Análisis de Datos de Dispositivos Inteligentes

Isabel Moreno

2024-09-28

Directorio de trabajo.

Es muy importante establecer el directorio de trabajo en R para poder realizar una correcta extracción de datos.

Podemos comprobar el directorio de trabajo en R con getwd() y si no es el correcto cambiar con setwd()

Es importante tener en cuenta que si trabajamos en Windows debemos introducir correctamente la dirección, ya que debemos invertir las barras de la dirección "/"

Instalación y Carga de Librerías.

En la carpeta de Datos contamos con 11 archivos para analizar.

Debemos tener en cuenta que debemos tener instaladas las librerías y cargarlas para poder utilizarlas.

Para instalar las librerías debemos utilizar la siguiente función de R install.packages("here") Para cargarlas la función library("here")

Como ya están instaladas sólo vamos a cargarlas en memoria, si es necesario se pueden instalar como se ha indicado más arriba.

Los paquetes más utilizados en limpieza de datos en el curso de Analista de Datos son: here, skimr, janitor y tidyverse.

```
library("here")
```

```
## here() starts at D:/google_analista/analisis_datos_bellabeat
```

```
library("skimr")
# permite limpiar los nombres de las columnas y manejar datos faltantes
de manera eficiente
library("janitor")
```

```
##
## Adjuntando el paquete: 'janitor'
```

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

# library("dplyer") no disponible para la versión de R que tengo.
# manipulación de datos y la visualización
library("tidyverse")

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.5
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ ggplot2   3.5.1     ✓ tibble    3.2.1
## ✓ lubridate 1.9.3     ✓ tidyr     1.3.1
## ✓ purrr     1.0.2

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

#fechas y horas
library("lubridate")
library(dplyr)
library(ggplot2)
```

Vista general de datos

Los datos utilizados para este estudio son de dominio público, disponibles a través de Mobius.

Este conjunto de datos contiene el seguimiento de la actividad física de usuarios que han dado su consentimiento para que sus datos sean tratados. Se incluye información sobre actividad física en minutos, ritmo cardíaco y sueño. Según el archivo weighLogInfo_merged.csv son 33 usuarios los que han prestado sus datos.

Respecto a la muestra, si es representativa respecto a la población, podemos indicar que puede estar sesgada, es decir que no sea representativa de la población, ya que puede haber sectores de la población que pueden estar más representados en esta muestra. En efecto, hay usuarios que pueden no haber dado sus datos porque velen por la privacidad de estos, aunque sean anónimos. Por otra parte, desconocemos el género de las personas que han dado consentimiento para la utilización de sus datos. La empresa Bellabeat, está especialmente enfocada a mujeres, y desconocemos si estos datos pueden ser representativos para esta empresa, no obstante, pueden dar ciertas ideas para describir tendencias, e influir en la estrategia de marketing de Bellabeat.

A continuación, detallo los archivos que tenemos para nuestro análisis, son un total de 11 archivos.

Archivo	Descripción
dailyActivity_merged.csv	Actividad Diaria. Los campos que nos encontramos en este archivo son: Id, ActivityDate, TotalSteps, TotalDistance, TrackerDistance, LoggedActivitiesDistance, VeryActiveDistance, ModeratelyActiveDistance, LightActiveDistance, SedentaryActiveDistance, VeryActiveMinutes, FairlyActiveMinutes, LightlyActiveMinutes, SedentaryMinutes, Calories
heartrate_seconds_merged.csv	Frecuencia Cardíaca en Segundos. Campos: Id, Time, Value
hourlyCalories_merged.csv	Calorías quemadas por horas. Campos: Id, ActivityHour, Calories
hourlyIntensities_merged.csv	Intensidad por horas. Campos: Id, Hora de actividad, Intensidad total, Intensidad promedio
hourlySteps_merged.csv	Horas de sueño. Campos: Id, ActivityHour, StepTotal
minuteCaloriesNarrow_merged.csv	Calorías gastadas por minuto. Campos: Id, ActivityMinute, Calories
minuteIntensitiesNarrow_merged.csv	Campos: Id, ActivityMinute, Intensity
minuteMETsNarrow_merged.csv	Campos: Id, ActivityMinute, METs
minuteSleep_merged.csv	Minutos de sueño. Campos: Id, date, value, logId
minuteStepsNarrow_merged.csv	Pasos por minuto. Campos: Id, ActivityMinute, Steps
weightLogInfo_merged.csv	Información sobre peso. Campos: Id, Date, WeightKg, WeightPounds, Fat, BMI, IsManualReport, LogId

- Metabolic Equivalent of Task (Equivalente Metabólico de Tarea). Un MET es una unidad de medida que indica la cantidad de energía que gasta una persona en una actividad en comparación con estar en reposo.

Una vez analizado los archivos parece que el principal es dailyActivity_merged.csv que agrupa detalles de todos los archivos. Los siguientes archivos pueden ser interesantes analizarlos:

heartrate_seconds_merged.csv: Da datos de frecuencia cardíaca en segundos. Esto puede ser útil para análisis detallados sobre la salud cardiovascular o el esfuerzo físico durante momentos específicos del día.

hourlyCalories_merged.csv, **hourlyIntensities_merged.csv**, y **hourlySteps_merged.csv**: Ofrecen datos por hora. Estos archivos pueden usarse para entender patrones de actividad o consumo de calorías en diferentes momentos del día.
minuteCaloriesNarrow_merged.csv, **minuteIntensitiesNarrow_merged.csv**, **minuteMETsNarrow_merged.csv**, **minuteStepsNarrow_merged.csv**: Ofrecen una resolución aún más alta (minuto a minuto), lo que permite un análisis detallado del comportamiento de las personas durante el día.
weightLogInfo_merged.csv: Proporciona información sobre el peso, el índice de masa corporal (IMC), y si el registro fue manual o automático. Esta información puede estar correlacionada con los niveles de actividad y las calorías quemadas.

Carga de archivos csv a variables.

Vamos a realizar una exploración del primer archivo, creamos una variable y cargamos los datos del csv.

Primera visualización de datos.

Para una primera aproximación a los datos utilizamos la función spec(), para comprobar el nombre de las columnas incluidas y los tipos de datos.

La función colnames() nos muestra los nombres de las columnas. Lo más signifativo que encontramos es que la fecha está grabada en caracteres, no como fecha.

```
actividad_diaria <- read_csv("datos/dailyActivity_merged.csv")
```

```
## Rows: 457 Columns: 15
## ── Column specification
──────────────────────────────────────────────────────
## Delimiter: ","
## chr  (1): ActivityDate
## dbl (14): Id, TotalSteps, TotalDistance, TrackerDistance,
LoggedActivitiesDi...
##
## i Use `spec()` to retrieve the full column specification for this
data.
## i Specify the column types or set `show_col_types = FALSE` to quiet
this message.
```

```
#compruebo nombre de variables y datos asignados.
spec(actividad_diaria)
```

```
## cols(
##   Id = col_double(),
##   ActivityDate = col_character(),
##   TotalSteps = col_double(),
```

```
##   TotalDistance = col_double(),
##   TrackerDistance = col_double(),
##   LoggedActivitiesDistance = col_double(),
##   VeryActiveDistance = col_double(),
##   ModeratelyActiveDistance = col_double(),
##   LightActiveDistance = col_double(),
##   SedentaryActiveDistance = col_double(),
##   VeryActiveMinutes = col_double(),
##   FairlyActiveMinutes = col_double(),
##   LightlyActiveMinutes = col_double(),
##   SedentaryMinutes = col_double(),
##   Calories = col_double()
## )
```

```
colnames(actividad_diaria)
```

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

Una función común que puede utilizar para obtener una vista previa de los datos es la función head() , que muestra las columnas y las primeras filas de datos

```
head(actividad_diaria)
```

```
## # A tibble: 6 × 15
##           Id ActivityDate TotalSteps TotalDistance TrackerDistance
##        <dbl> <chr>              <dbl>         <dbl>           <dbl>
## 1 1503960366 3/25/2016         11004          7.11            7.11
## 2 1503960366 3/26/2016         17609         11.6            11.6
## 3 1503960366 3/27/2016         12736          8.53            8.53
## 4 1503960366 3/28/2016         13231          8.93            8.93
## 5 1503960366 3/29/2016         12041          7.85            7.85
## 6 1503960366 3/30/2016         10970          7.16            7.16
## # ℹ 10 more variables: LoggedActivitiesDistance <dbl>,
## #   VeryActiveDistance <dbl>, ModeratelyActiveDistance <dbl>,
## #   LightActiveDistance <dbl>, SedentaryActiveDistance <dbl>,
## #   VeryActiveMinutes <dbl>, FairlyActiveMinutes <dbl>,
## #   LightlyActiveMinutes <dbl>, SedentaryMinutes <dbl>, Calories <dbl>
```

Las funciones str()y glimpse() devolverán resúmenes de cada columna de los datos organizados horizontalmente.

```
str(actividad_diaria)
```

```
## spc_tbl_ [457 × 15] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Id                       : num [1:457] 1.5e+09 ...
```

```
##  $ ActivityDate            : chr [1:457] "3/25/2016" "3/26/2016"
"3/27/2016" "3/28/2016" ...
##  $ TotalSteps              : num [1:457] 11004 17609 12736 13231 12041
...
##  $ TotalDistance           : num [1:457] 7.11 11.55 8.53 8.93 7.85 ...
##  $ TrackerDistance         : num [1:457] 7.11 11.55 8.53 8.93 7.85 ...
##  $ LoggedActivitiesDistance: num [1:457] 0 0 0 0 0 0 0 0 0 ...
##  $ VeryActiveDistance      : num [1:457] 2.57 6.92 4.66 3.19 2.16 ...
##  $ ModeratelyActiveDistance: num [1:457] 0.46 0.73 0.16 0.79 1.09 ...
##  $ LightActiveDistance     : num [1:457] 4.07 3.91 3.71 4.95 4.61 ...
##  $ SedentaryActiveDistance : num [1:457] 0 0 0 0 0 0 0 0 0 ...
##  $ VeryActiveMinutes       : num [1:457] 33 89 56 39 28 30 33 47 40 15
...
##  $ FairlyActiveMinutes     : num [1:457] 12 17 5 20 28 13 12 21 11 30
...
##  $ LightlyActiveMinutes    : num [1:457] 205 274 268 224 243 223 239
200 244 314 ...
##  $ SedentaryMinutes        : num [1:457] 804 588 605 1080 763 ...
##  $ Calories                : num [1:457] 1819 2154 1944 1932 1886 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Id = col_double(),
##   ..   ActivityDate = col_character(),
##   ..   TotalSteps = col_double(),
##   ..   TotalDistance = col_double(),
##   ..   TrackerDistance = col_double(),
##   ..   LoggedActivitiesDistance = col_double(),
##   ..   VeryActiveDistance = col_double(),
##   ..   ModeratelyActiveDistance = col_double(),
##   ..   LightActiveDistance = col_double(),
##   ..   SedentaryActiveDistance = col_double(),
##   ..   VeryActiveMinutes = col_double(),
##   ..   FairlyActiveMinutes = col_double(),
##   ..   LightlyActiveMinutes = col_double(),
##   ..   SedentaryMinutes = col_double(),
##   ..   Calories = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
glimpse(actividad_diaria)
```

```
## Rows: 457
## Columns: 15
## $ Id                       <dbl> 1503960366, 1503960366, 1503960366,
150396036…
## $ ActivityDate             <chr> "3/25/2016", "3/26/2016",
"3/27/2016", "3/28/…
## $ TotalSteps               <dbl> 11004, 17609, 12736, 13231, 12041,
10970, 122…
## $ TotalDistance            <dbl> 7.11, 11.55, 8.53, 8.93, 7.85, 7.16,
```

```
7.86, 7.…
## $ TrackerDistance          <dbl> 7.11, 11.55, 8.53, 8.93, 7.85, 7.16,
7.86, 7.…
## $ LoggedActivitiesDistance <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, …
## $ VeryActiveDistance       <dbl> 2.57, 6.92, 4.66, 3.19, 2.16, 2.36,
2.29, 3.3…
## $ ModeratelyActiveDistance <dbl> 0.46, 0.73, 0.16, 0.79, 1.09, 0.51,
0.49, 0.8…
## $ LightActiveDistance      <dbl> 4.07, 3.91, 3.71, 4.95, 4.61, 4.29,
5.04, 3.6…
## $ SedentaryActiveDistance  <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.0…
## $ VeryActiveMinutes        <dbl> 33, 89, 56, 39, 28, 30, 33, 47, 40,
15, 43, 3…
## $ FairlyActiveMinutes      <dbl> 12, 17, 5, 20, 28, 13, 12, 21, 11,
30, 18, 18…
## $ LightlyActiveMinutes     <dbl> 205, 274, 268, 224, 243, 223, 239,
200, 244, …
## $ SedentaryMinutes         <dbl> 804, 588, 605, 1080, 763, 1174, 820,
866, 636…
## $ Calories                 <dbl> 1819, 2154, 1944, 1932, 1886, 1820,
1889, 186…
```

Comprobar datos faltantes

Con la función clean_names() podemos limpiar los nombres de las columnas para que sean más uniformes y fáciles de manejar.

mutate(activity_date = mdy(activity_date)): Convierte la fecha de actividad a un formato de fecha reconocible. rename(): Renombrar columnas como activity_date a date y total_steps a steps.

select(-c(5:10)) elimina columnas en el caso de que hiciera falta, en este caso las vamos a mantener.

Vamos a crear un nuevo marco de datos para la limpieza básica de datos, renombrar columnas y cambiar el tipo de dato de la variable activity_date, ya que se encuentra en character y al ser una fecha, es mejor convertirla a fecha (date), y comprobamos como se han realizado los cambios con str() y colnames()

```
# Creo un nuevo marco de datos para la Limpieza básica de datos
daily <- actividad_diaria %>%
  clean_names() %>% #cambia los nombres
  mutate(activity_date=mdy(activity_date)) %>% # cambia tipo de dato a
fecha.
  #rename(steps = total_steps) %>%
 # select(-c(5:10)) %>%
  distinct()  # Eliminar duplicados
```

```r
colnames(daily)
```

```
##  [1] "id"                          "activity_date"
##  [3] "total_steps"                 "total_distance"
##  [5] "tracker_distance"            "logged_activities_distance"
##  [7] "very_active_distance"        "moderately_active_distance"
##  [9] "light_active_distance"       "sedentary_active_distance"
## [11] "very_active_minutes"         "fairly_active_minutes"
## [13] "lightly_active_minutes"      "sedentary_minutes"
## [15] "calories"
```

```r
str(daily)
```

```
## tibble [457 × 15] (S3: tbl_df/tbl/data.frame)
##  $ id                          : num [1:457] 1.5e+09 ...
##  $ activity_date               : Date[1:457], format: "2016-03-25"
"2016-03-26" ...
##  $ total_steps                 : num [1:457] 11004 17609 12736 13231
12041 ...
##  $ total_distance              : num [1:457] 7.11 11.55 8.53 8.93 7.85
...
##  $ tracker_distance            : num [1:457] 7.11 11.55 8.53 8.93 7.85
...
##  $ logged_activities_distance: num [1:457] 0 0 0 0 0 0 0 0 0 ...
##  $ very_active_distance        : num [1:457] 2.57 6.92 4.66 3.19 2.16
...
##  $ moderately_active_distance: num [1:457] 0.46 0.73 0.16 0.79 1.09
...
##  $ light_active_distance       : num [1:457] 4.07 3.91 3.71 4.95 4.61
...
##  $ sedentary_active_distance : num [1:457] 0 0 0 0 0 0 0 0 0 ...
##  $ very_active_minutes         : num [1:457] 33 89 56 39 28 30 33 47 40
15 ...
##  $ fairly_active_minutes       : num [1:457] 12 17 5 20 28 13 12 21 11
30 ...
##  $ lightly_active_minutes      : num [1:457] 205 274 268 224 243 223 239
200 244 314 ...
##  $ sedentary_minutes           : num [1:457] 804 588 605 1080 763 ...
##  $ calories                    : num [1:457] 1819 2154 1944 1932 1886
...
```

Como los datos obtenidos son registros por día del mismo usuario, realizo una agrupación de registros por usuario, y calculamos la media de los datos de cada variable.

Con la función dim podemos ver cuántas filas y columnas tienen nuestros datos. 35 usuarios y 14 variables de registro.

```r
print(colnames(actividad_diaria))
```

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

```
# Agrupar por Id y calcular la media
daily_summary <- actividad_diaria %>%
  group_by(Id) %>%
  summarise(
    mean_steps = mean(TotalSteps, na.rm = TRUE),
    mean_distance = mean(TotalDistance, na.rm = TRUE),
    mean_tracker_distance = mean(TrackerDistance, na.rm = TRUE),
    mean_logged_distance = mean(LoggedActivitiesDistance, na.rm = TRUE),
    mean_very_active_distance = mean(VeryActiveDistance, na.rm = TRUE),
    mean_moderately_active_distance = mean(ModeratelyActiveDistance,
na.rm = TRUE),
    mean_light_active_distance = mean(LightActiveDistance, na.rm = TRUE),
    mean_sedentary_active_distance = mean(SedentaryActiveDistance, na.rm
= TRUE),
    mean_very_active_minutes = mean(VeryActiveMinutes, na.rm = TRUE),
    mean_fairly_active_minutes = mean(FairlyActiveMinutes, na.rm = TRUE),
    mean_lightly_active_minutes = mean(LightlyActiveMinutes, na.rm =
TRUE),
    mean_sedentary_minutes = mean(SedentaryMinutes, na.rm = TRUE),
    mean_calories = mean(Calories, na.rm = TRUE)
  )

# Ver el resumen
print(daily_summary)
```

```
## # A tibble: 35 × 14
##           Id mean_steps mean_distance mean_tracker_distance
mean_logged_distance
##        <dbl>      <dbl>         <dbl>                 <dbl>
<dbl>
##  1    1.50e9     11641.          7.61                  7.61
0
##  2    1.62e9      4226.          2.75                  2.75
0
##  3    1.64e9      9275.          6.75                  6.75
0
##  4    1.84e9      3641.          2.41                  2.41
0
##  5    1.93e9      2181.          1.51                  1.51
0
##  6    2.02e9     12175.          8.77                  8.77
```

```
0
##  7    2.03e9      3393.          2.10                  2.10
0
##  8    2.32e9      3138.          2.12                  2.12
0
##  9    2.35e9      9800.          6.51                  6.51
0
## 10    2.87e9      6637.          4.47                  4.47
0
## # i 25 more rows
## # i 9 more variables: mean_very_active_distance <dbl>,
## #   mean_moderately_active_distance <dbl>, mean_light_active_distance
<dbl>,
## #   mean_sedentary_active_distance <dbl>, mean_very_active_minutes
<dbl>,
## #   mean_fairly_active_minutes <dbl>, mean_lightly_active_minutes
<dbl>,
## #   mean_sedentary_minutes <dbl>, mean_calories <dbl>
```

```
dim(daily_summary)
```

```
## [1] 35 14
```

Comprobamos que el usuario 4388161847, tiene las medias a cero, sería conveniente quitar esta columna ya que puede desvirtualizar nuestros datos, al no tener resgistros puede desplazar las medidas estadísticas.

```
 # Eliminar el Id específico
daily_summary_filtered <- daily_summary %>%
  filter(Id != 4388161847)
print(daily_summary_filtered)
```

```
## # A tibble: 34 × 14
##           Id mean_steps mean_distance mean_tracker_distance
mean_logged_distance
##        <dbl>      <dbl>         <dbl>                 <dbl>
<dbl>
##  1    1.50e9     11641.          7.61                  7.61
0
##  2    1.62e9      4226.          2.75                  2.75
0
##  3    1.64e9      9275.          6.75                  6.75
0
##  4    1.84e9      3641.          2.41                  2.41
0
##  5    1.93e9      2181.          1.51                  1.51
0
##  6    2.02e9     12175.          8.77                  8.77
0
##  7    2.03e9      3393.          2.10                  2.10
0
```

```
##  8    2.32e9      3138.          2.12                  2.12
0
##  9    2.35e9      9800.          6.51                  6.51
0
## 10    2.87e9      6637.          4.47                  4.47
0
## # ℹ 24 more rows
## # ℹ 9 more variables: mean_very_active_distance <dbl>,
## #   mean_moderately_active_distance <dbl>, mean_light_active_distance
<dbl>,
## #   mean_sedentary_active_distance <dbl>, mean_very_active_minutes
<dbl>,
## #   mean_fairly_active_minutes <dbl>, mean_lightly_active_minutes
<dbl>,
## #   mean_sedentary_minutes <dbl>, mean_calories <dbl>
```

La primera variable que vamos a estudiar son los pasos dados, según la Organización Mundial de la Salud, se considera que dar alrededor de 10.000 pasos diarios es una meta saludable para mantenerse activo.

Para ello establezco los siguientes límites:

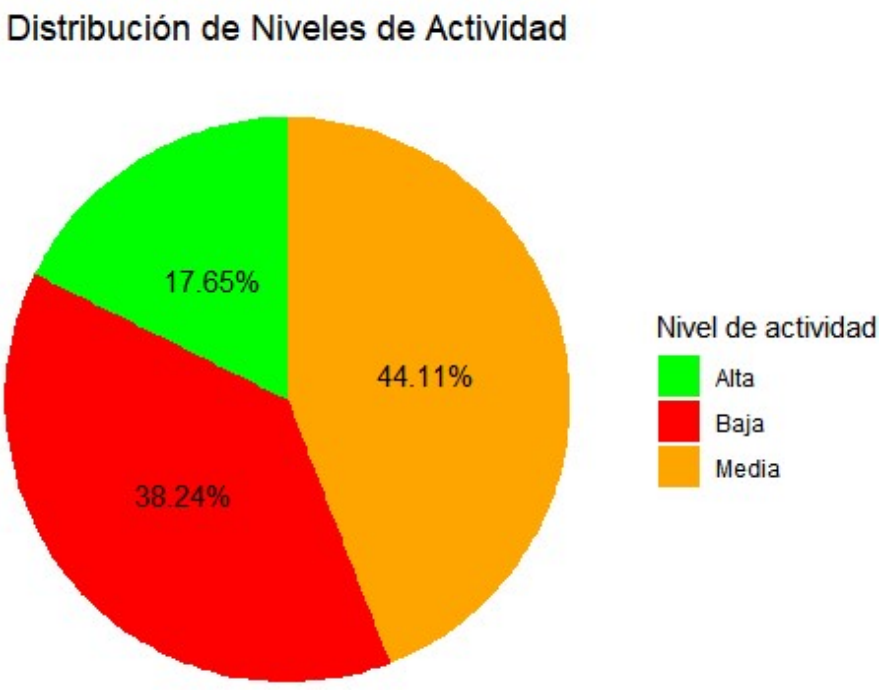
Baja: Menos de 4500 pasos. Media: Entre 4500 y 9900 pasos. Alta: Más de 9900 pasos.

```r
# Crear una nueva columna que clasifique la actividad con los nuevos
umbrales en daily_summary_filtered
 <- daily_summary_filtered %>%
  mutate(activity_level = case_when(
    mean_steps < 4500 ~ "Baja",
    mean_steps >= 4500 & mean_steps < 9900 ~ "Media",
    mean_steps >= 9900 ~ "Alta"
  ))

# Contar la frecuencia de cada nivel de actividad
activity_frequencies <- daily_summary_filtered %>%
  count(activity_level)

# Calcular los porcentajes y agregarlos a activity_frequencies
total <- sum(activity_frequencies$n)
activity_frequencies <- activity_frequencies %>%
  mutate(percentage = round(n / total * 100, 2))

# Ajustar el último porcentaje para que sume 100
activity_frequencies$percentage[nrow(activity_frequencies)] <-
  100 - sum(activity_frequencies$percentage[-nrow(activity_frequencies)])

print(activity_frequencies)
```

```
## # A tibble: 3 × 3
##   activity_level     n percentage
```

```
##   <chr>          <int>      <dbl>
## 1 Alta               6       17.6
## 2 Baja              13       38.2
## 3 Media             15       44.1
```

```r
activity_frequencies <- activity_frequencies %>%
  filter(n > 0)

# Crear gráfico de tarta con porcentajes
ggplot(activity_frequencies, aes(x = "", y = n, fill = activity_level)) +
  geom_bar(width = 1, stat = "identity") +   # Crear las barras que se convertirán en secciones de la tarta
  coord_polar("y") +                         # Convertir el gráfico de barras en un gráfico de tarta
  labs(fill = "Nivel de actividad", title = "Distribución de Niveles de Actividad") +
  theme_void() +                             # Eliminar ejes y cuadrícula para que se vea como tarta
  scale_fill_manual(values = c("Baja" = "red", "Media" = "orange", "Alta" = "green")) +  # Asignar colores
  theme(plot.title = element_text(hjust = 0.5)) +  # Centrar el título del gráfico
  geom_text(aes(label = paste0(percentage, "%")),
            position = position_stack(vjust = 0.5), color = "black")  # Añadir etiquetas de porcentaje
```



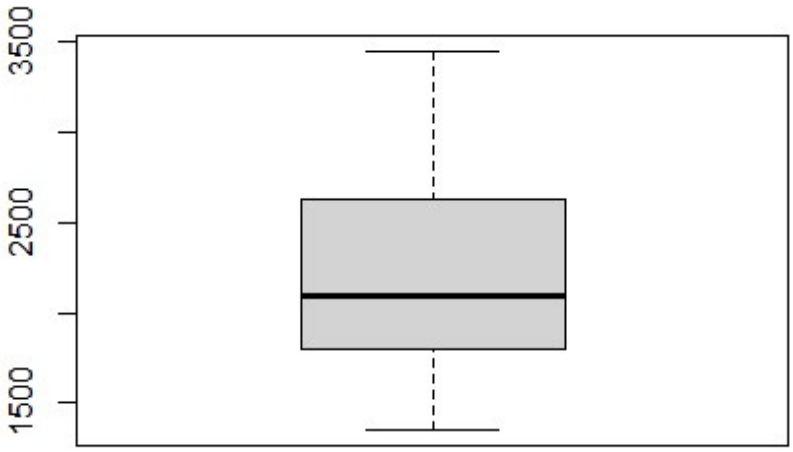
Según esta división podemos comprobar, que de los sujetos que se han presentado para estos datos el 44.11% realiza una actividad media, entre 4500 y 9500 pasos diarios, puede ser un sesgo que los que hayan dado sus datos para este estudio puedan ser personas más activas, y queden fuera de este estudio personas que no realicen pasos diarios.

Observamos también que el 38.24% de los usuarios, realizan menos de 4500 pasos diarios y el 17.65% realizan una actividad por encima de los 9900 pasos.

– Al crear el gráfico de tarta, me ha dado un error, he tenido que añadir la línea: activity_frequencies <- activity_frequencies %>% filter(n > 0) El error se debió a la falta de filtrado de categorías con valores nulos o cero, lo que resultó en la creación de porciones vacías en el gráfico. Al filtrar esos datos y ajustar el código, logramos generar un gráfico de tarta claro y representativo.

Otra variable interesante puede ser el Consumo de Calorías y relacionarlas con los pasos dados.

```
# Resumen estadístico de las calorías
summary(daily_summary_filtered$mean_calories)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1353    1798    2094    2196    2589    3451

boxplot(daily_summary_filtered$mean_calories)
```



Esto nos indica:

El valor mínimo: 1353 calorías, este valor es el más bajo en consumo de calorías.

El 1er Cuartil (Q1): 1798 calorías. El 25% de los datos tienen un consumo de calorías inferior a este valor. Esto sugiere que un cuarto de los sujetos consumen menos de 1798 calorías.

La mediana es el valor central de tu conjunto de datos, lo que significa que el 50% de los sujetos consumen menos de 2094 calorías. Es un buen indicador del consumo típico.

La media es un poco mayor que la mediana, lo que puede indicar que hay algunos valores atípicos (consumos de calorías mucho más altos) que están influyendo en este promedio. 3er Cuartil (Q3): 2589 calorías

El 75% de los sujetos tienen un consumo de calorías por debajo de este valor, lo que indica que un cuarto de ellos consume más de 2589 calorías.

El valor Máximo: 3451 calorías

```
# Calcular la correlación entre mean_steps y mean_calories
correlation <- cor(daily_summary_filtered$mean_steps,
daily_summary_filtered$mean_calories)
print(paste("Correlación entre pasos y calorías:", round(correlation,
2)))
```

```
## [1] "Correlación entre pasos y calorías: 0.47"
```

Este valor nos indica que existe una relación moderada positiva, aunque pueden influir otras variables, como por ejemplo el tipo de ejercicio físico y la dieta que se tiene. Incluyo más variables en el estudio.

```
# Modelo de regresión múltiple
model <- lm(mean_calories ~ mean_steps + mean_distance +
              mean_very_active_distance + mean_moderately_active_distance
+
              mean_light_active_distance + mean_very_active_minutes +
              mean_fairly_active_minutes + mean_lightly_active_minutes,
            data = daily_summary_filtered)

# Resumen del modelo
summary(model)
```

```
##
## Call:
## lm(formula = mean_calories ~ mean_steps + mean_distance +
mean_very_active_distance +
##     mean_moderately_active_distance + mean_light_active_distance +
##     mean_very_active_minutes + mean_fairly_active_minutes +
mean_lightly_active_minutes,
```

```
##     data = daily_summary_filtered)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -609.26 -166.60    3.99  175.27  418.05
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    1761.0591   119.8676  14.692 8.38e-14
***
## mean_steps                       -0.6926     0.1097  -6.311 1.32e-06
***
## mean_distance                  1686.4963   226.6003   7.443 8.54e-08
***
## mean_very_active_distance      -918.4426   178.6773  -5.140 2.59e-05
***
## mean_moderately_active_distance -494.3575   213.6386  -2.314 0.029175
*
## mean_light_active_distance     -793.8631   195.8386  -4.054 0.000432
***
## mean_very_active_minutes         20.6871     4.4126   4.688 8.36e-05
***
## mean_fairly_active_minutes        3.2368     3.8006   0.852 0.402488
## mean_lightly_active_minutes       1.9070     1.5549   1.226 0.231452
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 253.9 on 25 degrees of freedom
## Multiple R-squared:  0.8595, Adjusted R-squared:  0.8145
## F-statistic: 19.11 on 8 and 25 DF,  p-value: 7.405e-09
```

mean_steps: Por cada paso adicional que una persona camina, las calorías consumidas disminuyen en 0.69 calorías, lo cual podría parecer extraño, ya que generalmente se espera que a más pasos dados haya un mayor gasto calórico. Sin embargo, es importante considerar el contexto y los otros factores en juego. Faltaría más información para interpretar esto, por ejemplo, tipo de dieta seguida y una especificación de los ejercicios realizados.

mean_light_active_distance: Este coeficiente negativo de -793.86 indica que un aumento en la distancia recorrida de manera ligera se asocia con una disminución significativa en las calorías consumidas.

En este punto resulta interesante analizar los datos del archivo:

weightLogInfo_merged.csv: Proporciona información sobre el peso, el índice de masa corporal (IMC), y si el registro fue manual o automático. Esta información puede estar correlacionada con los niveles de actividad y las calorías quemadas

```r
setwd("D:/google_analista/analisis_datos_bellabeat")

getwd()
```

```
## [1] "D:/google_analista/analisis_datos_bellabeat"
```

```r
peso <- read_csv("datos/weightLogInfo_merged.csv")
```

```
## Rows: 33 Columns: 8
## ── Column specification
──────────────────────────────────────────────────────
## Delimiter: ","
## chr (1): Date
## dbl (6): Id, WeightKg, WeightPounds, Fat, BMI, LogId
## lgl (1): IsManualReport
##
## ℹ Use `spec()` to retrieve the full column specification for this
data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet
this message.
```

```r
#compruebo nombre de variables y datos asignados.
spec(peso)
```

```
## cols(
##   Id = col_double(),
##   Date = col_character(),
##   WeightKg = col_double(),
##   WeightPounds = col_double(),
##   Fat = col_double(),
##   BMI = col_double(),
##   IsManualReport = col_logical(),
##   LogId = col_double()
## )
```

```r
colnames(peso)
```

```
## [1] "Id"             "Date"           "WeightKg"       "WeightPounds"
## [5] "Fat"            "BMI"            "IsManualReport" "LogId"
```

```r
# Contar los IDs únicos
num_unique_ids <- length(unique(peso$Id))
print(num_unique_ids)
```

```
## [1] 11
```

En este caso sólo 11 de los sujetos que han cedido sus datos han dado su peso, por lo tanto, al no tener el mínimo de datos requeridos, que son como mínimo para una muestra 30 no podemos obtener conclusiones definitivas si quisiéramos cruzar los datos con el consumo de calorías.

Lo que si observamos es que algunos registros se han obtenido de manera manual y que solo 11 usuarios han facilitado los datos, sería interesante diseñar los dispositivos para que sean ligeros, fácil de llevar y de utilizar, así como la recogida automática de datos para facilitar la labor a los usuarios. Con esto conseguiríamos realizar un seguimiento del peso, y podríamos relacionarlos con las calorías consumidas, y dar consejos en cuanto a pérdida de peso y salud.

Con la función summary() obtenemos unas estadísticas básicas sobre los datos recogidos en este archivo. Se observa una gran dispersión en los datos que van desde 53.30 hasta 129.60

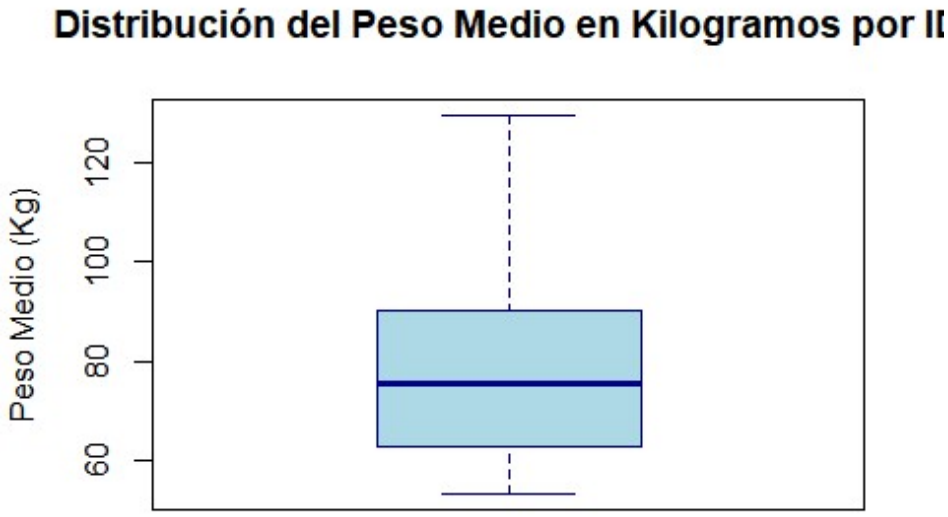
Podemos realizar una visualización de los datos. Observamos que hay un dato que está distorsionando, el valor de 129.60 Kg.

Con la función plot podemos comprobar la distribución de los resultados en cuanto a peso.

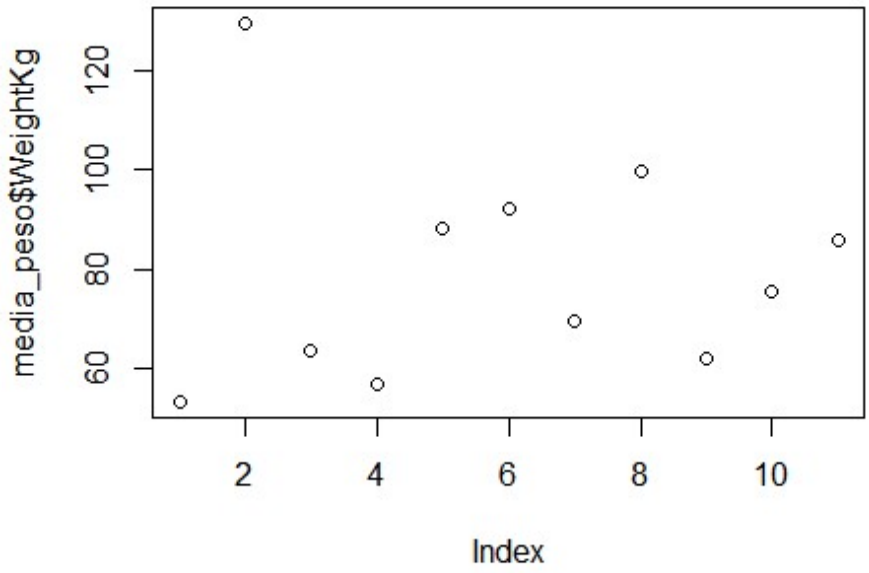
```
peso$WeightKg <- as.numeric(as.character(peso$WeightKg))

# Calcular la media de WeightKg por ID
media_peso <- aggregate(WeightKg ~ Id, data = peso, FUN = mean, na.rm =
TRUE)

# Verificar si se ha calculado correctamente
print(media_peso)
```

```
##            Id  WeightKg
## 1  1503960366  53.30000
## 2  1927972279 129.60001
## 3  2347167796  63.40000
## 4  2873212765  56.95000
## 5  2891001357  88.40000
## 6  4445114986  92.40000
## 7  4558609924  69.40000
## 8  4702921684  99.70000
## 9  6962181067  61.86429
## 10 8253242879  75.60000
## 11 8877689391  85.75556
```

```
# Crear boxplot básico en R
boxplot(media_peso$WeightKg,
        main = "Distribución del Peso Medio en Kilogramos por ID",
        ylab = "Peso Medio (Kg)",
        col = "lightblue",
        border = "darkblue",
        outline = TRUE)  # Para asegurar que los outliers se muestren
```




```
plot(media_peso$WeightKg)
```



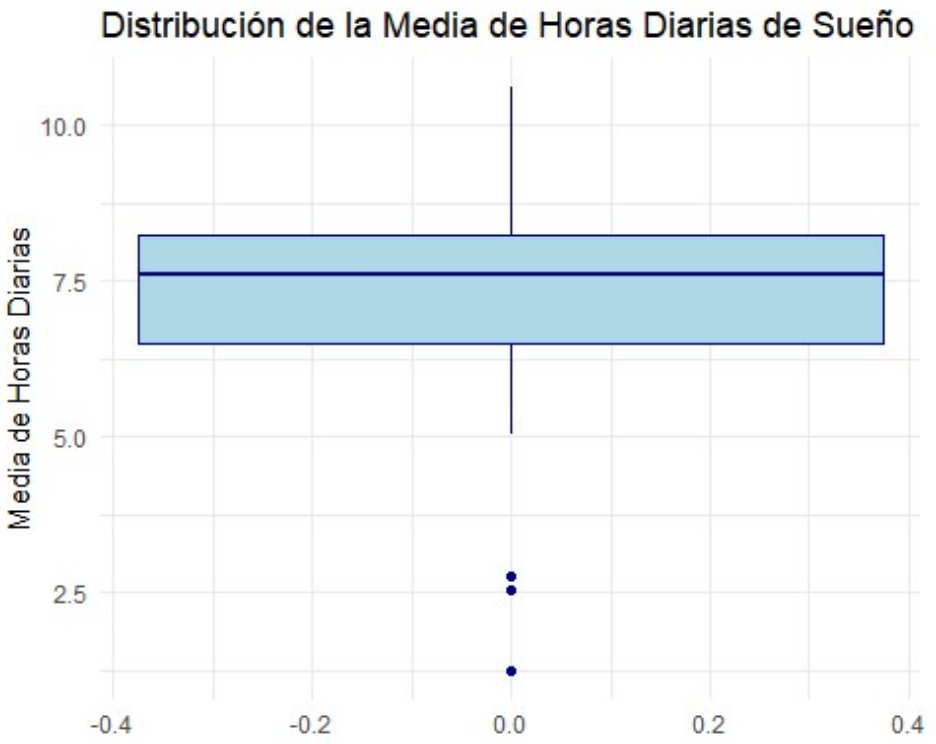
Otro conjunto de datos que me parece interesante son las horas de sueño, ya que fomentar hábitos que prioricen una buena calidad de sueño y actividad física regular puede ser fundamental para mejorar la salud y el bienestar. También puede ayudar a disminuir el estrés.

En este caso, como podemos comprobar tenemos los registros de 23 sujetos, tampoco coinciden con el total de registros del primer archivo.

Después de revisar los datos de sueño comprobamos que hay dos sujetos que duermen menos de tres horas diarias, esto es muy inusual, sería conveniente revisar la manera de registrar los datos, como no lo podemos hacer, podemos tomar como referencia para nuestra estrategia de marketing la mejora de la medida del sueño en nuestros dispositivos y facilitar su uso, para que todos los usuarios puedan hacer registros.

```
sueno <- read_csv("datos/minuteSleep_merged.csv")

## Rows: 198559 Columns: 4
## ── Column specification
──────────────────────────────────────────────────────
## Delimiter: ","
## chr (1): date
## dbl (3): Id, value, logId
##
## ℹ Use `spec()` to retrieve the full column specification for this
data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet
this message.

#compruebo nombre de variables y datos asignados.

head(sueno)

## # A tibble: 6 × 4
##           Id date                 value       logId
##        <dbl> <chr>                <dbl>       <dbl>
## 1 1503960366 3/13/2016 2:39:30 AM     1 11114919637
## 2 1503960366 3/13/2016 2:40:30 AM     1 11114919637
## 3 1503960366 3/13/2016 2:41:30 AM     1 11114919637
## 4 1503960366 3/13/2016 2:42:30 AM     1 11114919637
## 5 1503960366 3/13/2016 2:43:30 AM     1 11114919637
## 6 1503960366 3/13/2016 2:44:30 AM     1 11114919637

# Convertir la columna 'date' al formato de fecha
sueno$date <- as.POSIXct(sueno$date, format = "%m/%d/%Y %I:%M:%S %p")


# Agrupar por 'Id' y fecha, sumar minutos diarios
sueno_diario <- sueno %>%
  mutate(date_only = as.Date(date)) %>%
```

```r
  group_by(Id, date_only) %>%
  summarise(total_minutos_sueno = sum(value))

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

# Calcular la media diaria de minutos de sueño por sujeto
media_sueno_por_id <- sueno_diario %>%
  group_by(Id) %>%
  summarise(media_minutos_diarios = mean(total_minutos_sueno))

# Convertir minutos a horas
media_sueno_por_id <- media_sueno_por_id %>%
  mutate(media_horas_diarias = media_minutos_diarios / 60)

# Ver el resultado
print(media_sueno_por_id)

## # A tibble: 23 × 3
##            Id media_minutos_diarios media_horas_diarias
##         <dbl>                 <dbl>               <dbl>
##  1 1503960366                  418.                6.97
##  2 1644430081                  630                10.5
##  3 1844505072                  636.               10.6
##  4 1927972279                  390.                6.51
##  5 2022484408                  153                 2.55
##  6 2026352035                  554.                9.24
##  7 2347167796                  482.                8.03
##  8 3977333714                  556.                9.26
##  9 4020332650                  407.                6.78
## 10 4319703577                  474.                7.90
## # ℹ 13 more rows

# Representación de las medias de horas diarias por persona
ggplot(media_sueno_por_id, aes(y = media_horas_diarias)) +
  geom_boxplot(fill = "lightblue", color = "darkblue") +
  labs(title = "Distribución de la Media de Horas Diarias de Sueño",
       y = "Media de Horas Diarias") +
  theme_minimal()
```



El diagrama de cajas distribuye las observaciones en cuatro grupos, que contienen aproximadamente el mismo número de observaciones, pero no todos los grupos ocupan el mismo espacio. La caja corresponde a los datos que van del Q1 al Q3.

Las horas aconsejadas para sueño están entre 7 y 8 horas, los sujetos analizados tienen una media de 7,5, sería interesante fomentar más horas de sueño en aquellas personas que no llegan a la media.

Conclusiones del Análisis de Datos de Dispositivos Inteligentes

Muestra y Representatividad:

El conjunto de datos proviene de 33 usuarios que han consentido compartir su información, lo que puede introducir sesgos. Es importante considerar que podría haber subrepresentación de ciertos sectores de la población, particularmente aquellos que valoran su privacidad, y no se ha podido determinar a qué género pertenecen las personas del estudio. Recordemos que la empresa se centra en mujeres.

Se ha intentado añadir datos públicos al estudio, pero no ha sido posible, ya que no se han encontrado. Sería interesante que se pudieran analizar los datos obtenidos en la propia empresa porque estarían más contextualizados y enfocados al público objetivo.

Resumen de Datos:

Se analizaron 11 archivos de datos relacionados con la actividad física, frecuencia cardíaca, calorías, pasos y registros de sueño.

El archivo dailyActivity_merged.csv se ha identificado como el principal, proporcionando la mayor cantidad de información relevante.

Actividad Física:

Actividad física

Se clasifica la actividad física de los usuarios en tres niveles: baja (menos de 4500 pasos), media (4500-9900 pasos) y alta (más de 9900 pasos). Los resultados mostraron que el 44.11% de los usuarios tiene actividad media, el 38.24% tiene baja actividad, y el 17.65% tiene alta actividad.

Esto sugiere que varias estrategias de marketing según el nivel de actividad.

Podríamos concluir que nuestro público objetivo para las campañas de marketing se pueden dividir en dos según la actividad que realizan:

- Baja y media actividad: para estas personas podemos desarrollar avisos y consejos para aumentar la salud aumentando el número de pasos para que lleguen a los 10000
- Alta: motivarlos a continuar, se podrían promover productos relacionados con el fitness, como ropa deportiva, zapatos de alta calidad o equipos de ejercicio que pueden ayudarles a seguir mejorando su rendimiento.

Para ambos:

- Ofrecer productos que sean ligeros, cómodos y lo más automático posibles para registrar su rendimiento físico.
- Utilización de una aplicación premium con consejos de profesionales de la salud para resolver dudas.

- Crear una comunidad de usuarios para compartir experiencias: aplicación gratuita.

Posibilidad de incluir hombres y niños-as en el target objetivo, esto depende ya de la dirección de la empresa.

Tendencias: ampliar hacia el cuidado de la salud en los hombres y niños-as.

Se observó una correlación moderada positiva entre los pasos dados y las calorías consumidas, aunque se identificaron factores que podrían influir en esta relación, como la dieta y el tipo de ejercicio, los datos aportados del peso no son suficientes para relacionar, lo que nos lleva de nuevo a diseñar dispositivos que sean cómodos, fáciles de llevar y que permitan un registro automático de los datos con garantías de privacidad para los usuarios.

Se realizó un análisis de regresión múltiple que indicó que el aumento en pasos y distancias activas se relaciona de manera compleja con el consumo calórico.

Datos de Peso:

Se observó que solo 11 usuarios proporcionaron datos sobre su peso, limitando la capacidad de hacer inferencias significativas. Las estadísticas de peso mostraron una gran dispersión, y un valor atípico (129.60 kg) podría distorsionar los resultados. Sería necesario recoger datos sobre edad, altura y peso para adecuar una dieta y mejorar la salud de los usuarios.

Recomendación: facilitar la recogida de datos, que también está relacionado con el punto anterior del diseño de los dispositivos.

Datos de sueño:

El análisis de los registros de sueño reveló que algunos usuarios reportan menos de tres horas de sueño, eliminando estos datos, se obtiene que la media de sueño es de 7,5 horas. Al presentar valores extremos, es necesario mejorar la recogida de datos por parte de nuestros productos, y facilitar su uso.

En este sentido, nuestros productos tienen que garantizar la privacidad de los datos conforme a la normativa vigente.