Introduction to Statistical term

Statistics is the science of analyzing data.

When we have created a model for prediction, we must assess the prediction's reliability.

After all, what is a prediction worth, if we cannot rely on it?

Descriptive Statistics

We will first cover some basic descriptive statistics.

Descriptive statistics summarizes important features of a data set such as:

* Count
* Sum
* Standard Deviation
* Percentile
* Average
* Etc..

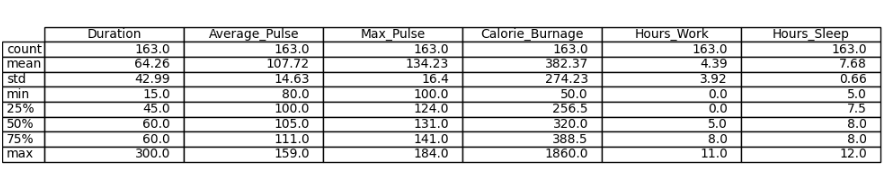
It is a good starting point to become familiar with the data.

We can use the describe() function in Python to summarize the data:

Example

print (full\_health\_data.describe())

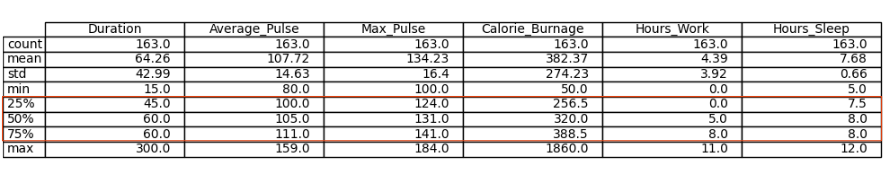
Output:



# Statistics Percentiles

25%, 50% and 75% - Percentiles

Percentiles are used in statistics to give you a number that describes the value that a given percent of the values are lower than.



Let us try to explain it by some examples, using Average\_Pulse.

* The 25% percentile of Average\_Pulse means that 25% of all of the training sessions have an average pulse of 100 beats per minute or lower. If we flip the statement, it means that 75% of all of the training sessions have an average pulse of 100 beats per minute or higher
* The 75% percentile of Average\_Pulse means that 75% of all the training session have an average pulse of 111 or lower. If we flip the statement, it means that 25% of all of the training sessions have an average pulse of 111 beats per minute or higher

Task: Find the 10% percentile for Max\_Pulse

The following example shows how to do it in Python:

Example

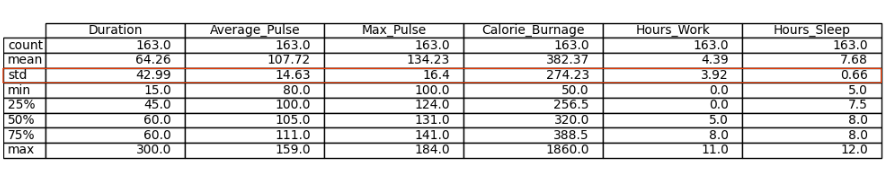
import numpy as np  
  
Max\_Pulse= full\_health\_data["Max\_Pulse"]  
percentile10 = np.percentile(Max\_Pulse, 10)  
print(percentile10)

* Max\_Pulse = full\_health\_data["Max\_Pulse"] - Isolate the variable Max\_Pulse from the full health data set.
* np.percentile() is used to define that we want the 10% percentile from Max\_Pulse.

The 10% percentile of Max\_Pulse is 120. This means that 10% of all the training sessions have a Max\_Pulse of 120 or lower.

Standard Deviation

Standard deviation is a number that describes how spread out the observations are.



A mathematical function will have difficulties in predicting precise values, if the observations are "spread". Standard deviation is a measure of uncertainty.

A low standard deviation means that most of the numbers are close to the mean (average) value.

A high standard deviation means that the values are spread out over a wider range.

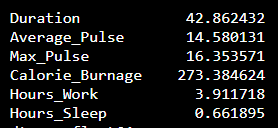
**Tip:** Standard Deviation is often represented by the symbol Sigma: σ

We can use the std() function from Numpy to find the standard deviation of a variable:

Example

import numpy as np  
  
std = np.std(full\_health\_data)  
print(std)

The output:



What does these numbers mean?

Coefficient of Variation

The coefficient of variation is used to get an idea of how large the standard deviation is.

Mathematically, the coefficient of variation is defined as:

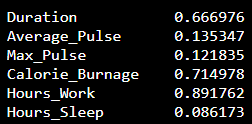
Coefficient of Variation = Standard Deviation / Mean

 We can do this in Python if we proceed with the following code:

Example

import numpy as np  
  
cv = np.std(full\_health\_data) / np.mean(full\_health\_data)  
print(cv)

The output:



We see that the variables Duration, Calorie\_Burnage and Hours\_Work has a high Standard Deviation compared to Max\_Pulse, Average\_Pulse and Hours\_Sleep.

Variance

Variance is another number that indicates how spread out the values are.

In fact, if you take the square root of the variance, you get the standard deviation. Or the other way around, if you multiply the standard deviation by itself, you get the variance!

We will first use the data set with 10 observations to give an example of how we can calculate the variance:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Duration** | **Average\_Pulse** | **Max\_Pulse** | **Calorie\_Burnage** | **Hours\_Work** | **Hours\_Sleep** |
| 30 | 80 | 120 | 240 | 10 | 7 |
| 30 | 85 | 120 | 250 | 10 | 7 |
| 45 | 90 | 130 | 260 | 8 | 7 |
| 45 | 95 | 130 | 270 | 8 | 7 |
| 45 | 100 | 140 | 280 | 0 | 7 |
| 60 | 105 | 140 | 290 | 7 | 8 |
| 60 | 110 | 145 | 300 | 7 | 8 |
| 60 | 115 | 145 | 310 | 8 | 8 |
| 75 | 120 | 150 | 320 | 0 | 8 |
| 75 | 125 | 150 | 330 | 8 | 8 |

**Tip:** Variance is often represented by the symbol Sigma Square: σ^2

Step 1 to Calculate the Variance: Find the Mean

We want to find the variance of Average\_Pulse.

1. Find the mean:

(80+85+90+95+100+105+110+115+120+125) / 10 = 102.5

The mean is 102.5

Step 2: For Each Value - Find the Difference From the Mean

2. Find the difference from the mean for each value:

80 - 102.5 = -22.5  
85 - 102.5 = -17.5  
90 - 102.5 = -12.5  
95 - 102.5 = -7.5  
100 - 102.5 = -2.5  
105 - 102.5 = 2.5  
110 - 102.5 = 7.5  
115 - 102.5 = 12.5  
120 - 102.5 = 17.5  
125 - 102.5 = 22.5

Step 3: For Each Difference - Find the Square Value

3. Find the square value for each difference:

(-22.5)^2 = 506.25  
(-17.5)^2 = 306.25  
(-12.5)^2 = 156.25  
(-7.5)^2 = 56.25  
(-2.5)^2 = 6.25  
2.5^2 = 6.25  
7.5^2 = 56.25  
12.5^2 = 156.25  
17.5^2 = 306.25  
22.5^2 = 506.25

**Note:** We must square the values to get the total spread.

Step 4: The Variance is the Average Number of These Squared Values

4. Sum the squared values and find the average:

(506.25 + 306.25 + 156.25 + 56.25 + 6.25 + 6.25 + 56.25 + 156.25 + 306.25 + 506.25) / 10 = 206.25

The variance is 206.25.

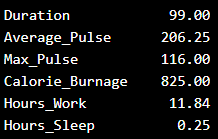
Use Python to Find the Variance of health\_data

We can use the var() function from Numpy to find the variance (remember that we now use the first data set with 10 observations):

Example

import numpy as np  
  
var = np.var(health\_data)  
print(var)

The output:



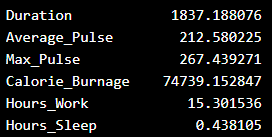
Use Python to Find the Variance of Full Data Set

Here we calculate the variance for each column for the full data set:

Example

import numpy as np  
  
var\_full = np.var(full\_health\_data)  
print(var\_full)

The output:



Correlation

Correlation measures the relationship between two variables.

We mentioned that a function has a purpose to predict a value, by converting input (x) to output (f(x)). We can say also say that a function uses the relationship between two variables for prediction.

Correlation Coefficient

The correlation coefficient measures the relationship between two variables.

The correlation coefficient can never be less than -1 or higher than 1.

* 1 = there is a perfect linear relationship between the variables (like Average\_Pulse against Calorie\_Burnage)
* 0 = there is no linear relationship between the variables
* -1 = there is a perfect negative linear relationship between the variables (e.g. Less hours worked, leads to higher calorie burnage during a training session)

Example of a Perfect Linear Relationship (Correlation Coefficient = 1)

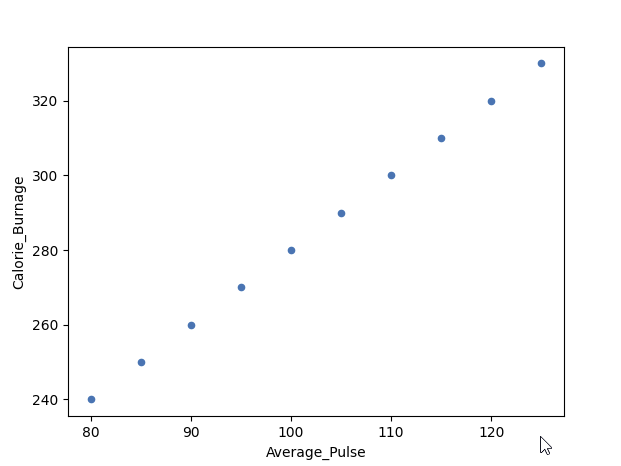
We will use scatterplot to visualize the relationship between Average\_Pulse and Calorie\_Burnage (we have used the small data set of the sports watch with 10 observations).

This time we want scatter plots, so we change kind to "scatter":

Example

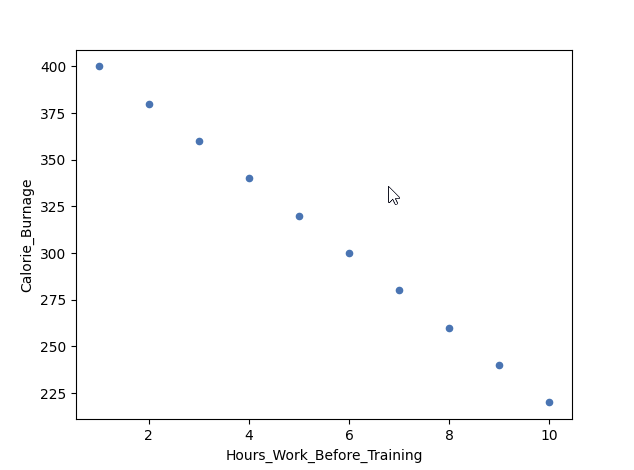
import matplotlib.pyplot as plt  
  
health\_data.plot(x ='Average\_Pulse', y='Calorie\_Burnage', kind='scatter')  
plt.show()

Output:



As we saw earlier, it exists a perfect linear relationship between Average\_Pulse and Calorie\_Burnage.

Example of a Perfect Negative Linear Relationship (Correlation Coefficient = -1)



We have plotted fictional data here. The x-axis represents the amount of hours worked at our job before a training session. The y-axis is Calorie\_Burnage.

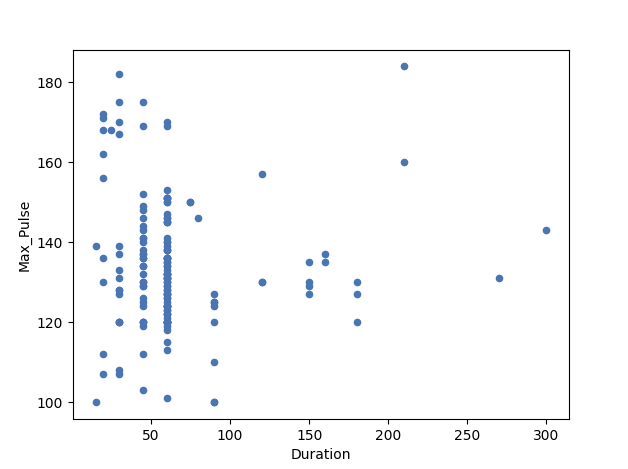
If we work longer hours, we tend to have lower calorie burnage because we are exhausted before the training session.

The correlation coefficient here is -1.

Example

import pandas as pd  
import matplotlib.pyplot as plt  
  
negative\_corr = {'Hours\_Work\_Before\_Training': [10,9,8,7,6,5,4,3,2,1],  
'Calorie\_Burnage': [220,240,260,280,300,320,340,360,380,400]}  
negative\_corr = pd.DataFrame(data=negative\_corr)  
  
negative\_corr.plot(x ='Hours\_Work\_Before\_Training', y='Calorie\_Burnage', kind='scatter')  
plt.show()

Example of No Linear Relationship (Correlation coefficient = 0)



Here, we have plotted Max\_Pulse against Duration from the full\_health\_data set.

As you can see, there is no linear relationship between the two variables. It means that longer training session does not lead to higher Max\_Pulse.

The correlation coefficient here is 0.

Example

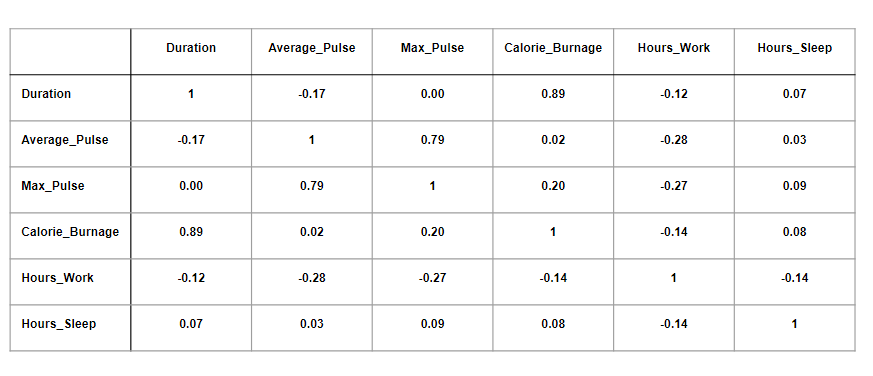
import matplotlib.pyplot as plt  
  
full\_health\_data.plot(x ='Duration', y='Max\_Pulse', kind='scatter')  
plt.show()

Correlation Matrix

A matrix is an array of numbers arranged in rows and columns.

A correlation matrix is simply a table showing the correlation coefficients between variables.

Here, the variables are represented in the first row, and in the first column:



The table above has used data from the full health data set.

Observations:

* We observe that Duration and Calorie\_Burnage are closely related, with a correlation coefficient of 0.89. This makes sense as the longer we train, the more calories we burn
* We observe that there is almost no linear relationships between Average\_Pulse and Calorie\_Burnage (correlation coefficient of 0.02)
* Can we conclude that Average\_Pulse does not affect Calorie\_Burnage? No. We will come back to answer this question later!

Correlation Matrix in Python

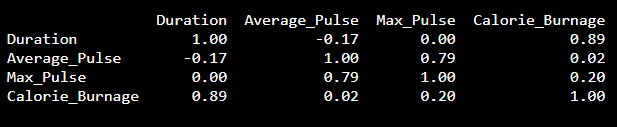
We can use the corr() function in Python to create a correlation matrix. We also use the round() function to round the output to two decimals:

Example

Corr\_Matrix = round(full\_health\_data.corr(),2)  
print(Corr\_Matrix)

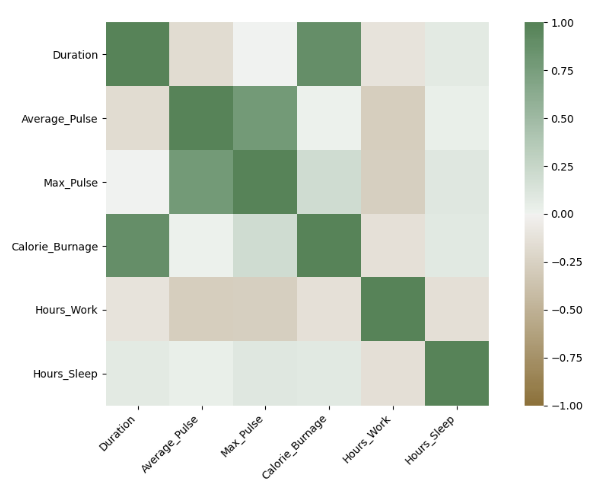
[Try it Yourself »](https://www.w3schools.com/datascience/trypython_multi_csv.asp?filename=demo_stat_matrix&multi=demo_data_full_multi)

Output:



Using a Heatmap

We can use a Heatmap to Visualize the Correlation Between Variables:



The closer the correlation coefficient is to 1, the greener the squares get.

The closer the correlation coefficient is to -1, the browner the squares get.

Use Seaborn to Create a Heatmap

We can use the Seaborn library to create a correlation heat map (Seaborn is a visualization library based on matplotlib):

Example

import matplotlib.pyplot as plt  
import seaborn as sns  
  
correlation\_full\_health = full\_health\_data.corr()  
  
axis\_corr = sns.heatmap(  
correlation\_full\_health,  
vmin=-1, vmax=1, center=0,  
cmap=sns.diverging\_palette(50, 500, n=500),  
square=True  
)  
  
plt.show()

Example Explained:

* Import the library seaborn as sns.
* Use the full\_health\_data set.
* Use sns.heatmap() to tell Python that we want a heatmap to visualize the correlation matrix.
* Use the correlation matrix. Define the maximal and minimal values of the heatmap. Define that 0 is the center.
* Define the colors with sns.diverging\_palette. n=500 means that we want 500 types of color in the same color palette.
* square = True means that we want to see squares.

Correlation Does Not Imply Causality

Correlation measures the numerical relationship between two variables.

A high correlation coefficient (close to 1), does not mean that we can for sure conclude an actual relationship between two variables.

A classic example:

* During the summer, the sale of ice cream at a beach increases
* Simultaneously, drowning accidents also increase as well

Does this mean that increase of ice cream sale is a direct cause of increased drowning accidents?

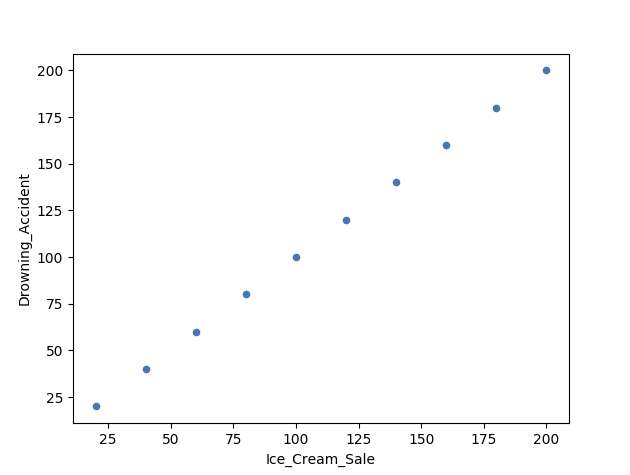
The Beach Example in Python

Here, we constructed a fictional data set for you to try:

Example

import pandas as pd  
import matplotlib.pyplot as plt  
  
Drowning\_Accident = [20,40,60,80,100,120,140,160,180,200]  
Ice\_Cream\_Sale = [20,40,60,80,100,120,140,160,180,200]  
Drowning = {"Drowning\_Accident": [20,40,60,80,100,120,140,160,180,200],  
"Ice\_Cream\_Sale": [20,40,60,80,100,120,140,160,180,200]}  
Drowning = pd.DataFrame(data=Drowning)  
  
Drowning.plot(x="Ice\_Cream\_Sale", y="Drowning\_Accident", kind="scatter")  
plt.show()  
  
correlation\_beach = Drowning.corr()  
print(correlation\_beach)

Output:



Correlation vs Causality - The Beach Example

In other words: can we use ice cream sale to predict drowning accidents?

The answer is - Probably not.

It is likely that these two variables are accidentally correlating with each other.

What causes drowning then?

* Unskilled swimmers
* Waves
* Cramp
* Seizure disorders
* Lack of supervision
* Alcohol (mis)use
* etc.

Let us reverse the argument:

Does a low correlation coefficient (close to zero) mean that change in x does not affect y?

Back to the question:

* Can we conclude that Average\_Pulse does not affect Calorie\_Burnage because of a low correlation coefficient?

The answer is no.

There is an important difference between correlation and causality:

* Correlation is a number that measures how closely the data are related
* Causality is the conclusion that x causes y.

**Tip:** Always critically reflect over the concept of causality when doing predictions!