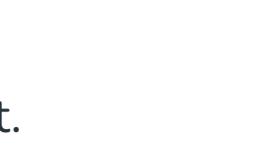


Pearson Correlation Test Between Two Variables - Python

Last Updated : 12 Jul, 2025



Correlation is a way to measure how **strongly** two variables are related. In simple terms, it tells us whether two things **increase or decrease** together. For example:

- Do people who study more hours get higher scores?
- Does car weight affect fuel efficiency?

These questions can be answered using a correlation test.

Uses of Correlation in Data Science

- Understand relationships between two numeric values.
- Make better decisions in data analysis and machine learning.
- Select features that are useful for prediction.
- Avoid using variables that are too similar (which can cause problems).

Types of Correlation Methods

There are two main types of correlation methods:

1. Parametric Correlation

- Measures the linear dependence between two variables (e.g., x and y).
- Assumes that the data follows a normal distribution.
- Example: **Pearson Correlation** (most commonly used).

2. Non-Parametric Correlation

- Used when data doesn't meet parametric assumptions.
- Based on rankings, not raw values.
- Examples: **Kendall's Tau**, **Spearman's Rho**

Note: The Pearson correlation method is the most widely used for linear relationships.

In this article, we will learn about **Pearson Correlation**:

Pearson Correlation

Pearson correlation is a number that tells us **how strongly** two values are **linearly related**.

It gives a result between **-1** and **+1**:

- **+1**: Perfect positive relationship (both **increase** together)
- **-1**: Perfect negative relationship (one **increases**, the other **decreases**)
- **0**: No linear relationship

Pearson Correlation Formula

$$r = \frac{\sum (x - m_x)(y - m_y)}{\sqrt{\sum (x - m_x)^2 \sum (y - m_y)^2}}$$

*x, y: Two numeric vectors of the same length n
m_x, m_y: Mean values of x and y respectively*

Important Notes on Pearson Correlation

- Not suitable for ordinal variables.
- Requires moderate sample size (**20–30**) for reliable estimates.
- Sensitive to **outliers**, which can distort results.

Computing Pearson Correlation in Python

Python has a built-in method **pearsonr()** from the **scipy.stats** module to find the Pearson correlation.

Syntax

```
from scipy.stats import pearsonr
pearsonr(x, y)
```

Parameters:

- **x, y** are the numeric lists or series.

Return Type: A tuple - (correlation coefficient, p-value)

Example 1: Pearson Correlation with Car Data

In this example, we find the correlation between **car weight** and **miles per gallon (mpg)**.

Here is a snapshot of the csv file used for this example:

	A	B
1	mpg	weight
2	18	3504
3	15	3693
4	18	3436
5	16	3433
6	17	3449

data.csv

To download the above csv file used in this article, [click here](#).

Code:

```
import pandas as pd
from scipy.stats import pearsonr

df = pd.read_csv("path_to_Auto.csv")

# Convert dataframe into series
l1 = df['weight']
l2 = df['mpg']

# Apply the pearsonr()
corr, _ = pearsonr(l1, l2)
print(f"Persons correlation: {corr:.3f} % corr")
```

Output:

Pearson correlation is: -0.878

Example 2: Anscombe's Quartet – Same Correlation, Different Patterns

Anscombe's Quartet is a famous example that shows why just using correlation numbers can be misleading. It has four small datasets with almost the same Pearson correlation, but very different shapes when plotted.

In this example, we will:

- Load the **four datasets** from a CSV file.
- Calculate the **Pearson correlation** for each dataset.
- **Plot** all datasets to see how they **differ** visually.

To download those 4 sets of 11 data-points, [click here](#).

```
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import pearsonr

# Load your CSV file
df = pd.read_csv("path_of_dataset")

# Store dataset names for looping
datasets = {
    "I": ("x1", "y1"),
    "II": ("x2", "y2"),
    "III": ("x3", "y3"),
    "IV": ("x4", "y4")
}

# Loop through each dataset and calculate Pearson correlation
for name, (x_col, y_col) in datasets.items():
    x = df[x_col]
    y = df[y_col]
    corr, _ = pearsonr(x, y)
    print(f"Dataset {name}: Pearson correlation = {corr:.3f}")

# Plot each dataset in a grid
fig, axs = plt.subplots(2, 2, figsize=(10, 8))
fig.suptitle('Anscombe-like Quartet Plots', fontsize=16)

for i, (name, (x_col, y_col)) in enumerate(datasets.items()):
    row = i // 2
    col = i % 2
    axs[row, col].scatter(df[x_col], df[y_col])
    axs[row, col].set_title(f"Dataset {name}")
    axs[row, col].set_xlabel(x_col)
    axs[row, col].set_ylabel(y_col)

plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```

Terminal Output:

Dataset I: Pearson correlation = 0.816
Dataset II: Pearson correlation = 0.816
Dataset III: Pearson correlation = 0.816
Dataset IV: Pearson correlation = 0.817

Here we can see that the correlation is same for all the datasets but let's take a look at their correlation graphs:

Graph Output:

Snapshot of the Plots

We can clearly see that the visual representation of them is very different, this shows why it's important to look at your data visually, not just rely on correlation values.

To know more about correlation please refer: [Covariance and Correlation](#).

Article Tags : Python data-science

Explore

Python Fundamentals

Python Data Structures

Advanced Python

Data Science with Python

Web Development with Python

Python Practice

Happy Honda Days

Love your Honda even more with this limited-time offer!

SEE OFFERS

Honda Passport

LEARN MORE

LEARN MORE