Learning to Calculate Mean, Median, and Mode using Pandas in Python

Authored by

Mohammed looti

February 7, 2026

Understanding the central tendency of a dataset is often the critical first step in any robust data analysis project. For users of Python, the powerful Pandas library provides highly optimized and accessible methods for calculating core descriptive statistics. Among the most essential measures are the **mean**, the **median**, and the **mode**, which together offer distinct perspectives on the typical value and distribution of numerical data within a DataFrame.

These measures are fundamental because they provide rapid insight into the characteristics of your data, allowing you to quickly grasp where the data points cluster. Pandas simplifies these complex statistical calculations into efficient, single-line method calls applied directly to your DataFrame objects. Specifically, you utilize the `df.mean()`, `df.median()`, and `df.mode()` functions. When calculating these statistics, it is common practice to include the argument `numeric_only=True`, which instructs Pandas to exclude non-numeric columns, such as text identifiers, ensuring the calculation proceeds smoothly and accurately.

```
print(df.mean(numeric_only=True))
print(df.median(numeric_only=True))
print(df.mode(numeric_only=True))
```

This comprehensive guide will walk you through the practical application of these functions. We will establish a clear context using a sample dataset, demonstrate the necessary Python code, and critically interpret the results derived from each statistical measure. By the end of this article, you will be equipped to select and apply the most appropriate measure of central tendency for any given data analysis scenario.

Setting Up the Pandas DataFrame for Analysis

To effectively demonstrate the calculation and interpretation of the mean, median, and mode, we must first establish a representative sample dataset. For this tutorial, we will construct a Pandas DataFrame simulating hypothetical performance scores achieved by a group of basketball players across four competitive games. This structure provides a tangible and easily relatable scenario for understanding how these statistical measures translate into meaningful insights about performance trends.

A Pandas DataFrame serves as the foundational data structure for nearly all data manipulation tasks in Python. It is a two-dimensional, mutable table designed to hold potentially heterogeneous data types, featuring both labeled rows and columns. Our example DataFrame will contain player identifiers (categorical data) alongside their numerical scores recorded in 'game1', 'game2', 'game3', and 'game4' columns (quantitative data).

The following Python code initializes the DataFrame. We begin by importing the Pandas library,

followed by defining a Python dictionary where the keys correspond to the future column names and the values are lists containing the actual data points. This dictionary is then seamlessly converted into the DataFrame object, which we display to visualize the raw data structure we will be analyzing.

import pandas as pd

#create DataFrame
df = pd.DataFrame({'player': ,
'game1': ,
'game2': ,
'game3': ,
'game4': })

#view DataFrame
print(df)

player game1 game2 game3 game4
0 A 18 5 11 9
1 B 22 7 8 8
2 C 19 7 10 10
3 D 14 9 6 9
4 E 14 12 6 14
5 F 11 9 5 15
6 G 20 9 9 10
7 H 28 4 12 11

The resulting DataFrame, named `df`, successfully organizes the data into eight observations (players) and five columns, four of which contain numerical score data suitable for statistical computation. This structured format allows us to proceed directly to calculating the measures of central tendency to understand the aggregate performance across the games.

Calculating the Mean (Arithmetic Average) in Pandas

The **mean**, formally known as the arithmetic average, is the most common measure of central tendency. It is calculated by summing all values within a dataset and then dividing that sum by the total number of values. The mean provides a straightforward indication of the typical value in a dataset, assuming the data is evenly distributed, effectively representing the center of mass of the data distribution.

However, it is vital to recognize that the mean is highly susceptible to the influence of outliers--

extreme values that are significantly higher or lower than the majority of the data points. If a dataset is heavily skewed, a few exceptional values can pull the mean dramatically in their direction, potentially leading to a misleading representation of the true "average." Therefore, analysts must always consider the data's distribution profile before relying solely on the mean.

In the Pandas ecosystem, calculating the mean for every numerical column in a DataFrame is achieved simply using the `df.mean()` method. By passing the argument `numeric_only=True`, we ensure that the method selectively processes only the columns containing numerical data types, such as 'game1' through 'game4', avoiding errors related to non-numeric columns like 'player'. Below is the application of this function to our basketball scores DataFrame:

#calculate mean of each numeric column
print(df.mean(numeric_only=True))

game1 18.250
game2 7.750
game3 8.375
game4 10.750
dtype: float64

The output provides the arithmetic average for each game score column:

The average score for **game1** is **18.25**.
The average score for **game2** is **7.75**.
The average score for **game3** is **8.375**.
The average score for **game4** is **10.75**.

These figures allow for immediate comparison of overall performance; for instance, the average performance was strongest in game1 and weakest in game2 across all recorded player attempts.

Calculating the Median in Pandas

The **median** stands as another essential measure of central tendency, defined as the middle value in a dataset when all data points are ordered sequentially from smallest to largest. For datasets with an odd number of observations, the median is the single central value. When dealing with an even number of observations, the median is typically determined by calculating the average of the two middle values.

A significant advantage of the median is its exceptional resistance to the influence of outliers. Unlike the mean, the median's value is purely positional, meaning it is not affected by the magnitude of extreme data points. Consequently, the median is the preferred measure of central

tendency for datasets exhibiting skewed distributions--such as income or housing prices--where it provides a far more accurate representation of the "typical" value that is not distorted by a few exceptional entries.

In Pandas, the calculation of the median is handled by the straightforward `df.median()` method. Similar to the mean calculation, we specify `numeric_only=True` to focus the calculation exclusively on the numerical columns of the DataFrame. This method reliably returns the middle point for the score distributions in each game, offering a robust measure that complements the mean.

#calculate median of each numeric column
print(df.median(numeric_only=True))

game1 18.5
game2 8.0
game3 8.5
game4 10.0
dtype: float64

Reviewing the median scores for our sample data:

The median score for **game1** is **18.5**.
The median score for **game2** is **8**.
The median score for **game3** is **8.5**.
The median score for **game4** is **10**.

By comparing these median values (e.g., game1 mean 18.25 vs. median 18.5) with the previously calculated means, we can quickly infer the symmetry of the data's distribution. The proximity of the mean and median in our example suggests that the score distributions are relatively symmetric and not heavily skewed by extreme individual performance.

Calculating the Mode in Pandas

The **mode** is defined as the value or values that occur most frequently within a dataset. It holds a unique position among central tendency measures because it is the only one applicable to both numerical data and categorical data. A distribution may possess a single mode (unimodal), two modes (bimodal), or several modes (multimodal), or conversely, have no mode at all if every value appears only once.

For numerical datasets, the mode helps identify the most prevalent scores or measurements, indicating dominant trends or common outcomes. For categorical data, the mode directly points to

the most popular category, such as the most frequently chosen product option or brand. Understanding the mode is essential for tasks like inventory management or identifying the most typical customer behavior.

The Pandas method for calculating the mode is `df.mode()`. It is important to note that `df.mode()` consistently returns a DataFrame as its output, even if only one mode exists for a column. This design accommodates the possibility of multiple modes (ties for the highest frequency). If a column has multiple values that share the highest count, all of them will be listed in separate rows of the resulting DataFrame, with `NaN` filling cells where fewer modes exist for corresponding columns.

#calculate mode of each numeric column
print(df.mode(numeric_only=True))

game1 game2 game3 game4
0 14.0 9.0 6.0 9
1 NaN NaN NaN 10

Interpreting the output of the mode calculation yields the most common scores:

The mode in **game1** is **14**.
The mode in **game2** is **9**.
The mode in **game3** is **6**.
The **game4** column is multimodal, featuring two modes: **9** and **10**. This indicates that scores of 9 and 10 were achieved with equal and maximum frequency among the players in that specific game.

The mode's ability to reveal multiple points of peak frequency provides invaluable insight into distributions that might have distinct clusters or common performance levels, unlike the single-point measurements provided by the mean and median.

Comparing Mean, Median, and Mode: Selecting the Right Statistic

The decision of which measure of central tendency--mean, median, or mode--to utilize is fundamentally dependent on the nature of your data and its underlying distribution. Each statistic presents a unique summary of the data's "center," and selecting the appropriate one is crucial for drawing valid conclusions.

The **mean** is the optimal choice for data that is normally or symmetrically distributed and lacks significant outliers. It incorporates every value in its calculation, providing a comprehensive measure of the entire dataset. However, its major drawback is its sensitivity; for instance, when analyzing salaries, the mean can be inflated by a small number of extremely high earners, giving a

false sense of the typical wage.

The **median** is the superior choice for analyzing data that is highly skewed or contains notable outliers. Because the median relies only on the position of the values, it remains unaffected by the magnitude of extreme scores. This makes it particularly robust for variables such as real estate prices, personal income, or reaction times, where the integrity of the central measure must be preserved against distorting factors.

The **mode** is irreplaceable when dealing with categorical or discrete numerical data where you need to identify the most common occurrence. While it may not strictly represent the mathematical center, it provides essential information about prevalence and frequency. For example, determining the most common software version used by customers or the most popular color of a product demands the use of the mode.

In advanced descriptive statistics, it is highly recommended to calculate all three measures simultaneously. Comparing the mean, median, and mode offers a powerful diagnostic tool regarding the data's shape: if they are nearly identical, the distribution is symmetrical. If the mean is greater than the median, the data is positively skewed (long tail to the right), suggesting the presence of high-value outliers. Conversely, if the mean is less than the median, the data is negatively skewed (long tail to the left).

Beyond Basic Measures: Leveraging `describe()` for Comprehensive Analysis

While the individual calculation methods for mean, median, and mode are indispensable, Pandas offers an accelerated, all-in-one solution for numerical data summarization: the `df.describe()` method. This powerful function generates a comprehensive suite of descriptive statistics for every numerical column in your DataFrame using just a single command, significantly enhancing the efficiency of initial data exploration.

The standard output of `df.describe()` includes not only the count and **mean**, but also the standard deviation, the minimum and maximum values, and the interquartile range defined by the 25th percentile (Q1), the 50th percentile (which is the **median**), and the 75th percentile (Q3). This holistic view provides immediate insight into the central tendency, dispersion, and range of your data.

Integrating `df.describe()` into your data profiling workflow is highly advantageous. It allows for the rapid identification of data quality issues, such as extreme outliers (which would show up in the minimum/maximum values), and provides a quick assessment of data spread and variance before undertaking more complex modeling or visualization tasks. It serves as an essential foundation for any thorough data analysis.

Conclusion

Mastering the application of mean, median, and mode within the Pandas framework is an essential cornerstone of modern data analysis. These three measures of central tendency, while seemingly basic, provide distinct and critical views into the nature of your data--from the arithmetic average to the positional center and the most frequent occurrence.

The efficiency and simplicity of the Pandas API--including `df.mean()`, `df.median()`, and `df.mode()`--make extracting these key statistics from numerical columns within your DataFrames an effortless task. By understanding the specific strengths and weaknesses of each measure--especially their relationship to data distribution and the presence of outliers--you ensure that your interpretation of the data is accurate and not skewed by statistical artifacts.

Whether you are performing quality assurance checks, evaluating key performance indicators, or preparing data for machine learning models, the ability to quickly and correctly compute and interpret these central tendency statistics using Pandas is a fundamental skill that significantly strengthens your overall data science toolkit.

Additional Resources

To further enhance your Pandas skills and explore more advanced data manipulation techniques, consider reviewing the following tutorials: