How to Find Mean, Median and Mode Using Python

This is the first article in a series of tutorials on data science. We will cover the following topics in this article:

Sum(): total

Max(): highest value

Min(): lowest value

Mean()/avg(): total/no. of values

Var():

Stddev(): sqrt of variance

Median(): 1 2 3 4 5 , =3 , 1 2 3 4 5 6 = (3+4)/2

Mode(): freq of digit

Range(): max-min =

Quartertile() : 25% , 50% , 75% , 100%

Percentile(): 13%, 24%, 45%, 88%, 97% ,100%

1. Types of data
2. Mean
3. Median
4. Impact of outliers on mean and median results
5. Mode

Without delving too deep into the coding aspect, we will see what mean, median, and mode are, and how to derive them in Python. We will discuss codes in the subsequent articles that focus on Python libraries. Let us begin by discussing the three different types of data:

1. Numerical Data
2. Categorical Data
3. Ordinal Data

## ****1.**** Numerical Data

It’s probably the most common type of data. Basically, it represents some quantifiable thing that you can measure. Some examples are heights of people, page load times, and stock prices.

Numerical data can be subdivided into two types:

**1.1)Discrete data**

Discrete data refers to the measure of things in whole numbers (integers). For example, the number of purchases made by a customer in a year. Since the number of things that a person buys cannot be three and a half, or four and a third – it must be a whole number like four or five things – this kind of data falls under the discrete category.

**1.2) Continuous data**

In contrast to discrete data, continuous data includes all numbers possible between any two integers or whole numbers. For example, the height of something. It could be 9.2345 inches or 9.7219 inches, or any other fraction between the two whole numbers nine and ten. Another example could be the amount of rainfall recorded in a day. Again, the amount does not necessarily have to be a whole number. It could be 6.5 mm or 23.1 mm of rainfall, depending on the shower God’s fancy.

## ****2.****Categorical Data

This type of data is non-numeric. We use it to quantify things in categories like gender, ethnicity, nationality, political party, etc. We can assign numbers to the categories, but the numbers would not, in that case, represent their value per say. They will only separate one type from the other – type one from type two or three. For example, while calculating India’s population, Bangalore could be city number one, Mumbai number two, and so on. The data collected, however, would still represent the number of people in Bangalore and Mumbai, and not the population of one and two. These numbers have no value of their own in this context.

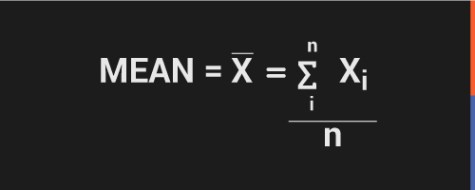
## ****3.**** Ordinal Data

Ordinal data is an amalgamation of numerical and categorical data. Simply put, this data type consists of categories that are in order. The intervals between categories are not known. Good examples of this data type are movie or music ratings that use stars to denote quality. Numbers simply represent the good and bad categories.  A movie with a 5-star rating is obviously very good as opposed to a movie with only 1-star, which, very likely, is terrible. Note that the numbers in this example do denote value. Mathematically speaking, 5 is greater than 1. This difference in value is used to differentiate good films from bad. Good films receive a higher rating of 4 or 5, while bad films only get a lower rating of 1 or 2.

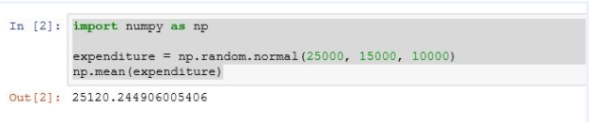
## Mean

Mean is simply another name for average. To calculate the mean of a data set, divide the sum of all values by the number of values.

Consider the following set of numbers: {5,2,2,7}. The mean is  (5 + 2 + 2 + 7) / 4 = 16 / 4 = 4. We use the symbol “x-bar” to represent the mean of a sample data. The formula to compute the mean for a set of n values is:



We will explain terms like standard deviation and normal distribution in subsequent blogs. For now, all we need to keep in mind is the sample size (10,000), and the mean (25,000). Don’t worry about other components like numpy for code, or the criteria for calculation.



**Code:**

|  |  |
| --- | --- |
| 1  2  3 | import numpy as np  expenditure = np.random.normal(25000, 15000, 10000)  np.mean(expenditure) |

## Median

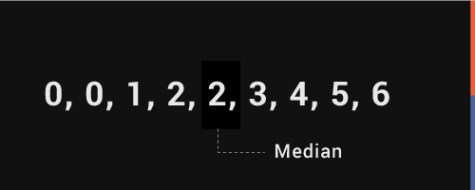
Median, in simple words, is the number that lies in the middle of a list of ordered numbers. The numbers may be in the ascending or descending order. Let us consider the following data set:

0,2,3,4,5,1,2,0,6

After sorting these numbers in the ascending order, we get the following list:

0,0,1,2,2,3,4,5,6

2 – the number in the center (fifth from either side) – is the median in this example.



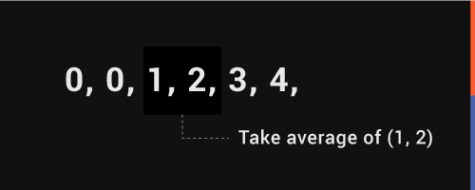
The median is easy to find when there are odd number of elements in the data set. When there are even number of elements, you need to take the average of the two numbers that fall in the center of the ordered list. So, if we consider the following data set:

0,0,1,4,2,3

After sorting the numbers, we get the following list:

0,0,1,2,3,4

The average of 1 and 2, in this case, is the median.

  
Median = (1 + 2) / 2

         = 1.5

Median is 1.5.

**Let us now see how to find the median in Python.**

To get the median of a data set in Python, run the script “np.median(expenditure)” in Jupyter notebook.

The median of expenditures from the previous example is 25,179.05. In this case, it is not very far from the mean, which is 25,120.24.

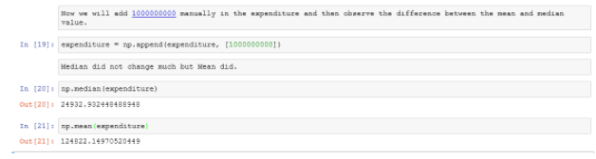
Before we discuss mode, let us understand what outliers are, and how they impact the mean and median of a data set.

* Any value in a dataset that is at an abnormal distance from all other values can be termed as an outlier. Outliers generally tend to skew the mean radically.
* Outliers can be present in the dataset with very high value or with a very low value.

Let us see how by passing a large value(1000000000) manually in the expenditure and then calculating the mean and median.

**Code:**

|  |  |
| --- | --- |
| 1  2  3 | expenditure = np.append(expenditure, [1000000000])  np.median(expenditure)  np.mean(expenditure) |



What we find is that the large value, or the outlier, changes the median to some extent (from 25,179.05 to 24,932.93), and the mean to a great extent (from 25,120.244 to 1,24,822.14). The outlier is an abnormal value because of its potential to skew the mean of a data set radically, and thereby misrepresenting the data set altogether.

## Mode

Mode is not used as often as mean or median. It is that value which appears the most number of times in a data set. For example, in the following data set, 0 appears the most number of times. Therefore, it is the mode.

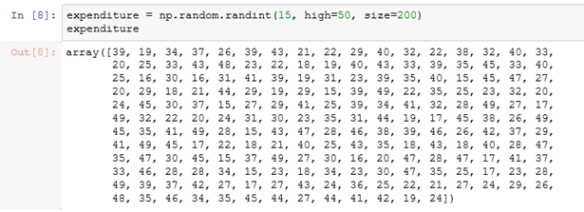
0,0,1,2,3,0,4,5,0

**Mode in Python:**

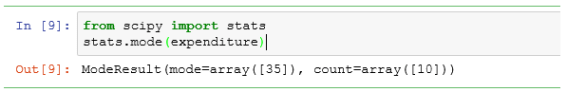
Let’s generate a random expenditure set data using the script below.

expenditure = np.random.randint(15, high=50, size=200)

expenditure



|  |  |
| --- | --- |
| 1  2 | from scipy import stats  stats.mode(expenditure) |



# [statistics](https://docs.python.org/3/library/statistics.html#module-statistics) — Mathematical statistics functions[¶](https://docs.python.org/3/library/statistics.html#module-statistics)

## Averages and measures of central location

These functions calculate an average or typical value from a population or sample.

|  |  |
| --- | --- |
| [mean()](https://docs.python.org/3/library/statistics.html#statistics.mean) | Arithmetic mean (“average”) of data. |
| [harmonic\_mean()](https://docs.python.org/3/library/statistics.html#statistics.harmonic_mean) | Harmonic mean of data. |
| [median()](https://docs.python.org/3/library/statistics.html#statistics.median) | Median (middle value) of data. |
| [median\_low()](https://docs.python.org/3/library/statistics.html#statistics.median_low) | Low median of data. |
| [median\_high()](https://docs.python.org/3/library/statistics.html#statistics.median_high) | High median of data. |
| [median\_grouped()](https://docs.python.org/3/library/statistics.html#statistics.median_grouped) | Median, or 50th percentile, of grouped data. |
| [mode()](https://docs.python.org/3/library/statistics.html#statistics.mode) | Mode (most common value) of discrete data. |

## 9.7.2. Measures of spread

These functions calculate a measure of how much the population or sample tends to deviate from the typical or average values.

|  |  |
| --- | --- |
| [pstdev()](https://docs.python.org/3/library/statistics.html#statistics.pstdev) | Population standard deviation of data. |
| [pvariance()](https://docs.python.org/3/library/statistics.html#statistics.pvariance) | Population variance of data. |
| [stdev()](https://docs.python.org/3/library/statistics.html#statistics.stdev) | Sample standard deviation of data. |
| [variance()](https://docs.python.org/3/library/statistics.html#statistics.variance) | Sample variance of data. |

## Function details

Note: The functions do not require the data given to them to be sorted. However, for reading convenience, most of the examples show sorted sequences.

statistics.**mean**(data)

Return the sample arithmetic mean of data which can be a sequence or iterator.

The arithmetic mean is the sum of the data divided by the number of data points. It is commonly called “the average”, although it is only one of many different mathematical averages. It is a measure of the central location of the data.

If data is empty, [StatisticsError](https://docs.python.org/3/library/statistics.html" \l "statistics.StatisticsError" \o "statistics.StatisticsError) will be raised.

Some examples of use:

>>>

**>>>** mean([1, 2, 3, 4, 4])

2.8

**>>>** mean([-1.0, 2.5, 3.25, 5.75])

2.625

**>>> from** **fractions** **import** Fraction **as** F

**>>>** mean([F(3, 7), F(1, 21), F(5, 3), F(1, 3)])

Fraction(13, 21)

**>>> from** **decimal** **import** Decimal **as** D

**>>>** mean([D("0.5"), D("0.75"), D("0.625"), D("0.375")])

Decimal('0.5625')

**Note**

The mean is strongly affected by outliers and is not a robust estimator for central location: the mean is not necessarily a typical example of the data points. For more robust, although less efficient, measures of central location, see [median()](https://docs.python.org/3/library/statistics.html#statistics.median) and [mode()](https://docs.python.org/3/library/statistics.html#statistics.mode). (In this case, “efficient” refers to statistical efficiency rather than computational efficiency.)

The sample mean gives an unbiased estimate of the true population mean, which means that, taken on average over all the possible samples, mean(sample)converges on the true mean of the entire population. If data represents the entire population rather than a sample, then mean(data) is equivalent to calculating the true population mean μ.

statistics.**harmonic\_mean**(data)

Return the harmonic mean of data, a sequence or iterator of real-valued numbers.

The harmonic mean, sometimes called the subcontrary mean, is the reciprocal of the arithmetic [mean()](https://docs.python.org/3/library/statistics.html#statistics.mean) of the reciprocals of the data. For example, the harmonic mean of three values a, b and c will be equivalent to 3/(1/a + 1/b + 1/c).

The harmonic mean is a type of average, a measure of the central location of the data. It is often appropriate when averaging quantities which are rates or ratios, for example speeds. For example:

Suppose an investor purchases an equal value of shares in each of three companies, with P/E (price/earning) ratios of 2.5, 3 and 10. What is the average P/E ratio for the investor’s portfolio?

>>>

**>>>** harmonic\_mean([2.5, 3, 10]) *# For an equal investment portfolio.*

3.6

Using the arithmetic mean would give an average of about 5.167, which is too high.

[StatisticsError](https://docs.python.org/3/library/statistics.html#statistics.StatisticsError) is raised if data is empty, or any element is less than zero.

*New in version 3.6.*

statistics.**median**(data)

Return the median (middle value) of numeric data, using the common “mean of middle two” method. If data is empty, [StatisticsError](https://docs.python.org/3/library/statistics.html" \l "statistics.StatisticsError" \o "statistics.StatisticsError) is raised. data can be a sequence or iterator.

The median is a robust measure of central location, and is less affected by the presence of outliers in your data. When the number of data points is odd, the middle data point is returned:

>>>

**>>>** median([1, 3, 5])

3

When the number of data points is even, the median is interpolated by taking the average of the two middle values:

>>>

**>>>** median([1, 3, 5, 7])

4.0

This is suited for when your data is discrete, and you don’t mind that the median may not be an actual data point.

**See also**

[median\_low()](https://docs.python.org/3/library/statistics.html#statistics.median_low), [median\_high()](https://docs.python.org/3/library/statistics.html" \l "statistics.median_high" \o "statistics.median_high), [median\_grouped()](https://docs.python.org/3/library/statistics.html" \l "statistics.median_grouped" \o "statistics.median_grouped)

statistics.**median\_low**(data)

Return the low median of numeric data. If data is empty, [StatisticsError](https://docs.python.org/3/library/statistics.html" \l "statistics.StatisticsError" \o "statistics.StatisticsError) is raised. data can be a sequence or iterator.

The low median is always a member of the data set. When the number of data points is odd, the middle value is returned. When it is even, the smaller of the two middle values is returned.

>>>

**>>>** median\_low([1, 3, 5])

3

**>>>** median\_low([1, 3, 5, 7])

3

Use the low median when your data are discrete and you prefer the median to be an actual data point rather than interpolated.

statistics.**median\_high**(data)

Return the high median of data. If data is empty, [StatisticsError](https://docs.python.org/3/library/statistics.html" \l "statistics.StatisticsError" \o "statistics.StatisticsError) is raised. data can be a sequence or iterator.

The high median is always a member of the data set. When the number of data points is odd, the middle value is returned. When it is even, the larger of the two middle values is returned.

>>>

**>>>** median\_high([1, 3, 5])

3

**>>>** median\_high([1, 3, 5, 7])

5

Use the high median when your data are discrete and you prefer the median to be an actual data point rather than interpolated.

statistics.**median\_grouped**(data, interval=1)

Return the median of grouped continuous data, calculated as the 50th percentile, using interpolation. If data is empty, [StatisticsError](https://docs.python.org/3/library/statistics.html" \l "statistics.StatisticsError" \o "statistics.StatisticsError) is raised. data can be a sequence or iterator.

>>>

**>>>** median\_grouped([52, 52, 53, 54])

52.5

In the following example, the data are rounded, so that each value represents the midpoint of data classes, e.g. 1 is the midpoint of the class 0.5–1.5, 2 is the midpoint of 1.5–2.5, 3 is the midpoint of 2.5–3.5, etc. With the data given, the middle value falls somewhere in the class 3.5–4.5, and interpolation is used to estimate it:

>>>

**>>>** median\_grouped([1, 2, 2, 3, 4, 4, 4, 4, 4, 5])

3.7

Optional argument interval represents the class interval, and defaults to 1. Changing the class interval naturally will change the interpolation:

>>>

**>>>** median\_grouped([1, 3, 3, 5, 7], interval=1)

3.25

**>>>** median\_grouped([1, 3, 3, 5, 7], interval=2)

3.5

This function does not check whether the data points are at least interval apart.

**CPython implementation detail:** Under some circumstances, [median\_grouped()](https://docs.python.org/3/library/statistics.html" \l "statistics.median_grouped" \o "statistics.median_grouped) may coerce data points to floats. This behaviour is likely to change in the future.

**See also**

* “Statistics for the Behavioral Sciences”, Frederick J Gravetter and Larry B Wallnau (8th Edition).
* Calculating the [median](https://www.ualberta.ca/~opscan/median.html).
* The [SSMEDIAN](https://help.gnome.org/users/gnumeric/stable/gnumeric.html#gnumeric-function-SSMEDIAN) function in the Gnome Gnumeric spreadsheet, including [this discussion](https://mail.gnome.org/archives/gnumeric-list/2011-April/msg00018.html).

statistics.**mode**(data)

Return the most common data point from discrete or nominal data. The mode (when it exists) is the most typical value, and is a robust measure of central location.

If data is empty, or if there is not exactly one most common value, [StatisticsError](https://docs.python.org/3/library/statistics.html" \l "statistics.StatisticsError" \o "statistics.StatisticsError) is raised.

mode assumes discrete data, and returns a single value. This is the standard treatment of the mode as commonly taught in schools:

>>>

**>>>** mode([1, 1, 2, 3, 3, 3, 3, 4])

3

The mode is unique in that it is the only statistic which also applies to nominal (non-numeric) data:

>>>

**>>>** mode(["red", "blue", "blue", "red", "green", "red", "red"])

'red'

statistics.**pstdev**(data, mu=None)

Return the population standard deviation (the square root of the population variance). See [pvariance()](https://docs.python.org/3/library/statistics.html" \l "statistics.pvariance" \o "statistics.pvariance) for arguments and other details.

>>>

**>>>** pstdev([1.5, 2.5, 2.5, 2.75, 3.25, 4.75])

0.986893273527251

statistics.**pvariance**(data, mu=None)

Return the population variance of data, a non-empty iterable of real-valued numbers. Variance, or second moment about the mean, is a measure of the variability (spread or dispersion) of data. A large variance indicates that the data is spread out; a small variance indicates it is clustered closely around the mean.

If the optional second argument mu is given, it should be the mean of data. If it is missing or None (the default), the mean is automatically calculated.

Use this function to calculate the variance from the entire population. To estimate the variance from a sample, the [variance()](https://docs.python.org/3/library/statistics.html#statistics.variance) function is usually a better choice.

Raises [StatisticsError](https://docs.python.org/3/library/statistics.html" \l "statistics.StatisticsError" \o "statistics.StatisticsError) if data is empty.

Examples:

>>>

**>>>** data = [0.0, 0.25, 0.25, 1.25, 1.5, 1.75, 2.75, 3.25]

**>>>** pvariance(data)

1.25

If you have already calculated the mean of your data, you can pass it as the optional second argument mu to avoid recalculation:

>>>

**>>>** mu = mean(data)

**>>>** pvariance(data, mu)

1.25

This function does not attempt to verify that you have passed the actual mean as mu. Using arbitrary values for mu may lead to invalid or impossible results.

Decimals and Fractions are supported:

>>>

**>>> from** **decimal** **import** Decimal **as** D

**>>>** pvariance([D("27.5"), D("30.25"), D("30.25"), D("34.5"), D("41.75")])

Decimal('24.815')

**>>> from** **fractions** **import** Fraction **as** F

**>>>** pvariance([F(1, 4), F(5, 4), F(1, 2)])

Fraction(13, 72)

**Note**

When called with the entire population, this gives the population variance σ². When called on a sample instead, this is the biased sample variance s², also known as variance with N degrees of freedom.

If you somehow know the true population mean μ, you may use this function to calculate the variance of a sample, giving the known population mean as the second argument. Provided the data points are representative (e.g. independent and identically distributed), the result will be an unbiased estimate of the population variance.

statistics.**stdev**(data, xbar=None)

Return the sample standard deviation (the square root of the sample variance). See [variance()](https://docs.python.org/3/library/statistics.html#statistics.variance) for arguments and other details.

>>>

**>>>** stdev([1.5, 2.5, 2.5, 2.75, 3.25, 4.75])

1.0810874155219827

statistics.**variance**(data, xbar=None)

Return the sample variance of data, an iterable of at least two real-valued numbers. Variance, or second moment about the mean, is a measure of the variability (spread or dispersion) of data. A large variance indicates that the data is spread out; a small variance indicates it is clustered closely around the mean.

If the optional second argument xbar is given, it should be the mean of data. If it is missing or None (the default), the mean is automatically calculated.

Use this function when your data is a sample from a population. To calculate the variance from the entire population, see [pvariance()](https://docs.python.org/3/library/statistics.html" \l "statistics.pvariance" \o "statistics.pvariance).

Raises [StatisticsError](https://docs.python.org/3/library/statistics.html" \l "statistics.StatisticsError" \o "statistics.StatisticsError) if data has fewer than two values.

Examples:

>>>

**>>>** data = [2.75, 1.75, 1.25, 0.25, 0.5, 1.25, 3.5]

**>>>** variance(data)

1.3720238095238095

If you have already calculated the mean of your data, you can pass it as the optional second argument xbar to avoid recalculation:

>>>

**>>>** m = mean(data)

**>>>** variance(data, m)

1.3720238095238095

This function does not attempt to verify that you have passed the actual mean as xbar. Using arbitrary values for xbar can lead to invalid or impossible results.

Decimal and Fraction values are supported:

>>>

**>>> from** **decimal** **import** Decimal **as** D

**>>>** variance([D("27.5"), D("30.25"), D("30.25"), D("34.5"), D("41.75")])

Decimal('31.01875')

**>>> from** **fractions** **import** Fraction **as** F

**>>>** variance([F(1, 6), F(1, 2), F(5, 3)])

Fraction(67, 108)