Helping BellaBeat learn from FitBit

2026-01-29

Contents

1 Introduction

This case study, part of the Google Data Analytics Capstone, analyzes Fitbit tracker data to provide actionable insights for Bellabeat's marketing strategy. Bellabeat is a high-tech manufacturer of health-focused smart products for womenand was co-founded by Urška Sršen and Sando Mur.

Note: This project will use R and SQL to explore the Fitness Tracker Data available at Kaggle

1.1 Business task

The goal of this analysis is to identify smart device usage trends and determine how these patterns can be applied to Bellabeat's customers to optimize their marketing strategy. It aims to answer the following 3 questions

1. What are some trends in smart device usage?
2. How could these trends apply to Bellabeat customers?
3. How could these trends help influence Bellabeat marketing strategy?

1.2 Key stakeholders

1. Urška Sršen: Bellabeat's cofounder and Chief Creative Officer
2. Sando Mur: Mathematician and Bellabeat's cofounder; key member of the Bellabeat executive team
3. Bellabeat marketing analytics team

1.3 Data Source and Description

Data source

The dataset, hosted on Kaggle, contains personal tracker data from 30 Fitbit users who consented to a distributed survey via Amazon Mechanical Turk. The data can be accessed from the following link:

https://www.kaggle.com/datasets/arashnic/fitbit

The orignal source of the data is the following:

https://zenodo.org/records/53894#.X9oeh3Uzaao

It was collected by

- Furberg, Robert
- Brinton, Julia
- Keating, Michael
- Ortiz, Alexa1

Data Description (as described by Authors)

As mentioned on Kaggle:

These datasets were generated by respondents to a distributed survey via Amazon Mechanical Turk between 03.12.2016-05.12.2016. Thirty eligible Fitbit users consented to the submission of personal tracker data, including minute-level output for physical activity, heart rate, and sleep monitoring. Individual reports can be parsed by export session ID (column A) or time stamp (column B). Variation between output represents use of different types of Fitbit trackers and individual tracking behaviors / preferences.

This data set has a total of 29 files spanning over two time periods:

1. 3-12-16 to 4-11-16
2. 4-12-16 to 5-12-16

This analysis focuses specifically on the period between April 12, 2016, and May 12, 2016

There are **18** CSV files for the following indicators for the given time period:

- Physical Activities (Daily, Hourly, Minute-level)
- Calories (Daily, Hourly, Minute-level)
- Daily Intensities (Daily, Hourly, Minute-level)
- Daily Steps (Daily, Hourly, Minute-level)
- Heart Rate in Seconds
- Sleep (Daily, Minute)
- Weight

The majority of the data is in a long format. However, minute-level output for intensities, steps, and calories from April to May is also provided in a wide format.

This case study is structured into two parts:

- Part 1 will analyze **Daily** data for the following indicators:
  - Daily Activity
  - Daily Calories
  - Daily Intensities
  - Daily Steps
  - Daily Sleep
  - Weight
- Part 2 will analyze **Hourly** data for the following indicators:
  - Hourly Calories
  - Hourly Intensities
  - Hourly Steps

1.4 Issues with bias or credibility in this data

The dataset has a limited sample size (**33** respondents) and lacks critical demographic information such as age, gender, and occupation. Additionally, the weight dataset is too small (8 respondents) to provide statistically significant insights and will be excluded.

To gain meaningful insight into seasonal distribution, data collection should span at least a full year. Two months is insufficient for drawing definitive conclusions, as it doesn't account for variation across an entire annual cycle.

1.5 Privacy concerns

The kaggle data is open source and all Fitbit users consented to the submission of personal tracker data removing any privacy concerns

1.6 Using R and SQL

This data study uses a combination of *R* and *SQL*. (via SQlite)

SQL will be used for all Data Processing and Cleaning upto and including the Join

R will be used for formatting any SQL data and Data Visualization and Plots (including any analysis and transformations post the join).

Setting Up

The code below loads up the relevant libraries and sets up an SQL connection via SQlite.

```
#Load libraries
library(tidyverse)
library(readxl)
library(readr)
```

```
library(dplyr)
library(ggplot2)
library(ggcorrplot)
library(cowplot)
library(knitr)
library(kableExtra)
library(DBI)
library(RSQLite)

#Set up the SQL connection with a new database created purely in memory.
# I am storing it in memory rather than disk as I do not need these tables
# outside of this Rmd file
con <- dbConnect(RSQLite::SQLite(), dbname = ":memory:")

#General Settings for Code chunks
knitr::opts_chunk$set(connection = con) # Enable the connection for all chunks
knitr::opts_chunk$set(fig.align = "center") # Align output when knitting the document
```

Creating tables from the CSV files

Dedicated CSV data files exist for each of the indicators mentioned above. The code below loads all six of these CSV files as separate tables for analysis using SQL

```
dbWriteTable(con, "dailyactivity_merged",
             read.csv("Fitbit_Data/dailyActivity_merged.csv"))
dbWriteTable(con, "dailyintensities_merged",
             read.csv("Fitbit_Data/dailyIntensities_merged.csv"))
dbWriteTable(con, "dailycalories_merged",
             read.csv("Fitbit_Data/dailyCalories_merged.csv"))
dbWriteTable(con, "dailysteps_merged",
             read.csv("Fitbit_Data/dailySteps_merged.csv"))
dbWriteTable(con, "sleepday_merged",
             read.csv("Fitbit_Data/sleepDay_merged.csv"))
dbWriteTable(con, "weightLogInfo_merged",
             read.csv("Fitbit_Data/weightLogInfo_merged.csv"))
dbWriteTable(con, "hourlyintensities_merged",
             read.csv("Fitbit_Data/hourlyIntensities_merged.csv"))
dbWriteTable(con, "hourlycalories_merged",
             read.csv("Fitbit_Data/hourlyCalories_merged.csv"))
dbWriteTable(con, "hourlysteps_merged",
             read.csv("Fitbit_Data/hourlySteps_merged.csv"))
```

2 Part 1: Daily Level Analysis

2.1 Data Processing and Cleaning

This section details the process for data processing and cleaning in preparation for the daily analysis.

Initially, it will identify the tables required for joining to derive information for meaningful insights. Subsequently, it will select the relevant columns from these tables through in-depth examination, data validation, and checks for duplicate and missing values. The final steps will involve cleaning and transforming the data, followed by joining it into a consolidated data frame.

2.1.1 Shortlisting Tables

As mentioned in the description, individual records in each data file represent an individual at a specific time period. This is significant because it implies that any joins between the tables will necessitate being performed at the individual-datetime level.

We have identified a total of six relevant daily data indicators. We will assess which ones require joining.

- Activity
- Calories
- Intensities
- Steps
- Sleep
- Weight

2.1.1.1 Unique ID's in each table Let's first check the number of Unique ID's in each of the daily tables. This will help determine any tables that do not contain information for all tables

The following SQL query checks the number of unique ID's in all 6 tables.

```
SELECT                              -- Activity Table
    'Activity' AS DataIndicator, -- Data indicators representing each table
    COUNT(DISTINCT Id) AS UniqueIds  -- Count of Unique IDs
FROM
    dailyactivity_merged
UNION             -- UNION to stack rows for all tables
    SELECT          -- Process Repeated for Weight
        'Weight' AS DataIndicator,
        COUNT(DISTINCT Id) AS UniqueIds
    FROM
        weightloginfo_merged
UNION
    SELECT           -- Process Repeated for Sleep
        'Sleep' AS DataIndicator,
        COUNT(DISTINCT Id) AS UniqueIds
    FROM
        sleepday_merged
UNION
    SELECT      -- Process Repeated for Steps
        'Steps' AS DataIndicator,
        COUNT(DISTINCT Id) AS UniqueIds
    FROM
        dailysteps_merged
UNION
    SELECT        -- Process Repeated for Intensities
        'Intensities' AS DataIndicator,
```

```
        COUNT(DISTINCT Id) AS UniqueIds
    FROM
        dailyintensities_merged
UNION
    SELECT              -- Process Repeated for Calories
        'Calories' AS DataIndicator,
        COUNT(DISTINCT Id) AS UniqueIds
    FROM
        dailycalories_merged
ORDER BY        -- Sort by Unique ID's
  UniqueIds
```

Table 1: 6 records

DataIndicator	UniqueIds
Weight	8
Sleep	24
Activity	33
Calories	33
Intensities	33
Steps	33

The results above indicate a maximum sample size of 33 respondents.

The weight data set will need to be dropped from this analysis due to insufficient data. Only 8 respondents provided weight information, making the sample size too small for meaningful insights

The sleep data, while limited to only 24 respondents, will be retained. However, it's important to be aware that merging this data with other tables will likely result in missing values for the columns originating from the sleep table.

After dropping weight, we now have a total of 5 data indicators:

- Activity
- Calories
- Intensities
- Steps
- Sleep

2.1.1.2 Checking for Shared Column Names via a Pivot Table Next, we will determine which of the remaining tables require joining. Although each indicator has its own dedicated data file, we need to check if any of these already contain combined data from others.

Our first step is to review the column names in each table to identify shared names.

The following code will generate a pivot table with the following details:

- Rows: Combined a list of unique column names
- Columns: The 5 data indicators (Activity, Calories, Intensities, Steps and Sleep) columns.
- Values: 1 if a column is present in the tables as represented by the Data Indicator. 0 otherwise.

```
-- The Outer Query creates a pivot table from the pivot_data created by the subquery

-- Unique values from the Field column represent the rows of the pivot table
-- This is achieved through group aggregation using the GROUP BY clause in
-- conjunction with the SUM function
```

```sql
-- GROUP By groups all rows for a given field together
-- The SUM() function for the CASE statement collapses all rows for a field in one

-- Columns for the Pivot Table and the Values in them are created using CASE
-- For each Field/Column Name, CASE checks if it is present in the table
-- represented by the Data Indicator
-- If a given field is present, a Value of 1 is assigned to the relevant Column
-- If a given field is not present, a value of 0 is assigned instead


SELECT
      Field,
    SUM(CASE WHEN DataIndicator = 'Activity' THEN 1 ELSE 0 END) AS Activity,
    SUM(CASE WHEN DataIndicator = 'Intensities' THEN 1 ELSE 0 END) AS Intensities,
    SUM(CASE WHEN DataIndicator = 'Steps' THEN 1 ELSE 0 END) AS Steps,
    SUM(CASE WHEN DataIndicator = 'Calories' THEN 1 ELSE 0 END) AS Calories,
      SUM(CASE WHEN DataIndicator = 'Sleep' THEN 1 ELSE 0 END) AS Sleep,
    Count(*) as Total -- Count total tables the field column is in
FROM
-- The Inner query below stores column names from each of the 6 tables
-- in a column named Field

-- Column names are retrieved using PRAGMA table_info(table-name)
-- This is a specialized command in SQLite used to retrieve metadata about the
-- columns within a specific table

-- A DataIndicator value was also assigned to each table to represent it.
-- Results from the six tables rows are combined into a single output using the
-- UNION operation and are aliased as pivot_data
    (SELECT
       name AS Field, 'Sleep' AS DataIndicator
    FROM
      PRAGMA_TABLE_INFO('sleepday_merged')
    UNION
      SELECT
        name AS Field, 'Activity' AS DataIndicator
      FROM
        PRAGMA_TABLE_INFO('dailyactivity_merged')
    UNION
      SELECT
          name AS Field, 'Steps' AS DataIndicator
        FROM
        PRAGMA_TABLE_INFO('dailysteps_merged')
    UNION
      SELECT
          name AS Field, 'Intensities' AS DataIndicator
      FROM
          PRAGMA_TABLE_INFO('dailyintensities_merged')
    UNION
      SELECT
        name AS Field, 'Calories' AS DataIndicator
      FROM
          PRAGMA_TABLE_INFO('dailycalories_merged')) as pivot_data
```

```
Group By
    Field -- Group By Field and then use SUM() in the select statement for unique rows
Order by
    Activity, Total DESC, Field
    -- Order by the Activity column, then by Total
    -- Count (in descending order) and then by Field for easier visualization.
```

The results derived from the SQL query will be color-coded and formatted to facilitate visualization.

- All rows where 'Activity' is equal to 0, signifying rows for all columns not present in the 'Activity' table, are formatted in bold. Given that the SQL table was ordered by 'Activity,' these rows will appear at the top of the pivot table.
- The row corresponding to the 'ID' field is highlighted in **green**, as this column will serve as the key for joining tables.
- The rows for 'ActivityDay,' 'ActivityDate,' and 'Sleepday' are highlighted in **salmon red**.
- The rows for 'TotalSteps' and 'StepTotal' are highlighted in *yellow**.

```
pivot_table %>%
  kable(caption  = "Presence of a Field in each Table. 1 if Yes, 0 if not",
        booktabs=TRUE) %>% kable_styling("striped")  %>% #For striped rows
  row_spec(which(pivot_table$Activity == 0), bold=TRUE) %>%
  row_spec(which(pivot_table$Field == 'Id'), background ='#C1FFC1') %>%
  row_spec(which(pivot_table$Field == 'ActivityDay' |
                   pivot_table$Field == 'ActivityDate' |
                   pivot_table$Field == 'SleepDay'),background ='#FA8072') %>%
  row_spec(which(pivot_table$Field == 'TotalSteps'|
              pivot_table$Field == 'StepTotal'), background ='#FAFAD2')
```

Table 2: Presence of a Field in each Table. 1 if Yes, 0 if not

Field	Activity	Intensities	Steps	Calories	Sleep	Total
ActivityDay	**0**	**1**	**1**	**1**	**0**	**3**
SleepDay	**0**	**0**	**0**	**0**	**1**	**1**
StepTotal	**0**	**0**	**1**	**0**	**0**	**1**
TotalMinutesAsleep	**0**	**0**	**0**	**0**	**1**	**1**
TotalSleepRecords	**0**	**0**	**0**	**0**	**1**	**1**
TotalTimeInBed	**0**	**0**	**0**	**0**	**1**	**1**
Id	1	1	1	1	1	5
Calories	1	0	0	1	0	2
FairlyActiveMinutes	1	1	0	0	0	2
LightActiveDistance	1	1	0	0	0	2
LightlyActiveMinutes	1	1	0	0	0	2
ModeratelyActiveDistance	1	1	0	0	0	2
SedentaryActiveDistance	1	1	0	0	0	2
SedentaryMinutes	1	1	0	0	0	2
VeryActiveDistance	1	1	0	0	0	2
VeryActiveMinutes	1	1	0	0	0	2
ActivityDate	1	0	0	0	0	1
LoggedActivitiesDistance	1	0	0	0	0	1
TotalDistance	1	0	0	0	0	1
TotalSteps	1	0	0	0	0	1
TrackerDistance	1	0	0	0	0	1

Insights

The following insights were derived from the Pivot Table analysis:

- Most columns are present in the Activity Data set. Six out of a total of 21 rows exhibit a value of 0 for Activity (the top six rows are highlighted in bold).
- The 'Id' column is common across all five tables, as indicated in the green row.
- A single, consolidated Date Column is absent. Three potential date fields were identified due to their similar naming conventions: 'ActivityDate', 'ActivityDay', and 'SleepDay'. These rows are highlighted in red.
- 'ActivityDay' and 'SleepDay' are included in the six rows highlighted in bold. 'ActivityDay' is found in the Steps, Calories, and Intensities tables, while 'SleepDay' is exclusive to the Sleep table. Further investigation is necessary to ascertain if these are equivalent to 'ActivityDate' in the Activity table.
- 'StepTotal' is another column name not present in the Activity table but found in the Steps table. Scrolling through the pivot table reveals that the Activity table contains 'TotalSteps'. Additional analysis is required to confirm the equivalence of these two columns. See rows highlighted in Yellow
- The remaining three columns not present in the Activity table, 'TotalMinutesAsleep', 'TotalSleepRecords', and 'TotalTimeinBed', lack similarly named alternatives in the Activity data set and are therefore unique columns from the Sleep table.

2.1.1.3 Resolving Inconsistencies in Column Names and Values Insights form the pivot table show that when compared with the Activity table:

- Steps contains two additional columns; ActivityDay and StepTotal
- Calories contains one additional columns; ActivityDay
- Intensities contains one additional columns; ActivityDay
- Sleep contains 4 additional columns; SleepDay, TotalMinutesAsleep', 'TotalSleepRecords', and 'TotalTimeinBed. *Any analysis for the last 3 columns with the data from any other table warrants a join*

The SQL query below will check:

- The number of records when TotalSteps is not the same as StepTotal
- The number of records when ActivityDate is not the same as ActivityDay in each of Calories, Steps and Intensities tables
- The number of records when ActivityDate is not the same as SleepDay

```
WITH step_diff as (   -- Create a temporary table when TotalSteps != StepTotal
SELECT
    Count(*) as StepsDiff, -- Count for total Rows
    1 as UniId       -- 1 set as a uniqueid which will be used for the Join
FROM
    (SELECT
      TotalSteps
     FROM
      dailyactivity_merged
    EXCEPT           -- EXCEPT operator is a set operation that returns distinct
    -- rows from the result set of the first SELECT query that are not present
    -- in the result set of the second
      SELECT
        StepTotal
    FROM
       dailysteps_merged) as tab1
),        -- Repeat for ActivityDate != ActivityDay in Steps
date_diff_step as (
SELECT
    Count(*) as DatesDiffSteps,
```

```
    1 as UniId
FROM
    (SELECT
       ActivityDate
    FROM
         dailyactivity_merged
    EXCEPT
    SELECT
        ActivityDay
    FROM
          dailysteps_merged) as tab2
),   -- Repeat for ActivityDate != ActivityDay in Calories
date_diff_cal as (
SELECT
    Count(*) as DatesDiffCal,
    1 as UniId
FROM
    (SELECT
        ActivityDate
    FROM
        dailyactivity_merged
    EXCEPT
    SELECT
      ActivityDay
  FROM
        dailycalories_merged) as tab3
),  -- -- Repeat for ActivityDate != ActivityDay in Intensities
date_diff_inten as (
SELECT
    Count(*) as DatesDiffInten,
    1 as UniId
FROM
    (SELECT
        ActivityDate
    FROM
        dailyactivity_merged
    EXCEPT
    SELECT
      ActivityDay
  FROM
        dailyintensities_merged) as tab4
), -- Repeat for ActivityDate != SleepDay
date_diff_sleep as (
SELECT
    Count(*) as DatesDiffSleep,
    1 as UniId
FROM
    (SELECT
        ActivityDate
    FROM
        dailyactivity_merged
    EXCEPT
    SELECT
```

```
      SleepDay
    FROM
        sleepday_merged) as tab5
)       -- Join rows from all 5 temporary tables in 1 using UniId as the key
SELECT
  A.StepsDiff,
  B.DatesDiffSteps,
  C.DatesDiffCal,
  D.DatesDiffInten,
  E.DatesDiffSleep
FROM
  step_diff as A
JOIN
  date_diff_step as B ON A.UniId=B.UniId
JOIN
 date_diff_cal as C ON A.UniId=C.UniId
JOIN
 date_diff_inten as D ON A.UniId=D.UniId
JOIN
 date_diff_sleep as E ON A.UniId=E.UniId
```

Table 3: 1 records

StepsDiff	DatesDiffSteps	DatesDiffCal	DatesDiffInten	DatesDiffSleep
0	0	0	0	31

- 0 records are returned when TotalSteps in the Activity table is *not* the same as StepTotal in the Steps table.
- 0 records are returned when ActivityDate in the Activity table is *not* the seams as ActivityDay in the Steps table.
- 0 records are returned when ActivityDate in the Activity table is *not* the same as ActivityDay in the Calories table.
- 0 records are returned when ActivityDate in the Activity table is *not* the same as ActivityDay in the Intensities table
- 31 records are returned when ActivityDate is *not* the same as SleepDay. This includes all 31 days for the whole time-period. This will need to be explored and cleaned further in the next section.

Subsequently, the Steps, Calories, and Intensities tables were found to be redundant, as their data is already captured within the Daily Activity table. Therefore, only the **Activity** and **Sleep** tables will be merged for this analysis.

2.1.2 Shortlisting Columns to Include in the Join

Having identified that the Activity and Sleep tables require a join, the next steps are to determine the specific columns for the join and to identify any necessary data cleaning or transformations.

2.1.2.1 Cleaning Dates The inconsistency between the ActivityDate and SleepDay values, as previously noted, will be addressed in this section to ensure they are uniform.

Compare SleepDay with ActivityDate

First, let's retrieve the values from both tables for comparison. .

```
With sleep as ( -- Create a temporary table with the first row of the Sleep table
SELECT
    SleepDay as Dates,    -- Rename as Dates
    'Sleep' as Tablename  -- Tablename to distinguish table
FROM
    sleepday_merged
LIMIT
  1     -- 1st row only
),
activity as (    -- Repeat for Activity with a new temporary table
SELECT
     ActivityDate as Dates,
     'Activity' as Tablename
FROM
     dailyactivity_merged
LIMIT
  1
)
SELECT
  *
FROM
  sleep
UNION      -- Union results from both tables
SELECT
    *
FROM
  activity
```

Table 4: 2 records

Dates	Tablename
4/12/2016	Activity
4/12/2016 12:00:00 AM	Sleep

The results show that while the SleepDay column also represents dates, its values are in a completely different format; they contain time stamps as well.

The next step involves creating new columns, labeled RecordedDate, in both tables. These columns will store the date values after they have been cleaned and transformed into a consistent format.

Activity Table

Currently values in ActivityDate are either in the M/DD/YYYY or M/D/YYYY format. These need to be standardized to the YYYY-MM-DD format and stored in a column named RecordedDate

The code below creates a new column for RecordedDate

```
ALTER TABLE dailyactivity_merged -- Create a new column named as RecordedDate
ADD COLUMN RecordedDate TEXT
```

The code below updates the RecordedDate column standardizing corresponding values in the ActivityDate table to the YYYY-MM-DD format

```
UPDATE dailyactivity_merged

-- Update RecordedDate with Cleaned Values from the ActivityDate field
-- Last 4 digits as YYYY
-- Month as MM (All months are currently either 4 or 5, these will be formatted
-- to 04 or 05)
-- Day as DD (Dates when Days are represented in single digits eg 1 will need to
-- be treated separately from Days in double digits eg 10)

SET RecordedDate = (
    SUBSTR(ActivityDate, -4) || '-0' ||  -- Last 4 digits for YYYY
    SUBSTR(ActivityDate, 1,1) || --  0 and First digit for MM
    CASE -- 0 and Third, or, Third and Fourth Digits for DD
        WHEN SUBSTR(ActivityDate, 4, 1) = '/'
        THEN '-0' || SUBSTR(ActivityDate, 3, 1)
        ELSE '-' || SUBSTR(ActivityDate, 3, 2)
    END
)
```

The code beow checks that RecordedDate stores the correct values from ActivityDate regardless of the format (M/DD/YYYY or M/D/YYYY)

```
With doubleday as ( -- Create a temporary table with rows where the ActivityDate
SELECT               --format is  M/DD/YYYY
   ActivityDate, RecordedDate
FROM
    dailyactivity_merged
WHERE
  LENGTH(ActivityDate) = 9
LIMIT
  1     -- 1st row only
),
singleday as (    -- Create a temporary table with rows where the ActivityDate
SELECT              --format is  M/D/YYYY
   ActivityDate, RecordedDate
FROM
    dailyactivity_merged
WHERE
  LENGTH(ActivityDate) = 8
LIMIT
  1     -- 1st row only
)
SELECT
  *
FROM
  doubleday
UNION      -- Union results from both tables
SELECT
```

```
    *
FROM
  singleday
```

Table 5: 2 records

ActivityDate	RecordedDate
4/12/2016	2016-04-12
5/1/2016	2016-05-01

Sleep Table

Currently values in SleepDay are either in the M/DD/YYYY hh:mm:s or M/D/YYYY hh:mm:s format. These need to be standardized to the YYYY-MM-DD format and stored in a column named RecordedDate

The code below creates a new column for RecordedDate

```
ALTER TABLE sleepday_merged -- Create a new column named as RecordedDate
ADD COLUMN RecordedDate TEXT
```

The code below updates the RecordedDate column standardizing corresponding values in the SleepDay table to the YYYY-MM-DD format

```
-- Update RecordedDate with Cleaned Values from the SleepyDaty field

-- When Day are in Single Digits:
-- Digits 5-9 as YYYY
-- 0 and First digit for MM
-- 0 and Third Digit, or , Third and Fourth Digit DDD
-- Day as DD (Dates when Days are represented in single digits eg 1 will need to
-- be treated separately from Days in double digits eg 10)


UPDATE sleepday_merged
SET RecordedDate = (
    CASE
        WHEN SUBSTR(SleepDay, 4, 1) = '/'
        THEN SUBSTR(SleepDay, 5,4) || '-0' || SUBSTR(SleepDay, 1,1) ||
        '-0' || SUBSTR(SleepDay, 3, 1)
        ELSE  SUBSTR(SleepDay, 6,4) || '-0' || SUBSTR(SleepDay, 1,1) ||
        '-' || SUBSTR(SleepDay, 3, 2)
    END
)
```

The code beow checks that RecordedDate stores the correct values from SleepDay regardless of the format (M/DD/YYYY or M/D/YYYY)

```
With doubleday as ( -- Create a temporary table with rows where the SleepDay
SELECT               --format is  M/DD/YYYY hh:mm:ss
   SleepDay, RecordedDate
FROM
    SleepDay_merged
WHERE
  LENGTH(SleepDay) = 21
LIMIT
```

```
  1     -- 1st row only
),
singleday as (    -- Create a temporary table with rows where the ActivityDate
SELECT             --format is  M/D/YYYY hh:mm:ss
   SleepDay, RecordedDate
FROM
    sleepday_merged
WHERE
  LENGTH(SleepDay) = 20
LIMIT
  1     -- 1st row only
)
SELECT
  *
FROM
  doubleday
UNION      -- Union results from both tables
SELECT
    *
FROM
  singleday
```

Table 6: 2 records

SleepDay	RecordedDate
4/12/2016 12:00:00 AM	2016-04-12
5/1/2016 12:00:00 AM	2016-05-01

The fields ActivityDate and SleepDay will now be excluded from the Join and any subsequent relevant code, as we have successfully created new date fields with cleaned data

2.1.2.2 Finalize and Validate Remaining Columns

Next, we will assess if any other columns in the two tables require dropping or transformation. To begin, let's obtain a brief description of the columns within both tables.

```
PRAGMA table_info(dailyactivity_merged)
```

Table 7: 16 records

cid	name	type	notnull	dflt_value	pk
0	Id	REAL	0	NA	0
1	ActivityDate	TEXT	0	NA	0
2	TotalSteps	INTEGER	0	NA	0
3	TotalDistance	REAL	0	NA	0
4	TrackerDistance	REAL	0	NA	0
5	LoggedActivitiesDistance	REAL	0	NA	0
6	VeryActiveDistance	REAL	0	NA	0
7	ModeratelyActiveDistance	REAL	0	NA	0
8	LightActiveDistance	REAL	0	NA	0
9	SedentaryActiveDistance	REAL	0	NA	0
10	VeryActiveMinutes	INTEGER	0	NA	0
11	FairlyActiveMinutes	INTEGER	0	NA	0

cid	name	type	notnull	dflt_value	pk
12	LightlyActiveMinutes	INTEGER	0	NA	0
13	SedentaryMinutes	INTEGER	0	NA	0
14	Calories	INTEGER	0	NA	0
15	RecordedDate	TEXT	0	NA	0

```
PRAGMA table_info(sleepday_merged)
```

Table 8: 6 records

cid	name	type	notnull	dflt_value	pk
0	Id	REAL	0	NA	0
1	SleepDay	TEXT	0	NA	0
2	TotalSleepRecords	INTEGER	0	NA	0
3	TotalMinutesAsleep	INTEGER	0	NA	0
4	TotalTimeInBed	INTEGER	0	NA	0
5	RecordedDate	TEXT	0	NA	0

Let's also review their descriptions. The table below lists descriptions for the relevant columns from the Fitabase data dictionary avaialble at this link:

https://www.fitabase.com/media/2126/fitabase-fitbit-data-dictionary-as-of-05162025.pdf

```
datadescription <- read_excel("datadescription.xlsx", sheet='Daily')
kable(datadescription, booktabs=TRUE) %>%  kable_styling("striped") %>%
  column_spec(2, width = "12cm")
```

Data Header	Data Description
ActivityDate	Date value in mm/dd/yyyy format.
TotalSteps	Total number of steps taken.
TotalDistance	Total kilometers tracked.
TrackerDistance	Total kilometers tracked by Fitbit device.
LoggedActivitiesDistance	Total kilometers from logged activities.
VeryActiveDistance	Kilometers travelled during very active activity.
ModeratelyActiveDistance	Kilometers travelled during moderate activity.
LightActiveDistance	Kilometers travelled during light activity.
SedentaryActiveDistance	Kilometers travelled during sedentary activity.
VeryActiveMinutes	Total minutes spent in very active activity.
FairlyActiveMinutes	Total minutes spent in moderate activity.
LightlyActiveMinutes	Total minutes spent in light activity.
SedentaryMinutes	Total minutes spent in sedentary activity.
Calories	Total estimated energy expenditure (in kilocalories).
SleepDay	Date on which the sleep event started. (in mm/dd/yyyy hh:mm:ss format)
TotalSleepRecords	Number of recorded sleep periods for that day. Includes naps > 60min
TotalMinutesAsleep	Total number of minutes classified as being "asleep".
TotalTimeInBed	Total minutes spent in bed, including asleep, restless, and awake, that occurred during a defined sleep record.

Finalize and Validate Columns from the Activity table

Let's examine the distance and minutes columns from the Activity table to finalize which ones to keep.

Note: Variables name are renamed below for easier analysis and visualization. This renaming will be repeatedly used in other codes as well including the Join

Validate Distance Columns

The code below list first 5 observarions for the distance columns

```
SELECT
   LoggedActivitiesDistance as LogActDist,
   TotalDistance as TotalDist,
   TrackerDistance as TrackDist,
   SedentaryActiveDistance as SedActDist,
   LightActiveDistance as LightActDist,
   VeryActiveDistance as VeryActDist,
   ModeratelyActiveDistance as ModActDist
FROM
  dailyactivity_merged
LIMIT
  5
```

Table 10: 5 records

LogActDist	TotalDist	TrackDist	SedActDist	LightActDist	VeryActDist	ModActDist
0	8.50	8.50	0	6.06	1.88	0.55
0	6.97	6.97	0	4.71	1.57	0.69
0	6.74	6.74	0	3.91	2.44	0.40
0	6.28	6.28	0	2.83	2.14	1.26
0	8.16	8.16	0	5.04	2.71	0.41

It appears that Tracker Distance and Total Distance are largely equivalent. To confirm this, we can calculate the difference between these two variables and then compute summary statistics to check if the difference is zero.

Additionally, LoggedActivitiesDistance and Sedentary Active Distance appear to be mostly zero. We can calculate summary statistics to verify this observation.

The following query calculates the average difference between Tracker Distance and Total Distance and determines the percentage of rows where the difference is zero. It also computes the average values for LoggedActivitiesDistance and Sedentary Active Distance and counts the percentage of rows where these values are zero

```
SELECT
  avg(DistDiff) as AvgDistDiff,  -- Mean of the difference
  COUNT(CASE WHEN DistDiff=0 THEN 1 END)*100/940 as PerDistDiffZero, -- % when 0
  avg(LogActDist) as AvgLogActDist, -- Mean of LoggedActivitiesDistance
  COUNT(CASE WHEN LogActDist=0 THEN 1 END)*100/940 as PerLogActDistZero, -- % when 0
  avg(SedActDist) as AvgSedActDist, -- Mean of SedentaryActiveDistance
  COUNT(CASE WHEN SedActDist=0 THEN 1 END)*100/940 as PedActDistZero -- % when 0
  FROM -- Inner query to select relevant columns
  (SELECT
    TrackerDistance,
    TotalDistance,
    LoggedActivitiesDistance as LogActDist,
    SedentaryActiveDistance as SedActDist,
```

```
    TrackerDistance - TotalDistance as DistDiff -- Difference the two
FROM
    dailyactivity_merged) as table1
```

Table 11: 1 records

AvgDistDiff	PerDistDiffZero	AvgLogActDist	PerLogActDistZero	AvgSedActDist	PedActDistZero
-0.0143511	98	0.1081709	96	0.0016064	91

The mean difference between 'Tracker distance' and 'total distance' is negligible, averaging approximately -0.01. The difference is precisely zero in 98.4% of instances.

Similarly, the mean of 'LoggedActivitiesDistance' is close to zero, registering as zero in 96.6% of cases. The Sedentary Active Distance is consistently zero (0) across all observations. This anomaly may be attributed to an import error; however, it has been confirmed that the actual values were so minuscule—only negligibly above zero—that the data import process rounded them down to zero.

Given the preceding analysis, Tracker Distance, LoggedActivitiesDistance, and SedentaryActiveDistance can be excluded from the scope of our analysis.

It is also apparent that the Total Distance is the sum of VeryActiveDistance, ModeratelyActiveDistance, and LightActiveDistance (disregarding Sedentary Active Distance, as its value is uniformly zero). We can confirm this by summing the values for VeryActiveDistance, ModeratelyActiveDistance, and LightActiveDistance and subsequently calculating the difference of the sum from the Total Distance.

The subsequent query calculates the difference between the sum, of VeryActiveDistance, ModeratelyActiveDistance, and LightActiveDistance, and TotalDistance. It then computes the minimum, maximum, and average values for this difference.

```
SELECT
  'TotalDistanceDifference' as Field,
  min((TotalDistance_Diff)) as Minimum,
  avg(TotalDistance_Diff) as Average,
  max((TotalDistance_Diff)) as Maximum
FROM
(SELECT
    VeryActiveDistance + ModeratelyActiveDistance +
    LightActiveDistance - TotalDistance as TotalDistance_Diff
FROM
    dailyactivity_merged) as table1
```

Table 12: 1 records

Field	Minimum	Average	Maximum
TotalDistanceDifference	-9.37	-0.0786596	0.0100006

The "Total distance" metric is not precisely equivalent to the sum of "VeryActiveDistance," "ModeratelyActiveDistance," and "LightActiveDistance"; however, it is highly comparable. **Consequently, the "TotalDistance" column will be excluded, and a new, comprehensive total will be calculated by aggregating the three active distance categories.**

Validate Minutes Columns

The code below list first 5 observarions for the minutes columns

```sql
SELECT
   VeryActiveMinutes as VeryActMin,
   FairlyActiveMinutes as FairlyActMin,
   LightlyActiveMinutes as LightlyActMin,
   SedentaryMinutes as SedMin
FROM
  dailyactivity_merged
LIMIT
  5
```

Table 13: 5 records

VeryActMin	FairlyActMin	LightlyActMin	SedMin
25	13	328	728
21	19	217	776
30	11	181	1218
29	34	209	726
36	10	221	773

It would be beneficial to introduce columns for "Total Active Minutes" The creation of an aggregate value for Total Active Minutes will be advantageous for analyzing activity as a whole. We will keep this in mind while constructing the SQL query for the join

Finalize and Validate Columns from the Sleep table

Now let's explore the first few rows from the Sleep table.

The code below list first 5 observarions for the minutes columns

```sql
SELECT
  *
FROM
  sleepday_merged
LIMIT 5
```

Table 14: 5 records

Id	SleepDay	TotalSleepRecords	TotalMinutesAsleep	TotalTimeInBed	RecordedDate
1503960366	4/12/2016 12:00:00 AM	1	327	346	2016-04-12
1503960366	4/13/2016 12:00:00 AM	2	384	407	2016-04-13
1503960366	4/15/2016 12:00:00 AM	1	412	442	2016-04-15
1503960366	4/16/2016 12:00:00 AM	2	340	367	2016-04-16
1503960366	4/17/2016 12:00:00 AM	1	700	712	2016-04-17

The Sleep table just contains 5 columns. As discussed earlier, the SleepDay variable would need to be cleaned to make it consistent with the ActivityDate column in the Activity table.

Lets explore the TotalSleepRecords column with some summary statistics.

```sql
SELECT
  'TotalSleepRecords' as Field,
  min(TotalSleepRecords) as Minimum,
  avg(TotalSleepRecords) as Average,
  max(TotalSleepRecords) as Maximum,
  COUNT(CASE WHEN TotalSleepRecords = 1 THEN 1 END)*100/413 as PerEqualOne
FROM
(SELECT
  TotalSleepRecords
FROM
  sleepday_merged) as table1
```

Table 15: 1 records

Field	Minimum	Average	Maximum	PerEqualOne
TotalSleepRecords	1	1.118644	3	88

The **TotalSleepRecords** variable ranges from 1 to 3, with a value of 1 occurring approximately 89% of the time. As defined in the Data Description this is the *"Number of recorded sleep periods for that day. Includes naps > 60min"*. The data suggests that most people only sleep once in a day (presumably at night)

As this variable does not appear to provide meaningful variation in data, it will be excluded from the analysis.

Next, we will verify whether "minutes asleep" is consistently less than "time in bed."

```sql
SELECT
  count(*) as AsleepMoreThanInBed
FROM
  sleepday_merged
WHERE
  TotalMinutesAsleep > TotalTimeInBed
```

Table 16: 1 records

AsleepMoreThanInBed
0

Yes, we can confirm that time spent asleep is always less than the time spent in bed

2.1.2.3 Checking for Duplicates and Missing Values

Next we will check for any duplicates and missing values that exist for our finalized column lists for both activities and sleep,

```sql
WITH dup_act as (     -- Create a temporary table to calculate Duplicates
SELECT
  1 as UniId,     -- Define UniId in each temporary table to facilitate the join
  COUNT(*) -      -- Calculate the difference between the count of total rows
(SELECT           -- and distinct rows. This difference will be the # of duplicates
  count(*)
FROM
(SELECT DISTINCT      -- This inner query counts distinct rows
  RecordedDate,
  Id,
```

```
    Calories,
    TotalSteps,
    LightlyActiveMinutes,
    FairlyActiveMinutes,
    VeryActiveMinutes,
    SedentaryMinutes,
    LightActiveDistance,
    ModeratelyActiveDistance,
    VeryActiveDistance
FROM
  dailyactivity_merged) as t1) as DuplicatesActivity
FROM
  dailyactivity_merged
),
miss_act as ( -- Create a temporary table to calculate Missing Values
SELECT
    COUNT(*) AS MissingValuesActivity,  -- Count values when any one of the columns
    1 as UniId                          -- specified is null. See where
FROM
    dailyactivity_merged
WHERE
    RecordedDate IS NULL OR
  Id IS NULL OR
    Calories IS NULL OR
    TotalSteps IS NULL OR
    LightlyActiveMinutes IS NULL OR
    FairlyActiveMinutes IS NULL OR
    VeryActiveMinutes IS NULL OR
    SedentaryMinutes IS NULL OR
    LightActiveDistance IS NULL OR
    ModeratelyActiveDistance IS NULL OR
    VeryActiveDistance IS NULL
),
dup_sleep as (     -- Repeat the process for the sleep table
SELECT
  1 as UniId,
  COUNT(*) -
(SELECT
  count(*)
FROM
(SELECT DISTINCT
  Id,
  RecordedDate,
  TotalMinutesAsleep,
  TotalTimeInBed
FROM
  sleepday_merged) as t2) as DuplicatesSleep
FROM
  sleepday_merged
),
miss_sleep as (
SELECT
    COUNT(*) AS MissingValuesSleep,
```

```
    1 as UniId
FROM
  sleepday_merged
WHERE
    Id IS NULL OR
  RecordedDate IS NULL OR
  TotalMinutesAsleep IS NULL OR
    TotalTimeInBed IS NULL
)
SELECT                -- Join results from all tables into one
  A.DuplicatesActivity,
  B.MissingValuesActivity,
  C.DuplicatesSleep,
  D.MissingValuesSleep
FROM
  dup_act as A
JOIN
  miss_act as B on A.UniId=B.UniId
JOIN
  dup_sleep as C on A.UniId=C.UniId
JOIN
  miss_sleep as D on A.UniId=D.UniId
```

Table 17: 1 records

DuplicatesActivity	MissingValuesActivity	DuplicatesSleep	MissingValuesSleep
0	0	3	0

Both tables are clean, with zero missing values. While the activity table also has no duplicate rows, the Sleep Data set contains three duplicate rows. These three duplicates will be excluded when the SQL query for the join is constructed.

2.1.3 Joining the Daily tables

Now that the columns are finalized, the Activity and Sleep tables can be joined. This joining process will exclude the previously identified columns and incorporate any newly discussed ones.

- A column for Total Active Minutes (LightlyActiveMinutes + FairlyActiveMinutes + VeryActiveMinutes) will be created.
- A column for Total Distance (LightActiveDistance + ModeratelyActiveDistance + VeryActiveDistance) will be created.
- A column for DayType is created (Weekend or Weekday)
- The cleaned date columns for both datasets are used
- A left join will be utilized to combine the two tables.
- Unique rows will be selected from the Right i.e. Sleep table in order to account for the duplicate rows in this table.

As disucssed earlier, variables name are renamed for easier analysis and visualization.

The subsequent SQL query employs a left join (with the Activity as the Left and Sleep as the right tables) using the Id and RecordedDate columns. It also generates the new columns mentioned above and cleans the date columns. This code chunk will produce a data frame named **daily_activity_sleep**, containing the results of the query.

```
-- This SQL Query will output a data frame named as named as daily_activity_sleep
-- with the results of the query.

SELECT
    A.Id,
    A.RecordedDate,       -- Cleaned Date variable
    CASE CAST(STRFTIME('%w', A.RecordedDate) AS INTEGER)
      WHEN 0 THEN 'Weekend'
      WHEN 6 THEN 'Weekend'
      ELSE 'Weekday' -- DayType; Weekday or Weekend
    END AS DayType,
    A.Calories,
  A.TotalSteps as Steps,
    A.LightlyActiveMinutes as LightlyActMin,
    A.FairlyActiveMinutes as FairlyActMin,
    A.VeryActiveMinutes VeryActMin,
    A.LightlyActiveMinutes + A.FairlyActiveMinutes +
    A.VeryActiveMinutes as TotalActMin,  -- Add all active minutes
    A.SedentaryMinutes as SedMin,
    A.LightActiveDistance as LightActDist,
    A.ModeratelyActiveDistance as ModActDist,
    A.VeryActiveDistance as VeryActDist,
    A.LightActiveDistance + A.ModeratelyActiveDistance +
    A.VeryActiveDistance as TotalDist, -- Add all distance
    B.TotalMinutesAsleep as AsleepMin,
    B.TotalTimeInBed as TimeInBed
FROM
    dailyactivity_merged as A
LEFT JOIN
    (SELECT DISTINCT        -- Select Distinct rows to drop duplicates
        Id,
        TotalMinutesAsleep,
        TotalTimeInBed,
        RecordedDate -- Cleaned Date variable
```

```
    FROM
        sleepday_merged) as B
ON
    A.Id = B.Id  -- Join on ID and Date
    AND
    A.RecordedDate = B.RecordedDate
```

2.1.4 Review Joined Data

Lets now the first few rows of the joined dataframe for some quick validation. We will split the view for easier visualization

```
kable(head(daily_activity_sleep,5)[,1:5]) %>%
  kable_styling("striped")
```

Id	RecordedDate	DayType	Calories	Steps
1503960366	2016-04-12	Weekday	1985	13162
1503960366	2016-04-13	Weekday	1797	10735
1503960366	2016-04-14	Weekday	1776	10460
1503960366	2016-04-15	Weekday	1745	9762
1503960366	2016-04-16	Weekend	1863	12669

```
kable(head(daily_activity_sleep,5)[,6:10]) %>%
  kable_styling("striped")
```

LightlyActMin	FairlyActMin	VeryActMin	TotalActMin	SedMin
328	13	25	366	728
217	19	21	257	776
181	11	30	222	1218
209	34	29	272	726
221	10	36	267	773

```
kable(head(daily_activity_sleep,5)[,11:16]) %>%
  kable_styling("striped")
```

LightActDist	ModActDist	VeryActDist	TotalDist	AsleepMin	TimeInBed
6.06	0.55	1.88	8.49	327	346
4.71	0.69	1.57	6.97	384	407
3.91	0.40	2.44	6.75	NA	NA
2.83	1.26	2.14	6.23	412	442
5.04	0.41	2.71	8.16	340	367

The data has been successfully and correctly joined. We will now proceed to the Data Visualization phase.

2.2 Data Visualizations and Plots

Now that we have our data cleaned and consolidated, we can move forward with the Data Visualization phase. This Phase will first focus on understanding the individual distributions and composition of variables and then proceed to identifying any key relationships between them

2.2.1 Distributions and Compositions

This section will concentrate on generating summary statistics, distribution charts, and time trends to facilitate an understanding of distributions and compositions.

2.2.1.1 Summary Statistics The initial step involves generating summary statistics to facilitate a more comprehensive understanding of the dataset. We will split the view easier visualization

```
kable(daily_activity_sleep[,1:5] %>% summary(digits=2)) %>%
  kable_styling("striped")
```

Id	RecordedDate	DayType	Calories	Steps
Min. :1.5e+09	Length:940	Length:940	Min. : 0	Min. : 0
1st Qu.:2.3e+09	Class :character	Class :character	1st Qu.:1828	1st Qu.: 3790
Median :4.4e+09	Mode :character	Mode :character	Median :2134	Median : 7406
Mean :4.9e+09	NA	NA	Mean :2304	Mean : 7638
3rd Qu.:7.0e+09	NA	NA	3rd Qu.:2793	3rd Qu.:10727
Max. :8.9e+09	NA	NA	Max. :4900	Max. :36019

```
kable(daily_activity_sleep[,6:10] %>% summary(digits=2)) %>%
  kable_styling("striped")
```

LightlyActMin	FairlyActMin	VeryActMin	TotalActMin	SedMin
Min. : 0	Min. : 0	Min. : 0	Min. : 0	Min. : 0
1st Qu.:127	1st Qu.: 0	1st Qu.: 0	1st Qu.:147	1st Qu.: 730
Median :199	Median : 6	Median : 4	Median :247	Median :1058
Mean :193	Mean : 14	Mean : 21	Mean :228	Mean : 991
3rd Qu.:264	3rd Qu.: 19	3rd Qu.: 32	3rd Qu.:317	3rd Qu.:1230
Max. :518	Max. :143	Max. :210	Max. :552	Max. :1440

```
kable(daily_activity_sleep[,11:16] %>% summary(digits=2)) %>%
  kable_styling("striped")
```

LightActDist	ModActDist	VeryActDist	TotalDist	AsleepMin	TimeInBed
Min. : 0.0	Min. :0.00	Min. : 0.00	Min. : 0.0	Min. : 58	Min. : 61
1st Qu.: 1.9	1st Qu.:0.00	1st Qu.: 0.00	1st Qu.: 2.5	1st Qu.:361	1st Qu.:404
Median : 3.4	Median :0.24	Median : 0.21	Median : 5.2	Median :432	Median :463
Mean : 3.3	Mean :0.57	Mean : 1.50	Mean : 5.4	Mean :419	Mean :458
3rd Qu.: 4.8	3rd Qu.:0.80	3rd Qu.: 2.05	3rd Qu.: 7.6	3rd Qu.:490	3rd Qu.:526
Max. :10.7	Max. :6.48	Max. :21.92	Max. :28.0	Max. :796	Max. :961
NA	NA	NA	NA	NA's :530	NA's :530

Some quick insights:

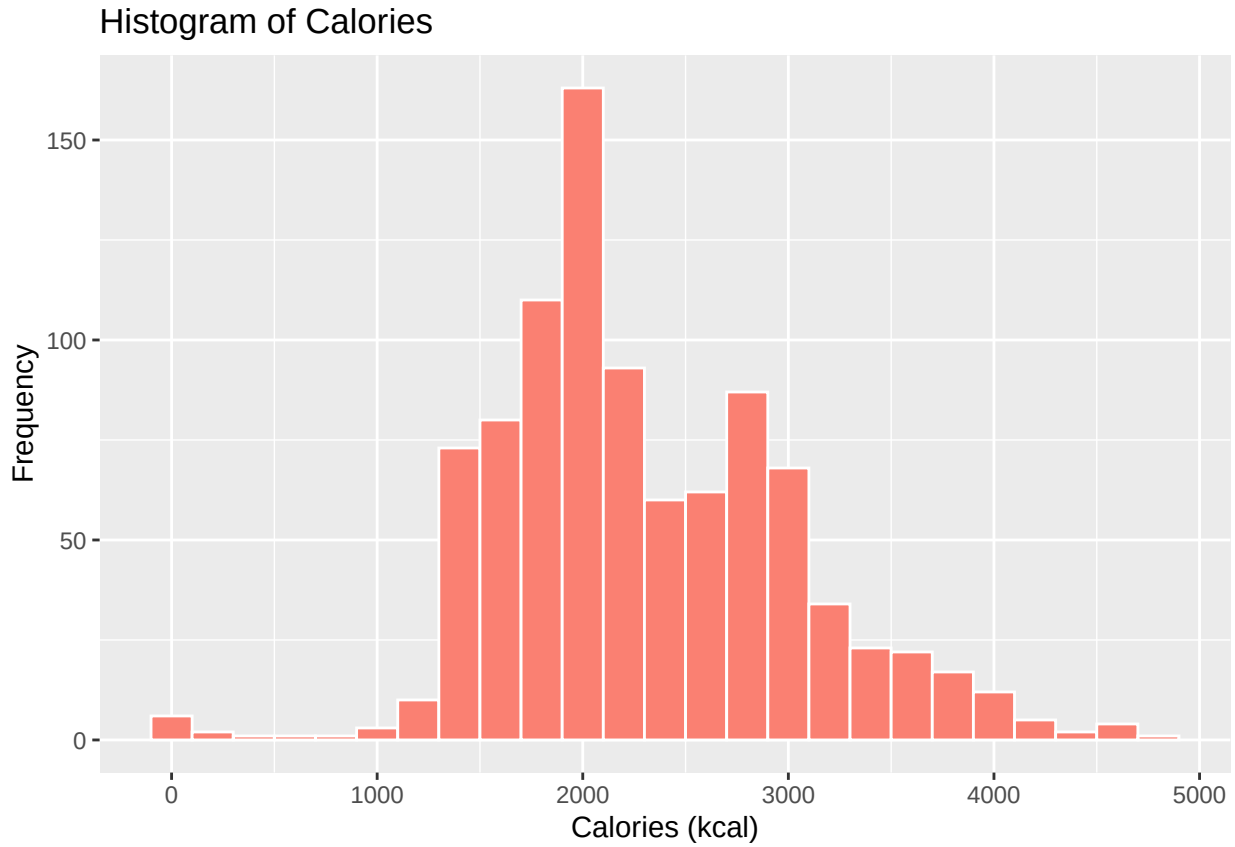
- Daily average for calories burned, 2,304, aligns with standard dietary guidelines.
- Interestingly, the average step count,7,600 is slightly higher than the typical 5,000–7,000 step range, suggesting an active user base. Source: https://www.health.harvard.edu/heart-health/just-7000-daily-steps-reduces-heart-disease-risk
- Lightly Active Minutes are significantly higher than both Fairly Active and Very Active Minutes. The mean for lightly active minutes, at approximately 193, is about 14 times greater than Fairly Active Minutes and 9 times greater than Very Active Minutes.
- Sedentary Minutes are considerably higher than the Total Active Minutes, with the mean for sedentary minutes being roughly four times that of total active minutes.
- The differences in the means of the Distance columns are not that glaring. However, the mean for Light Active Distance is 6 times that of Moderately Active Distance and around twice that of Very Active Distance.
- The means for time in bed are and minutes asleep are very close to each other as expected.
- The data set sourced from the Sleep table exhibits 530 missing values. This discrepancy is logical, given that our initial analysis identified that the sleep table only included data from 24 respondents, while the other tables contained data from 33 respondents. Consequently, columns derived from the sleep table will contain missing values. The 530 number also makes sense as it is the difference obtained after subtracting 410 (number of rows in the Sleep table after dropping the 3 duplicates) and 940 (number of rows present in the activities data set).

We will now proceed to delve deeper into these statistics using pie charts, histograms, and time-trend analysis.

2.2.1.2 Analyzing Calories First lets analyze calories beginning with a histogram to visualize the distribution.

Distribution of Total Calories

```
ggplot(daily_activity_sleep, aes(Calories)) +
 geom_histogram(fill = 'salmon', color = 'white', binwidth=200) +
  labs(title ="Histogram of Calories",
  x = 'Calories (kcal)',
  y = "Frequency")
```

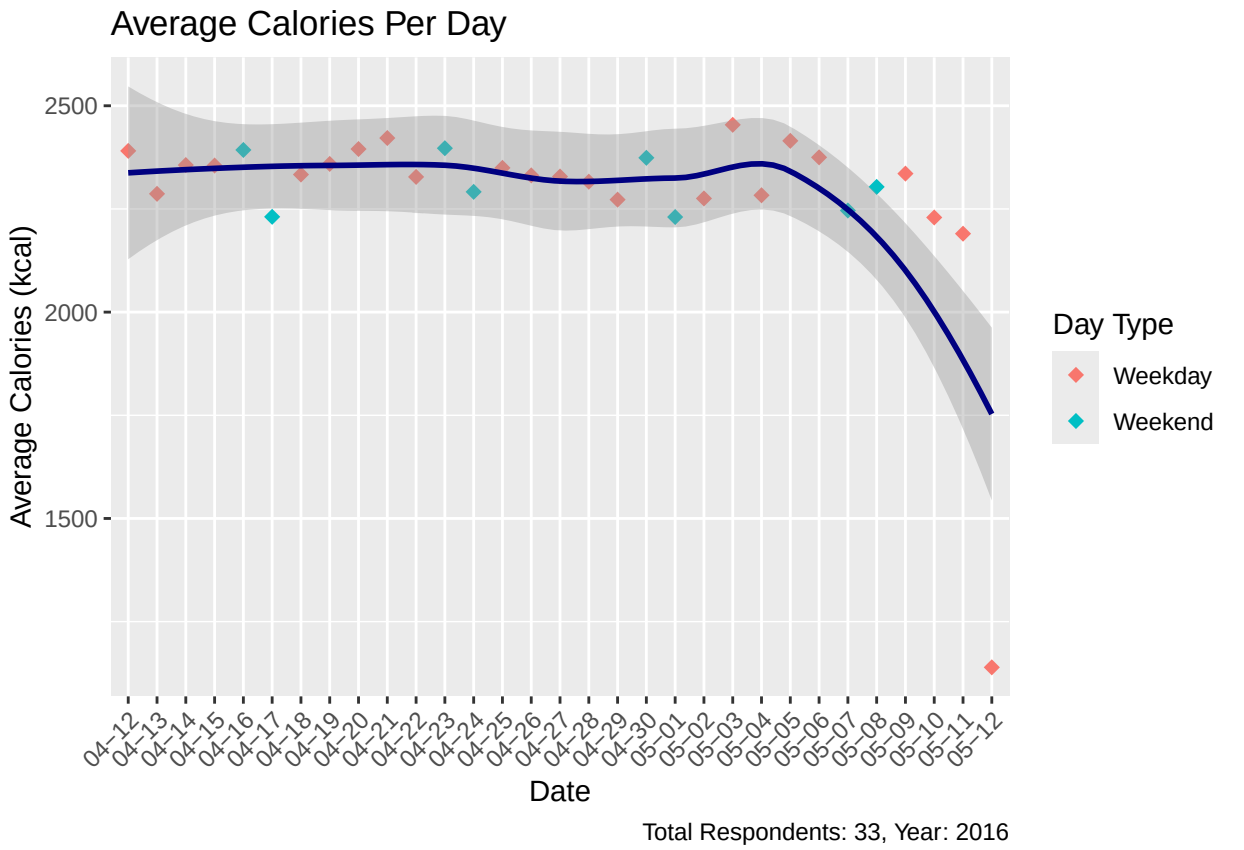


The data shows a peak calorie distribution around 2000 kcal, which is at the lower end of the 2,000–3,000+ Daily Calorie Intake Guidelines for Active Individuals maintaining their weight. This suggests that many participants are not highly active, indicating a potential target audience for Bellabeat.

Average Daily Calories Burned Trend

We will now examine the time-based trend of the average daily calories burned.

```
daily_activity_sleep %>% group_by(RecordedDate,DayType) %>%
  summarise(daily_avg= mean(Calories), .groups = 'drop') %>% # Calculate mean per day
  ggplot(aes(x=format(as.Date(RecordedDate), format = "%m-%d"), # Remove Year
             y=daily_avg, group = 1, color=DayType)) +
  geom_point(shape=18, size=2.5) +
  geom_smooth(color = "navyblue") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1)) +
  labs(
    x = "Date", # New x-axis title
    y = "Average Calories (kcal)", # New y-axis title
    title = "Average Calories Per Day", # New title
    color="Day Type", # New legend title
    caption = "Total Respondents: 33, Year: 2016"
  )
```



The average calories burned level remains relatively stable throughout April but shows a decline in May.There is also no observed variation based on day type (weekends compared to weekdays).

The drop in calories burned could be attributed to two main factors:

1. **Rising Summer Temperatures**: As summer progresses, people may be less inclined to participate in outdoor activities due to increased heat. Bellabeat could address this by promoting and recommending indoor workouts and activity options during the summer months.
2. **Possible Attrition of High Performers**: The decrease in average could also result from the attrition of users who were previously high performers in the final days of the period, thereby lowering the overall average.

We can check whether general attrition took place through the code below:

```
# Group by Recorded Date and count observations
# Arrange with lowest values of observations at the top
daily_activity_sleep %>%
  group_by(RecordedDate, DayType) %>%
  summarize(Observations = n(), .groups = 'drop') %>% arrange(Observations) %>%
  head(5) %>% kable(booktabs = TRUE) %>% kable_styling("striped")
```

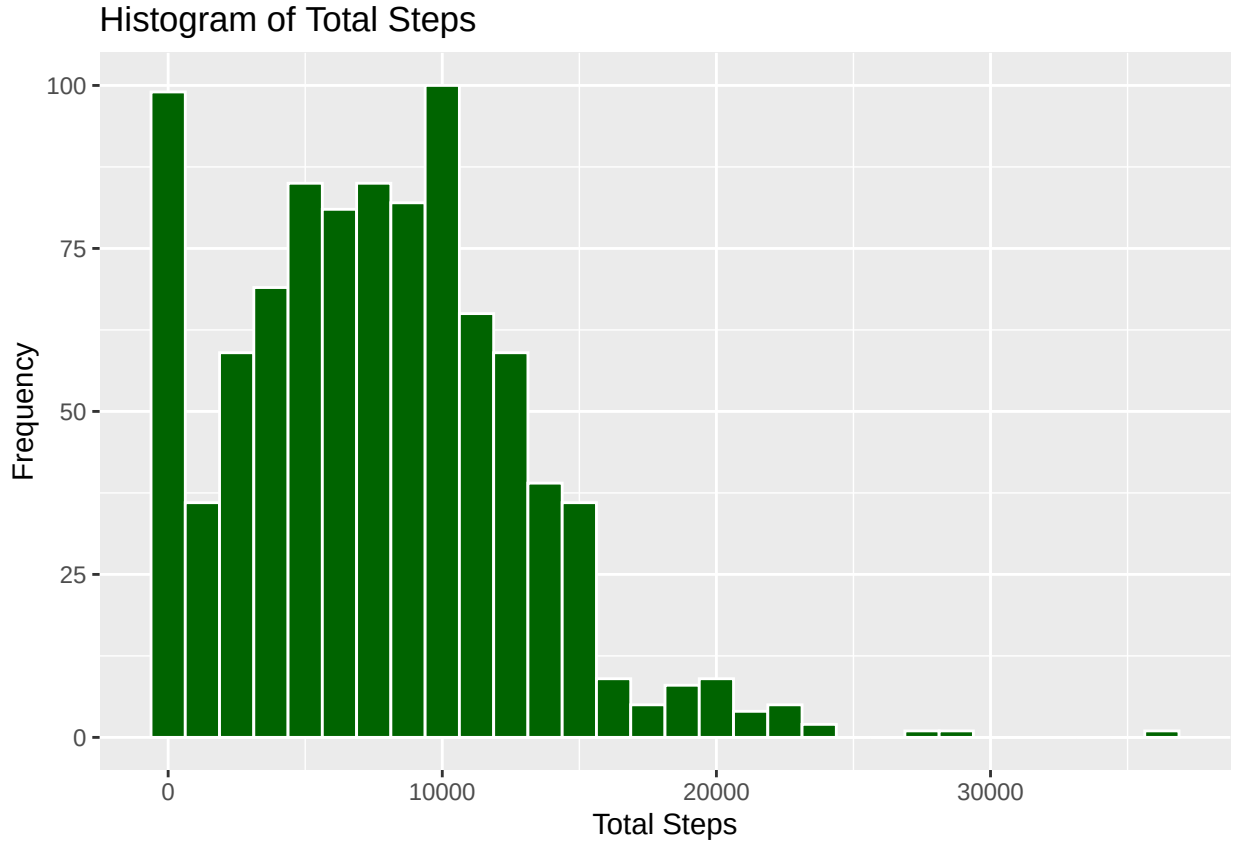
RecordedDate	DayType	Observations
2016-05-12	Weekday	21
2016-05-11	Weekday	24
2016-05-10	Weekday	26
2016-05-08	Weekend	27
2016-05-09	Weekday	27

The code analysis above does confirm the decline, as the lowest observations are from the last 5 days in May. However, it does not confirm that the missing data is exclusively from high performers, which would verify this as the reason for the lower overall average.

2.2.1.3 Analyzing Steps Lets next analyze Total Steps beginning with a histogram to visualize the distribution.

Distribution of Total Steps

```
ggplot(daily_activity_sleep, aes(Steps)) +
 geom_histogram(fill = 'darkgreen', color = 'white', binwidth=1250) +
  labs(title ="Histogram of Total Steps",
  x = 'Total Steps',
  y = "Frequency")
```



The Total Steps data exhibits a right-skewed (positively skewed) distribution, meaning most data points are concentrated on the left side of the graph. A few extremely high values (outliers) influence the mean, pulling it above the median, as evidenced by the summary statistics.

Additionally the distribution for Total Steps is also bimodal or has two peaks/bumps. The data suggests that the participant group includes individuals who are both significantly high and significantly low performers as:

1. Low Performers (Low Outliers): A significant observation is the peak of observations with daily steps less than 1250. In fact, a good portion of the distribution is concentrated on the left of 10,000. This suggests a substantial number of users are not meeting the recommended daily goal of 5,000–7,000 steps. Bellabeat could consider targeting these users with daily reminders.
2. High Performers (Enthusiasts): Conversely, there is a notable number of observations recording well over 10,000 steps. These high-value observations raise the average step count to approximately 7,600 (refer

to summary statistics), placing it within the 5,000–7,000 range. Bellabeat can target these enthusiasts by recommending workout plans and fitness goals.

It would be beneficial to determine the percentage of all individuals whose average daily steps fall below 6,000 (the midpoint of 5000-7000).

This code calculates the percentage of the 33 respondents whose Average Daily Step count is below 6000

```
daily_activity_sleep %>% group_by(Id) %>% ##
  summarize(meanStep=mean(Steps)) %>% ## Caclulate avg steps for each Id
  filter(meanStep < 6000) %>%  ## Filter for less than 500
  nrow() * 100 / 33 ## Calculate %age
```

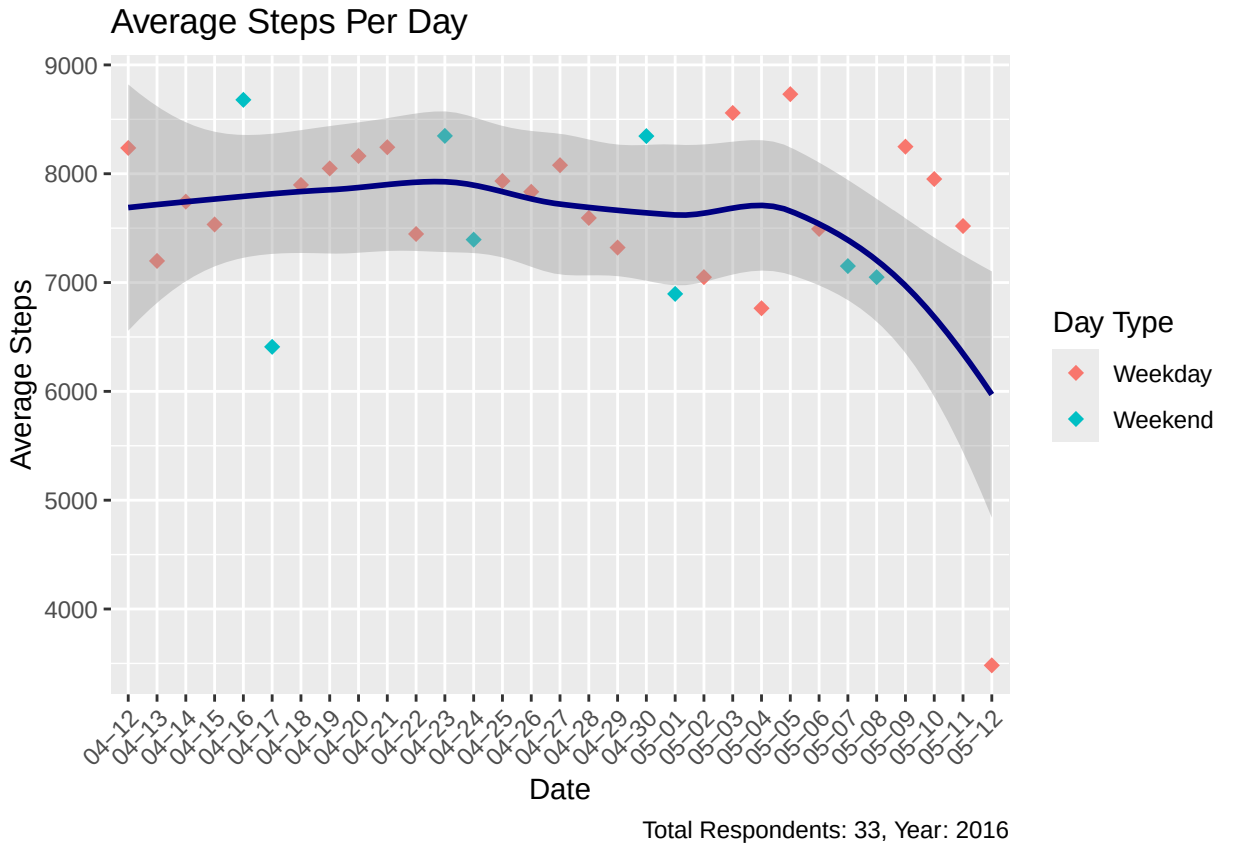
```
## [1] 33.33333
```

Approximately 33.3% or **one third** of participants have a daily average step count of **less than 6000**. This is a very high percentage and Bellabeat should work towards targeting this

Average Daily Steps Trend

We will now examine the time-based trend of the average total steps covered.

```
daily_activity_sleep %>% group_by(RecordedDate,DayType) %>%
  summarise(daily_avg= mean(Steps), .groups = 'drop') %>% # Calculate mean per day
  ggplot(aes(x=format(as.Date(RecordedDate), format = "%m-%d"), # Remove Year
             y=daily_avg, group = 1, color=DayType)) +
  geom_point(shape=18, size=2.5) +
  geom_smooth(color = "navyblue") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1)) +
  labs(
    x = "Date", # New x-axis title
    y = "Average Steps", # New y-axis title
    title = "Average Steps Per Day", # New title
    color="Day Type", # New legend title
    caption = "Total Respondents: 33, Year: 2016"
  )
```



Similar to Calories, the average of daily steps remains relatively stable throughout April but shows a decline in May. There is also no observed variation based on day type (weekends compared to weekdays).

The drop in daily steps could be attributed to the same two factors discussed earlier (Rising Summer Temperatures or Possible Attrition of High Performers)

2.2.1.4 Analyzing Total Active Minutes and Sedantary Minutes

Lets next analyze the Minutes variables

Recall that in the Join statement Total Active Minutes was created as the sum of Fairly Active Minutes, Lightly Active Minutes and Very Active Minutes

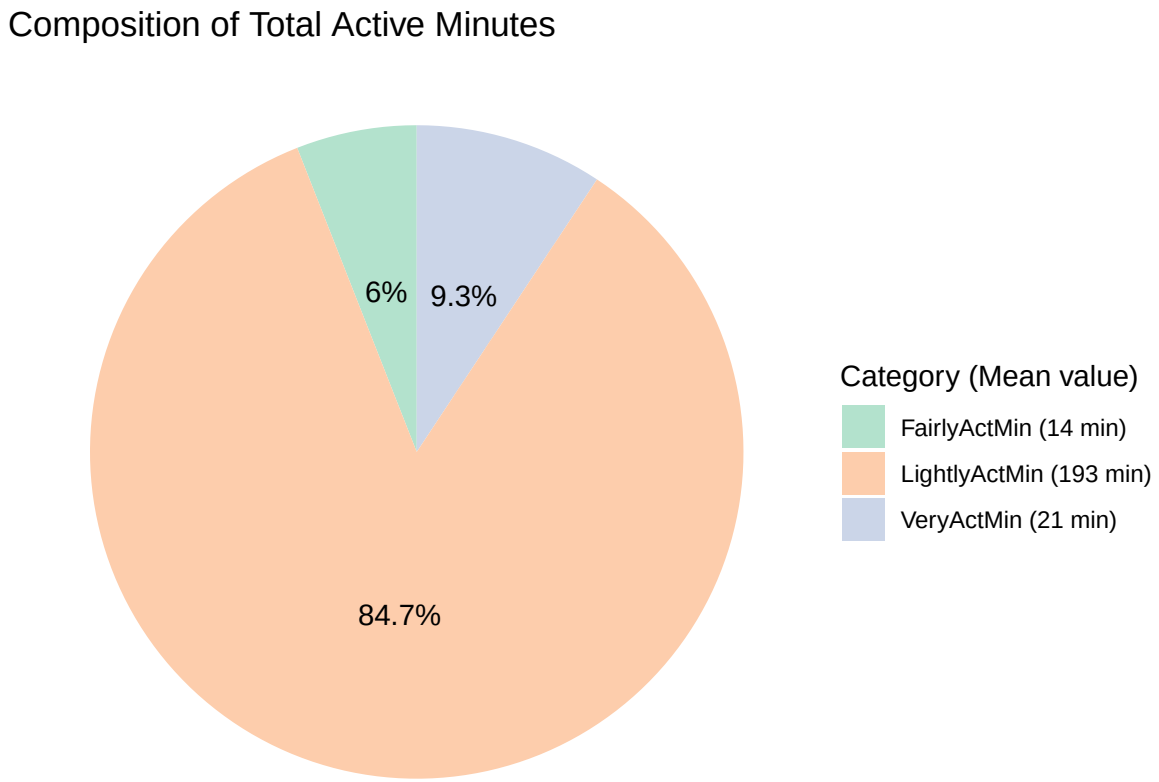
Composition of Total Active Min

Lets first review the composition of Total Active Minutes.

The code below calculates the sum of Fairly Active Minutes, Lightly Active Minutes and Very Acive Minutes and then creates a column for percentages as a portion of the whole. A pie-chart is then created to visualize the proportions. Means for respective categories are also shown.

```
# Identify data for the plot

# Select FairlyActMin, LightlyActMin, VeryActMin
# Use the apply function to sum all rows in each of the columns
# Calculate percentage by dividing each sum value with the sum of TotalActiveMinutes
# Update Category to include means by dividing each sum values with total rows
# in the joined data set

plot_data <- daily_activity_sleep[,c("FairlyActMin","LightlyActMin","VeryActMin")] %>%
  apply(2,sum) %>% as.data.frame() %>% setNames("value") %>%
```

```
  rownames_to_column("category") %>%
  mutate(percentage = value / sum(daily_activity_sleep$TotalActMin) * 100,
         means=value/nrow(daily_activity_sleep),
         category = paste0(category, " (", round(means, 0)," min)")) %>%
  arrange(value)

#Plot the data using geom_bar
ggplot(plot_data, aes(x = "", y = value, fill = category)) +
  geom_bar(stat = "identity", width = 1) + #stacked bar chart
  coord_polar(theta = "y") + #To convert the chart into a pie-chart
  theme_void() + # Removes background grid and axes
  labs(title = "Composition of Total Active Minutes",
       fill='Category (Mean value)') + #Add title and legend
  geom_text(aes(label = paste0(round(percentage, 1), "%")),
            position = position_stack(vjust = 0.5)) + #Add labels for %age
  scale_fill_brewer(palette = "Pastel2")
```



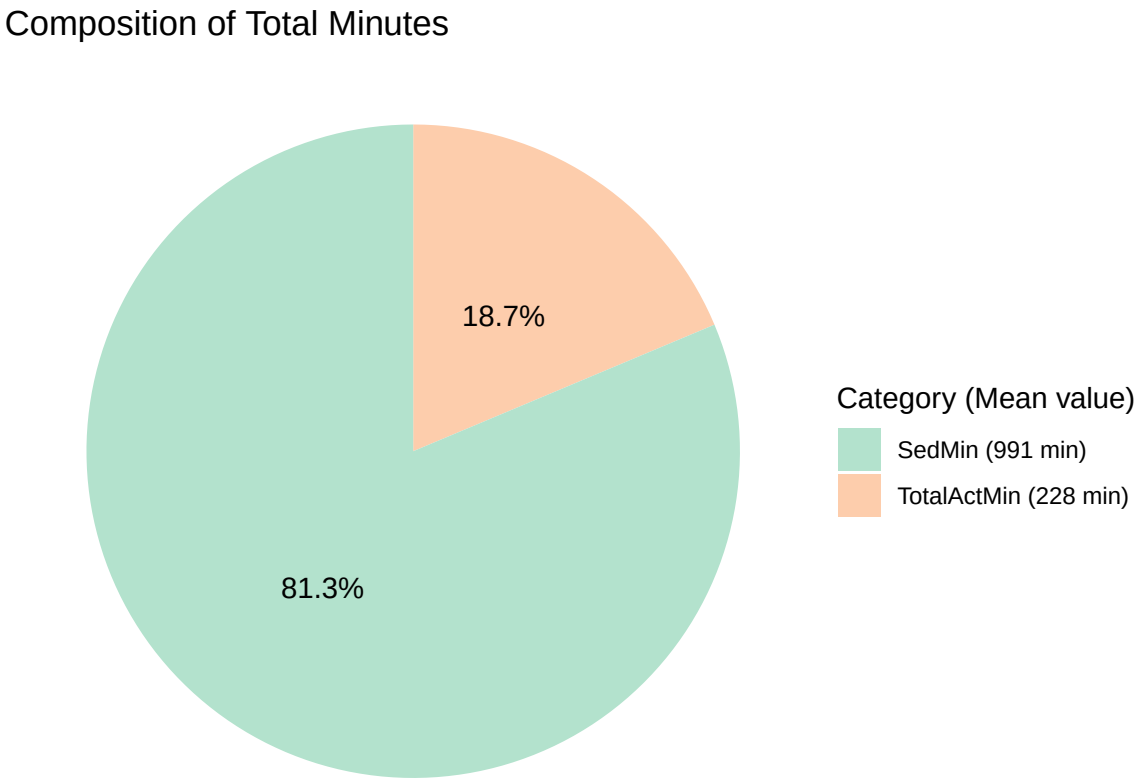
Lightly Active Minutes account for approximately 85% of the total activity. This significant proportion explains why the average for Lightly Active Minutes (193 minutes or 3.2 hours) is notably higher than the averages for Fairly Active Minutes (14 minutes) and Very Active Minutes (21 minutes). Overall, the data indicates that most respondents are only lightly active.

Composition of Total Minutes

Lets next review the composition of Total Minutes as a sum of Total Active Minutes and Sedentary Minutes.

The code below calculates the sum of Total Active Minutes and Sedentary Minutes and then creates a column for percentages as a portion of the whole. A pie-chart is then created to visualize the proportions. Means for respective categories are also shown. Means for respective categories are also shown.

```r
# Identify data for the plot

# Select TotalActiveMinutes and SedentaryMinutes
# Use the apply function to sum all rows in each of the columns
# Calculate  percentage by dividing each sum value with the sum of TotalMinutes
# Update Category to include means by dividing each sum values with total rows
# in the joined data set

plot_data <- daily_activity_sleep[,c("TotalActMin","SedMin")] %>%
  apply(2,sum) %>% as.data.frame() %>% setNames("value") %>%
  rownames_to_column("category") %>%
  mutate(percentage = value / sum(value) * 100,
         means=value/nrow(daily_activity_sleep),
         category = paste0(category, " (", round(means, 0)," min)")) %>%
  arrange(value)

#Plot the data using geom_bar
ggplot(plot_data, aes(x = "", y = value, fill = category)) +
  geom_bar(stat = "identity", width = 1) + #stacked bar chart
  coord_polar(theta = "y") + #To convert the chart into a pie-chart
  theme_void() + # Removes background grid and axes
  labs(title = "Composition of Total Minutes",
       fill='Category (Mean value)') + #Add title and legend
  geom_text(aes(label = paste0(round(percentage, 1), "%")),
            position = position_stack(vjust = 0.5)) + #Add labels for %age
  scale_fill_brewer(palette = "Pastel2")
```



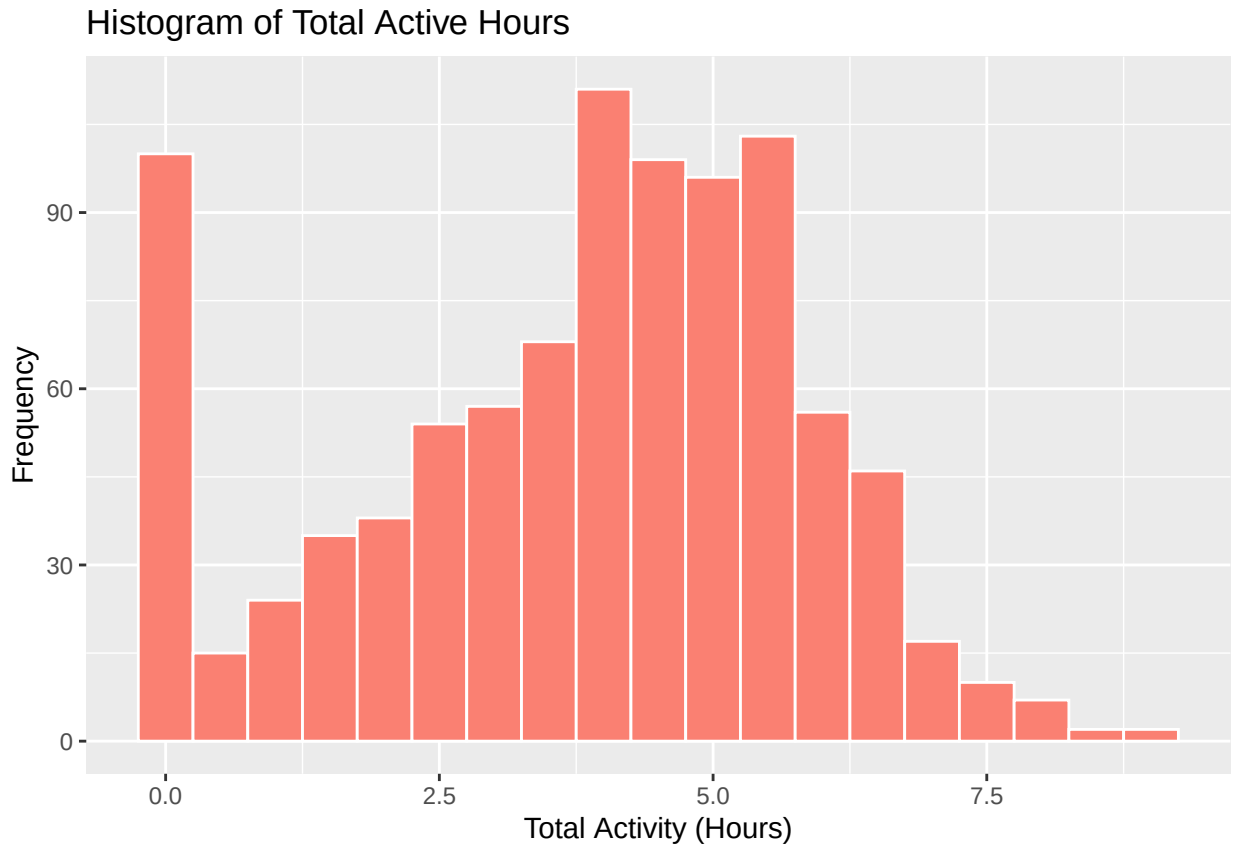
Sedentary Minutes constitute a striking 82% of the total time. This high proportion directly leads to a very high average of 991 minutes (or 16.5 hours) for Sedentary Minutes, significantly overshadowing the average of

228 minutes (or 3.8 hours) for Total Active Minutes. This data strongly suggests a considerably sedentary lifestyle among the users, which is a key area Bellabeat should target for intervention to promote lower sedentary behavior.

Distribition of Total Active Hours

Next, lets use a histogram to visualize the distribution of Total Active Hours. Minutes are converted to hours for easier readability

```
ggplot(daily_activity_sleep, aes(TotalActMin/60)) +
 geom_histogram(fill = 'salmon', color = 'white', binwidth = 0.5) +
  labs(title ="Histogram of Total Active Hours",
  x = "Total Activity (Hours)",
  y = "Frequency")
```



Similar to Calories and Steps, the distribution for active hours exhibits a bimodal nature, indicating two distinct peaks. The data suggests that the participant group includes individuals who are both significantly high and significantly low performers as:

1. Low Performers (Low Outliers): A notable peak exists at observations with Total Activity less than 30 minutes, which is below the recommended daily activity time*. Bellabeat can address this group by sending reminders to users on days when their daily activity falls below 30 minutes.

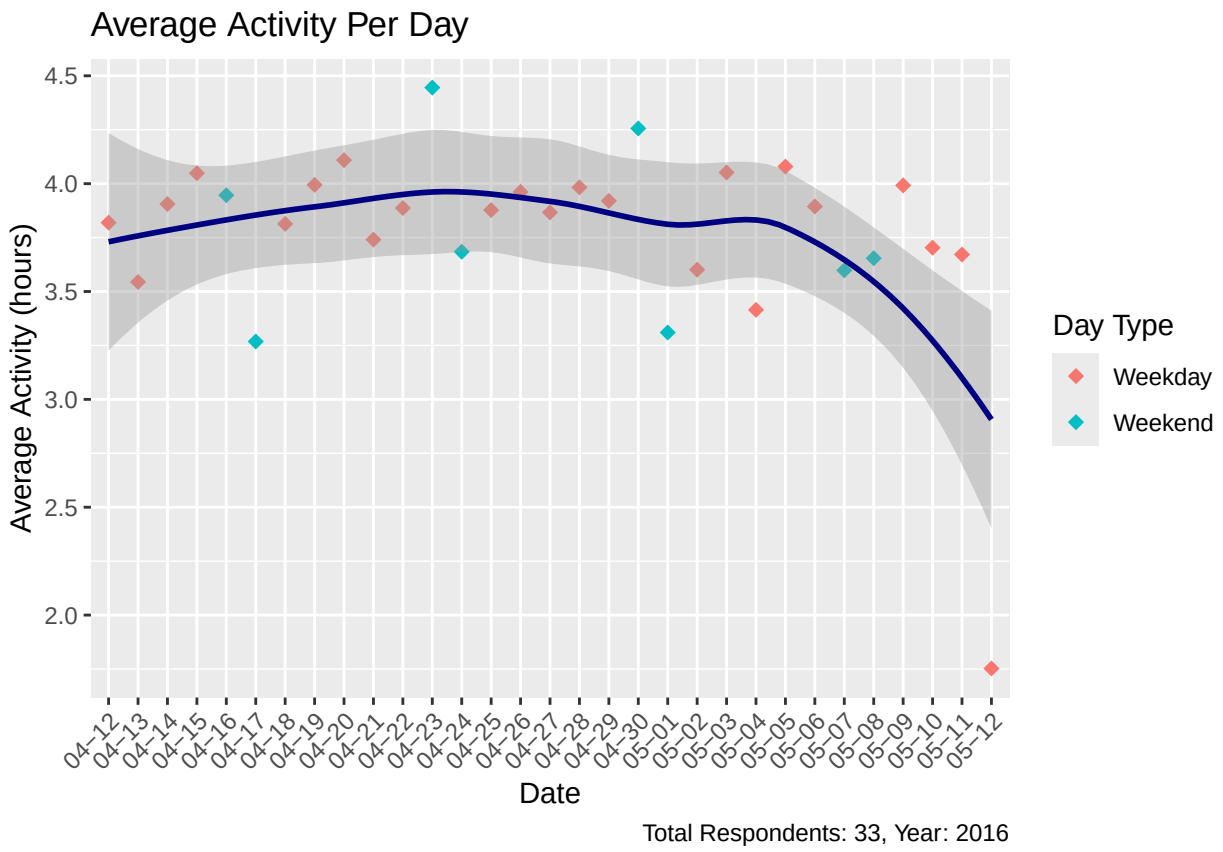
https://www.cdc.gov/physical-activity-basics/guidelines/adults.html

2. High Performers (Enthusiasts): The remaining distribution peaks around 3.75 hours, suggesting the presence of users with high activity levels. Bellabeat can engage these enthusiasts by recommending workout plans and fitness goals.

Average of Daily Activity Trend

We will now examine the time-based trend of the average daily activity.

```
daily_activity_sleep %>% group_by(RecordedDate,DayType) %>%
  summarise(daily_avg= mean(TotalActMin/60), .groups = 'drop') %>% # Calculate mean per day
  ggplot(aes(x=format(as.Date(RecordedDate), format = "%m-%d"), # Remove Year
             y=daily_avg, group = 1, color=DayType)) +
  geom_point(shape=18, size=2.5) +
  geom_smooth(color = "navyblue") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1)) +
  labs(
    x = "Date", # New x-axis title
    y = "Average Activity (hours)", # New y-axis title
    title = "Average Activity Per Day", # New title
    color="Day Type", # New legend title
    caption = "Total Respondents: 33, Year: 2016"
  )
```



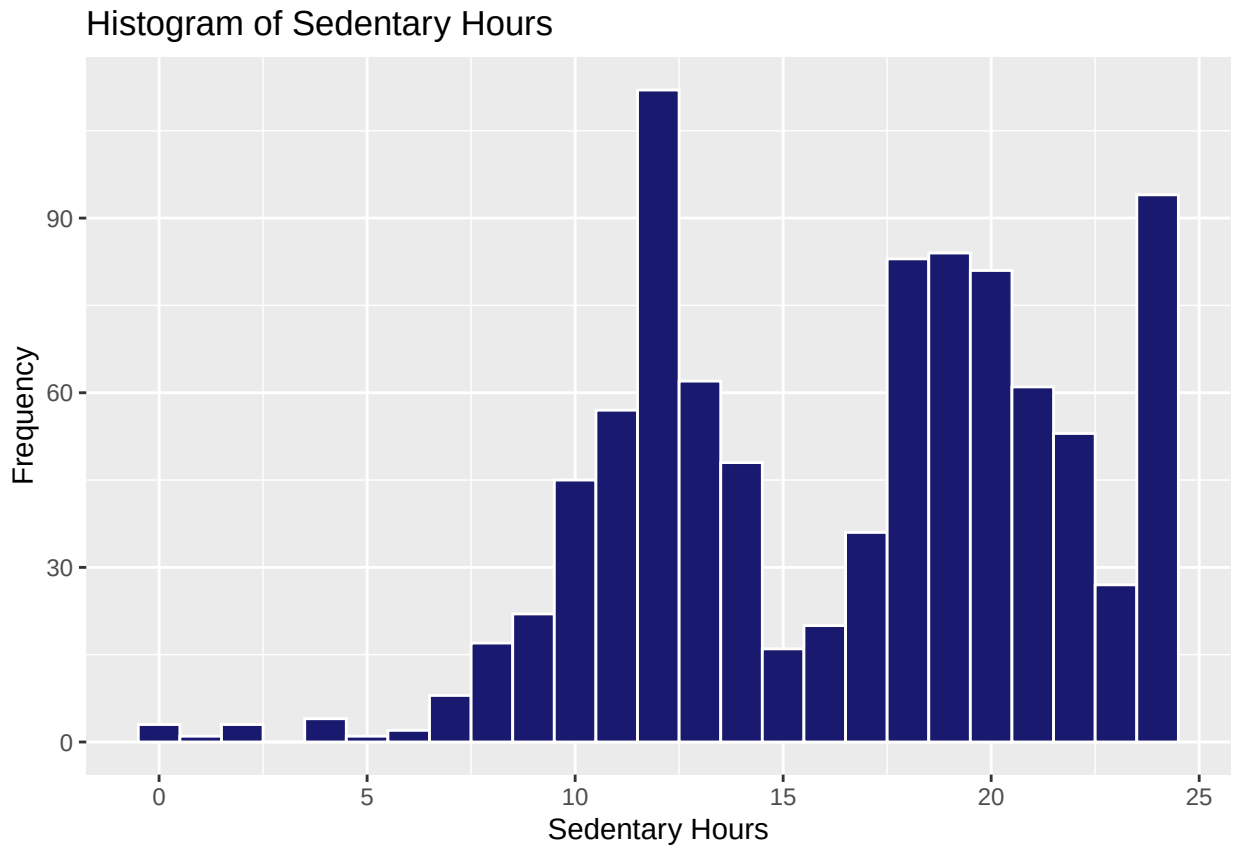
Similar to Calories and Steps, the average of activity level remains relatively stable throughout April but shows a decline in May. There is also no observed variation based on day type (weekends compared to weekdays).

The drop in activity level could be attributed to the same two factors discussed earlier (Rising Summer Temperatures or Possible Attrition of High Performers)

Distribition of Sedentary Hours

Next, lets use a histogram to visualize the distribution of Sedantary hours. Minutes are converted to hours for easier readability

```r
ggplot(daily_activity_sleep, aes(SedMin/60)) +
 geom_histogram(fill = 'midnightblue', color = 'white', binwidth = 1) +
  labs(title ="Histogram of Sedentary Hours",
  x = "Sedentary Hours",
  y = "Frequency")
```



The distribution of Sedentary Hours, similar to the Activity data, exhibits a multi-modal nature, suggesting the presence of three distinct peaks. This indicates that the participant group consists of individuals with varying performance levels:

1. Low Performers: A notable peak is observed for observations with Total Sedentary Hours exceeding 17.5 hours, which represents a very high level. Accounting for 8 hours of sleep, this leaves 9.5 hours, equating to approximately 60% of the time awake (60% of 16 hours). This signifies that a portion of users spend more than half of their day in a sedentary lifestyle. Bellabeat should again target this group with reminders to engage in activity when sedentary hours surpass a specific threshold. Suggestions such as "Time to take a walk" or "Time for some stretches" could be beneficial.

2. High Performers (Enthusiasts): A peak is also evident at approximately 12 hours, which is commendable, considering the allocation of 8 hours for sleep. Bellabeat can engage these enthusiasts by recommending tailored workout plans and fitness goals, while also incorporating the recommended amount of breaks.

2.2.1.5 Analyzing Distance

Lets next analyze the Distance Variables.

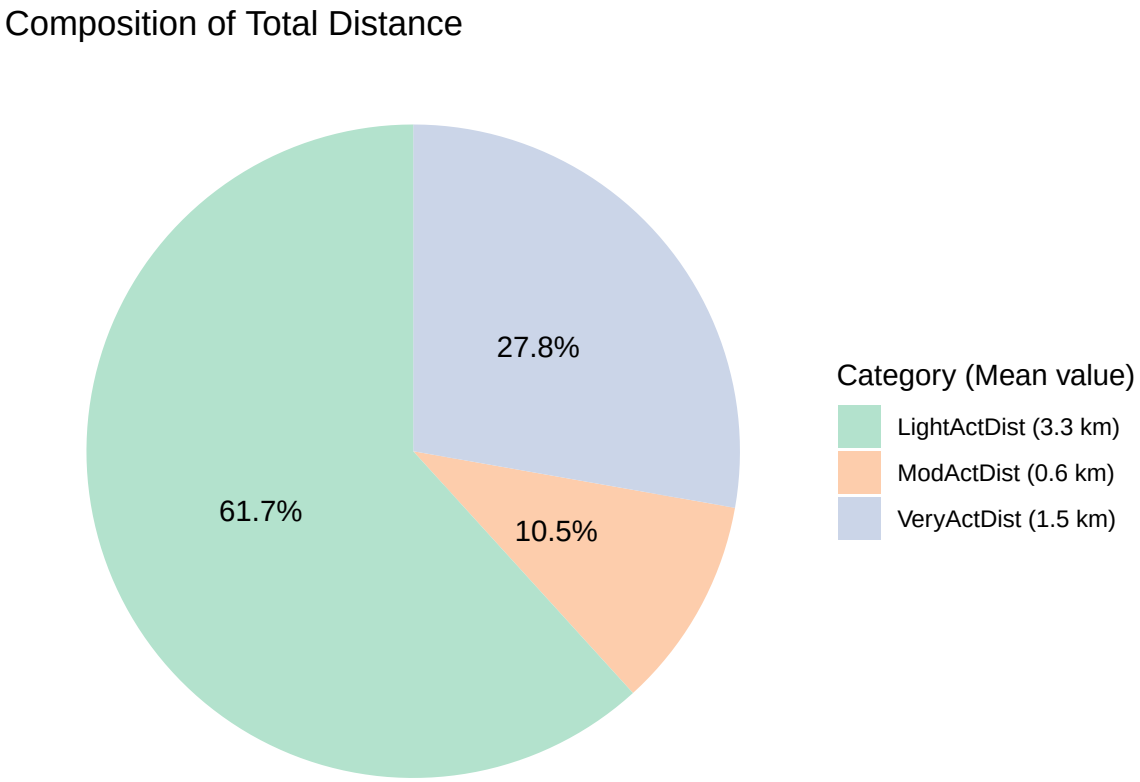
Recall that, in the Join statement, Total Distance was created as the sum of Light Active Distance, Moderately Active Distance and Very Active Distance

Composition of Total Distance

Lets first review the composition of Total Distance.

The code below calculates the sum of Light Active Distance, Moderately Active Distance and Very Active Distance and then divides each sum by the sum of Total Distance. A pie-chart is then created to visualize the proportions.

```
# Identify data for the plot

# Select LightActDist, ModActDist, VeryActDist
# Use the apply function to sum all rows in each of the columns
# Calculate  percentage by dividing each sum value with the sum of TotalDistance
# Update Category to include means by dividing each sum values with total rows
# in the joined data set

plot_data <- daily_activity_sleep[,c("LightActDist","ModActDist","VeryActDist")] %>%
  apply(2,sum) %>% as.data.frame() %>% setNames("value") %>%
  rownames_to_column("category") %>%
  mutate(percentage = value / sum(daily_activity_sleep$TotalDist) * 100,
         means=value/nrow(daily_activity_sleep),
         category = paste0(category, " (", round(means, 1)," km)")) %>%
  arrange(value)

#Plot the data using geom_bar
ggplot(plot_data, aes(x = "", y = value, fill = category)) +
  geom_bar(stat = "identity", width = 1) + #stacked bar chart
  coord_polar(theta = "y") + #To convert the chart into a pie-chart
  theme_void() + # Removes background grid and axes
  labs(title = "Composition of Total Distance",
       fill='Category (Mean value)') + #Add title and legend
  geom_text(aes(label = paste0(round(percentage, 1), "%")),
            position = position_stack(vjust = 0.5)) + #Add labels for %age
  scale_fill_brewer(palette = "Pastel2")
```

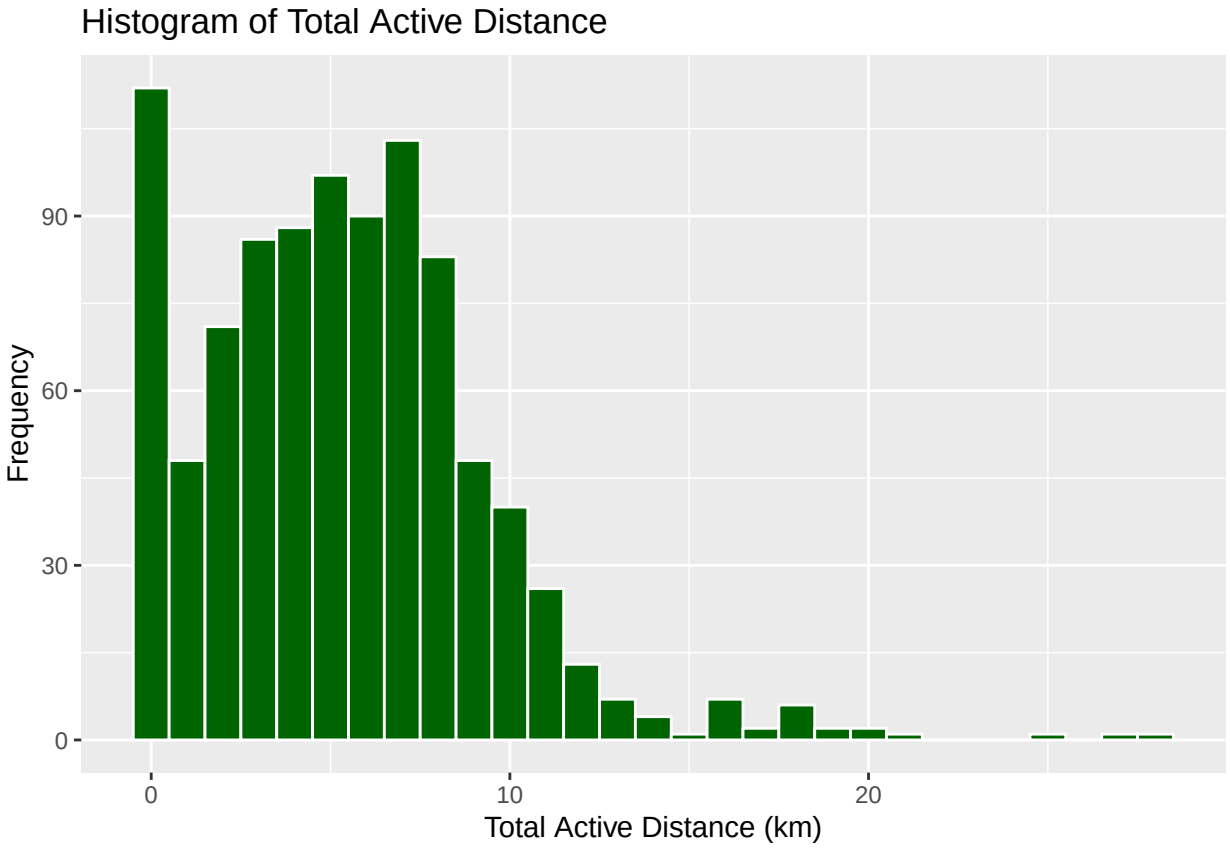


The analysis of activity distance reveals a significant variation in participant performance levels, underscoring the necessity for personalized recommendations. Specifically, LightlyActiveDistance accounts for the largest proportion of the data, comprising about 62% with an average of 3.3 km. This is followed by VeryActiveDistance at approximately 28% (mean of 1.5 km) and ModeratelyActiveDistance at 10% (mean of 0.6 km). These disparate distribution patterns strongly indicate that a one-size-fits-all approach is ineffective, necessitating the development of user-targeted plans.

Distribition of Total Distance

Next, lets use a histogram to visualize the distribution of TotalDistance

```
ggplot(daily_activity_sleep, aes(TotalDist)) +
 geom_histogram(fill = 'darkgreen', color = 'white', binwidth = 1) +
  labs(title ="Histogram of Total Active Distance",
  x = "Total Active Distance (km)",
  y = "Frequency")
```



The distribution of Total Active Distance data, similar to Total Steps, is right-skewed (positively skewed), indicating a concentration of data points on the left side of the graph. A small number of extremely high values (outliers) skew the mean, resulting in a value greater than the median, as demonstrated by the summary statistics.

Furthermore, the distribution for Total Active Distance is bimodal, which suggests the participant cohort comprises individuals who are both significantly high and significantly low performers:

1. Low Performers (Low Outliers): A notable peak occurs for observations with a daily active distance less than 1 km. A substantial portion of the distribution is concentrated below 4 km (2.5 miles), which is the recommended threshold for classifying an individual as "active"*. Bellabeat may consider engaging these users with daily reminders.

*https://www.health.com/how-many-miles-should-you-walk-a-day-8736501

2. High Performers (Enthusiasts): Conversely, the data exhibits a second peak around 5–7 km (3–4 miles). These relatively high observations elevate the average daily distance to approximately 5.5 km (as indicated in the summary statistics). Bellabeat has the opportunity to target these enthusiasts by recommending personalized workout plans and fitness goals to facilitate further improvement.

It would be beneficial to ascertain the percentage of all 33 participants whose average daily active distance is less than 4 km (2.5 miles). The code below calculates this

```
daily_activity_sleep %>% group_by(Id) %>% ##
  summarize(meanStep=mean(TotalDist)) %>% ## Caclulate avg steps for each Id
  filter(meanStep < 4) %>%  ## Filter for less than 500
  nrow() * 100 / 33 ## Calculate %age
```

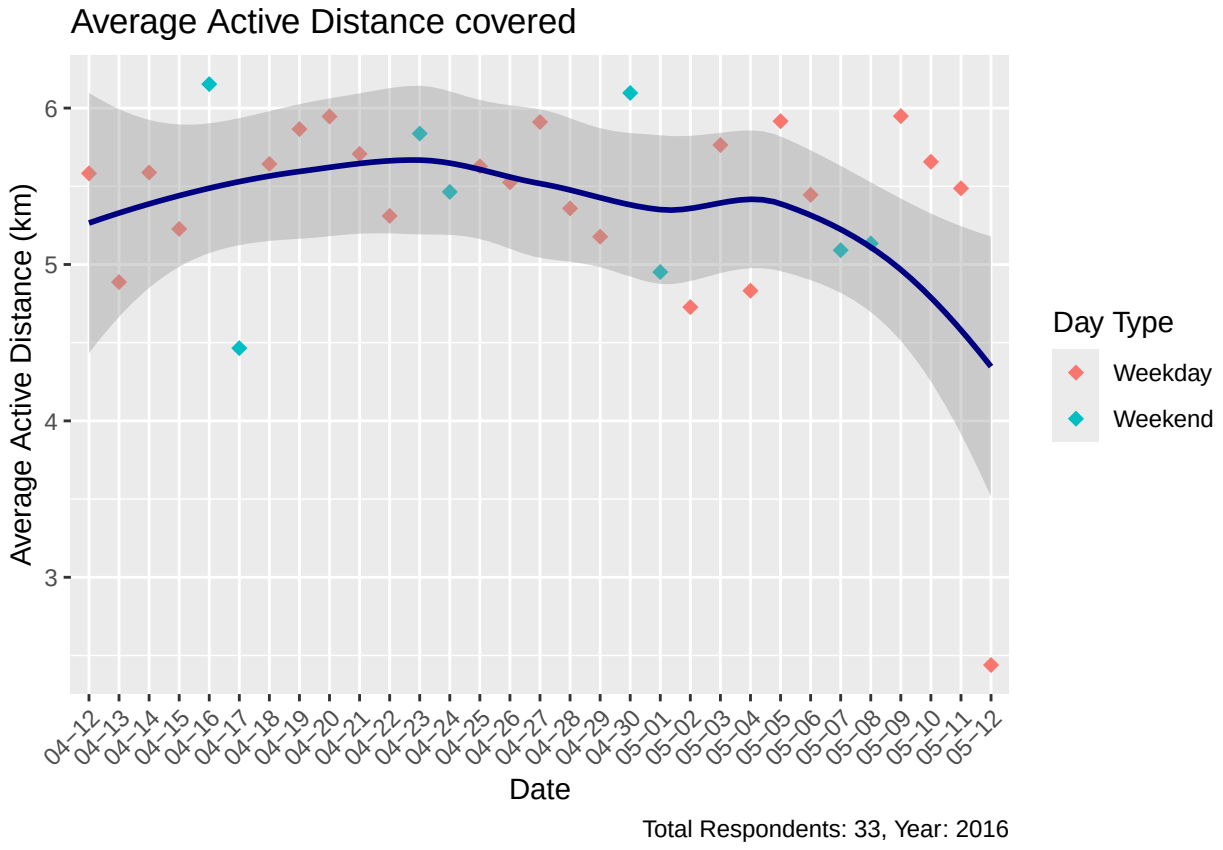
```
## [1] 30.30303
```

Approximately **30%** of participants exhibit a daily Active Distance of **less than 4 km**. This constitutes a substantial percentage that Bellabeat should prioritize in its targeting strategy.

Average of Daily Total Distance Trend

We will now examine the time-based trend of the average total distance covered.

```
daily_activity_sleep %>% group_by(RecordedDate,DayType) %>%
  summarise(daily_avg= mean(TotalDist), .groups = 'drop') %>% #Calculate mean per day
  ggplot(aes(x=format(as.Date(RecordedDate), format = "%m-%d"), #Remove Year
             y=daily_avg, group = 1, color=DayType)) +
  geom_point(shape=18, size=2.5) +
  geom_smooth(color = "navyblue") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1)) +
  labs(
    x = "Date", # New x-axis title
    y = "Average Active Distance (km)", # New y-axis title
    title = "Average Active Distance covered", # New title
    color="Day Type", # New legend title
    caption = "Total Respondents: 33, Year: 2016"
  )
```



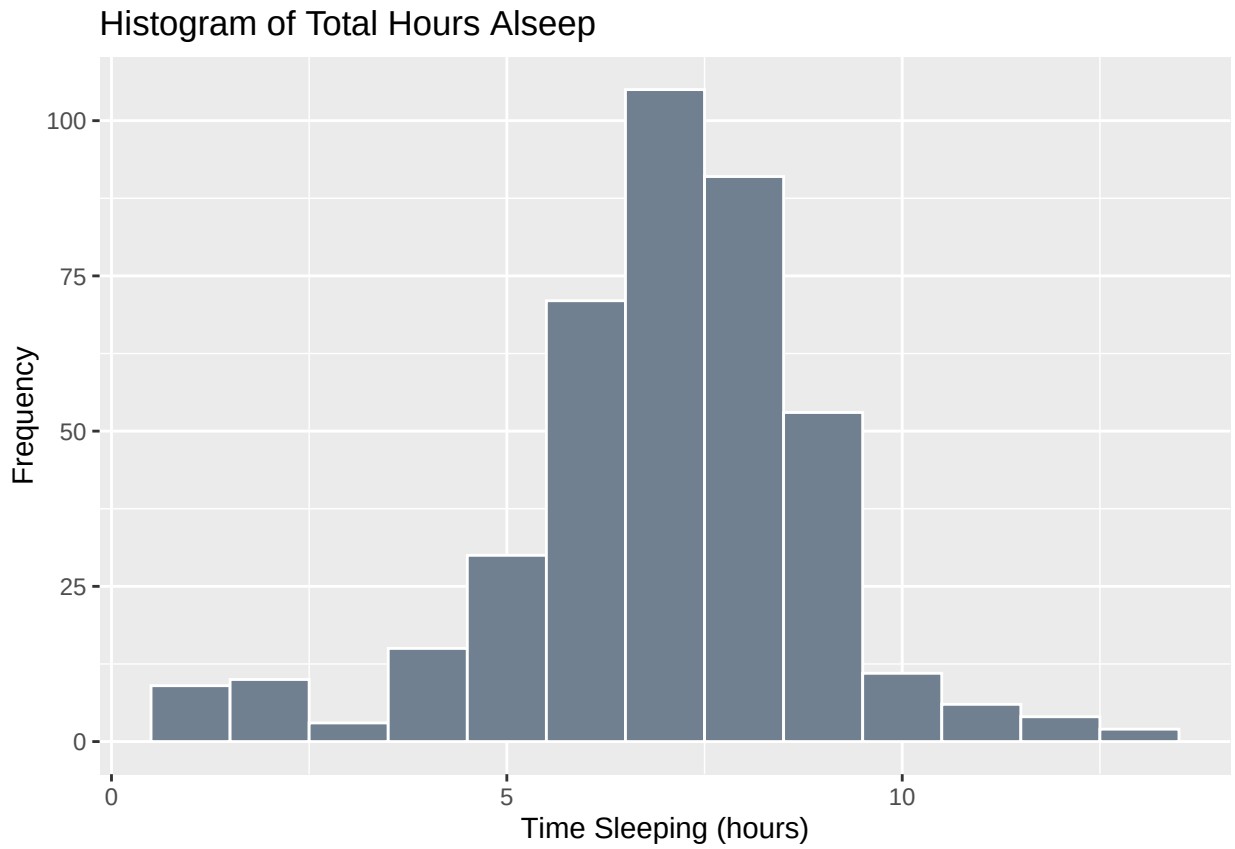
Similar to Calories, Steps and Activities, the average of Total Active Distance remains relatively stable throughout April but shows a slight decline in May. There is also no observed variation based on day type (weekends compared to weekdays).

The drop in activity level could be attributed to the same two factors discussed earlier (Rising Summer Temperatures or Possible Attrition of High Performers)

2.2.1.6 Analyzing Hours Asleep Composition of Total Hours Asleep

Next, lets use a histogram to visualize the distribution of Total Hours Asleep. Minutes are converted to hours for easier readability

```
ggplot(daily_activity_sleep, aes(AsleepMin/60)) +
 geom_histogram(fill = 'slategray', color = 'white', binwidth = 1) +
  labs(title ="Histogram of Total Hours Alseep",
  x = 'Time Sleeping (hours)',
  y = "Frequency")
```



The majority of users report approximately 7–8 hours of sleep per day, which aligns with general recommendations for adults*. As noted in the Data Processing and Cleaning section, most individuals sleep only once per day, with the TotalSleepRecords variable indicating a value of 1 for 88% of observations.

*https://www.sleepfoundation.org/how-sleep-works/how-much-sleep-do-we-really-need

For users at the extremes of the distribution (those sleeping less than 4 hours or more than 6 hours), Bellabeat may implement reminders or alarms to promote adherence to the recommended guidelines. Bellabeat can also help users set up sleep schedules.

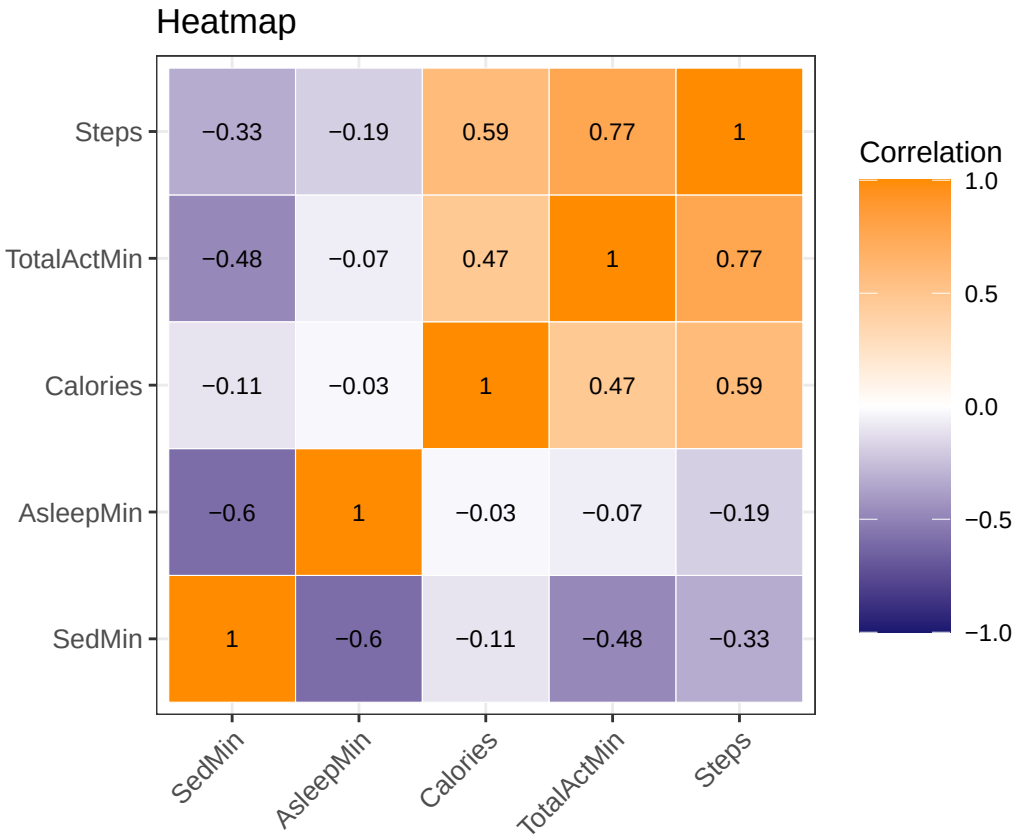
2.2.2 Identifying Relationships

2.2.2.1 Correlation Matrix Heat Map Exclude indvidual components Exclude TotalDistance as it is likely to be almost linearly correlated with Steps Exclude TimeInBed as it is likely to be almost linearly correlated with MinutesAsleep

```
cat(paste0("Correlation between Total Distand and Total Steps: ",
           cor(daily_activity_sleep[,c("TotalDist", "Steps")],
               use="pairwise.complete.obs")[1,2] %>% round(2),
           "\nCorrelation between Time In Bed and Minutes Asleep: ",
           cor(daily_activity_sleep[,c("TimeInBed", "AsleepMin")],
               use="pairwise.complete.obs")[1,2] %>% round(2)))
```

```
## Correlation between Total Distand and Total Steps: 0.97
## Correlation between Time In Bed and Minutes Asleep: 0.93
```

```
hmap_cols <- c("TotalActMin", "Steps","Calories","SedMin","AsleepMin")

# Create correlation matrix
corr_matrix <- cor(daily_activity_sleep[,hmap_cols], #Only select numeric data
                   use="pairwise.complete.obs") %>%
  round(2) #round correlation matrix

ggcorrplot(corr_matrix,
           method='square',
           hc.order = TRUE,    # Automatic hierarchical clustering
           colors = c("midnightblue", "white", "darkorange"),
           lab = TRUE,         # Add correlation coefficients
           outline.color='white', #Add outline color
           lab_size = 3.2,      # Adjust label size
           tl.cex = 10,  # Adjust text label size
           legend.title='Correlation', #Add legend title
           ggtheme = ggplot2::theme_bw, # Set theme
           title= 'Heatmap') +
           theme(legend.key.size = unit(1.2, "cm")) #Set legend size
```



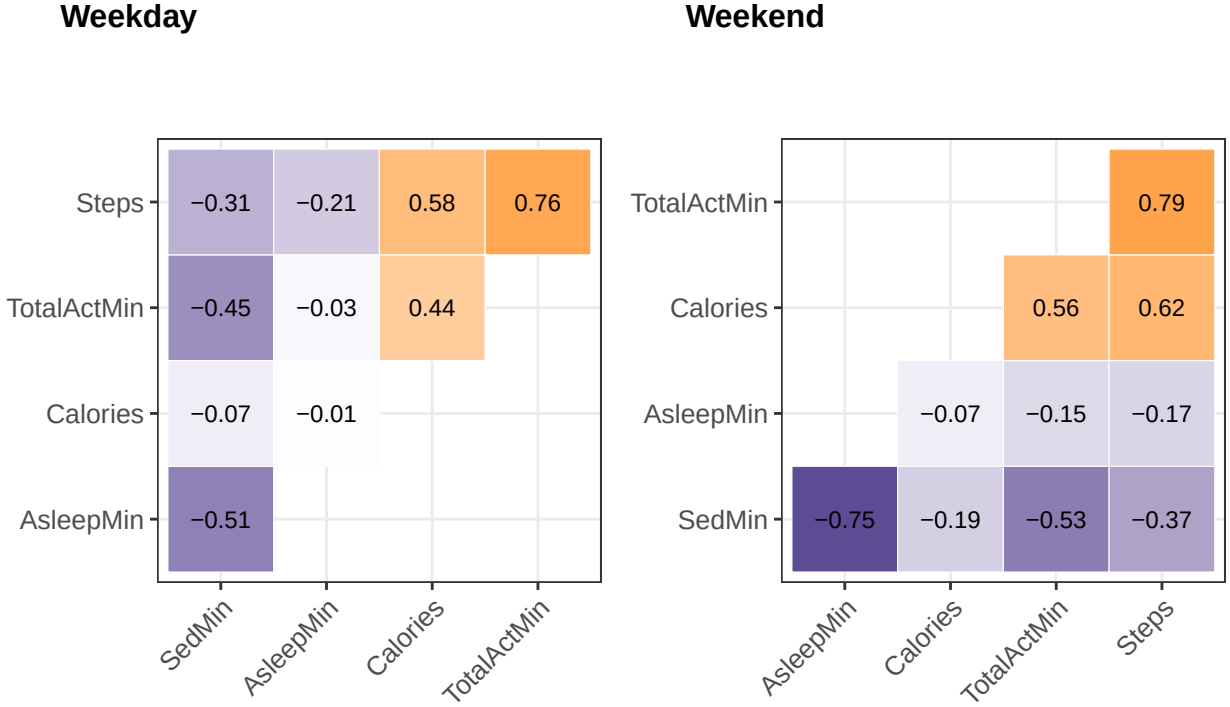
sed min all very strong -ve correlations asleep min relatively weaker correlations (with variables other than sed min) but all negative

calories, total actmin and steps, strong positive correlations

Weekdays Vs Weekend

```
# correlation matrix with data filtered for weekday
corr_mat_wd <- cor(daily_activity_sleep %>% filter(DayType == 'Weekday')  %>% .[,hmap_cols],
                   use="pairwise.complete.obs")  %>% round(2)

#correlation matrix with data filtered for weekend
corr_mat_wend <- cor(daily_activity_sleep %>% filter(DayType == 'Weekend') %>% .[,hmap_cols],
                     use="pairwise.complete.obs") %>% round(2)

p1 <- ggcorrplot(corr_mat_wd,
           method='square',
           type='upper',
           hc.order = TRUE,
           colors = c("midnightblue", "white", "darkorange"),
           lab = TRUE,
           outline.color='white',
           lab_size = 3.2,
           tl.cex = 10,
           ggtheme = ggplot2::theme_bw,
           show.legend=FALSE)

p2 <- ggcorrplot(corr_mat_wend,
           method='square',
```

```
            type='lower',
            hc.order = TRUE,
            colors = c("midnightblue", "white", "darkorange"),
            lab = TRUE,
            outline.color='white',
            lab_size = 3.2,
            tl.cex = 10,
            ggtheme = ggplot2::theme_bw,
            show.legend = FALSE)

plot_grid(p1, p2, ncol = 2, labels = c('Weekday', 'Weekend'), label_size = 12)
```

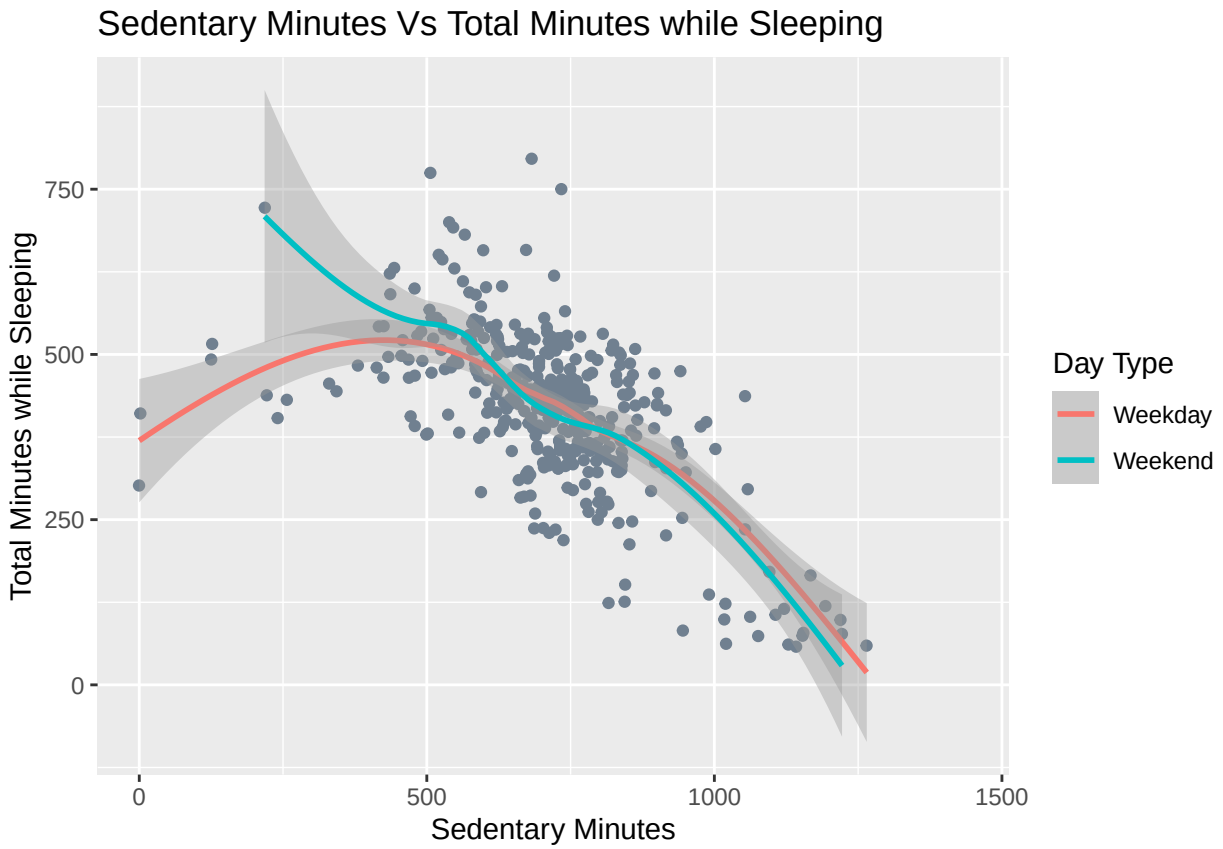


switches: sed min and asleep min, sed min and calories total act min and asleeo min,

Calories and sed minutes; relationship stronger on weekends rec: relationships stronger on weekend,

```
ggplot(data=daily_activity_sleep) +
geom_jitter(aes(x=SedMin,y=AsleepMin),color='slategray') +
geom_smooth(aes(x=SedMin,y=AsleepMin,color=DayType)) +
labs(
    x = "Sedentary Minutes", # New x-axis title
    y = "Total Minutes while Sleeping", # New y-axis title
    color = "Day Type", # New legend title
    title = "Sedentary Minutes Vs Total Minutes while Sleeping" # New title
  )
```

2.2.2.2 Sedentary Minutes and Total Minutes Asleep



very strong for weekends as compared to weekday. sharp fall for both for those days when an individual had high amount of sdeneatrt minutes. weekends: relaxing, watching movies, reading books, scrolling social media

2.2.2.3 Sedentary Minutes and Calories corr changed

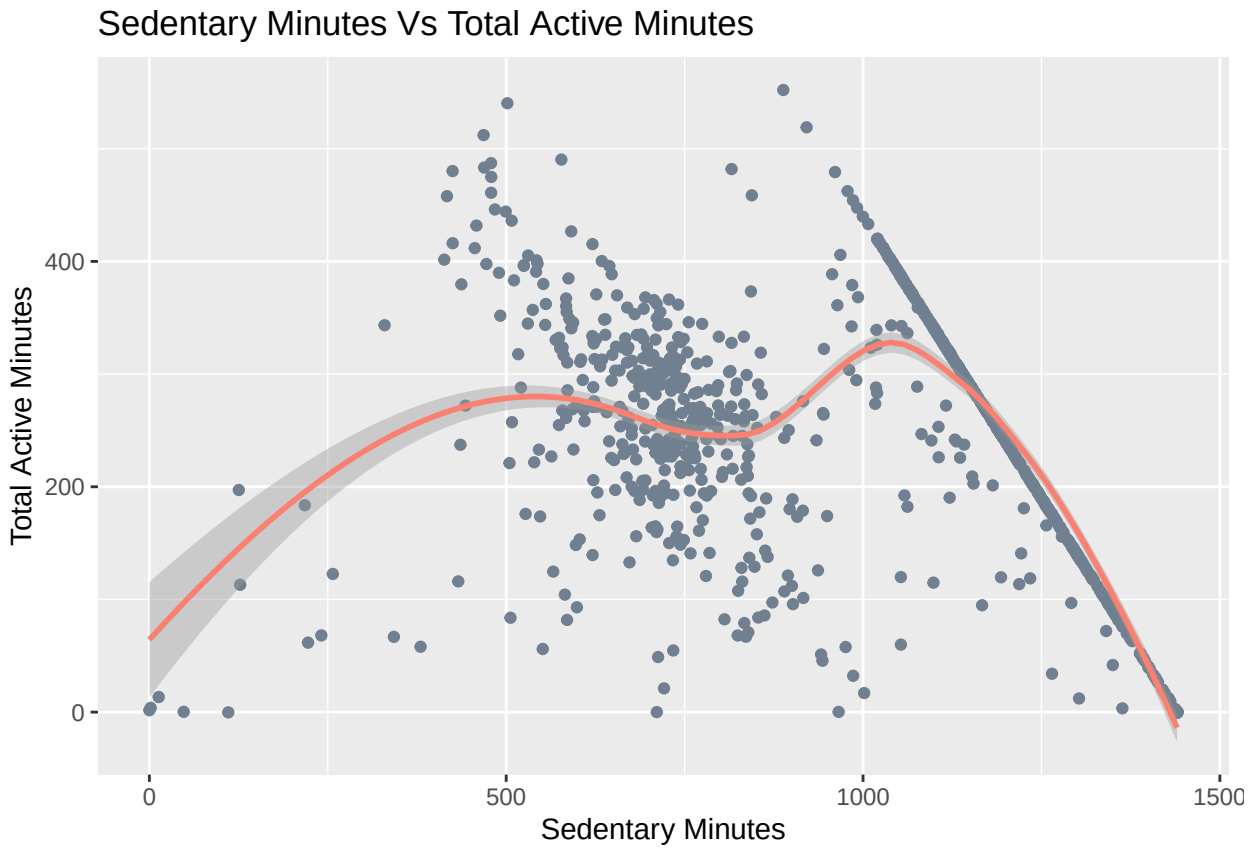
```
ggplot(data=daily_activity_sleep) +
geom_jitter(aes(x=SedMin,y=Calories), color='slategray') +
geom_smooth(aes(x=SedMin,y=Calories, color=DayType)) +
  labs(
    x = "Sedentary Minutes", # New x-axis title
    y = "Calories", # New y-axis title
    color = "Day Type", # New legend title
    title = "Sedentary Minutes Vs Calories Burned" # New title
  )
```

high performer, weekend stronger, +ve relationship, can have both inc activity and low levels of sed to relax. relaxing more and burning calories more

low perfomer, neg relationship

2.2.2.4 Sedentary Minutes and Total Active Minutes corr stronger

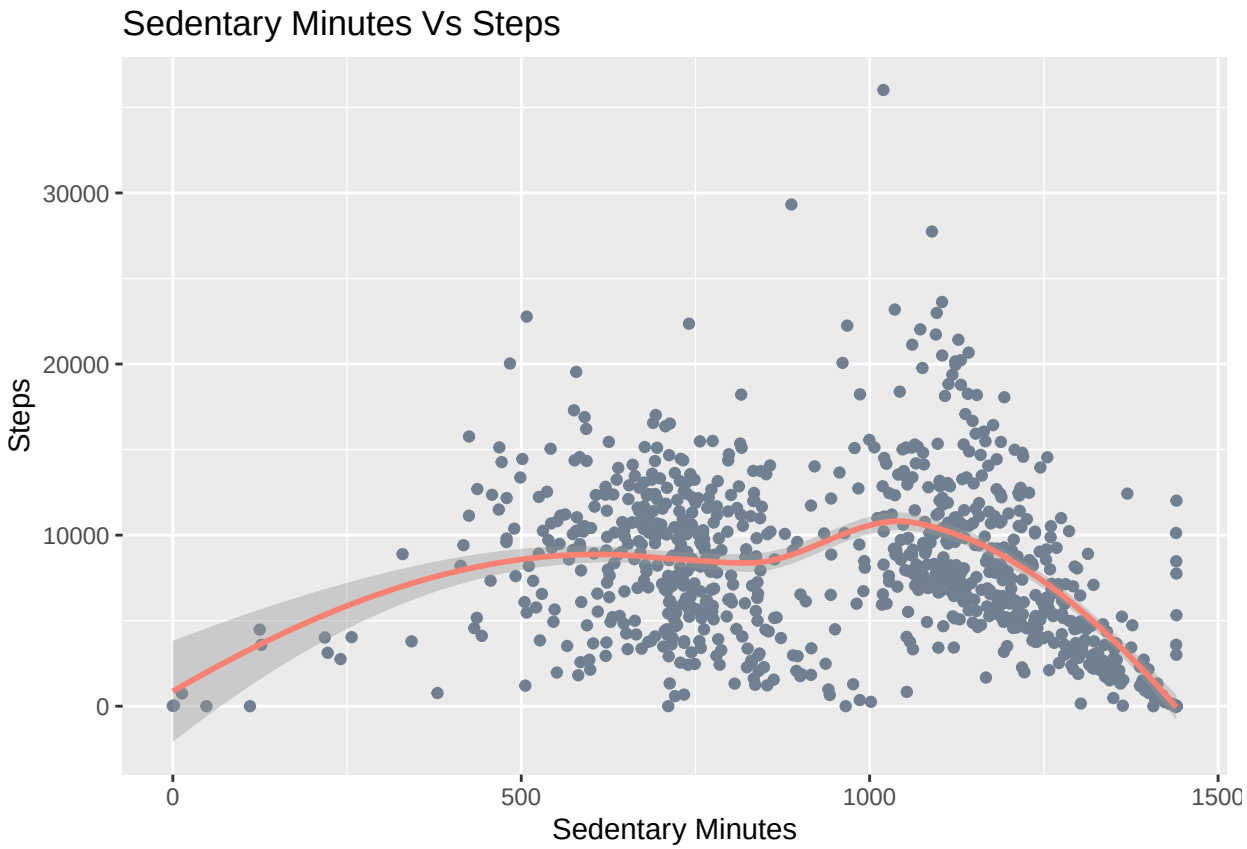
```
ggplot(data=daily_activity_sleep) +
geom_jitter(aes(x=SedMin,y=TotalActMin), color='slategray') +
geom_smooth(aes(x=SedMin,y=TotalActMin),color='salmon') +
  labs(
    x = "Sedentary Minutes", # New x-axis title
    y = "Total Active Minutes", # New y-axis title
    title = "Sedentary Minutes Vs Total Active Minutes" # New title
  )
```



high performer, weekend stronger we have extreme groups so two downfalss, sec 500 and 800, sed >900

```
ggplot(data=daily_activity_sleep) +
geom_jitter(aes(x=SedMin,y=Steps), color='slategray') +
geom_smooth(aes(x=SedMin,y=Steps), color='salmon') +
labs(
    x = "Sedentary Minutes", # New x-axis title
    y = "Steps", # New y-axis title
    title = "Sedentary Minutes Vs Steps" # New title
  )
```

2.2.2.5 Sedentary Minutes and Steps

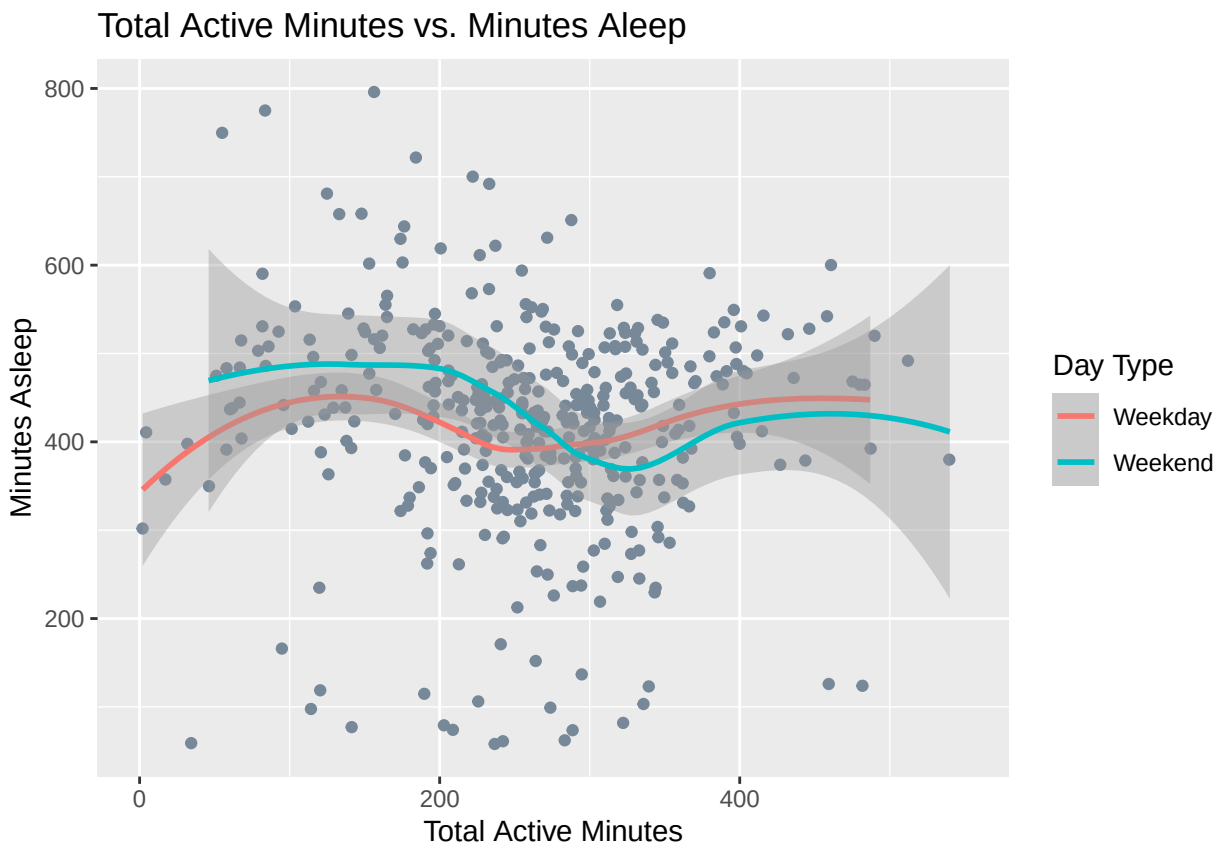


• Days when sedenatry minutes are upto 500, total steps also increase and total time asleep also increases • Days when sedenatry minutes are more 1000, total steps also decrease and total time asleep also increases

we have extreme groups so two downfalss, sec 500 and 800, sed >900

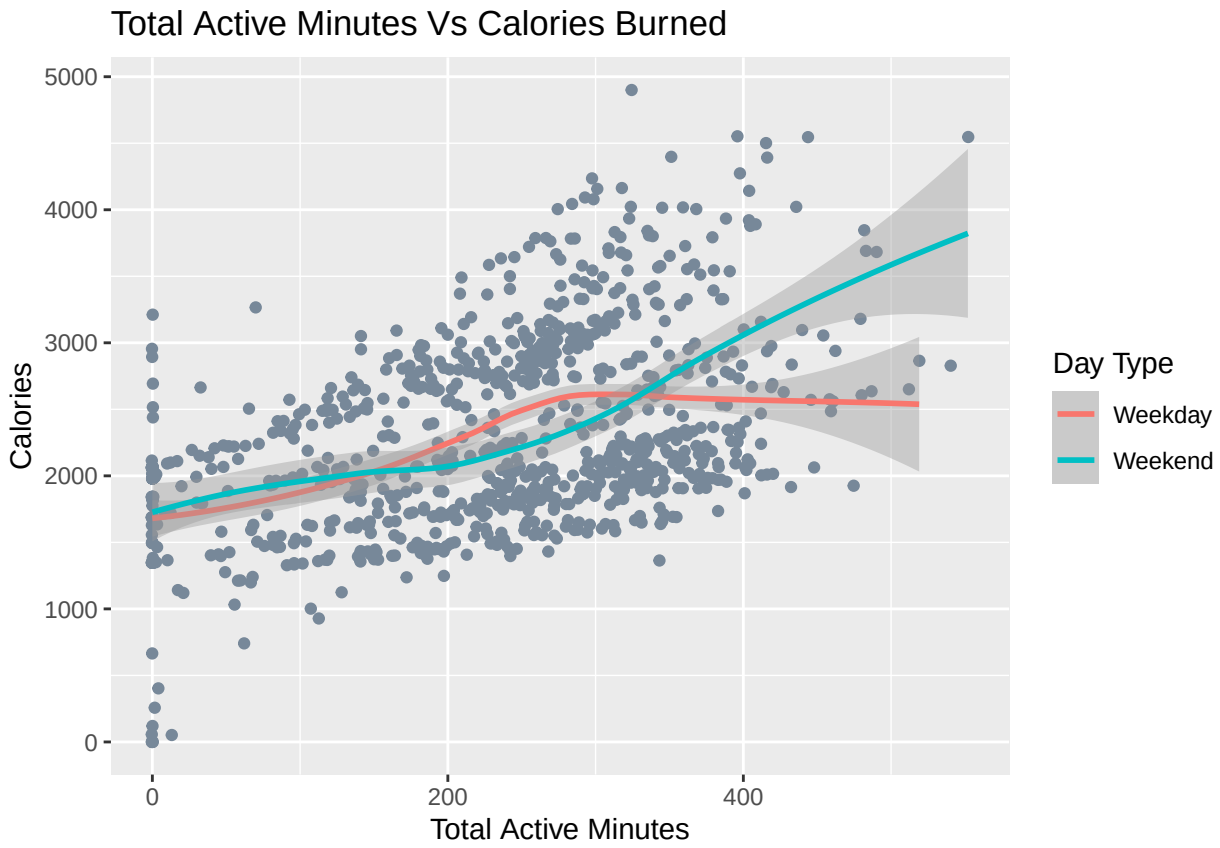
```
ggplot(data=daily_activity_sleep,) +
geom_jitter(aes(x=TotalActMin,y=AsleepMin), color='lightslategray') +
geom_smooth(aes(x=TotalActMin,y=AsleepMin, color=DayType)) +
  labs(
    x = "Total Active Minutes", # New x-axis title
    y = "Minutes Asleep", # New y-axis title
    color = "Day Type", # New legend title
    title = "Total Active Minutes vs. Minutes Aleep",
      )
```

2.2.2.6 Total Active Minutes and Total Minutes Asleep




```
ggplot(data=daily_activity_sleep) +
geom_jitter(aes(x=TotalActMin,y=Calories), color='lightslategray') +
geom_smooth(aes(x=TotalActMin,y=Calories, color=DayType)) +
  labs(
    x = "Total Active Minutes", # New x-axis title
    y = "Calories", # New y-axis title
    color = "Day Type", # New legend title
    title = "Total Active Minutes Vs Calories Burned" # New title
  )
```

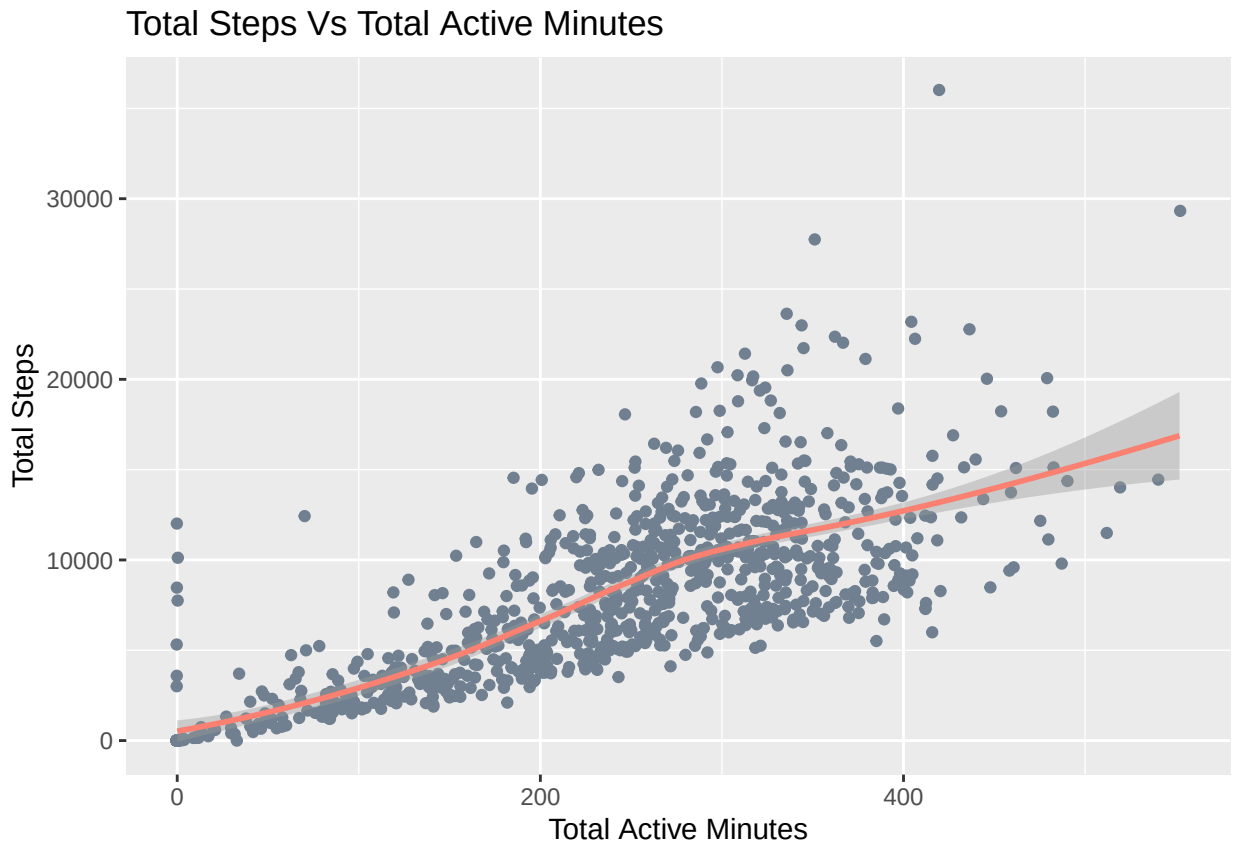
2.2.2.7 Total Active Minutes and Calories



steep rise on weekends

```
ggplot(data=daily_activity_sleep) +
geom_jitter(aes(x=TotalActMin, y=Steps), color='slategray') +
geom_smooth(aes(x=TotalActMin, y=Steps), color='salmon') +
labs(
    x = "Total Active Minutes", # New x-axis title
    y = "Total Steps", # New y-axis title
    title = "Total Steps Vs Total Active Minutes"
  )
```

2.2.2.8 Total Active Minutes and Total Steps



steep rise

```
ggplot(data=daily_activity_sleep,) +
geom_jitter(aes(x=Steps,y=Calories), color='lightslategray') +
geom_smooth(aes(x=Steps,y=Calories), color='salmon') +
  labs(title = "Total Steps vs. Calories Burned")
```

2.2.2.9 Total Steps and Calories

very sharp trend

3 Part 2: Hourly Level Analysis

Moving forward, we will now extende our analysis to a higher level of granularity. The subsequent analysis will focus on the hourly level data for some of the same indicators as above. Since a thorough analysis has already been conducted at the daily level, we will bypass repeating this process for the hourly analysis. Our primary goal for the hourly data will be to review how the distributions of these indicators change throughout the hours of the day.

3.1 Data Processing and Cleaning

This section details the process for data processing and cleaning in preparation for the hourly analysis

3.1.1 Shortlisting Tables and Columns

We have identified a total of three relevant hourly data indicators. We will assess which ones require joining.

- Intensities
- Steps
- Calories

3.1.1.1 Unique ID's in each table Just like before, let's first check the number of Unique ID's in each of the hourly tables. This will help determine any tablet that do not contain information for all tables

The following SQL query checks the number of unique ID's in all 3 tables.

```
SELECT        -- Intensities Table
  'HourlyIntensities' AS DataIndicator, -- Data indicators representing each table
    COUNT(DISTINCT Id) AS UniqueIds  -- Count of Unique IDs
FROM
    hourlyintensities_merged
UNION
SELECT      -- Process Repeated for Steps
    'HourlySteps' AS DataIndicator,
    COUNT(DISTINCT Id) AS UniqueIds
FROM
        hourlysteps_merged
UNION
SELECT             -- Process Repeated for Calories
    'Calories' AS DataIndicator,
    COUNT(DISTINCT Id) AS UniqueIds
FROM
    hourlycalories_merged
ORDER BY        -- Sort by Unique ID's
  UniqueIds
```

Table 25: 3 records

DataIndicator	UniqueIds
Calories	33
HourlyIntensities	33
HourlySteps	33

All 3 tables contain information on all 33 respondents, therefore none need to be dropped

3.1.1.2 Finalize and Validate Remaining Columns

Next, we will assess if any columns in the three tables require dropping or transformation. To begin, let's obtain a brief description of the columns within all tables.

```
PRAGMA table_info(hourlyintensities_merged)
```

Table 26: 4 records

cid	name	type	notnull	dflt_value	pk
0	Id	REAL	0	NA	0
1	ActivityHour	TEXT	0	NA	0
2	TotalIntensity	INTEGER	0	NA	0
3	AverageIntensity	REAL	0	NA	0

```
PRAGMA table_info(hourlysteps_merged)
```

Table 27: 3 records

cid	name	type	notnull	dflt_value	pk
0	Id	REAL	0	NA	0
1	ActivityHour	TEXT	0	NA	0
2	StepTotal	INTEGER	0	NA	0

```
PRAGMA table_info(hourlycalories_merged)
```

Table 28: 3 records

cid	name	type	notnull	dflt_value	pk
0	Id	REAL	0	NA	0
1	ActivityHour	TEXT	0	NA	0
2	Calories	INTEGER	0	NA	0

We can see that Id and ActivityHour are in all 3 tables which can be used to join the tables

All 3 of the remaining columns, TotalIntensity, AverageIntensity,Calories and Steps only exist on one of the tables each. Let's also review their descriptions to see if we should exclude any amongst them. The table below lists descriptions for the relevant columns from the Fitabase data dictionary avaialble at this link:

https://www.fitabase.com/media/2126/fitabase-fitbit-data-dictionary-as-of-05162025.pdf

```
datadescription <- read_excel("datadescription.xlsx", sheet='Hourly')
kable(datadescription) %>%
  kable_styling("striped") %>% column_spec(2, width = "12cm")
```

Data Header	Data Description
ActivityHour	Date and hour value in mm/dd/yyyy hh:mm:ss format.
TotalIntensity	Value calculated by adding all the minute-level intensity values that occurred within the hour
AverageIntensity	Average intensity state exhibited during that hour (TotalIntensity for that ActivityHour divided by 60).
StepTotal	Total number of steps taken.

Calories	Total number of estimated calories burned.

Since AverageIntensity is just a transformation of TotalIntensity, we will drop it from our analysis

3.1.1.3 Checking for Duplicates and Missing Values Next we will check for any duplicates and missing values that exist for our finalized column lists for all 3 tables

```
WITH dup_inten as (     -- Create a temporary table to calculate Duplicates
SELECT
  1 as UniId,     -- Define UniId in each temporary table to facilitate the join
  COUNT(*) -      -- Calculate the difference between the count of total rows
(SELECT           -- and distinct rows. This difference will be the # of duplicates
  count(*)
FROM
(SELECT DISTINCT      -- This inner query counts distinct rows
  Id,
  ActivityHour,
  TotalIntensity
FROM
  hourlyintensities_merged) as t1) as DuplicatesIntensities
FROM
  hourlyintensities_merged
),
miss_inten as ( -- Create a temporary table to calculate Missing Values
SELECT
    COUNT(*) AS MissingIntensities,  -- Count values when any one of the columns
    1 as UniId                       -- specified is null.
FROM
    hourlyintensities_merged
WHERE
    Id IS NULL OR
  ActivityHour IS NULL OR
  TotalIntensity IS NULL
),
dup_steps as (     -- Repeat the process for the steps table
SELECT
  1 as UniId,
  COUNT(*) -
(SELECT
  count(*)
FROM
(SELECT DISTINCT
  Id,
  ActivityHour,
  StepTotal
FROM
  hourlysteps_merged) as t2) as DuplicatesSteps
FROM
  hourlysteps_merged
),
miss_steps as (
SELECT
    COUNT(*) AS MissingSteps,
```

```
    1 as UniId
FROM
  hourlysteps_merged
WHERE
    Id IS NULL OR
  ActivityHour IS NULL OR
  StepTotal IS NULL
),
dup_cals as (     -- Repeat the process for the calories table
SELECT
  1 as UniId,
  COUNT(*) -
(SELECT
  count(*)
FROM
(SELECT DISTINCT
  Id,
  ActivityHour,
  Calories
FROM
  hourlycalories_merged) as t2) as DuplicatesCals
FROM
  hourlycalories_merged
),
miss_cals as (
SELECT
    COUNT(*) AS MissingCals,
    1 as UniId
FROM
  hourlycalories_merged
WHERE
    Id IS NULL OR
  ActivityHour IS NULL OR
  Calories IS NULL
)
SELECT               -- Join results from all tables into one
  A.DuplicatesIntensities,
  B.MissingIntensities,
  C.DuplicatesSteps,
  D.MissingSteps,
  E.DuplicatesCals,
  F.MissingCals
FROM
  dup_inten as A
JOIN
  miss_inten as B on A.UniId=B.UniId
JOIN
  dup_steps as C on A.UniId=C.UniId
JOIN
  miss_steps as D on A.UniId=D.UniId
JOIN
  dup_cals as E on A.UniId=E.UniId
JOIN
```

```
  miss_cals as F on A.UniId=F.UniId
```

Table 30: 1 records

DuplicatesIntensities	MissingIntensities	DuplicatesSteps	MissingSteps	DuplicatesCals	MissingCals
0	0	0	0	0	0

All three tables are clean, with zero missing values and duplicates.

3.1.2 Joining the Daily tables

Next we will join all 3 of the tables on the Id and ActivityHour columns. As discussed earlier, AverageIntensity is not included

```
SELECT
  I.Id,
  I.ActivityHour,
  I.TotalIntensity,
  S.StepTotal,
  C.Calories
FROM hourlyintensities_merged as I
JOIN
    hourlysteps_merged as S
ON
    I.Id = S.Id      -- Join on Id and Activity Hour
AND
    I.ActivityHour = S.ActivityHour
JOIN
    hourlycalories_merged as C
ON
    I.Id = C.Id
AND
    I.ActivityHour = C.ActivityHour
```

3.1.2.1 Review Joined data

Lets now the first few rows of the joined dataframe for some quick validation

```
kable(head(hourly_activity,5)) %>%
  kable_styling("striped")
```

Id	ActivityHour	TotalIntensity	StepTotal	Calories
1503960366	4/12/2016 12:00:00 AM	20	373	81
1503960366	4/12/2016 1:00:00 AM	8	160	61
1503960366	4/12/2016 2:00:00 AM	7	151	59
1503960366	4/12/2016 3:00:00 AM	0	0	47
1503960366	4/12/2016 4:00:00 AM	0	0	48

The data has been successfully and correctly joined. We will now proceed to the Data Visualization phase.

3.2 Data Visualization and Plots

Now that we have our data we can move forward with the Data Visualization phase. As previously stated, the analysis will remain concise, focusing solely on summary statistics and an examination of how the distributions of the three indicators fluctuate throughout the hours of the day.

3.2.1 Summary Statistics

The initial step involves generating summary statistics to facilitate a more comprehensive understanding of the dataset.

```
kable(hourly_activity %>% summary(digits=2)) %>%
  kable_styling("striped")
```

Id	ActivityHour	TotalIntensity	StepTotal	Calories
Min. :1.5e+09	Length:22099	Min. : 0	Min. : 0	Min. : 42
1st Qu.:2.3e+09	Class :character	1st Qu.: 0	1st Qu.: 0	1st Qu.: 63
Median :4.4e+09	Mode :character	Median : 3	Median : 40	Median : 83
Mean :4.8e+09	NA	Mean : 12	Mean : 320	Mean : 97
3rd Qu.:7.0e+09	NA	3rd Qu.: 16	3rd Qu.: 357	3rd Qu.:108
Max. :8.9e+09	NA	Max. :180	Max. :10554	Max. :948

Some quick insights:

- The mean for Total Intensity is 12 minutes. This means that on average, in a given hour, respondents engage in 12 minutes of activity.
- The mean of StepTotal is 320. This means that on average, in a given hour, respondents walk 320 steps.
- The mean of Calories burned is 97. This means that on average, in a given hour, respondents burn 320 97 calories

3.2.2 Variation in Averages at each Hour of the Day

We will now proceed to delve deeper into these statistics by identifying how the average values vary across each hour of the day.

To prepare for this, we must first add two new columns: one representing the hour of the day (in 24-hour format) and another specifying the Day Type (differentiating between Weekend and Weekday)

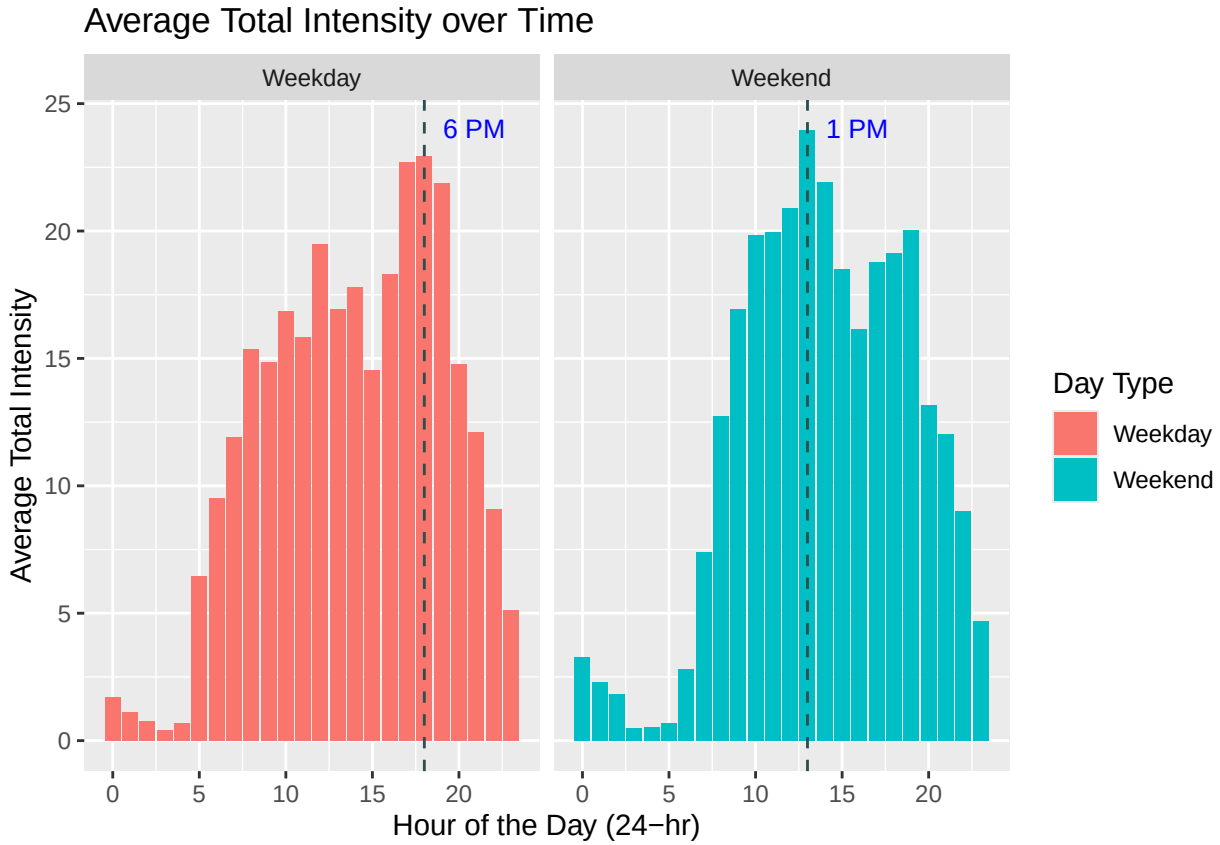
```
hourly_activity <- hourly_activity %>%
  mutate(activity_time = mdy_hms(ActivityHour), ##Cleaned ActivityHour
         hour_of_day = hour(activity_time),      ## Calculate Hour of the day
         daytype= ifelse(weekdays(activity_time) == "Saturday" |
                           weekdays(activity_time) == "Sunday",
                         "Weekend", "Weekday")) ##Calculate DayType
```

3.2.2.1 Average Total Intensities at each Hour of the Day This code computes the average total intensity for each hour of the day and then plots the results against time. Distinct plots are generated to compare weekday and weekend averages. A gray line is drawn at peak hours

```
line_data <- data.frame(        #Data to draw vertical lines at peak times
  daytype = c("Weekday", "Weekend"),   #6 pm on weekedays and 1 pm on weekend
  x_pos = c(18, 13), # Unique positions for each facet
  label = c("6 PM", "1 PM") # The text for each facet
)
```

```r
hourly_activity %>% group_by(hour_of_day, daytype) %>% #For each hour
  summarise(hourly_avg = mean(TotalIntensity), .groups="drop") %>% #Calculate Avg
  ggplot(aes(x=hour_of_day, y=hourly_avg, fill = daytype)) + geom_col() +
  facet_wrap(~daytype) + #Separate plots for Weekday and Weekend
  geom_vline(data = line_data, aes(xintercept = x_pos), # Draw lines
             linetype = "dashed", color = "darkslategray", size = 0.5) +
  geom_text(data = line_data, aes(x = x_pos, y = Inf, label = label), #Add text
        vjust = 2, hjust = -0.3, inherit.aes = FALSE, size=3.5,color='blue') +
  labs(title="Average Total Intensity over Time", # New Title
                              x='Hour of the Day (24-hr)', # New x and y labels
                              y='Average Total Intensity', fill="Day Type")
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once per session.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



Activity levels are observed to increase as the day progresses in both charts before declining later. Distinct variations are evident in the activity distributions for weekends and weekdays. On weekdays, peak activity occurs between 5-7 p.m., while on weekends, the peak is notably at 1 p.m.

This pattern is logical, as individuals on weekdays may engage in exercise following the conclusion of typical office hours at 5 p.m. Conversely, on weekends, they have the flexibility to schedule these workouts earlier, such as at 1 p.m., a time that still permits a later start to the day.

This provides an important insight for Bellabeat: reminders prompting users to engage in activity could be strategically timed differently for weekdays and weekends. On weekdays, reminders such as, “Time to take a

walk," or "Time to visit the gym," could be dispatched at 5 p.m. On weekends, these prompts could be sent at around 12:45 p.m.

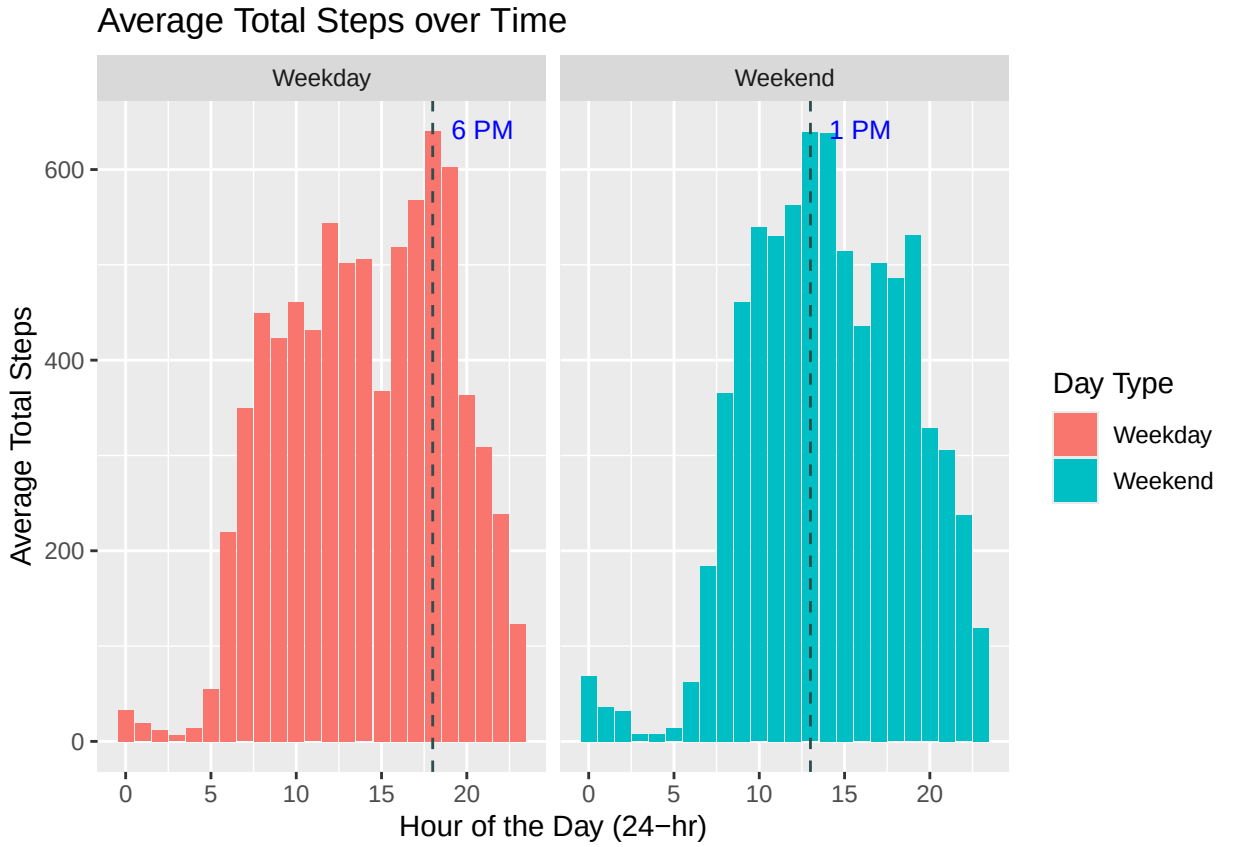
3.2.2.2 Average Calories at each Hour of the Day This code computes the average calories burned for each hour of the day and then plots the results against time. Distinct plots are generated to compare weekday and weekend averages.

```
hourly_activity %>% group_by(hour_of_day, daytype) %>% #For each hour
  summarise(hourly_avg = mean(Calories), .groups="drop") %>% #Calculate Avg
  ggplot(aes(x=hour_of_day, y=hourly_avg, fill = daytype)) + geom_col() +
  facet_wrap(~daytype) + #Separate plots for Weekday and Weekend
  geom_vline(data = line_data, aes(xintercept = x_pos), # Draw lines
             linetype = "dashed", color = "darkslategray", size = 0.5) +
  geom_text(data = line_data, aes(x = x_pos, y = Inf, label = label), #Add text
        vjust = 2, hjust = -0.3, inherit.aes = FALSE, size=3.5,color='blue') +
  labs(title="Average Calories Burned over Time", # New Title
                              x='Hour of the Day (24-hr)', # New x and y labels
                              y='Average Calories Burned', fill="Day Type")
```

A similar variation is observed for Calories burned; the values increase before declining. During weekdays, the peak is observed at 5-7 pm, while on weekends, it occurs at 6 pm. This further reinforces the recommendation to send reminders at 5 pm on weekdays and 1 pm on weekends.

3.2.2.3 Average Total Steps at each Hour of the Day This code computes the average total steps for each hour of the day and then plots the results against time. Distinct plots are generated to compare weekday and weekend averages.

```r
hourly_activity %>% group_by(hour_of_day, daytype) %>% #For each hour
  summarise(hourly_avg = mean(StepTotal), .groups="drop") %>%  #Calculate Avg
  ggplot(aes(x=hour_of_day, y=hourly_avg, fill = daytype)) + geom_col() +
  facet_wrap(~daytype) + #Separate plots for Weekday and Weekend
  geom_vline(data = line_data, aes(xintercept = x_pos), # Draw lines
             linetype = "dashed", color = "darkslategray", size = 0.5) +
  geom_text(data = line_data, aes(x = x_pos, y = Inf, label = label), #Add text
      vjust = 2, hjust = -0.3, inherit.aes = FALSE, size=3.5, color='blue') +
  labs(title="Average Total Steps over Time", # New Title
                              x='Hour of the Day (24-hr)', # New x and y labels
                              y='Average Total Steps', fill="Day Type")
```



The expected trend of a rise followed by a decline is also observed for step counts. Step activity reaches its maximum at 6:00 PM on weekdays and between 1:00 PM and 2:00 PM on weekends. Reminders such as "Time to talk a walk" and "Ready for a Run" are optimally timed for these periods on the respective days of the week.

4 Conclusion and Recommendations

daily- dist and comp: the data suggests that most people only sleep once in a day (presumably at night) calorie burned dist peak is on lower end of recommended range one third have steps below 6000 lightly activ pop very sed lifestyle 16 hours activity potentially drops in summers we have high performers and low performers, mkr strategy should be targeted , varying performance levels. h These disparate distribution patterns strongly indicate that a one-size-fits-all approach is ineffective, necessitating the development of user-targeted plans. 30% walk less than 4 km sleep is fine but bellabeat can help set up sleep schedule

daily: relationships

hourly: times for remoinder 5-7 on weekend, 1-2 on weekdays

```
# Code to close database connection
DBI::dbDisconnect(con)
```