Bellabeat_Capstone_Project

Baraka Mtana

2024-10-16

```r
# Set up environment
# import libraries
library(tidyverse)
```

```
## -- 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 error
```

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

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

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

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

```
##  [1] "Bellabeat_Capstone-project.R"
##  [2] "Bellabeat_Capstone-project.Rproj"
##  [3] "Bellabeat_Capstone_Project.html"
##  [4] "Bellabeat_Capstone_Project.log"
##  [5] "Bellabeat_Capstone_Project.tex"
##  [6] "BellabeatCapstoneProject.log"
##  [7] "BellabeatCapstoneProject.Rmd"
##  [8] "BellabeatCapstoneProject.tex"
##  [9] "BellabeatCapstoneProject_files"
## [10] "customer_averages.csv"
## [11] "LICENSE"
## [12] "mturkfitbit_export_3.12.16-4.11.16"
## [13] "README.md"
## [14] "Screenshot.png"
## [15] "UYTzmqN3SS-WCbei0buGkw_cbefaed7e1354ffb8e992c8e198906f1_Case-Study-2_-How-can-a-wellness-techn
```

```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
# 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))

  # 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] "Working on heartrate_seconds_merged.csv"
## [1] "Working on hourlyCalories_merged.csv"
## [1] "Working on hourlyIntensities_merged.csv"
## [1] "Working on hourlySteps_merged.csv"
## [1] "Working on minuteCaloriesNarrow_merged.csv"
## [1] "Working on minuteIntensitiesNarrow_merged.csv"
## [1] "Working on minuteMETsNarrow_merged.csv"
## [1] "Working on minuteSleep_merged.csv"
## [1] "Working on minuteStepsNarrow_merged.csv"
## [1] "Working on weightLogInfo_merged.csv"
```

```
# 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"
```

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

# Check for missing values
missing_value <- function(df){
  print(paste("Count of total missing values", sum(is.na(df))))
}


# 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"))

}



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 column to datetime
change_to_date <- function(df, column_of_interest){
  df[[column_of_interest]] <-
      as.POSIXct(
      df[[column_of_interest]],
      format="%m/%d/%Y" ,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], 10))


  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.")
  }
}


# calculate averages
averages <- function(df){
  averages_per_athlete <- df %>%
    group_by(Id)  %>%
      summarise(
        Avg_Steps = mean(TotalSteps),
        Avg_Distance = mean(TotalDistance),
        Avg_TrackedDistance = mean(TrackerDistance),
        Avg_LoggedActivityDistance = mean(LoggedActivitiesDistance),
        Avg_VeryActiveDistance = mean(VeryActiveDistance),
        Avg_ModeratelyActiveDistance = mean(ModeratelyActiveDistance),
        Avg_LightActiveDistance = mean(LightActiveDistance),
        Avg_SedentaryActiveDistance = mean(SedentaryActiveDistance),
        Avg_Calories = mean(Calories)
    )

  write.csv(averages_per_athlete, "customer_averages.csv", row.names = FALSE)

  print(head(averages_per_athlete, 5))
}


# line plot function
line_plot <- function(df, x_col, y_col){
  ggplot (data = df) +
    geom_line(mapping = aes(x= .data[[x_col]] , y= .data[[y_col]] )) +
    labs(title = paste(x_col, "and", y_col, "relationsip"))
}
```

```
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
# Assuming TotalSteps is cadence we'll see the average length of a step per person
# See if TotalDistance and TrackerDistance distance record the same data
# Calculate average TotalDistance, TrackerDistance, VeryActiveDistance,
# ModeratelyActiveDistance, LightActiveDistance, SedentaryActiveDistance,
# Average calories lost per day
```

Check for missing values in dailyActivity_merged_df

```
missing_value(dailyActivity_merged_df)
```

```
## [1] "Count of total missing values 0"
```

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"
```

1 see the number of character in each character of the ActivityDate column

2 since we've already seen there are inconsistencies in the ActivityDate, –>

3 let's see if there are some date characters saved with hrs,min,and secs

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

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

We have different datatypes let's see if this will affect how changing the datatype of ActivityDate

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

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

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

```
## [1] "ActivityDate has POSIXct datatype" "ActivityDate has POSIXt datatype"
##    ActivityDate
## 1    2016-03-25
## 2    2016-03-26
## 3    2016-03-27
## 4    2016-03-28
## 5    2016-03-29
## 6    2016-03-30
## 7    2016-03-31
## 8    2016-04-01
## 9    2016-04-02
## 10   2016-04-03
```

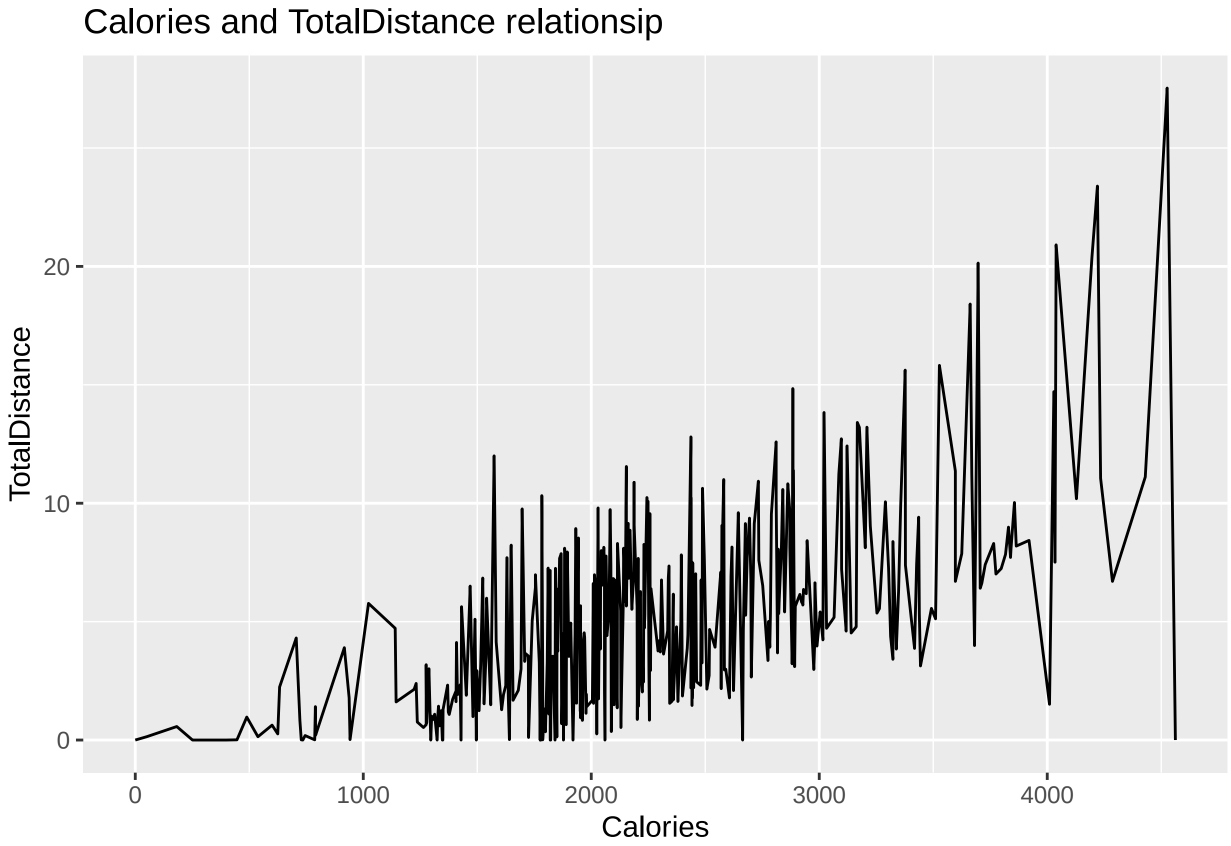
```
# call identical_columns on dailyActivity_merged_df
identical_columns(dailyActivity_merged_df, "TotalDistance", "TrackerDistance" )
```

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

```
averages(dailyActivity_merged_df)
```

```
## # A tibble: 5 x 10
##           Id Avg_Steps Avg_Distance Avg_TrackedDistance Avg_LoggedActivityDist~1
##        <dbl>     <dbl>        <dbl>               <dbl>                    <dbl>
## 1 1503960366    11641.         7.61                7.61                        0
## 2 1624580081     4226.         2.75                2.75                        0
## 3 1644430081     9275.         6.75                6.75                        0
## 4 1844505072     3641.         2.41                2.41                        0
## 5 1927972279     2181.         1.51                1.51                        0
## # i abbreviated name: 1: Avg_LoggedActivityDistance
## # i 5 more variables: Avg_VeryActiveDistance <dbl>,
## #   Avg_ModeratelyActiveDistance <dbl>, Avg_LightActiveDistance <dbl>,
## #   Avg_SedentaryActiveDistance <dbl>, Avg_Calories <dbl>
```

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



There is a positive correlation between Calories and TotalDistance the is

3.1 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 A
##  $ Value: int  93 91 96 98 100 101 104 105 102 106 ...
```

```
# 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
```

```
#Check for missing values in heartrate_seconds_merged_df
missing_value(heartrate_seconds_merged_df)
```

```
## [1] "Count of total missing values 0"
```