Transforming Data into Success: Bellabeat Case Study

José Jaén Delgado <www.linkedin.com/in/jose-jaen/>

Summary – The project consists in a complete **business break-down investigation** using a variety of **data-based analytical tools**. Visualizations, statistical summaries and conclusions will be presented in a comprehensive fashion, showing every single step that was taken as to introduce the reader to my **data analysis mindset** and **problem-solving capacities**

Keyconcepts – Data Analytics, R, Data Cleaning, SQL, Data Manipulation

1 Business Task

I will analyze smart device fitness data for Bellabeat, a high-tech manufacturer of health-focused products for women. The main objective is to give sound advice so that it can **improve its current business plan** by applying a **data-driven decision making** framework. More concretely, I will select a Bellabeat product upon which the findings of my analysis of trends in smart device usage will be applied.

- The first step of my analysis is **describing the available data**, which will provide a general idea about the situation we are facing.
- A **data-cleaning process** will follow, so as to avoid dealing with misleading information.
- Then, I will present relevant **summary statistics** along with insightful **data visualizations**, for which a proper **data manipulation** exercise is essential.
- Finally, a conclusion will be reached in the form of **data-informed recommendations** that specifically target Bellabeat's current and potential customers. Throughout the analysis described above, **Spreadsheets, SQL and R programming** will be used.

2 Data Description

To collect the data I resorted to *Kaggle*, one of the largest online Data Science communities. Among an extensive list of datasets, FitBit Fitness Tracker data seem appropriate for our analysis owing to the possibility of recognizing patterns in smart device usage by studying human temporal routine behavior. Said datasets were generated by respondents to a distributed survey via Amazon Mechanical Turk in December 2016. There are 18 CSV files in total, each one offering imperative information, for instance: *minute-level output for physical activity, heart rate*, and *sleep monitoring*.

3 Data Cleaning and Analysis setup

For the sake of simplicity, I will centralize all of the data into a **Relational Database** that will be concocted using **PostgreSQL**. This will allow me to easily manage the entirety of the files and make relevant queries, as the CSV files can be transformed into tables which I will link by joining common attributes. Note that the server for the aforementioned Database is *localhost*, thus, Kaggle notebooks will not be able to retrieve SQL queries' insights. Nonetheless, it is fairly easy to figure out a simple workaround: the results of my queries will be exported to a CSV file, allowing me to display a glimpse of said queries with R.

4 Getting to know Smart Device Users

4.1 Basic Information and Limitations of Data

After creating the database, it is time for a first glance at the available data:

```
SELECT COUNT(DISTINCT id) AS total_ids FROM daily_activity;
```

```
## 33
```

```
SELECT COUNT(DISTINCT id) AS sleep_ids FROM daily_sleep;
```

```
## 24
```

```
SELECT COUNT(DISTINCT id) AS weight_ids FROM weight;
```

```
## 8
```

```
SELECT COUNT(DISTINCT activitydate) AS days FROM daily_activity;
```

```
## 31
```

A quick SQL query shows that data come from **33 unique users** for a time dimension of **31 days**. Only 8 runners have submitted their weight and approximately 27% did not use smart devices during their daily sleep. In addition, there is **no data about users' age, sex and height**, which is imperative in order to **draw conclusions** and **carry out statistical inference**. The missing runners can be identified by the following queries:

```
SELECT DISTINCT id
FROM daily_activity
WHERE id NOT IN (SELECT DISTINCT id FROM daily_sleep);
```

```
SELECT DISTINCT id
FROM daily_activity
WHERE id NOT IN (SELECT DISTINCT id FROM weight);
```

Runers ID's are not displayed due to data privacy reasons, keeping this sensitive information for internal analysis. The **total days of smart devices usage by user** and **average number of days that all runners tracked their activity** are, respectively:

```
SELECT
COUNT(DISTINCT activitydate) AS usage_days
FROM daily_activity
GROUP BY id;
```

```
## 31 31 30 31 31 31 31 31 18 31 20 30 31  4 31 31 31 31 31 31 30 28 29 26 31
## 26 31 31 19 31 31 29 31
```

```
WITH temp_table AS (
    SELECT COUNT(DISTINCT activitydate)
    FROM daily_activity
    GROUP BY id)
SELECT ROUND(AVG(usage_days), 2) AS average_days
FROM temp_table;
```

```
## 28.48
```

As the usage of FitBit trackers is high in the sample, the aforementioned issue about missing information of daily sleep and weight may be explained by a lack of appropriate app reminder messages rather than users refusing to record their actions.

4.2 Types of runners

Now let us **categorize** runners according to their activity. For this purpose I will create **three user types**: *beginners, intermediate users* and *pro runners*. The **metrics threshold rule** to be applied is that of average total distance, in other words, if a user runs on average less than 5 kilometers, it will fall in the first group. Should its average distance lies between 5 kilometers and 8 kilometers, it will be assigned to the intermediate class. Lastly, a runner will be considered an experienced one if its mean distance is greater than 8 kilometers. The

thresholds above have been chosen taking into account the total average distance, minimum average distance and maximum average distance of the most dedicated runner:

```
WITH temp_table2 AS (
    SELECT
        ROUND(AVG(TotalDistance), 2) AS mean_distance,
        COUNT(DISTINCT activitydate) AS number_of_days,
        id,
        CASE
            WHEN ROUND(AVG(TotalDistance), 2) < 5
            THEN 'Beginner Level'
            WHEN ROUND(AVG(TotalDistance), 2) >= 5 AND
                 ROUND(AVG(TotalDistance), 2) < 8
            THEN 'Intermediate Level'
        ELSE 'Pro level'
        END AS user_type
    FROM daily_activity
    GROUP BY id)
SELECT
    MAX(TEMP.mean_distance) AS average_distance_best_runner
    MIN(ACT.totaldistance) AS minimum_distance
    ROUND(AVG(ACT.totaldistance), 2) AS average_distance_all_runners
FROM temp_table2 TEMP
INNER JOIN daily_activity ACT ON TEMP.id = ACT.id;
```

```
## Average best runner | Minimum average distance | Average Distance of All Users
         13.21                        0                          5.49
```

Having expounded the reasoning behind the present threshold rule, we proceed with our analysis:

```
## Setting up the data frame I will work with

average_by_user_data <- activity %>%
    select(Id, TotalDistance) %>%
    group_by(Id) %>%
    summarize(average_distance = mean(TotalDistance)) %>%
    arrange(-average_distance)

## Transforming the data frame above into a matrix so I can extract
the average distance column

average_by_user_data <- as.matrix(average_by_user_data )
average_by_user  <- as.vector(average_by_user_data [, 2])

## Algorithm to categorize users based on the metrics threshold rule
```

```
categorize_users <- function(x) {
    i <- 0 ;
    runner_type <- c(1:length(x))
    while (i < length(x)) {
        i <- i + 1
        if (x[i] < 5) {
            runner_type[i] <- "Beginner Level"
        } else if (x[i] >= 5 & x[i] < 8) {
            runner_type[i] <- "Intermediate Level"
        } else {
            runner_type[i] <- "Pro Level"
        }
    }
    return(runner_type)
}
```

```
## Adding a column with the user types

average_by_user_data <- cbind(average_by_user_data,
    categorize_users(average_by_user))

colnames(average_by_user_data) <- c("Id", "average_distance", "user_type")

## Removing Id's for privacy

average_by_user_data <-
    average_by_user_data[, colnames(average_by_user_data) != "Id"]

## Showing average distance and user type per runner

average_by_user_data
```

```
##       average_distance    user_type
##  [1,] "13.2129031381299"  "Pro Level"
##  [2,] "11.4751611986468"  "Pro Level"
##  [3,] "8.39322589289757"  "Pro Level"
##  [4,] "8.08419349116664"  "Pro Level"
##  [5,] "8.01538459154276"  "Pro Level"
##  [6,] "7.80967738551478"  "Intermediate Level"
##  [7,] "7.51699994405111"  "Intermediate Level"
##  [8,] "6.95516128309312"  "Intermediate Level"
##  [9,] "6.91354846185253"  "Intermediate Level"
## [10,] "6.5858064774544"   "Intermediate Level"
## [11,] "6.38806450781563"  "Intermediate Level"
## [12,] "6.355555535915"    "Intermediate Level"
## [13,] "6.2133333047231"   "Intermediate Level"
```

```
## [14,] "5.63967744958016"  "Intermediate Level"
## [15,] "5.61548382236112"  "Intermediate Level"
## [16,] "5.34214291402272"  "Intermediate Level"
## [17,] "5.29533335367839"  "Intermediate Level"
## [18,] "5.10161286015664"  "Intermediate Level"
## [19,] "5.08064517667217"  "Intermediate Level"
## [20,] "4.8922580470361"   "Beginner Level"
## [21,] "4.707000041008"    "Beginner Level"
## [22,] "4.66736846848538"  "Beginner Level"
## [23,] "4.27241380461331"  "Beginner Level"
## [24,] "3.91483872936618"  "Beginner Level"
## [25,] "3.45483871525334"  "Beginner Level"
## [26,] "3.24580644023034"  "Beginner Level"
## [27,] "3.18774190448946"  "Beginner Level"
## [28,] "2.86250001192093"  "Beginner Level"
## [29,] "1.81346151612413"  "Beginner Level"
## [30,] "1.70612903684378"  "Beginner Level"
## [31,] "1.62612903893234"  "Beginner Level"
## [32,] "1.18655171682095"  "Beginner Level"
## [33,] "0.634516123081408" "Beginner Level"
```

Being of considerable interest to know the **internal composition** of these groups, I will write an **algorithm** that counts the total number of runners in each group and then **plot a pie chart**.

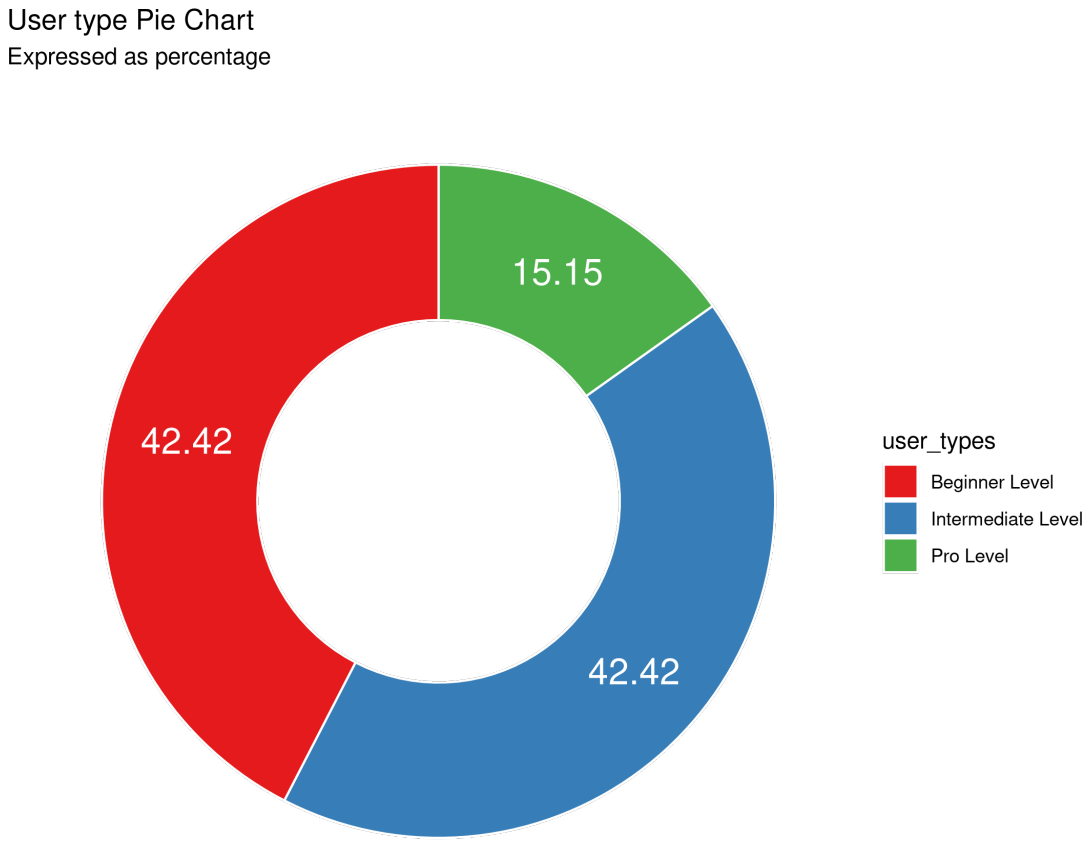
```
count_user_types <- function(x) {
    i <- 0 ; intermediate <- 0
    beginner <- 0 ; pro <- 0
    for (i  in 1:length(x)) {
        i <- i + 1
        if (x[i] < 5) {
            beginner <- beginner + 1
        } else if (x[i] >= 5 & x[i] < 8) {
            intermediate <- intermediate + 1
        } else {
            pro <- pro + 1
        }
    }
    return(paste("Beginners: ", beginner,
    ", Intermediate: ", intermediate, ", Pro users: ", pro))
}
```

```
## Group composition

count_user_types(as.numeric(average_by_user_data[, 1]))
```

```
## "Beginners:  14 , Intermediate:  14 , Pro users:  5"
```

```
## Setting up the pie chart with categories and number of users by type

user_types <- c("Pro Level", "Intermediate Level", "Beginner Level")

group_composition <- c(
           count_user_types(as.numeric(average_by_user_data[, 1]))[3],
           count_user_types(as.numeric(average_by_user_data[, 1]))[2],
           count_user_types(as.numeric(average_by_user_data[, 1]))[1]
           )

## Creating a data frame with the information above

datafr <- data.frame(user_types, group_composition)

## Calculating the relevant values that will make up the plot

data <- datafr %>%
    arrange(desc(user_types)) %>%
    mutate(prop = group_composition / sum(group_composition) * 100) %>%
    mutate(ypos = cumsum(prop) - 0.5 * prop)

pie_chart <- ggplot(data = data, aes(x = 2, y = prop, fill = user_types)) +
  geom_bar(stat = "identity", color = "white") +
  coord_polar("y", start = 0) + theme_void() +
  geom_text(aes(y = ypos, label = round(prop, 2)), color = "white", size = 6) +
  scale_fill_brewer(palette = "Set1") +
  labs(title = "User type Pie Chart", subtitle = "Expressed as percentage") +
  xlim(0.5, 2.5)
```

In the pie chart below, it can be observed that whereas **84.84%** of total users are either **getting started to running or moderately doing the sport**, the rest have taken it to another level by enduring long distance running.



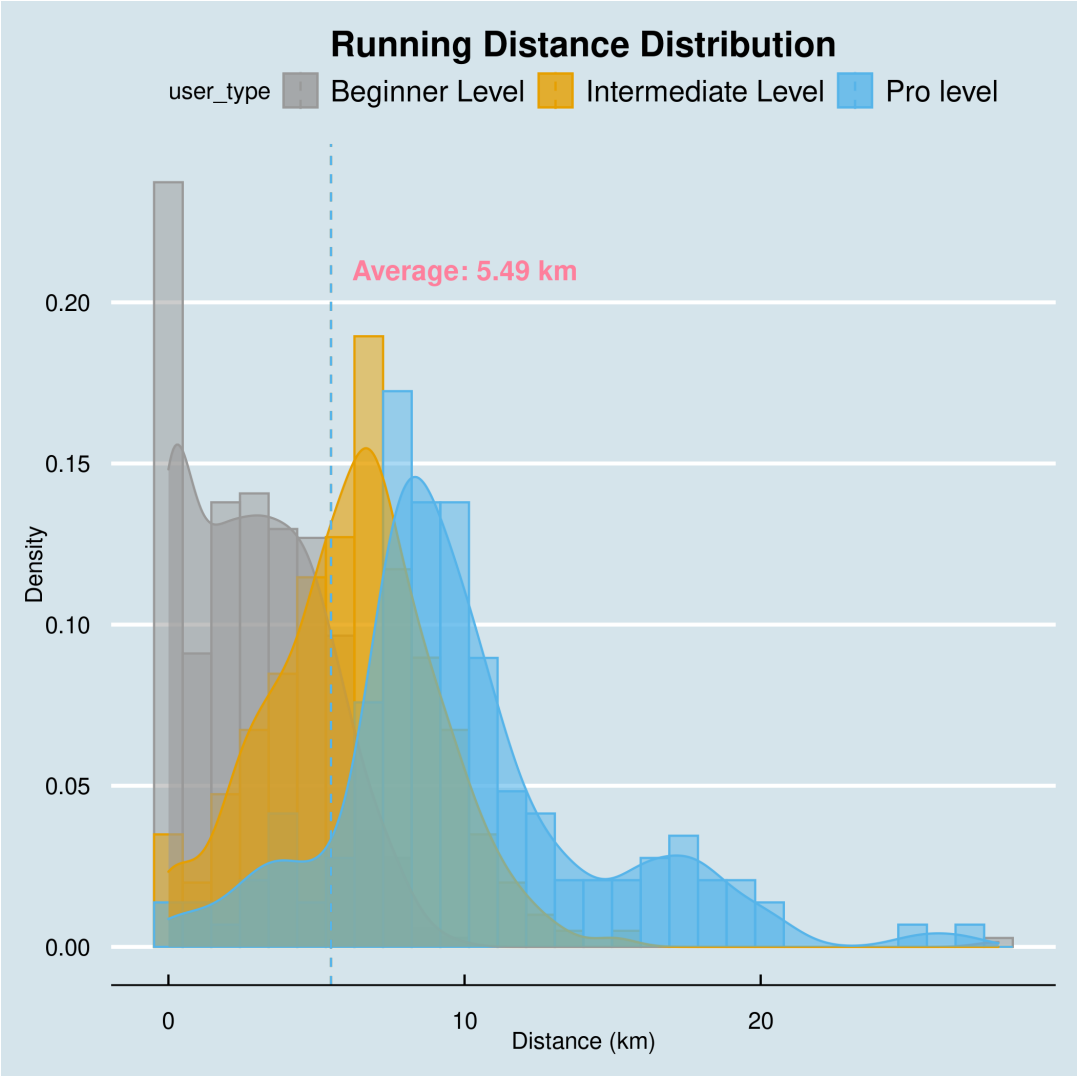
We could also dig deep through the data by studying the **distribution of running distance per user type**, which certainly enriches the statistical aspect of the project as it would allow to **compute the probability of running a certain distance given a user type**.

In order to graphically show the distribution of running distance, we perform a SQL query first, displaying essential data for our histogram. We then export such information to a CSV file and read it with R:

```
-- Note that exporting files is possible in PostgreSQL by executing:

COPY (
    #QUERY RESULTS TO BE EXPORTED#
    ) TO 'file_directory/file_name.csv' DELIMITER ',' CSV HEADER;
```

```sql
WITH temp_table3 AS (
SELECT
    CASE
        WHEN ROUND(AVG(TotalDistance), 2) < 5
        THEN 'Beginner Level'
        WHEN ROUND(AVG(TotalDistance), 2) >= 5 AND
        ROUND(AVG(TotalDistance), 2) < 8
        THEN 'Intermediate Level'
    ELSE 'Pro level'
    END AS user_type,
    id
FROM daily_activity
GROUP BY id)
SELECT
    ACT.*, TEMPO.*
FROM daily_activity ACT
INNER JOIN temp_table3 TEMPO ON TEMPO.id = ACT.id;
```

```r
## Histogram by groups setup

distribution <- ggplot(data = query3,
    aes(x = TotalDistance, color = user_type, fill = user_type)) +
    geom_histogram(aes(y = ..density..), position = "identity", alpha = 0.5) +
    geom_density(alpha = 0.6) +
    geom_vline(data = query3, aes(xintercept = mean(TotalDistance),
        color = user_type), linetype = "dashed") +
    scale_color_manual(values = c("#999999", "#E69F00", "#56B4E9")) +
    scale_fill_manual(values = c("#999999", "#E69F00", "#56B4E9")) +
    labs(title = "Running Distance Distribution",
        x = "Distance (km)", y = "Density") +
    theme_economist() +
    annotate("text", x = 10, y = 0.21,
        label = "Average: 5.49 km",
        color = "#FE7F9C", fontface = "bold", size = 4.5) +
    theme(plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5, vjust = -2))

## Using custom background theme and color palette
```



As an example, let us calculate the probability of running less than the average total distance for each user type. Note that the probability density function of a continuous variable is defined as

$$Pr(a \leq X \leq b) = \int_a^b f_x(x)\, dx$$

Particularizing for our case, we can approximate said expression with the estimator below as the distance variabe is discretized:

$$\frac{\sum_{i=1}^{n} X_{i,g}}{\sum_{i=1}^{n} X_{i,g} + \sum_{i=1}^{n} Y_{i,g}}$$

N = Total number of runs ; i = Individual i ; g = User type (beginner, intermediate or pro)

X = Binary variable taking value 1 if a certain run of individual i pertaining to group g exceeds 5.49 km and 0 otherwise

Y = Counterpart of X (dummy values are swapped)

```
## Probability is calculated as the number of times a user ran less
than 5.49 km with respect to its total runs

prob_pro <- length(query3$totaldistance[query3$user_type == "Pro level" &
    query3$totaldistance < 5.49])/length(query3$totaldistance[
        query3$user_type == "Pro level"])

prob_mid <- length(query3$totaldistance[query3$user_type == "Intermediate" &
    query3$totaldistance < 5.49])/length(query3$totaldistance[
        query3$user_type == "Intermediate Level"])

prob_noob <- length(query3$totaldistance[query3$user_type == "Beginner Level" &
    query3$totaldistance < 5.49])/length(query3$totaldistance[
        query3$user_type == "Beginner Level"])


round(c(prob_noob, prob_mid, prob_pro),2)
```

```
## Beginner   |   Intermediate   |   Pro
    0.86            0.38             0.12
```

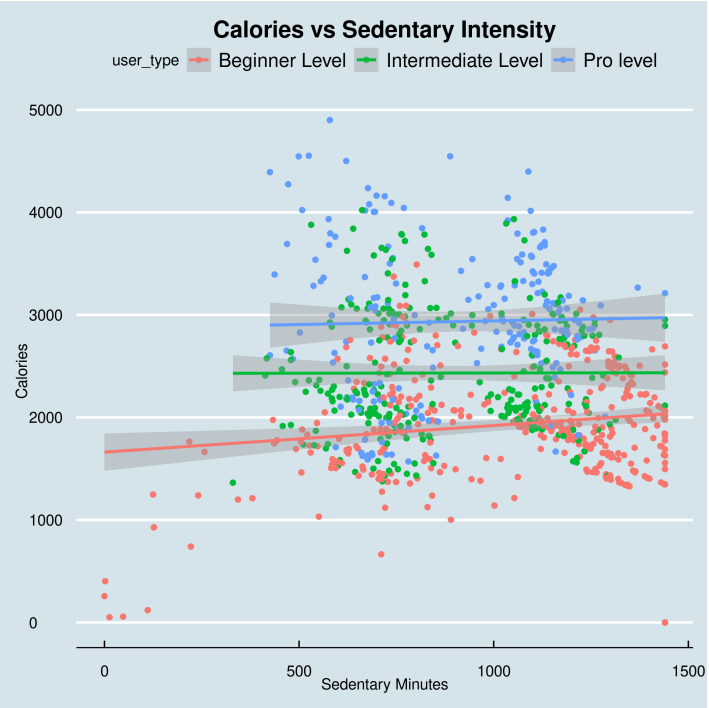
It is no surprise to find that **86% of beginners' runs** do not overcome 5.49 kilometers, while **62% of intermediate-level runs** do. Experienced users are pretty unlikely to record a run whose distance is under the average (only 12%).

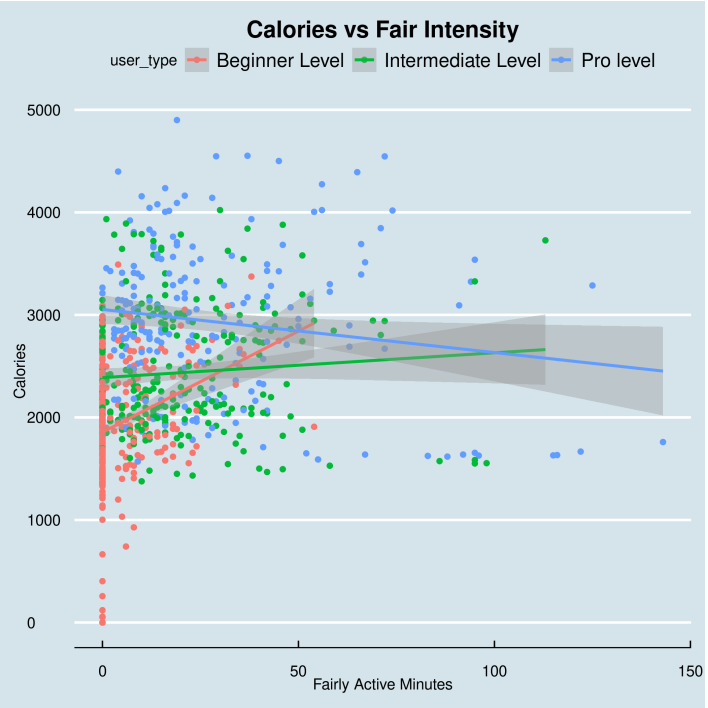
4.3 Intensity & Calories

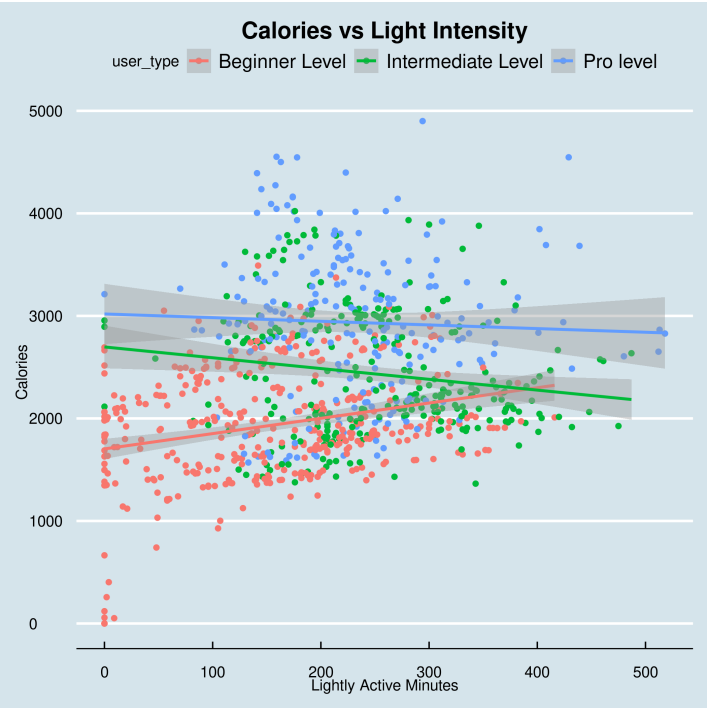
Moving on to other variables, the **relationship between intensity and number of burnt calories** is of considerable importance. Let us gradually define the former as *sedentary, fair, light* and *active*. Continuing with the user categorization of section 4.2, let's take a closer look at how these variables correlate conditional on the runner's type. Firstly we set up the data frame by making a SQL query and then we resort to R.

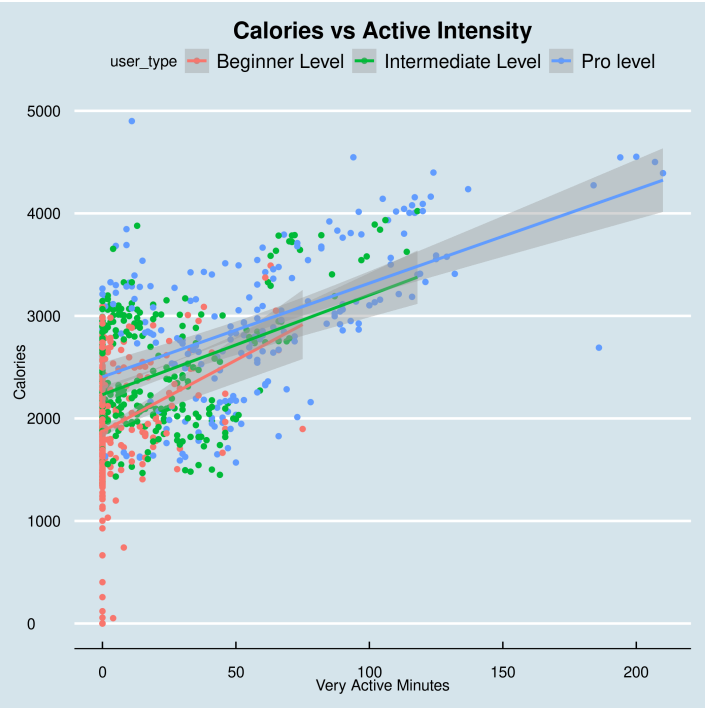
```
SELECT
    sedentaryminutes AS sedentary_minutes,
    lightlyactiveminutes AS lightly_minutes,
    fairlyactiveminutes AS fairly_active_minutes,
    veryactiveminutes AS very_active_minutes, calories,
     CASE
        WHEN ROUND(AVG(TotalDistance), 2) < 5 THEN 'Beginner'
        WHEN ROUND(AVG(TotalDistance), 2) >= 5 AND
        ROUND(AVG(TotalDistance), 2) < 8 THEN 'Intermediate'
     ELSE 'Pro level' END AS user_type
FROM daily_activity
GROUP BY
  id, sedentaryminutes, lightlyactiveminutes,
  fairlyactiveminutes, veryactiveminutes, calories;
```

```
## Sedentary Intensity Scatterplot

sedentary <- ggplot(data = query5,
        aes(y = calories, x = sedentary_minutes, color = user_type)) +
    geom_point() +
    theme_economist() + geom_smooth(method = "lm") +
    labs(title = "Calories vs Sedentary Intensity",
            x = "Sedentary Minutes", y = "Calories") +
    theme(plot.title = element_text(hjust = 0.5),
            plot.subtitle = element_text(hjust = 0.5, vjust = -2))

##Fair Intensity Scatterplot

fair <- ggplot(data = query5,
     aes(y = calories, x = fairly_active_minutes, color = user_type)) +
    geom_point() +
    theme_economist() + geom_smooth(method = "lm") +
    labs(title = "Calories vs Fair Intensity",
            x = "Fairly Active Minutes", y = "Calories") +
    theme(plot.title = element_text(hjust = 0.5),
            plot.subtitle = element_text(hjust = 0.5, vjust = -2))

## Light Intensity Scatterplot

light <- ggplot(data = query5,
        aes(y = calories, x = lightly_minutes, color = user_type)) +
    geom_point() +
    theme_economist() + geom_smooth(method = "lm") +
    labs(title = "Calories vs Light Intensity",
            x = "Lightly Active Minutes", y = "Calories") +
    theme(plot.title = element_text(hjust = 0.5),
            plot.subtitle = element_text(hjust = 0.5, vjust = -2))

## Active Intensity Scatterplot

active <- ggplot(data = query5,
        aes(y = calories, x = very_active_minutes, color = user_type)) +
    geom_point() +
    theme_economist() + geom_smooth(method = "lm") +
    labs(title = "Calories vs Active Intensity",
            x = "Very Active Minutes", y = "Calories") +
    theme(plot.title = element_text(hjust = 0.5),
            plot.subtitle = element_text(hjust = 0.5, vjust = -2))
```





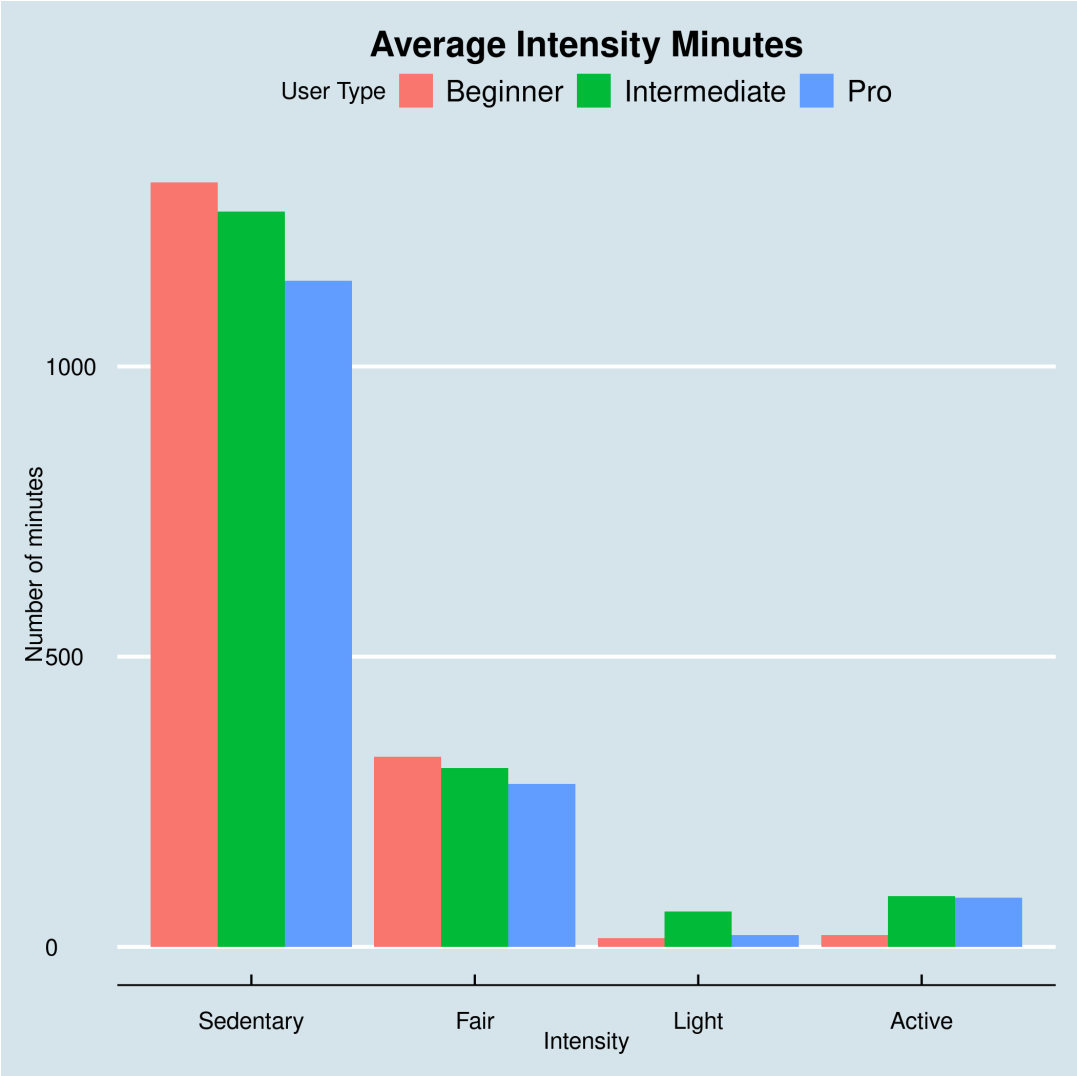




It can be observed that **the higher the number of running minutes, the more calories individuals burn**. Advanced runners tend to come on top of the rest, which is expected. Let us now take a closer look at the **average activity intensity minutes by user type**:

```sql
SELECT
    INTEN.id,
    ROUND(AVG(INTEN.sedentaryminutes), 2) AS average_sedentary,
    ROUND(AVG(INTEN.lightlyactiveminutes), 2) AS average_light,
    ROUND(AVG(INTEN.fairlyactiveminutes), 2) AS average_faire,
    ROUND(AVG(INTEN.veryactiveminutes), 2) AS average_active,
    CASE
        WHEN ROUND(AVG(ACT.TotalDistance), 2) < 5
        THEN 'Beginner Level'
        WHEN ROUND(AVG(ACT.TotalDistance), 2) >= 5 AND
        ROUND(AVG(ACT.TotalDistance), 2) < 8
        THEN 'Intermediate Level'
    ELSE 'Pro level'
    END AS user_type
FROM daily_activity ACT
JOIN daily_intensities INTEN
ON INTEN.id = ACT.id
GROUP BY INTEN.id
ORDER BY ROUND(AVG(ACT.TotalDistance), 2) DESC;
```

```r
## Changing the data format to 'long'

means.long <- melt(query6, id.vars = "user_type")

## Setting up our visualization

intensity_by_type <- ggplot(means.long,
        aes(x = variable, y = value, fill = user_type)) +
    geom_bar(stat = "identity", position = "dodge") +
    scale_fill_discrete(name = "User Type",
        labels = c("Beginner", "Intermediate", "Pro")) +
    scale_x_discrete(labels = c('Sedentary','Fair','Light', 'Active')) +
    xlab("Intensity") + ylab("Number of minutes") +
    theme_economist() +
    labs(title = "Average Intensity Minutes") +
    theme(plot.title = element_text(hjust = 0.5),
            plot.subtitle = element_text(hjust = 0.5, vjust = -2))
```



Sedentary minutes are ubiquitous across all groups, standing out how poorly runners perform at intensity. However, it is not surprising that any individual does not devote all of their time to doing sport. Consequently, the distribution shape of time allocation should not take us aback as, among other variables, working, sleeping and commuting take up an important number of hours.

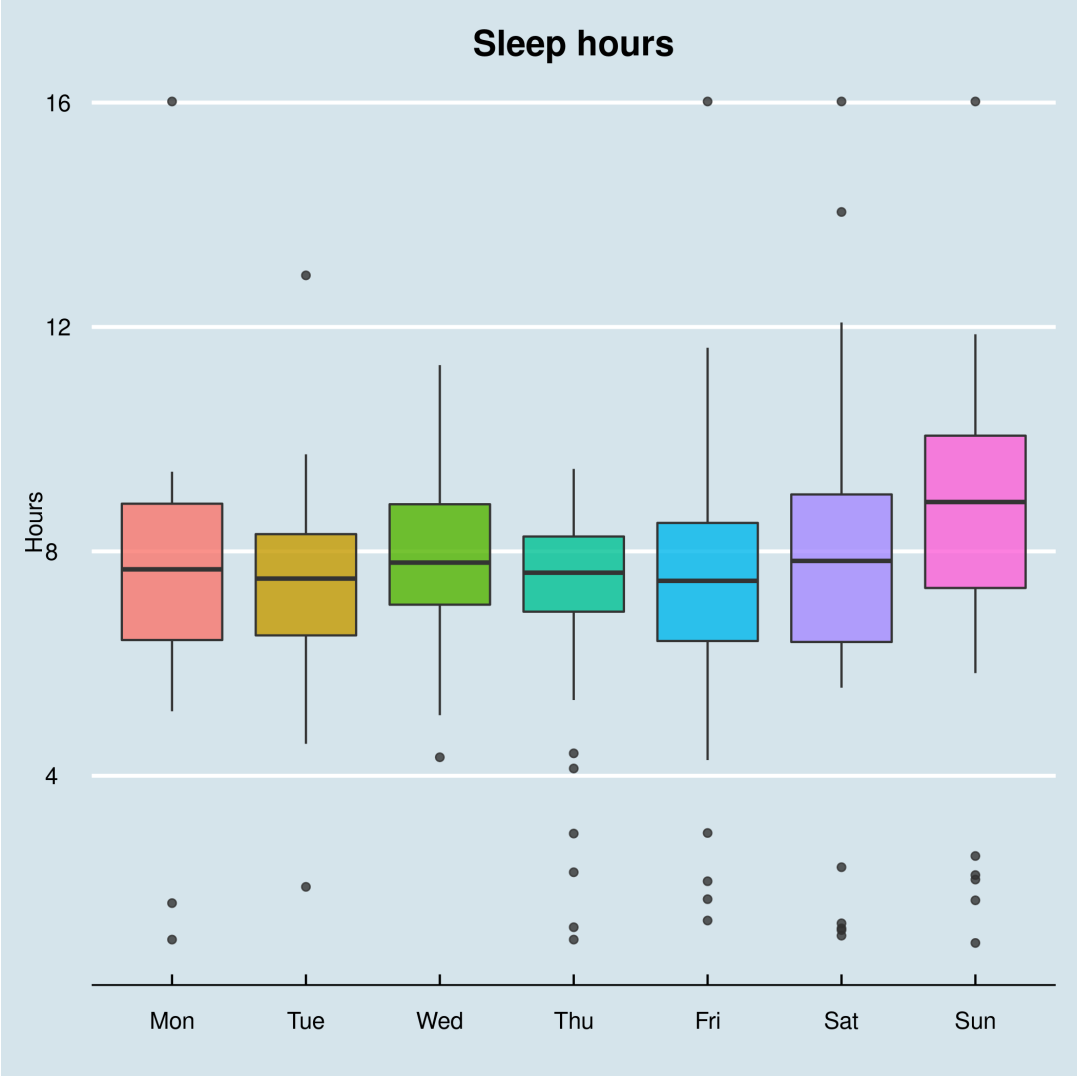
4.4 Sleep time

We can study the relationship between sleeping and running performance. Firstly, let's analyze the **distribution of daily average hours of sleep**:

```
SELECT
    TO_CHAR(TO_DATE(sleepday, 'MM/DD/YYYY'), 'Dy') AS dates,
    ROUND(totaltimeinbed/60, 2) AS bed_hours
FROM daily_sleep
GROUP BY dates, bed_hours
ORDER BY dates ASC;
```

Exporting the resulting data frame to a CSV file and reading it with R the following **boxplot** is retrieved:

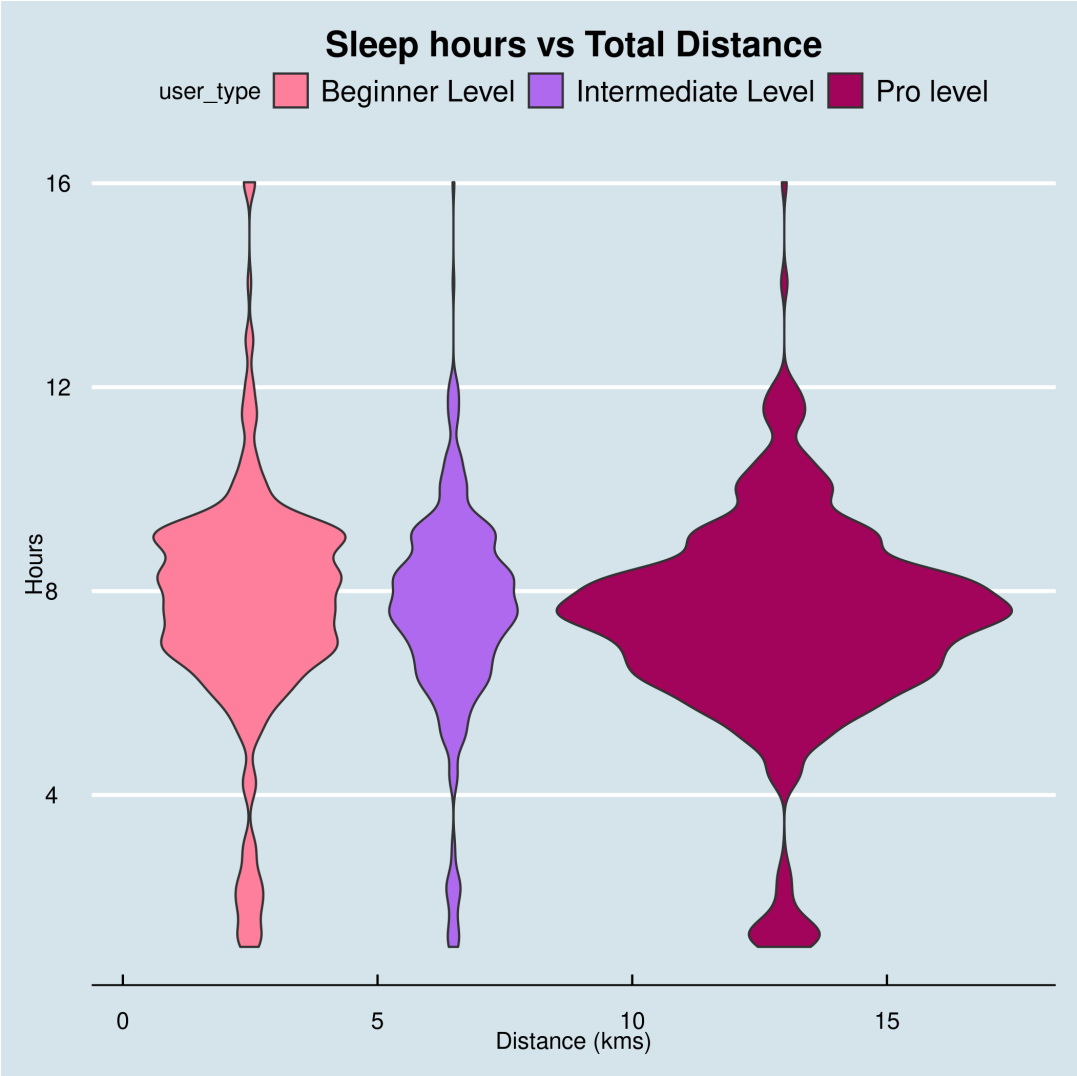
```
## Converting strings into factors so that we can manually order days

boxplot_data$dates <- factor(boxplot_data$dates, levels =
    c("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"))

## Boxplot

boxplot <- ggplot(data = boxplot_data,
        aes(x = dates, y = bed_hours, fill = dates)) +
    geom_boxplot(alpha = 0.8) +
    guides(fill = FALSE, color = FALSE) +
    theme(legend.position = "none") + theme_economist() +
    labs(title = "Sleep hours", x = "", y = "Hours") +
    theme(plot.title = element_text(hjust = 0.5),
            plot.subtitle = element_text(hjust = 0.5, vjust = -2))
```



Outliers in the lower tail of the distribution are present throughout all days, especially on Thursdays, Fridays and Sundays. There is a general **upward trend as weekend approaches**, which gives users the opportunity to increase their sleep time. Focusing on user type analysis:

```sql
SELECT
    ROUND(SLEEP.totaltimeinbed/60, 2) AS sleep_hours,
    ROUND(ACT.totaldistance, 2) AS distance,
    ACT.activitydate AS date,
    CASE
        WHEN ROUND(AVG(ACT.TotalDistance), 2) < 5
        THEN 'Beginner Level'
        WHEN ROUND(AVG(ACT.TotalDistance), 2) >= 5 AND
        ROUND(AVG(ACT.TotalDistance), 2) < 8
        THEN 'Intermediate Level'
    ELSE 'Pro level'
    END AS user_type
FROM daily_activity ACT
INNER JOIN daily_sleep SLEEP
ON SLEEP.id = ACT.id
GROUP BY ACT.id, sleep_hours, distance, date
ORDER BY date ASC;
```

```r
## Visualizing a violin plot

violin_plot <- ggplot(data = relations,
        aes(x = distance, y = sleep_hours, fill = user_type)) +
    geom_violin() + theme_economist() +
    scale_fill_manual(values = c("#FE7F9C" , "#AF69EF" , "#A1045A")) +
    scale_color_manual(values = c("#FE7F9C" , "#AF69EF" , "#A1045A")) +
    labs(title = "Sleep hours vs Total Distance",
        x = "Distance (kms)", y = "Hours") +
    theme(plot.title = element_text(hjust = 0.5),
            plot.subtitle = element_text(hjust = 0.5, vjust = -2))
```



No clear relationship between sleeping time and running distance can be drawn from the violin plot. Beginners and intermediate runners seem to show a stable sleeping schedule, centered around their respective means. High-level users' distance is volatile, suggesting there may be different subcategories within said group.

5 Regression Analysis

Leveraging the available data and taking into account that one of the top reasons people start running is the fast pace at which fat is lost, **it is possible to predict how many calories an individual will burn given certain characteristics**. In this section I propose a statistical way to attain such objective. Nevertheless, considering that we lack sleep information about some individuals, weight data is definitely scarce and we have no clue about users' age, sex and height, **Gauss-Markov assumptions are unlikely to hold**, specifically zero expectation of the error term conditional on the regressors: $E[\epsilon_i|X_1, X_2, ..., X_n] = 0$. Thus, **estimates will probably be biased**, namely: $E[\hat{\beta}_i] \neq \beta_i \ \forall$ i that $cov(X_i, \epsilon_i) \neq 0$. The drawbacks noted before should not refrain us from proceeding with the regression analysis, as we are not so interested in obtaining a clear causal relationship between burnt calories and other variables but rather gain general insights.

Let us estimate the following linear regression model with pooled OLS:

$$calories_{i,t} = \gamma_0 + \gamma_1 pro_{i,t} + \gamma_2 intermediate_{i,t} + \gamma_3 A_{i,t} + \gamma_4 L_{i,t} + \gamma_5 S_{i,t} + \gamma_6 bed_{i,t} + \epsilon_{i,t}$$

Where *pro* and *intermediate* are binary variables that take the value 1 if individual i is from that user type. Notice that we drop *beginner* to avoid perfect multicollinearity (so $\hat{\gamma}_0$ captures the effect of being a starter). *Active (A), light (L)* and *sedentary (S)* stand for running intensity measured in minutes and *bed* is total minutes in bed.

```
## Constructing the model

model <- lm(formula = calories ~ pro + intermediate + veryactiveminutes +
    lightlyactiveminutes + sedentaryminutes +
    totaltimeinbed, data = regression)

## Getting Results

model_summary <- tab_model(model)
```

Coefficient	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	997.4045	34.7779	28.68	< 2e-16
pro	624.2372	20.8093	30.00	< 2e-16
intermediate	68.9483	11.7858	5.85	5.04e-09
active	14.0105	0.1506	93.05	< 2e-16
light	2.3739	0.0570	41.68	< 2e-16
sedentary	0.5071	0.0213	23.79	< 2e-16
bed	0.0594	0.0385	1.54	0.1227

As we pointed out before, no causal effect can be retrieved from our model. However, there are some relevant aspects:

- Being an **experienced runner** is associated with burning, on average, **624.24 additional calories** in comparison with a beginner, ceteris paribus.
- Pertaining to the **intermediate-level group** also presents an advantage: on average they **burn 68.95 extra calories** than startes, ceteris paribus.
- On average, each additional minute at a **very active intensity is associated with burning 14 calories**, ceteris paribus.
- Marginal effect of an additional **minute at light intensity is associated with burning 2.38 calories**, ceteris paribus.
- The average marginal effect of an additional **sedentary minute is associated with burning 0.5 calories**, keeping the rest of the variables constant.
- The effect of **bed time is not statistically significant** (consistent with the visual intuition presented in section 4.4).

6 Recommendations

Based in the data analysis I have provided my final recommendations are:

- Develop the Bellabeat app in such a way that **essential information** such as weight, height and age **is mandatory** in order to sign up.

  How: Algorithm with an 'input' command that asks for this data with a loop that repeats said questions in case they have been incorrectly answered.

  Benefit: Further specific user data to be used for an enhanced segmentation.

- Implement the **algorithm** that was proposed in section 4.2 to categorize users. **Specific messages** can target runners according to their intensity/distance level. For instance, if the user is a beginner, the app could give advice on proper breathing for running, best way to limber up and congratulating messages once a new distance record is attained. In the case of intermediate or high-level users, news about upcoming competitions, top-notch diets and challenges seem more appropriate than giving out general wisdom that they most certainly already know.

  How: Algorithm that updates user's status based on latest available data. Linking path to running news and blogs that serve as a guide to improve.

  Benefit: Motivation to train and as a consequence higher usage of Bellabeat's app.

- Remind users of the relationship between burning calories and activity intensity as well a total distance. Display an interactive dashboard with up-to-date running statistics to show historical performance.

  How: Add new variables to the linear regression model proposed in section 5 and run the 'predict' code in R to show predicted burnt calories if the user increased running intensity or opted for longer distances.

  Benefit: Provide useful information specifically targetted to runners, ultimately nudging clients to set new goals and assiduously use the app.