Bella beat analysis

Bellabeat Analysis

This is the analysis project about high-tech manufacturer of health products for women. The smart device collects the data and send it to the app on users' smartphones and to the main database of the company. We are going to conduct an analysis using that dataset to identify trends and relationships to target customers with clear marketing strategies.

I already downloaded the dataset so I can use data easily since I am using RStudio. The dataset can be found here on my Kaggle (https://www.kaggle.com/datasets/shokirjonotamirzaev/bellabeat-marketing-analysis).

1. Prepare

We start with installing and loading common packages we will use throughout this project.

Let's check if we have the right directory first.

```
getwd()
```

```
## [1] "C:/my filess/Data-Analysis/Case_study_2_Bellabeat"
```

```
library(ggpubr)
```

```
## Warning: package 'ggpubr' was built under R version 4.2.3
```

```
## Loading required package: ggplot2
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.2.3
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.0     ✔ readr     2.1.4
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ lubridate 1.9.2     ✔ tibble    3.1.8
## ✔ purrr     1.0.1     ✔ tidyr     1.3.0
```

```
## ── Conflicts ─────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the ]8;;http://conflicted.r-lib.org/conflicted package]8;; to force all conflicts to
become errors
```

```
library(here)
```

```
## Warning: package 'here' was built under R version 4.2.3
```

```
## here() starts at C:/my filess/Data-Analysis/Case_study_2_Bellabeat
```

```
library(skimr)
```

```
## Warning: package 'skimr' was built under R version 4.2.3
```

```
library(janitor)
```

```
## Warning: package 'janitor' was built under R version 4.2.3
```

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

```
library(lubridate)
library(ggrepel)
```

```
## Warning: package 'ggrepel' was built under R version 4.2.3
```

Now we create our dataframes. For now we need 3 files: Daily Activity, Daily sleep, and Hourly steps

```
daily_activity<- read.csv("dailyActivity_merged.csv")
daily_sleep<- read.csv("sleepDay_merged.csv")
hourly_steps<- read.csv("hourlySteps_merged.csv")
```

Now we explore the dataframes we uploaded:

```
head(daily_activity)
```

	Id	ActivityDate	TotalSteps	TotalDistance	TrackerDistance	LoggedActivitiesDistance
	<dbl>	<chr>	<int>	<dbl>	<dbl>	<dbl>
1	1503960366	4/12/2016	13162	8.50	8.50	0

	Id	ActivityDate	TotalSteps	TotalDistance	TrackerDistance	LoggedActivitiesDistance
	<dbl>	<chr>	<int>	<dbl>	<dbl>	<dbl>
2	1503960366	4/13/2016	10735	6.97	6.97	0
3	1503960366	4/14/2016	10460	6.74	6.74	0
4	1503960366	4/15/2016	9762	6.28	6.28	0
5	1503960366	4/16/2016	12669	8.16	8.16	0
6	1503960366	4/17/2016	9705	6.48	6.48	0

6 rows | 1-7 of 16 columns

```
colnames(daily_activity)
```

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

```
str(daily_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 ...
##  $ 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 ...
##  $ 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(daily_sleep)
```

	Id	SleepDay	TotalSleepRecords	TotalMinutesAsleep	TotalTimeInBed
	<dbl>	<chr>	<int>	<int>	<int>
1	1503960366	4/12/2016 12:00:00 AM	1	327	346
2	1503960366	4/13/2016 12:00:00 AM	2	384	407
3	1503960366	4/15/2016 12:00:00 AM	1	412	442

	Id	SleepDay	TotalSleepRecords	TotalMinutesAsleep	TotalTimeInBed
	<dbl>	<chr>	<int>	<int>	<int>
4	1503960366	4/16/2016 12:00:00 AM	2	340	367
5	1503960366	4/17/2016 12:00:00 AM	1	700	712
6	1503960366	4/19/2016 12:00:00 AM	1	304	320

6 rows

```
colnames(daily_sleep)
```

```
## [1] "Id"                 "SleepDay"           "TotalSleepRecords"
## [4] "TotalMinutesAsleep" "TotalTimeInBed"
```

```
str(daily_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 ...
```

```
head(hourly_steps)
```

	Id	ActivityHour	StepTotal
	<dbl>	<chr>	<int>
1	1503960366	4/12/2016 12:00:00 AM	373
2	1503960366	4/12/2016 1:00:00 AM	160
3	1503960366	4/12/2016 2:00:00 AM	151
4	1503960366	4/12/2016 3:00:00 AM	0
5	1503960366	4/12/2016 4:00:00 AM	0
6	1503960366	4/12/2016 5:00:00 AM	0

6 rows

```
colnames(hourly_steps)
```

```
## [1] "Id"           "ActivityHour" "StepTotal"
```

```
str(hourly_steps)
```

```
## 'data.frame':    22099 obs. of  3 variables:
##  $ Id          : num  1.5e+09 1.5e+09 1.5e+09 1.5e+09 1.5e+09 ...
##  $ ActivityHour: chr  "4/12/2016 12:00:00 AM" "4/12/2016 1:00:00 AM" "4/12/2016 2:00:00 AM" "4/
12/2016 3:00:00 AM" ...
##  $ StepTotal   : int  373 160 151 0 0 0 0 0 250 1864 ...
```

Understanding some summary stats:

How many unique participants are there in each dataframe? It looks like there may be more participants in the daily activity dataset than the sleep dataset.

```
n_distinct(daily_activity$Id)
```

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

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

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

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

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

How many observations each table contains:

```
nrow(daily_activity)
```

```
## [1] 940
```

```
nrow(daily_sleep)
```

```
## [1] 413
```

```
nrow(hourly_steps)
```

```
## [1] 22099
```

2. Cleaning

Let's check for **duplicates**

```
sum(duplicated(daily_activity))
```

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

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

```
## [1] 3
```

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

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

It checks out that daily_sleep table has 3 duplicates. Thus we get rid of them on the next step

```
daily_sleep<- daily_sleep %>% distinct() %>% drop_na()
sum(duplicated(daily_sleep)) #we verify that duplicates are gone
```

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

Clean names with rename_with

We want column names to have the same syntax in all datasets so we can merge them if necessary

```
clean_names(daily_activity)
```

id	activity_date	total_steps	total_distance	tracker_distance	logged_activities_dista
<dbl>	<chr>	<int>	<dbl>	<dbl>	<
1503960366	4/12/2016	13162	8.50	8.50	0.000
1503960366	4/13/2016	10735	6.97	6.97	0.000
1503960366	4/14/2016	10460	6.74	6.74	0.000
1503960366	4/15/2016	9762	6.28	6.28	0.000
1503960366	4/16/2016	12669	8.16	8.16	0.000
1503960366	4/17/2016	9705	6.48	6.48	0.000
1503960366	4/18/2016	13019	8.59	8.59	0.000
1503960366	4/19/2016	15506	9.88	9.88	0.000
1503960366	4/20/2016	10544	6.68	6.68	0.000
1503960366	4/21/2016	9819	6.34	6.34	0.000

1-10 of 940 rows | 1-6 of 15 columns Previous **1** 2 3 4 5 6 … 94 Next

```
daily_activity<- rename_with(daily_activity, tolower)
clean_names(daily_sleep)
```

id	sleep_day	total_sleep_records	total_minutes_asleep	total_time_in_bed
<dbl>	<chr>	<int>	<int>	<int>
1503960366	4/12/2016 12:00:00 AM	1	327	346

id	sleep_day	total_sleep_records	total_minutes_asleep	total_time_in_bed
<dbl>	<chr>	<int>	<int>	<int>
1503960366	4/13/2016 12:00:00 AM	2	384	407
1503960366	4/15/2016 12:00:00 AM	1	412	442
1503960366	4/16/2016 12:00:00 AM	2	340	367
1503960366	4/17/2016 12:00:00 AM	1	700	712
1503960366	4/19/2016 12:00:00 AM	1	304	320
1503960366	4/20/2016 12:00:00 AM	1	360	377
1503960366	4/21/2016 12:00:00 AM	1	325	364
1503960366	4/23/2016 12:00:00 AM	1	361	384
1503960366	4/24/2016 12:00:00 AM	1	430	449

1-10 of 410 rows Previous **1** 2 3 4 5 6 … 41 Next

```
daily_sleep<-rename_with(daily_sleep, tolower)
clean_names(hourly_steps)
```

id	activity_hour	step_total
<dbl>	<chr>	<int>
1503960366	4/12/2016 12:00:00 AM	373
1503960366	4/12/2016 1:00:00 AM	160
1503960366	4/12/2016 2:00:00 AM	151
1503960366	4/12/2016 3:00:00 AM	0
1503960366	4/12/2016 4:00:00 AM	0
1503960366	4/12/2016 5:00:00 AM	0
1503960366	4/12/2016 6:00:00 AM	0
1503960366	4/12/2016 7:00:00 AM	0
1503960366	4/12/2016 8:00:00 AM	250
1503960366	4/12/2016 9:00:00 AM	1864

1-10 of 10,000 rows Previous **1** 2 3 4 5 6 … 1000 Next

```
hourly_steps<-rename_with(hourly_steps, tolower)
```

Consistency of date and time columns

Now that we have verified our column names and change them to lower case, we will focus on cleaning date-time format for daily_activity and daily_sleep since we will merge both data frames. Since we can disregard the time on daily_sleep data frame we are using as_date instead as as_datetime

```
daily_activity<- daily_activity %>% rename(date = activitydate) %>%
  mutate(date = as_date(date,format = "%m/%d/%Y"))
```

```
daily_sleep<- daily_sleep %>% rename(date = sleepday) %>%
  mutate(date = as_date(date,format = "%m/%d/%Y %I:%M:%S %p", tz=Sys.timezone()))
```

```
## Warning: There was 1 warning in `mutate()`.
## ℹ In argument: `date = as_date(date, format = "%m/%d/%Y %I:%M:%S %p", tz =
##   Sys.timezone())`.
## Caused by warning:
## ! `tz` argument is ignored by `as_date()`
```

We will check our cleaned datasets

```
head(daily_activity)
```

	id <dbl>	date <date>	totalsteps <int>	totaldistance <dbl>	trackerdistance <dbl>	loggedactivitiesdistance <dbl>
1	1503960366	2016-04-12	13162	8.50	8.50	0
2	1503960366	2016-04-13	10735	6.97	6.97	0
3	1503960366	2016-04-14	10460	6.74	6.74	0
4	1503960366	2016-04-15	9762	6.28	6.28	0
5	1503960366	2016-04-16	12669	8.16	8.16	0
6	1503960366	2016-04-17	9705	6.48	6.48	0

6 rows | 1-7 of 16 columns

```
head(daily_sleep)
```

	id <dbl>	date <date>	totalsleeprecords <int>	totalminutesasleep <int>	totaltimeinbed <int>
1	1503960366	2016-04-12	1	327	346
2	1503960366	2016-04-13	2	384	407
3	1503960366	2016-04-15	1	412	442
4	1503960366	2016-04-16	2	340	367
5	1503960366	2016-04-17	1	700	712
6	1503960366	2016-04-19	1	304	320

6 rows

We change from date string to date-time string for hourly_steps data

```
hourly_steps<- hourly_steps %>% rename(date_time = activityhour) %>%
  mutate(date_time = as.POSIXct(date_time,format = "%m/%d/%Y %I:%M:%S %p", tz=Sys.timezone()))
head(hourly_steps)
```

	id	date_time	steptotal
	<dbl>	<dttm>	<int>
1	1503960366	2016-04-12 00:00:00	373
2	1503960366	2016-04-12 01:00:00	160
3	1503960366	2016-04-12 02:00:00	151
4	1503960366	2016-04-12 03:00:00	0
5	1503960366	2016-04-12 04:00:00	0
6	1503960366	2016-04-12 05:00:00	0

6 rows

Merging datasets

We will merge daily_activity and daily_sleep into one to make further analysis using their primary keys: id and date

```
daily_activity_sleep<- merge(daily_activity, daily_sleep, by=c("id","date"))
glimpse(daily_activity_sleep)
```

```
## Rows: 410
## Columns: 18
## $ id                       <dbl> 1503960366, 1503960366, 1503960366, 150396036…
## $ date                     <date> 2016-04-12, 2016-04-13, 2016-04-15, 2016-04-…
## $ totalsteps               <int> 13162, 10735, 9762, 12669, 9705, 15506, 10544…
## $ totaldistance            <dbl> 8.50, 6.97, 6.28, 8.16, 6.48, 9.88, 6.68, 6.3…
## $ trackerdistance          <dbl> 8.50, 6.97, 6.28, 8.16, 6.48, 9.88, 6.68, 6.3…
## $ loggedactivitiesdistance <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
## $ veryactivedistance       <dbl> 1.88, 1.57, 2.14, 2.71, 3.19, 3.53, 1.96, 1.3…
## $ moderatelyactivedistance <dbl> 0.55, 0.69, 1.26, 0.41, 0.78, 1.32, 0.48, 0.3…
## $ lightactivedistance      <dbl> 6.06, 4.71, 2.83, 5.04, 2.51, 5.03, 4.24, 4.6…
## $ sedentaryactivedistance  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
## $ veryactiveminutes        <int> 25, 21, 29, 36, 38, 50, 28, 19, 41, 39, 73, 3…
## $ fairlyactiveminutes      <int> 13, 19, 34, 10, 20, 31, 12, 8, 21, 5, 14, 23,…
## $ lightlyactiveminutes     <int> 328, 217, 209, 221, 164, 264, 205, 211, 262, …
## $ sedentaryminutes         <int> 728, 776, 726, 773, 539, 775, 818, 838, 732, …
## $ calories                 <int> 1985, 1797, 1745, 1863, 1728, 2035, 1786, 177…
## $ totalsleeprecords        <int> 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
## $ totalminutesasleep       <int> 327, 384, 412, 340, 700, 304, 360, 325, 361, …
## $ totaltimeinbed           <int> 346, 407, 442, 367, 712, 320, 377, 364, 384, …
```

3. Analyze Phase

In the first step, we analyze the trends of users and see it can help us on BellaBeats marketing

Type of users per activeness level average - daily steps

1. Sedentary - Less than 5000 steps a day
2. Lightly active - Between 5000-7499 steps a day
3. Fairly active - Between 7500-9999 steps a day

4. Very active - More than 10000 steps a day

First we have to calculate the mean daily steps per user

```
daily_average<- daily_activity_sleep %>% group_by(id) %>%
  summarise(mean_daily_steps = mean(totalsteps), mean_daily_calories = mean(calories), mean_daily_
sleep = mean(totalminutesasleep))

head(daily_average)
```

id <dbl>	mean_daily_steps <dbl>	mean_daily_calories <dbl>	mean_daily_sleep <dbl>
1503960366	12405.680	1872.280	360.2800
1644430081	7967.750	2977.750	294.0000
1844505072	3477.000	1676.333	652.0000
1927972279	1490.000	2316.200	417.0000
2026352035	5618.679	1540.786	506.1786
2320127002	5079.000	1804.000	61.0000

6 rows

Now we will classify our users by their average daily steps

```
user_type<- daily_average %>%
  mutate(user_type = case_when(
    mean_daily_steps < 5000 ~ "sedentary",
    mean_daily_steps >=5000 & mean_daily_steps < 7500 ~ "lightly active",
    mean_daily_steps >=7500 & mean_daily_steps < 10000 ~ "fairly active",
    mean_daily_steps >= 10000 ~ "very active"
  ))

head(user_type)
```

id <dbl>	mean_daily_steps <dbl>	mean_daily_calories <dbl>	mean_daily_sleep <dbl>	user_type <chr>
1503960366	12405.680	1872.280	360.2800	very active
1644430081	7967.750	2977.750	294.0000	fairly active
1844505072	3477.000	1676.333	652.0000	sedentary
1927972279	1490.000	2316.200	417.0000	sedentary
2026352035	5618.679	1540.786	506.1786	lightly active
2320127002	5079.000	1804.000	61.0000	lightly active

6 rows

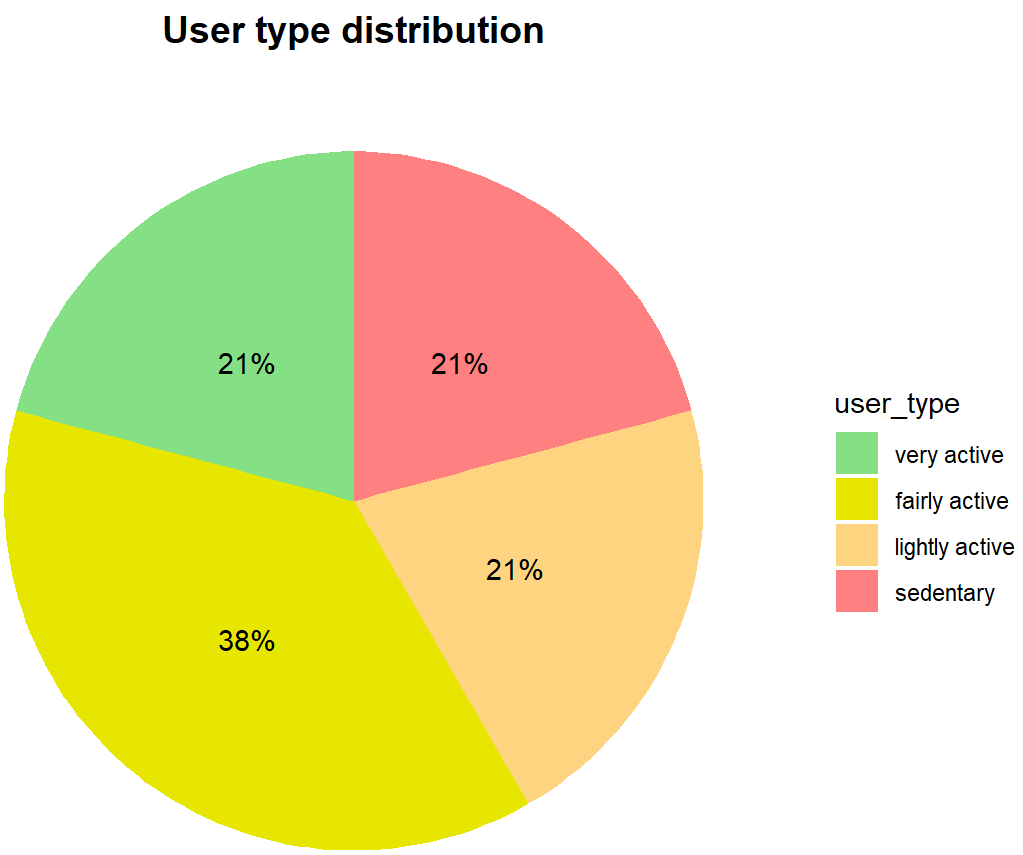
Now that we have a new column with the user type we will create a data frame with the percentage of each user type to better visualize them on a graph

```
user_type_percentage <- user_type %>%
  group_by(user_type) %>%
  summarise(total = n()) %>%
  mutate(totals = sum(total)) %>%
  group_by(user_type) %>%
  summarize(total_percent = total / totals) %>%
  mutate(labels = scales::percent(total_percent))

user_type_percentage$user_type <- factor(user_type_percentage$user_type, levels = c("very active",
"fairly active", "lightly active", "sedentary"))

head(user_type_percentage)
```

user_type <fct>	**total_percent** <dbl>	**labels** <chr>
fairly active	0.3750000	38%
lightly active	0.2083333	21%
sedentary	0.2083333	21%
very active	0.2083333	21%

4 rows

```
user_type_percentage %>%
  ggplot(aes(x="",y=total_percent,fill=user_type))+
  geom_bar(stat = "identity",width = 1)+
  coord_polar("y", start = 0)+
  theme_minimal()+ theme(axis.title.x = element_blank(),
                         axis.title.y = element_blank(),
                         panel.border = element_blank(),
                         panel.grid = element_blank(),
                         axis.ticks = element_blank(),
                         axis.text.x = element_blank(),
                         plot.title = element_text(hjust = 0.5, size = 14, face = "bold"))+
  scale_fill_manual(values = c("#85e085","#e6e600","#ffd480","#ff8080"))+
  geom_text(aes(label=labels), position = position_stack(vjust = 0.5))+
  labs(title = "User type distribution")
```



We can see that users are fairly distributed by their activity considering the daily amount of steps. We can determine that based on users activity all kind of users wear smart-devices.

Steps and minutes asleep per weekday

We would like to know what days of the week users are the most active and also what days of the week users sleep more. Moreover, we will also check if the users are completing recommended number of steps and getting recommended amount of sleep. Firstly, we calculate the weekdays based on our column date, as well as the average number of steps and minutes of sleep by weekday.

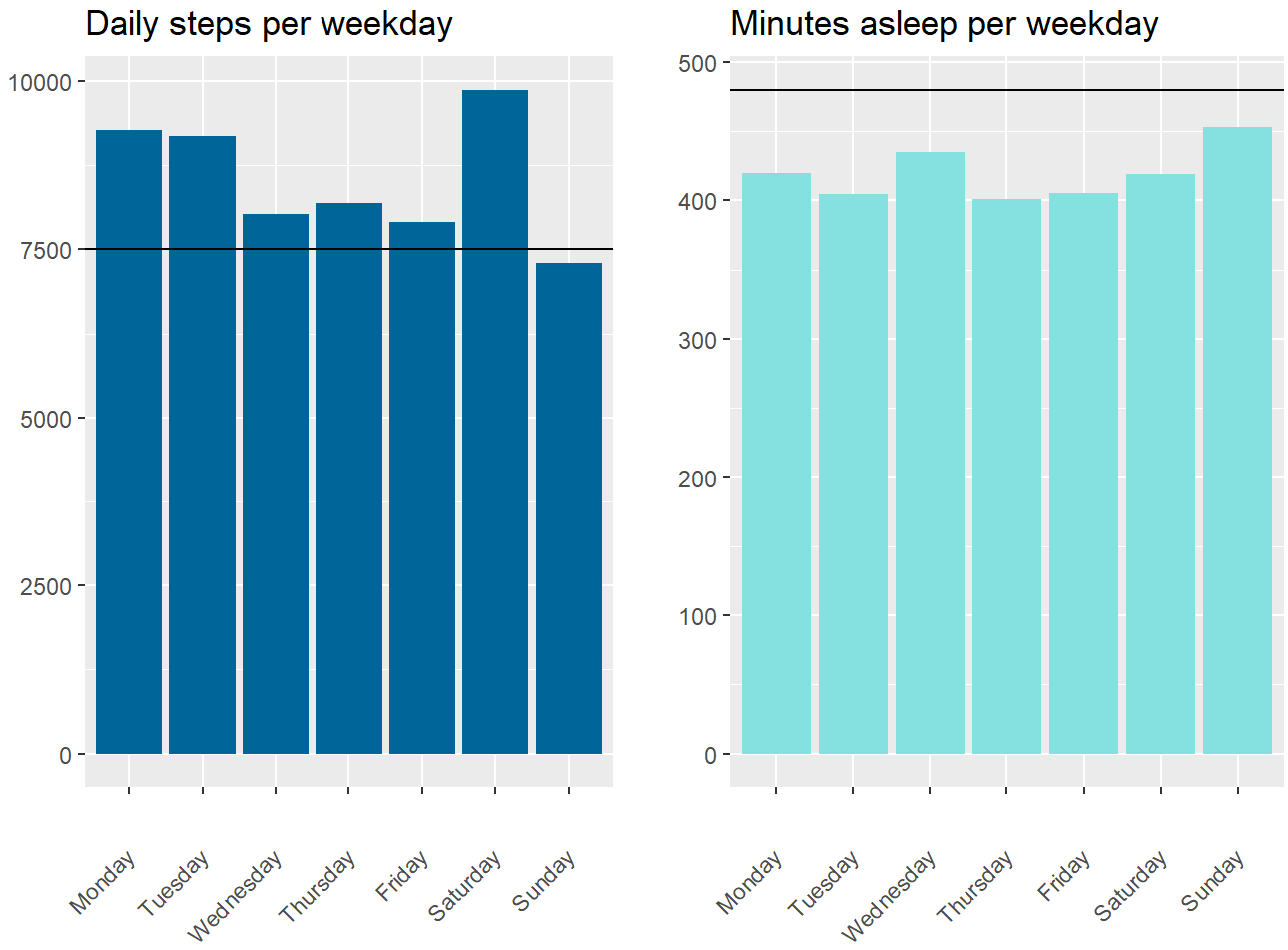
```
weekday_steps_sleep<- daily_activity_sleep %>%
  mutate(weekday = weekdays(date))

weekday_steps_sleep$weekday<- ordered(weekday_steps_sleep$weekday, levels=c("Monday","Tuesday","We
dnesday","Thursday","Friday","Saturday","Sunday"))

weekday_steps_sleep<- weekday_steps_sleep %>%
  group_by(weekday) %>%
  summarise(daily_steps = mean(totalsteps), daily_sleep = mean(totalminutesasleep))

head(weekday_steps_sleep)
```

weekday <ord>	daily_steps <dbl>	daily_sleep <dbl>
Monday	9273.217	419.5000
Tuesday	9182.692	404.5385

weekday <ord>	daily_steps <dbl>	daily_sleep <dbl>
Wednesday	8022.864	434.6818
Thursday	8183.516	401.2969
Friday	7901.404	405.4211
Saturday	9871.123	419.0702

6 rows

```
ggarrange(
  ggplot(weekday_steps_sleep)+
    geom_col(aes(weekday,daily_steps), fill="#006699")+
    geom_hline(yintercept = 7500)+
    labs(title = "Daily steps per weekday", x= "",y= "")+
    theme(axis.text.x = element_text(angle = 45, vjust = 0.5, hjust = 1)),
  ggplot(weekday_steps_sleep, aes(weekday, daily_sleep))+
    geom_col(fill="#85e0e0")+
    geom_hline(yintercept = 480)+
    labs(title = "Minutes asleep per weekday", x="",y="")+
    theme(axis.text.x = element_text(angle = 45, vjust=0.5, hjust = 1))
)
```



In the graphs above we can determine the following:

- Users walk daily the recommended amount of steps of 7500 besides Sunday's.
- Users don't sleep the recommended amount of minutes/ hours - 8 hours.

Hourly steps throughout the day

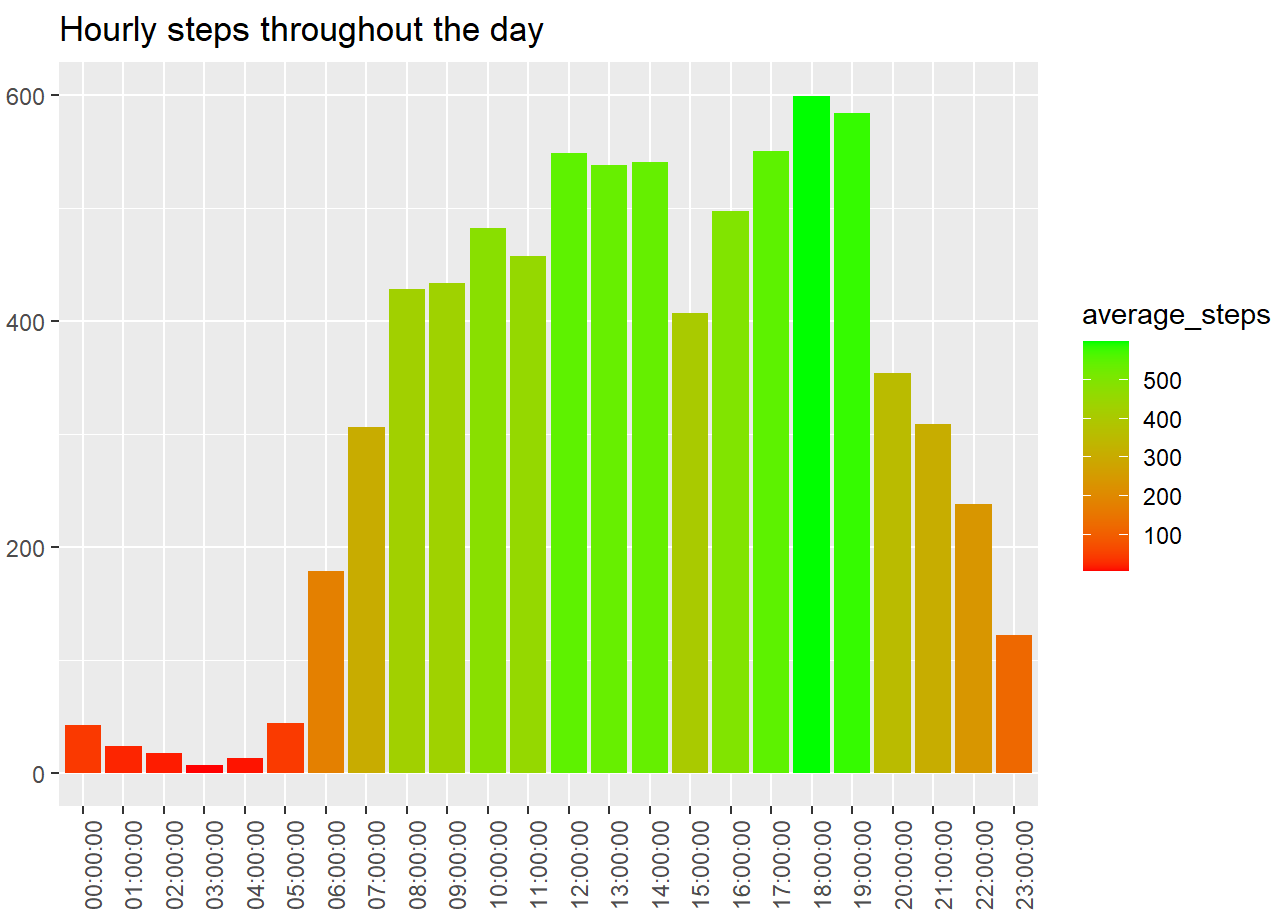
We want to know exactly what time of the day users are the most active by diving deeper into the data For that, we use hourly_sleep data frame and separate date_time column

```
hourly_steps<- hourly_steps %>%
  separate(date_time, into = c("date","time"), sep = " ") %>%
  mutate(date = ymd(date))

head(hourly_steps)
```

	id	date	time	steptotal
	<dbl>	<date>	<chr>	<int>
1	1503960366	2016-04-12	00:00:00	373
2	1503960366	2016-04-12	01:00:00	160
3	1503960366	2016-04-12	02:00:00	151
4	1503960366	2016-04-12	03:00:00	0
5	1503960366	2016-04-12	04:00:00	0
6	1503960366	2016-04-12	05:00:00	0

6 rows

```
hourly_steps %>%
  group_by(time) %>%
  summarise(average_steps = mean(steptotal)) %>%
  ggplot()+
  geom_col(mapping = aes(x=time, y=average_steps, fill= average_steps))+
  labs(title = "Hourly steps throughout the day", x="",y="")+
  scale_fill_gradient(low = "red", high = "green")+
  theme(axis.text.x = element_text(angle = 90))
```



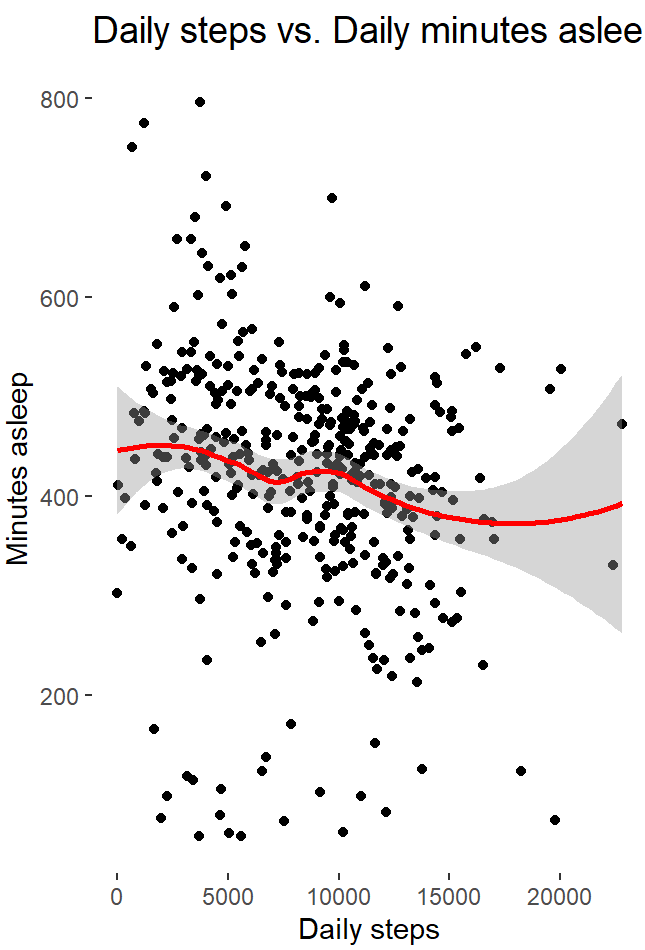
We can confirm from the graph that users are more active during the day (8am and 7pm) and the most active hours corresponds to lunch time 12pm~2pm, as well as, evening of 5pm~7pm.

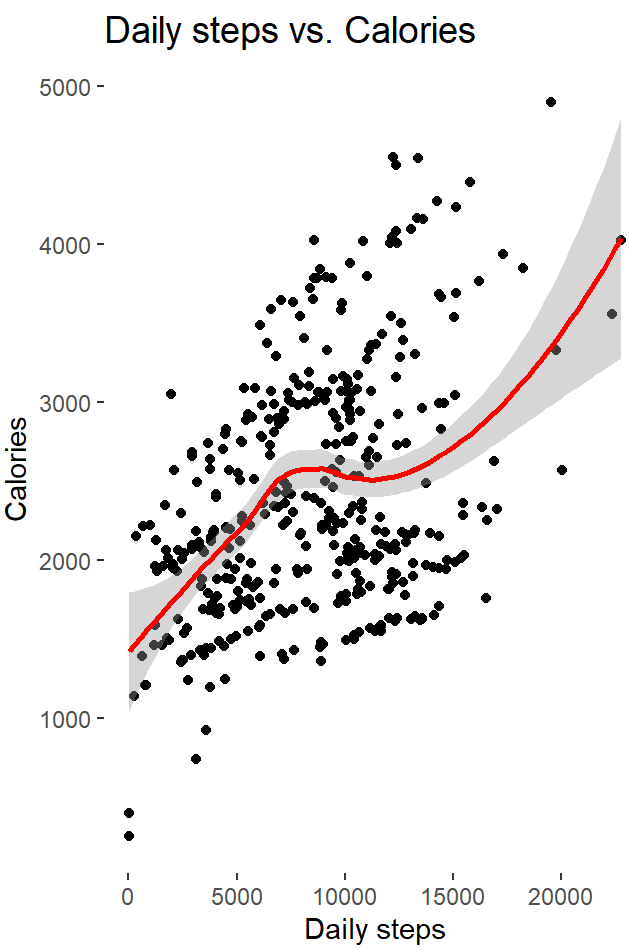
Correlations

Now we focus on to determine if there is connection between different variables, which are: Daily steps and daily sleep Daily steps and calories

```
ggarrange(
  ggplot(daily_activity_sleep, aes(x=totalsteps, y=totalminutesasleep))+
  geom_jitter()+
  geom_smooth(color="red")+
  labs(title = "Daily steps vs. Daily minutes asleep", x="Daily steps", y="Minutes asleep")+
  theme(panel.background = element_blank(),
        plot.title = element_text(size = 14)),
ggplot(daily_activity_sleep, aes(x=totalsteps, y=calories))+
  geom_jitter()+
  geom_smooth(color="red")+
  labs(title = "Daily steps vs. Calories", x="Daily steps", y="Calories")+
  theme(panel.background = element_blank(),
        plot.title = element_text(size = 14))
)
```

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





Insights from plots:

- There is no connection between daily steps and the amount of sleep users get
- However, we can see a positive correlation daily steps and calories burned

Use of smart device

Days used smart device Now that we have seen some trends in activity, sleep and calories burned, we want to see how often do the users in our sample use their device. That way we can plan our marketing strategy and see what features would benefit the use of smart devices.

We will calculate the number of users that use their smart device on a daily basis, classifying our sample into three categories knowing that the date interval is 31 days:

- high use - users who use their device between 21 and 31 days;
- moderate use - users who use their device between 10 and 20 days;
- low use - users who use their device between 1 and 10 days.

First we have to create a new data frame grouped by id to calculate the number of days the device used with the classification column to explain

```
daily_use<- daily_activity_sleep %>%
  group_by(id) %>%
  summarise(days_used = sum(n())) %>%
  mutate(usage = case_when(
    days_used >=1 & days_used<=10 ~ "low use",
    days_used >=11 & days_used<=20 ~ "moderate use",
    days_used >=21 & days_used<=31 ~ "high use",
    ))

head(daily_use)
```

id <dbl>	days_used <int>	usage <chr>
1503960366	25	high use
1644430081	4	low use
1844505072	3	low use
1927972279	5	low use
2026352035	28	high use
2320127002	1	low use

6 rows

We will now create a percentage data frame to better visualize the results in the graph. We are also ordering our usage levels.

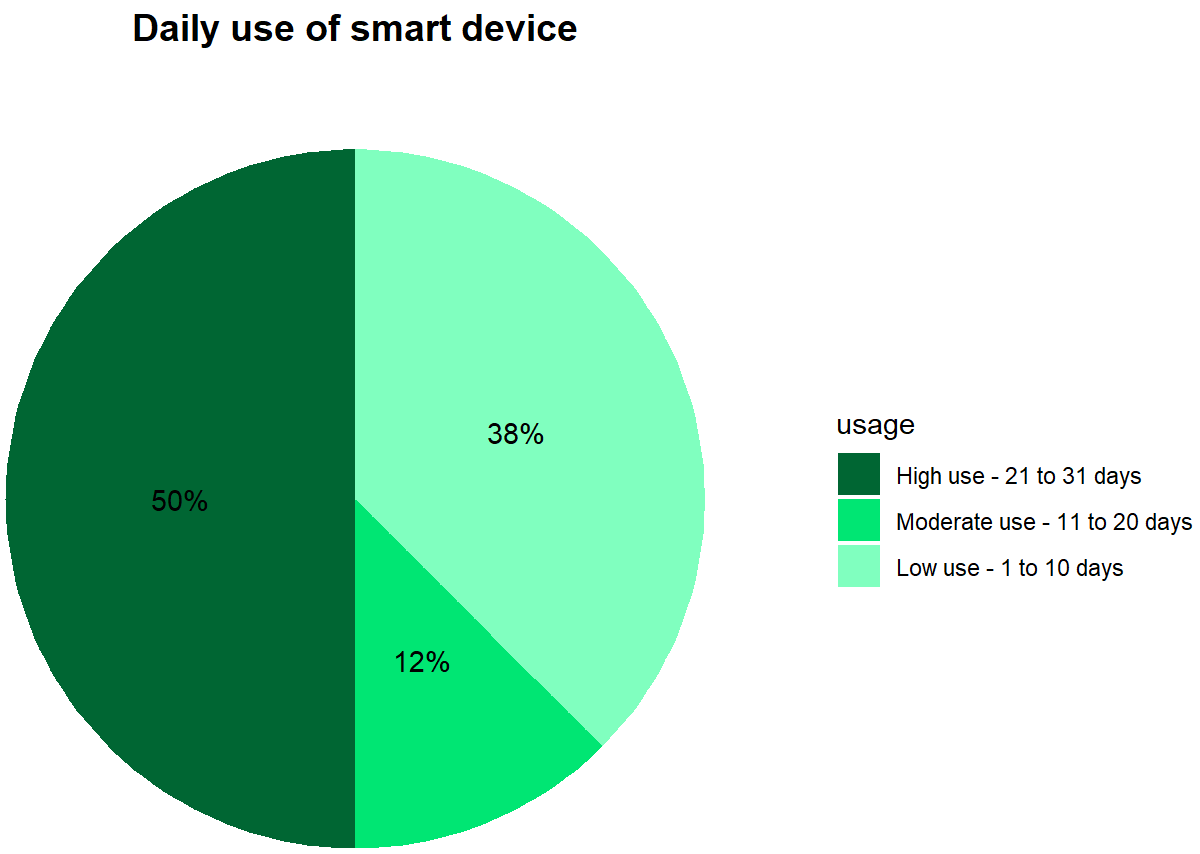
```
daily_use_percentage <- daily_use %>%
  group_by(usage) %>%
  summarise(total = n()) %>%
  mutate(totals = sum(total)) %>%
  group_by(usage) %>%
  summarise(total_percent = total / totals) %>%
  mutate(labels = scales::percent(total_percent))

daily_use_percentage$usage <- factor(daily_use_percentage$usage, levels = c("high use", "moderate use", "low use"))

head(daily_use_percentage)
```

usage <fct>	total_percent <dbl>	labels <chr>
high use	0.500	50%
low use	0.375	38%
moderate use	0.125	12%

3 rows

Then we turn to create its plot

```
daily_use_percentage %>%
  ggplot(aes(x="",y=total_percent,fill=usage))+
  geom_bar(stat = "identity", width = 1)+
  coord_polar("y",start = 0)+
  theme_minimal()+
  theme(axis.title.x = element_blank(),
        axis.title.y = element_blank(),
        panel.border = element_blank(),
        panel.grid = element_blank(),
        axis.ticks = element_blank(),
        axis.text.x = element_blank(),
        plot.title = element_text(hjust = 0.5, size = 14, face = "bold"))+
  geom_text(aes(label=labels),
            position = position_stack(vjust = 0.5))+
  scale_fill_manual(values = c("#006633","#00e673","#80ffbf"),
                    labels = c("High use - 21 to 31 days",
                               "Moderate use - 11 to 20 days",
                               "Low use - 1 to 10 days"))+
  labs(title = "Daily use of smart device")
```



After analyzing our results we can say that:

- 50% of the users of the given sample use their frequently between 21~31 days;
- 12% of the users use their device 11 to 20 days;
- 38% of them use really rarely their device.

Time used smart device

Being more precise we want to see how many minutes users wear their device per day. For that we will merge created daily_use data frame with daily_activity table in order to filter results by daily use of device.

```
daily_use_merged<- merge(daily_activity, daily_use, by=c("id"))
head(daily_use_merged)
```

id <dbl>	date <date>	totalsteps <int>	totaldistance <dbl>	trackerdistance <dbl>	loggedactivitiesdistance <dbl>
1 1503960366	2016-05-07	11992	7.71	7.71	0
2 1503960366	2016-05-06	12159	8.03	8.03	0
3 1503960366	2016-05-01	10602	6.81	6.81	0
4 1503960366	2016-04-30	14673	9.25	9.25	0
5 1503960366	2016-04-12	13162	8.50	8.50	0
6 1503960366	2016-04-13	10735	6.97	6.97	0

6 rows | 1-7 of 18 columns

In the next step, we need to create a new data frame calculating the total amount of minutes users wore the device every day and creating three different categories:

- All day - device was worn all day.
- More than half day - device was worn more than half of the day.
- Less than half day - device was worn less than half of the day.

```
minutes_worn_daily<- daily_use_merged %>%
  mutate(total_minutes_worn =veryactiveminutes+fairlyactiveminutes+lightlyactiveminutes+sedentarym
inutes) %>%
  mutate(total_minutes_worn_percentage = (total_minutes_worn)/1440*100) %>% # a day consists of 14
40 minutes
  mutate(worn = case_when(
    total_minutes_worn_percentage == 100 ~ "All day",
    total_minutes_worn_percentage < 100 & total_minutes_worn_percentage>=50 ~ "More than half da
y",
    total_minutes_worn_percentage <50 & total_minutes_worn_percentage>0 ~ "Less than half day"
  ))

head(minutes_worn_daily)
```

id <dbl>	date <date>	totalsteps <int>	totaldistance <dbl>	trackerdistance <dbl>	loggedactivitiesdistance <dbl>
1 1503960366	2016-05-07	11992	7.71	7.71	0
2 1503960366	2016-05-06	12159	8.03	8.03	0
3 1503960366	2016-05-01	10602	6.81	6.81	0
4 1503960366	2016-04-30	14673	9.25	9.25	0
5 1503960366	2016-04-12	13162	8.50	8.50	0

	id	date	totalsteps	totaldistance	trackerdistance	loggedactivitiesdistance
	<dbl>	<date>	<int>	<dbl>	<dbl>	<dbl>
6	1503960366	2016-04-13	10735	6.97	6.97	0

6 rows | 1-7 of 21 columns

As we have done before, to better visualize our results we will create new data frames. In this case we will create **four different data frames** to arrange them later on on a same visualization.

- First data frame will show the total of users and will calculate percentage of minutes worn the device taking into consideration the three categories created.
- The three other data frames are filtered by category of daily users so that we can see also the difference of daily use and time use.

```
minutes_worn_percentage<- minutes_worn_daily %>%
  group_by(worn) %>%
  summarise(total = n()) %>%
  mutate(totals = sum(total)) %>%
  group_by(worn) %>%
  summarise(total_percentage = total/totals) %>%
  mutate(labels = scales::percent(total_percentage))


minutes_worn_highuse <- minutes_worn_daily%>%
  filter (usage == "high use")%>%
  group_by(worn) %>%
  summarise(total = n()) %>%
  mutate(totals = sum(total)) %>%
  group_by(worn) %>%
  summarise(total_percentage = total / totals) %>%
  mutate(labels = scales::percent(total_percentage))


minutes_worn_moduse<- minutes_worn_daily %>%
  filter(usage == "moderate use") %>%
  group_by(worn) %>%
  summarise(total = n()) %>%
  mutate(totals  = sum(total)) %>%
  group_by(worn) %>%
  summarise(total_percentage = total/totals) %>%
  mutate(labels = scales::percent(total_percentage))


minutes_worn_lowuse<- minutes_worn_daily %>%
  filter(usage == "low use") %>%
  group_by(worn) %>%
  summarise(total = n()) %>%
  mutate(totals = sum(total)) %>%
  group_by(worn) %>%
  summarise(total_percentage = total/totals) %>%
  mutate(labels = scales::percent(total_percentage))


minutes_worn_percentage$worn<- factor(minutes_worn_percentage$worn, levels = c("All day","More tha
n half day","Less than half day"))
minutes_worn_highuse$worn<- factor(minutes_worn_highuse$worn, levels = c("All day","More than half
day","Less than half day"))
minutes_worn_moduse$worn<- factor(minutes_worn_moduse$worn, levels = c("All day","More than half d
ay","Less than half day"))
minutes_worn_lowuse$worn<- factor(minutes_worn_lowuse$worn, levels = c("All day","More than half d
ay","Less than half day"))


head(minutes_worn_percentage)
```

worn	total_percentage	labels
<fct>	<dbl>	<chr>
All day	0.36465638	36%

worn <fct>	total_percentage <dbl>	labels <chr>
Less than half day	0.03506311	4%
More than half day	0.60028050	60%

3 rows

```
head(minutes_worn_highuse)
```

worn <fct>	total_percentage <dbl>	labels <chr>
All day	0.06756757	6.8%
Less than half day	0.04324324	4.3%
More than half day	0.88918919	88.9%

3 rows

```
head(minutes_worn_moduse)
```

worn <fct>	total_percentage <dbl>	labels <chr>
All day	0.2666667	27%
Less than half day	0.0400000	4%
More than half day	0.6933333	69%

3 rows

```
head(minutes_worn_lowuse)
```

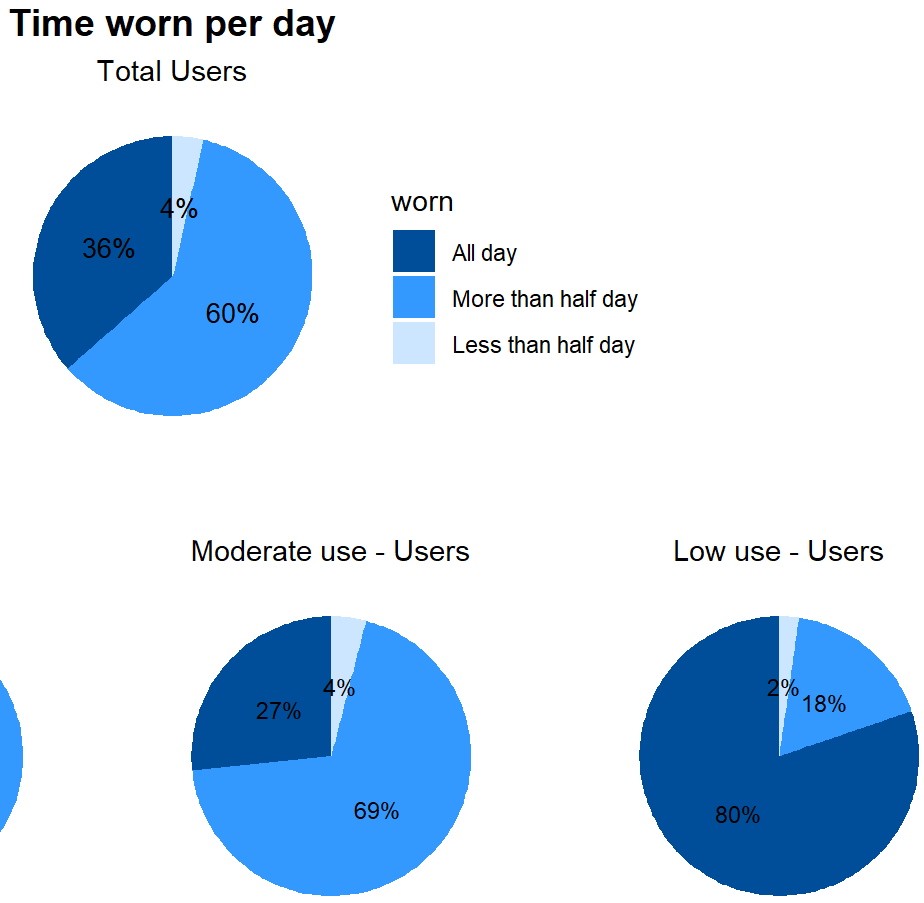
worn <fct>	total_percentage <dbl>	labels <chr>
All day	0.80223881	80%
Less than half day	0.02238806	2%
More than half day	0.17537313	18%

3 rows

Now that we have created the four data frames and also ordered worn level categories, we can visualize our results in the following plots. All the plots have been arranged together for a better visualization.

```
ggarrange(      #we combine the graph of 'total user' with other separate 3 graphs
 ggplot(minutes_worn_percentage, aes(x="", y=total_percentage, fill=worn))+
  geom_bar(stat = "identity", width = 1)+
  coord_polar("y", start = 0)+
  theme_minimal()+
  theme(axis.title.x = element_blank(),
        axis.title.y = element_blank(),
        panel.border = element_blank(),
        panel.grid = element_blank(),
        axis.ticks = element_blank(),
        axis.text.x = element_blank(),
        plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
        plot.subtitle = element_text(hjust = 0.5))+
  scale_fill_manual(values = c("#004d99","#3399ff","#cce6ff"))+
  geom_text(aes(label = labels),
            position = position_stack(vjust = 0.5), size = 3.5)+
  labs(title = "Time worn per day", subtitle = "Total Users"),  # end of plot code for total users


ggarrange(   # it combines 3 separate graphs of 'high', 'mod', 'low'
  ggplot(minutes_worn_highuse, aes(x="", y=total_percentage, fill=worn))+
    geom_bar(stat = "identity", width = 1)+
    coord_polar("y", start = 0)+
    theme_minimal()+
    theme(axis.title.x = element_blank(),
          axis.title.y = element_blank(),
          panel.border = element_blank(),
          panel.grid = element_blank(),
          axis.ticks = element_blank(),
          axis.text.x = element_blank(),
          plot.title = element_text(hjust = 0.5, size = 14, face="bold"),
          plot.subtitle = element_text(hjust = 0.5),
          legend.position = "none")+
    scale_fill_manual(values = c("#004d99", "#3399ff", "#cce6ff"))+
    geom_text(aes(label = labels),
                    position = position_stack(vjust = 0.5), size = 3)+
    labs(title ="", subtitle = "High use - Users"), #end of plot for 'high use -Users'

  ggplot(minutes_worn_moduse, aes(x="", y=total_percentage, fill=worn))+
    geom_bar(stat = "identity", width = 1)+
    coord_polar("y", start = 0)+
    theme_minimal()+
    theme(axis.title.x = element_blank(),
          axis.title.y = element_blank(),
          panel.border = element_blank(),
          panel.grid = element_blank(),
          axis.ticks = element_blank(),
          axis.text.x = element_blank(),
          plot.title = element_text(hjust = 0.5, size = 14, face="bold"),
          plot.subtitle = element_text(hjust = 0.5),
          legend.position = "none")+
    scale_fill_manual(values = c("#004d99", "#3399ff", "#cce6ff"))+
    geom_text(aes(label=labels),
              position = position_stack(vjust = 0.5), size=3)+
    labs(title="",subtitle = "Moderate use - Users"), # end of plot for moderate use Users
```

```
  ggplot(minutes_worn_lowuse, aes(x="", y=total_percentage, fill=worn))+
  geom_bar(stat="identity", width = 1)+
  coord_polar("y", start = 0)+
  theme_minimal()+
  theme(axis.title.x = element_blank(),
        axis.title.y = element_blank(),
        panel.border = element_blank(),
        panel.grid = element_blank(),
        axis.text.x = element_blank(),
        axis.ticks = element_blank(),
        plot.title = element_text(hjust = 0.5, size = 14, face="bold"),
        plot.subtitle = element_text(hjust = 0.5),
        legend.position = "none")+
  scale_fill_manual(values = c("#004d99", "#3399ff", "#cce6ff"))+
  geom_text(aes(label = labels),
            position = position_stack(vjust = 0.5), size = 3)+
  labs(title = "", subtitle = "Low use - Users"), # end of 'low use-Users' plot
ncol = 3), # end of combining 3 small separate plots
nrow = 2) # end of combining total + 3 separate small plots
```



Per our plots we can see that 36% of the total of users wear the device all day long, 60% more than half day long and just 4% less than half day.

If we filter the total users considering the days they have used the device and also check each day how long they have worn the device, we have the following results:

Reminder:

- high use - users who use their device between 21 and 31 days.
- moderate use - users who use their device between 10 and 20 days.
- low use - users who use their device between 1 and 10 days.

High users - Just 6.8% of the users that have used their device between 21 and 31 days wear it all day. 88.9% wear the device more than half day but not all day.

Moderate users are the ones who wear the device less on a daily basis.

Being low users who wear more time their device the day they use it.

Conclusion

Based on the analysis we have done with the help of given data, I can provide some **recommendations**:

- **Daily notifications** -> We classified users into 4 categories and saw that the average of users walk more than 7,500 steps daily besides Sundays. We can encourage users to reach at least daily recommended 8000 steps sending them alarms if they haven't reached the steps and creating also posts on our app explaining the benefits of reaching that goal. We also saw a positive correlation between steps and calories. Moreover, we detected that users get sleep less than 8 hours a day. They could set up a desired time to go to sleep and receive a notification minutes before to prepare to sleep.
- **Reward system** -> We are aware that some people don't get motivated by notifications so we could create a kind of competition among BellaBeat users. After they agree to participate in the competition, users will be able to see how well other users are doing and get motivated to reach higher levels and users would get online badges as they reach certain level.

On our analysis we didn't just check trends on daily users habits we also realized that just 50% of the users use their device on a daily basis and that just 36% of the users wear the device all time the day they used it. We can continue promote Bellabeat's products features: ***Water-resistant & Long-lasting batteries & Elegant design***

We can forward our marketing focus on the characteristic that people can wear the product anytime to any occasion.