standard_deviation

February 5, 2025

1 Exploring Standard Deviation

Standard deviation is a useful measure when performing exploratory data analysis because it provides insight into the spread or variability of your data. It does this by quantifying, on average, how far each data point lies from the mean of the dataset. Here are several key reasons why standard deviation proves useful:

- Identifying Data Dispersion: If the standard deviation is small, it indicates that most of the observations cluster tightly around the mean, suggesting relatively low variability. Conversely, a large standard deviation signals that the observations are more widely dispersed, suggesting greater variability in the dataset.
- Comparing Different Datasets: When you have multiple datasets—for instance, separate columns in a dataframe—you can compare their standard deviations to assess which variables exhibit more fluctuation. This is often one of the first steps in exploratory analysis, helping you determine which features may warrant further investigation.
- Spotting Outliers and Anomalies: While the mean alone can be skewed by extreme values, looking at the standard deviation alongside the mean can help flag outliers. If certain observations are more than a couple of standard deviations away from the mean, they might be anomalies that deserve extra scrutiny.
- Understanding Distribution Shape: Though it doesn't replace a full examination of the data distribution (like creating histograms or boxplots), the standard deviation can hint at whether the data is spread out or tightly centred. This assists in deciding which further statistical techniques might be appropriate or whether transformations may be needed.
- Normalisation and Scaling: Standard deviation is often used in techniques that require standard scaling or z-score normalisation. By subtracting the mean and dividing by the standard deviation, each variable can be placed on a comparable scale—useful in numerous machine learning and statistical methods.

Including the standard deviation in your initial data summaries helps guide further exploration. It reveals whether data is tightly packed or widely scattered, aids in spotting unusual observations, and can inform decisions on the choice of subsequent analytical or modelling methods.

[1]:
```
import pandas as pd
import numpy as np
```

1.1 First, get some data

```
[2]: students_df = pd.read_csv('./data/students.csv')
```

Before continuing, we'll get rid of rows with missing data and duplicate rows.

```
[3]: students_df = students_df.dropna().drop_duplicates()
```

1.2 Calculating the standard deviation

The describe method gives us a statistical summary of the numeric variables in the data frame. This includes standard deviation.

```
[4]: students_df.describe()
```

```
[4]:              Age  Year of Study
     count   97.000000      97.000000
     mean    23.917526       2.773196
     std     22.852184       3.398851
     min      5.000000       1.000000
     25%     20.000000       1.000000
     50%     22.000000       2.000000
     75%     24.000000       3.000000
     max    245.000000      34.000000
```

Pandas Series objects have an std method that we can use to calculate the standard deviation.

```
[5]: students_df['Age'].std()
```

```
[5]: 22.852183567756345
```

Pandas DataFrame objects also have an std method. Before using it, we need to make sure that the data frame consists only of numeric columns.

```
[6]: students_df_numeric = students_df.select_dtypes(include=np.number)

     students_df_numeric.std()
```

```
[6]: Age              22.852184
     Year of Study     3.398851
     dtype: float64
```

1.3 Sample and Population Standard Deviation

When performing an exploratory data analysis, you may encounter two versions of the standard deviation: the population standard deviation and the sample standard deviation. Here's what distinguishes them and how to decide which one to use:

1. **Formulaic Difference**
   - **Population Standard Deviation** (σ): Uses the entire set of data (the population). The denominator is N, the total number of data points in the population.

$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N}}$
Here, μ is the population mean.
Sample Standard Deviation (s): Used when you have a subset (a sample) of a larger population. The denominator is *N - 1*, rather than *N*, a correction called Bessel's correction.
$s = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{n-1}}$
Here, $\bar{x}$ is the sample mean, and n is the sample size.

2. **Conceptual Reasoning**
   - If you have data for the *entire population*, then every value that possibly exists is included, so you are directly measuring the true spread of the population.

   - If you only have a *sample*, you have less than the full information, so the sample-based standard deviation needs an adjustment to avoid underestimating the true population variance.
3. **When to Use Which**
   - **Population Standard Deviation**: When you are certain that you have complete data for every individual or entity of interest. This situation is more common in theoretical examples or when the dataset is truly exhaustive (for instance, the exact heights of all pupils in one specific classroom if you have measured them all).

   - **Sample Standard Deviation**: When you have only a portion of the population's data, which is the usual scenario in most research and exploratory data analysis efforts. In practice, you often don't have access to the entire population, so the sample standard deviation (with *N - 1* in the denominator) is the appropriate choice.

Understanding this distinction helps ensure your analyses reflect the correct level of uncertainty in your data. In most real-world data scenarios, you'll rely on the sample standard deviation to avoid underestimating the variability of the population you are studying.

We've seen that we can use Pandas to calculate standard deviations. We can also use NumPy.

By default, Pandas gives us the **sample** standard deviation.

[7]:
```
students_df['Age'].std()
```

[7]:
```
22.852183567756345
```

Whereas NumPy gives us the **population** standard deviation. Notice that the two are not the same value.

[8]:
```
age_array = students_df['Age'].to_numpy()

age_array.std()
```

[8]:
```
22.73408363727589
```

We can get the population standard deviation by passing in 0 for ddof, and the sample standard deviation by passing in 1 for ddof.

[9]:
```
students_df['Age'].std(ddof=0)
```

[9]:
```
22.73408363727589
```

This is parameter is included in the NumPy std method as well.

[10]:
```
age_array.std(ddof=1)
```

[10]:
```
22.85218356775634
```

1.4 Calculating standard deviation step-by-step

Here we break down the calculation of standard deviation, and show where the difference in sample and population standard deviation comes in.

1.4.1 1. Calculate the difference between each value in the series and the mean

[11]:
```
diffs = students_df['Age'] - students_df['Age'].mean()

diffs
```

[11]:
```
0      1.082474
1      1.082474
2     -1.917526
3     -1.917526
4     -5.917526
         …
95    -3.917526
96    -1.917526
97    -4.917526
98    -1.917526
99    -0.917526
Name: Age, Length: 97, dtype: float64
```

1.4.2 2. Square each of the diffs

[12]:
```
sqaures = diffs ** 2

sqaures
```

[12]:
```
0      1.171750
1      1.171750
2      3.676905
3      3.676905
4     35.017111
         …
95    15.347008
96     3.676905
97    24.182060
98     3.676905
```

```
99     0.841854
Name: Age, Length: 97, dtype: float64
```

1.4.3 3. Sum the squares

[13]:
```
sum_of_squares = sqaures.sum()

sum_of_squares
```

[13]:
```
50133.34020618557
```

1.4.4 4. Divide the squares by the number of elements in the series

This is known as the variance. This is where the ddof (delta degrees of freedom) parameter comes in.

[14]:
```
sample_variance = sum_of_squares / (len(students_df) - 1)
population_variance = sum_of_squares / len(students_df)
```

1.4.5 5. Take the square root of the variance to get the standard deviation

[15]:
```
sample_standard_deviation = sample_variance ** 0.5
population_standard_deviation = population_variance ** 0.5
```

[16]:
```
print(f'Sample standard deviation: {sample_standard_deviation}')
print(f'Population standard deviation: {population_standard_deviation}')
```

```
Sample standard deviation: 22.85218356775634
Population standard deviation: 22.73408363727589
```