# Bellabeat_Capstone_Project

Baraka Mtana

2024-11-01

## 0.1 Important shortcuts we'll reuse

#### 0.1.0.1 Restart r = "Ctrl + Shift + F10"

#### 0.1.0.2 Add <- = "Alt + i"

#### 0.1.0.3 %>% = "Ctrl + Shift + M"

```
#options(repos = c(CRAN = "https://cloud.r-project.org/"))
# Set up environment
# import libraries
library(tidyverse)
```

#### 0.1.0.4 New r line code = "Ctrl + Alt + I"

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
# install.packages("lubridate")
library(lubridate)
library(ggplot2)
library(dplyr)
```

```
# get and print working directory
currentDir <- getwd()
print(currentDir)
```

```
## [1] "C:/Users/LENOVO/Desktop/Bellabeat_Capstone-project"
```

```
# list file in the working DIR
list.files(currentDir)
```

```
##  [1] "Bellabeat_Capstone-project.R"
##  [2] "Bellabeat_Capstone-project.Rproj"
##  [3] "BellabeatCapstoneProject.log"
##  [4] "BellabeatCapstoneProject.Rmd"
```

```
##  [5] "BellabeatCapstoneProject_files"
##  [6] "customer_averages.csv"
##  [7] "Functions.R"
##  [8] "LICENSE"
##  [9] "mturkfitbit_export_3.12.16-4.11.16"
## [10] "README.md"
## [11] "Screenshot.png"
## [12] "UYTzmqN3SS-WCbei0buGkw_cbefaed7e1354ffb8e992c8e198906f1_Case-Study-2_-How-can-a-wellness-techno
```

```r
# we are interested in the csv files in 'mturkfitbit_export_3.12.16-4.11.16'
# and 'Fitabase Data 3.12.16-4.11.16'
csv_files_Dir <- file.path(
  currentDir, 'mturkfitbit_export_3.12.16-4.11.16', 'Fitabase Data 3.12.16-4.11.16'
)

csv_files <- list.files(csv_files_Dir)
len_cvs = length(csv_files)

# csv_files <- list.files(csv_files_Dir, pattern = "\\.csv$", full.names = TRUE)
# print(csv_files)
```

```r
# Here we'll write function we'll reuse

# Check for missing values
missing_value <- function(df, file){

  # Initialize an empty list that will be a key value pair
  # It will hold df_name with missing values and the specific column with the missing values
  df_with_missing_values <- list()

  if (sum(is.na(df)) == 0){

    print(paste(file, "Has no missing values"))

  }else if (sum(is.na(df)) > 0){


    #append df_name with missing vales to df_with_missing_values
    df_with_missing_values[[file]] <- c() # initialize an empty vector to store
     #column names


    #get df columns with missing values
    colNames = colnames(df)

    for (col in colNames){
      if (sum(is.na(df[[col]]))  > 0){

        df_with_missing_values[[file]] <- c(df_with_missing_values[[file]], col)

        print(paste("Total missing values in", file, col, "is",
                    sum(is.na(df[col]))))
      }
    }
```

```
  }

  return(df_with_missing_values)
}



# get number of unique values
unique_value <- function(df, column_of_interest) {
  #get the unique values
  uniques <- unique(df[[column_of_interest]])

  # Get the unique values from the specified column
  n_unique <- length(uniques)

  print(paste(column_of_interest, "has",  n_unique, "values"))

}


# Let's check if we are dealing with the same clients
uniqueIDs_Comparison <- function (df1, column1, df2, column2){
  missing_ids = c()

  initial_list = unique(df1[[column1]])
  comparison_list = unique(df2[[column2]])


  for (i in initial_list){
    if (!(i %in% comparison_list)){
      missing_ids <- c(missing_ids, i)

    }
  }

  for (i in comparison_list){
    if (!(i %in% initial_list)){
      missing_ids <- c(missing_ids, i)

    }

  }

  if (length(missing_ids) == 0){
    print("Dataframes have similar values")


  }else{
    print("Dataframes have different values")
    print(paste("No. of missing values", length(unique(missing_ids))))
  }

  return(unique(missing_ids))
```

```
}


# Get the length of values in a column
N_unique_char <- function(df, column_of_interest){
  # Initialize/ pre-allocate a numeric vector of a specific length, initialized with   zeros
  n_char <- numeric(nrow(df))

  for (i in seq_along(df[[column_of_interest]])){
    n_char[i] <- nchar(df[[column_of_interest]][i])
  }

  # get the number of unique characters
  character_lens <- unique(n_char)

  return(character_lens)
}



change_to_dateTime <- function(df, column_of_interest){

  Values_Ncha_Column = N_unique_char(df, column_of_interest)

  #Check if all elements in the vector Values_Ncha_Column are either 8 or 9
  if (all(Values_Ncha_Column %in% c(8, 9))){
    df[[column_of_interest]] <-
      as.POSIXct(
        df[[column_of_interest]],
        format="%m/%d/%Y",
        tz=Sys.timezone()
    )

  }else if (!all(Values_Ncha_Column
                 %in% c(8, 9))){

    df[[column_of_interest]] <-
      as.POSIXct(
        df[[column_of_interest]],
        # %I: Hour (01-12, for 12-hour clock)
        # %p: AM/PM marker
        format = "%m/%d/%Y %I:%M:%S %p",
        tz=Sys.timezone()
    )
  }

  # confirm the datatype of the column of interest
  #print(paste(column_of_interest, "has" ,class(df[[column_of_interest]]),
              #"datatype"))

  print(head(df[column_of_interest], 5))
  print(tail(df[column_of_interest], 5))
```

```
  return(df)

}



# Check if the columns are identical
identical_columns <- function(df, col1, col2){
  if (identical(df[[col1]], df[[col2]])) {
    print("The columns are identical.")
  } else {
    print("The columns are NOT identical.")
  }
}



# Merge fucntion which we'll use with the reduce inbuilt function
merge_dfs <- function(df1, df2){
  merged_df <- merge(df1, df2, by = "Id")

  return(merged_df)
}

# calculate column averages
Avgs <- function(df, column_names_vector){

  # initialize and empty list
  column_Avgs = list()

  for (i in column_names_vector){
    # get the data of the column
    column_data <- df[, c("Id", i)]


    avg_per_athlete <- column_data %>%
      group_by(Id) %>%
      summarise(
        # use "get" to retrieves the variable from key which is i and
        # Rename the column_Avgs keys to avoid conflicts
        !!paste0("Avg_", i) := mean(get(i), na.rm = TRUE)
      )


    column_Avgs[[i]] = avg_per_athlete


  }

  # Merge DataFrames on the 'ID' column
  # How the reduce function works
  # 1.Reduce will take the first two data frames from column_Avgs and
  #pass them to merge_dfs.
```

```r
  # 2. The result of the first merge will be passed as the first argument
  # to merge_dfs for the next data frame in the list, continuing this process until
  #all data frames have been merged.

  # Here the reduce function and
  merged_df <- Reduce(merge_dfs, column_Avgs)

  return(merged_df)
}


# line plot function
line_plot <- function(df, x_col, y_col, i = NULL) {
  # Determine the title based on whether 'i' is provided
  plot_title <- if (!is.null(i)) {
    paste(i, x_col, "and", y_col, "relationship")
  } else {
    paste(x_col, "and", y_col, "relationship")
  }

  # Create the plot
  plot <- ggplot(data = df) +
    geom_line(mapping = aes(x = .data[[x_col]], y = .data[[y_col]])) +
    labs(title = plot_title)

  # Print the plot
  print(plot)
}



weekDate_grouping <- function(df, weekOrdate){
  # get group by the 1st 2 columns which will be "Id and column with POSIXct
  #datatype"
  grouping_Cols <- c(colnames(df)[1:2])
  summarise_Cols <- c(colnames(df)[-1:-2])

  # Check if the first column is "Id" and the second column is of type POSIXct
  if (grouping_Cols[1] == "Id" && inherits(df[[grouping_Cols[2]]], "POSIXct")){


    # Determine grouping based on the 'weekOrdate' parameter
    if(weekOrdate == "week"){
      # Group by week number
      df <- df %>%
        mutate(Group = format(.data[[grouping_Cols[2]]], "%Y-%U"))
    }else if(weekOrdate == "date"){
      df <- df %>%
        mutate(Group = as.Date(.data[[grouping_Cols[2]]]))
    }else {
      stop("Invalid string value for 'weekOrdate'. Use 'week' or 'date'.")
    }
```

```
    # If the above run successfully perform grouping and summarization
    df <- df %>%
      group_by(.data[[grouping_Cols[1]]], Group) %>%
      summarize(
        across(
          all_of(summarise_Cols),
          list(
            Total = ~sum(.x, na.rm = TRUE),
            Avg = ~mean(.x, na.rm = TRUE)
          ),
          #{col} acts as a Placeholder for the name of the original column being
          #processed
          #{fn}: Placeholder for the name of the function applied (e.g., "Total"
          #or "Avg").
          .names = "{col}_{fn}"
        ),
        .groups = "drop"
      )

  }else{
    stop("Please check the name and datatype of the 1st 2 columns.")
  }

  return(df)
}


random_plots <- function(df, fraction_of_all_len_df, X_col, Y_col){

  IDS <- c(unique(df[["Id"]]))

  num_samples <- ceiling(length(IDS) * fraction_of_all_len_df)
  random_Ids <- sample(IDS, size = num_samples, replace = FALSE)

  for (i in random_Ids){
    filtered_df <- df %>%
    filter(Id == i)
    line_plot(filtered_df,
              X_col, Y_col, i
    )
  }
}
```

```
# Initialize an empty list to store data frames
dfs <- list()


for (file in csv_files) {

  print(paste("Working on", file))

  # create df names
  # split the file name str character
  df_name <- strsplit(file, split = '\\.')[[1]] #Access the first and only string
```

```
  # get the first part of the string character which is basically the name
  # without the csv extension
  df_name <- df_name[1]

  # concatenate the df_name with df
  df_name <- paste0(df_name, "_df") # Use paste0 for no space between parts

  # create full path for each file so that we can import them
  filepath <- file.path(csv_files_Dir, file)

  # read csv
  df <- read.csv(filepath)

  #Check if df has missing value
  missing_value(df = df, file = file)

  # append dfs and their names
  dfs[[df_name]] <- df  # Store the data frame in the list, keyed by file path

}
```

```
## [1] "Working on dailyActivity_merged.csv"
## [1] "dailyActivity_merged.csv Has no missing values"
## [1] "Working on heartrate_seconds_merged.csv"
## [1] "heartrate_seconds_merged.csv Has no missing values"
## [1] "Working on hourlyCalories_merged.csv"
## [1] "hourlyCalories_merged.csv Has no missing values"
## [1] "Working on hourlyIntensities_merged.csv"
## [1] "hourlyIntensities_merged.csv Has no missing values"
## [1] "Working on hourlySteps_merged.csv"
## [1] "hourlySteps_merged.csv Has no missing values"
## [1] "Working on minuteCaloriesNarrow_merged.csv"
## [1] "minuteCaloriesNarrow_merged.csv Has no missing values"
## [1] "Working on minuteIntensitiesNarrow_merged.csv"
## [1] "minuteIntensitiesNarrow_merged.csv Has no missing values"
## [1] "Working on minuteMETsNarrow_merged.csv"
## [1] "minuteMETsNarrow_merged.csv Has no missing values"
## [1] "Working on minuteSleep_merged.csv"
## [1] "minuteSleep_merged.csv Has no missing values"
## [1] "Working on minuteStepsNarrow_merged.csv"
## [1] "minuteStepsNarrow_merged.csv Has no missing values"
## [1] "Working on weightLogInfo_merged.csv"
## [1] "Total missing values in weightLogInfo_merged.csv Fat is 31"
```

```
# Confirm that all df were read successfully
if (len_cvs == length(dfs)) {
  print("All files read successfully")
} else {
  print("Some files were not read correctly")
}
```

```
## [1] "All files read successfully"
```

```
print(names(dfs))
```

```
##  [1] "dailyActivity_merged_df"           "heartrate_seconds_merged_df"
##  [3] "hourlyCalories_merged_df"          "hourlyIntensities_merged_df"
##  [5] "hourlySteps_merged_df"             "minuteCaloriesNarrow_merged_df"
##  [7] "minuteIntensitiesNarrow_merged_df" "minuteMETsNarrow_merged_df"
##  [9] "minuteSleep_merged_df"             "minuteStepsNarrow_merged_df"
## [11] "weightLogInfo_merged_df"
```

```
dailyActivity_merged_df <-dfs[["dailyActivity_merged_df"]]
str(dailyActivity_merged_df)
```

```
## 'data.frame':    457 obs. of  15 variables:
##  $ Id                       : num  1.5e+09 1.5e+09 1.5e+09 1.5e+09 1.5e+09 ...
##  $ ActivityDate             : chr  "3/25/2016" "3/26/2016" "3/27/2016" "3/28/2016" ...
##  $ TotalSteps               : int  11004 17609 12736 13231 12041 10970 12256 12262 11248 10016 ...
##  $ TotalDistance            : num  7.11 11.55 8.53 8.93 7.85 ...
##  $ TrackerDistance          : num  7.11 11.55 8.53 8.93 7.85 ...
##  $ LoggedActivitiesDistance: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ VeryActiveDistance       : num  2.57 6.92 4.66 3.19 2.16 ...
##  $ ModeratelyActiveDistance: num  0.46 0.73 0.16 0.79 1.09 ...
##  $ LightActiveDistance      : num  4.07 3.91 3.71 4.95 4.61 ...
##  $ SedentaryActiveDistance : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ VeryActiveMinutes        : int  33 89 56 39 28 30 33 47 40 15 ...
##  $ FairlyActiveMinutes      : int  12 17 5 20 28 13 12 21 11 30 ...
##  $ LightlyActiveMinutes     : int  205 274 268 224 243 223 239 200 244 314 ...
##  $ SedentaryMinutes         : int  804 588 605 1080 763 1174 820 866 636 655 ...
##  $ Calories                 : int  1819 2154 1944 1932 1886 1820 1889 1868 1843 1850 ...
```

```
# we'll have a look at all unique value in if non of them is repeated
# change ActivityDate from character to datetime
# See if TotalDistance and TrackerDistance distance record the same data
# Calculate average TotalDistance, TrackerDistance, VeryActiveDistance,
# ModeratelyActiveDistance, LightActiveDistance, SedentaryActiveDistance,
# Average calories lost per day
```

check how many customers are we dealing with

```
# call function on dailyActivity_merged_df
unique_value(dailyActivity_merged_df, "Id")
```

```
## [1] "Id has 35 values"
```

See the number of character in each character of the ActivityDate column since we've already seen there are inconsistencies in the ActivityDate, –> let's see if there are some date characters saved with hrs,min,and secs

```
# see if there is uniformity in the Activity date column
N_unique_char(dailyActivity_merged_df, "ActivityDate")
```

```
## [1] 9 8
```

```
# call change_to_date on dailyActivity_merged_df
dailyActivity_merged_df <- change_to_dateTime(dailyActivity_merged_df,
                                              "ActivityDate")
```

```
##   ActivityDate
## 1   2016-03-25
## 2   2016-03-26
## 3   2016-03-27
```

```
## 4   2016-03-28
## 5   2016-03-29
##     ActivityDate
## 453   2016-04-08
## 454   2016-04-09
## 455   2016-04-10
## 456   2016-04-11
## 457   2016-04-12
```

```
# call identical_columns on dailyActivity_merged_df
identical_columns(dailyActivity_merged_df, "TotalDistance", "TrackerDistance" )
```

```
## [1] "The columns are NOT identical."
```

```
error_df <-  dailyActivity_merged_df[, c('Id', 'TotalDistance',
                                         'TrackerDistance')]
Id_s <- c()
difference <- c()
total_distance <- c()

for (i in 1:nrow(error_df)){

  Id_i <-  error_df[["Id"]][i]
  total_dist_i <- error_df[["TotalDistance"]][i]
  tracker_dist_i <- error_df[["TrackerDistance"]][i]


  error <-total_dist_i - tracker_dist_i
  if(error != 0){
    Id_s <- c(Id_s, Id_i)
    difference <- c(difference, error)
    total_distance <- c(total_distance, total_dist_i)

  }

}

# see how many of all the athletes have faulty devices
print(length(unique(Id_s)))
```

```
## [1] 5
```

```
# If 5 out of all athletes are the only ones with faulty devices
# see what percentage of Bellabeat devices are faulty
Number_of_clients <- length(unique(dailyActivity_merged_df[['Id']]))
Client_with_faulty_devices <- length(unique(Id_s))


print(
  paste(
    "Percentage of devices with errors",
            round((Client_with_faulty_devices/Number_of_clients) * 100, 2)
  )
)
```

```
## [1] "Percentage of devices with errors 14.29"
```

```r
# calculate error percentage
percentage_error <- (sum(abs(difference))/ sum(total_distance)) *100
print(
  paste(
    "Faulty Devices have a percentahe error of",
    round(percentage_error, 2)
  )
)
```

```
## [1] "Faulty Devices have a percentahe error of 20.92"
```

#### 0.1.0.5 with a 14.29 percent of Bellabeat devices not giving accurate data, this is significantly high which show devices should be improves,

```r
# Call the Avgs function to get the averages of some columns in dailyActivity_merged_df

column_names <- c(names(dailyActivity_merged_df)[3:10], tail(names(dailyActivity_merged_df), 1))

averages_df = Avgs(dailyActivity_merged_df, column_names)

# Write the data frame to a CSV file
write.csv(averages_df, file = "customer_averages.csv", row.names = FALSE)
```

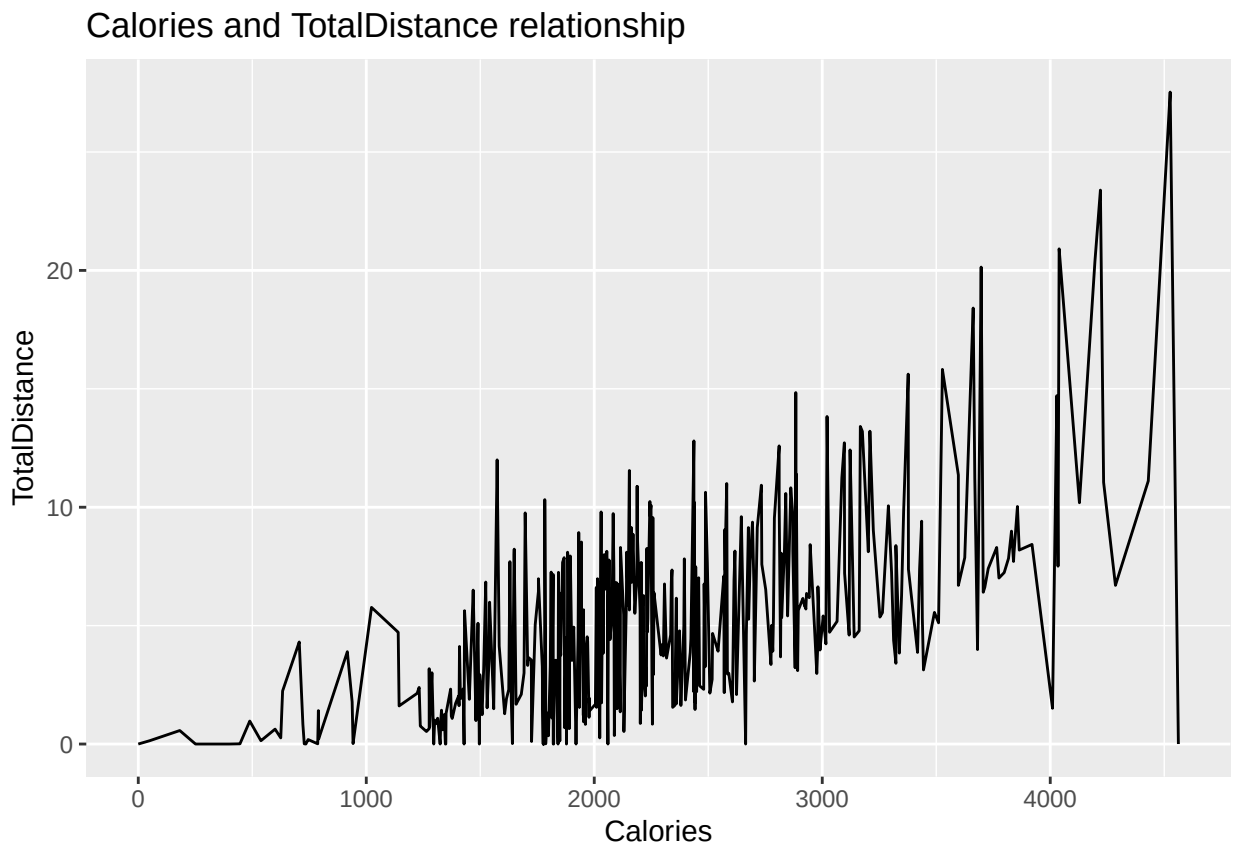
```r
head(averages_df, 5)
```

#### 0.1.0.6 Faulty devices have a 20% error which is rather too high to work with this shows that tracker should be impriove to atleast reduce this figure to a negligible one atleast 5% since achieving 100% accuracy maynot be achieveable.

```
##           Id Avg_TotalSteps Avg_TotalDistance Avg_TrackerDistance
## 1 1503960366      11640.526          7.607368           7.607368
## 2 1624580081       4226.263          2.746842           2.746842
## 3 1644430081       9274.800          6.749000           6.749000
## 4 1844505072       3640.583          2.406667           2.406667
## 5 1927972279       2180.833          1.510833           1.510833
##   Avg_LoggedActivitiesDistance Avg_VeryActiveDistance
## 1                            0             2.78368420
## 2                            0             0.05157895
## 3                            0             1.13399998
## 4                            0             0.05083333
## 5                            0             0.00000000
##   Avg_ModeratelyActiveDistance Avg_LightActiveDistance
## 1                   0.63526316                4.178421
## 2                   0.02736842                2.659474
## 3                   2.03000001                3.580000
## 4                   0.03083333                2.324167
## 5                   0.06166667                1.449167
##   Avg_SedentaryActiveDistance Avg_Calories
## 1                0.000000000     1796.211
## 2                0.005263158     1352.895
## 3                0.008000000     2916.400
## 4                0.000000000     1615.917
## 5                0.000000000     2254.000
```

Check the correlation between TotalDistance and Calories

```
# call function on dailyActivity_merged_df
line_plot(df = dailyActivity_merged_df, x_col= "Calories", y_col = "TotalDistance")
```




There is a general positive correlation between Calories and TotalDistance

## 0.2 Let's get to see heartrate_seconds_merged_df

```
heartrate_seconds_merged_df <- dfs[["heartrate_seconds_merged_df"]]
str(heartrate_seconds_merged_df)
```

```
## 'data.frame':    1154681 obs. of  3 variables:
##  $ Id   : num  2.02e+09 2.02e+09 2.02e+09 2.02e+09 2.02e+09 ...
##  $ Time : chr  "4/1/2016 7:54:00 AM" "4/1/2016 7:54:05 AM" "4/1/2016 7:54:10 AM" "4/1/2016 7:54:15 AM
##  $ Value: int  93 91 96 98 100 101 104 105 102 106 ...
```

```
head(heartrate_seconds_merged_df, 5)
```

```
##           Id                Time Value
## 1 2022484408 4/1/2016 7:54:00 AM    93
## 2 2022484408 4/1/2016 7:54:05 AM    91
## 3 2022484408 4/1/2016 7:54:10 AM    96
## 4 2022484408 4/1/2016 7:54:15 AM    98
## 5 2022484408 4/1/2016 7:54:20 AM   100
```

```
tail(heartrate_seconds_merged_df, 5)
```

```
##                 Id                 Time Value
```

```
## 1154677 8877689391 4/12/2016 9:46:25 AM   113
## 1154678 8877689391 4/12/2016 9:46:30 AM   108
## 1154679 8877689391 4/12/2016 9:46:35 AM   102
## 1154680 8877689391 4/12/2016 9:46:40 AM    99
## 1154681 8877689391 4/12/2016 9:46:45 AM    98
```

```
# Check if the unique Ids are similar to the ones in dailyActivity_merged_df
# change time column to datetime
# see the average heart rate per unique id, see the correlation between the average heart rate and calo
# see if there is any negative hr values
```

Confirm if we are dealing with the same athletes. Check if the unique Ids are similar to the ones in dailyActivity_merged_df

```
IDs <- uniqueIDs_Comparison(df1 = dailyActivity_merged_df, column1 = "Id", df2 = heartrate_seconds_merg
```

```
## [1] "Dataframes have different values"
## [1] "No. of missing values 21"
```

These means we we are only dealing with 14 clients in both dailyActivity_merged_df, and heartrate_seconds_merged_df dataframes.

```
print(IDs)
```

```
##  [1] 1503960366 1624580081 1644430081 1844505072 1927972279 2320127002
##  [7] 2873212765 2891001357 3372868164 3977333714 4057192912 4319703577
## [13] 4388161847 4445114986 4702921684 6290855005 7086361926 8053475328
## [19] 8253242879 8378563200 8583815059
```

Change the time columns to datetime

```
heartrate_seconds_merged_df <- change_to_dateTime(heartrate_seconds_merged_df, "Time")
```

```
##                  Time
## 1 2016-04-01 07:54:00
## 2 2016-04-01 07:54:05
## 3 2016-04-01 07:54:10
## 4 2016-04-01 07:54:15
## 5 2016-04-01 07:54:20
##                         Time
## 1154677 2016-04-12 09:46:25
## 1154678 2016-04-12 09:46:30
## 1154679 2016-04-12 09:46:35
## 1154680 2016-04-12 09:46:40
## 1154681 2016-04-12 09:46:45
```

see the average heart rate per unique id, see the correlation between the average heart rate and calories lost

```
# calculate averages
average_hr_per_athlete <- Avgs(heartrate_seconds_merged_df, c(names(heartrate_seconds_merged_df)[3]))


new_col_name = names(average_hr_per_athlete)[2]

print(average_hr_per_athlete)
```

```
## # A tibble: 14 x 2
##            Id Avg_Value
##         <dbl>     <dbl>
```

```
##  1 2022484408      81.7
##  2 2026352035      65.8
##  3 2347167796      76.1
##  4 4020332650      81.9
##  5 4558609924      79.6
##  6 5553957443      69.4
##  7 5577150313      65.0
##  8 6117666160      83.5
##  9 6391747486      84.1
## 10 6775888955      97.7
## 11 6962181067      80.7
## 12 7007744171      89.9
## 13 8792009665      76.3
## 14 8877689391      87.6
```

```r
print(max(average_hr_per_athlete[[new_col_name]]))
```

```
## [1] 97.73704
```

```r
print(min(average_hr_per_athlete[[new_col_name]]))
```

```
## [1] 65.00254
```

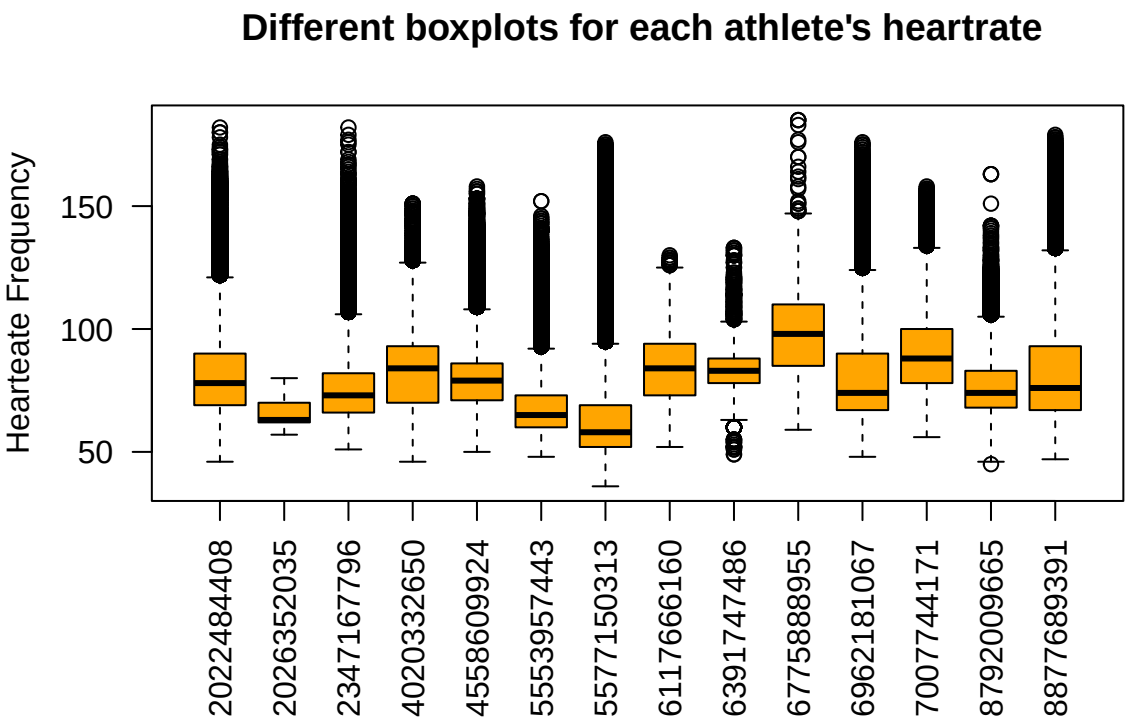
```r
print(median(ceiling(average_hr_per_athlete[[new_col_name]]), na.rm = TRUE))
```

```
## [1] 81.5
```

```r
(mean(ceiling(average_hr_per_athlete[[new_col_name]])))
```

```
## [1] 80.42857
```

```r
# Increase bottom margin to fit x-axis labels(Clients/Athletes) better
par(mar = c(8, 5, 4, 2) + 0.1)


boxplot(Value ~ Id,
        data = heartrate_seconds_merged_df,
        main =" Different boxplots for each athlete's heartrate",
        xlab = " ",
        ylab = "Hearteate Frequency",
        col = "orange",
        #main = "Perpendicular",
        las = 2
        #horizontal = TRUE
        #border="brown"
)
```




From the above plot we can see that we have huge outliers with some of the athletes having heart rate close to 200 and some close to almost 0

```
hourlyCalories_merged_df <- dfs[['hourlyCalories_merged_df']]
str(hourlyCalories_merged_df)
```

#### 0.2.0.1 Here we'll analyse hourlyCalories_merged_df

```
## 'data.frame':    24084 obs. of  3 variables:
##  $ Id          : num  1.5e+09 1.5e+09 1.5e+09 1.5e+09 1.5e+09 ...
##  $ ActivityHour: chr  "3/12/2016 12:00:00 AM" "3/12/2016 1:00:00 AM" "3/12/2016 2:00:00 AM" "3/12/20
##  $ Calories    : int  48 48 48 48 48 48 48 48 48 49 ...
```

```
# Check if the unique Ids are similar to the ones in heartrate_seconds_merged_df
# see average calories lost each hour.
# Check which our are people most active
# change ActivityHour to datetime
```

```
# Check if the unique Ids are similar to the ones in dailyActivity_merged_df
uniqueIDs_Comparison(dailyActivity_merged_df, "Id",  hourlyCalories_merged_df, "Id")
```

```
## [1] "Dataframes have different values"
## [1] "No. of missing values 1"
```

```
## [1] 4388161847
```

```
# Change to datetime
hourlyCalories_merged_df <-  change_to_dateTime(hourlyCalories_merged_df, "ActivityHour")
```

```
##          ActivityHour
## 1 2016-03-12 00:00:00
## 2 2016-03-12 01:00:00
## 3 2016-03-12 02:00:00
## 4 2016-03-12 03:00:00
## 5 2016-03-12 04:00:00
##              ActivityHour
## 24080 2016-04-12 04:00:00
## 24081 2016-04-12 05:00:00
## 24082 2016-04-12 06:00:00
## 24083 2016-04-12 07:00:00
## 24084 2016-04-12 08:00:00
```

```
hourlyIntensities_merged_df <- dfs[["hourlyIntensities_merged_df"]]
str(hourlyIntensities_merged_df)
```

```
## 'data.frame':    24084 obs. of  4 variables:
##  $ Id              : num  1.5e+09 1.5e+09 1.5e+09 1.5e+09 1.5e+09 ...
##  $ ActivityHour    : chr  "3/12/2016 12:00:00 AM" "3/12/2016 1:00:00 AM" "3/12/2016 2:00:00 AM" "3/1
##  $ TotalIntensity  : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ AverageIntensity: num  0 0 0 0 0 ...
```

```
# Check if the unique Ids are similar to the ones in dailyActivity_merged_df
# and hourlyCalories_merged_df and hourlyCalories_merged_df since they have similar number row
# merge with hourlyCalories_merged_df see intensity and calories correlation on the same plot
# see AverageIntensity and Calories corelaion
```

```
# See head and tail
head(hourlyIntensities_merged_df, 5)
```

```
##           Id          ActivityHour TotalIntensity AverageIntensity
## 1 1503960366 3/12/2016 12:00:00 AM              0                 0
## 2 1503960366  3/12/2016 1:00:00 AM              0                 0
## 3 1503960366  3/12/2016 2:00:00 AM              0                 0
## 4 1503960366  3/12/2016 3:00:00 AM              0                 0
## 5 1503960366  3/12/2016 4:00:00 AM              0                 0
```

```
#print(tail(hourlyIntensities_merged_df), 5)
```

```
column1 = "Id"
column2 = "Id"
uniqueIDs_Comparison(hourlyCalories_merged_df, column1,
                     hourlyIntensities_merged_df,
                     column2)
```

```
## [1] "Dataframes have similar values"
```

```
## NULL
```

```
hourlyIntensities_merged_df <- change_to_dateTime(hourlyIntensities_merged_df, "ActivityHour")
```

```
##          ActivityHour
## 1 2016-03-12 00:00:00
## 2 2016-03-12 01:00:00
## 3 2016-03-12 02:00:00
## 4 2016-03-12 03:00:00
## 5 2016-03-12 04:00:00
##              ActivityHour
```

```
## 24080 2016-04-12 04:00:00
## 24081 2016-04-12 05:00:00
## 24082 2016-04-12 06:00:00
## 24083 2016-04-12 07:00:00
## 24084 2016-04-12 08:00:00
```

```
# confirm that the ActivityHour in both hourlyCalories_merged_df and hourlyIntensities_merged_df are si
column1 = "ActivityHour"
column2 = "ActivityHour"
non_unifom_dates =uniqueIDs_Comparison(hourlyCalories_merged_df, column1,
                    hourlyIntensities_merged_df,column2)
```

```
## [1] "Dataframes have similar values"
```

```
head_values <-  (head(non_unifom_dates, 5))
tail_values <- (tail(non_unifom_dates, 5))

print(head_values)
```

```
## NULL
```

```
print(tail_values)
```

```
## NULL
```

```
# Join dataframes
CaloriesIntensities_df <- hourlyCalories_merged_df%>%inner_join(hourlyIntensities_merged_df,
                            by=c('Id','ActivityHour'))


head(CaloriesIntensities_df, 5)
```

```
##           Id        ActivityHour Calories TotalIntensity AverageIntensity
## 1 1503960366 2016-03-12 00:00:00       48              0                0
## 2 1503960366 2016-03-12 01:00:00       48              0                0
## 3 1503960366 2016-03-12 02:00:00       48              0                0
## 4 1503960366 2016-03-12 03:00:00       48              0                0
## 5 1503960366 2016-03-12 04:00:00       48              0                0
```

```
if(nrow(hourlyCalories_merged_df) == nrow(CaloriesIntensities_df) &&
   (nrow(hourlyCalories_merged_df) == nrow(hourlyIntensities_merged_df))){

  print("Dataframes have same number of rows")

}
```

```
## [1] "Dataframes have same number of rows"
```

```
hourlySteps_merged_df <- dfs[["hourlySteps_merged_df"]]
str(hourlySteps_merged_df)
```

```
## 'data.frame':    24084 obs. of  3 variables:
##  $ Id          : num  1.5e+09 1.5e+09 1.5e+09 1.5e+09 1.5e+09 ...
##  $ ActivityHour: chr  "3/12/2016 12:00:00 AM" "3/12/2016 1:00:00 AM" "3/12/2016 2:00:00 AM" "3/12/20
##  $ StepTotal   : int  0 0 0 0 0 0 0 0 0 8 ...
```

```
# Check if the unique Ids are similar to the ones in hourlyCalories_merged_df
# merge with  hourlyCalories_merged_df
# see the at what time are people most active
```

```
# see correlation between steps and calories
```

```
hourlySteps_merged_df <- change_to_dateTime(hourlySteps_merged_df, "ActivityHour")
```

```
##          ActivityHour
## 1 2016-03-12 00:00:00
## 2 2016-03-12 01:00:00
## 3 2016-03-12 02:00:00
## 4 2016-03-12 03:00:00
## 5 2016-03-12 04:00:00
##              ActivityHour
## 24080 2016-04-12 04:00:00
## 24081 2016-04-12 05:00:00
## 24082 2016-04-12 06:00:00
## 24083 2016-04-12 07:00:00
## 24084 2016-04-12 08:00:00
```

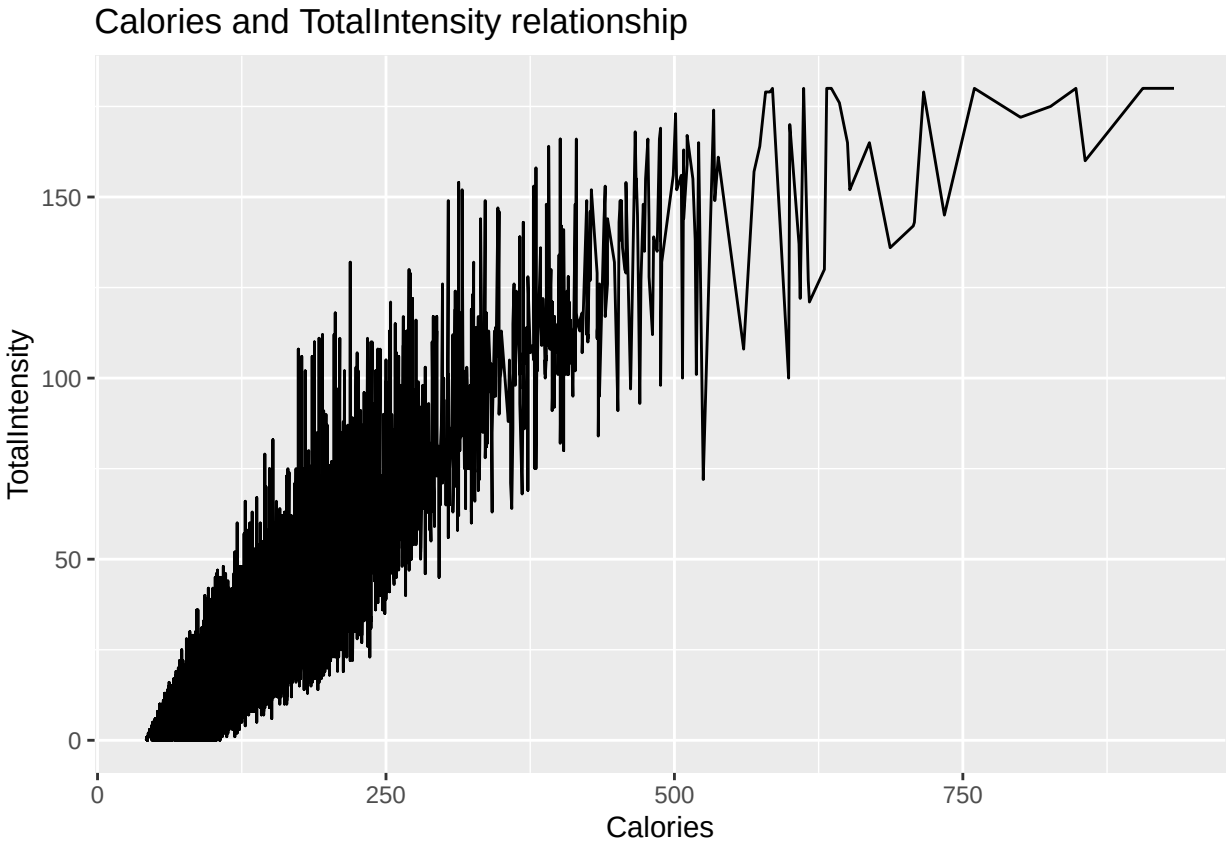
```
CaloriesIntensitiesSteps_df <- CaloriesIntensities_df%>%inner_join(
  hourlySteps_merged_df, by=c('Id', 'ActivityHour'))

head(CaloriesIntensitiesSteps_df, 5)
```

```
##           Id        ActivityHour Calories TotalIntensity AverageIntensity
## 1 1503960366 2016-03-12 00:00:00       48              0                0
## 2 1503960366 2016-03-12 01:00:00       48              0                0
## 3 1503960366 2016-03-12 02:00:00       48              0                0
## 4 1503960366 2016-03-12 03:00:00       48              0                0
## 5 1503960366 2016-03-12 04:00:00       48              0                0
##   StepTotal
## 1         0
## 2         0
## 3         0
## 4         0
## 5         0
```

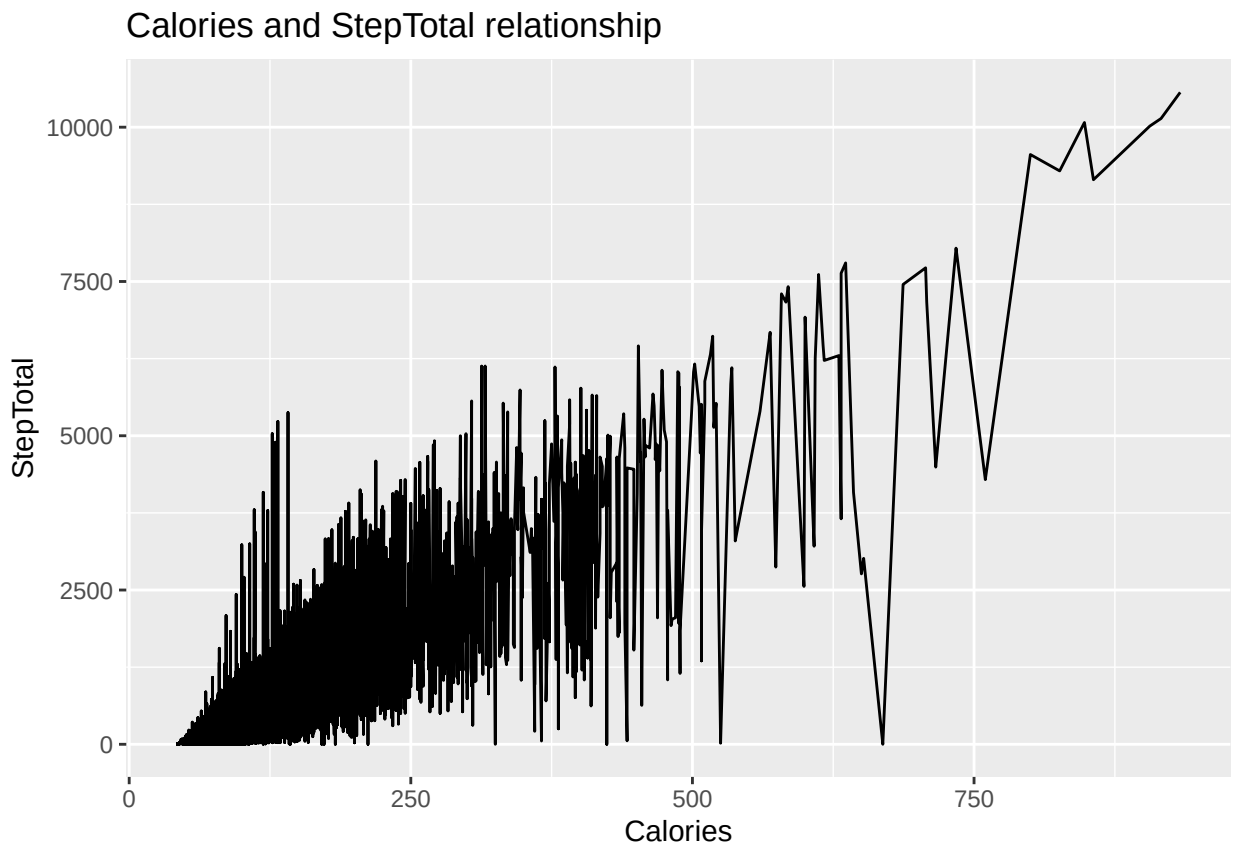
```
if(nrow(CaloriesIntensitiesSteps_df) == nrow(CaloriesIntensities_df) &&
   (nrow(CaloriesIntensitiesSteps_df) == nrow(hourlySteps_merged_df))){

  print("Dataframes have same number of rows")

}
```

```
## [1] "Dataframes have same number of rows"
```

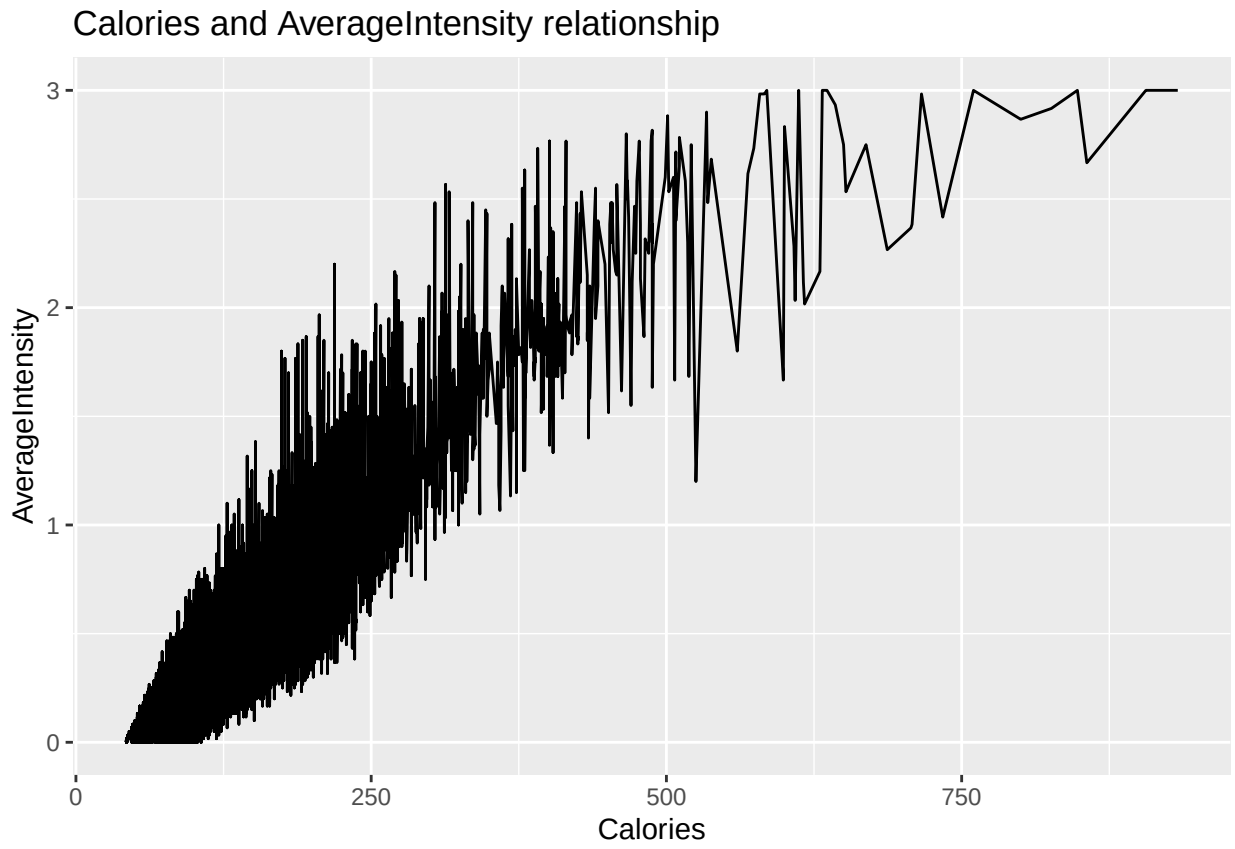
```
# plot a line plot see see the correlation between calories, intensity and steps
line_plot(CaloriesIntensitiesSteps_df, "Calories", "TotalIntensity")
```




```
line_plot(CaloriesIntensitiesSteps_df, "Calories", "StepTotal")
```




```
line_plot(CaloriesIntensitiesSteps_df, "Calories", "AverageIntensity")
```




```
colnames(CaloriesIntensitiesSteps_df)
```

```
## [1] "Id"               "ActivityHour"     "Calories"         "TotalIntensity"
## [5] "AverageIntensity" "StepTotal"
```

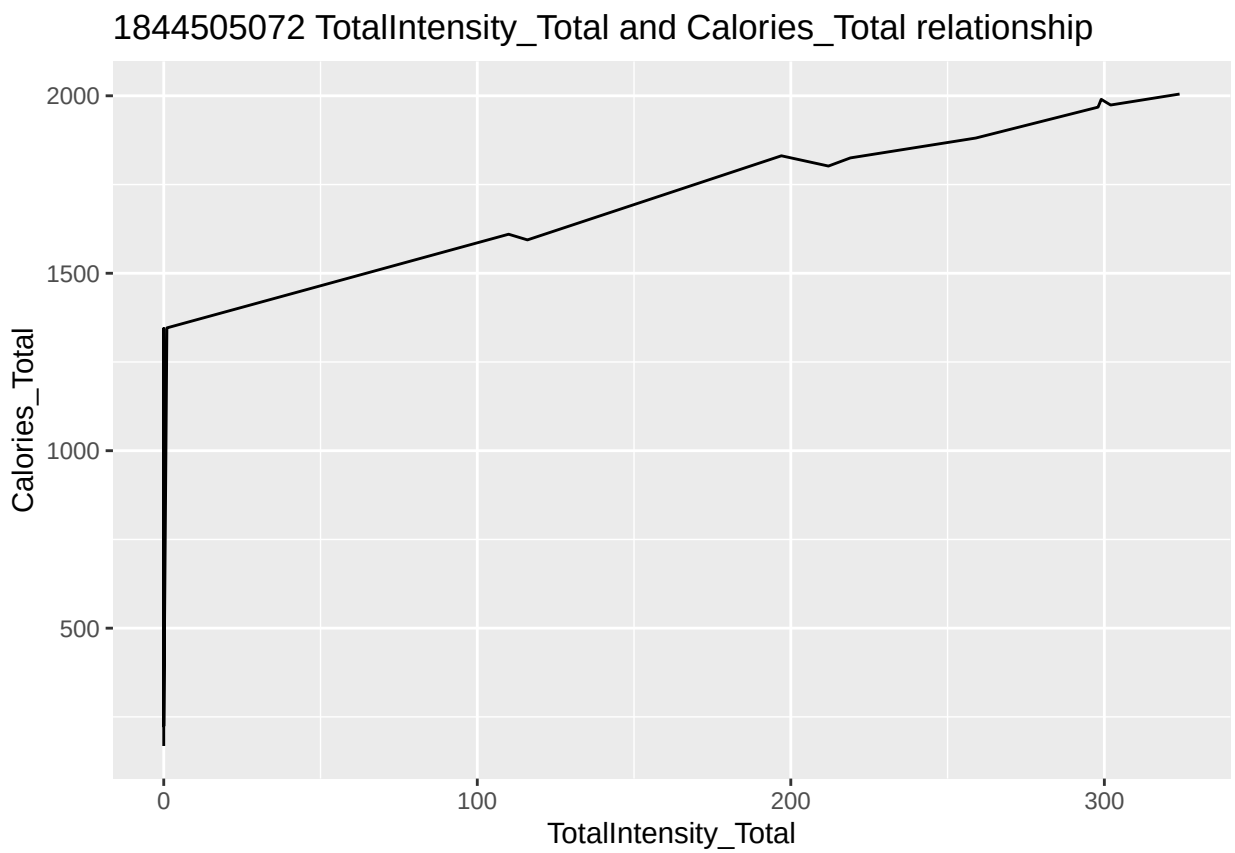
```
CaloriesIntensitiesSteps_Daily_df <-
  weekDate_grouping(CaloriesIntensitiesSteps_df,"date")
head(CaloriesIntensitiesSteps_Daily_df,5)
```

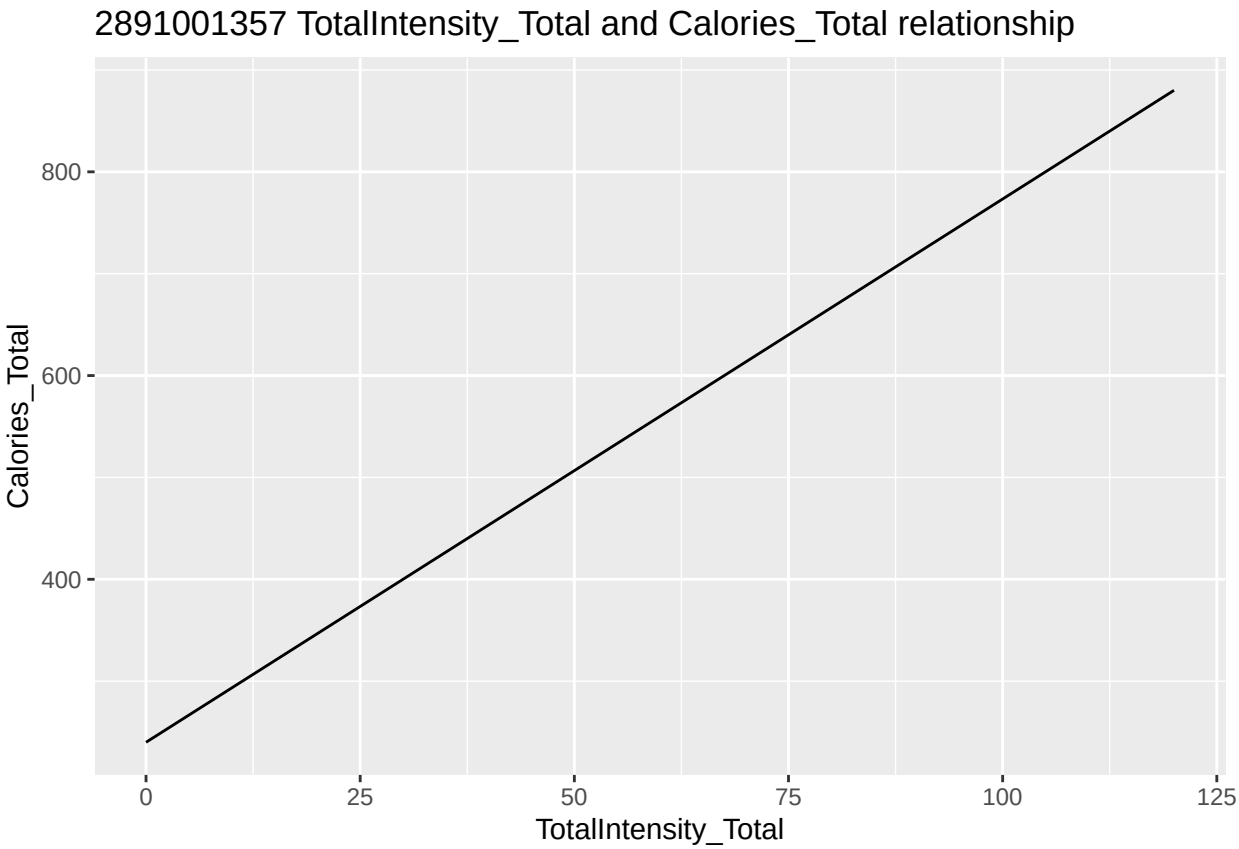
```
## # A tibble: 5 x 10
##           Id Group      Calories_Total Calories_Avg TotalIntensity_Total
##        <dbl> <date>              <int>        <dbl>                 <int>
## 1 1503960366 2016-03-11            144         48                       0
## 2 1503960366 2016-03-12           2237         93.2                   600
## 3 1503960366 2016-03-13           2106         87.8                   560
## 4 1503960366 2016-03-14           1836         76.5                   371
## 5 1503960366 2016-03-15           2092         87.2                   487
## # i 5 more variables: TotalIntensity_Avg <dbl>, AverageIntensity_Total <dbl>,
## #   AverageIntensity_Avg <dbl>, StepTotal_Total <int>, StepTotal_Avg <dbl>
```

```
names(CaloriesIntensitiesSteps_Daily_df)
```

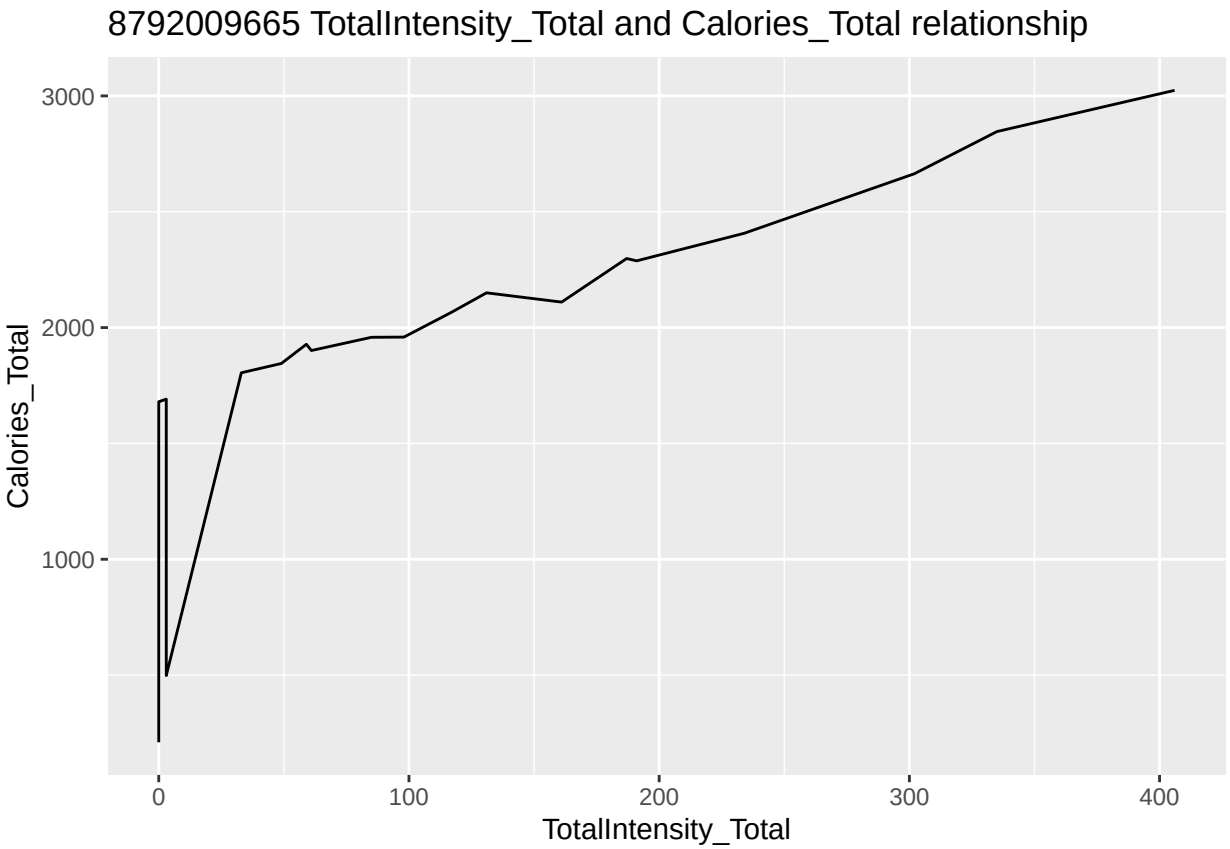
```
##  [1] "Id"                     "Group"                  "Calories_Total"
##  [4] "Calories_Avg"           "TotalIntensity_Total"   "TotalIntensity_Avg"
##  [7] "AverageIntensity_Total" "AverageIntensity_Avg"   "StepTotal_Total"
## [10] "StepTotal_Avg"
```

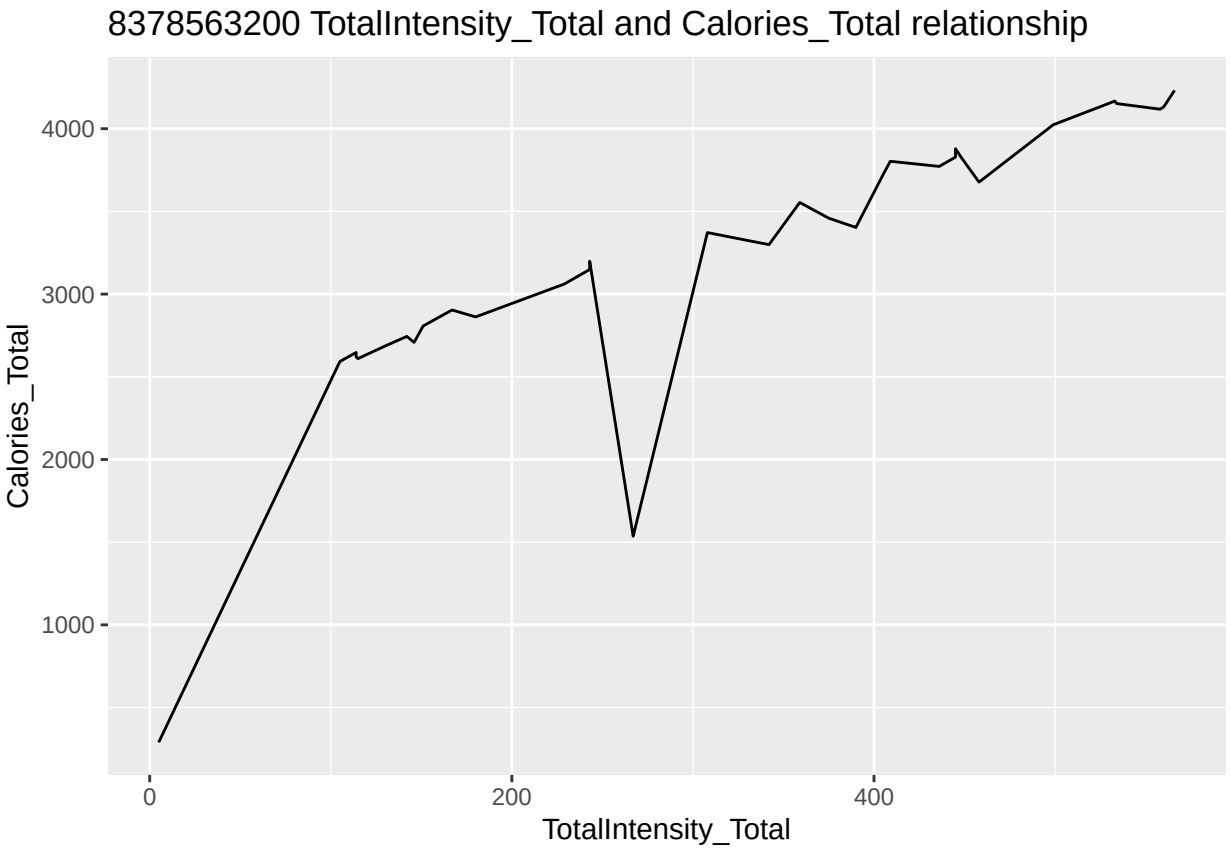
```
random_plots(df = CaloriesIntensitiesSteps_Daily_df,
             fraction_of_all_len_df = 0.3,
             X_col = "TotalIntensity_Total",
             Y_col ="Calories_Total")
```



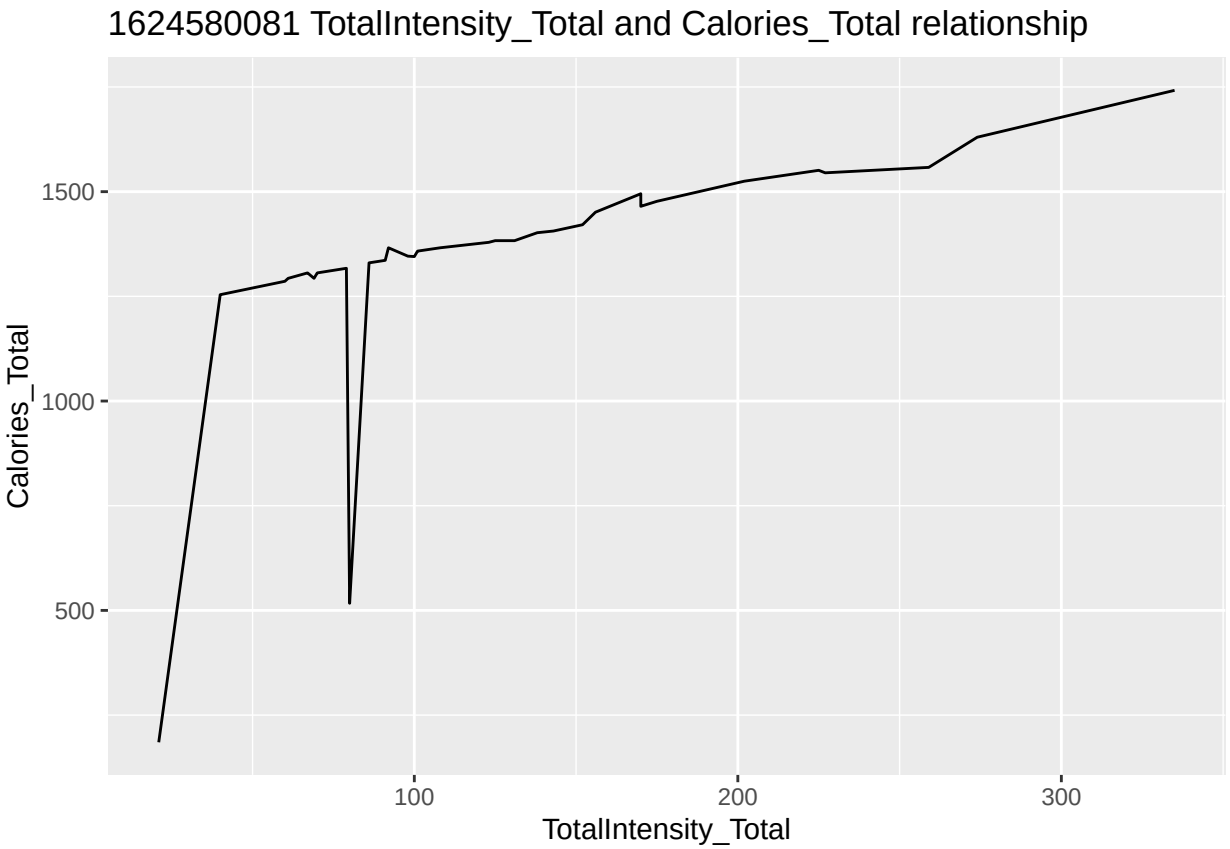

2891001357 TotalIntensity_Total and Calories_Total relationship
Calories_Total
800
600
400
0
25
50
75
100
125
TotalIntensity_Total

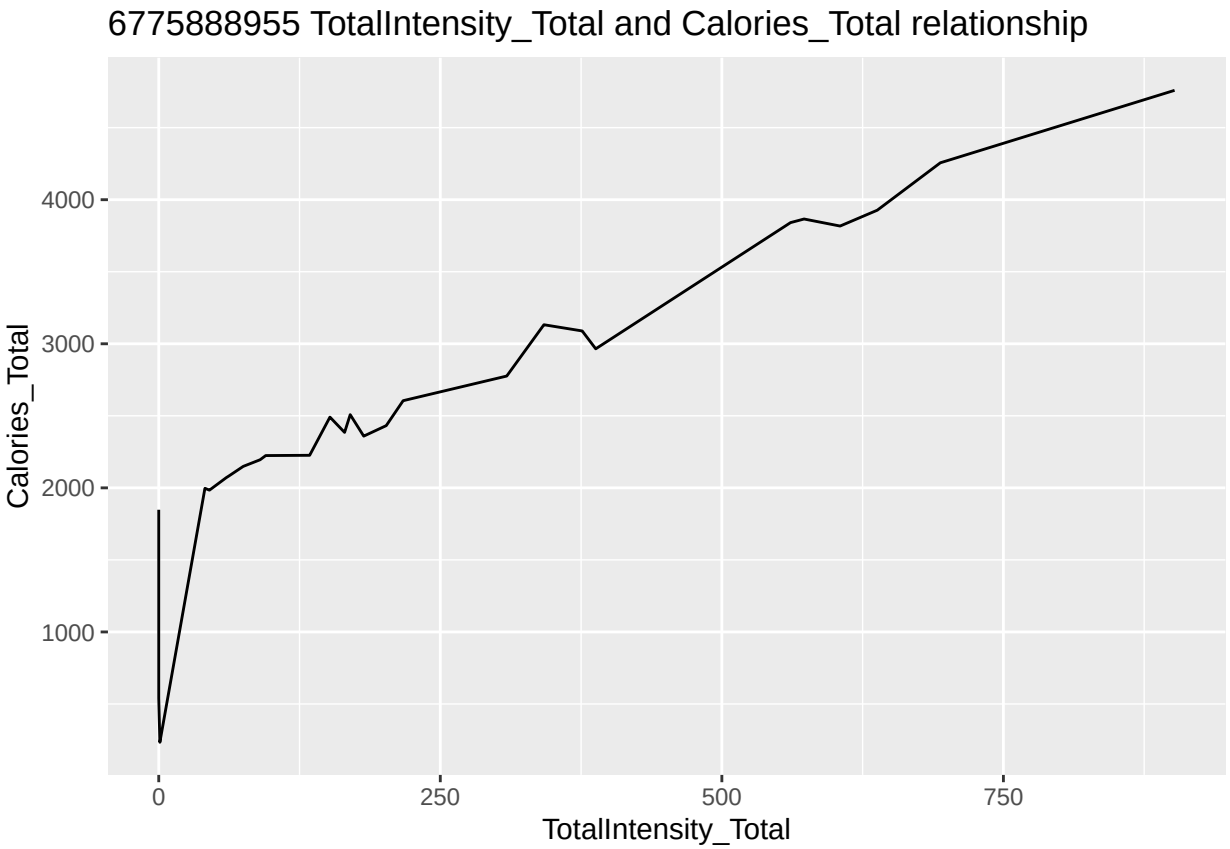

7007744171 TotalIntensity_Total and Calories_Total relationship
4000
3000
2000
1000
Calories_Total
0
250
500
750
TotalIntensity_Total

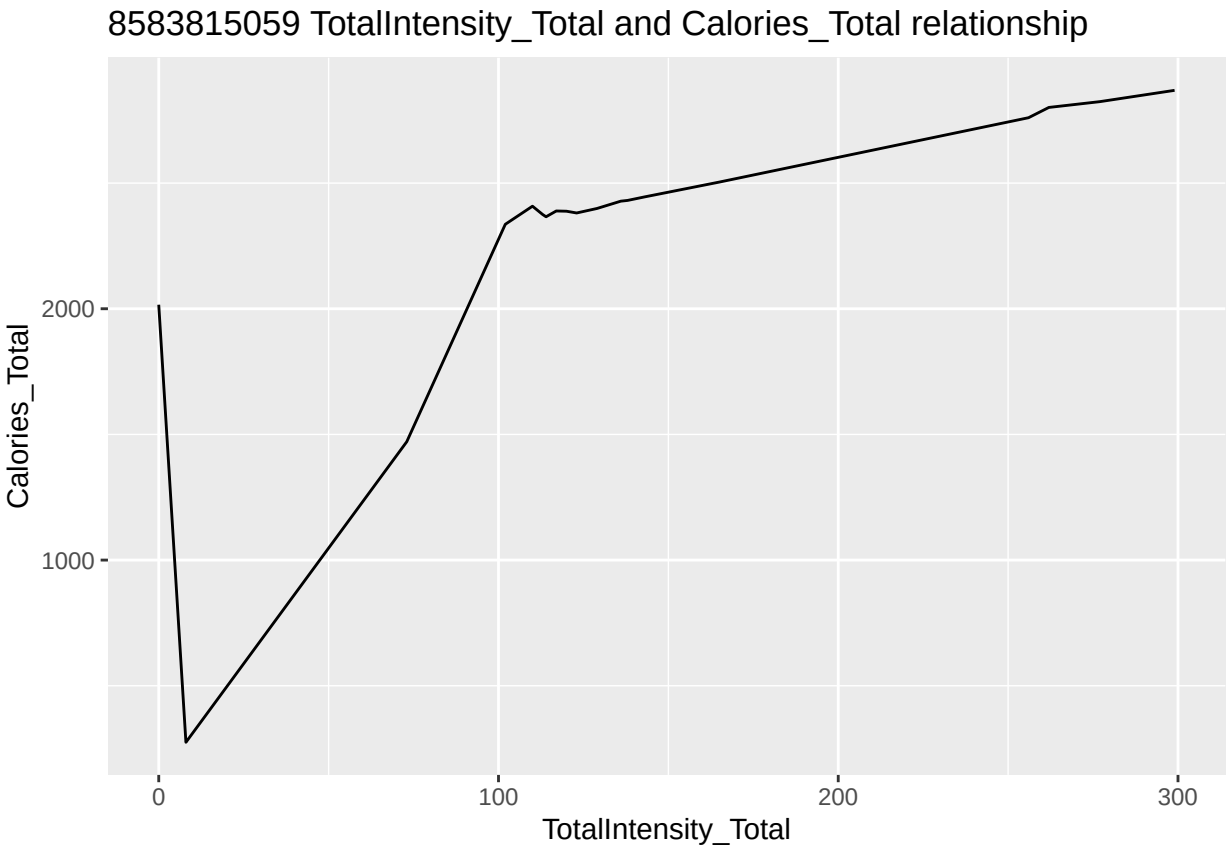

8792009665 TotalIntensity_Total and Calories_Total relationship
Calories_Total
3000
2000
1000
0
100
200
300
400
TotalIntensity_Total


8378563200 TotalIntensity_Total and Calories_Total relationship
Calories_Total
4000
3000
2000
1000
0
200
400
TotalIntensity_Total

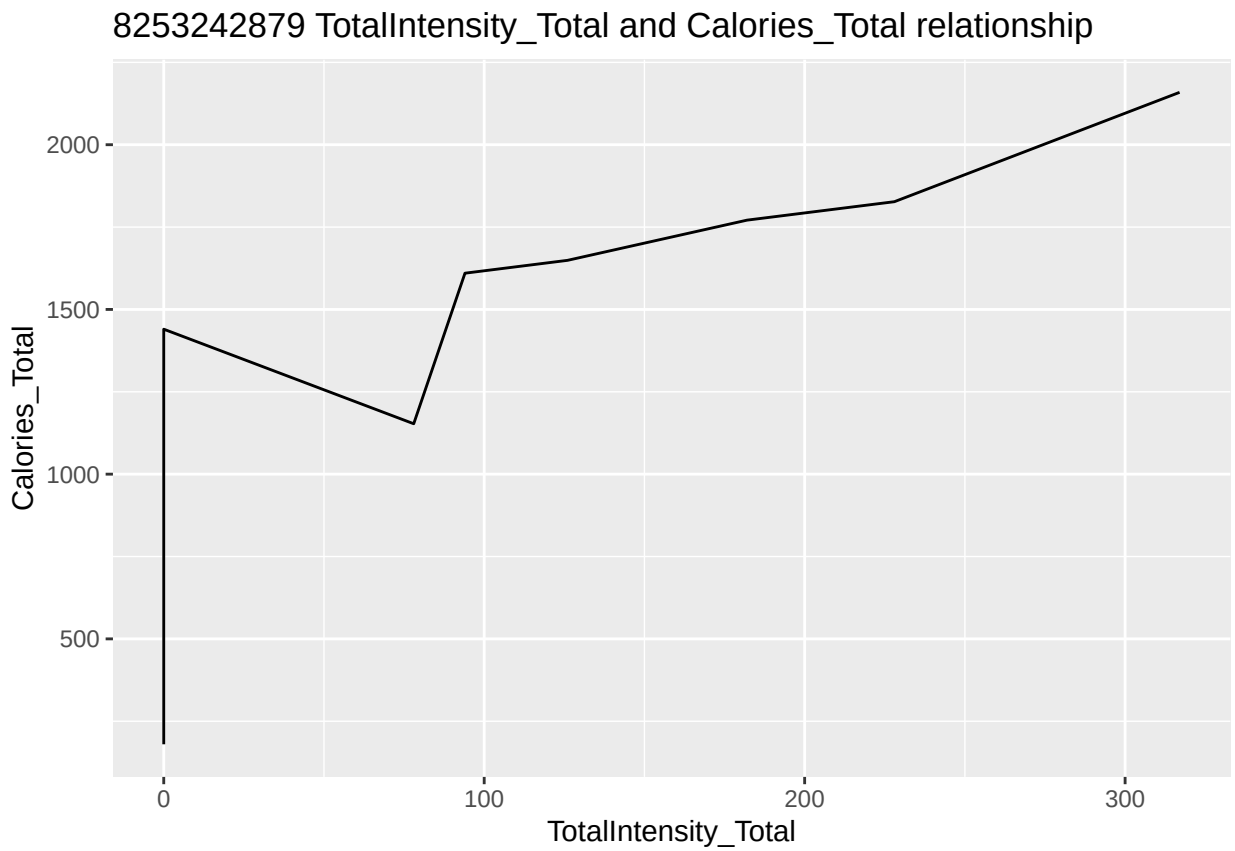

2873212765 TotalIntensity_Total and Calories_Total relationship
2500
2000
1500
1000
500
Calories_Total
0
200
400
600
TotalIntensity_Total


1624580081 TotalIntensity_Total and Calories_Total relationship
Calories_Total
1500
1000
500
100
200
300
TotalIntensity_Total


6775888955 TotalIntensity_Total and Calories_Total relationship
Calories_Total
4000
3000
2000
1000
0
250
500
750
TotalIntensity_Total


8583815059 TotalIntensity_Total and Calories_Total relationship
Calories_Total
2000
1000
0
100
200
300
TotalIntensity_Total


2347167796 TotalIntensity_Total and Calories_Total relationship
Calories_Total
2500
2000
1500
1000
500
0
200
400
TotalIntensity_Total




### Most people hit a point of diminishing returns between intensity of 50 to 200

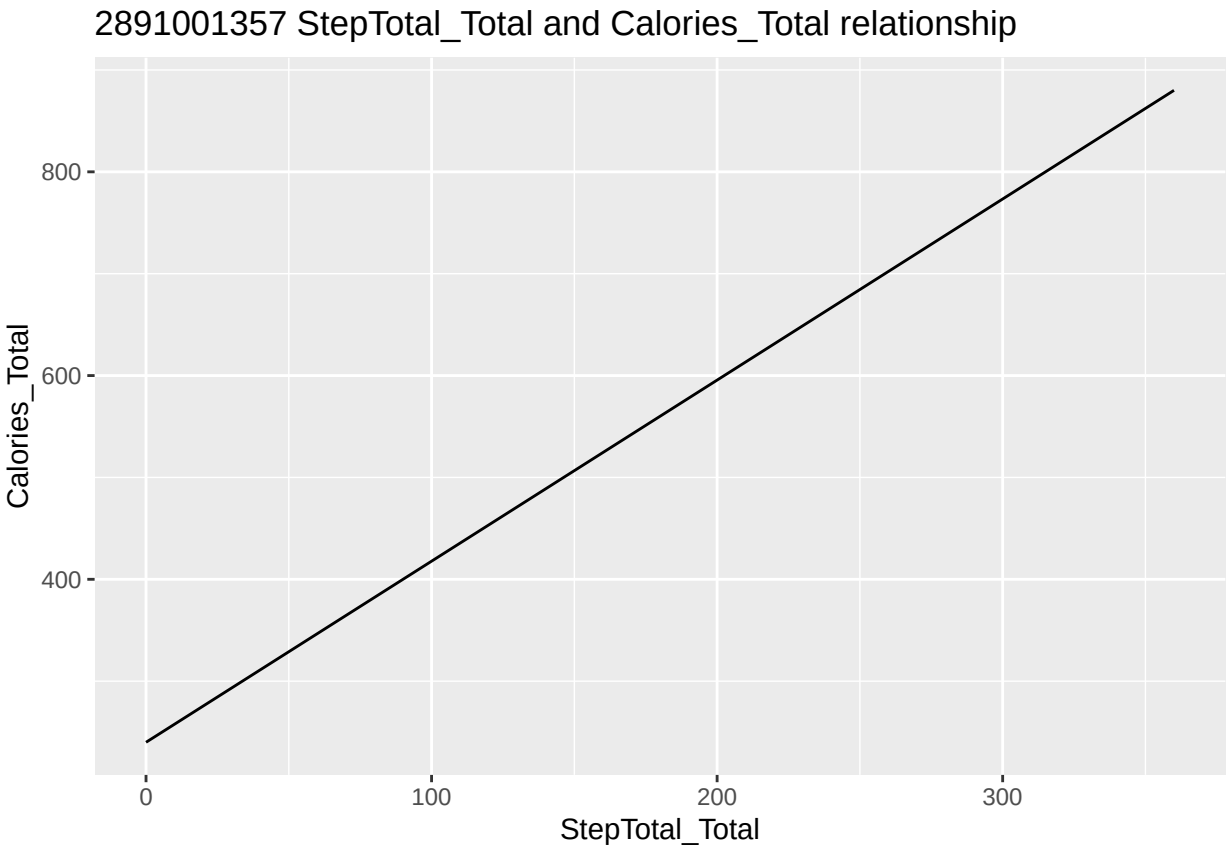
#### 0.2.1 Advise clients to atleast hit an intensity of 1250 daily to atleat make the most of their exercise

```
random_plots(df = CaloriesIntensitiesSteps_Daily_df,
             fraction_of_all_len_df = 0.3,
             X_col = "StepTotal_Total",
             Y_col ="Calories_Total")
```


4558609924 StepTotal_Total and Calories_Total relationship
Calories_Total
2000
1500
1000
500
0
2500
5000
7500
StepTotal_Total


1927972279 StepTotal_Total and Calories_Total relationship
4000
3000
2000
1000
Calories_Total
0
5000
10000
15000
20000
StepTotal_Total


6962181067 StepTotal_Total and Calories_Total relationship
3000
2000
1000
Calories_Total
0
10000
20000
StepTotal_Total


2891001357 StepTotal_Total and Calories_Total relationship
800
600
400
Calories_Total
0
100
200
300
StepTotal_Total


8053475328 StepTotal_Total and Calories_Total relationship
Calories_Total
4000
3000
2000
1000
0
10000
20000
30000
StepTotal_Total

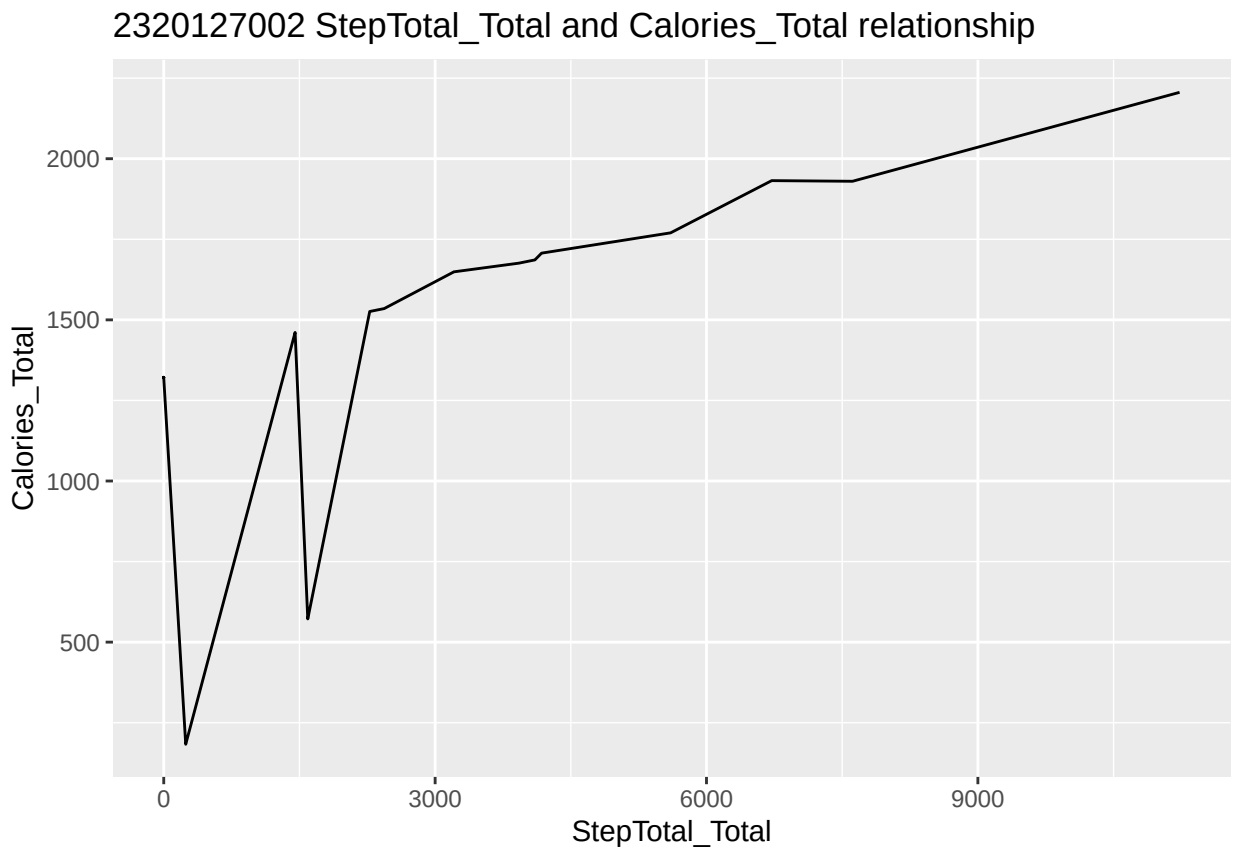

4702921684 StepTotal_Total and Calories_Total relationship
Calories_Total
3000
2000
1000
0
5000
10000
15000
StepTotal_Total


2873212765 StepTotal_Total and Calories_Total relationship
Calories_Total
2500
2000
1500
1000
500
0
5000
10000
15000
StepTotal_Total


8378563200 StepTotal_Total and Calories_Total relationship
Calories_Total
4000
3000
2000
1000
0
5000
10000
StepTotal_Total


4057192912 StepTotal_Total and Calories_Total relationship
Calories_Total
2500
2000
1500
1000
500
0
2000
4000
6000
8000
StepTotal_Total


5553957443 StepTotal_Total and Calories_Total relationship
Calories_Total
2000
1500
1000
500
0
5000
10000
15000
StepTotal_Total




### Findings: Different people get to the point of diminishing returns at different points there is no concrete pattern in the number of steps that would get on to hit a plateau, however most people once they hit 2000 calories they hit a point of diminishing return

**0.2.1.1 Reccomendation 1:** Once a person has hit 2000 calories a day they can quite exercing since since even if they continue exercisings since the steps thereafter are quit insignificant

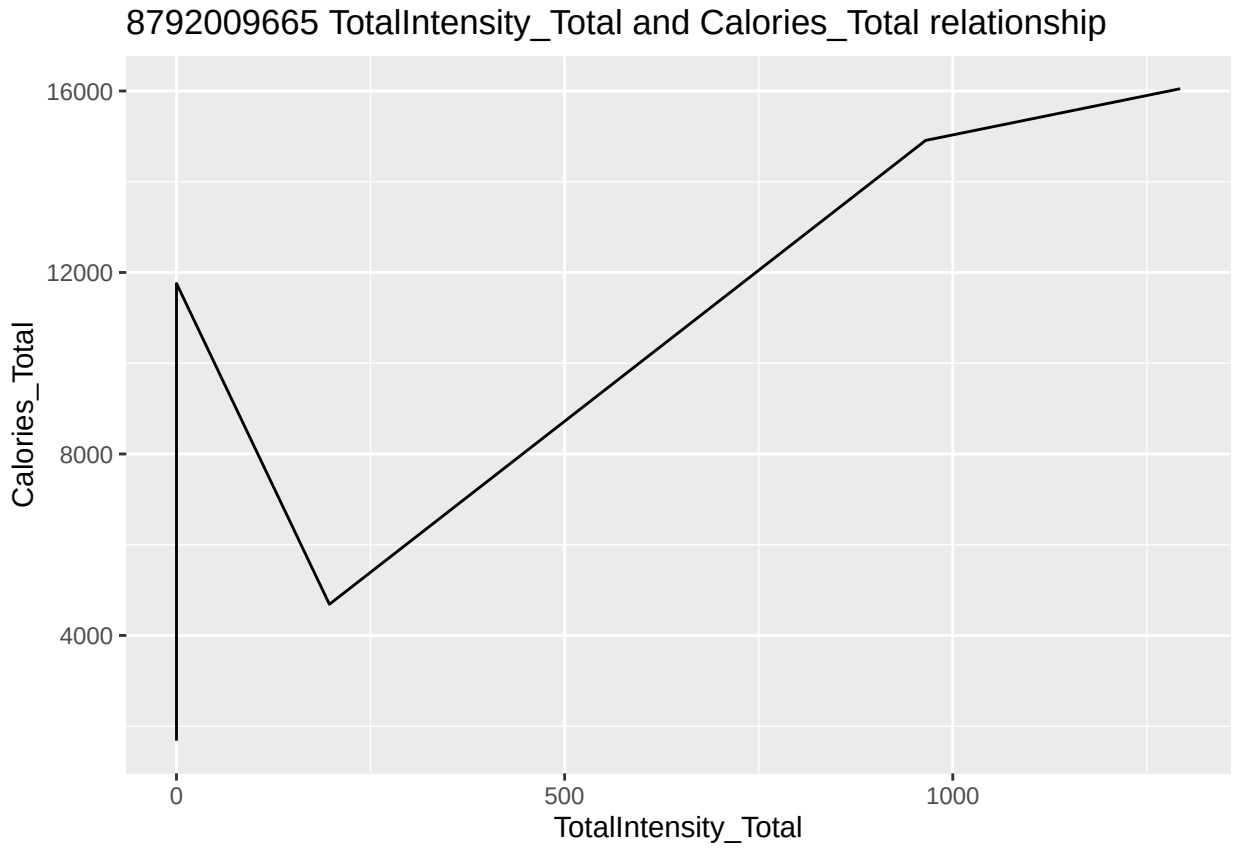
**0.2.1.2 Reccomendation 2:** clients should cover the number of steps that get them to 2000 calories.

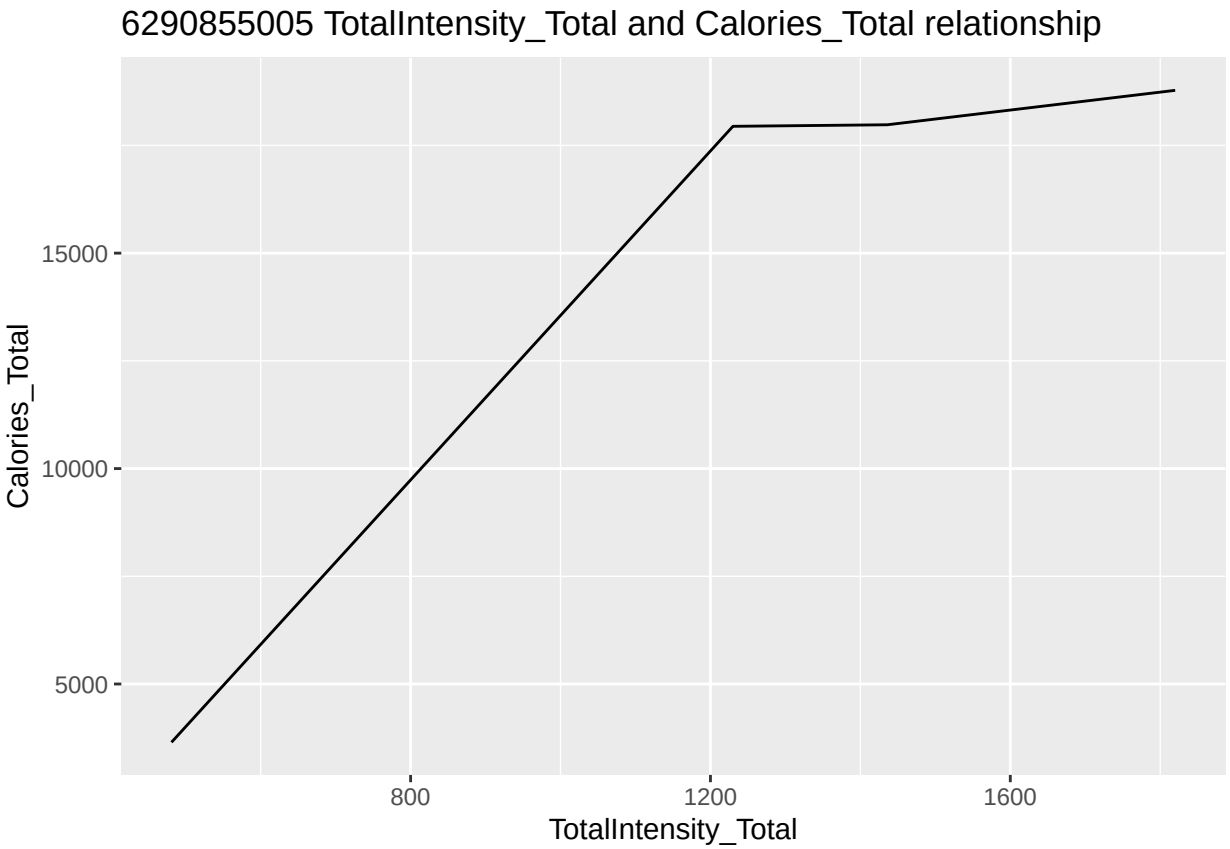
```
CaloriesIntensitiesSteps_Weekly_df <-
  weekDate_grouping(CaloriesIntensitiesSteps_df, "week")

head(CaloriesIntensitiesSteps_Weekly_df, 5)
```

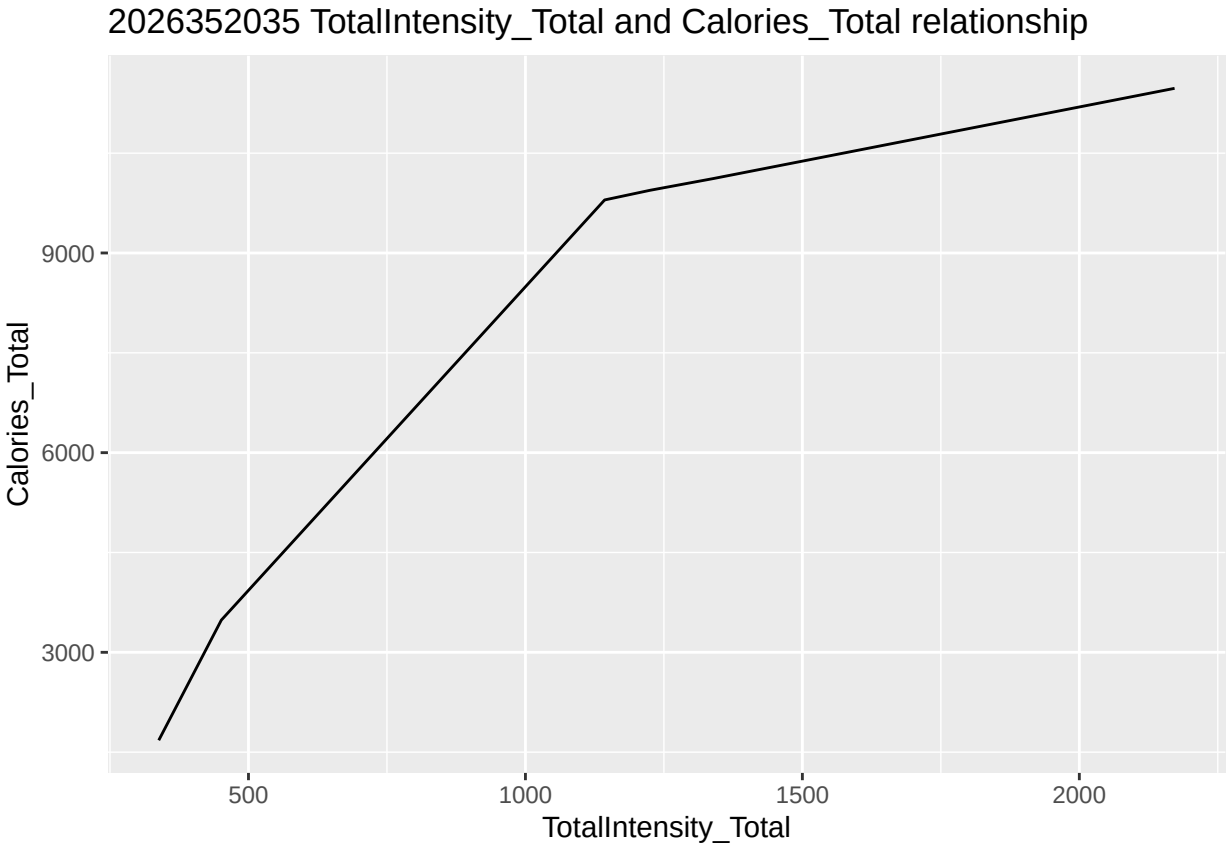
```
## # A tibble: 5 x 10
##           Id Group   Calories_Total Calories_Avg TotalIntensity_Total
##        <dbl> <chr>            <int>        <dbl>                <int>
## 1 1503960366 2016-10           2228         92.8                  596
## 2 1503960366 2016-11          14106         84.0                 3177
## 3 1503960366 2016-12          13886         82.7                 2986
## 4 1503960366 2016-13          13179         78.4                 2680
## 5 1503960366 2016-14          13358         79.5                 2741
## # i 5 more variables: TotalIntensity_Avg <dbl>, AverageIntensity_Total <dbl>,
## #   AverageIntensity_Avg <dbl>, StepTotal_Total <int>, StepTotal_Avg <dbl>
```

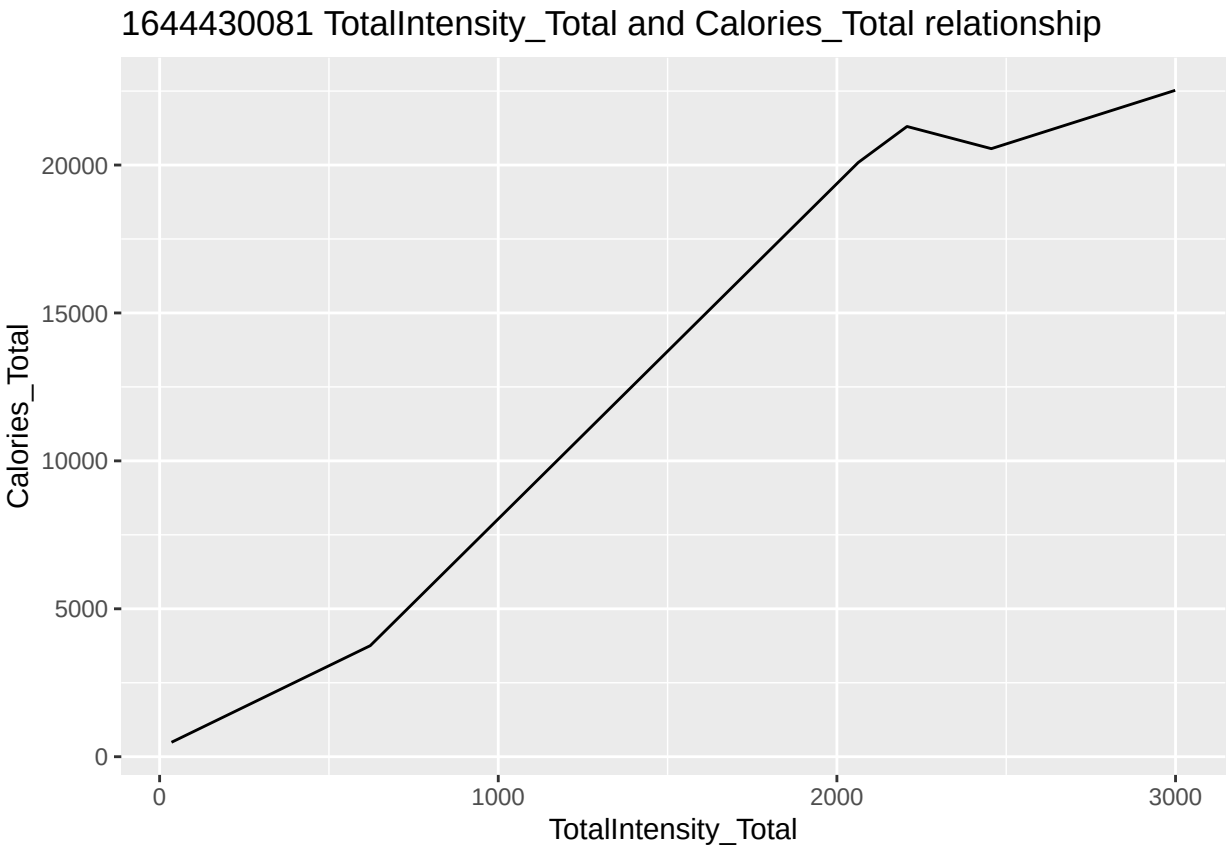
```
random_plots(df = CaloriesIntensitiesSteps_Weekly_df,
             fraction_of_all_len_df = 0.3,
```

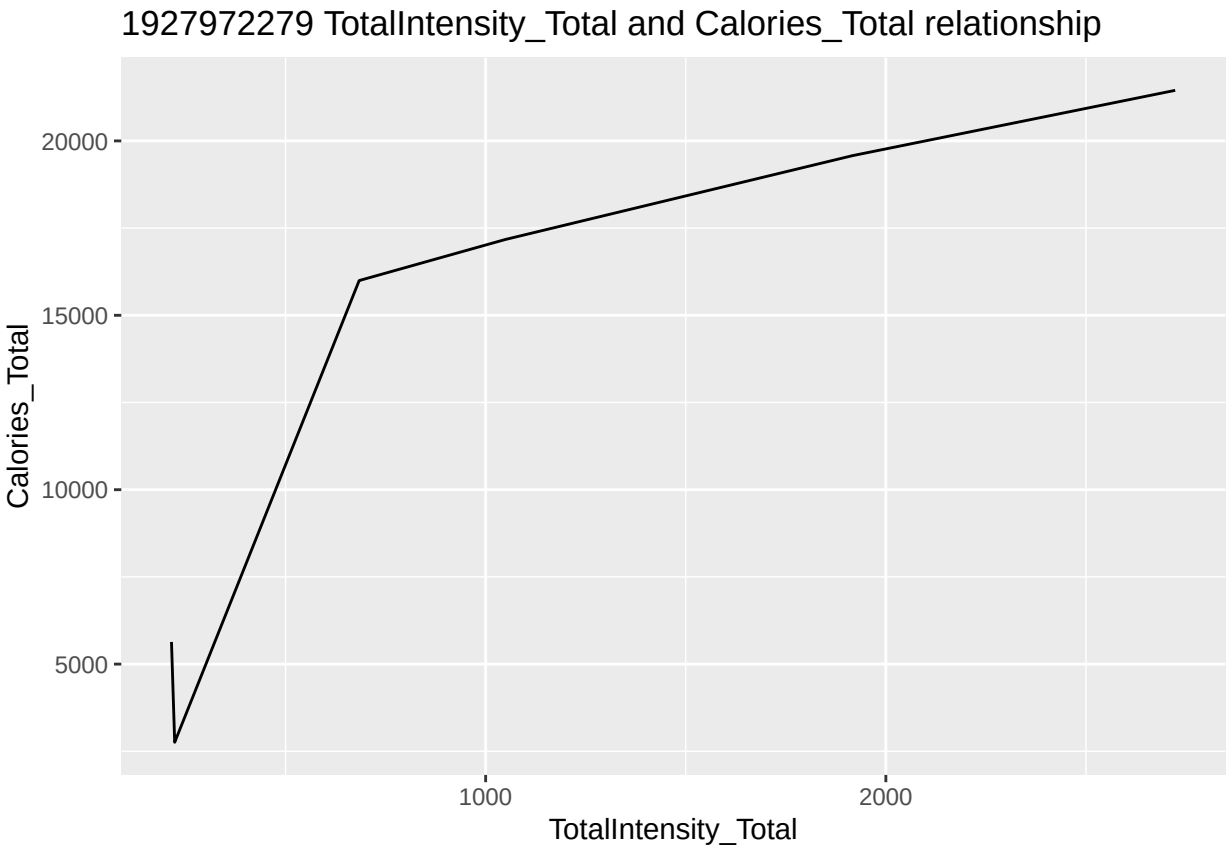
```
            X_col = "TotalIntensity_Total",
            Y_col = "Calories_Total")
```



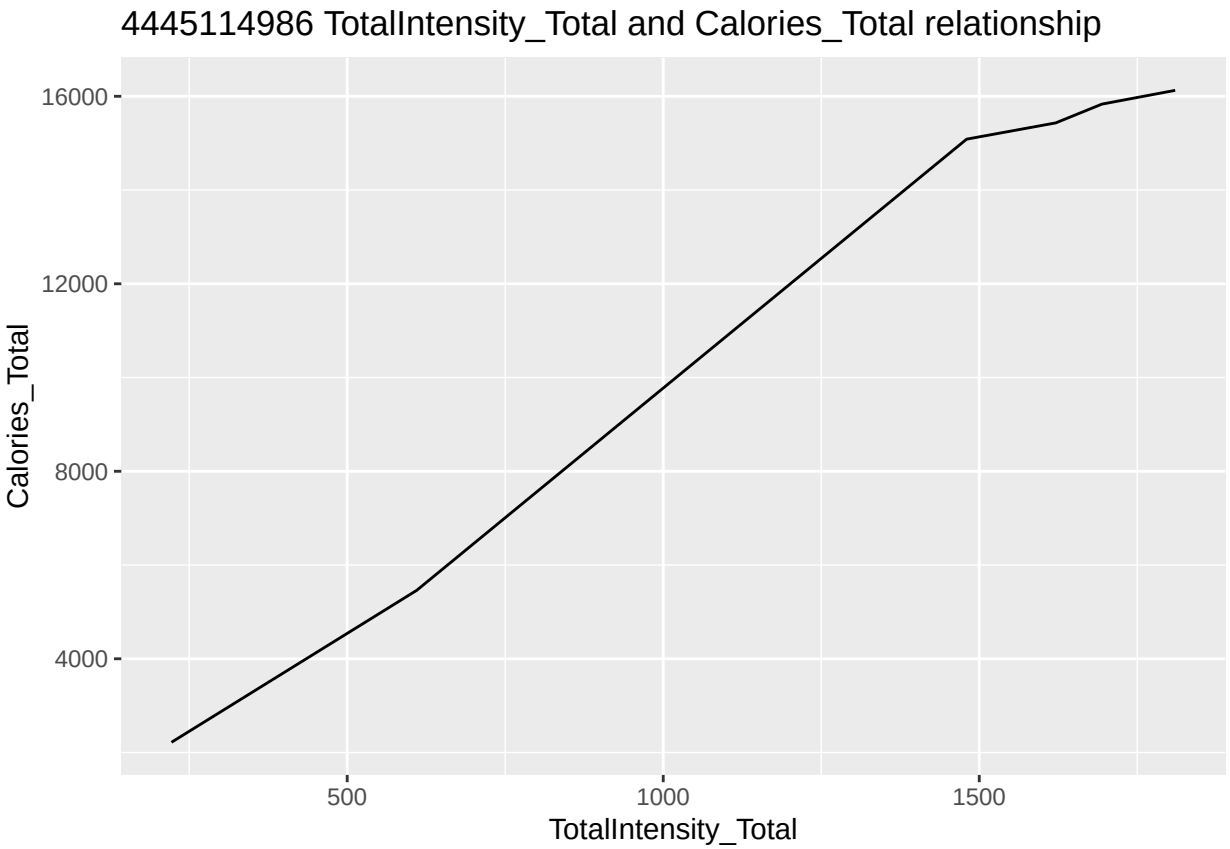

6290855005 TotalIntensity_Total and Calories_Total relationship
15000
10000
5000
Calories_Total
800
1200
1600
TotalIntensity_Total

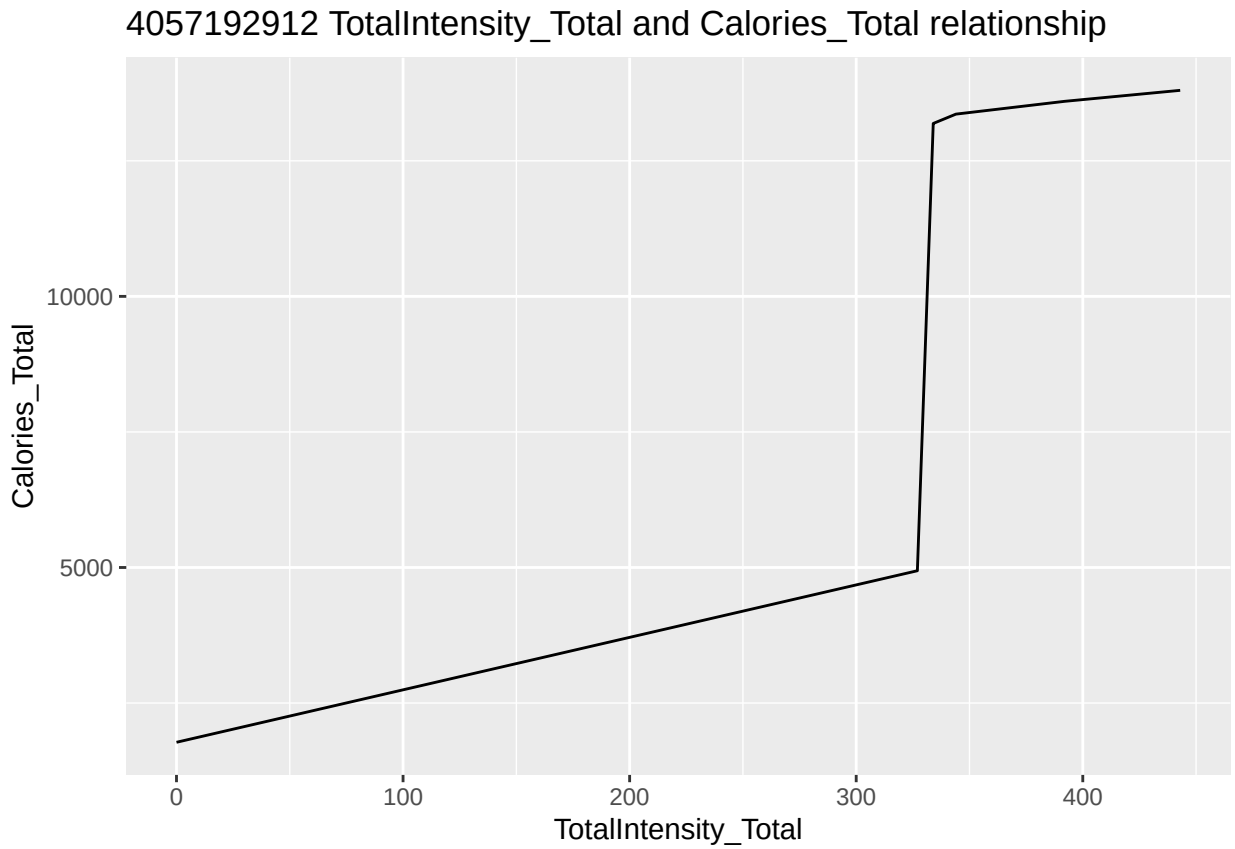

2026352035 TotalIntensity_Total and Calories_Total relationship
Calories_Total
9000
6000
3000
500
1000
1500
2000
TotalIntensity_Total


1644430081 TotalIntensity_Total and Calories_Total relationship
Calories_Total
20000
15000
10000
5000
0
0
1000
2000
3000
TotalIntensity_Total

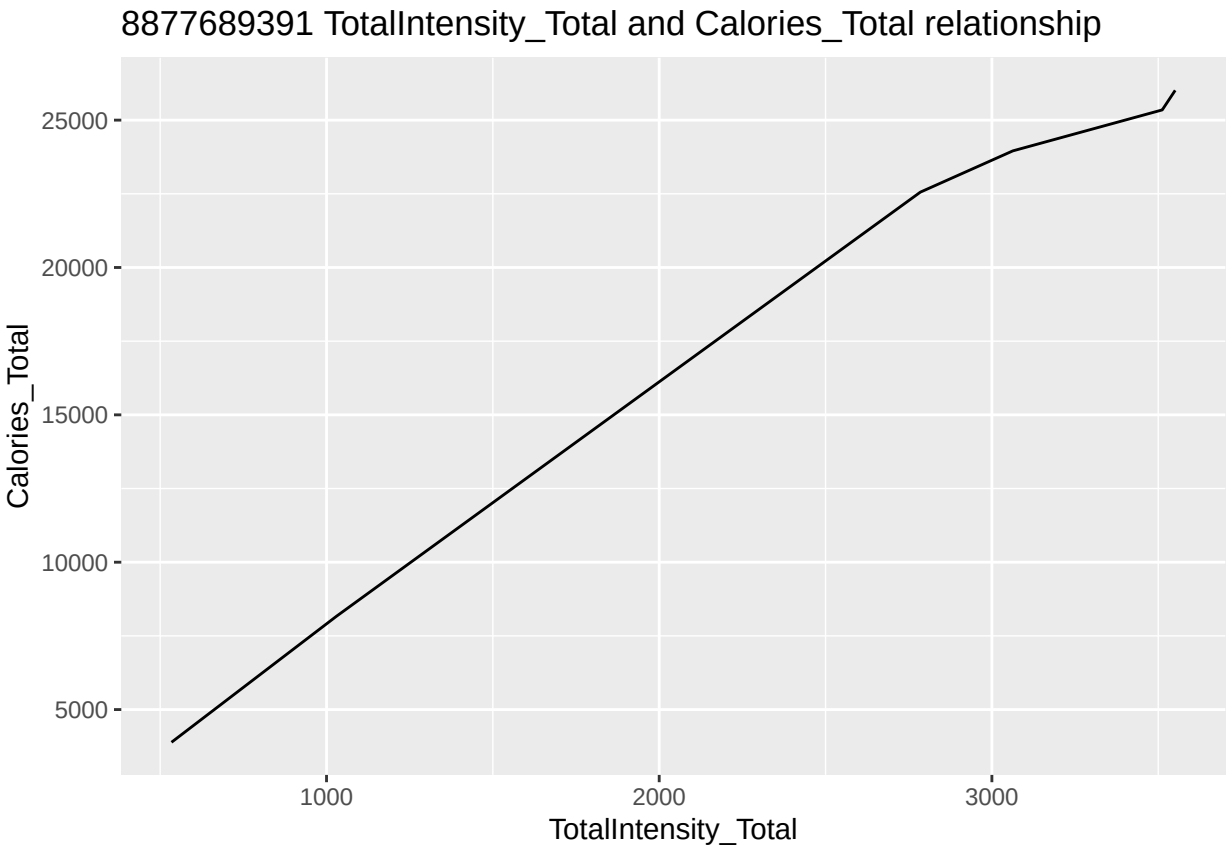

1927972279 TotalIntensity_Total and Calories_Total relationship
20000
15000
10000
5000
Calories_Total
1000
2000
TotalIntensity_Total

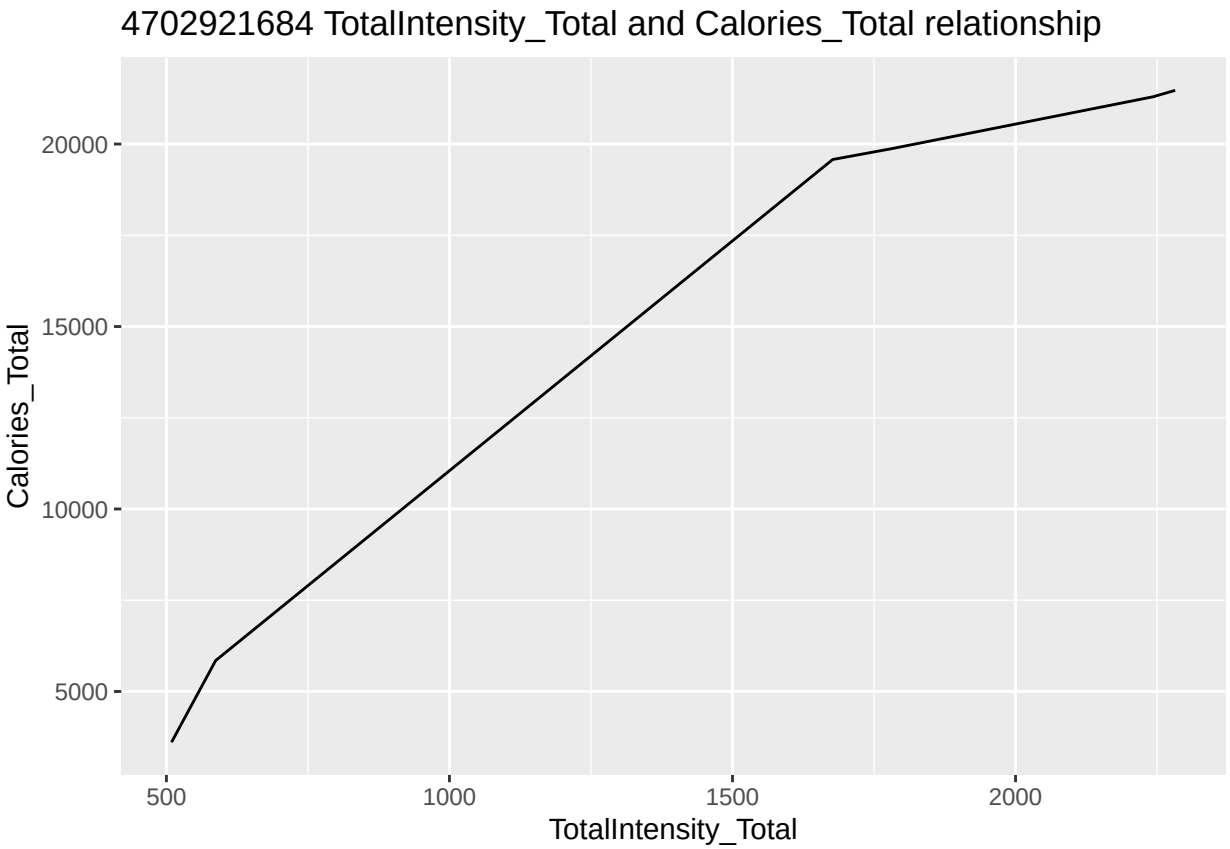

4020332650 TotalIntensity_Total and Calories_Total relationship
Calories_Total
20000
15000
10000
5000
0
500
1000
1500
2000
TotalIntensity_Total


4445114986 TotalIntensity_Total and Calories_Total relationship
16000
12000
8000
4000
Calories_Total
500
1000
1500
TotalIntensity_Total


4057192912 TotalIntensity_Total and Calories_Total relationship
Calories_Total
10000
5000
0
100
200
300
400
TotalIntensity_Total


6775888955 TotalIntensity_Total and Calories_Total relationship
20000
15000
10000
5000
Calories_Total
500
1000
1500
2000
TotalIntensity_Total


8877689391 TotalIntensity_Total and Calories_Total relationship
Calories_Total
25000
20000
15000
10000
5000
1000
2000
3000
TotalIntensity_Total

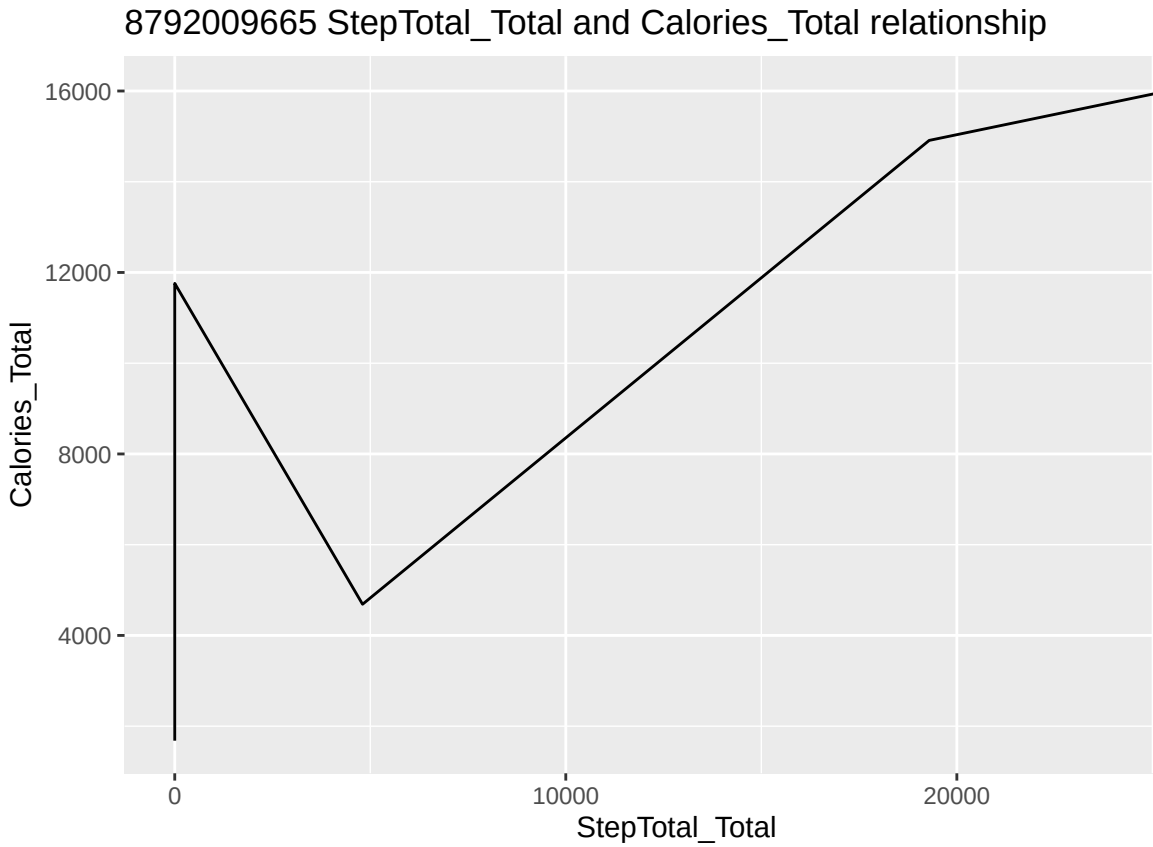



####Key observation most athletes hit a point of diminishing returns after an intensity of 2000, thus they loose less calories yet the intensity is high,

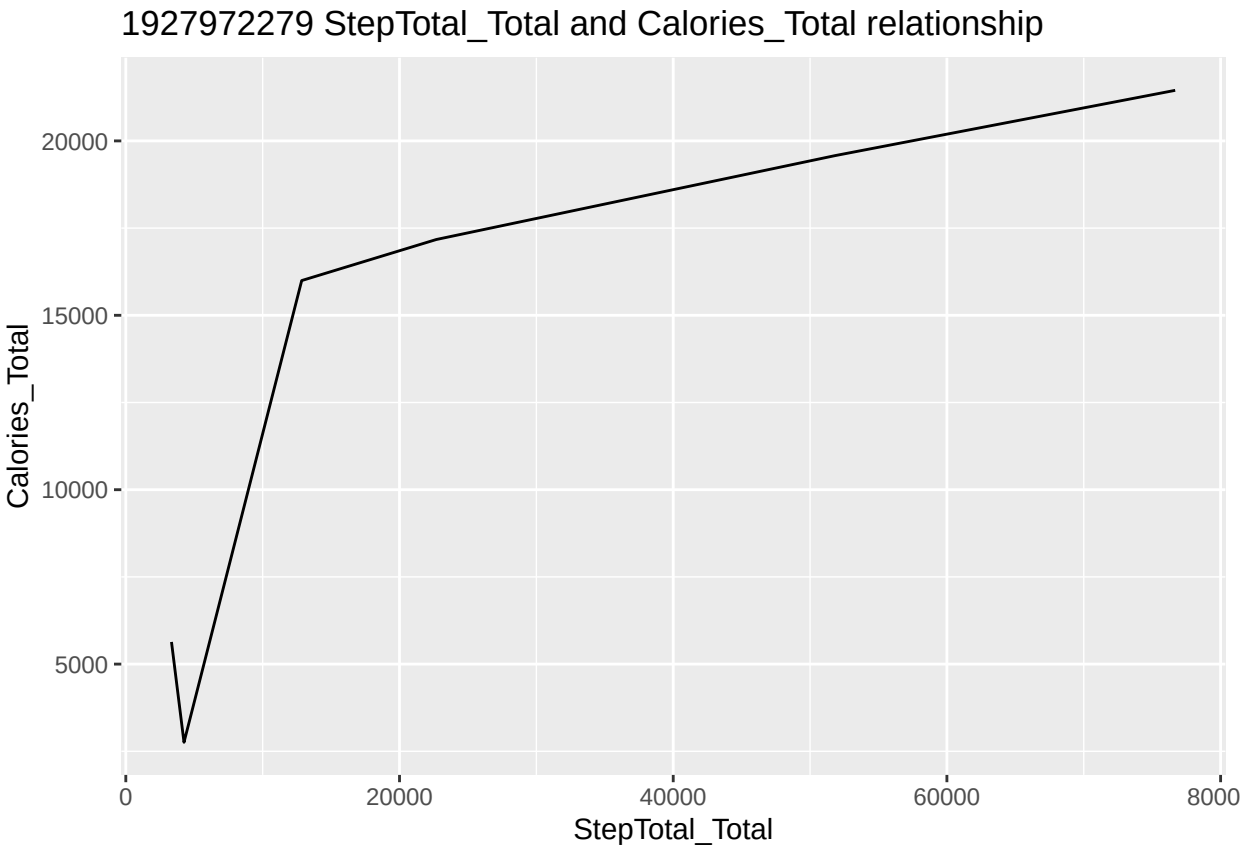
#### 0.2.1.3 recommendation: athletes should maintain an optimal intensity of about 2000

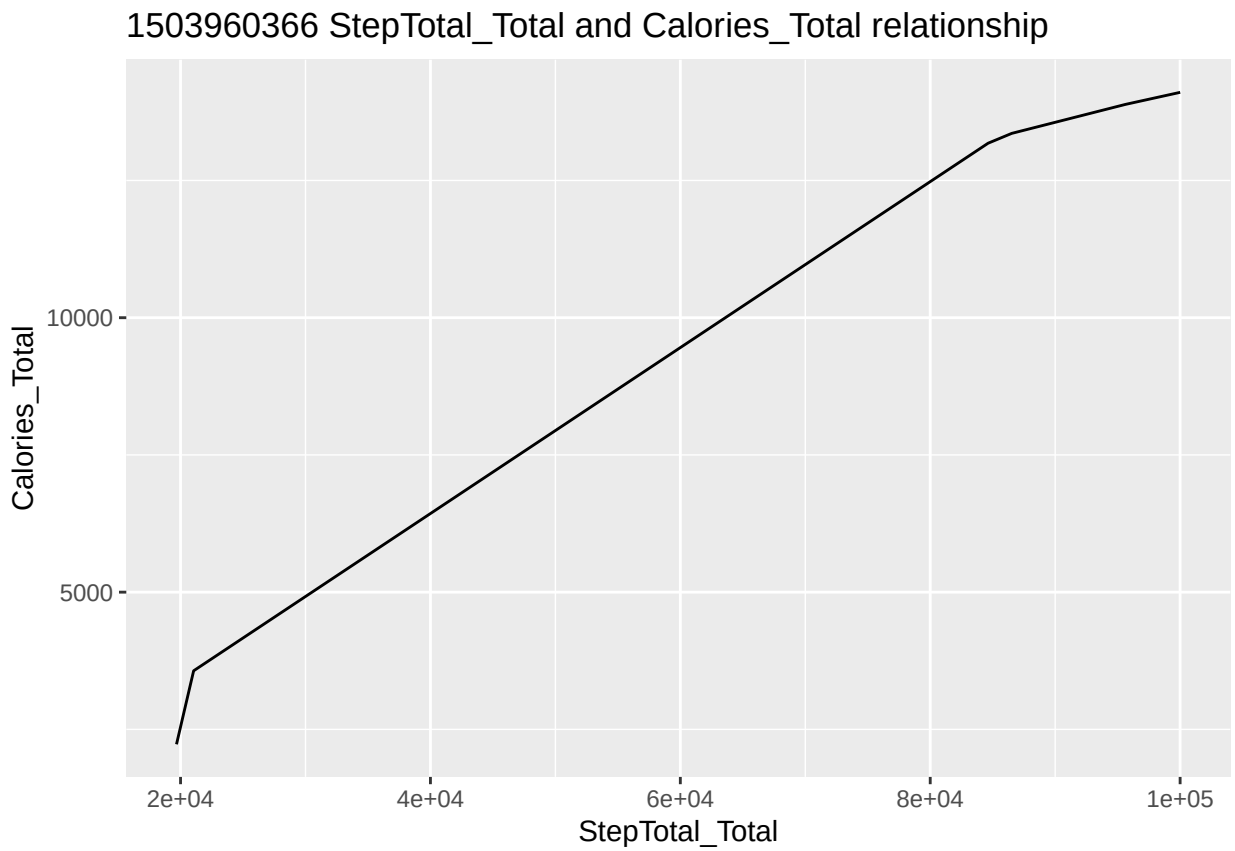
```
random_plots(df = CaloriesIntensitiesSteps_Weekly_df,
             fraction_of_all_len_df = 0.1,
             X_col = "StepTotal_Total",
             Y_col = "Calories_Total")
```

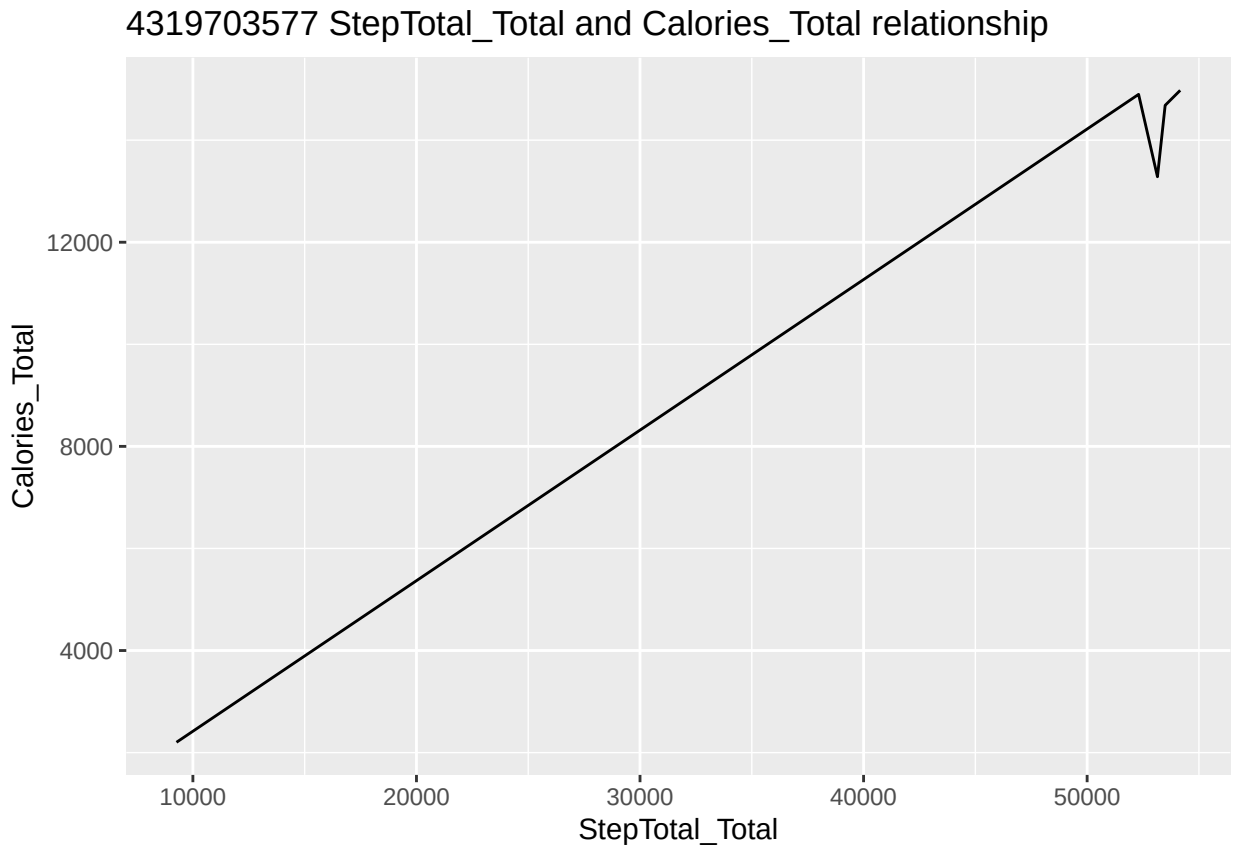
#### 0.2.1.4 This is a reflection of the relationship daily relationship between TotalIntensity_Total




and Calories_Total


1927972279 StepTotal_Total and Calories_Total relationship
Calories_Total
20000
15000
10000
5000
0
20000
40000
60000
8000
StepTotal_Total


1503960366 StepTotal_Total and Calories_Total relationship
Calories_Total
10000
5000
2e+04
4e+04
6e+04
8e+04
1e+05
StepTotal_Total




#### Where as there in no partern in the number of steps that it takes to hit a point of diminishing returns, there's is pattern in the number of calories lost beyond which an athlete hits a point of diminishing returns which is about 10000 caries weekly

```
# Loop through the list and see if key has minute in its name
minutes_dfs = list()

for (key in names(dfs)) {

  if (grepl("minute", key, ignore.case = TRUE)){

    print(paste("working on", key))

    col_Name <- colnames(dfs[[key]])
    column_of_interest <- col_Name[2]

    df <- change_to_dateTime(dfs[[key]], column_of_interest)

    minutes_dfs[[key]] <- df
  }

}
```

```
## [1] "working on minuteCaloriesNarrow_merged_df"
##        ActivityMinute
## 1 2016-03-12 00:00:00
## 2 2016-03-12 00:01:00
## 3 2016-03-12 00:02:00
```

```
## 4 2016-03-12 00:03:00
## 5 2016-03-12 00:04:00
##              ActivityMinute
## 1445036 2016-04-12 08:55:00
## 1445037 2016-04-12 08:56:00
## 1445038 2016-04-12 08:57:00
## 1445039 2016-04-12 08:58:00
## 1445040 2016-04-12 08:59:00
## [1] "working on minuteIntensitiesNarrow_merged_df"
##        ActivityMinute
## 1 2016-03-12 00:00:00
## 2 2016-03-12 00:01:00
## 3 2016-03-12 00:02:00
## 4 2016-03-12 00:03:00
## 5 2016-03-12 00:04:00
##              ActivityMinute
## 1445036 2016-04-12 08:55:00
## 1445037 2016-04-12 08:56:00
## 1445038 2016-04-12 08:57:00
## 1445039 2016-04-12 08:58:00
## 1445040 2016-04-12 08:59:00
## [1] "working on minuteMETsNarrow_merged_df"
##        ActivityMinute
## 1 2016-03-12 00:00:00
## 2 2016-03-12 00:01:00
## 3 2016-03-12 00:02:00
## 4 2016-03-12 00:03:00
## 5 2016-03-12 00:04:00
##              ActivityMinute
## 1445036 2016-04-12 08:55:00
## 1445037 2016-04-12 08:56:00
## 1445038 2016-04-12 08:57:00
## 1445039 2016-04-12 08:58:00
## 1445040 2016-04-12 08:59:00
## [1] "working on minuteSleep_merged_df"
##                  date
## 1 2016-03-13 02:39:30
## 2 2016-03-13 02:40:30
## 3 2016-03-13 02:41:30
## 4 2016-03-13 02:42:30
## 5 2016-03-13 02:43:30
##                       date
## 198555 2016-04-09 18:38:00
## 198556 2016-04-09 18:39:00
## 198557 2016-04-09 18:40:00
## 198558 2016-04-09 18:41:00
## 198559 2016-04-09 18:42:00
## [1] "working on minuteStepsNarrow_merged_df"
##        ActivityMinute
## 1 2016-03-12 00:00:00
## 2 2016-03-12 00:01:00
## 3 2016-03-12 00:02:00
## 4 2016-03-12 00:03:00
## 5 2016-03-12 00:04:00
```

```
##              ActivityMinute
## 1445036 2016-04-12 08:55:00
## 1445037 2016-04-12 08:56:00
## 1445038 2016-04-12 08:57:00
## 1445039 2016-04-12 08:58:00
## 1445040 2016-04-12 08:59:00
```

```
minuteSleep_merged_df <- minutes_dfs[["minuteSleep_merged_df"]]
print(head(minuteSleep_merged_df))
```

```
##           Id                date value       logId
## 1 1503960366 2016-03-13 02:39:30     1 11114919637
## 2 1503960366 2016-03-13 02:40:30     1 11114919637
## 3 1503960366 2016-03-13 02:41:30     1 11114919637
## 4 1503960366 2016-03-13 02:42:30     1 11114919637
## 5 1503960366 2016-03-13 02:43:30     1 11114919637
## 6 1503960366 2016-03-13 02:44:30     1 11114919637
```

```
# See that hour are people most asleep
# change the date column to time date
unique(minuteSleep_merged_df[['value']])
```

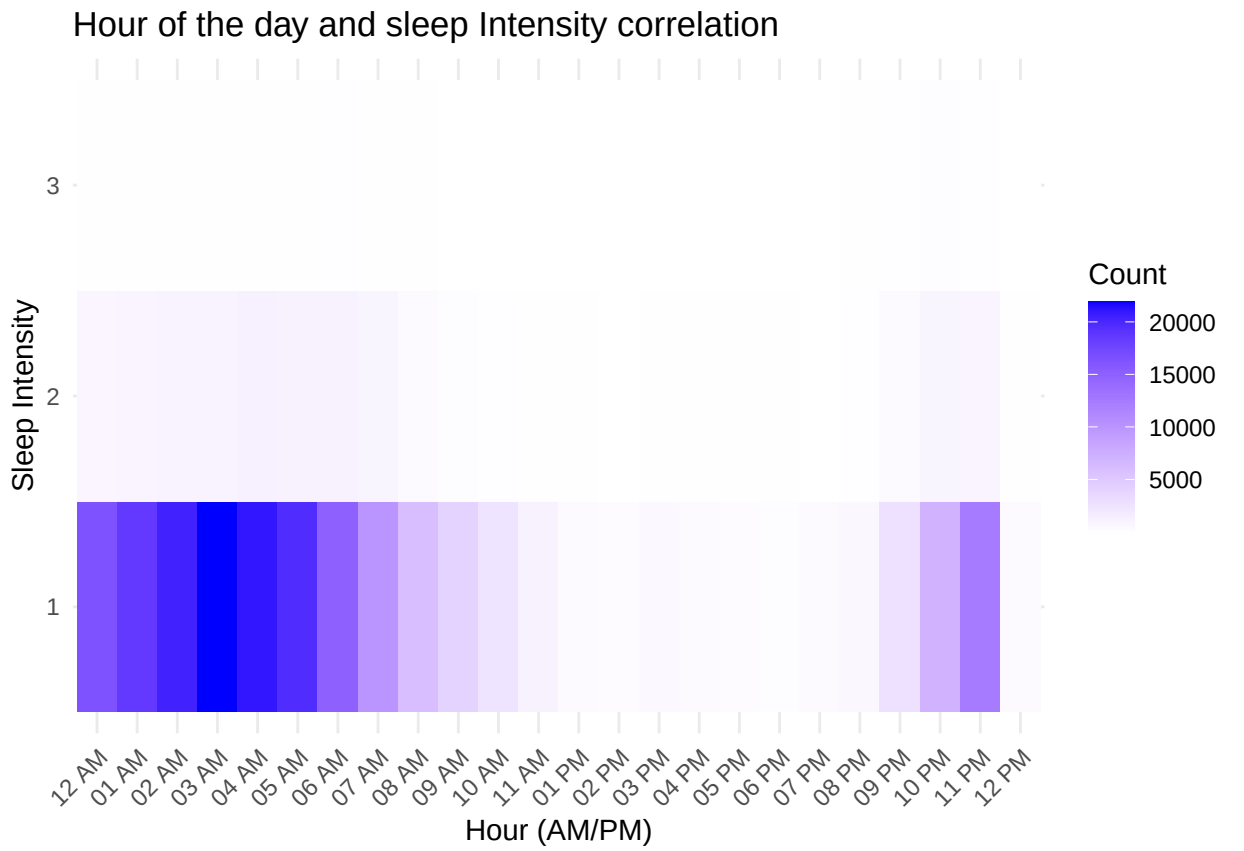
```
## [1] 1 2 3
```

```
HrSleep_df <- minuteSleep_merged_df[, c('Id','date', 'value')]

# Create a new column with the hour in AM/PM format
HrSleep_df$Hr <- format(HrSleep_df$date, "%I %p")

print(head(HrSleep_df))
```

```
##           Id                date value    Hr
## 1 1503960366 2016-03-13 02:39:30     1 02 AM
## 2 1503960366 2016-03-13 02:40:30     1 02 AM
## 3 1503960366 2016-03-13 02:41:30     1 02 AM
## 4 1503960366 2016-03-13 02:42:30     1 02 AM
## 5 1503960366 2016-03-13 02:43:30     1 02 AM
## 6 1503960366 2016-03-13 02:44:30     1 02 AM
```

```
# Ensure 'value' is treated as a factor (categorical)
HrSleep_df$value <- factor(HrSleep_df$value, levels = c(1, 2, 3))


# Define the correct order for the Hr factor
HrSleep_df$Hr <- factor(HrSleep_df$Hr,
                        levels = c("12 AM","01 AM", "02 AM", "03 AM", "04 AM",
                                   "05 AM","06 AM", "07 AM", "08 AM", "09 AM",
                                   "10 AM","11 AM", "01 PM", "02 PM", "03 PM",
                                   "04 PM", "05 PM", "06 PM", "07 PM", "08 PM",
                                   "09 PM", "10 PM", "11 PM", "12 PM"))

# Get count of occcurances per hour
heatmap_data <- HrSleep_df %>%
  group_by(Hr, value) %>%
  summarize(count = n(), .groups = 'drop')
```

```r
# Plot the heatmap
ggplot(heatmap_data, aes(x = Hr, y = value, fill = count)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "blue") +
  labs(title = "Hour of the day and sleep Intensity correlation",
       x = "Hour (AM/PM)",
       y = "Sleep Intensity", fill = "Count"
    ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




#### We might not know what the values 1,2,3 represent but from the correlation plot it seem like they represent sleep intensity with 1 being highly asleep as 3 as less asleep or awake

#### 0.2.1.5 Further we could conclude that most people sleep start sleeping between 9PM and 10PM all the way to 7Am

```r
daily_sleep_count <- minuteSleep_merged_df %>%
  # Create day column to represent the date alone without time
  mutate(day = as.Date(date)) %>%
  group_by(Id, day, value) %>%
  summarize(
    count = n(),
    .groups = "drop"
  ) %>%
```

```
  # Create a new column such that 1,2, 3 counts will be represented daily
  pivot_wider(names_from = value, values_from = count, values_fill = 0)

# see if people sleep during weekday or weekends
# Process the data to add weekday/weekend information
daily_sleep_count <- daily_sleep_count %>%
  mutate(
    week_day = weekdays(day)
  )

# Get a matrix with counts of 1, 2, 3 for each day of the week
weekly_counts <- daily_sleep_count %>%
  group_by(week_day) %>%
  summarize(
    "1" = sum(`1`),
    "2" = sum(`2`),
    "3" = sum(`3`),
    .groups = "drop"
  )
weekly_counts
```

#### 0.2.1.6 Most people are heavily asleep between 2AM all the way to 5AM and 3PM being the point when they are deepest asleep.

```
## # A tibble: 7 x 4
##   week_day    `1`   `2`   `3`
##   <chr>     <int> <int> <int>
## 1 Friday    23531  1541   258
## 2 Monday    27047  1905   245
## 3 Saturday  30440  2576   436
## 4 Sunday    31804  2187   298
## 5 Thursday  22119  1442   196
## 6 Tuesday   23905  1500   275
## 7 Wednesday 24680  1859   315
```

```
# Remove the key 'b' from the list
print(length(minutes_dfs))
```

#### 0.2.1.7 From the above matrix we can see that people are sleep the most on Sunday closely followed by Saturday and Monday

```
## [1] 5
```

```
minutes_dfs[["minuteSleep_merged_df"]] <- NULL
print(paste("Length After removing minuteSleep_merged_df",length(minutes_dfs)))
```

```
## [1] "Length After removing minuteSleep_merged_df 4"
```

```
# See if the the remaining dfs in minutes_df have similar values in ActivityMinute column since the are
for (i in minutes_dfs){
  main_df = minutes_dfs[[1]]

  column1 <- "ActivityMinute"
  column2 <- "ActivityMinute"
```

```
  missing_values <- uniqueIDs_Comparison(main_df, column1, minutes_dfs[[1]],
                                         column2)
}
```

```
## [1] "Dataframes have similar values"
## [1] "Dataframes have similar values"
## [1] "Dataframes have similar values"
## [1] "Dataframes have similar values"
```

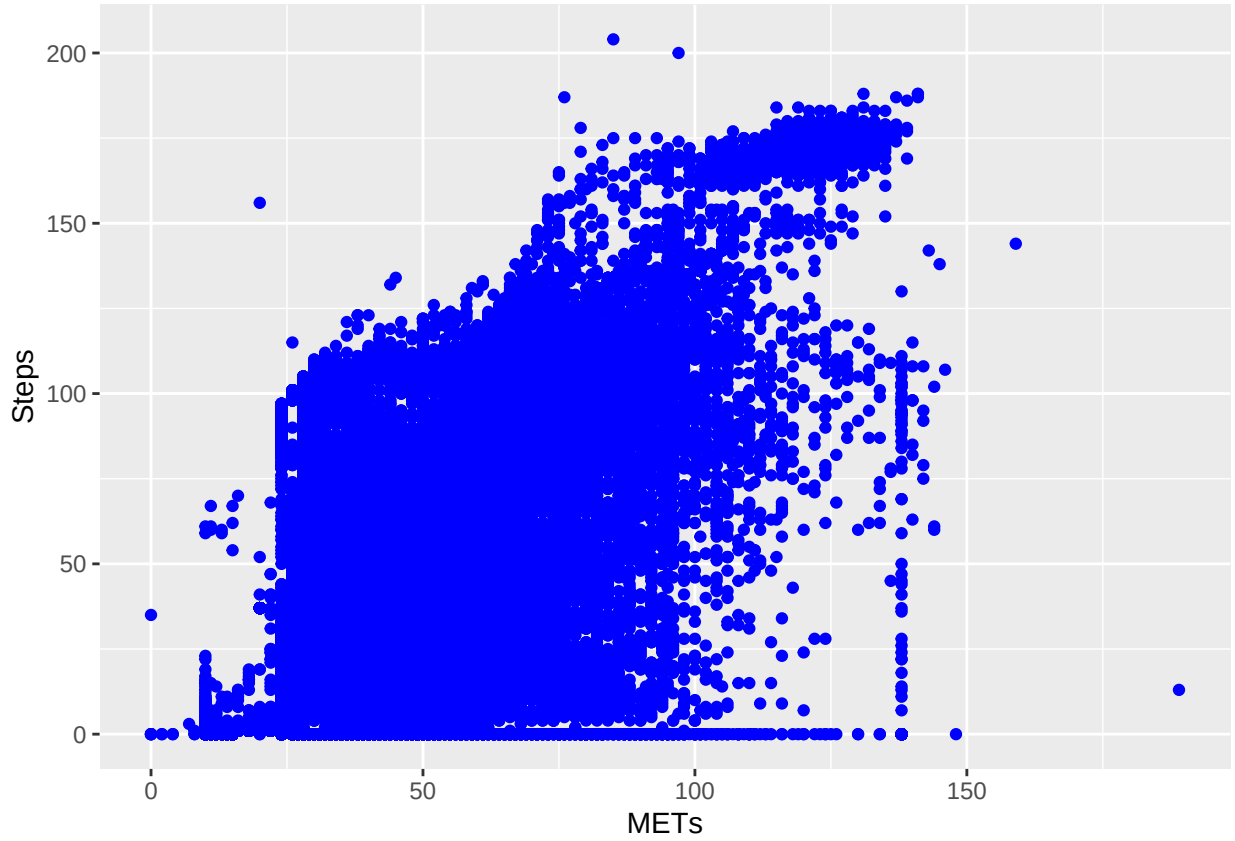
```
# Merge all dataframes on 'Id' and 'Date'
Minuted_Merged_df <- reduce(minutes_dfs, function(x, y) {
  merge(x, y, by = c("Id", "ActivityMinute"), all = TRUE)
})

print(head(Minuted_Merged_df, 5))
```

```
##           Id      ActivityMinute Calories Intensity METs Steps
## 1 1503960366 2016-03-12 00:00:00   0.7973         0   10     0
## 2 1503960366 2016-03-12 00:01:00   0.7973         0   10     0
## 3 1503960366 2016-03-12 00:02:00   0.7973         0   10     0
## 4 1503960366 2016-03-12 00:03:00   0.7973         0   10     0
## 5 1503960366 2016-03-12 00:04:00   0.7973         0   10     0
```

```
print(tail(Minuted_Merged_df, 5))
```

```
##                 Id      ActivityMinute Calories Intensity METs Steps
## 1445036 8877689391 2016-04-12 08:55:00  1.22480         0   10     0
## 1445037 8877689391 2016-04-12 08:56:00  1.22480         0   10     0
## 1445038 8877689391 2016-04-12 08:57:00  1.22480         0   10     0
## 1445039 8877689391 2016-04-12 08:58:00  1.34728         0   11     0
## 1445040 8877689391 2016-04-12 08:59:00  1.22480         0   10     0
```

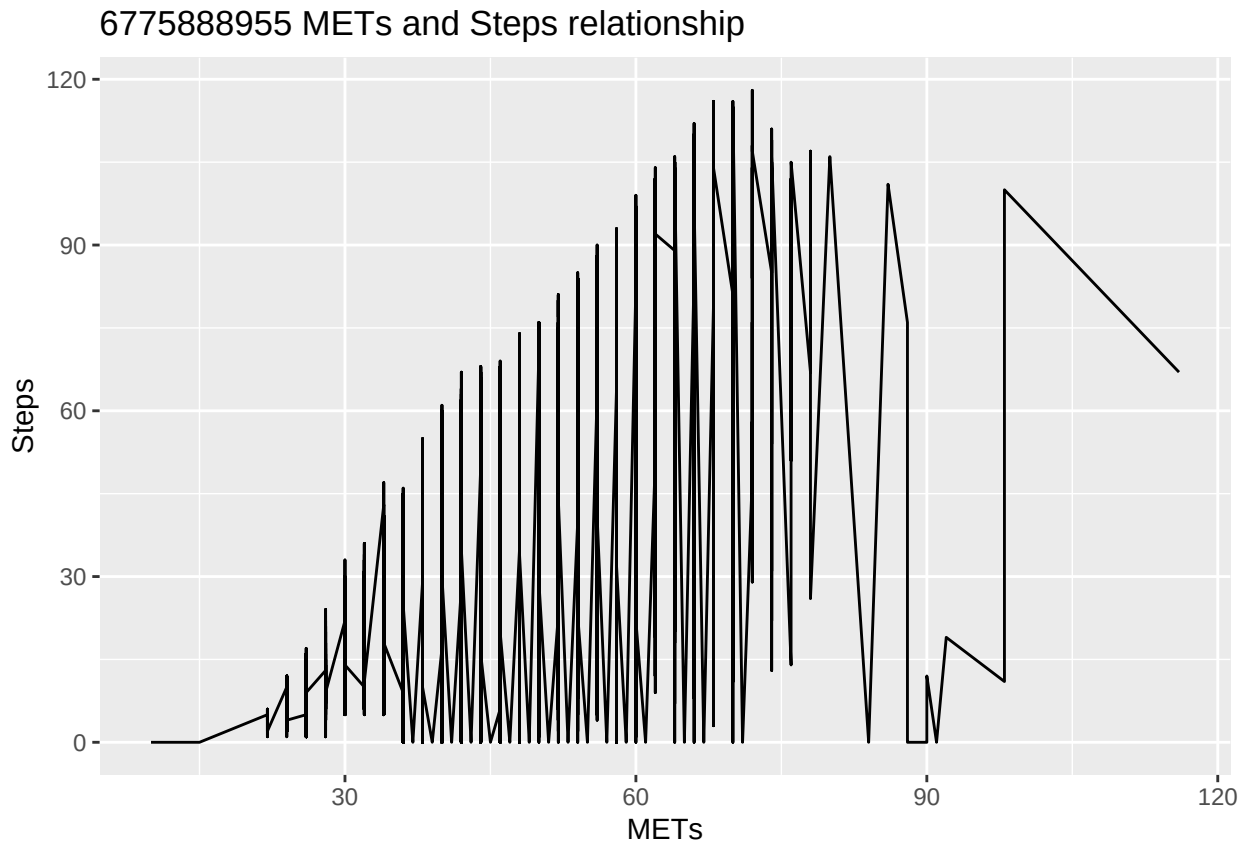
```
ggplot (data = Minuted_Merged_df) +
  geom_point(mapping = aes(x = METs , y = Steps), color = "blue" )
```

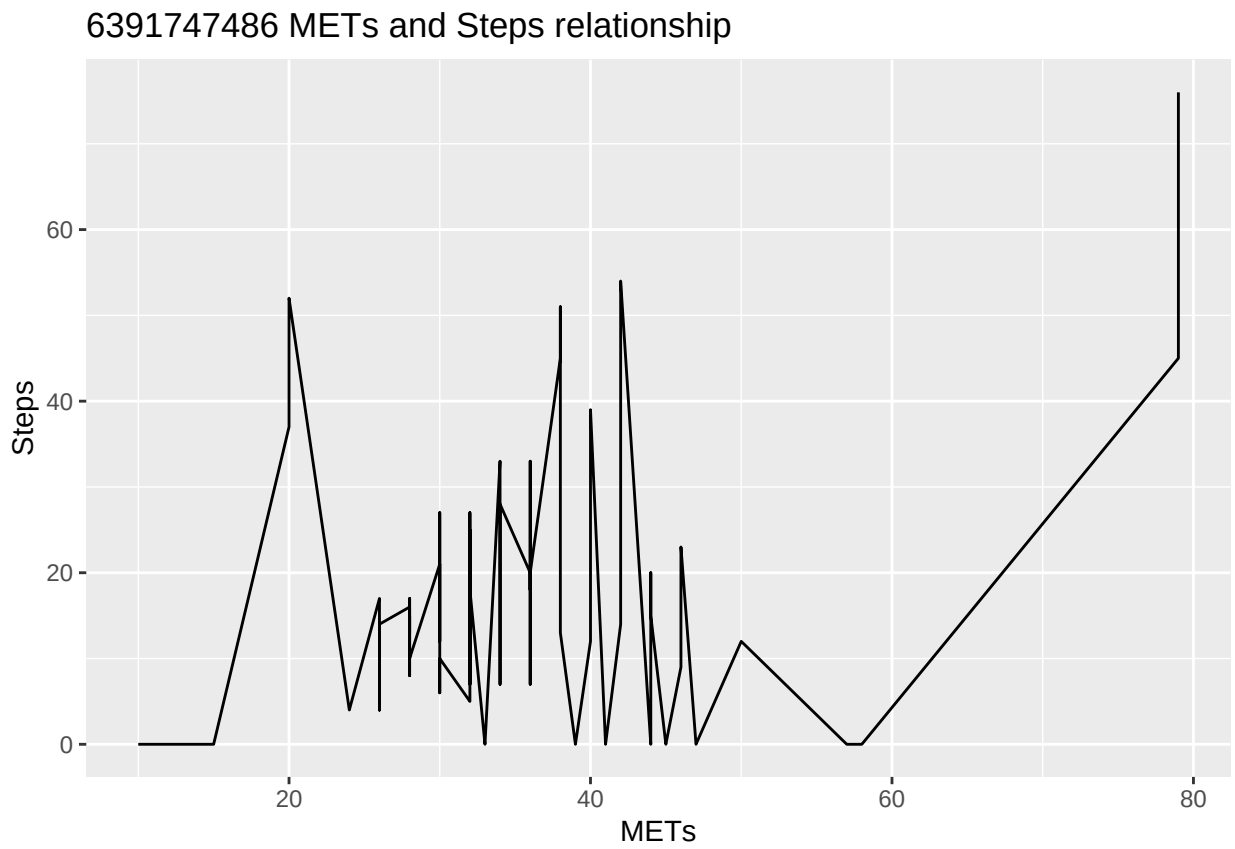


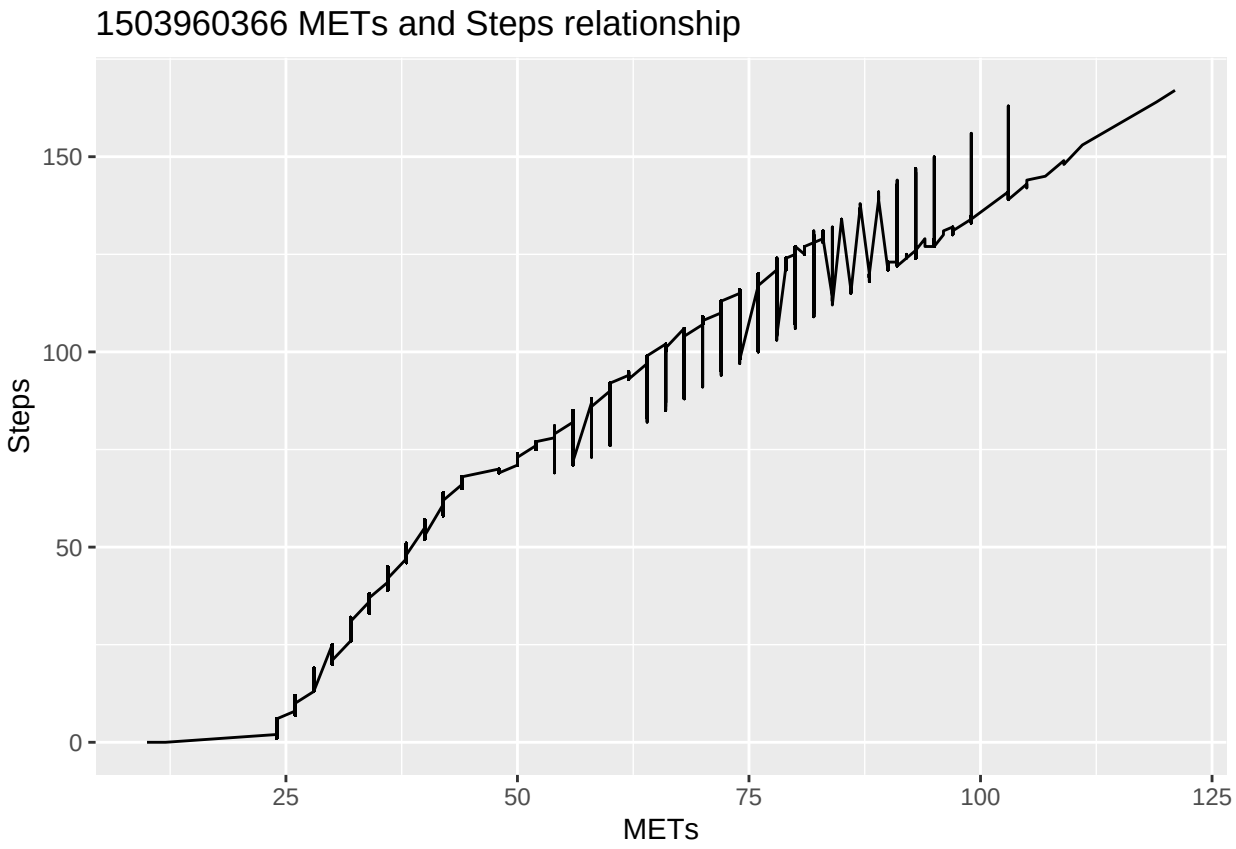

#### We can't make any sence out of this lets perfomm weighted average on it

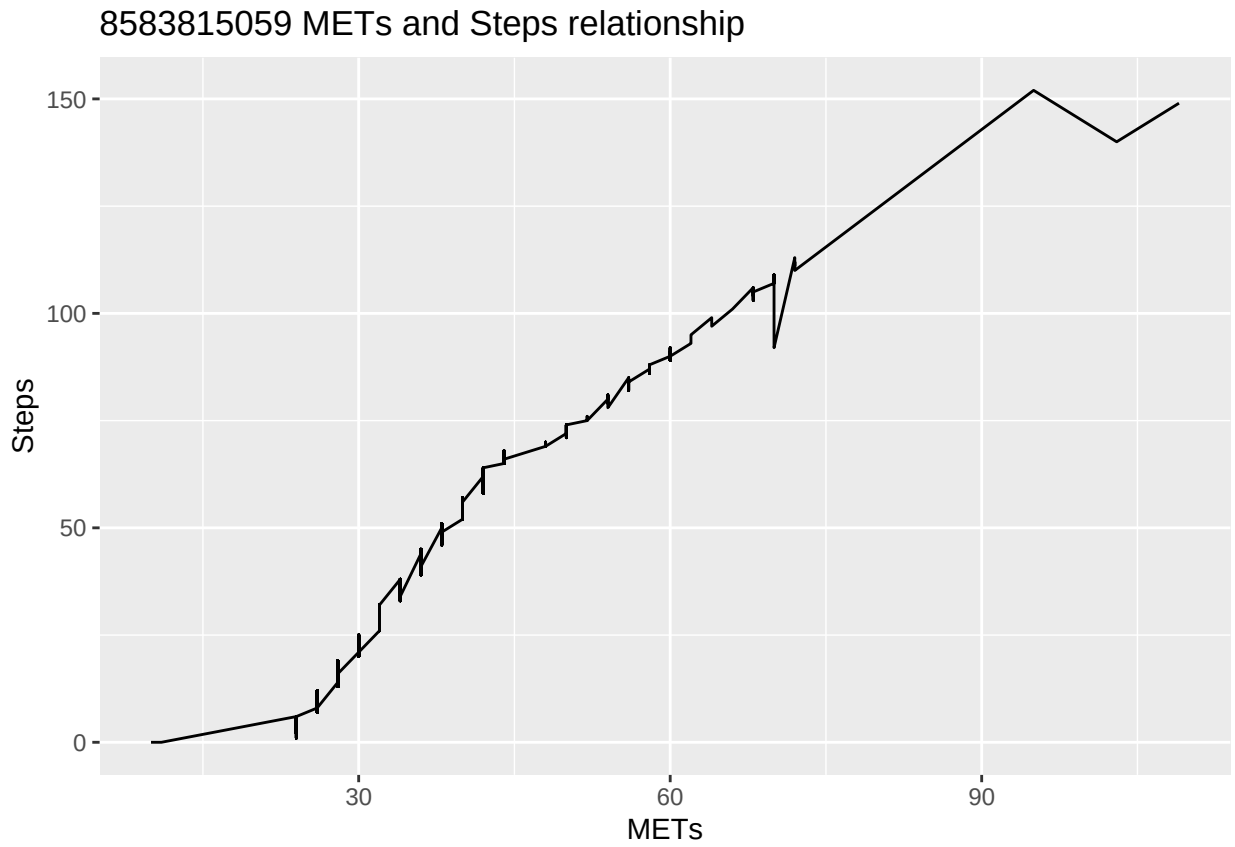
```
weighted_AVG_df <- weekDate_grouping(Minuted_Merged_df, "date")
```

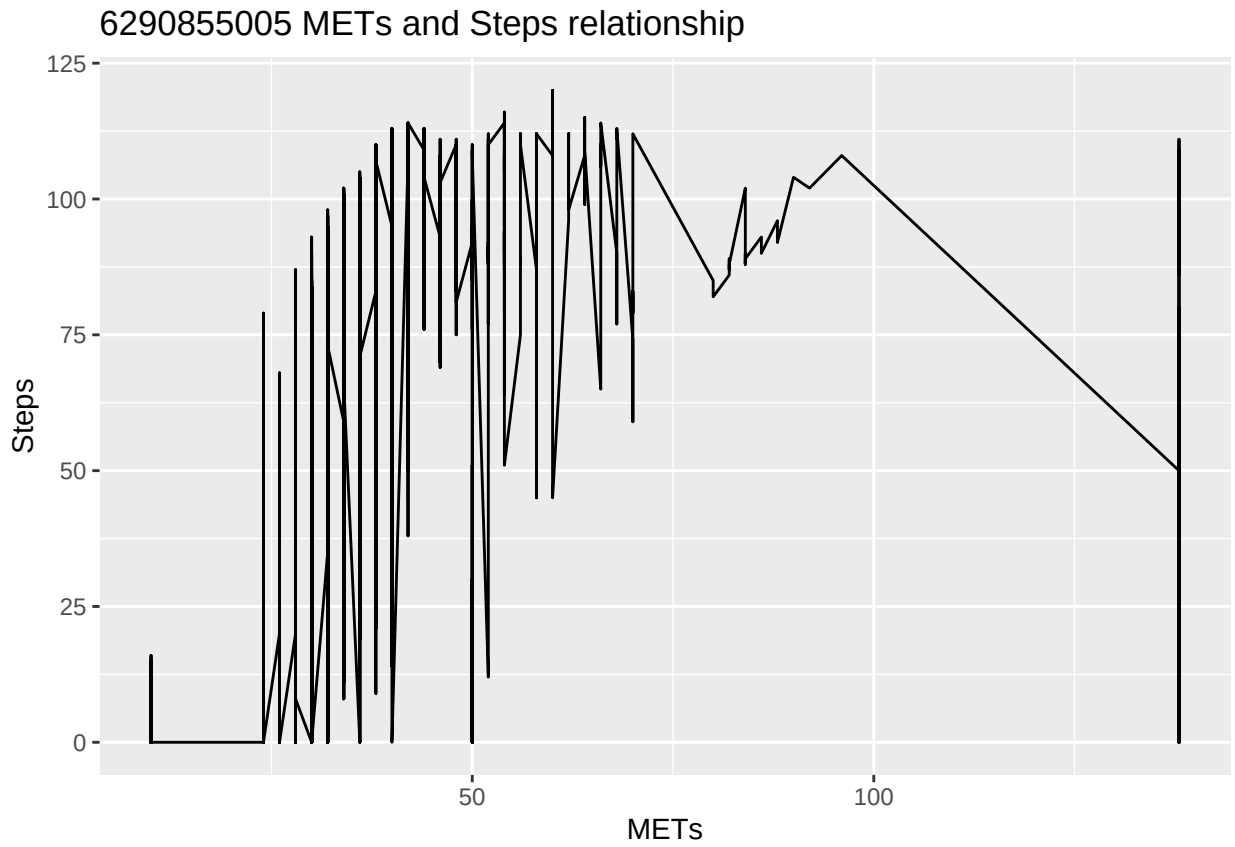
```
random_plots(df = Minuted_Merged_df,
             fraction_of_all_len_df = 0.5,
             X_col = "METs",
             Y_col = "Steps")
```

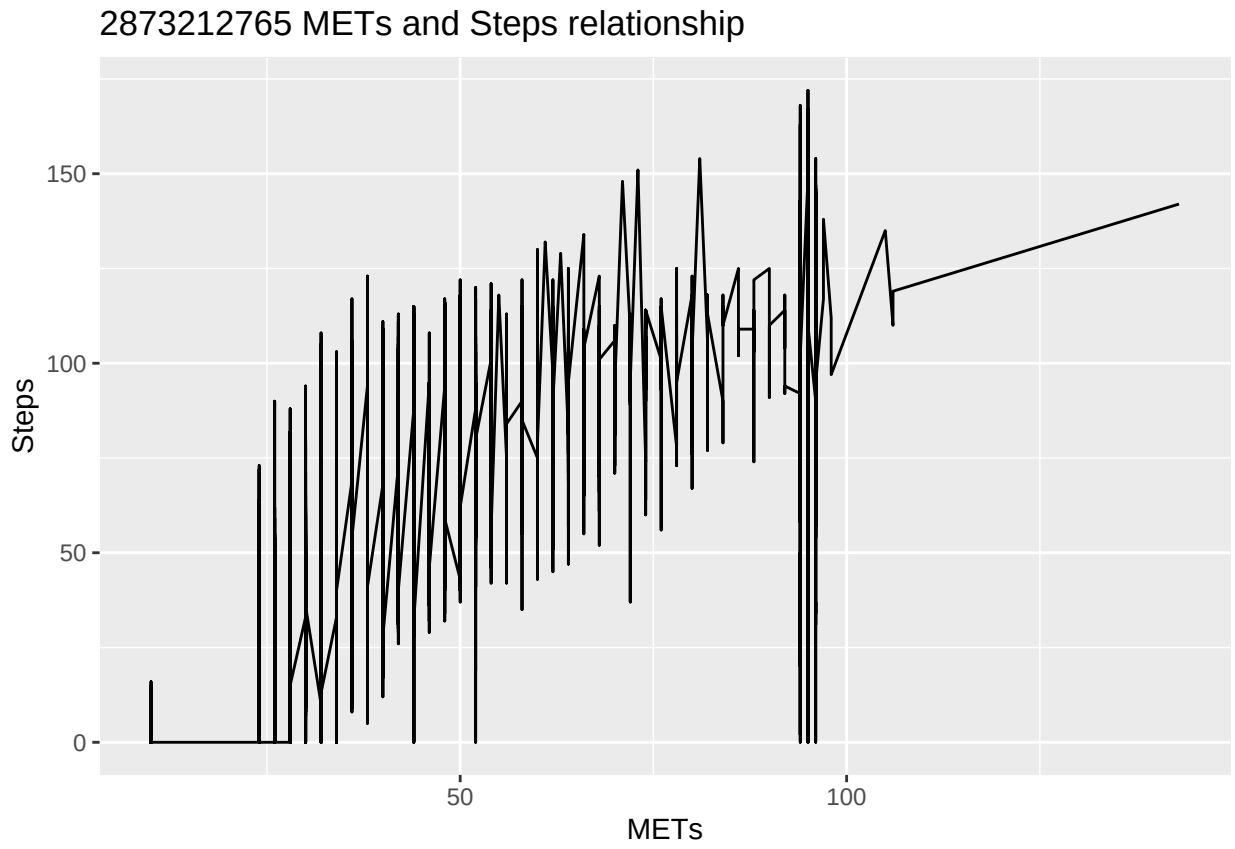

6775888955 METs and Steps relationship
120
90
60
30
0
Steps
30
60
90
120
METs

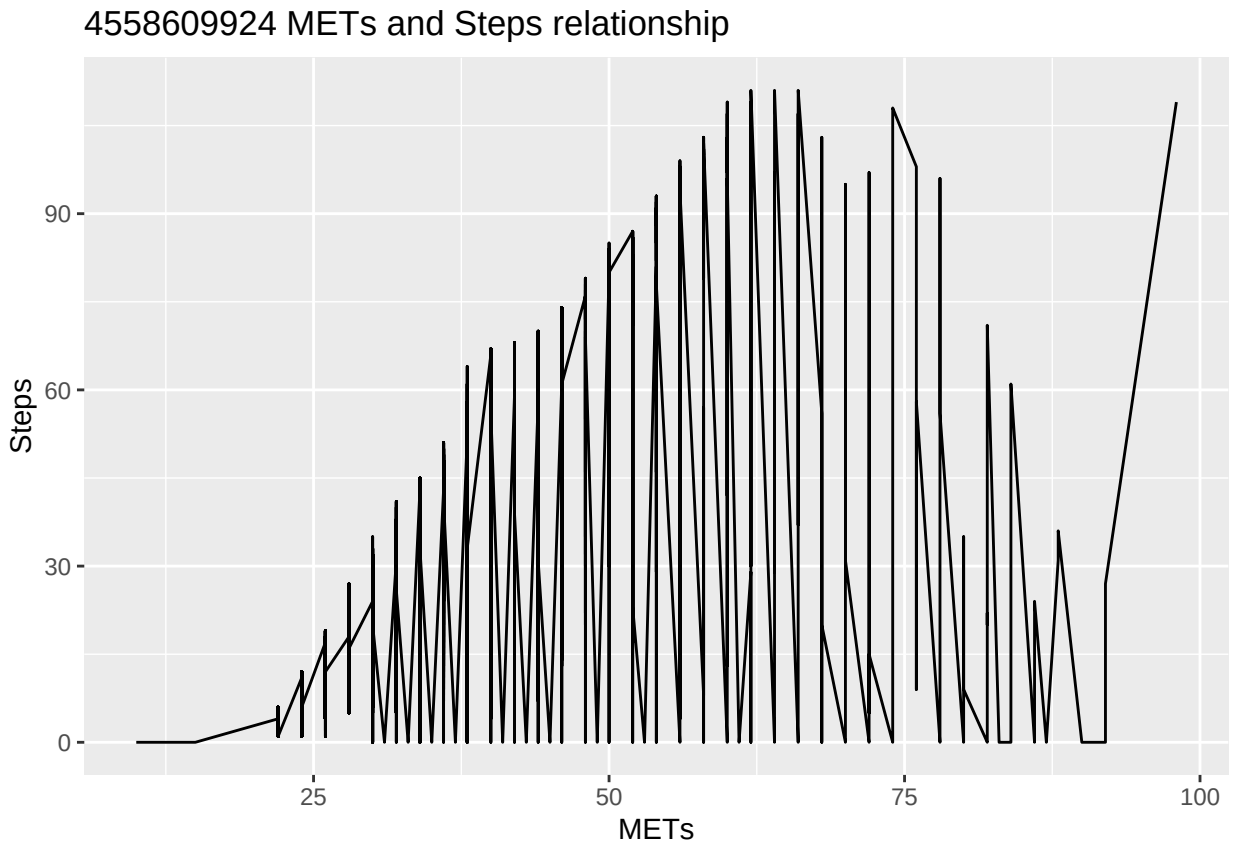

6391747486 METs and Steps relationship
Steps
0
20
40
60
20
40
60
80
METs

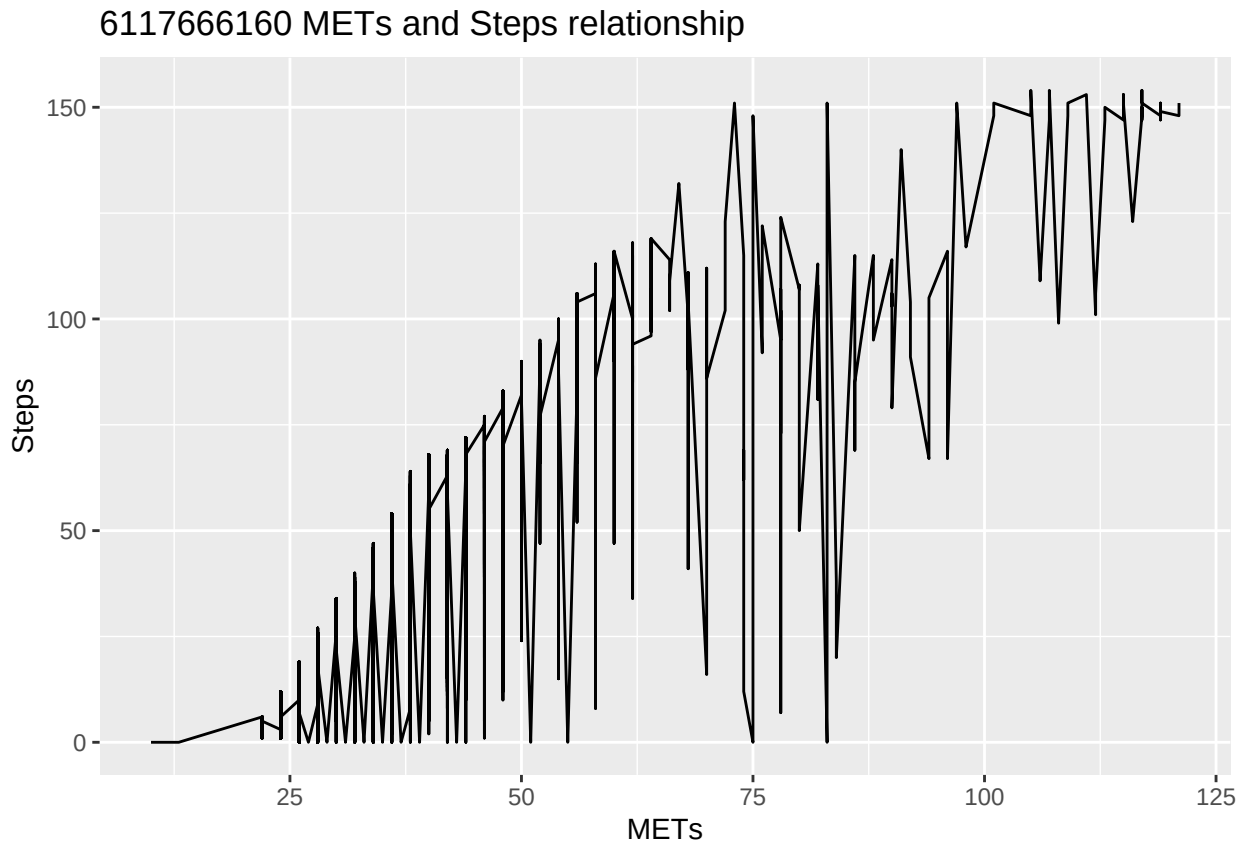

1503960366 METs and Steps relationship
Steps
150
100
50
0
25
50
75
100
125
METs

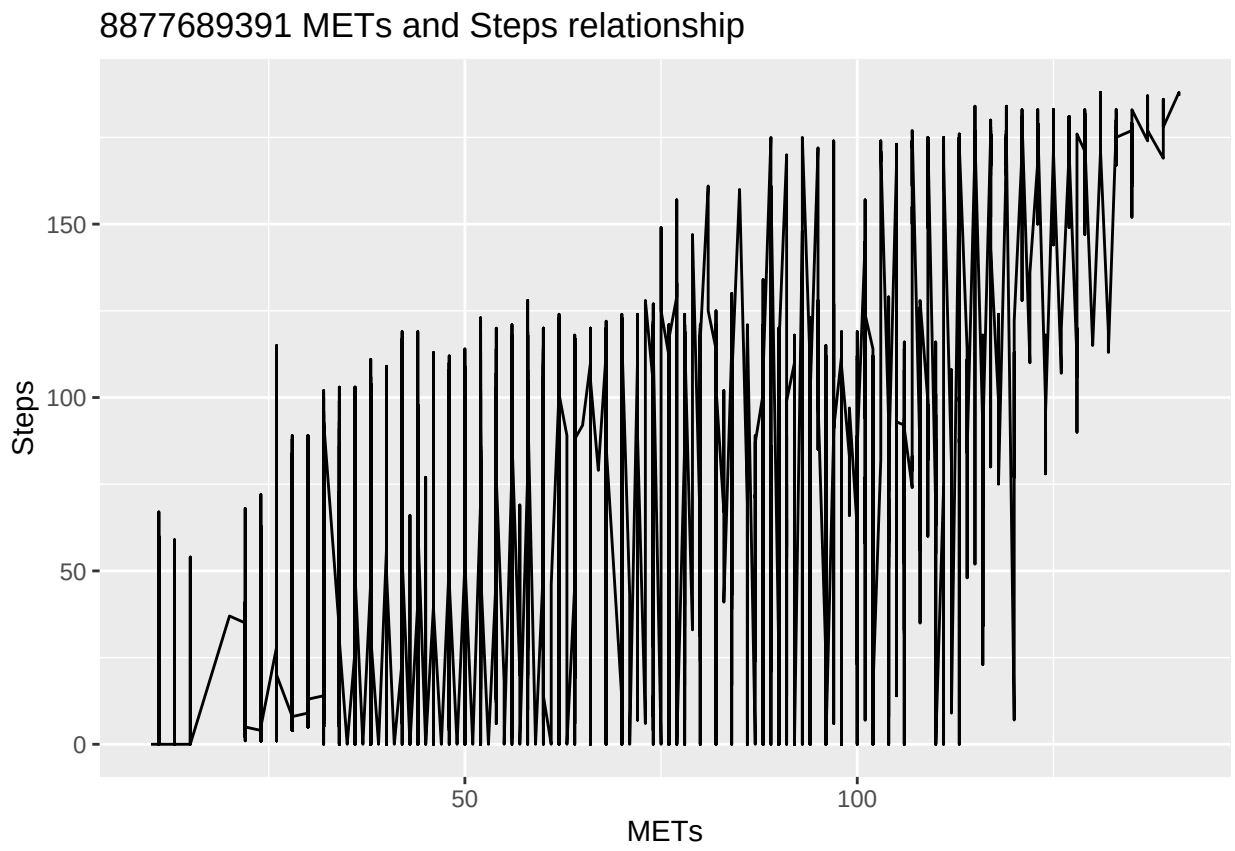

8583815059 METs and Steps relationship
150
100
50
0
Steps
30
60
90
METs

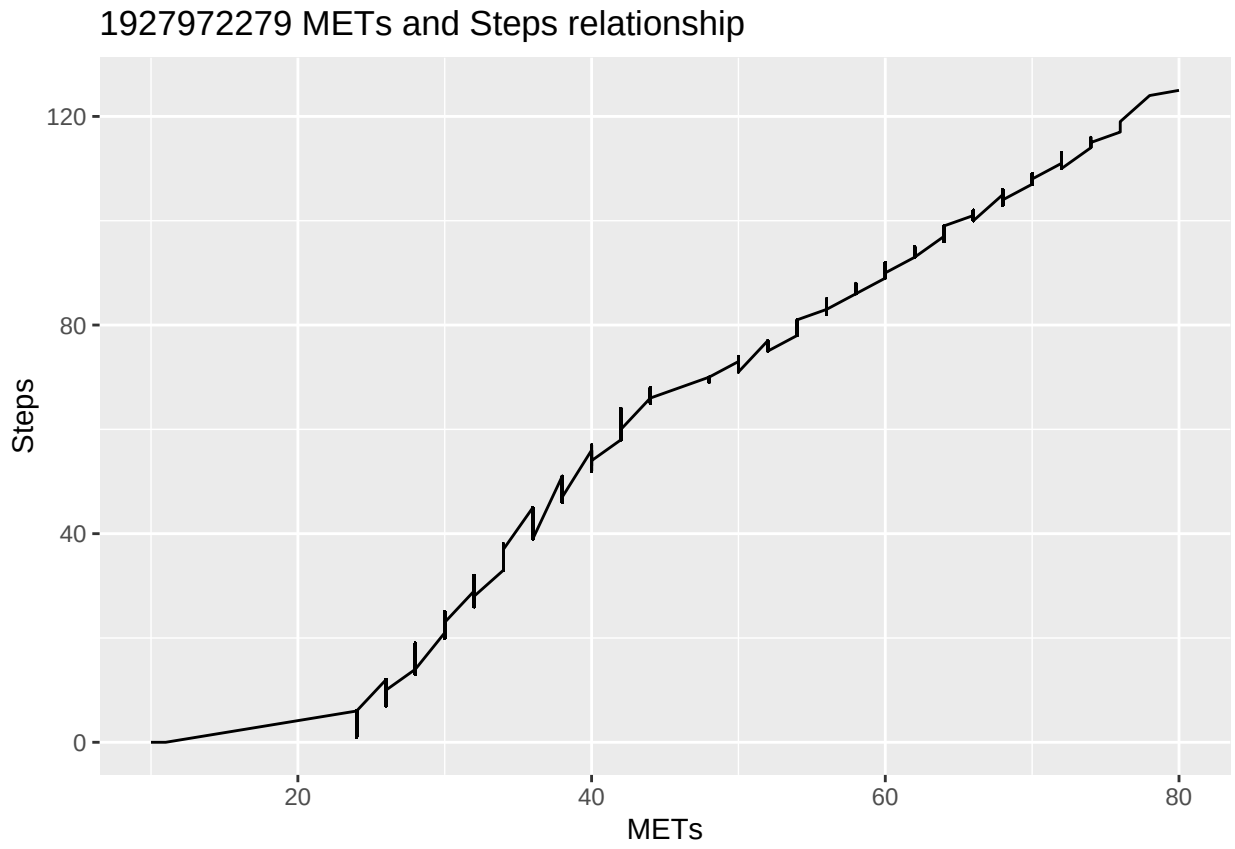

6290855005 METs and Steps relationship
125
100
75
50
25
0
Steps
50
100
METs

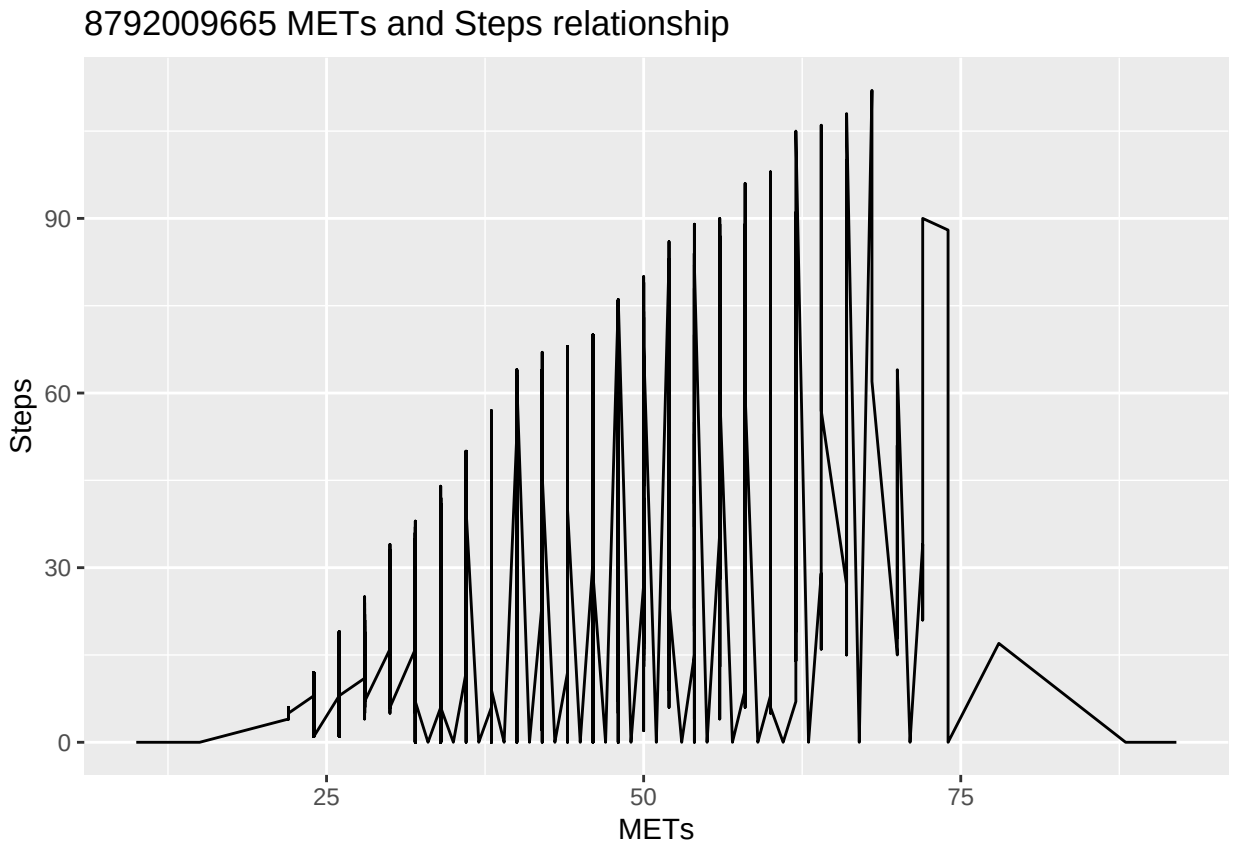

2873212765 METs and Steps relationship
Steps
150
100
50
0
50
100
METs

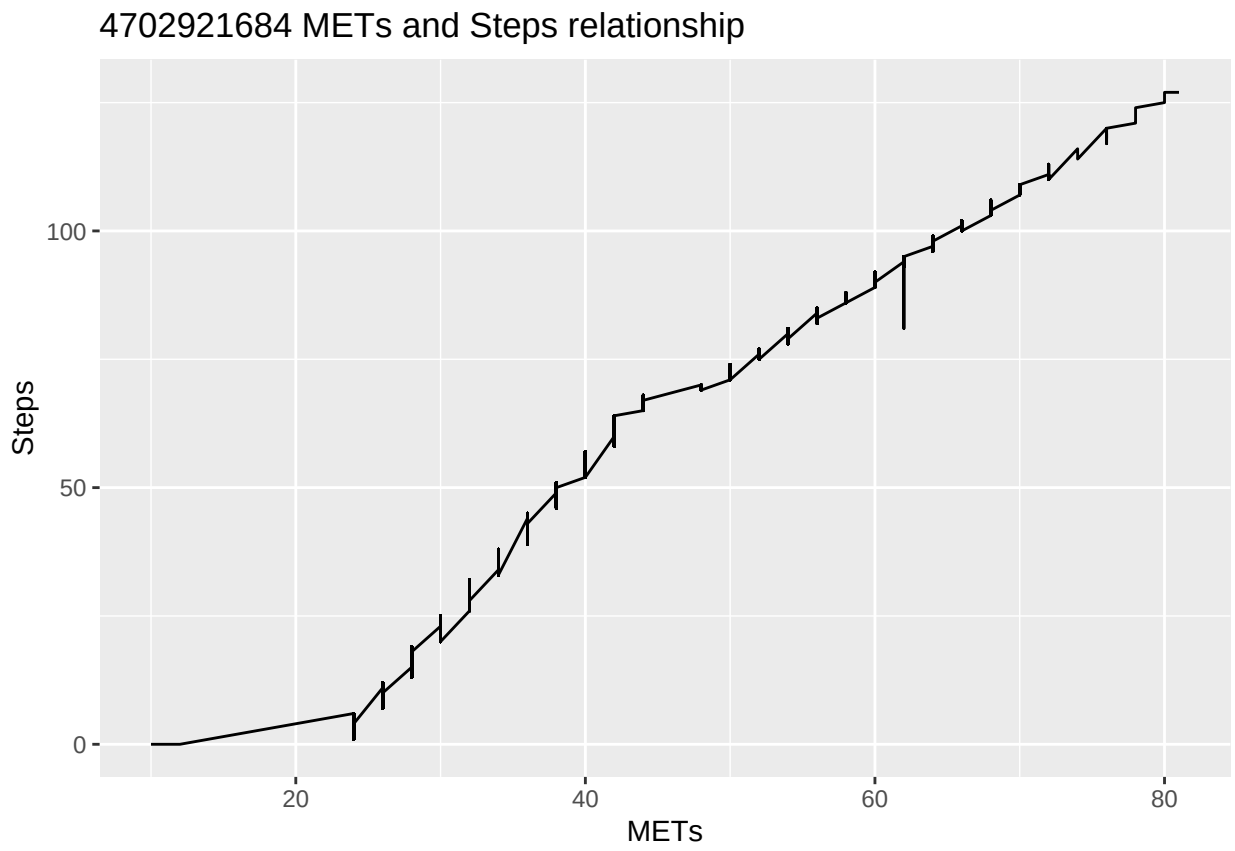

4558609924 METs and Steps relationship
Steps
90
60
30
0
25
50
75
100
METs

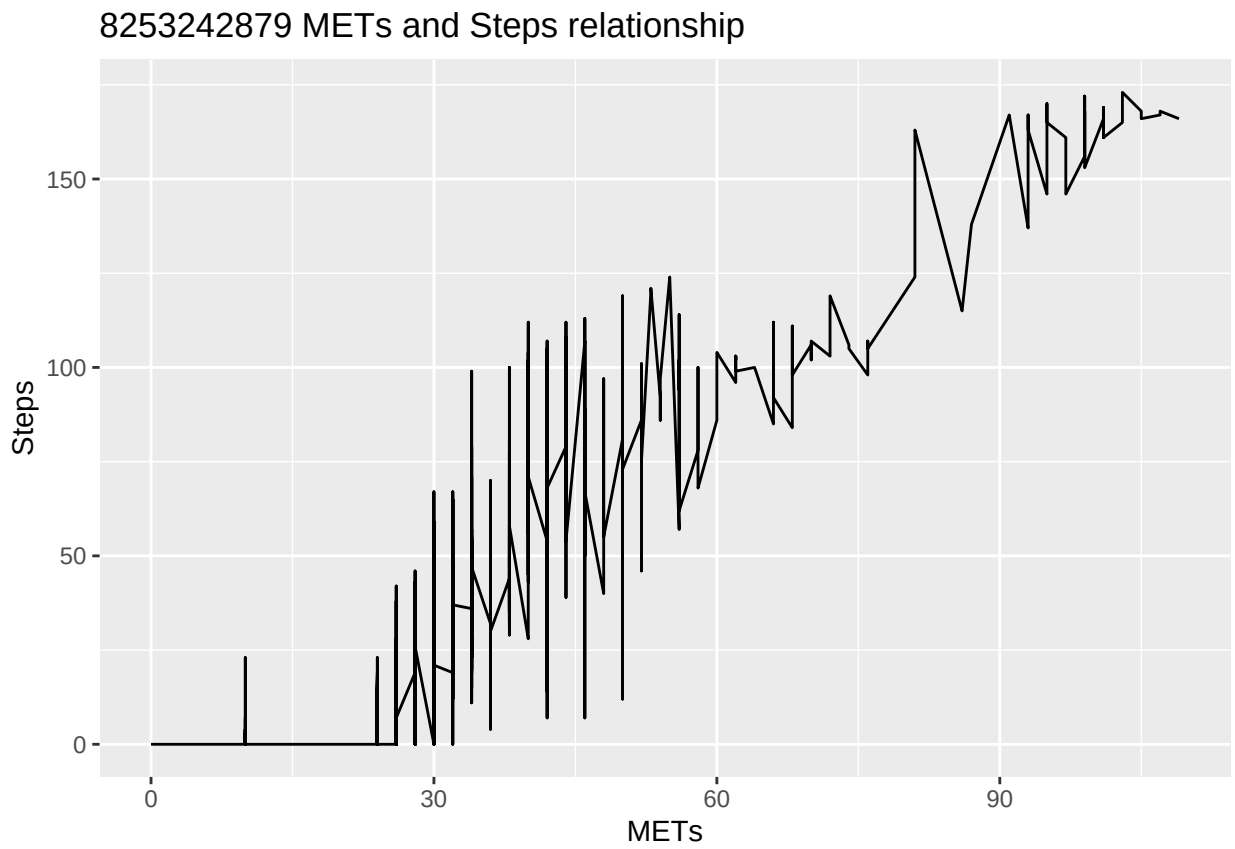

6117666160 METs and Steps relationship
Steps
150
100
50
0
25
50
75
100
125
METs

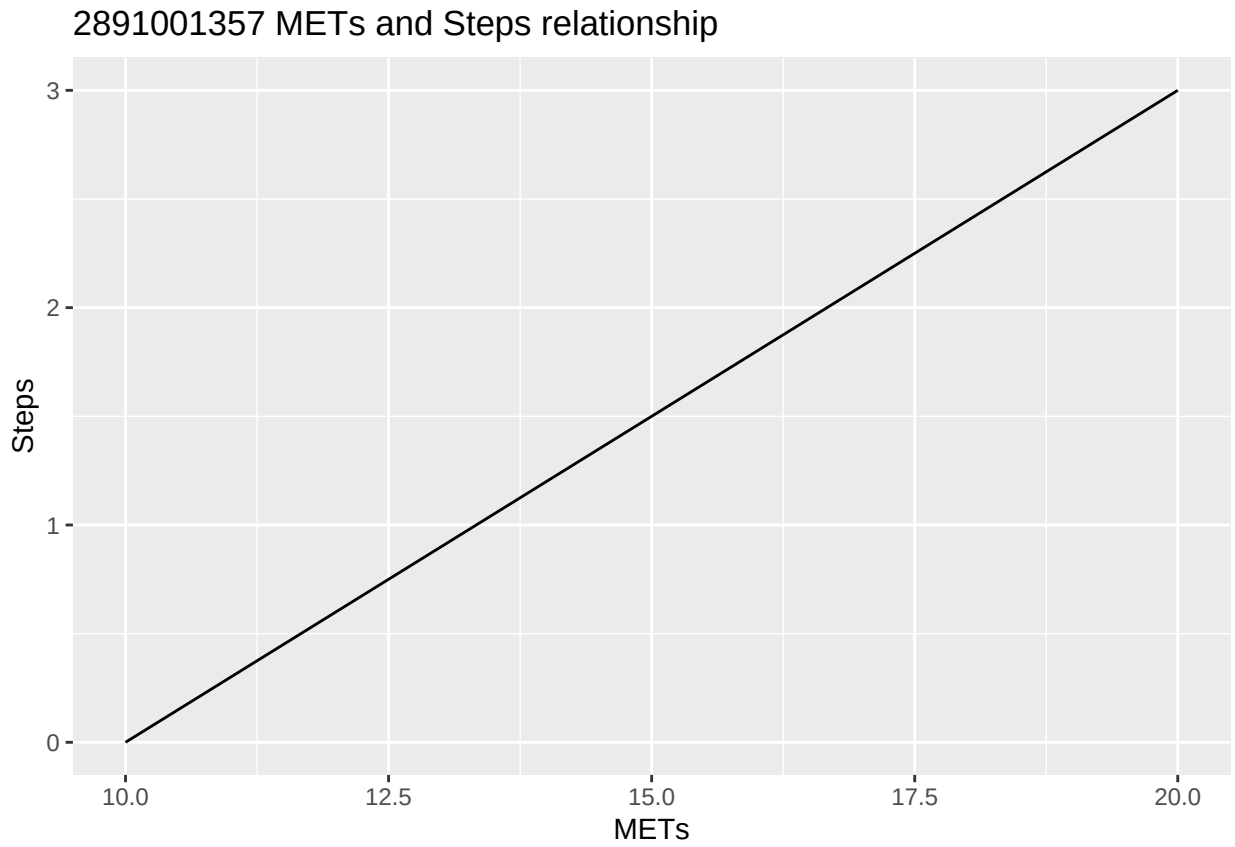

8877689391 METs and Steps relationship
Steps
150
100
50
0
50
100
METs

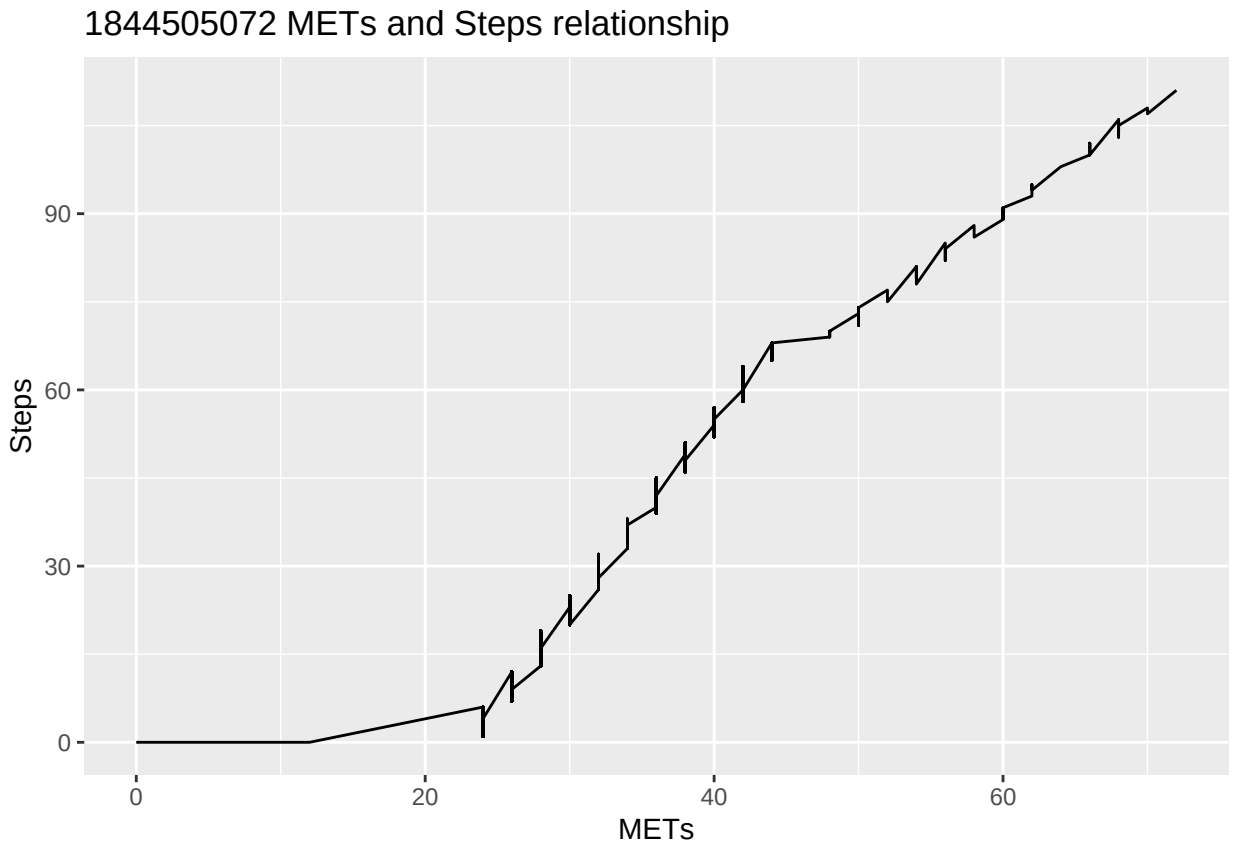

1927972279 METs and Steps relationship
Steps
120
80
40
0
20
40
60
80
METs

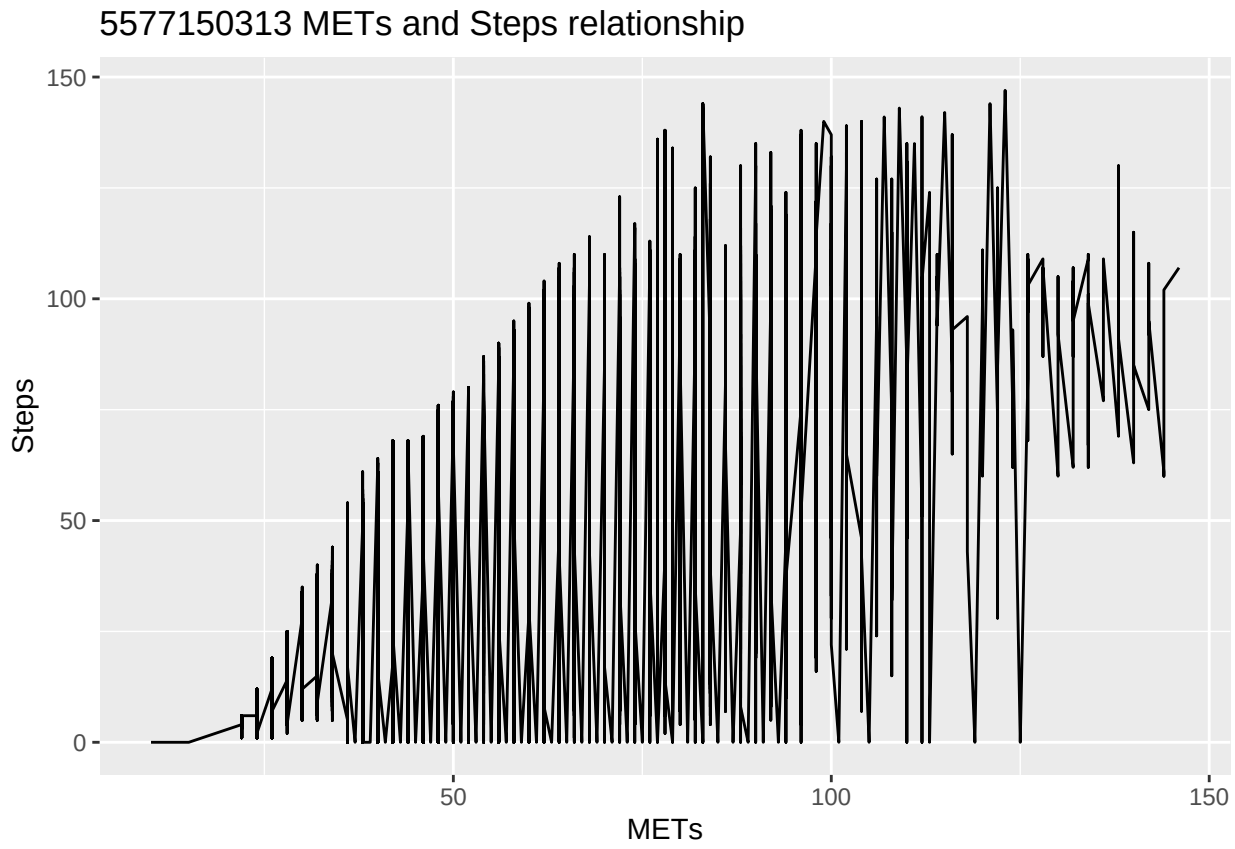

8792009665 METs and Steps relationship
Steps
90
60
30
0
25
50
75
METs

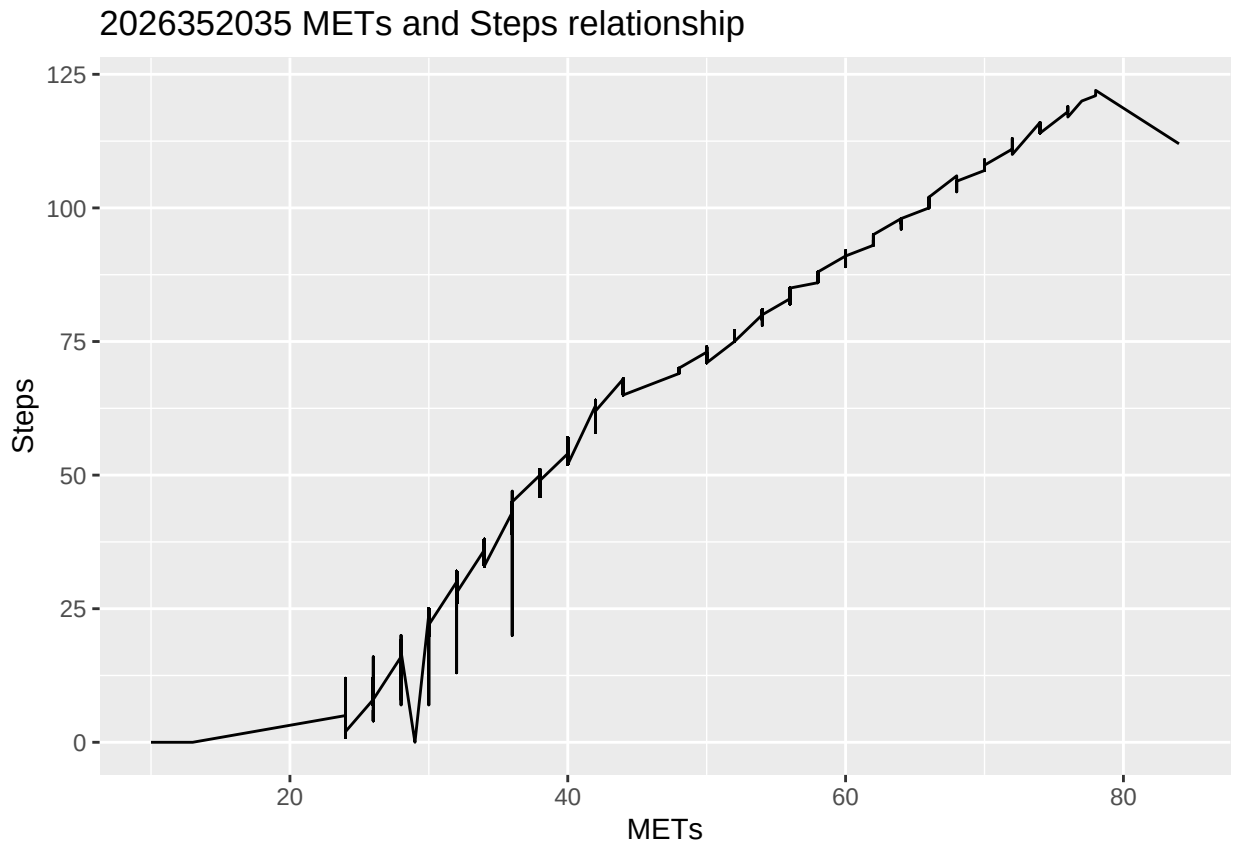

4702921684 METs and Steps relationship
Steps
100
50
0
20
40
60
80
METs


8253242879 METs and Steps relationship
Steps
150
100
50
0
0
30
60
90
METs


2891001357 METs and Steps relationship
Steps
3
2
1
0
10.0
12.5
15.0
17.5
20.0
METs


1844505072 METs and Steps relationship
Steps
90
60
30
0
0
20
40
60
METs


5577150313 METs and Steps relationship
150
100
50
0
Steps
50
100
150
METs




#### Still there seems like to be no meaningfull corelations

```
weightLogInfo_merged_df <- dfs[["weightLogInfo_merged_df"]]
head(weightLogInfo_merged_df, 5)
```

```
##           Id                 Date WeightKg WeightPounds Fat   BMI
## 1 1503960366 4/5/2016 11:59:59 PM     53.3     117.5064  22 22.97
## 2 1927972279 4/10/2016 6:33:26 PM    129.6     285.7191  NA 46.17
## 3 2347167796 4/3/2016 11:59:59 PM     63.4     139.7731  10 24.77
## 4 2873212765 4/6/2016 11:59:59 PM     56.7     125.0021  NA 21.45
## 5 2873212765 4/7/2016 11:59:59 PM     57.2     126.1044  NA 21.65
##   IsManualReport        LogId
## 1           True 1.459901e+12
## 2          False 1.460313e+12
## 3           True 1.459728e+12
## 4           True 1.459987e+12
## 5           True 1.460074e+12
```

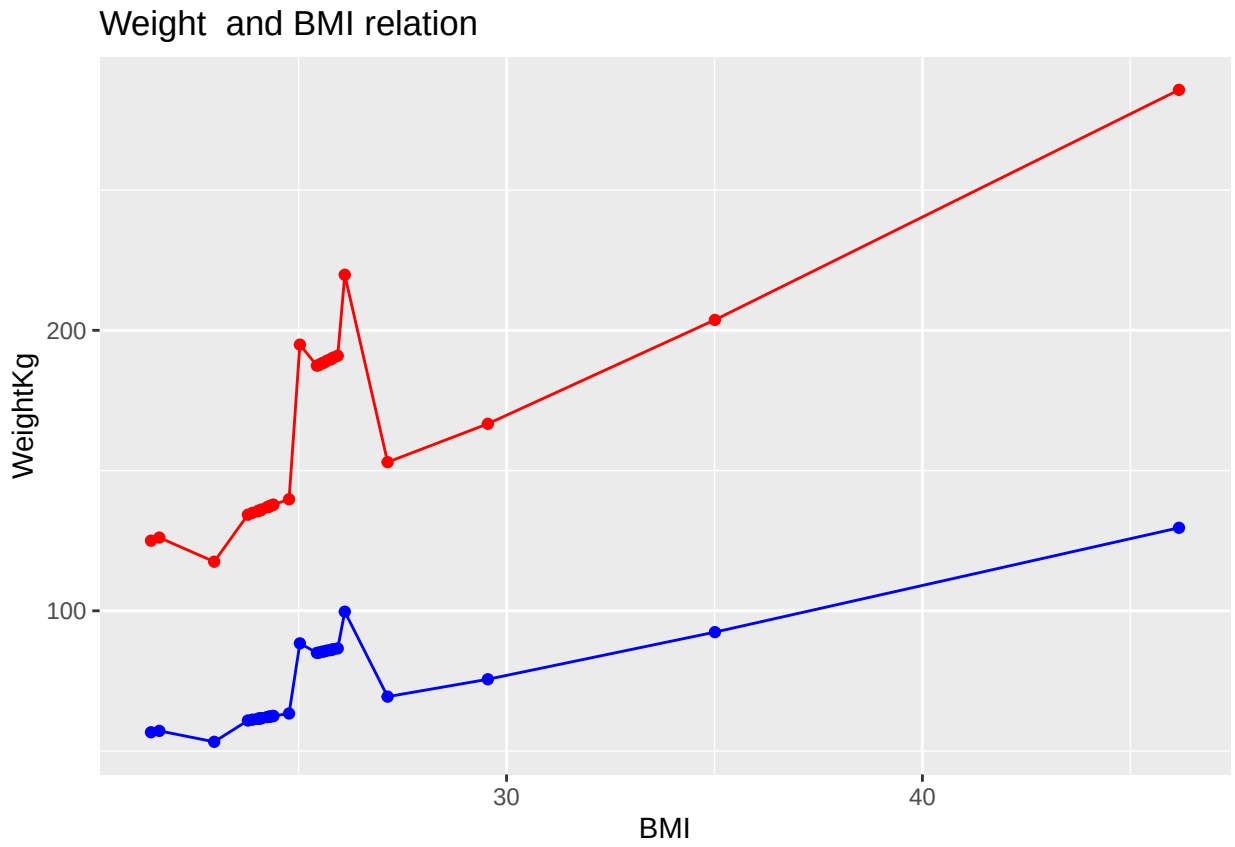
```
print(length(weightLogInfo_merged_df[['Id']]))
```

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

```
# see the correlation between weight and fat and and BMI
```

```
ggplot (data = weightLogInfo_merged_df) +
  geom_point(mapping = aes(x = BMI , y = WeightKg), color = "blue" ) +
  geom_point(mapping = aes(x = BMI , y = WeightPounds), color = "red" )+
  geom_line(mapping = aes(x = BMI, y = WeightKg), color = "blue") +
  geom_line(mapping = aes(x = BMI, y = WeightPounds), color = "red") +
```

```
labs(title = "Weight  and BMI relation", ylab = "Weight in Kg/Pounds")
```




### 0.2.2 Findings:

14.29 percent of Bellabeat devices not giving accurate data, this is significantly high.

Faulty devices have a 20% error which is rather too high to work with

#### 0.2.2.1 Recomendations:

Bella beat should improve their devices that they all give accurate data

Whereas getting a zero error in the tracker is almost not achievable but a 20 percent error in devices is quit alarming, Bellabeat should reduce the error to say 5 percent or lesser which is almost next to accurate.

### 0.2.3 Findings:

Most people hit a point of diminishing returns between intensity of 50 to 250

#### 0.2.3.1 Recomendations:

Get client's specific intensity that will make them get to the point of diminishing returns where the rate at which their exercise in impacting their calories lose in quite insignificant.

Since there is no one fits all intensity, tailor clients intensity as per their intensity based on the point where they make hit their plateau.

### 0.2.4 Findings:

Different people get to the point of diminishing returns at different points there is no concrete pattern in the number of steps that would get on to hit a plateau, however most people once they hit 2000 calories they hit a point of diminishing return

#### 0.2.4.1 Reccomendation 1:

Once a person has hit 2000 calories a day they can quite exercing since since even if they continue exercisings since the steps thereafter are quit insignificant

#### 0.2.4.2 Reccomendation 2:

clients should cover the number of steps that get them to 2000 calories.