Analysis of Fitness Data Project Proposal

Created by:MTE

April 2021

TABLE OF CONTENTS

Project Motivation 3

The Dataset 3

Project Tasks 4

- Task1: Obtain and review raw data 4
- Task 2 : Data preprocessing 5
- Task 3 : Dealing with missing values 6
- Task 4 : Plot running data 8
- Task 5 : Running statistics 9
- Task 6 : Visualization with averages 10
- Task 7 : Did I reach my goals? 11
- Task 8 : Am I progressing? 12
- Task 9 : Training intensity 13
- Task 10 : Detailed summary report 14
- Task 11 : Fun facts 16

Project Motivation

With the explosion in fitness tracker popularity, runners all over the world are collecting data with gadgets (smartphones, watches, etc.) to keep themselves motivated. In this project we are going to 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?

Eleven tasks will be set to achieve the aim of the project. In addition to that, we will create, import, clean, and analyze my data to answer the above questions.

The Dataset

The dataset was exported from Runkeeper. The ASICS Runkeeper™ app has the tools that are used to become a stronger runner. Insights, workouts, and trainer guidance will get him to achieve goals. The data is a CSV file that has 508 rows and 13 columns. training data recorded from 2012 to 2018. There are four different types of activity (running, walking, cycling and other) where each row is a single training activity with details about the distance, average Heart Rate and average Speed in addition to other information.

Project Tasks

Task1: Obtain and review raw data

```
1]: # 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=['Date'], index_col=['Date'])
df_activities.info()
# First look at exported data: select sample of 3 random rows
display(df_activities.sample(3))

#Print DataFrame summary
df_activities.sum()
```

```
<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
```

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
2018-04-11 18:17:35	5581c2b2-e254-46f8-99b0-388913baf5cf	Running	NaN	12.82	1:07:10	5:14	11.45	892.0	173	147.0	NaN	TomTom MySports Watch	2018-04-11-181735.gpx
2017-01-30 18:43:33	dcf483e9-421e-42b6-b116-884c794f5366	Running	NaN	6.98	37:28	5:22	11.18	497.0	66	143.0	NaN	TomTom MySports Watch	2017-01-30-184333.gpx
2016-10-31 19:34:32	86ea4944-dc1b-4cc9-8e20-eb3a0267e3ff	Running	NaN	10.15	56:36	5:35	10.76	701.0	109	132.0	NaN	TomTom MySports Watch	2016-10-31-193432.gpx

Using Pandas as a python library for data analysis and for importing the dataset Pandas has a **read_csv** method to read and save the data into dataframe. With **parse_dates** that convert the data column from string to date datatype and use this column as index to the dataframe. Applied statistical summaries to understand the data,investigate the properties and data type of columns.

Task 2 : Data preprocessing

In the preprocessing stage, we removed columns not useful for analysis using the **drop** method. Then, replace the "Other" activity type to "Unicycling" using the **replace** method. We can notice that running contributes the majority of the four category activities.

In [22]:
```
# 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 = df_activities.drop(cols_to_drop, axis=1)
print(df_activities.head())

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

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

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

```
                      Type  Distance (km) Duration Average Pace  \
Date
2018-11-11 14:05:12  Running         10.44    58:40         5:37
2018-11-09 15:02:35  Running         12.84  1:14:12         5:47
2018-11-04 16:05:00  Running         13.01  1:15:16         5:47
2018-11-01 14:03:58  Running         12.98  1:14:25         5:44
2018-10-27 17:01:36  Running         13.02  1:12:50         5:36

                     Average Speed (km/h)  Climb (m)  Average Heart Rate (bpm)
Date
2018-11-11 14:05:12                 10.68        130                     159.0
2018-11-09 15:02:35                 10.39        168                     159.0
2018-11-04 16:05:00                 10.37        171                     155.0
2018-11-01 14:03:58                 10.47        169                     158.0
2018-10-27 17:01:36                 10.73        170                     154.0

Running    459
Cycling     29
Walking     18
Other        2
Name: Type, dtype: int64
```

Out[22]:
```
Type                        0
Distance (km)               0
Duration                    0
Average Pace                0
Average Speed (km/h)        0
Climb (m)                   0
Average Heart Rate (bpm)    214
dtype: int64
```

Task 3 : Dealing with missing values

With using isnull() and sum() methods,It was found that there were 214 missing values in the average heart rate column. 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
We can notice that the data start from 22/08/2012 and end on 11/11 2018.

In [23]:
```
# 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()
avg_hr_walk = df_activities[df_activities['Type'] == 'Walking']['Average Heart Rate (bpm)'].mean()
avg_hr_unicycle = df_activities[df_activities['Type'] == 'Unicycling']['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()
df_unicycle = df_activities[df_activities['Type'] == 'Unicycling'].copy()

# Filling missing values with counted means
df_walk['Average Heart Rate (bpm)'].fillna(avg_hr_walk, 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)
df_unicycle['Average Heart Rate (bpm)'].fillna(int(avg_hr_unicycle), inplace=True)


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

Out[23]:
```
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
```

In [24]:
```
df_run.index.max()
```

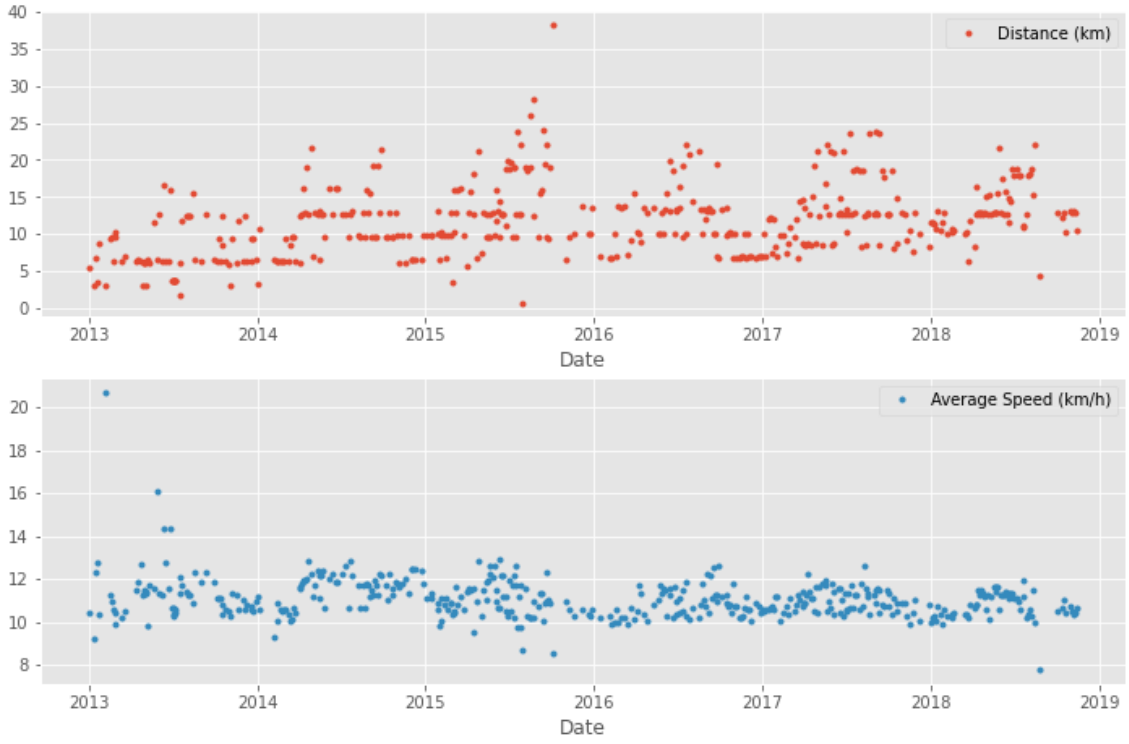
Out[24]:
```
Timestamp('2018-11-11 14:05:12')
```

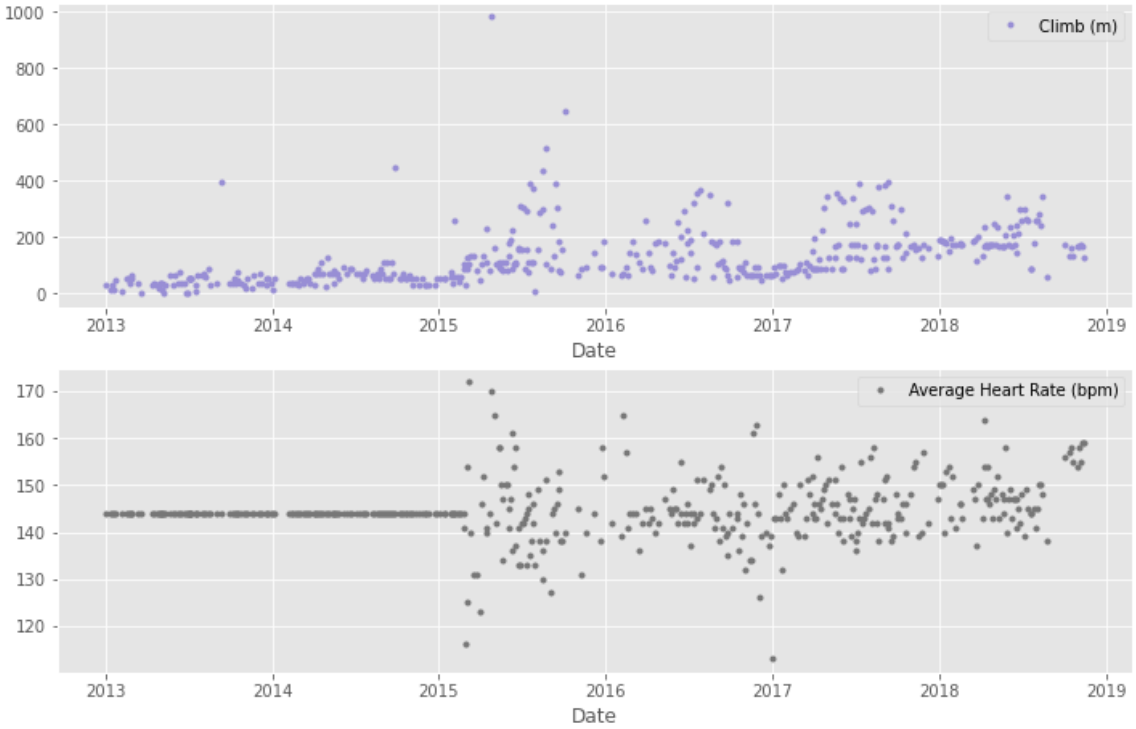
In [25]:
```
df_run.index.min()
```

Out[25]:
```
Timestamp('2012-08-22 18:53:54')
```

Task 4 : Plot running data

In [26]:
```
%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['2018':'2013']
print(runs_subset_2013_2018.index)

#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()
```





The four graphs illustrate scatter subplots for running activity for the different running metric (distance, average speed, climb and average heart rate) with the date for the six-period year from 2013 to 2019.

For the distance plot, we can see that There is an apparent constant trend slightly upwards and there is no obvious seasonality. While, the average speed we can notice a constant trend almost straight during the period date.

Regarding the climb and average heart rate the plots show a constant trend for the first three years when starting the training and then there were slightly upwards trends for the rest of the year's date.

Task 5 : Running statistics

In [27]:
```
# Prepare running data for the last 4 years
runs_subset_2015_2018 = df_run['2018':'2015']
#print(runs_subset_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())

# Mean weekly counts
weekly_counts_average = runs_subset_2015_2018['Distance (km)'].resample('W').count().mean()

print('How many trainings per week I had on average:',' ', weekly_counts_average)
```

```
How my average run looks in last 4 years:
```

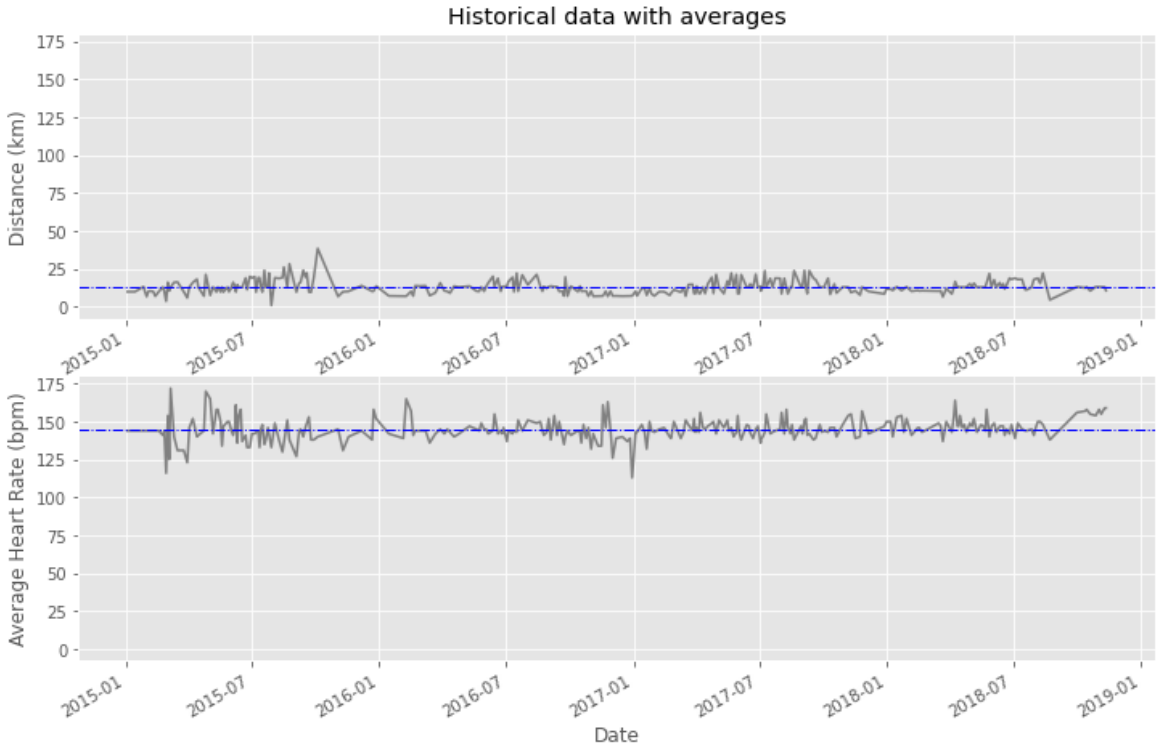
Date	Distance (km)	Average Speed (km/h)	Climb (m)	Average Heart Rate (bpm)
2015-12-31	13.602805	10.998902	160.170732	143.353659
2016-12-31	11.411667	10.837778	133.194444	143.388889
2017-12-31	12.935176	10.959059	169.376471	145.247059
2018-12-31	13.339063	10.777969	191.218750	148.125000

```
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
dtype: float64

How many trainings per week I had on average:   1.5
```

The table above shows information about the annual average distance, average speed, climb and average heart rate for the last four years from 2015 to 2018 for the running type activity.
We can notice that there was an increase in the average for the four metrics during this four-period year.
With more information about the mean of weekly training and how many times did the training per week.

Task 6 : Visualization with averages

In [28]:
```
# Prepare data
runs_subset_2015_2018 = df_run['2018':'2015']
print(runs_subset_2015_2018)

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, sharey=True, figsize =(12,8))

# Plot and customize first subplot
runs_distance.plot(ax=ax1, color='gray' )
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()
```

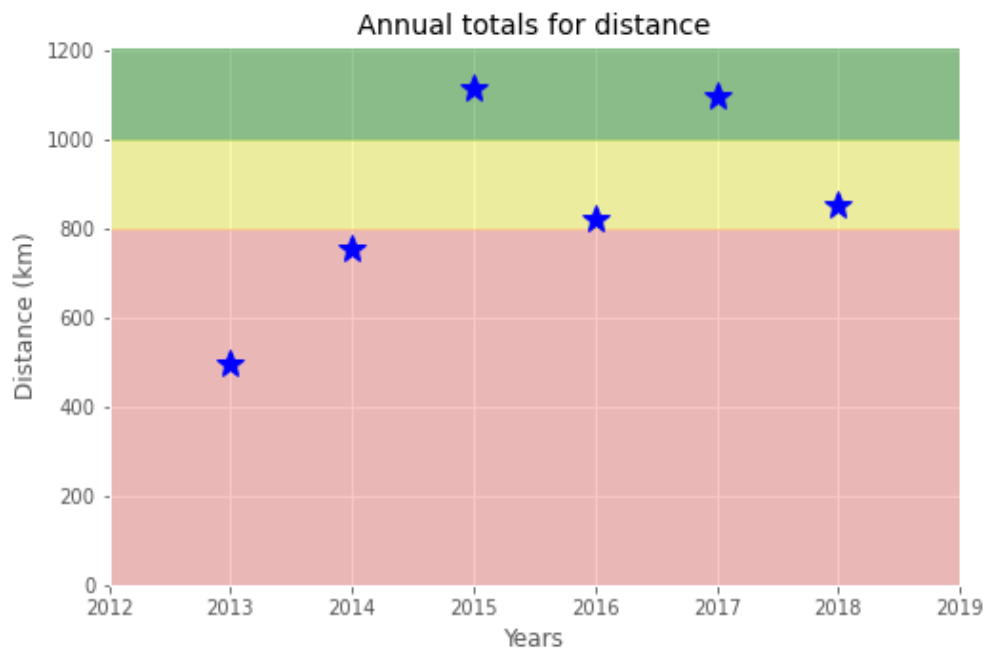


Plot the average distance and heart rate with raw data using the data from 2015 through 2018 . In this task, we will use **matplotlib** functionality for plot creation and customization.

What interesting point is that, There is no consistent trend (upward or downward) over the entire time span. The series appears to slowly wander up and down in both plots. The two blue horizontal lines drawn indicates the mean of the series. Notice that the series tends to stay on the same side of the mean (above or below) through the four years. There is no obvious **seasonality** and no potential **outliers**.

Task 7 : Did I reach my goals?

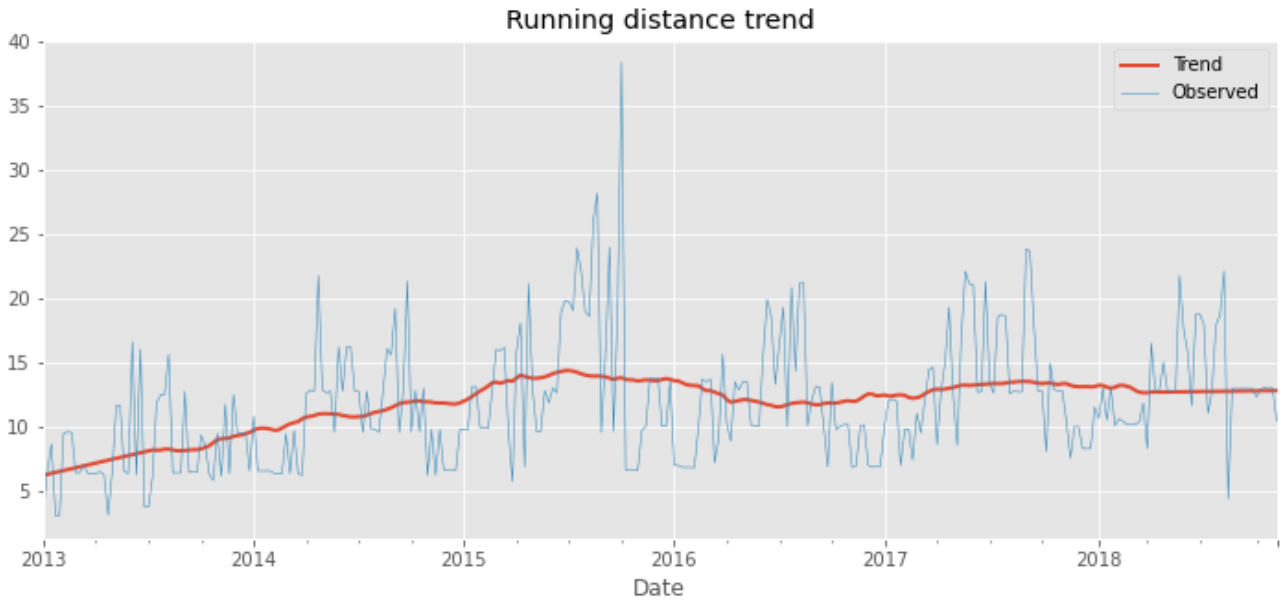
In [29]:
```
# Prepare data
df_run_dist_annual = df_run['2018':'2013']['Distance (km)'].resample('A').sum()

# Create plot
fig = plt.figure(figsize=( 8.0,5.0))

# 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()
```



In order to check if I reach my goals, a target goal of running was to set the distance to 1000 km per year. When visualizing the annual running distance (km) from 2013 through 2018 to see if the goal was reached each year. We can notice that only two stars in the green region indicate success.

Task 8 : Am I progressing?

In [30]:
```
# Import required library
import statsmodels.api as sm

# Prepare data
df_run_dist_wkly = df_run['2018':'2013']['Distance (km)'].resample('W').bfill()
print(df_run_dist_wkly)
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()
```



To answer this question, weekly distance run was decomposed and visually compared it to the raw data. A red trend line represents the weekly distance run. **statsmodels** library was used to decompose the weekly trend.

Time series decomposition involves thinking of a series as a combination of level, trend, seasonality, and noise components. The **seasonal_decompose()** function returns a result object. The result object contains arrays to access four pieces of data from the decomposition.

We can see that the trend information extracted from the series does seem that there is an apparent constant trend upwards until mid 2015 and then there was a slight downwards. After mid 2016 the trend line went straight.

Task 9 : Training intensity

To measure training intensity, Heart rate is a popular metric used Depending on age and fitness level. The heart rates are grouped into different zones that people can target depending on training goals. From the distribution plot of the heart rate data by training intensity, It is cleared that the most training is within the very hard zone (HR is between 142 and 151 bpm).

In [31]:
```
# 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['2018':'2015-03']['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)
ax.set_xticklabels(labels=zone_names, rotation=-30, ha='left')

# Show plot
plt.show()
```

Task 10 : Detailed summary report

In [32]:
```
# Concatenating three DataFrames
df_run_walk_cycle = df_run.append([df_walk, df_cycle]).sort_index(ascending=False)

print(df_run_walk_cycle)

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()
```

```
Totals for different training types:
```

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

The table above shows that total distance and climb in the whole period date of the dataset for the three different activities. We can see that the running activity is the most popular type which the runners exercised to get fitness with total distance 5224.5 km and climb 57278 m for 6 years period.

```
Summary statistics for different training types:
```

Out[32]:

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

The table above shows information about the list of summary statistics for the average speed (km/hr), climb (m), and distance (km) variables for each training activity.

Task 11 : Fun facts

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

These data (my running history) represent 6 years, 2 months and 21 days. And I remember how many running shoes I 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 I go through running shoes at the same rate, figure out how many pairs of shoes Forrest needed for his run.

In [33]:
```python
import math
Total_no_of_km_run_funfact = 5224
Number_of_running_shoes_funfact = 7
Total_no_of_km_Forrest_Gump = 24700

# Count average shoes per lifetime (as km per pair) using our fun facts
average_shoes_lifetime = Total_no_of_km_run_funfact / Number_of_running_shoes_funfact

# Count number of shoes for Forrest's run distance
shoes_for_forrest_run = math.floor(Total_no_of_km_Forrest_Gump / average_shoes_lifetime)

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

```
Forrest Gump would need 33 pairs of shoes!
```