Final Report of Internship Program 2021

On

"ANALYSE FITNESS DATA"

Bonet Funes, Ruth María

MEDTOUREASY, NEW DELHI

January 2021

1-ACKNOWLDEGMENTS

The internship opportunity I had with MedTourEasy was a huge change in learning and understanding the complexities of data science with Python; and also, for personal and professional development. I was forced to follow a learning process that helped me get to know the working process of a real project.

I want to express my deepest appreciation and special thanks to the MedTourEasy team, who gave me the opportunity to do this internship in their organization, working in a productive and very conducive environment.

2 - INDEX

Contenido

1-ACKNOWLDEGMENTS 2

2 - INDEX 3

3 - INTRODUCTION 4

3.1- Specific tasks 5

4 - METHODS AND MATERIALS 6

4.1 - Context of the project 6

4.2 - Data collection 6

4.3 - Materials 6

4.4 - Language and platform used 6

4.4.1 – Language: Python 6

4.4.2 – Platform used: Jupyter notebook 7

5 - DATA EXPLORATION 9

5.1 - Obtain and review raw data 9

5.2 - Data preprocessing 11

5.3 - Dealing with missing values 12

5. 4 - Plot running data 14

5.5 - Running statistics 17

5.6- Visualization with averages 19

5. 7- Did the user reach their goals? 20

5. 8 - Is the user progressing? 22

5. 9 - Training intensity 23

5.10 - Detailed summary report 25

5. 11 - Fun facts 28

DISCUSSION AND FUTURE SCOPE 30

BIBLIOGRAPHY 31

3 - INTRODUCTION

According to the World Health Organization (WHO), Physical inactivity is the fourth risk factor in global mortality. An adequate level of regular physical activity (PA) in adults reduces the risk of several diseases, improves bone and functional health and is essential for caloric balance and weight control. Increasing the level of physical activity is thus a social necessity. Under normal circumstances, adults aged 18 to 64 years need to engage in at least 150 min of moderate-intensity PA or 75 min of high-intensity PA per week. In addition, to derive more health benefits, 300 min of moderate-intensity PA or 150 min of high-intensity PA per week is required. (*WHO | Physical Activity and Adults*, 2020).

This is why health promotion activities strive to find modern physical activity disclosure tools that are attractive to people. The trend of recent years in terms of physical activity has been to use technological innovations to try to reduce levels of inactivity in society.

Recent years have seen a rapid development of wearable devices such as fitness trackers, which track PA in real time. They are able to enhance users' PA levels and cultivate healthy living habits because of their superior portability and user-friendly interface and are being accepted by more and more people (Bietz et al., 2016; Ridgers et al., 2016). Fitness trackers are equipped with accurate sensors and comprehensive software systems, which not only greatly reduce the discomfort of the wearer but also promote the development of the wearers' health habits with a friendly interface. Through a systematic review, Bravata et al. (2007) revealed that fitness tracker use not only increases PA levels but also lowers the user's body mass index and blood pressure.

Health intervention strategies based on physical activity trackers, such as goal reminders, feedback on progress, healthy behavior recommendations, social encouragement, and other strategies, can lead to greater health promotion in the future and improving fitness performances (Sullivan & Lachman, 2017).

To stay physically active and in order to achieve their fitness goals, people increasingly use these types of devices and apps to be able to analyze their physical activity over time. There is nothing better to keep people motivated than looking back and seeing how far they have already come. Fitness tracking devices and apps show the progress made by the user, and so he/she can learn from what he/she has accomplished and get excited about what follows.

That is why this report will analyze the physical activity of a Runkeeper® fitness tracker user. In order to answer different questions, such as How fast, long and intense was the training today? Is there any progress? How is this performance compared to others?

Knowing the methodology to generate significant information through the analysis of data from a particular case, allows the same analysis to be extended to different people or a group of people and thus be able to know what the level of physical activity of a certain group is, for example You can well point out which activities are practiced more frequently, which are more effective, or any concerns you may have on the subject.

3.1- Specific tasks

In order to analyze the raw fitness data of this sport enthusiast, certain tasks were carried out with the purpose of obtaining significant information about his performance. Those tasks are:

1 Obtain and review raw data

2 Data preprocessing

3 Dealing with missing values

4 Plot running data

5 Running statistics

6 Visualization with averages

7 Did the user reach their goals?

8 Is the user progressing?

9 Training intensity

10 Detailed summary report

11 Fun facts

4 - METHODS AND MATERIALS

4.1 - Context of the project

This project is part of the "Data Scientist Trainee" internship program carried out by MedTourEasy Company. The main goal is to provide smart, detail-oriented data science learners to work with internal projects and the management team as they collect and review data and use the findings to optimize processes and develop more robust and effective business strategies.

4.2 - Data collection

The work presented in this project is based on the analysis of data acquired from a Runkeeper® user. The data was exported from Runkeeper® and is a CSV file. The file contains information on the user's physical activity in the period between 2012 and 2018. The file counts with 508 rows and 14 columns, each row is a single training activity, and the columns are the followings: ‘date’, 'Activity Id', 'Type', 'Route Name', 'Distance (km)', 'Duration', 'Average Pace', 'Average Speed (km/h)', 'Calories Burned', 'Climb (m)', 'Average Heart Rate (bpm)', 'Friend's Tagged', 'Notes', 'GPX File'.

4.3 - Materials

The app used for fitness tracking is the Runkeeper®, developed by the ASICS company. Runkeeper® is a GPS fitness tracker app for iOS and Android released in 2008. It tracks physical activities like walking, running, and biking using the device's GPS sensor. Once an activity is complete, the app provides basic statistics for the activity (Stahl, 2015). In this way, the user can know what their scope is and train to meet their specific objectives.

4.4 - Language and platform used

To analyze the information extracted from the runkeeper® file, it was necessary to use different data analysis tools that will be explained below.

4.4.1 – Language: Python

Python is an interpreted programming language whose philosophy emphasizes the readability of its code. It is a multi-paradigm programming language, as it supports object-oriented, imperative programming and, to a lesser extent, functional programming. It is an interpreted, dynamic and multiplatform language. Although Python was created as a

general-purpose programming language, it has a series of libraries and development environments for each of the phases of the Data Science process. This, added to its power, its open source character and its ease of learning has led it to take the lead over other languages typical of data analytics (luca-d3.com, 2020).

Python libraries are packages that have a large number of programmed functions, tools and algorithms that save a lot of programming time and with a very easy to understand structure to be able to use them(Ceja, n.d.). In this project, different libraries have been used to meet the objectives and they will be detailed below (table 1).

Library name	Principal function
Pandas	Manipulation of datasets
Matplotlib	Data visualization
statsmodels	statistic analysis

Table 1: Python libraries used for the project

4.4.2 – Platform used: Jupyter notebook

Jupyter Notebook is a client-server application launched in 2015 by the non-profit organization Jupyter Project. Create and share web documents in JSON format that follow a versioned schema and an ordered list of input and output cells. These cells contain, among other things, code, text (in Markdown format), mathematical formulas and equations, or also multimedia content (Rich Media). The program runs from the client web application that works in any standard browser.

Jupyter Notebook provides an environment designed to meet specific needs and fit into the workflow of data science and numerical simulation. In a single interface, users can write, document, and execute code, visualize data, perform calculations, and view results. Specifically, the prototyping phase includes the advantage that the code is organized in independent cells, that is, it is possible to test specific blocks of code individually (*Jupyter Notebook*, n.d.). An example of the jupyter notebook interface is shown in the figure below (figure 1).

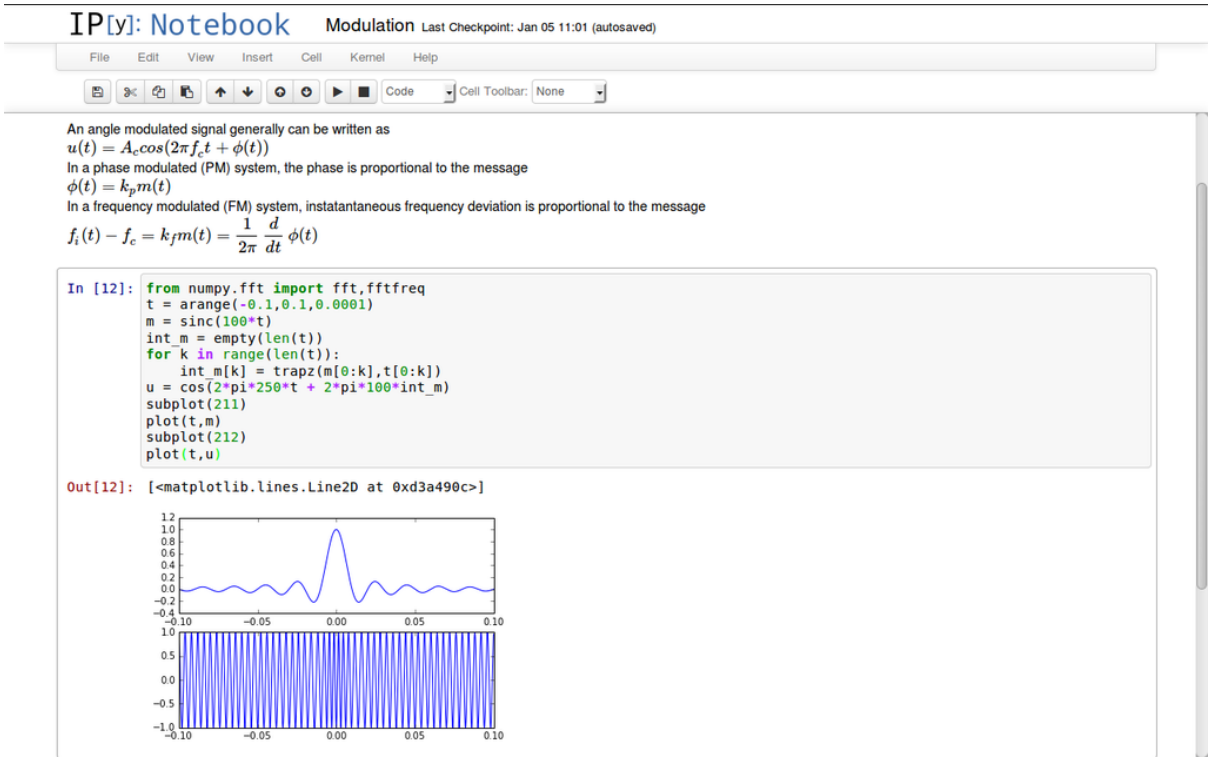

Figure 1: Example of the Jupyter notebook interface[1]

The MedTourEasy team provided a jupyter notebook file to complete this project. All data exploration and visualization was done there.

[1] Image taken from https://upload.wikimedia.org/wikipedia/commons/a/af/IPython-notebook.png

5 - DATA EXPLORATION

After having reviewed the framework of this project, its objectives as well as the materials and methods to carry it out, we will present in this section the implementation of the methodology to carry out the proposed tasks. There are eleven tasks, which will be developed in detail in order to achieve the objectives of the project and know the results.

Each task will include the following sections:

- Functions, method, attributes and/or statements used
- Task code
- Task Result

Note: the information about the functions, method, attributes and/or statements used will be taken from the official pages of pandas, matplotlib and statsmodels[2].

5.1 - Obtain and review raw data

In this step, pandas library and the data was imported as a dataframe, under the name of "df_activities", to the jupyter notebook. Next, a look at the exported data was taken, selecting a sample of 3 random rows. Finally, a resume of the information of the dataframe was shown.

- Functions, method, attributes and/or statements used:

Import statement: commonly used to import the packages and libraries needed. It was used to import the pandas library.

pandas.read_csv(): Read a comma-separated values (csv) file into DataFrame. This function has different parameters that can be modified to import the information properly. In this project, the dates were parsed with *parse_dates*, and the Date column of the data frame was considered as the index of the data frame, using the *index_col* parameter.

DataFrame.sample(): Return a random sample of items from an axis of object. Considering the project, a 3 random sample was taken to be able to see the dataframe.

DataFrame.info(): This method prints information about a DataFrame including the index dtype and columns, non-null values and memory usage.

-

[2] Pandas Official Site: https://pandas.pydata.org/
Matplotlib Official site: https://matplotlib.org/
Statsmodels Official site: https://www.statsmodels.org/

- Task 5.1 code:

```
# Import pandas
import pandas as pd

# Define file containing dataset
runkeeper_file = 'datasets/cardioActivities.csv'

# Create DataFrame with parse_dates and index_col parameters
df_activities = pd.read_csv(runkeeper_file, parse_dates = True, index_col = 'Date')

# First look at exported data: select sample of 3 random rows
display(df_activities.sample(3))

# Print DataFrame summary
print(df_activities.info())
df_activities.columns
```

Figure 2: task 5.1 code

- Task 5.1 result:

Date	Activity Id	Type	Route Name	Distance (km)	Duration	Average Pace	Average Speed (km/h)	Calories Burned	Climb (m)	Average Heart Rate (bpm)	Friend's Tagged	Notes	GPX File
2018-07-03 18:00:05	e91eba7e-dd4c-4ae4-8909-c35dc6e2debb	Running	NaN	18.75	1:41:30	5:25	11.08	1332.000000	300	139.0	NaN	TomTom MySports Watch	2018-07-03-180005.gpx
2016-09-04 17:08:28	cc9e2d95-fc5d-4633-9801-a6a92df8eebe	Cycling	NaN	13.64	40:43	2:59	20.09	342.000000	184	134.0	NaN	TomTom MySports Watch	2016-09-04-170828.gpx
2016-06-27 18:40:44	20ef6c75-37e5-48cc-ab4a-934347347632	Running	NaN	13.28	1:12:43	5:29	10.96	903.999999	180	146.0	NaN	TomTom MySports Watch	2016-06-27-184044.gpx

Figure 3: Result of the sample method applied to the DataFrame

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 508 entries, 2018-11-11 14:05:12 to 2012-08-22 18:53:54
Data columns (total 13 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Activity Id              508 non-null    object
 1   Type                     508 non-null    object
 2   Route Name               1 non-null      object
 3   Distance (km)            508 non-null    float64
 4   Duration                 508 non-null    object
 5   Average Pace             508 non-null    object
 6   Average Speed (km/h)     508 non-null    float64
 7   Calories Burned          508 non-null    float64
 8   Climb (m)                508 non-null    int64
 9   Average Heart Rate (bpm) 294 non-null    float64
 10  Friend's Tagged          0 non-null      float64
 11  Notes                    231 non-null    object
 12  GPX File                 504 non-null    object
dtypes: float64(5), int64(1), object(7)
memory usage: 55.6+ KB
None
```

Figure 4: info() method result, containing all the relevant information about the DataFrame

Figure 4 shows the result of the info() method. It is possible to notice there, that there are some columns that have missing values. It is not possible to know with certainty what is the reason for these values, because each case is different. However, we can notice that the column "Tagged Friend" is completely empty and does not provide important information. A similar situation occurs with the “Route Name”, only one row is not null, so it does not provide useful information. Considering the case of the “Notes” column, it is an optional field that was sometimes left blank. Finally, we found that the “Average Heart Rate (bpm)” column also have many missing values, which may be due to the fact that the heart rate was not always recorded.

As we can imagine, missing values have different reasons and each has to be treated differently.

5.2 - Data preprocessing

In section 5.1 it was notable that is necessary to do something with the missing values, so in this section we will:

- Remove columns not useful for our analysis: the columns that will be erased are 'Friend's Tagged', 'Route Name', 'GPX File', 'Activity Id', 'Calories Burned', 'Notes'
- Count types of training activities: The amount of activities carried out of each type will be counted.
- Count missing values for each column

- Functions, method, attributes and/or statements used:

DataFrame.drop(): Remove rows or columns by specifying label names and corresponding axis, or by specifying directly index or column names.

DataFrame.value_counts(): Return a Series containing counts of unique rows in the DataFrame.

DataFrame.isnull().sum(): isnull() determine if a value is null and sum() counts them.

str.replace(): replaces a specified phrase with another specified phrase.

- Task 5.2 code:

```
# Define list of columns to be deleted
cols_to_drop = ['Friend\'s Tagged','Route Name','GPX File','Activity Id','Calories Burned', 'Notes']

# Delete unnecessary columns
df_activities.drop(cols_to_drop,axis = 1, inplace = True)

# Count types of training activities
display(df_activities['Type'].value_counts())

# Rename 'Other' type to 'Unicycling'
df_activities['Type'] = df_activities['Type'].str.replace('Other','Unicycling')

# Count missing values for each column
print(df_activities.isnull().sum())
```

Figure 5: Task 5.2 code

- Task 5.2 result:

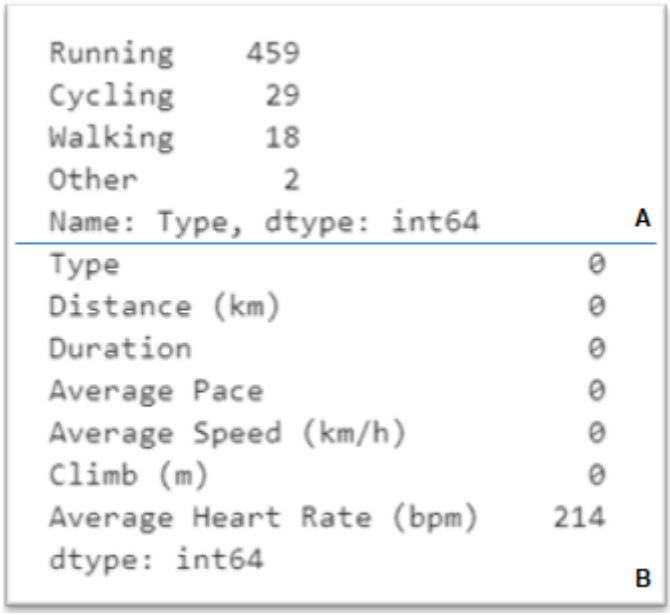

Figure 6: A) result of the count of each activity. B) Show the columns of the dataframe and the count of missing values

As is shown in figure 6.A running is the activity that was practiced the most, but we notice also that there a Type considered as “Other”, that we will replace for “Unicycling” because that was always the "Other" activity. The method used for this action was *str.replace()* that was explained before.

5.3 - Dealing with missing values

After preprocessing the data and counting the missing values, now it is necessary to take care of them. As we can see from figure 6.B, there are 214 missing entries for my average heart rate. As we do not have the values that have not been taken, what will be done is to

fill in the missing values with the general average of heart rate. This process is called "mean imputation". When imputing the mean to fill in missing data, we need to consider that the average heart rate varies for different activities (e.g., walking vs. running). We'll filter the DataFrame by activity type (Type) and calculate each activity's mean heart rate, and then fill in the missing values with those means.

After doing the process, the missing values will be counted again in the new DataFrame that counts the running activity (df_run) to confirm that know there are not missing values left.

- Functions, method, attributes and/or statements used:

DataFrame.mean(): Return the mean of the values over the requested axis. It is used to calculate the mean of the heart rate of each activity.

DataFrame.copy(): Make a copy of this object's indices and data. After filtering the data separating the 3 main activities (running, cycling, walking), the copy is created with this method.

DataFrame.fillna(): Fill missing values using with an specific value.

Isnull().sum(): explained in section 5.2.

- Task 5.3 code:

```
# Calculate sample means for heart rate for each training activity type
avg_hr_run = df_activities[df_activities['Type'] == 'Running']['Average Heart Rate (bpm)'].mean()
avg_hr_cycle = df_activities[df_activities['Type'] == 'Cycling']['Average Heart Rate (bpm)'].mean()

# Split whole DataFrame into several, specific for different activities
df_run = df_activities[df_activities['Type'] == 'Running'].copy()
df_walk = df_activities[df_activities['Type'] == 'Walking'].copy()
df_cycle = df_activities[df_activities['Type'] == 'Cycling'].copy()

# Filling missing values with counted means
df_walk['Average Heart Rate (bpm)'].fillna(110, inplace=True)
df_run['Average Heart Rate (bpm)'].fillna(int(avg_hr_run), inplace=True)
df_cycle['Average Heart Rate (bpm)'].fillna(int(avg_hr_cycle),inplace = True)

# Count missing values for each column in running data
print(df_run.isnull().sum())
```

Figure 7: Task 5.3 code

```
Type                          0
Distance (km)                 0
Duration                      0
Average Pace                  0
Average Speed (km/h)          0
Climb (m)                     0
Average Heart Rate (bpm)      0
dtype: int64
```

Figure 8: Result of the isnull().sum() method

Figure 8 shows how there are not missing values in the df_run dataframe.

5. 4 - Plot running data

As noted in the previous sections, the most practiced physical activity was running, which is why the analysis will focus on this activity.

Having already cleaned and filtered data, in this section we will plot the different running metrics for the period between 2013-2018.

An excellent first visualization is a figure with four subplots, one for each running metric (each numerical column). Each subplot will have a different y-axis, which is explained in each legend. The x-axis –Date- is shared among all subplots.

- Functions, method, attributes and/or statements used:

matplotlib.pyplot: is a state-based interface to matplotlib. pyplot is mainly intended for interactive plots and simple cases of programmatic plot generation.

DataFrame.loc Access a group of rows and columns by label(s) or a boolean array.

- Task 5.4 code:

```
%matplotlib inline

# Import matplotlib, set style and ignore warning
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
plt.style.use('ggplot')
warnings.filterwarnings(
    action='ignore', module='matplotlib.figure', category=UserWarning,
    message=('This figure includes Axes that are not compatible with tight_layout, so results might be incorrect.')
)

# Prepare data subsetting period from 2013 till 2018
runs_subset_2013_2018 = df_run.sort_index()

runs_subset_2013_2018 =  runs_subset_2013_2018.loc["2013":"2018"]

# Create, plot and customize in one step
runs_subset_2013_2018.plot(subplots=True,
                           sharex=False,
                           figsize=(12,16),
                           linestyle='none',
                           marker='o',
                           markersize=3,
                          )

# Show plot
plt.show()
```

Figure 9: Coding section of the 4 task.

- Task 5.4 result:

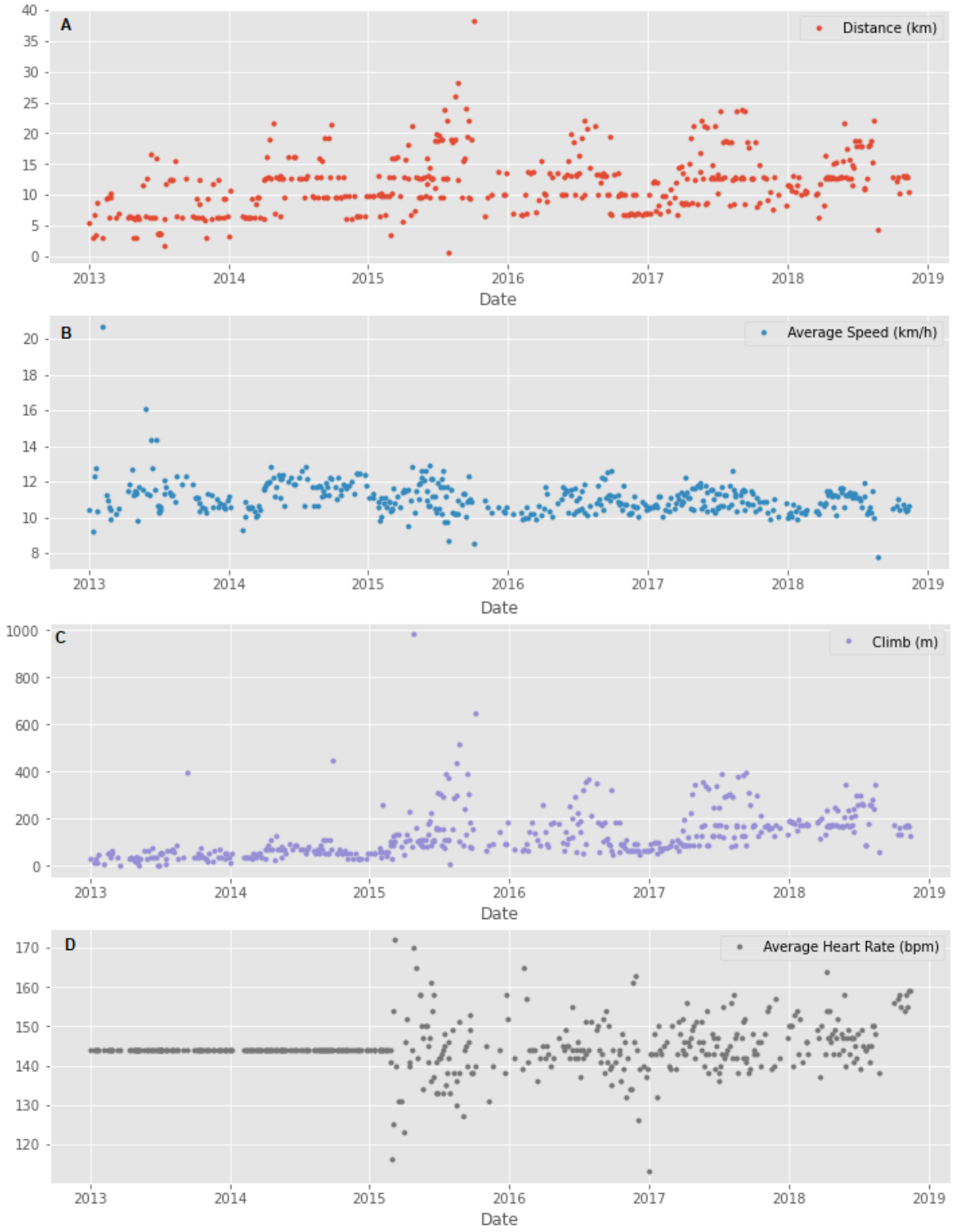

Figure 10: This graph represents the different metrics over time of the running activity. A) distance traveled (km) B) Average Speed (m). C) Climb (m). D) Average heart rate (bpm).

Figure 9 shows the code and figure 10 the resulting plot. It is noticeable how before 2015, the user didn't measure the heart rate (figure 10.D).

5.5 - Running statistics

What is your average distance? How fast do you run? Do you measure your heart rate? How often do you train? Those are some of the questions that a runner can ask himself and that kind of question we can answer exploring our data with statistics. It was already shown that the heart rate wasn't record until 2015 (figure 10.D) so for the present task it will only be used the period between 2015-2018. Annual and weekly means for Distance (km), Average Speed (km/h), Climb (m) and Average Heart Rate (bpm) will be calculated.

- Functions, method, attributes and/or statements used:

DataFrame.resample() Resample time-series data. Convenience method for frequency conversion and resampling of time series.

DataFrame.Count() Count non-NA cells for each column or row.

DataFrame.Mean() explained in section 5.3

- Task 5.5 code:

```
# Prepare running data for the last 4 years
runs_subset_2015_2018 = df_run.sort_index()
runs_subset_2015_2018 = runs_subset_2015_2018.loc["2015":"2018"]

# Calculate annual statistics
print('How my average run looks in last 4 years:')
display(runs_subset_2015_2018.resample('A').mean())

# Calculate weekly statistics
print('Weekly averages of last 4 years:')
display(runs_subset_2015_2018.resample('W').mean())

# Mean weekly counts
weekly_counts_average = runs_subset_2015_2018.resample('W')['Distance (km)'].count().mean()
print('How many trainings per week I had on average:', weekly_counts_average)
```

Figure 11: Coding from task 5.5

- Task 5.5 result:

A

How my average run looks in last 4 years:

Date	Distance (km)	Average Speed (km/h)	Climb (m)	Average Heart Rate (bpm)
2015-12-31	13.602805	10.998902	160.170732	143.353659
2016-12-31	11.411667	10.837778	133.194444	143.388889
2017-12-31	12.935176	10.959059	169.376471	145.247059
2018-12-31	13.339063	10.777969	191.218750	148.125000

B

Weekly averages of last 4 years:

Date	Distance (km)	Average Speed (km/h)	Climb (m)	Average Heart Rate (bpm)
2015-01-04	9.780000	11.120000	51.0	144.0
2015-01-11	NaN	NaN	NaN	NaN
2015-01-18	9.780000	11.230000	51.0	144.0
2015-01-25	NaN	NaN	NaN	NaN
2015-02-01	9.893333	10.423333	58.0	144.0
...	...	...	...	...
2018-10-14	12.620000	10.840000	146.5	157.5
2018-10-21	10.290000	10.410000	133.0	155.0
2018-10-28	13.020000	10.730000	170.0	154.0
2018-11-04	12.995000	10.420000	170.0	156.5
2018-11-11	11.640000	10.535000	149.0	159.0

202 rows × 4 columns

C

How many trainings per week I had on average: 1.5

Figure 12: Result from task 5.5. A) Annual average run in the period 2015-2018. B) Weekly average run in the period 2015-2018. C) Average number of trainings per week.

In figure 12.A it can be seen that the user has similar metrics over the years. In figure 12.B, where we find the weekly statistics of the user running sessions, we find many missing values, that is because the user did not exercise every week between 2015-2018 and that is

why we have no value. Finally, in Figure 12.C, we find that the user runs 1.5 times a week on average. Not that bad!

5.6- Visualization with averages

Let's plot the long term averages of my distance run and my heart rate with their raw data to visually compare the averages to each training session. Again, we'll use the data from 2015 through 2018.

- Functions, method, attributes and/or statements used:

Matplotlib.pyplot Explained in section 5.4.

- Task 5.6 code:

```
# Prepare data
runs_subset_2015_2018 = df_run['2018':'2015']
runs_distance = runs_subset_2015_2018['Distance (km)']
runs_hr = runs_subset_2015_2018['Average Heart Rate (bpm)']

# Create plot
fig, (ax1, ax2) = plt.subplots(2,sharex=True,figsize=(12,8))

# Plot and customize first subplot
runs_distance.plot(ax=ax1)
ax1.set(ylabel='Distance (km)', title='Historical data with averages')
ax1.axhline(runs_distance.mean(), color='blue', linewidth=1, linestyle='-.')

# Plot and customize second subplot
runs_hr.plot(ax=ax2, color='gray')
ax2.set(xlabel='Date', ylabel='Average Heart Rate (bpm)')
ax2.axhline(runs_hr.mean(),color='blue',linewidth = 1,linestyle = '-.')

# Show plot
plt.show()
```

Figure 13: Coding from task 5.6

- Task 5.6 result:

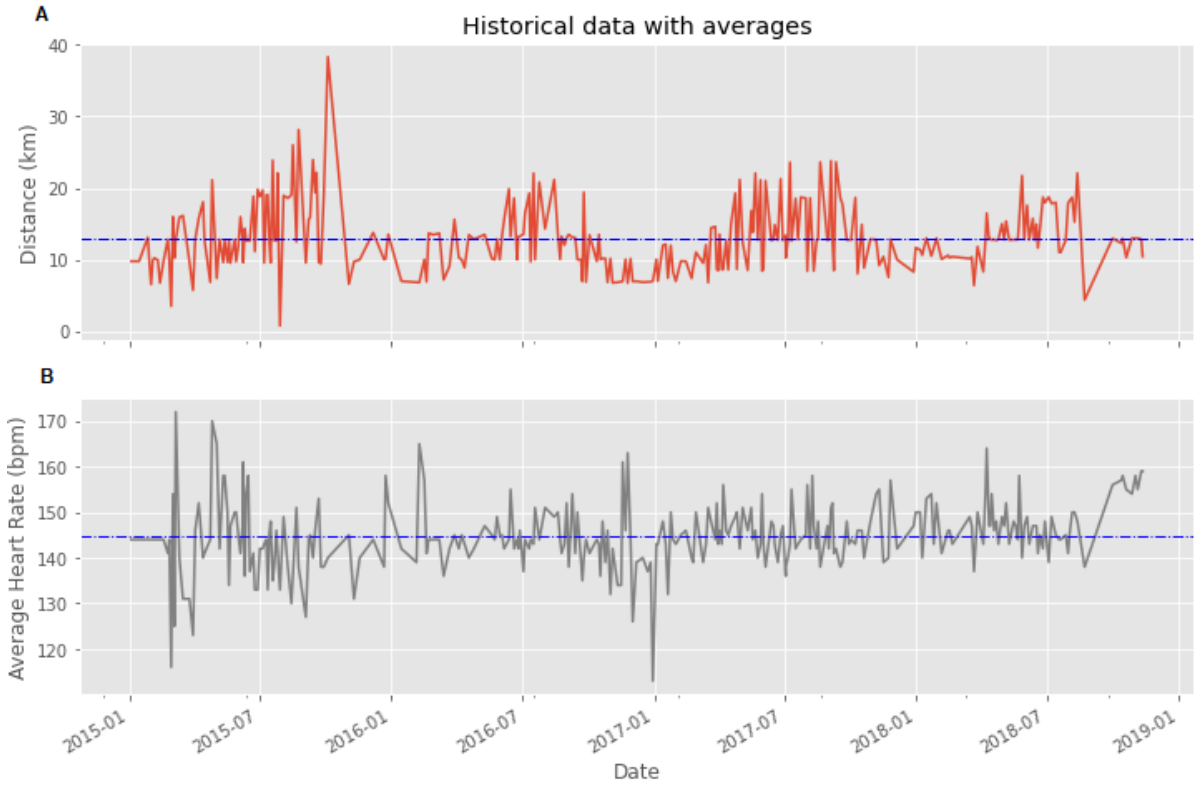

Figure 14: Result from task 5.6, Historical data with averages. A) Distance (km) vs Date. B) Average Heart Rate (bpm) vs Date

In figure 14 we find the advance in time of the heart rate and the distance traveled, in comparison with its historical average. In general, we found that the values were around the mean value and we did not find a tendency to increase (distance and / or heart rate) over time.

5. 7- Did the user reach their goals?

Let's consider a goal of running 1000 km per year. If we visualize the annual race distance (km) from 2013 to 2018 it is possible to see if the goal for each year was reached. The graph will show a star for each annual value of distance traveled, only the stars in the green region indicate success.

- Functions, method, attributes and/or statements used:

DataFrame.resample() Explained in section 5.5.

DataFrame.Sum() Explained in section 5.2.

Matplotlib.pyplot Explained in section 5.4.

- Task 5.7 code:

```
# Prepare data
df_run_dist_annual = df_run.sort_index()
df_run_dist_annual = df_run_dist_annual['2013':'2018'].resample('A')['Distance (km)'].sum()

# Create plot
fig = plt.figure(figsize=(8,5))

# Plot and customize
ax = df_run_dist_annual.plot(marker='*', markersize=14, linewidth=0, color='blue')
ax.set(ylim=[0, 1210],
       xlim=['2012','2019'],
       ylabel='Distance (km)',
       xlabel='Years',
       title='Annual totals for distance')

ax.axhspan(1000, 1210, color='green', alpha=0.4)
ax.axhspan(800, 1000, color='yellow', alpha=0.3)
ax.axhspan(0,800, color ='red', alpha = 0.2)

# Show plot
plt.show()
```

Figure 15: Coding from task 5.7.

- Task 5.7 result:

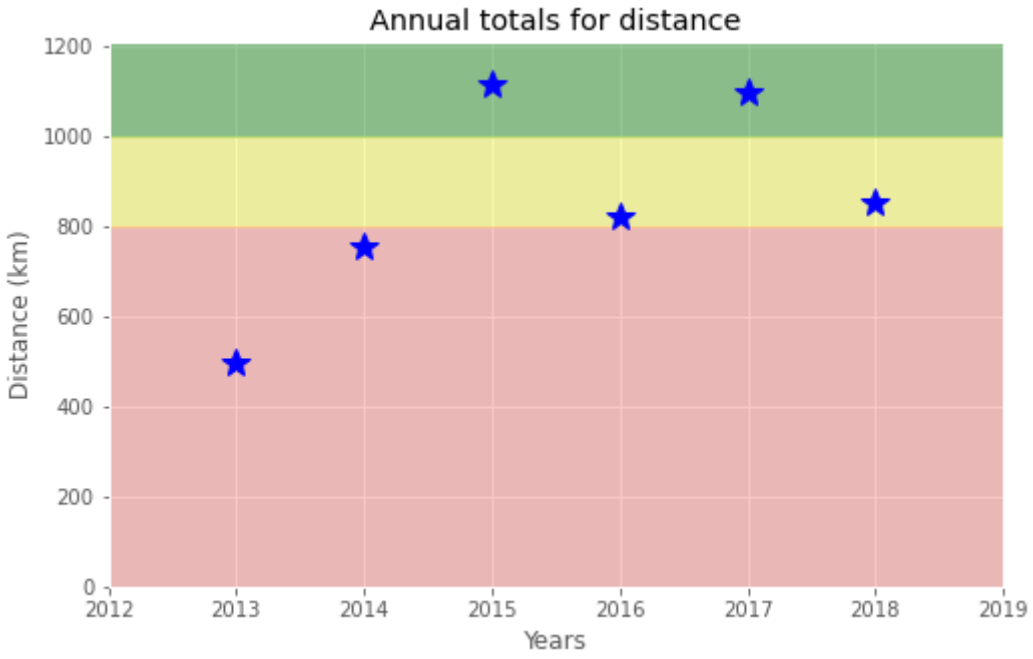

Figure 16: result from task 5.7. The graphic shows the distance traveled each year from 2013 until 2018.

Amazing! Figure 16 shows how the user only reached his goal in 2015 and 2017. He almost achieved it in 2016 and 2018, and it is possible to see how the first 2 years were far from achieving it.

5. 8 - Is the user progressing?

In order to answer this question and determine if the user is progressing in terms of running skills, it is necessary to dig deeper into the data. Therefore, the weekly distance run will be decomposed and visually compared to the raw data. A red trend line will represent the distance traveled weekly.

- Functions, method, attributes and/or statements used:

statsmodels.tsa.seasonal.seasonal_decompose() Seasonal decomposition using moving averages.

DataFrame.resample() Explained in section 5.5.

DadaFrame.bfill() Backward fill the new missing values in the resampled data. In statistics, imputation is the process of replacing missing data with substituted values. When resampling data, missing values may appear (e.g., when the resampling frequency is higher than the original frequency). The backward fill will replace NaN values that appeared in the resampled data with the next value in the original sequence. Missing values that existed in the original data will not be modified.

- Task 5.8 code:

```
# Import required library
import statsmodels.api as sm

# Prepare data
df_run_dist_wkly = df_run.sort_index()
df_run_dist_wkly = df_run_dist_wkly['2013':'2018'].resample('W')['Distance (km)'].bfill()
decomposed = sm.tsa.seasonal_decompose(df_run_dist_wkly, extrapolate_trend=1, freq=52)

# Create plot
fig = plt.figure(figsize=(12,5))

# Plot and customize
ax = decomposed.trend.plot(label='Trend', linewidth=2)
ax = decomposed.observed.plot(label='Observed', linewidth=0.5)

ax.legend()
ax.set_title('Running distance trend')

# Show plot
plt.show()
```

Figure 17: Task 5.8 code

- Task 5.8 result:

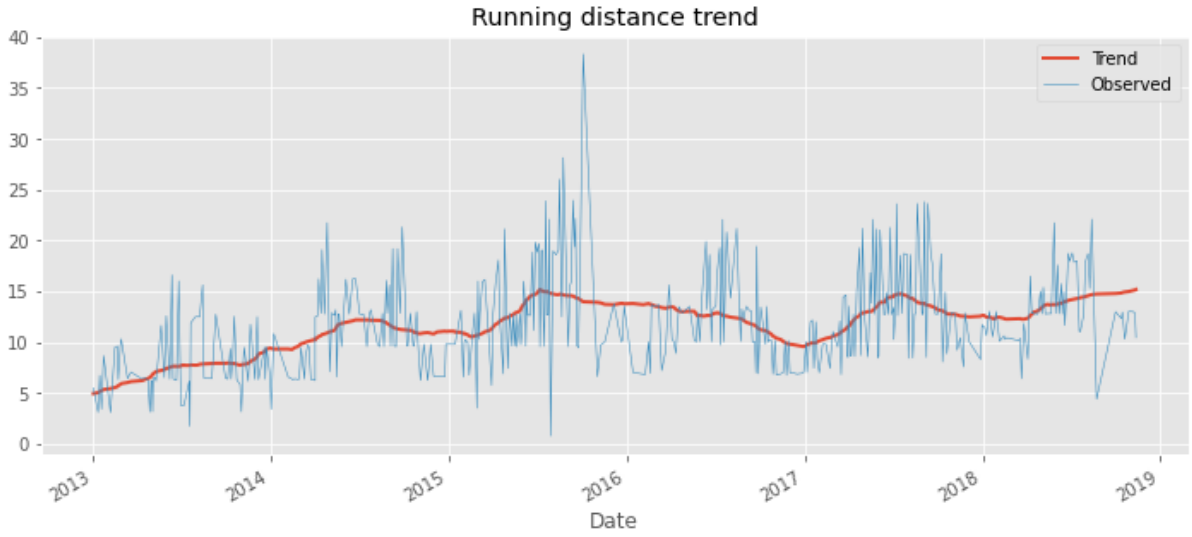

Figure 18: Task 5.8 result. The graphic shows the running distance trend over the years

Shocking! It is possible to notice how there is a growing trend, mainly in the first 3 years (figure 18). When we looked at the result of task 5.6, it appeared that no such trend existed. This is probably due to the fact that the graphs are not identical; figure 14.A is for the period 2015-2018 while figure 18 also includes the period between 2013 and 2015, where the greatest growing trend is observed. Also, it may be because in figure 14.A we are comparing with the mean and in figure 18 we are comparing with respect to the trend line. In conclusion, it is possible to note that both graphs provide us with information but to be able to see more clearly if there is progress or not, it is advisable to use a trend line.

5. 9 - Training intensity

Heart rate is a popular metric used to measure training intensity. Depending on age and fitness level, heart rates are grouped into different zones that people can target depending on training goals. A target heart rate during moderate-intensity activities is about 50-70% of maximum heart rate, while during vigorous physical activity it's about 70-85% of maximum.
A distribution plot of my heart rate data by training intensity will be created. It will be a visual presentation for the number of activities from predefined training zones.

- Functions, method, attributes and/or statements used:

Matplotlib.pyplot explained in section 5.4

- Task 5.9 code:

```
# Prepare data
hr_zones = [100, 125, 133, 142, 151, 173]
zone_names = ['Easy', 'Moderate', 'Hard', 'Very hard', 'Maximal']
zone_colors = ['green', 'yellow', 'orange', 'tomato', 'red']
df_run_hr_all = df_run.sort_index()
df_run_hr_all = df_run_hr_all['2015':'2018']['Average Heart Rate (bpm)']

# Create plot
fig, ax = plt.subplots(figsize=(8,5))

# Plot and customize
n, bins, patches = ax.hist(df_run_hr_all, bins=hr_zones, alpha=0.5)
for i in range(0, len(patches)):
    patches[i].set_facecolor(zone_colors[i])

ax.set(title='Distribution of HR', ylabel='Number of runs')
ax.xaxis.set(ticks = hr_zones)
print(zone_names)
zone_names.insert(0,'Minimal')
ax.set_xticklabels(labels = zone_names, rotation = (-30), ha ='left')

# Show plot
plt.show()
```

Figure 19: Task 5.9 code

- Task 5.9 result:

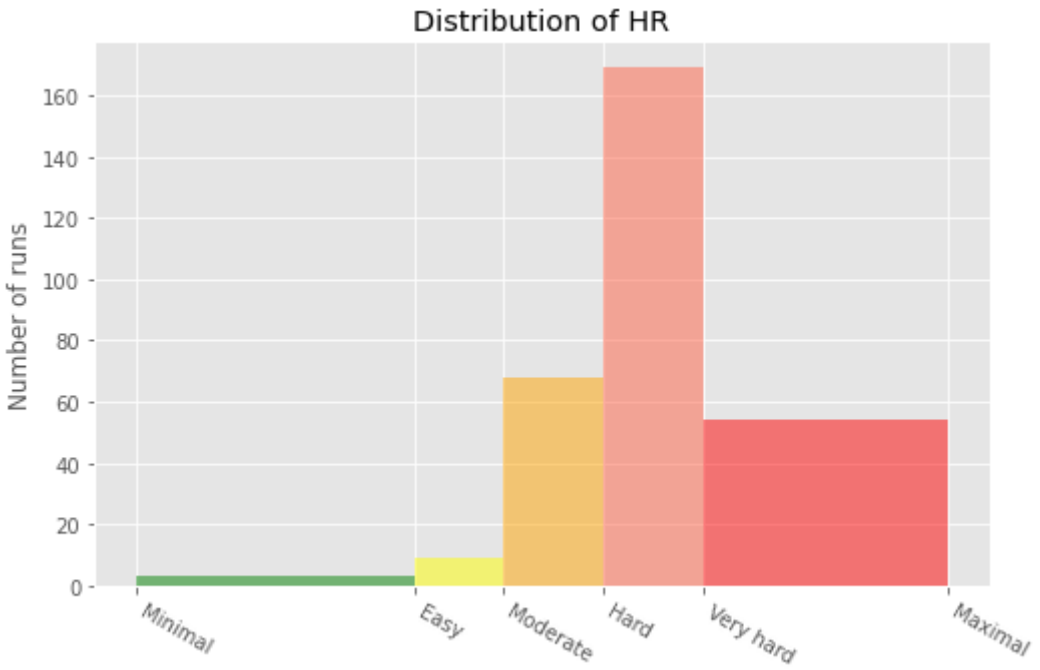

Figure 20: Task 5.9 result. The graph shows the distribution of the heart rate

Nice job runkeeper® user! Figure 20 clearly shows how the user worked mostly in a vigourous intensity. The higher the heart rate, the more vigorous the exercise and the more energy expenditure there will be, which translates into more calories being burned, that helps fight obesity and keep an optimal health (Brosh, 2007).

5.10 - Detailed summary report

After doing the data cleaning, analysis, and visualization; detailed summary tables of the training will be created, in order to have a general idea.

To do this, two tables will be created. The first table will be a summary of the distance (km) and climb (m) variables for each training activity. The second table will list the summary statistics for the average speed (km/h), climb (m), and distance (km) variables for each training activity.

- Functions, method, attributes and/or statements used

DataFrame.append() Append rows of other to the end of caller, returning a new object.

DataFrame.groupby() Group DataFrame using a mapper or by a Series of columns.

DataFrame.sum() Explained in section 5.2.

DataFrame.describe() Generate descriptive statistics. Descriptive statistics include those that summarize the central tendency, dispersion and shape of a dataset's distribution, excluding NaN values.

DataFrame.stack() Stack the prescribed level(s) from columns to index. Return a reshaped DataFrame or Series having a multi-level index with one or more new inner-most levels compared to the current DataFrame.

- Task 5.10 code:

```
# Concatenating three DataFrames
df_run_walk_cycle = df_run.append([df_walk,df_cycle],sort=True)

dist_climb_cols, speed_col = ['Distance (km)', 'Climb (m)'], ['Average Speed (km/h)']

# Calculating total distance and climb in each type of activities
df_totals = df_run_walk_cycle.groupby('Type')[dist_climb_cols].sum()

print('Totals for different training types:')
display(df_totals)

# Calculating summary statistics for each type of activities
df_summary = df_run_walk_cycle.groupby('Type')[dist_climb_cols + speed_col].describe()

# Combine totals with summary
for i in dist_climb_cols:
    df_summary[i, 'total'] = df_totals[i]

print('Summary statistics for different training types:')
df_summary.stack()
```

Figure 21: task 5.10 code

- Task 5.10 result:

A

Totals for different training types:

Type	Distance (km)	Climb (m)
Cycling	680.58	6976
Running	5224.50	57278
Walking	33.45	349

B

Summary statistics for different training types:

Type		Average Speed (km/h)	Climb (m)	Distance (km)
Cycling	25%	16.980000	139.000000	15.530000
	50%	19.500000	199.000000	20.300000
	75%	21.490000	318.000000	29.400000
	count	29.000000	29.000000	29.000000
	max	24.330000	553.000000	49.180000
	mean	19.125172	240.551724	23.468276
	min	11.380000	58.000000	11.410000
	std	3.257100	128.960289	9.451040
	total	NaN	6976.000000	680.580000
Running	25%	10.495000	54.000000	7.415000
	50%	10.980000	91.000000	10.810000
	75%	11.520000	171.000000	13.190000
	count	459.000000	459.000000	459.000000
	max	20.720000	982.000000	38.320000
	mean	11.056296	124.788671	11.382353
	min	5.770000	0.000000	0.760000
	std	0.953273	103.382177	4.937853
	total	NaN	57278.000000	5224.500000
Walking	25%	5.555000	7.000000	1.385000
	50%	5.970000	10.000000	1.485000
	75%	6.512500	15.500000	1.787500
	count	18.000000	18.000000	18.000000
	max	6.910000	112.000000	4.290000
	mean	5.549444	19.388889	1.858333
	min	1.040000	5.000000	1.220000
	std	1.459309	27.110100	0.880055
	total	NaN	349.000000	33.450000

Figure 22: Task 5.10 result. A) Summary of the distance traveled (km) and climb (m) for each activity. B) Summary statistics for the different training types.

Finally we can clearly see the user statistics. In figure 22.A, we can compare the distance traveled in each activity as well as the unevenness. It is noticeable how the values of running are much higher than the others, this is logical, since the user has run much more than he has walked or cycled.

On the other hand, in figure 22.B, we can study and compare the basic statistics of each activity, thanks to the *describe()* method. It is possible to compare the means, as well as the median and the standard deviation, among others. In this case, we will not delve into the analysis since that depends on the concerns of each person. Anyway we will use some of this values to determine some fun facts in the next section.

5. 11 - Fun facts

To wrap up, let's pick some fun facts out of the summary tables and solve the last exercise. These data represent 6 years, 2 months and 21 days. We will consider that the user wore 7 running shoes during all this time

Facts taken from figure 19:

- Average distance: 11.38 km
- Longest distance: 38.32 km
- Highest climb: 982 m
- Total climb: 57,278 m
- Total number of km run: 5,224 km
- Total runs: 459
- Number of running shoes gone through: 7 pairs

The story of Forrest Gump is well known–the man, who for no particular reason decided to go for a "little run." His epic run duration was 3 years, 2 months and 14 days (1169 days). In the picture you can see Forrest's route of 24,700 km.

Forrest run facts:

- Average distance: 21.13 km
- Total number of km run: 24,700 km
- Total runs: 1169

Assuming Forest and the user go through running shoes at the same rate, figure out how many pairs of shoes Forrest needed for his run.

- Functions, method, attributes and/or statements used:

Round() Round to a variable number of decimal places.

- Task 5.11 code:

```
# Count average shoes per lifetime (as km per pair) using our fun facts
average_shoes_lifetime = 5224/7

# Count number of shoes for Forrest's run distance
shoes_for_forrest_run = round(24700/average_shoes_lifetime)

print('Forrest Gump would need {} pairs of shoes!'.format(shoes_for_forrest_run))
```

Figure 23: Task 5.10 code

- Task 5.11 result:

Forrest Gump would need 33 pairs of shoes!

DISCUSSION AND FUTURE SCOPE

That a society remains physically active is of the utmost importance. The authorities must ensure to promote a society that complies with a minimum level of physical activity to avoid chronic diseases such as obesity, cancer, among others. Today, there are different strategies that people use to stay motivated to exercise, as well as to achieve their goals and objectives at the amateur and professional level. As we have seen in the literature, a widely used strategy is fitness tracking. Fitness tracking allows users to know their physical condition, set goals, know their limits and track their activity over time.

In this project, a dataset from a user of the fitness tracking Runkeeper® application has been exhaustively analyzed and very useful data has been revealed that would interest any athlete. Through a deep data analysis, exploration and visualization, it has been determined which activity is most interesting for the user, which are their average values, as well as how their training routines have improved (or not) over time, knowing how you have progressed in distance traveled, heart rate and average speed. It has also been discovered in which year it has reached its objectives, which have been its maximum and minimum values. It would also have been possible to compare its performance with other users if we had required it. Finally, it has even been possible to play with the values a bit, comparing them with data from the beloved character of Forrest Gump.

After this analysis, it is possible to realize that data analysis has a very important power and that it is possible to answer the questions that we ask ourselves if we carry out an orderly, correct process and know what tools to use.

Considering merely the technical-academic side, the realization of this project allows us to learn to handle the tools available to transform our raw data into fruitful information, learning the necessary path that must be followed for said result.

This knowledge could be used in the future to compare different users, set personal goals, or it could even be used to carry out a general analysis of the physical fitness of all the users of the application. It is truly necessary to assimilate that, possibly, if we propose it and have the necessary tools, any analysis can be carried out.

BIBLIOGRAPHY

Bietz, M. J., Bloss, C. S., Calvert, S., Godino, J. G., Gregory, J., Claffey, M. P., Sheehan, J., & Patrick, K. (2016). Opportunities and challenges in the use of personal health data for health research. *Journal of the American Medical Informatics Association: JAMIA*, *23*(e1), e42-48. https://doi.org/10.1093/jamia/ocv118

Bravata, D. M., Smith-Spangler, C., Sundaram, V., Gienger, A. L., Lin, N., Lewis, R., Stave, C. D., Olkin, I., & Sirard, J. R. (2007). Using pedometers to increase physical activity and improve health: A systematic review. *JAMA*, *298*(19), 2296–2304. https://doi.org/10.1001/jama.298.19.2296

Brosh, A. (2007). Heart rate measurements as an index of energy expenditure and energy balance in ruminants: A review1. *Journal of Animal Science*, *85*(5), 1213–1227. https://doi.org/10.2527/jas.2006-298

Ceja, A. (n.d.). *¿Por qué los Data Scientist utilizan Python?* Retrieved 26 January 2021, from https://www.neoland.es/blog/por-que-los-data-scientist-utilizan-python

Jupyter Notebook: Documentos web para análisis de datos, código en vivo y mucho más. (n.d.). IONOS Digitalguide. Retrieved 26 January 2021, from https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/jupyter-notebook/

¿Qué es Python? (2020, February 24). https://web.archive.org/web/20200224120525/https://luca-d3.com/es/data-speaks/diccionario-tecnologico/python-lenguaje

Ridgers, N. D., McNarry, M. A., & Mackintosh, K. A. (2016). Feasibility and Effectiveness of Using Wearable Activity Trackers in Youth: A Systematic Review. *JMIR MHealth and UHealth*, *4*(4), e129. https://doi.org/10.2196/mhealth.6540

Stahl, J. (2015, November 22). *The Technology That Created a New Generation of Runners*. The Atlantic. https://www.theatlantic.com/technology/archive/2015/11/the-technology-that-created-a-new-generation-of-runners/417126/

Sullivan, A. N., & Lachman, M. E. (2017). Behavior Change with Fitness Technology in Sedentary Adults: A Review of the Evidence for Increasing Physical Activity. *Frontiers in Public Health*, *4*. https://doi.org/10.3389/fpubh.2016.00289

WHO | Physical Activity and Adults. (2020, April 2). WHO; World Health Organization. https://www.who.int/dietphysicalactivity/factsheet_adults/en/