SKEWNESS

Skewness is a statistical measure that describes the asymmetry or lack of symmetry in the distribution of data points in a dataset. It indicates whether the data is skewed to the left (negatively skewed), skewed to the right (positively skewed), or if it has a relatively symmetrical distribution.

Here's a brief explanation of skewness types:

Negatively Skewed (Left Skewed): In a negatively skewed distribution, the tail on the left side (lower values) is longer or fatter than the right tail. This means that the majority of the data points are concentrated on the right side of the distribution.

Positively Skewed (Right Skewed): In a positively skewed distribution, the tail on the right side (higher values) is longer or fatter than the left tail. This means that most of the data points are concentrated on the left side of the distribution.

Symmetrical: A symmetrical distribution has a balanced, bell-shaped curve, and it is neither positively nor negatively skewed. In a symmetrical distribution, the mean, median, and mode are approximately equal.

How to Calculate Skewness:

There are several methods to calculate skewness, but one of the most commonly used methods is Pearson's First Coefficient of Skewness formula:

Skewness=3(Mean−Median)Standard DeviationSkewness=Standard Deviation3(Mean−Median)

Here's a step-by-step guide to calculating skewness:

Calculate the mean (average) of the dataset.

Calculate the median (middle value) of the dataset.

Calculate the standard deviation of the dataset.

Plug these values into the formula to calculate skewness.

Python Code:

```
import numpy as np

from scipy.stats import skew

# Sample dataset (replace this with your own data)

data = [10, 20, 25, 30, 35, 40, 45, 50, 60, 70]

# Calculate skewness using NumPy and SciPy
```

mean = np.mean(data)

median = np.median(data)

std_dev = np.std(data)

skewness = skew(data)

print(f"Mean: {mean}")

print(f"Median: {median}")

print(f"Standard Deviation: {std_dev}")

print(f"Skewness: {skewness}")

Pearson Coefficient of Correlation:

```
import numpy as np

# Sample data for X and Y (replace with your own data)
X = [10, 20, 30, 40, 50]
Y = [15, 25, 35, 45, 55]

# Calculate Pearson's correlation coefficient
correlation_coefficient = np.corrcoef(X, Y)[0, 1]

print(f"Pearson's Correlation Coefficient: {correlation_coefficient}")
```

Bowley's coefficient, also known as the Bowley's skewness coefficient, is another measure of skewness used in statistics. It is defined as:

$$B = \frac{Q_1 + Q_3 - 2Q_2}{Q_3 - Q_1}$$

Where:

- Q_1 is the first quartile (25th percentile).
- Q_2 is the second quartile (median).
- Q_3 is the third quartile (75th percentile).

Python Code:

```
import numpy as np
```

```
# Sample dataset (replace this with your own data)
data = [10, 20, 25, 30, 35, 40, 45, 50, 60, 70]

# Calculate the quartiles using numpy.percentile
q1 = np.percentile(data, 25)
q2 = np.percentile(data, 50)
q3 = np.percentile(data, 75)

# Calculate Bowley's coefficient
bowley_coefficient = (q1 + q3 - 2 * q2) / (q3 - q1)

print(f"First Quartile (Q1): {q1}")
print(f"Second Quartile (Q2): {q2}")
print(f"Third Quartile (Q3): {q3}")
print(f"Bowley's Coefficient: {bowley_coefficient}")
```

Bowleys Coeffcient for Group Data:

Bowley's coefficient can also be calculated for grouped data, which means data that is divided into intervals or bins. To calculate Bowley's coefficient for grouped data, you'll need to make some modifications to the formula. Here's the formula for Bowley's coefficient for grouped data:

$$B = \frac{3(M - m)}{Q_3 - Q_1}$$

Where:

- M is the midpoint of the interval containing the median.
- m is the midpoint of the interval preceding the one containing the median.
- Q_1 is the lower quartile, and Q_3 is the upper quartile.

To calculate Bowley's coefficient for grouped data in Python, you'll need the following information:

1. The frequency of each interval (how many data points fall into each interval).
2. The midpoints of the intervals.
3. The lower quartile (Q1) and upper quartile (Q3).

```
# Frequency of each interval
frequencies = [5, 10, 15, 20, 10]

# Midpoints of the intervals
midpoints = [10, 20, 30, 40, 50]

# Lower quartile (Q1) and upper quartile (Q3)
q1 = 25  # Replace with your Q1 value
q3 = 45  # Replace with your Q3 value
```

```
    # Calculate M and m based on the interval containing the
median
    median_interval_index = midpoints.index(q3)  # Replace with
the correct index of the median interval
    M = midpoints[median_interval_index]
    m = midpoints[median_interval_index - 1]

    # Calculate Bowley's coefficient
    bowley_coefficient = (3 * (M - m)) / (q3 - q1)

    print(f"M: {M}")
    print(f"m: {m}")
    print(f"Q1: {q1}")
    print(f"Q3: {q3}")
print(f"Bowley's Coefficient: {bowley_coefficient}")
```

Central moments are statistical measures that provide information about the shape and distribution of data around its mean. The central moments of data are typically calculated using the following formulas:

- The first central moment (mean) is simply the mean of the data.
- The second central moment (variance) measures the spread or dispersion of the data.
- The third central moment measures the asymmetry or skewness of the data distribution.
- The fourth central moment measures the kurtosis, which indicates the tails and the presence of outliers in the data.

You can calculate these central moments in Python using NumPy. Here's a Python code snippet to calculate central moments for a given dataset

Python code:

```
import numpy as np

# Sample dataset (replace this with your own data)
data = [10, 20, 30, 40, 50]

# Calculate the mean (first central moment)
mean = np.mean(data)

# Calculate the variance (second central moment)
variance = np.var(data)

# Calculate the third central moment (skewness)
skewness = np.mean((data - mean) ** 3) / np.power(variance, 3/2)

# Calculate the fourth central moment (kurtosis)
kurtosis = np.mean((data - mean) ** 4) / np.power(variance, 2) - 3

print(f"Mean (First Central Moment): {mean}")
print(f"Variance (Second Central Moment): {variance}")
```

```
print(f"Skewness (Third Central Moment): {skewness}")
print(f"Kurtosis (Fourth Central Moment): {kurtosis}")
----------------------------------------------------
```