5.1 NumPy Statistical Functions

Statistics involves gathering data, analyzing it, and drawing conclusions based on the information collected.
NumPy provides us with various statistical functions that can perform statistical data analysis.

Common NumPy Statistical Functions

Here are some of the statistical functions provided by NumPy:

Functions	Descriptions
`median()`	return the median of an array
`mean()`	return the mean of an array
`std()`	return the standard deviation of an array
`percentile()`	return the nth percentile of elements in an array
`min()`	return the minimum element of an array
`max()`	return the maximum element of an array

Find Median Using NumPy

The median value of a numpy array is the middle value in a sorted array.
In other words, it is the value that separates the higher half from the lower half of the data.

* It is important to note that if the number of elements is
Odd, the median is the middle element.
Even, the median is the average of the two middle elements.
Now, we will learn how to calculate the median using NumPy for arrays with odd and even number of elements.

Example 1: Compute Median for Odd Number of Elements

```
import numpy as np

array1 = np.array([1, 2, 3, 4, 5])

median = np.median(array1)

print(median)
```

```
muthanna@maxo79:~/Desktop$ python3 numpy1.py
3.0
```

Example 2: Compute Median for Even Number of Elements

```
import numpy as np

array1 = np.array([1, 2, 3, 4, 5, 6])

median = np.median(array1)

print(median)
```

```
muthanna@maxo79:~/Desktop$ python3 numpy1.py
3.5
```

Median of NumPy 2D Array

Calculation of the median is not just limited to 1D array. We can also calculate the median of the 2D array.

In a 2D array, median can be calculated either along the horizontal or the vertical axis individually, or across the entire array.

When computing the median of a 2D array, we use the axis parameter inside np.median() to specify the axis along which to compute the median.

If we specify,

axis = 0, median is calculated along vertical axis

axis = 1, median is calculated along horizontal axis

If we don't use the axis parameter, the median is computed over the entire array.

Example 3: Compute the median of a 2D array

```
import numpy as np

array1 = np.array([[2, 4, 6],
                   [8, 10, 12],
                   [14, 16, 18]])

result1 = np.median(array1, axis=1)

print("Median along horizontal axis :", result1)

result2 = np.median(array1, axis=0)

print("Median along vertical axis:", result2)

result3 = np.median(array1)

print("Median of entire array:", result3)
```

```
muthanna@maxo79:~/Desktop$ python3 numpy1.py
Median along horizontal axis : [ 4. 10. 16.]
Median along vertical axis: [ 8. 10. 12.]
Median of entire array: 10.0
```

Compute Mean Using NumPy

The mean value of a NumPy array is the average value of all the elements in the array.

It is calculated by adding all elements in the array and then dividing the result by the total number of elements in the array.

We use the np.mean() function to calculate the mean value. For example,

```
import numpy as np

array1 = np.array([[4, 8, 6],
                   [6, 4, 2],
                   [14, 18, 16]])

result1 = np.mean(array1, axis=1)

print("Mean along horizontal axis :", result1)

result2 = np.mean(array1, axis=0)

print("Mean along vertical axis:", result2)

result3 = np.mean(array1)

print("Mean of entire array:", result3)
```

```
muthanna@maxo79:~/Desktop$ python3 numpy1.py
Mean along horizontal axis : [ 6.  4. 16.]
Mean along vertical axis: [ 8. 10.  8.]
Mean of entire array: 8.666666666666666
```

Standard Deviation of NumPy 2D Array

In a 2D array, standard deviation can be calculated either along the horizontal or the vertical axis individually, or across the entire array.

Similar to mean and median, when computing the standard deviation of a 2D array, we use the axis parameter inside np.std() to specify the axis along which to compute the standard deviation.

```
import numpy as np

array1 = np.array([[4, 8, 6],
                   [6, 4, 2],
                   [14, 18, 16]])

result1 = np.std(array1, axis=1)

print("Standard deviation along horizontal axis :", result1)

result2 = np.std(array1, axis=0)

print("Standard deviation along vertical axis:", result2)

result3 = np.std(array1)

print("Standard deviation of entire array:", result3)
```

```
muthanna@maxo79:~/Desktop$ python3 numpy1.py
Standard deviation along horizontal axis : [1.63299316 1.63299316 1.63299316]
Standard deviation along vertical axis: [4.3204938  5.88784058 5.88784058]
Standard deviation of entire array: 5.497474167490214
```

Compute Percentile of NumPy Array

In NumPy, we use the percentile() function to compute the nth percentile of a given array.

Let's see an example.

```
import numpy as np

array1 = np.array([[7, 10, 6],
                   [9, 3, 8],
                   [13, 17, 16]])

result1 = np.percentile(array1, 25)
print("25th percentile:",result1)

result2 = np.percentile(array1, 75)
print("75th percentile:",result2)
```

```
muthanna@maxo79:~/Desktop$ python3 numpy1.py
25th percentile: 7.0
75th percentile: 13.0
```

Find Minimum and Maximum Value of NumPy Array

We use the min() and max() function in NumPy to find the minimum and maximum values in a given array.

Let's see an example.

```
import numpy as np

array1 = np.array([[7, 10, 6],
                   [9, 3, 8],
                   [13, 17, 16]])

min_val = np.min(array1)

max_val = np.max(array1)

print("Minimum value:", min_val)
print("Maximum value:", max_val)
```

```
muthanna@maxo79:~/Desktop$ python3 numpy1.py
Minimum value: 3
Maximum value: 17
```

5.2 NumPy Histogram

NumPy histograms is a graphical representation of the distribution of numerical data. Using functions like histogram() and plt(), we can create and plot histograms.

We'll take a closer look at histograms and how they can be created and plotted in NumPy.

NumPy has a built-in function histogram() that takes an array of data as a parameter.
In histogram, a bin is a range of values that represents a group of data (class interval). bin is an optional parameter.
Let's see an example.

```
import numpy as np

data = np.array([5, 10, 15, 18, 20])

bin = [0,10,20,30]

graph = np.histogram(data, bin)

print(graph)
```

```
muthanna@maxo79:~/Desktop$ python3 histogram.py
(array([1, 3, 1]), array([ 0, 10, 20, 30]))
```

In this example, we have used the histogram() function to calculate the frequency distribution of data. We have passed two parameters: data and bin.

The histogram() function returns a tuple containing two arrays:

- the first array contains the frequency counts of the data within each bin, and
- the second array contains the bin edges.

From the resulting output, we can see that:

- Only 1 data point (i.e., 5) from the array data lies between the bin edges 0 and 10
- 3 data points (i.e., 10, 15, 18) lie between 10 and 20, and
- 1 data point (i.e., 20) lies between 20 and 30.

Plot the Histogram

We can use the plt() function to plot the numerical value returned by the histogram.

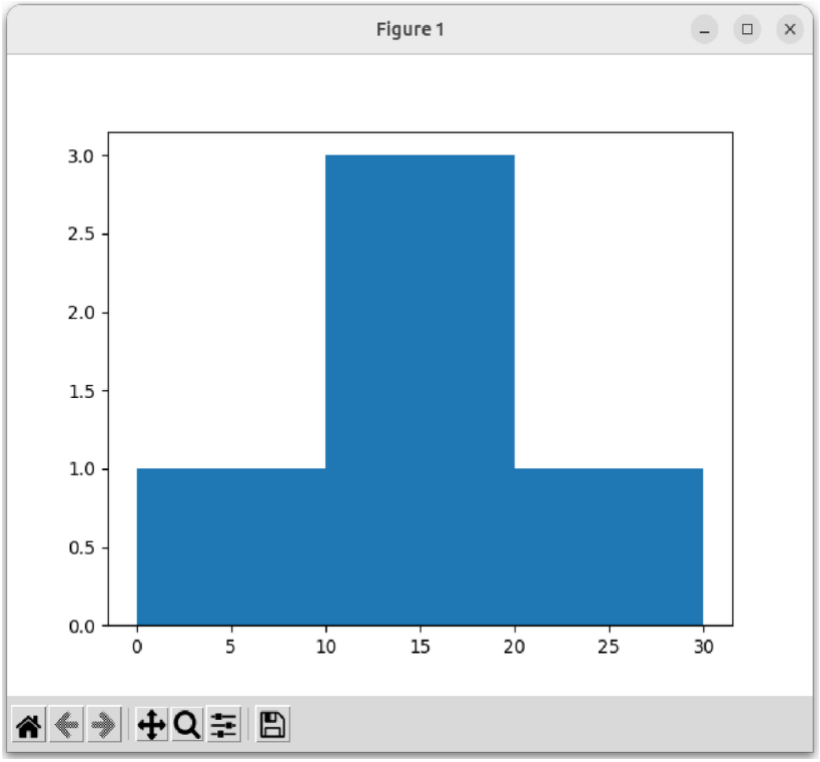
The plt() is a function provided by Matplotlib. To use plt(), we need to import the Matplotlib.

```
from matplotlib import pyplot as plt
```

Let's see an example.

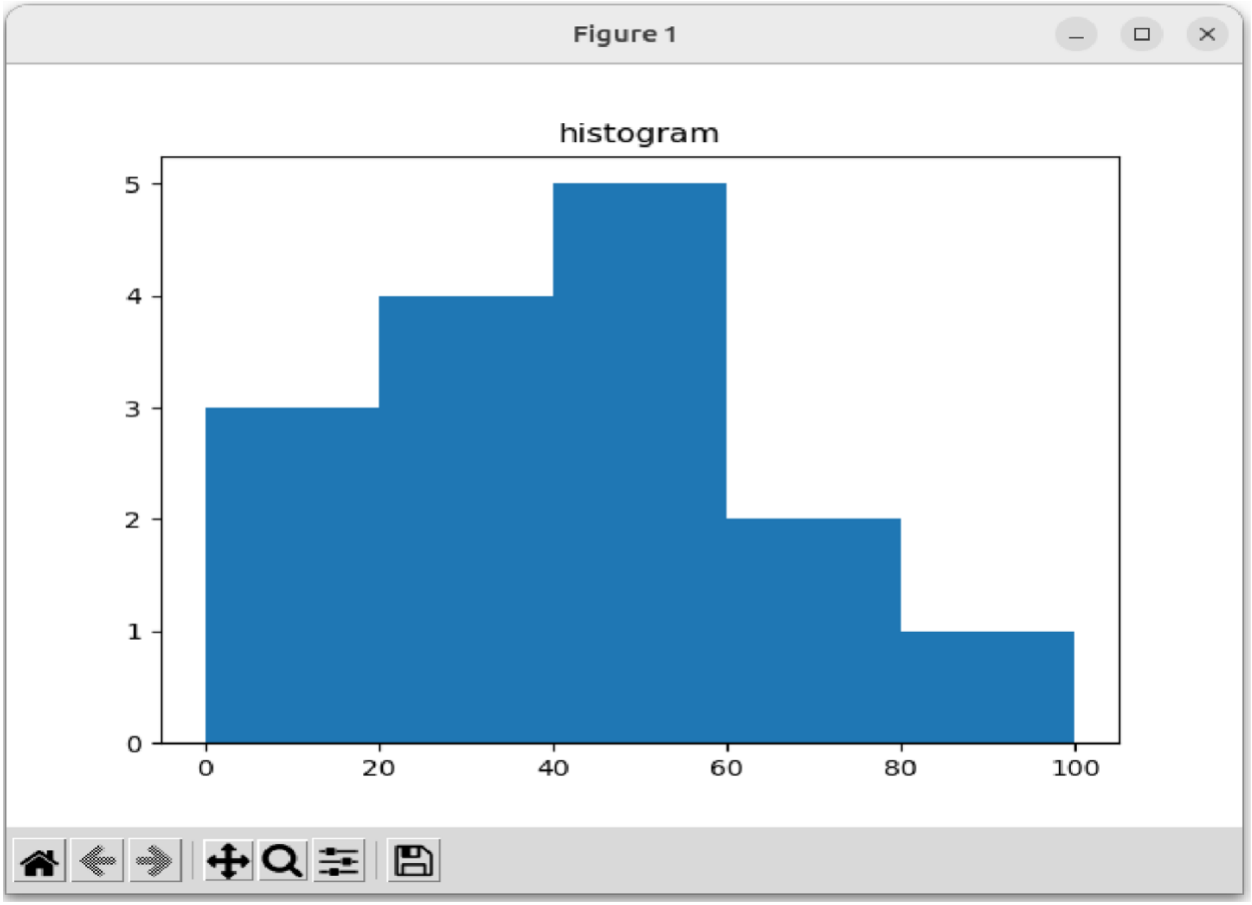
```
import numpy as np
from matplotlib import pyplot as plt

data = np.array([5, 10, 15, 18, 20])

bin = [0,10,20,30]

graph = np.histogram(data, bin)

print(graph)

plt.hist(data, bin)

plt.show()
```

```
muthanna@maxo79:~/Desktop$ python3 histogram.py
(array([1, 3, 1]), array([ 0, 10, 20, 30]))
```



```python
import numpy as np
from matplotlib import pyplot as plt

data = np.array([22,87,5,43,56,73,55,54,11,20,51,5,79,31,27])

bin = [0,20,40,60,80,100]

graph = np.histogram(data, bin)

print(graph)

plt.hist(data, bin)
plt.title("histogram")

plt.show()
```

```
muthanna@maxo79:~/Desktop$ python3 histogram.py
(array([3, 4, 5, 2, 1]), array([  0,  20,  40,  60,  80, 100]))
```



```
import numpy as np
from matplotlib import pyplot as plt

data = np.array([22,87,5,43,56,73,55,54,11,20,51,5,79,31,27])

bin = [0,20,40,60,80,100]

graph = np.histogram(data, bin)

print(graph)

plt.hist(data, bin,width=10,color='red')
plt.title("histogram")
plt.grid(axis='y', alpha=1)
plt.grid(axis='x', alpha=1)
plt.show()
```