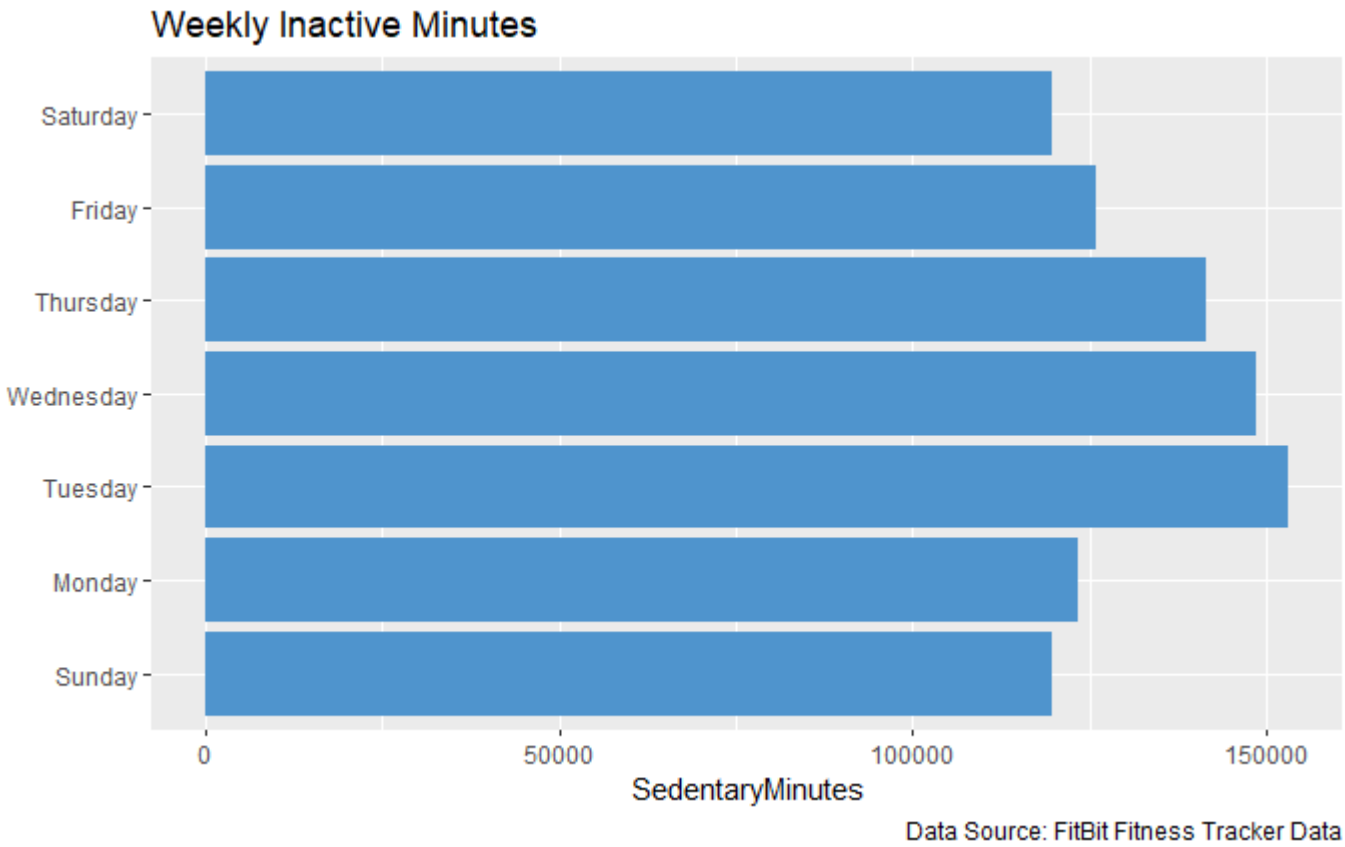
However, if we look at the charts for sleep and weight we can see that they're both being underutilized. Sleep logs show only 50% of users are logging regularly and weight has a mere 25%.

Key takeaway: Users are not logging their sleep and weight as often. Providing the devices with a feature to remind users to log activities would be helpful.

4.1.3 Monitoring Inactivity

Hide

```
# Creating a Weekday category for further evaluation.

Date <- daily_activity %>% distinct(Date)
Date$WeekDay <- weekdays(Date$Date)
daily_activity <- merge(daily_activity, Date, by= "Date")

# mapping our Weekday category with activity logs to see any trends of inactive logging.

ggplot(data=daily_activity)+
  geom_col(mapping=aes(x=SedentaryMinutes,y=factor(WeekDay,levels=c("Sunday","Monday","Tu
         esday", "Wednesday", "Thursday","Friday","Saturday"))),fill='steelblue3')+labs(t
         itle="Weekly Inactive Minutes",y= " ",
        caption = 'Data Source: FitBit Fitness Tracker Data')
```



Users are most inactive Tuesday to Thursday. These days could be spiked up higher due to a normal work week.

Key takeaway: Providing reminders or notifications for inactivity can also help encourage movement when users have been inactive for too long.

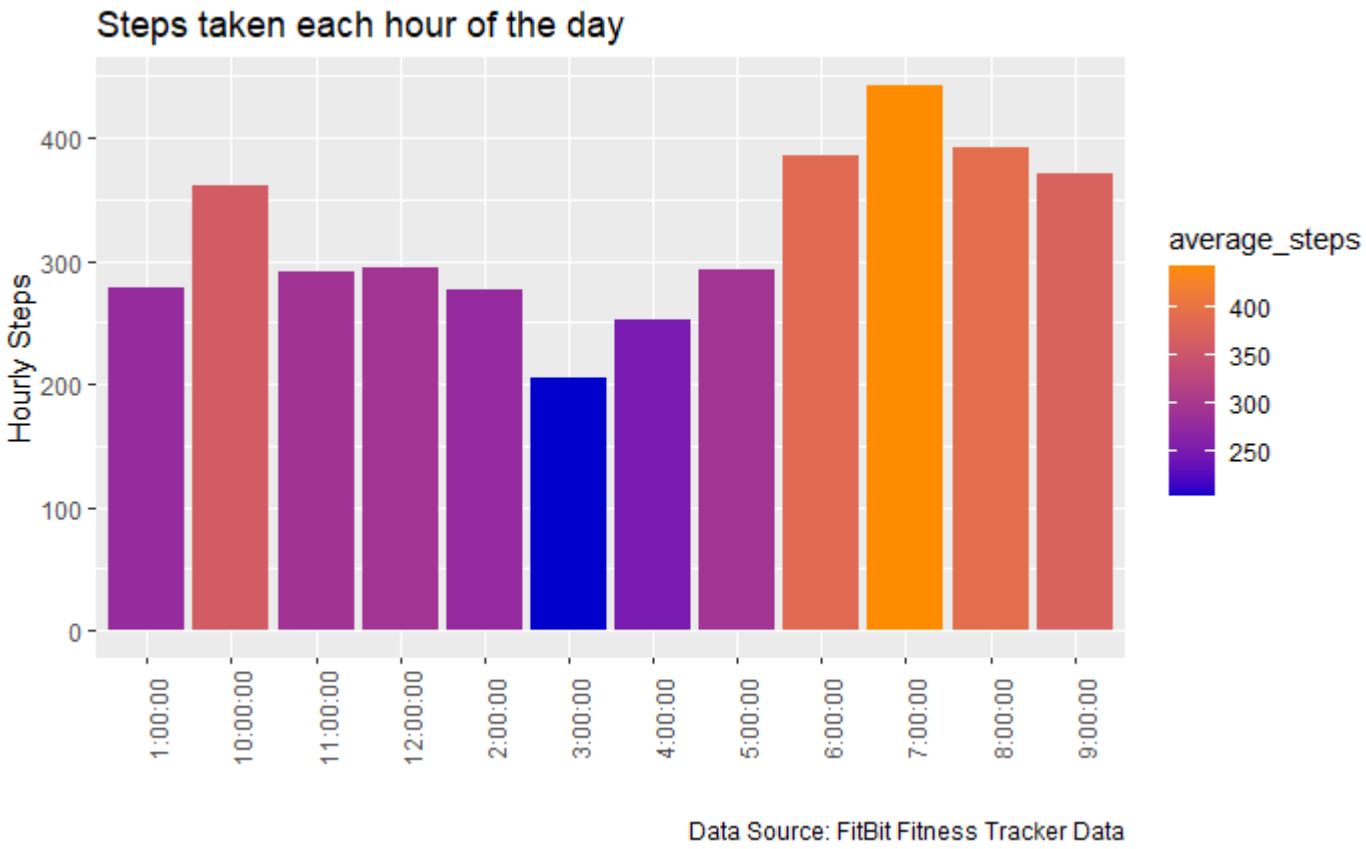
4.2 Steps

4.2.1 Analyse Hourly Steps

Now, let's look at is when users are most active.

Hide

```
# Checking Total Steps vs. Hours of the day.

hourly_steps %>%
  group_by(Time) %>%
  summarize(average_steps = mean(StepTotal)) %>%
  ggplot() +
  geom_col(mapping = aes(x=Time, y = average_steps, fill = average_steps)) +
  labs(title = "Steps taken each hour of the day",
       x= " ",
       y = "Hourly Steps",
       caption = 'Data Source: FitBit Fitness Tracker Data') +
  scale_fill_gradient(low = "mediumblue", high = "darkorange")+
  theme(axis.text.x = element_text(angle = 90))
```



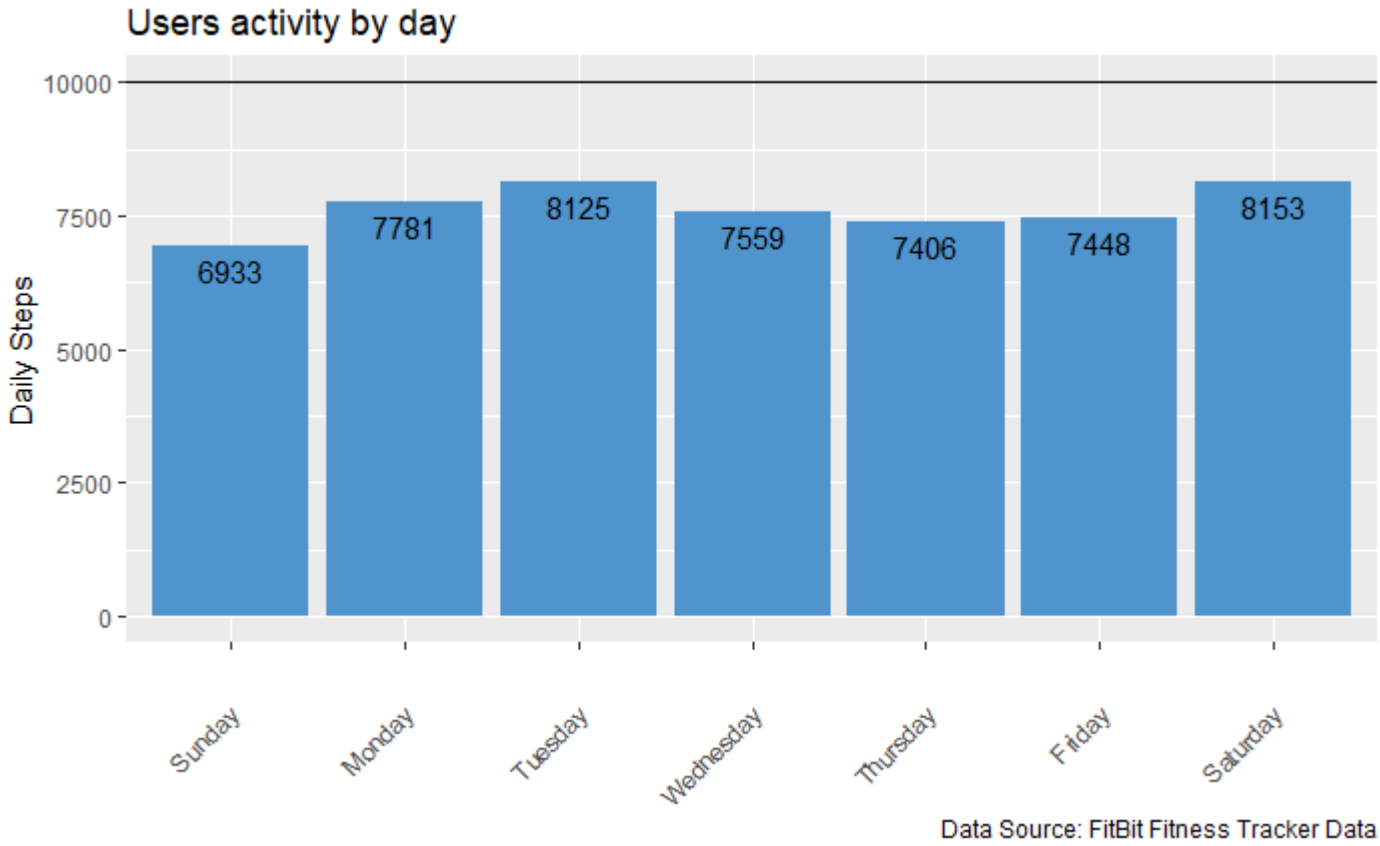
Users are most active between the hours of 5:30p.m.-9:30p.m., and again around 10:00a.m., which is consistent with a modern work schedule.

Key takeaway: It appears that our audience are mostly corporate workers who utilize their lunch and after work hours for movement. Possible recommendation would be articles on how to achieve movement while at work.

4.2.2 Daily step activity

Hide

```
# Checking users' daily activity throughout the week.
weekday_steps <- daily_activity %>%
  mutate(weekday = weekdays(Date))
weekday_steps$weekday <-ordered(weekday_steps$weekday, levels=c("Sunday","Monday", "Tuesd
         ay", "Wednesday", "Thursday","Friday", "Saturday"))

# Grouping by weekday
weekday_steps <-weekday_steps%>%
  group_by(weekday) %>%
  summarize (daily_steps = mean(TotalSteps))

# Visualizing users activity by day
ggplot(weekday_steps,aes(x=weekday, y=daily_steps)) +
  geom_col(fill = "steelblue3") +
  geom_hline(yintercept = 10000) +
  geom_text(data=weekday_steps,aes(label = round(daily_steps), vjust = 1.5))+
labs(title="Users activity by day",
    x= " ",
    y = "Daily Steps",
    caption = 'Data Source: FitBit Fitness Tracker Data')+
  theme(axis.text.x = element_text(angle = 45,vjust = 0.5, hjust = 1))
```



In this bar chart, we can see the difference between days of the week. Noting that **Tuesdays** and **Saturdays** are the best day for users'.

Hide

```
# calculating average steps less than 10,000 daily

round(nrow(daily_activity_user_summary[daily_activity_user_summary$steps < 10000,]) / nro
        w(daily_activity_user_summary) * 100)
```

```
[1] 79
```

79% of our users average below the recommended 10,000 steps a day (per CDC guidelines). Users get the highest amount of steps on Saturdays, and the least on Sundays. During the week their step count is generally lower.

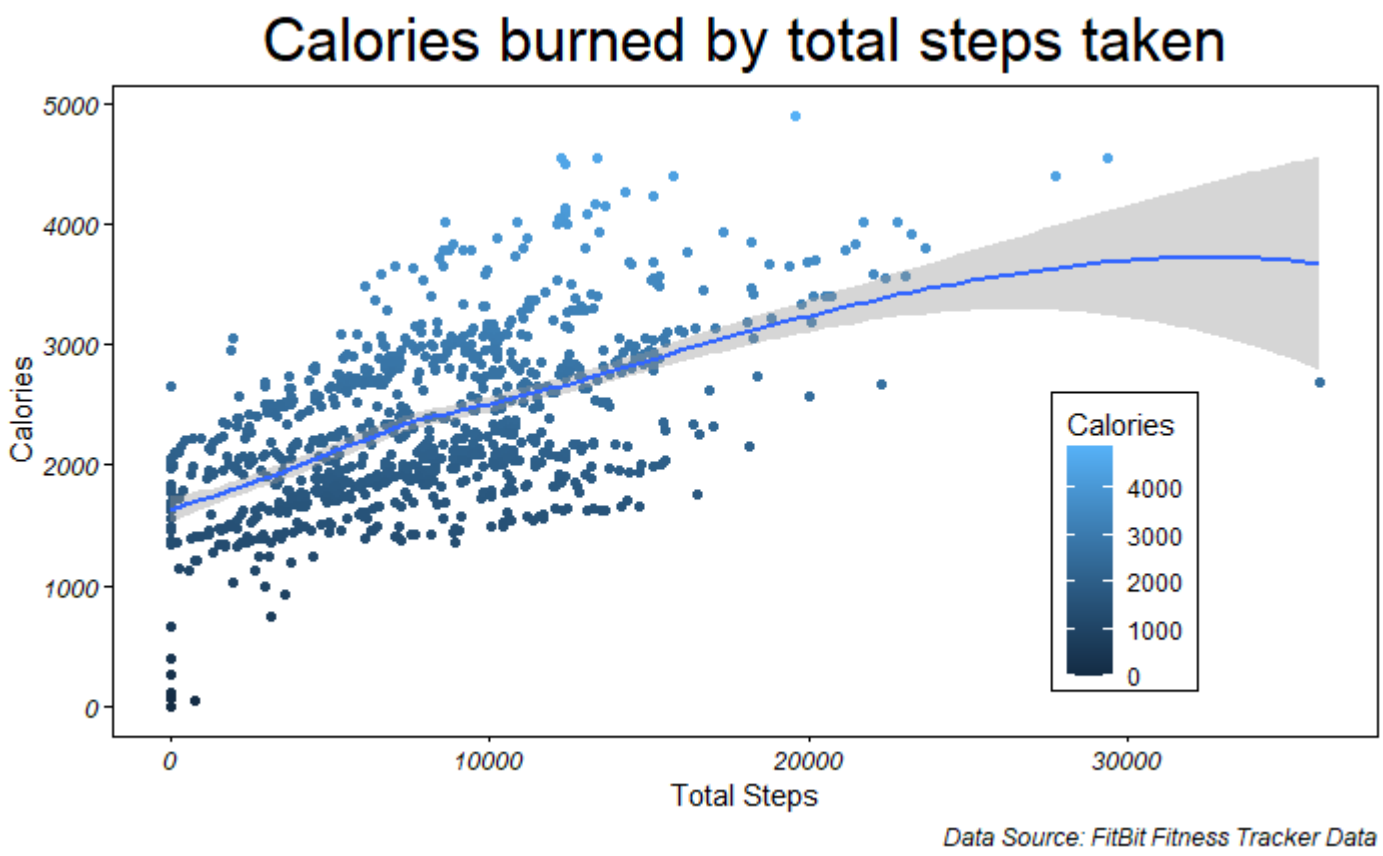
Key takeaway: To help users meet their target step count, Bellabeat can provide notifications and reminders to walk more when inactive and at specific times in the day.

4.3 Calories burned

4.3.1 Calories burned by steps

Hide

```
# Checking Calories burned by Total Steps

daily_activity %>%
  group_by(TotalSteps, Calories) %>%
  ggplot(aes(x = TotalSteps , y = Calories, color = Calories)) +
  geom_point() +
  geom_smooth() +
  custom_theme() +
  theme(legend.position = c(.8, .3),
        legend.spacing.y = unit(1, "mm"),
        panel.border = element_rect(colour = "black", fill=NA),
        legend.background = element_blank(),
        legend.box.background = element_rect(colour = "black")) +
  labs(title = 'Calories burned by total steps taken',
       y = 'Calories',
       x = 'Total Steps',
       caption = 'Data Source: FitBit Fitness Tracker Data')
```



In the above graph, we can clearly see a positive correlation between Steps and Calories burned. To check the correlation, we will use the Pearson Correlation Coefficient which will measure the linear correlation between the two variables. To read more, click here (https://en.wikipedia.org/wiki/Pearson_correlation_coefficient).

Hide

```
# Running the Pearsons Correlation Coefficient.

cor.test(daily_activity$TotalSteps, daily_activity$Calories, method = 'pearson', conf.lev
         el = 0.95)
```

```
    Pearson's product-moment correlation

data:  daily_activity$TotalSteps and daily_activity$Calories
t = 22.472, df = 938, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.5483688 0.6316184
sample estimates:
      cor
0.5915681
```

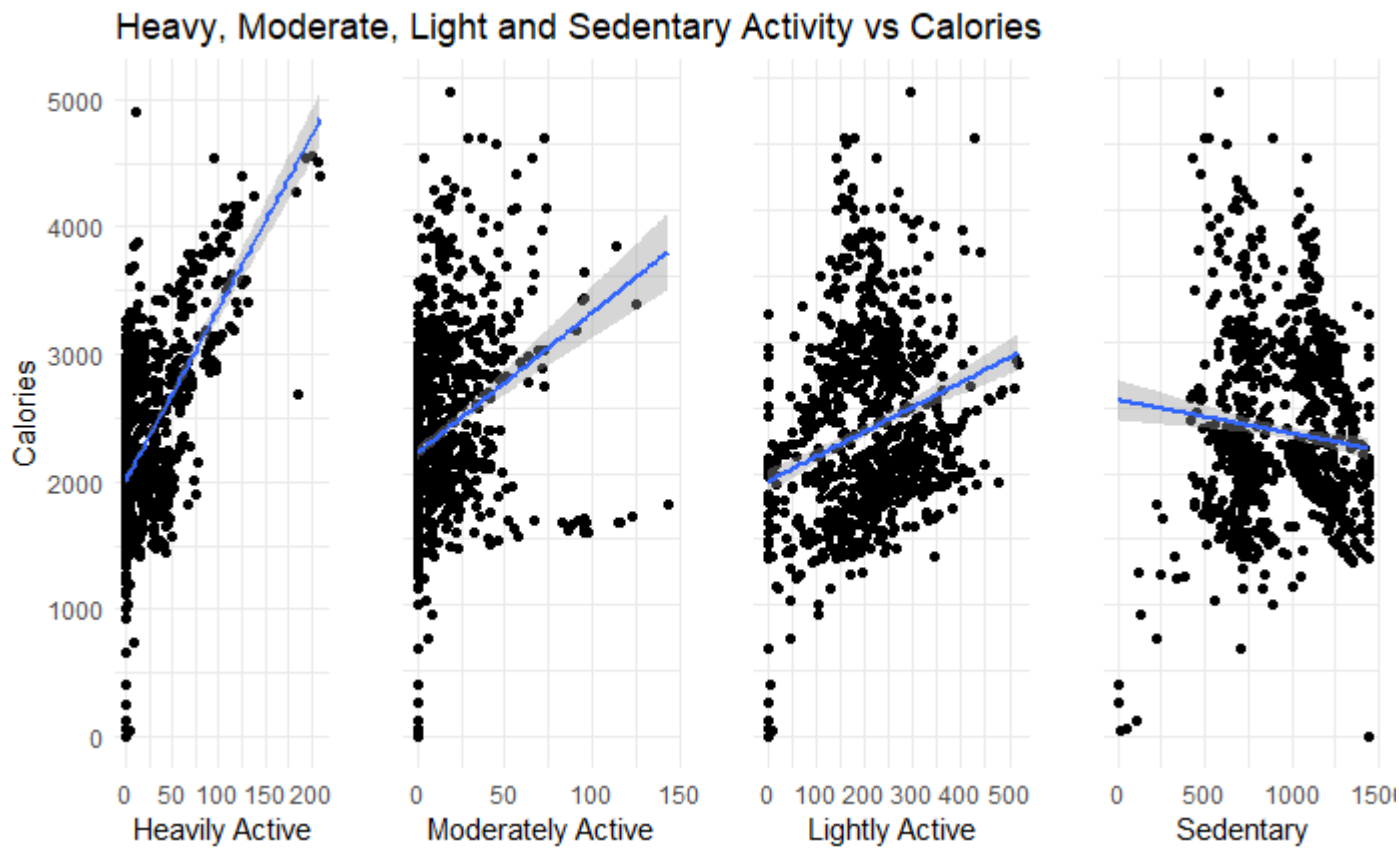
We can see that the correlation is almost 60%. This means that there is a strong relationship between steps and calories burned.

Key takeaway: There is a strong relationship between total steps taken and calories burned. Simply put, the more steps you take the more calories you burn.

4.3.2 Calories Burned by activity level

Hide

```
# Calories Burned vs. Daily activity level

par(mfrow = c(2, 2))

p1 <- ggplot(daily_activity, aes(VeryActiveMinutes, Calories)) + geom_point() + labs(x =
         "Heavily Active", y = "Calories", title = "Heavy, Moderate, Light and Sedentary
         Activity vs Calories", ) + theme_minimal() + geom_smooth(method = "lm")


p2 <- ggplot(daily_activity, aes(FairlyActiveMinutes, Calories)) + geom_point() + labs(x
         = "Moderately Active", y = "", title = " ") + theme_minimal() + theme(axis.text.
         y = element_blank()) + geom_smooth(method = "lm")

p3 <- ggplot(daily_activity, aes(LightlyActiveMinutes, Calories)) + geom_point() + labs(x
         = "Lightly Active", y = "", title = " ") + theme_minimal() + theme(axis.text.y =
         element_blank()) + geom_smooth(method = "lm")

p4 <- ggplot(daily_activity, aes(SedentaryMinutes, Calories)) + geom_point() + labs(x =
         "Sedentary", y = "", title = " ") + theme_minimal() + theme(axis.text.y = elemen
         t_blank()) + geom_smooth(method = "lm")

plot_grid(p1, p2, p3, p4, ncol = 4)
```



There is a positive correlation between the type of activity and the calories burned. Meaning the more active you are, the more calories you burn, and vice verse.

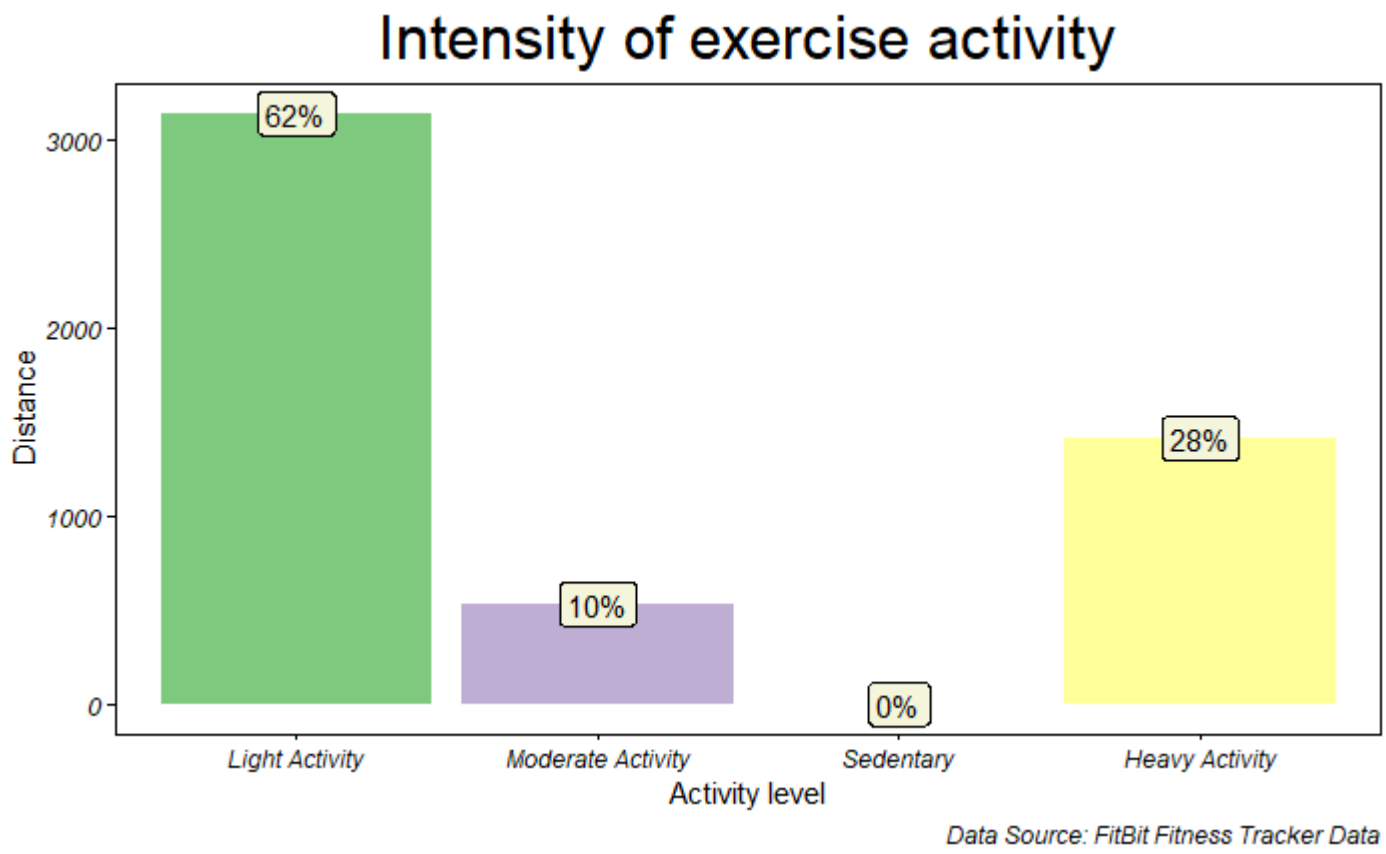
Key takeaway: Staying active will make a huge difference in the amount of calories you burn on a daily basis.

4.4 Exercise

4.4.1 Intensity of exercise activity

Hide

```
# Checking users' activity levels.

daily_activity %>%
  select(VeryActiveDistance,
         ModeratelyActiveDistance,
         LightActiveDistance, SedentaryActiveDistance) %>%
  summarise(across(everything(), list(sum))) %>%
  gather(activities, value) %>%
  mutate(ratio = value / sum(value),
         label = percent(ratio %>% round(4))) %>%
mutate(activities = factor(activities,
                           labels = c('Light Activity','Moderate Activity', 'Sedentary','H
         eavy Activity'))) %>%
  ggplot(aes(x = (activities),
             y = value,
             label = label,
             fill = activities)) +
  geom_bar(stat='identity') +
  geom_label(aes(label = label),
             fill = "beige",
             colour = "black",
             vjust = 0.5) +
  custom_theme() +
  scale_fill_brewer(palette="Accent") +
  theme(legend.position="none") +
  labs(
    title = "Intensity of exercise activity",
    x = "Activity level",
    y = "Distance",
    caption = 'Data Source: FitBit Fitness Tracker Data'
  )
```



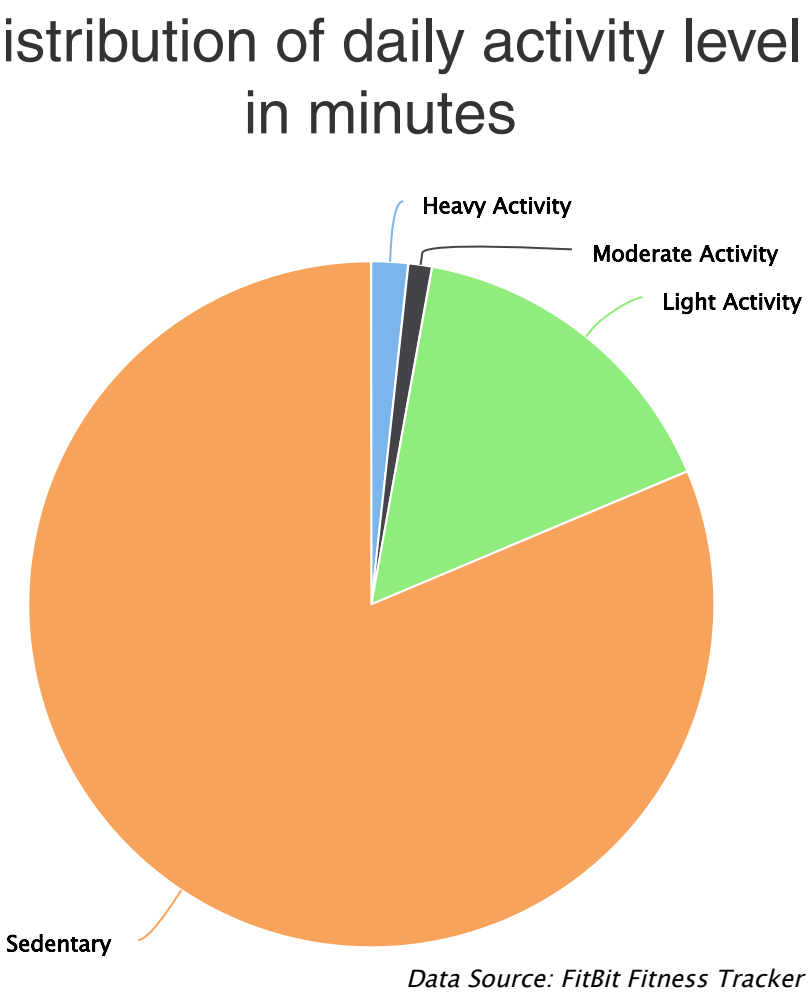
The most common exercise activity level is **light**.

Key takeaway: Users seem to prefer lighter activities. Adding in-app articles or even a guided exercise program could bring more engagement.

4.4.2 Distribution of daily activity level

Hide

```
# Checking the Distribution % of daily activity levels.

daily_activity %>%
  select(VeryActiveMinutes,
         FairlyActiveMinutes,
         LightlyActiveMinutes,
         SedentaryMinutes) %>%
  summarise(across(everything(), list(sum))) %>%
  gather(active_level, minutes) %>%
  mutate(active_level = factor(active_level,
                          labels = c('Moderate Activity','Light Activity',
                                     'Sedentary','Heavy Activity'))) %>%
  hchart('pie', hcaes(x = active_level, y = minutes)) %>%
  hc_title(text = "Distribution of daily activity level in minutes",
           style = list(fontFamily = "Helvetica", fontSize = "30px"),
           align = "center") %>%
  hc_tooltip(pointFormat = "<b>Value:</b> {point.y} <br>
                 <b>Percentage</b> {point.percentage:,.2f}%") %>%
  hc_credits(
    enabled = TRUE,
    text = "<em>Data Source: FitBit Fitness Tracker Data</em></span>",
    style = list(fontSize = "12px", color = 'black',type = "spline")
    )
```



Data Source: FitBit Fitness Tracker Data

Above we can see that the majority of people are **sedentary** during the day.

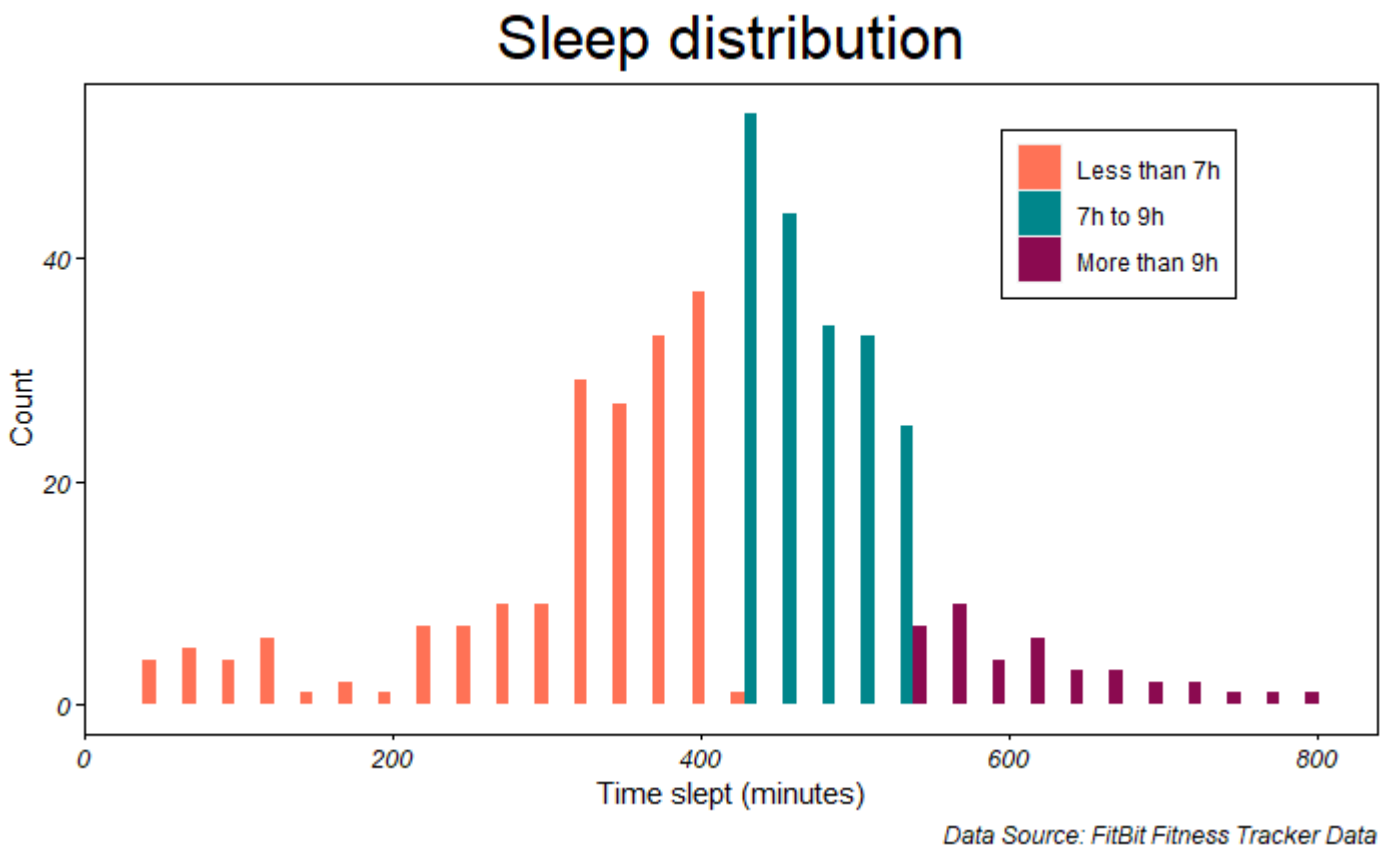
Key takeaway: Movement reminders would help reduce this number.

4.5 Sleep

4.5.1 Sleep distribution

Hide

```
# Distribution sleep time.

daily_sleep %>%
  select(TotalMinutesAsleep) %>%
  drop_na() %>%
  mutate(sleep_quality = ifelse(TotalMinutesAsleep <= 420, 'Less than 7h',
                         ifelse(TotalMinutesAsleep <= 540, '7h to 9h',
                         'More than 9h'))) %>%
  mutate(sleep_quality = factor(sleep_quality,
                          levels = c('Less than 7h','7h to 9h',
                                     'More than 9h'))) %>%
  ggplot(aes(x = TotalMinutesAsleep, fill = sleep_quality)) +
  geom_histogram(position = 'dodge', bins = 30) +
  custom_theme() +
  scale_fill_manual(values=c("coral1", "turquoise4", "deeppink4")) +
  theme(legend.position = c(.80, .80),
        legend.title = element_blank(),
        legend.spacing.y = unit(0, "mm"),
        panel.border = element_rect(colour = "black", fill=NA),
        legend.background = element_blank(),
        legend.box.background = element_rect(colour = "black")) +
  labs(
    title = "Sleep distribution",
    x = "Time slept (minutes)",
    y = "Count",
    caption = 'Data Source: FitBit Fitness Tracker Data'
  )
```



We can see by this graph that users sleep averages to about 300 to 520 minutes (or 5 to 8 hours) of sleep each day.

Key takeaway: according to the CDC guidelines, users should be getting approximately 7-9 hours of sleep each night. Users are not quite meeting this recommended goal. Developing an in-app sleep routine or schedule would help users meet their sleep goals. Articles about sleep routines would be helpful as well.

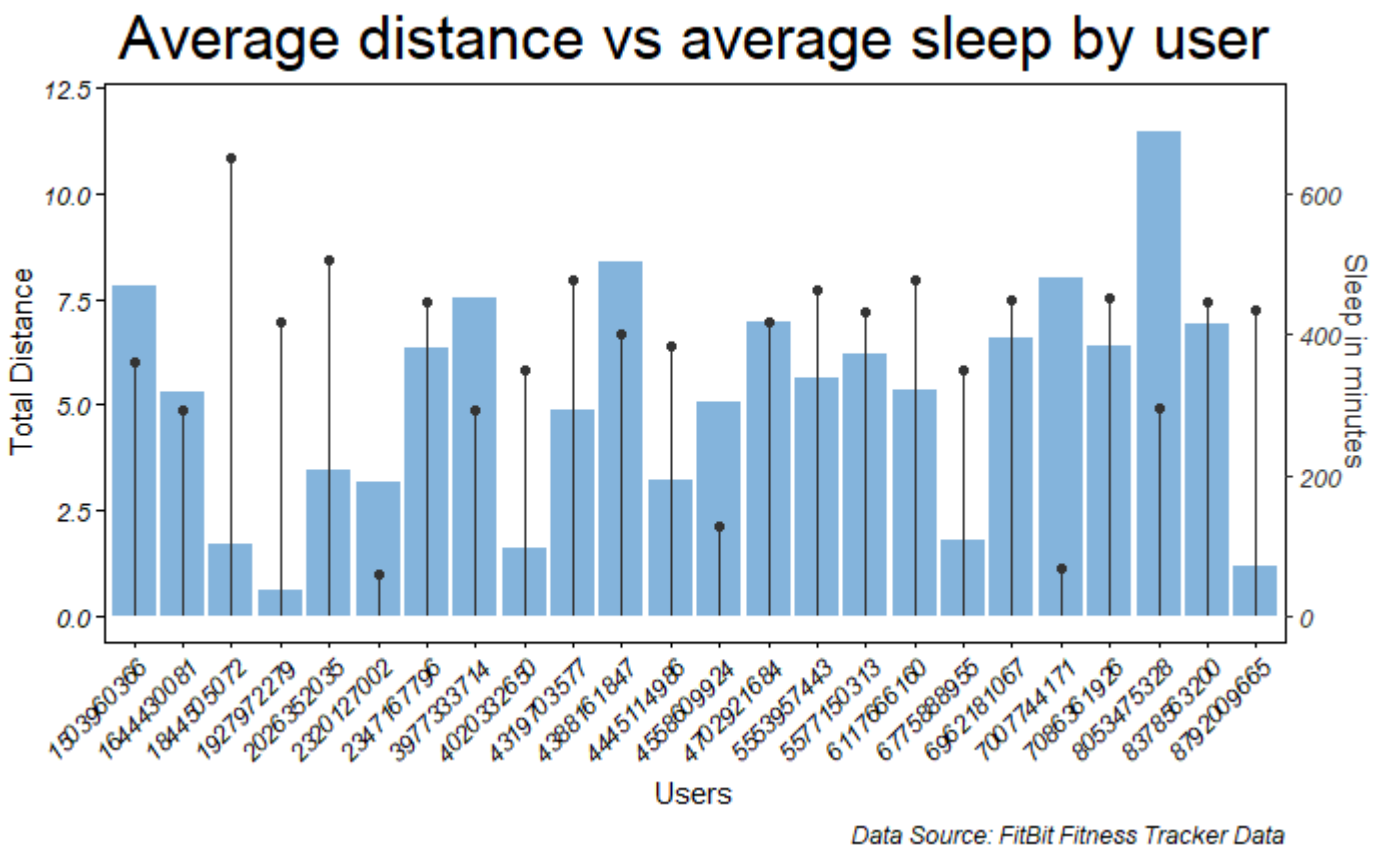
4.5.2 Sleep vs distance covered

Hide

```
# Merging daily activity and daily sleep for sleep analysis

daily_sleep_activity <- merge(daily_activity, daily_sleep, by = "Id")
```

Hide

```
# Checking Sleep vs. Distance Covered.

daily_sleep_activity %>%
  select(Id, TotalDistance, TotalMinutesAsleep) %>%
  group_by(Id) %>%
  summarise_all(list(~mean(., na.rm=TRUE))) %>%
  drop_na() %>%
  mutate(Id = factor(Id)) %>%
  ggplot() +
  geom_bar(aes(x = Id, y = TotalDistance), stat = "identity", fill = 'steelblue3', alpha
         = 0.7) +
  geom_point(aes(x = Id, y = TotalMinutesAsleep/60), color = 'gray20') +
  geom_segment(aes(x = Id, xend = Id, y = 0, yend = TotalMinutesAsleep/60), color = 'gray
         20' ,group = 1) +
  custom_theme() +
  scale_y_continuous(limits=c(0, 12), name = "Total Distance",
    sec.axis = sec_axis(~.*60, name = "Sleep in minutes")) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  theme(axis.title.y.right = element_text(color = "gray20"),
        axis.ticks.y.right = element_line(color = "gray20"),
        axis.text.y.right = element_text(color = "gray20")) +
  labs(
    title = "Average distance vs average sleep by user",
    x = "Users",
    caption = 'Data Source: FitBit Fitness Tracker Data'
  )
```



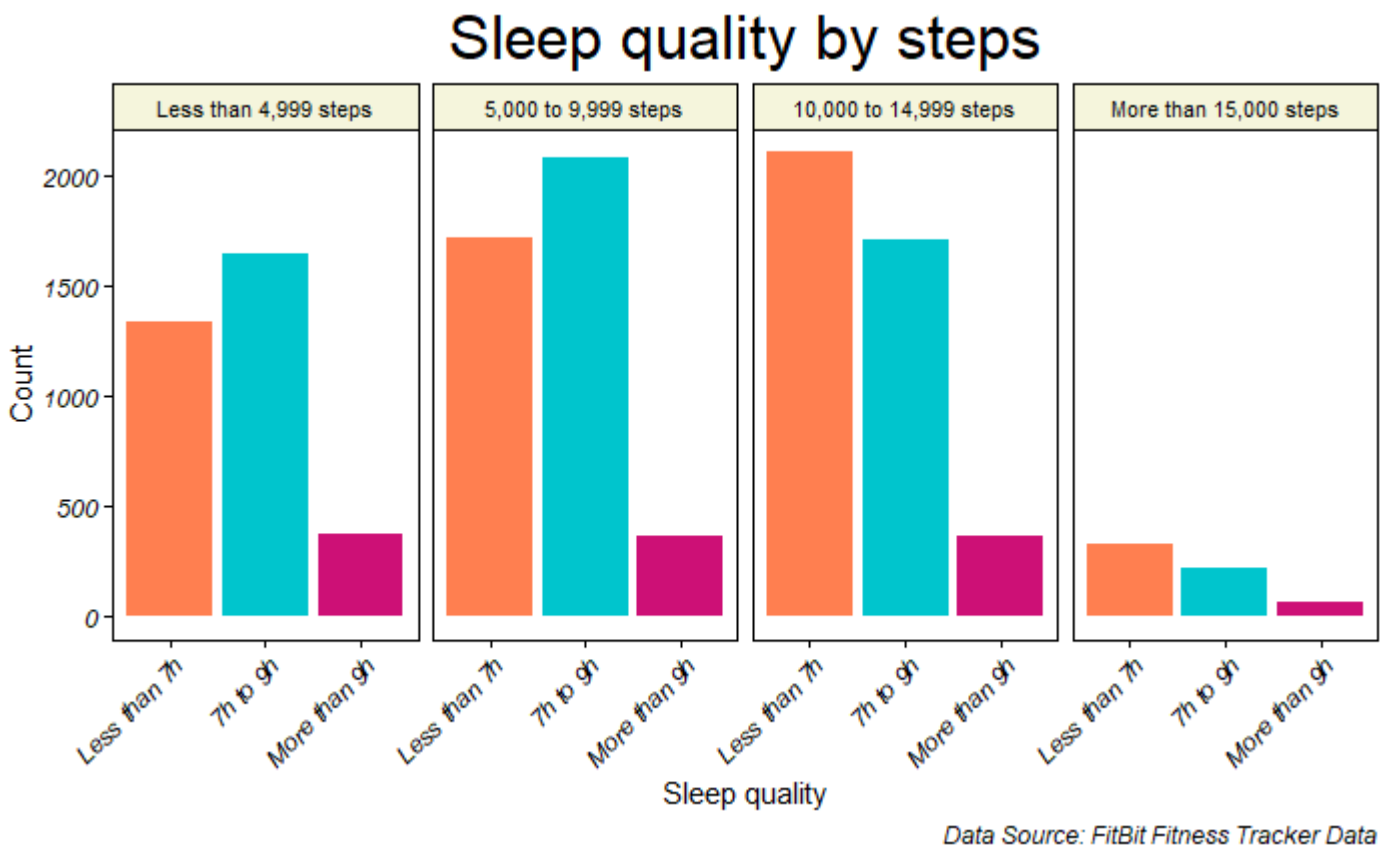
We can see that moving a greater distance doesn't equate to a better nights sleep (on average).

Let's look into this further by checking the sleep quality vs step totals.

4.5.3 Sleep quality vs Step totals

Hide

```
# Checking Sleep Quality vs. Step Totals.

daily_sleep_activity %>%
  select(TotalMinutesAsleep, TotalSteps) %>%
  mutate(sleep_quality = ifelse(TotalMinutesAsleep <= 420, 'Less than 7h',
                         ifelse(TotalMinutesAsleep <= 540, '7h to 9h',
                         'More than 9h'))) %>%
  mutate(active_level = ifelse(TotalSteps >= 15000,'More than 15,000 steps',
                        ifelse(TotalSteps >= 10000,'10,000 to 14,999 steps',
                        ifelse(TotalSteps >= 5000, '5,000 to 9,999 steps',
                        'Less than 4,999 steps')))) %>%
  select(-c(TotalMinutesAsleep, TotalSteps)) %>%
  drop_na() %>%
  group_by(sleep_quality, active_level) %>%
  summarise(counts = n()) %>%
  mutate(active_level = factor(active_level,
                               levels = c('Less than 4,999 steps',
                                          '5,000 to 9,999 steps',
                                          '10,000 to 14,999 steps',
                                          'More than 15,000 steps'))) %>%
  mutate(sleep_quality = factor(sleep_quality,
                         levels = c('Less than 7h','7h to 9h',
                                    'More than 9h'))) %>%
  ggplot(aes(x = sleep_quality,
             y = counts,
             fill = sleep_quality)) +
  geom_bar(stat = "identity") +
  custom_theme() +
  scale_fill_manual(values=c("coral", "turquoise3", "deeppink3")) +
  facet_wrap(~active_level, nrow = 1) +
  theme(legend.position = "none") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  theme(strip.text = element_text(colour = 'black', size = 8)) +
  theme(strip.background = element_rect(fill = "beige", color = 'black'))+
  labs(
    title = "Sleep quality by steps",
    x = "Sleep quality",
    y = "Count",
    caption = 'Data Source: FitBit Fitness Tracker Data'
  )
```



Here we can see that the best sleep on average is achieved when the users total sleep does not exceed 9,999 steps a day.

Key takeaway: Users sleep quality is the best when they stay within their step goals.

5 Share

5.1 Key takeaways

logging:

1. Users are not logging their sleep and weight as often. Providing the devices with a feature to remind users to log activities would be helpful.

Movement:

1. Providing reminders or notifications for inactivity can help encourage movement when users have been inactive for too long.
2. To help users meet their target step count, Bellabeat can provide notifications and reminders to walk more when inactive and at specific times in the day.
3. In order to create a habit of exercising every day, Bellabeat could send a notification at a specific time for the user to remain consistent throughout the week. Reminding users that even light activity, is still activity.

Calories:

1. Calories are burnt by the steps taken daily. Based on users' objectives, Bellabeat could recommend a minimum number of steps for users to take to encourage them to achieve their goals throughout the day.

Activities:

1. Users seem to prefer lighter activities. Adding in-app articles or even a guided exercise program could bring more engagement.

Sleep:

1. According to the CDC guidelines, users should be getting approximately 7-9 hours of sleep each night. Users are not quite meeting this recommended goal. Developing an in-app sleep routine or schedule would help users meet their sleep goals.
2. Studies have shown that a routine before bed can help achieve a better sleep. Bellabeat could send a reminder to start that routine at a certain time and recommend activities to relax and improve sleep. Articles about sleep routines would be helpful as well.
3. The data shows that in order to sleep better the best type of exercise is light to moderate (less than 10,000 steps). Bellabeat could loop this recommendation into the above reminder by encouraging light movement that could better suit their physical needs during the workweek.

Engagement:

1. Bellabeats main focus is to create more engagement with their users by providing a good consumer experience. It is easy to gain a following, but the idea is to maintain one as well and prevent any decrease in monthly users. That's why it is important to inform users how our services will benefit them in the long run. Small things such as congratulating a user for walking a farther distance today compared to yesterday will go a long way.
2. It appears that our audience are mostly corporate workers who utilize their lunch and after work hours for movement and exercise. A possible recommendation would be to add articles on how to achieve movement while at work.
3. A great way Bellabeat could motivate users is also recommending some mindful eating habits and recipe articles within the Bellabeat app.
4. Providing quarterly or annual health reports could prove to encourage ongoing user engagement.
5. A recommended diet plan based off of users BMI or health status can help users be more knowledgeable of their health.
6. 1 month free trials can help more users sign up for this service.
7. Establishing a social media presence can help consumers learn more about what services Bellabeat offers.

5.2 Recomendations

- Logging, inactivity, step goal, exercise and sleep routine reminders and notifications.
- Recommendation on minimum steps throughout the day to encourage users.
- In-app articles on exercise including one specifically geared towards corporate workers.
- In-app guided exercise program.
- In-app sleep routines or schedules.
- Good consumer experiences, such as congratulating users for reaching higher milestones than the previous days.
- Quarterly or annual health reports for members.
- Diet plan based off of users BMI or health report.
- 1 month free trials to bring in more consumers.
- Establishing a social media presence to help consumers learn more about our services.

Thank you for reading. If you would like to view more of my work, you can access my portfolio here ().