numpy-5

November 16, 2025

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

[8]:
```
# Covariance, Correlation

x = np.array([1, 2, 3, 4, 5])
y = np.array([2, 4, 6, 8, 10])

cov = np.cov(x, y, bias=True)
print(cov)
print(cov[0][1])
```

```
[[2. 4.]
 [4. 8.]]
4.0
```

[10]:
```
x = np.array([1, 2, 3, 4, 5])
y = np.array([10, 8, 8, 4, 10])

cov = np.cov(x, y, bias=True)
print(cov)
print(cov[0][1])
```

```
[[ 2.  -0.8]
 [-0.8  4.8]]
-0.8
```

[11]:
```
x = np.array([1, 2, 3, 4, 5])
y = np.array([2, 4, 6, 8, 10])

corr = np.corrcoef(x, y)
print(corr)
print(corr[0][1])
```

```
[[1. 1.]
 [1. 1.]]
0.9999999999999999
```

[12]:
```
x = np.array([1, 2, 3, 4, 5])
y = np.array([2, 4, 6, 5, 1])
```

```
corr = np.corrcoef(x, y)
print(corr)
print(corr[0][1])
```

```
[[ 1.         -0.07624929]
 [-0.07624929  1.        ]]
-0.07624928516630235
```

[71]:
```
# Q. Calculate correlation coficient of two random samples

X = np.random.randint(1, 100, 100)
Y = np.random.randint(1, 100, 100)

corr = np.corrcoef(X, Y)[0][1]
print(f"Correlation between X and Y: {corr}")
```

```
Correlation between X and Y: 0.22571743193231517
```

[76]:
```
# Q. Generate a random sample, calculate IQR and detect outliers

data = np.random.randint(1, 100, 20)

q1 = np.percentile(data, 25)
q3 = np.percentile(data, 75)

print(data)
print(q1)
print(q3)

iqr = q3 - q1
print(iqr)

lower_bound = q1 - 1.5*iqr
upper_bound = q3 + 1.5*iqr

print(lower_bound)
print(upper_bound)

outlier_cond = (data < lower_bound) | (data > upper_bound)
print(outlier_cond)

outliers = data[outlier_cond]
print(outliers)
```

```
[ 5 50 79 87 50 94  6 92 57 99  2 56 53 66 27 83 17 63 39 40]
36.0
80.0
44.0
```

```
-30.0
146.0
[False False False False False False False False False False False False
 False False False False False False False False]
[]
```

[79]:
```
# Q. Generate a random sample, calculate IQR and detect outliers

data = np.array([-100, 2, 1, 3, 5, 6, 70, 8, 9, 22, 22, 10])

q1 = np.percentile(data, 25)
q3 = np.percentile(data, 75)

print(data)
print(q1)
print(q3)

iqr = q3 - q1
print(iqr)

lower_bound = q1 - 1.5*iqr
upper_bound = q3 + 1.5*iqr

print(lower_bound)
print(upper_bound)

outlier_cond = (data < lower_bound) | (data > upper_bound)
print(outlier_cond)

outliers = data[outlier_cond]
print(outliers)
```

```
[-100    2    1    3    5    6   70    8    9   22   22   10]
2.75
13.0
10.25
-12.625
28.375
[ True False False False False False  True False False False False False]
[-100   70]
```

[81]:
```
print(np.mean(data))
print(np.median(data))
```

```
4.833333333333333
7.0
```

[ ]:
```

```