

How can I perform univariate analysis in Python with examples?

Authored by
stats writer

July 1, 2024

RECOMMENDED CITATION

stats writer (2024). *How can I perform univariate analysis in Python with examples?*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=165426>

Univariate analysis in Python is a statistical method used to analyze a single variable at a time. It is an essential tool for understanding the characteristics, patterns, and trends of a single variable. This analysis can be performed using various Python libraries such as NumPy, Pandas, and Matplotlib. With these libraries, various statistical measures such as mean, median, mode, and standard deviation can be calculated and visualized through histograms, box plots, and scatter plots. For example, if we have a dataset of student grades, we can use Python to perform univariate analysis to determine the average grade, the most common grade, and the variability of grades among the students. This analysis can provide valuable insights for decision making and further analysis.

Perform Univariate Analysis in Python (With Examples)

The term refers to the analysis of one variable. You can remember this because the prefix "uni" means "one."

There are three common ways to perform univariate analysis on one variable:

- 1. Summary statistics - Measures the center and spread of values.**
- 2. Frequency table - Describes how often different values occur.**
- 3. Charts - Used to visualize the distribution of values.**

This tutorial provides an example of how to perform univariate analysis with the following pandas DataFrame:

```
import pandas as pd
```

```
#create DataFrame
```

```
df = pd.DataFrame({'points': ,  
'assists': ,  
'rebounds': })
```

```
#view first five rows of DataFrame
```

```
df.head()
```

```
points assists rebounds
```

```
0 1.0 5 11
```

```
1 1.0 7 8
```

```
2 2.0 7 10
```

```
3 3.5 9 6
```

```
4 4.0 12 6
```

1. Calculate Summary Statistics

We can use the following syntax to calculate various summary statistics for the 'points' variable in the DataFrame:

```
#calculate mean of 'points'
```

```
df.mean()
```

5.706666666666667

```
#calculate median of 'points'  
df.median()
```

5.0

```
#calculate standard deviation of 'points'  
df.std()
```

3.858287308169384

2. Create Frequency Table

We can use the following syntax to create a frequency table for the 'points' variable:

```
#create frequency table for 'points'  
df.value_counts()
```

4.0 3

1.0 2

5.0 2

2.0 1

3.5 1

6.5 1

7.0 1

7.4 1

8.0 1

13.0 1

14.2 1

Name: points, dtype: int64

This tells us that:

**The value 4 occurs 3 times
The value 1 occurs 2 times
The value 5 occurs 2 times
The value 2 occurs 1 time**

And so on.

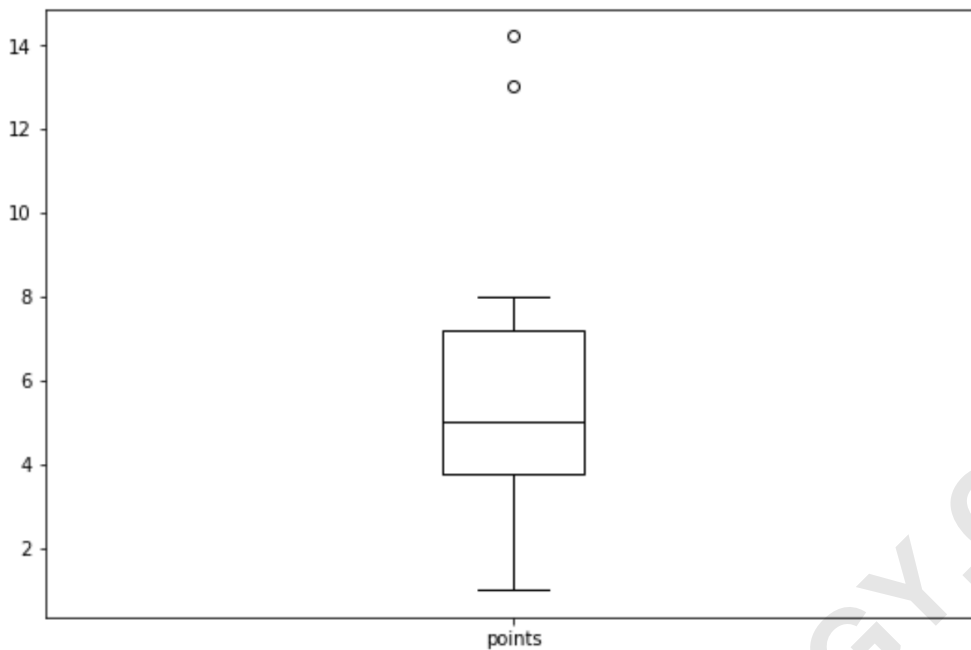
Related:

3. Create Charts

We can use the following syntax to create a for the 'points' variable:

```
import matplotlib.pyplot as plt
```

```
df.boxplot(column=, grid=False, color='black')
```

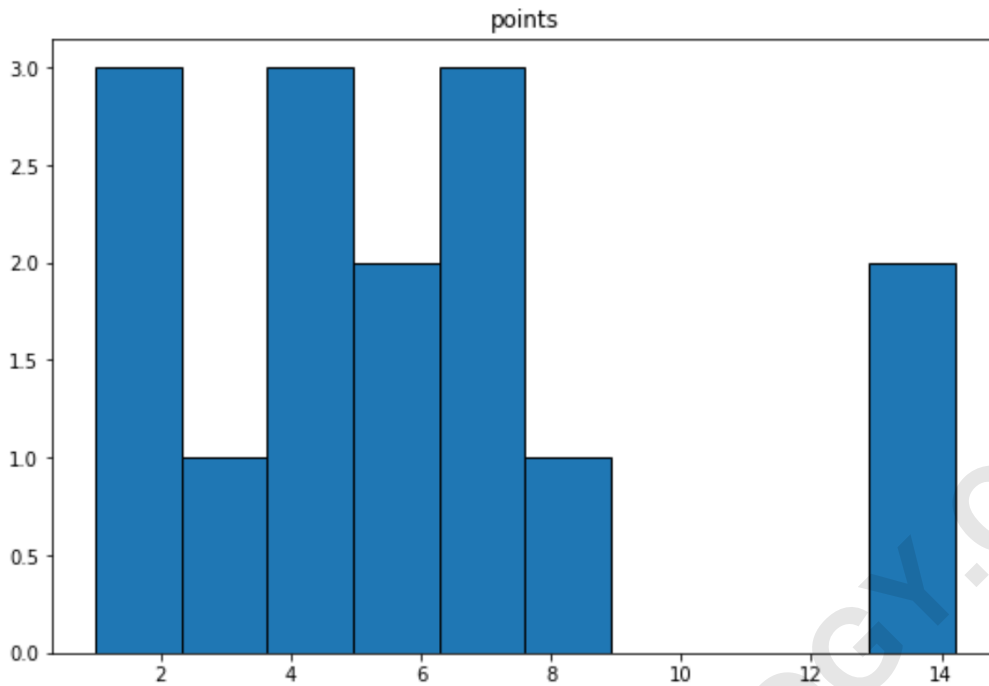


Related:

We can use the following syntax to create a histogram for the 'points' variable:

```
import matplotlib.pyplot as plt
```

```
df.hist(column='points', grid=False, edgecolor='black')
```

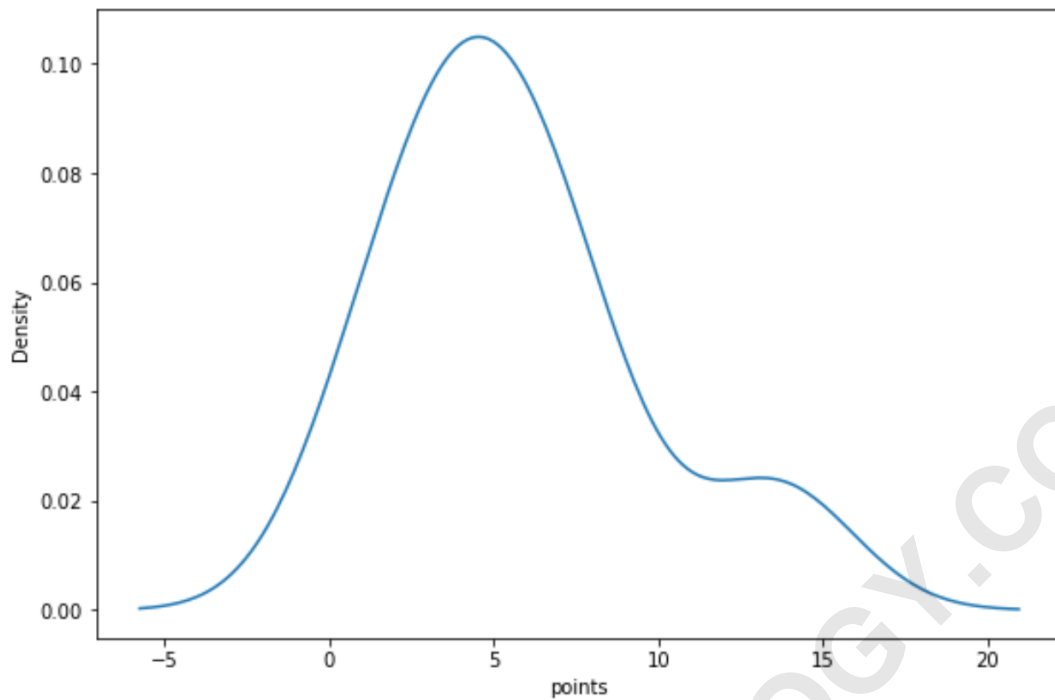


Related:

We can use the following syntax to create a for the 'points' variable:

```
import seaborn as sns
```

```
sns.kdeplot(df)
```



Related:

Each of these charts give us a unique way to visualize the distribution of values for the 'points' variable.