Data Scientist Project Report

on

ANALYSE FITNESS DATA

MedTourEasy, New Delhi

SUBMITTED BY:

Sanjeev Deka

ASSIGNED BY:

Mr. Ankit Hasija

Training Head

MedTourEasy

Introduction:

With the explosion in fitness tracker popularity, runners all of the world are collecting data with gadgets (smartphones, watches, etc.) to keep themselves motivated. They look for answers to questions like:

- How fast, long, and intense was my run today?
- Have I succeeded with my training goals?
- Am I progressing?
- What were my best achievements?
- How do I perform compared to others?

The data was exported seven years worth of an individual's training data from Runkeeper. The data is a CSV file where each row is a single training activity. In this project, we'll create import, clean, and analyze the data to answer the above questions.

Project Tasks:

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 I reach my goals?
8. Am I progressing?
9. Training intensity
10. Detailed summary report
11. Fun facts

1. Data Overview

The data we use here is a person's personal fitness data from Runkeeper. For anonymity we'll use the name Sam.

One day, Sam's old running friend and he was chatting about their running styles, training habits, and achievements, when he suddenly realized that he could take an in-depth analytical look at his training. Sam have been using a popular GPS fitness tracker called Runkeeper for years and decided it was time to analyze his running data to see how he was doing.

Since 2012, he has been using the Runkeeper app, and it's great. One key feature: its excellent data export. Anyone who has a smartphone can download the app and analyze their data like we will in this notebook.

After logging your run, the first step is to export the data from Runkeeper. Then import the data and start exploring to find potential problems. After that, create data cleaning strategies to fix the issues. Finally, analyze and visualize the clean time-series data.

I have exported seven years worth of his training data, from 2012 through 2018. The data is a CSV file where each row is a single training activity. Let's load and inspect it.

2. Data Preprocessing

First we import the data into a Pandas DataFrame for ease of data manipulation. The DataFrame contains:

- 508 rows - training activity from *2018-11-11 14:05:12* to *2012-08-22 18:53:54*
- 13 columns - *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.*

Lucky for us, the column names Runkeeper provides are informative, and we don't need to rename any columns.

But, we do notice missing values using the *info()* method. What are the reasons for these missing values? It depends. Some heart rate information is missing because Sam didn't always use a cardio sensor. In the case of the *Notes* column, it is an optional field that he sometimes left blank. Also, he only used the *Route Name* column once, and never used the *Friend's Tagged* column.

We'll fill in missing values in the heart rate column to avoid misleading results later, but right now, our first data preprocessing steps will be to:

- Remove columns not useful for our analysis.
- Replace the "Other" activity type to "Unicycling" because that was always the "Other" activity.
- Count missing values.

Types count of training activities:

```
Running    459
Cycling     29
Walking     18
Other        2
```

Missing values count for each column:

```
Type                          0
Distance (km)                 0
Duration                      0
Average Pace                  0
Average Speed (km/h)          0
Climb (m)                     0
Average Heart Rate (bpm)    214
```

3. Dealing with Missing Values

As we can see from the last output, there are 214 missing entries for average heart rate.

We can't go back in time to get those data, but we can fill in the missing values with an average value. 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 DataFrames by activity type (*Type*) and calculate each activity's mean heart rate, then fill in the missing values with those means.

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)
```

Missing values count for each column in running data after filling missing values:

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

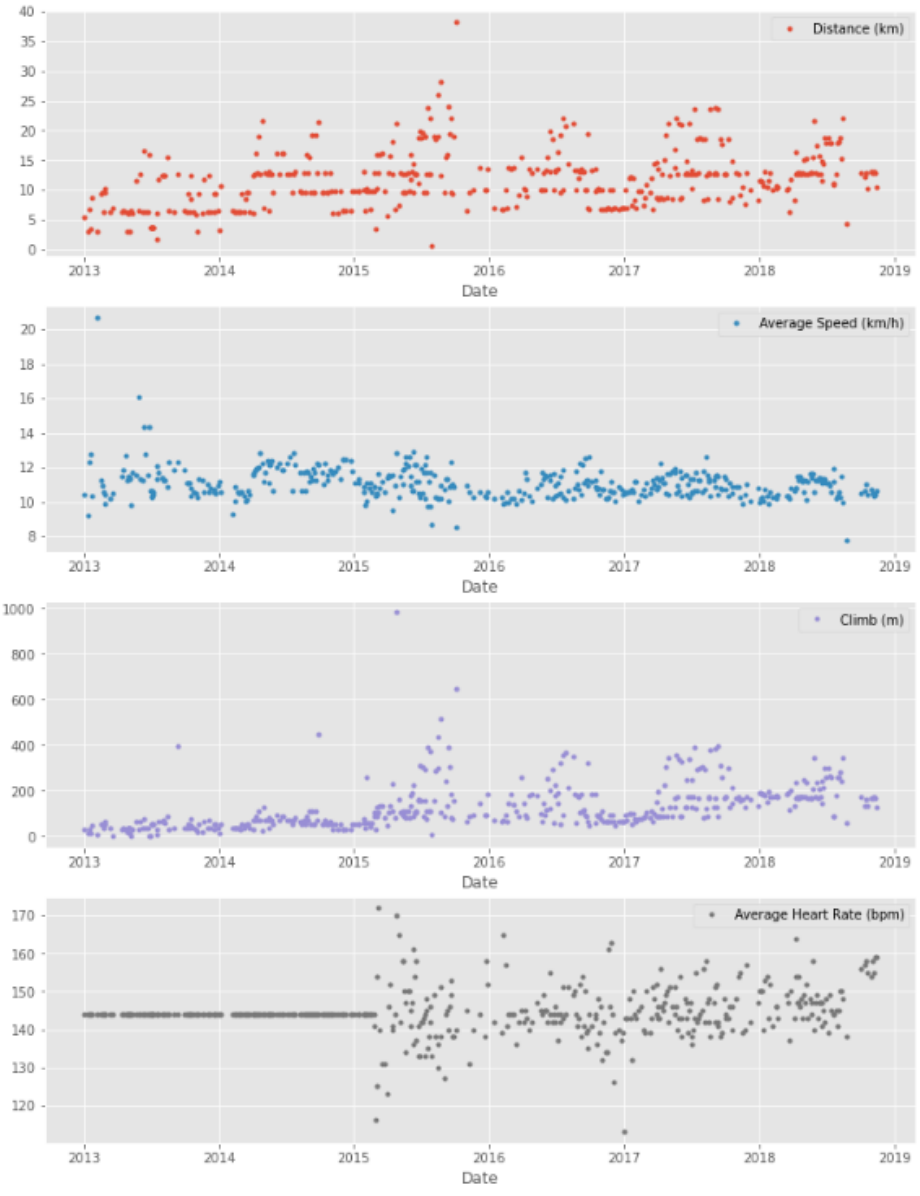
4. Plot Running Data

Now we can create our first plot. As we found earlier, most of the activities in the data were running (459 of them to be exact). There are only 29, 18, and two instances for cycling, walking, and unicycling, respectively. So for now, let's focus on plotting the different running metrics.

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.

First we prepare data subsetting period from 2013 till 2018 and then plot:

```
runs_subset_2013_2018 = df_run['2018':'2013']
```



5. Running Statistics

No doubt, running helps people stay mentally and physically healthy and productive at any age. And it is great fun! When runners talk to each other about their hobby, we not only discuss our results, but we also discuss different training strategies.

We'll know we are with a group of runners if we commonly hear questions like:

- What is your average distance?
- How fast do you run?
- Do you measure your heart rate?
- How often do you train?

Let's find the answers to these questions in the data. If we look back at plots in Task 4, we can see the answer to, *Do you measure your heart rate?* Before 2015: no. To look at the averages, let's only use the data from 2015 through 2018.

In pandas, the *resample()* method is similar to the *groupby()* method - with *resample()* we group by a specific time span. We'll use *resample()* to group the time series data by a sampling period and apply several methods to each sampling period. In our case, we'll resample annually and weekly.

Prepare running data for the last 4 years:

```
runs_subset_2015_2018 = df_run['2018':'2015']
```

Average run 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

Weekly averages of last 4 years:

```
Distance (km)                12.518176
Average Speed (km/h)         10.835473
Climb (m)                   158.325444
Average Heart Rate (bpm)    144.801775
```

Trainings per week on average: 1.5

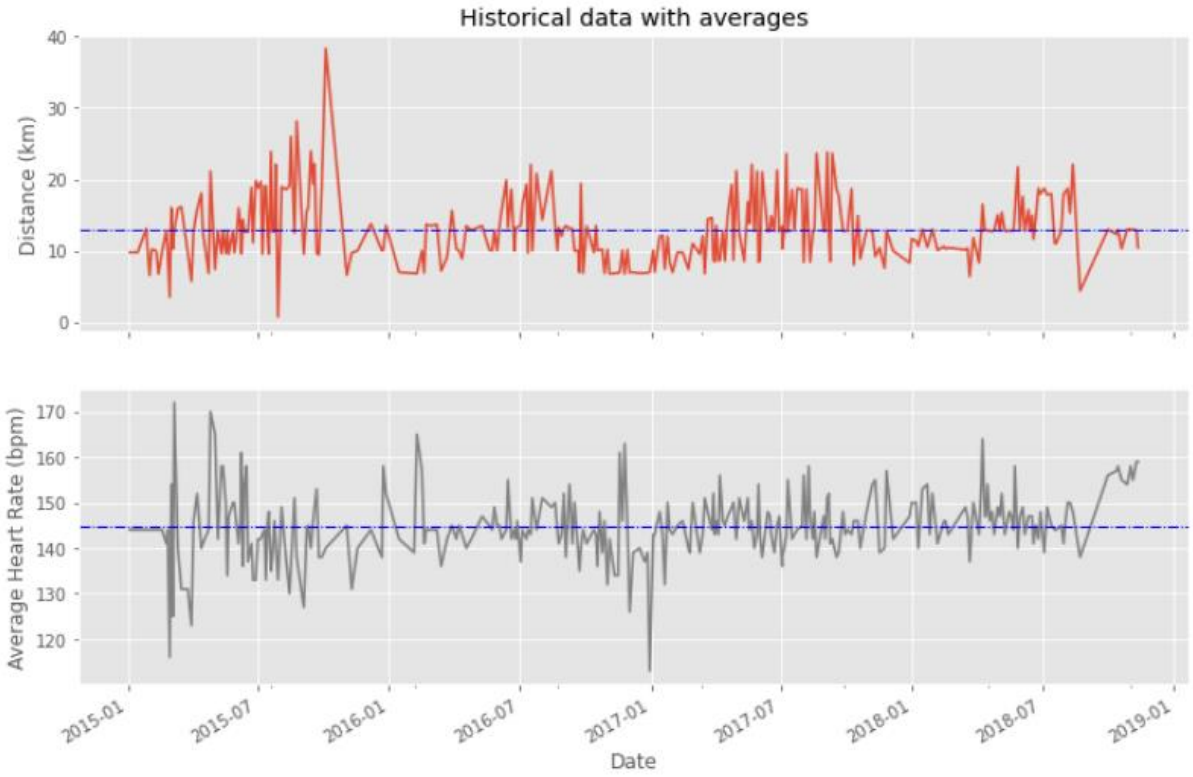
6. Visualization with averages

Let's plot the long term averages of distance run and 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.

In this task, we will use *matplotlib* functionality for plot creation and customization.

Prepare data and then plot:

```
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)']
```

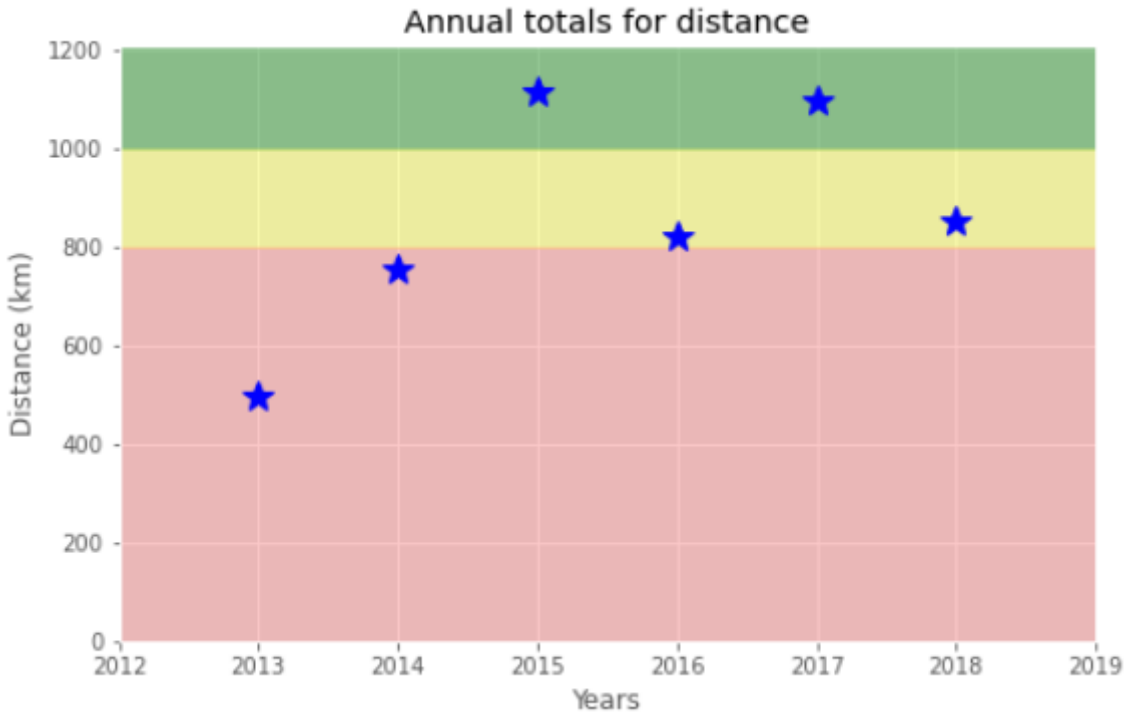


7. Did I reach my goals?

To motivate himself to run regularly, Sam set a target goal of running 1000 km per year. Let's visualize the annual running distance (km) from 2013 through 2018 to see if he reached his goal each year. Only stars in the green region indicate success.

Prepare data and then plot:

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



8. Am I progressing?

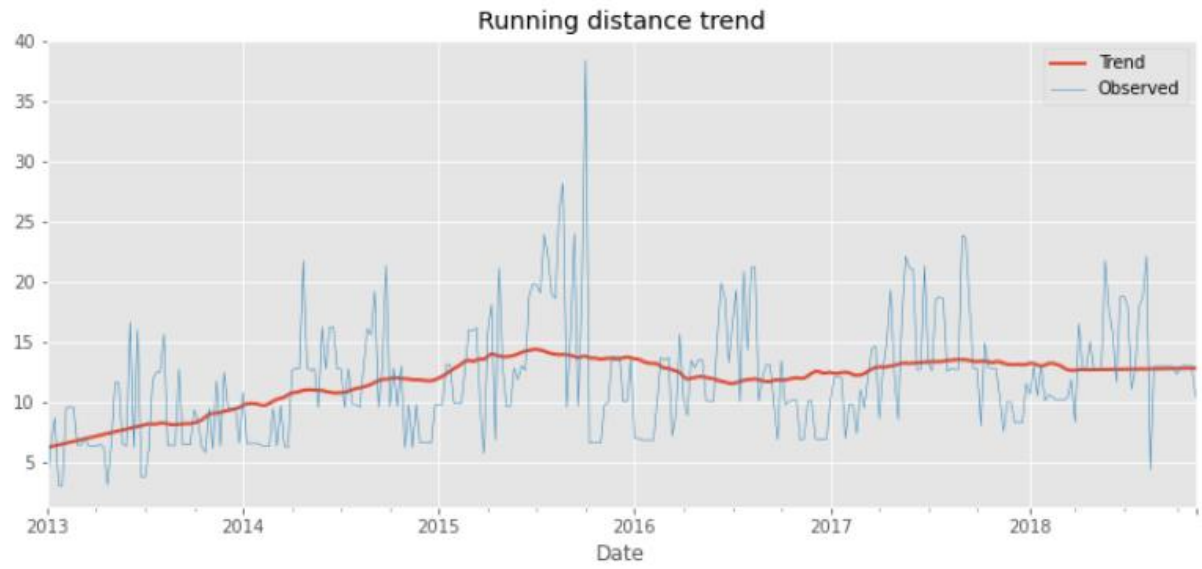
Let's dive a little deeper into the data to answer a tricky question: was Sam progressing in terms of his running skills?

To answer this question, we'll decompose the weekly distance run and visually compare it to the raw data. A red trend line will represent the weekly distance run.

We are going to use *statsmodels* library to decompose the weekly trend.

Prepare data and then plot:

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



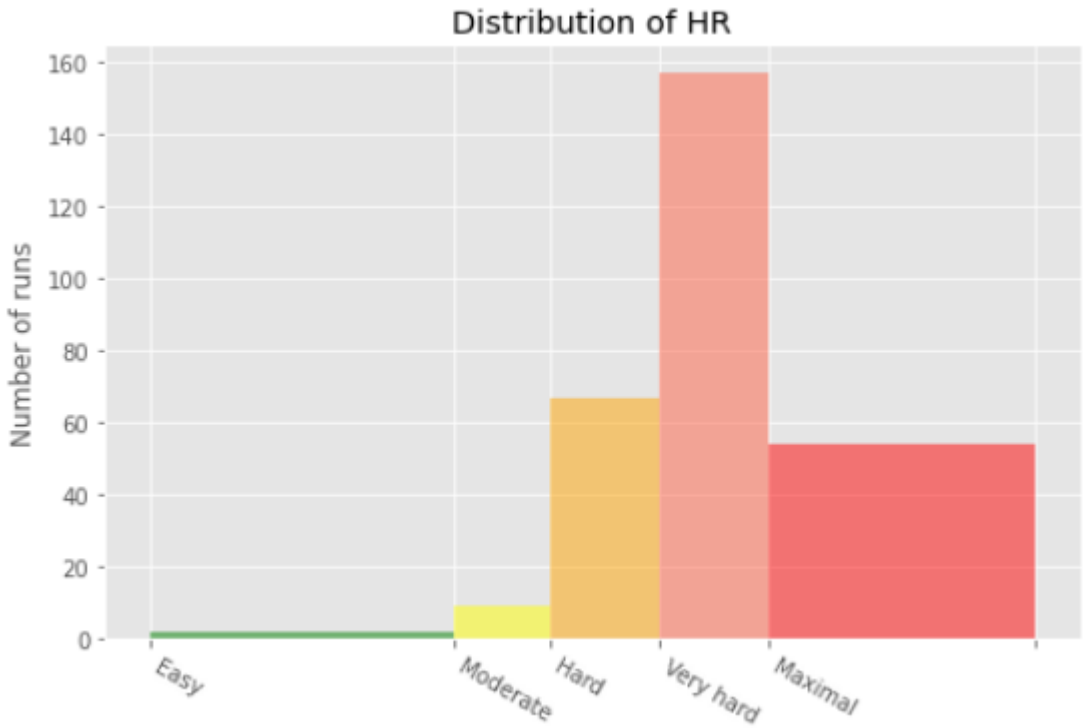
9. Training Intensity

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.

We'll create a distribution plot of the heart rate data by training intensity. It will be a visual presentation for the number of activities from predefined training zones.

Prepare data and then plot:

```
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['2018-12':'2015-03']['Average Heart Rate (bpm)']
```



10. Detailed Summary Report

With all this data cleaning, analysis, and visualization, let's create detailed summary tables of training.

To do this, we'll create two tables. 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/hr), climb (m), and distance (km) variables for each training activity.

Concatenating three DataFrames:

```
df_run_walk_cycle = df_run.append(df_walk).append(df_cycle).sort_index(ascending=False)
```

Total distance and climb in each type of activities:

```
df_totals = df_run_walk_cycle.groupby('Type')[dist_climb_cols].sum()
```

Totals for different training types:

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

Summary statistics for each type of activities:

```
df_summary = df_run_walk_cycle.groupby('Type')[dist_climb_cols + speed_col].describe()
```

Combining totals with summary:

```
for i in dist_climb_cols:
    df_summary[i, 'total'] = df_totals[i]
```

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

11. Conclusion & Fun Facts

To wrap up, let’s pick some fun facts out of the summary tables and solve the last exercise.

These data (Sam's running history) represent 6 years, 2 months and 21 days. And how many running shoes Sam went through - 7.

FUN FACTS:

- 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
- Number of running shoes gone through: ?

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

Average shoes per lifetime (as km per pair)

= Total number of km run / Number of running shoes gone through

= 5224/7

= 746.3

Number of shoes for Forrest's run distance

= Total number of km run / Average shoes per lifetime

= 24700/746.3

= 33.1

Forrest Gump would need 33 pairs of shoes!