Calculate the Coefficient of Variation in Python

Authored by

Mohammed looti

November 6, 2025

What is the Coefficient of Variation (CV)?

The **Coefficient of Variation** (CV), often abbreviated as CV, is a standardized measure of dispersion of a probability distribution or **dataset**. Unlike the standard deviation, which is an absolute measure of variability, the CV expresses variability relative to the mean.

This relative measure makes the CV highly effective for comparing the level of variation between two different datasets, even if their underlying units or means are vastly different. It is generally presented as a percentage.

The Coefficient of Variation is calculated using the following simple formula:

CV = σ / μ

Where the components represent the core statistics of the sample or population:

σ: Denotes the **standard deviation** of the dataset, measuring the absolute spread of data points. (Linked 1/5)
μ: Denotes the arithmetic **mean** of the dataset, establishing the central reference point. (Linked 1/5)

Understanding the Importance of Relative Variability

In essence, the coefficient of variation is the ratio of the standard deviation to the mean. This normalization process provides critical context. For instance, a standard deviation of 10 might represent significant volatility if the mean is 50 (CV=20%), but negligible volatility if the mean is 1,000 (CV=1%).

By providing a unitless measure, the CV allows data scientists and analysts to objectively determine the consistency or risk associated with different data distributions. A lower coefficient of variation signifies that the data points are more tightly clustered around the mean, implying lower relative variability or risk.

Applications in Data Analysis and Finance

The coefficient of variation is frequently utilized in comparative statistics, especially when the datasets being analyzed are measured on different scales or have substantially different baseline values.

One of the most prominent real-world uses of the CV is in **finance**, where it helps investors evaluate the risk-return trade-off of potential investments. In this context, the standard deviation represents the volatility (risk), and the mean represents the expected return. By calculating the CV,

investors can compare the amount of risk taken for every unit of expected return.

Consider two mutual funds an investor is reviewing:

Mutual Fund A: Mean Expected Return (μ) = 9%, **Standard Deviation** (σ) = 12.4% (Linked 2/5)
Mutual Fund B: Mean Expected Return (μ) = 5%, Standard Deviation (σ) = 8.2%

Calculating the CV for both funds yields:

CV for Mutual Fund A = 12.4% / 9% = **1.38**

CV for Mutual Fund B = 8.2% / 5% = **1.64**

Since Mutual Fund A possesses a lower coefficient of variation (1.38 < 1.64), it offers a more favorable risk-adjusted return compared to Fund B, making it the statistically preferable investment, despite Fund B having a lower absolute standard deviation.

Implementing the Coefficient of Variation in Python

To calculate the Coefficient of Variation efficiently in **Python**, we utilize the numerical computing package **NumPy**, which provides optimized functions for statistical operations like standard deviation (`std`) and **mean** (`mean`). (Linked 2/5)

When working with sample data, it is standard practice to apply Bessel's correction by setting the Delta Degrees of Freedom (`ddof`) parameter to `1` in the `np.std()` function. This ensures an unbiased estimate of the population standard deviation. We can define a reusable lambda function for this calculation, multiplying by 100 to convert the result into a percentage:

import numpy as np

cv = lambda x: np.std(x, ddof=1) / np.mean(x) * 100

The following examples demonstrate how to apply this concise function across various data structures commonly used in Python data analysis.

Example 1: Calculating CV for a Single Array (Vector)

This first practical demonstration shows the calculation of the CV for a basic list of numerical observations. The function defined above handles the underlying **NumPy** computations seamlessly. (Linked 3/5)

 # Create vector of data points

data =

# Define function to calculate CV (ddof=1 for sample data)
cv = lambda x: np.std(x, ddof=1) / np.mean(x) * 100

# Execute calculation
cv(data)

9.234518

The calculated coefficient of variation for this array is **9.23%**. This percentage indicates that the standard deviation of the data is slightly more than nine percent of its mean value.

Example 2: CV for Multiple Variables in a Pandas DataFrame

For complex data structures involving multiple variables, the **Pandas** library is indispensable. Pandas DataFrames allow us to calculate the CV across several columns simultaneously using the `apply()` method. (Linked 1/5)

In the code below, we create a DataFrame with three columns, 'a', 'b', and 'c', and apply our established CV function to each column independently:

import numpy as np
import pandas as pd

# Define function to calculate cv
cv = lambda x: np.std(x, ddof=1) / np.mean(x) * 100

# Create pandas DataFrame
df = pd.DataFrame({'a': ,
'b': ,
'c': })

# Calculate CV for each column in data frame
df.apply(cv)

a 11.012892
b 8.330843
c 7.154009
dtype: float64

The results confirm that column 'c' has the lowest relative variation (7.15%), while column 'a'

exhibits the highest relative variation (11.01%) among the three variables.

Robustness: Handling Missing Values (NaN)

A key advantage of using **Pandas** and **NumPy** is their native handling of missing data points, represented as `np.nan`. When statistical operations like calculating the standard deviation or mean are performed, these libraries automatically exclude `NaN` values from the computation, thus preventing calculation errors and ensuring reliable results based on the existing observations.

The example below demonstrates the calculation when columns 'b' and 'c' contain missing entries:

import numpy as np
import pandas as pd

# Define function to calculate cv
cv = lambda x: np.std(x, ddof=1) / np.mean(x) * 100

# Create pandas DataFrame, now including missing values
df = pd.DataFrame({'a': ,
'b': ,
'c': })

# Calculate CV for each column in data frame
df.apply(cv)

a 11.012892
b 8.497612
c 5.860924
dtype: float64

Even with missing values, the Coefficient of Variation is successfully calculated for the valid entries, providing reliable comparative statistics for each variable.

Additional Resources for Statistical Analysis

The Coefficient of Variation is a powerful introductory tool for understanding relative risk and dispersion. Further exploration into descriptive statistics and advanced Python libraries will enrich your data analysis toolkit.

We recommend deepening your knowledge of related metrics that build upon the concepts of the standard deviation and the mean, such as skewness and kurtosis, or advanced financial ratios like the Sharpe Ratio, which often uses volatility (standard deviation) in its denominator.