numpy-4

November 16, 2025

0.1 Stats

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

[3]:
```
# Measure of central tendency

# mean -> if there is outlier we dont use mean, normally distributed data
# median -> ideal for skewed data or when outliers are present
# mode -> categorical data to identify the most common category

data = np.array([1, 2, 1, 4, 5, 6, 5, 5, 9])

mean = np.mean(data)
median = np.median(data)
mode = np.bincount(data).argmax()

print(mean)
print(median)
print(mode)
```

```
4.222222222222222
5.0
5
```

[12]:
```
mu, sigma = 0, 0.1
```

[13]:
```
rng = np.random.default_rng()
```

[14]:
```
s = rng.normal(mu, sigma, 1000)
```

[15]:
```
s.mean()
```

[15]:
```
np.float64(0.0015727221068597534)
```

[17]:
```
np.median(s)
```

[17]:
```
np.float64(0.0007219989106038398)
```

[18]:
```
np.mean(s)
```

[18]:
```
np.float64(0.0015727221068597534)
```

[19]:
```
data = np.array([1, 2, 3, 4, 5, 6])

mean = np.mean(data)
median = np.median(data)
mode = np.bincount(data).argmax()

print(mean)
print(median)
print(mode)
```

```
3.5
3.5
1
```

[20]:
```
# Measure of Dispersion

# standard deviation
# Variance
# Range value

np.std(data)
```

[20]:
```
np.float64(1.707825127659933)
```

[21]:
```
np.var(data)
```

[21]:
```
np.float64(2.9166666666666665)
```

[22]:
```
pow(np.std(data), 2)
```

[22]:
```
np.float64(2.9166666666666665)
```

[23]:
```
np.ptp(data)
```

[23]:
```
np.int64(5)
```

[24]:
```
# Quartile, percentile

q1 = np.percentile(data, 25)
q2 = np.percentile(data, 50) # also called median
q3 = np.percentile(data, 75)

print(q1, q2, q3)
```

```
2.25 3.5 4.75
```

[25]:
```
iqr = q3 - q1
print(iqr)
```

```
2.5
```

0.2 Probability

[28]:
```
# P = number of occurance of fav event / total no of events or sample space
# P(A) = Number of favorable outcomes / Total number of outcomes

# Probability of uniform event

# P(A="even number")=? fair 6-sided die

sample_space = np.array([1, 2, 3, 4, 5, 6])
event = (sample_space % 2 == 0)

print(sample_space)
print(event)

print(np.sum(event))

probability = np.sum(event) / len(sample_space)

print(f"P(A='Even Number') = {probability}")
```

```
[1 2 3 4 5 6]
[False  True False  True False  True]
3
P(A='Even Number') = 0.5
```

[64]:
```
# Empirical probability / Experimental

# Find probability of rolling 6 from 1000 experiments

experments = np.random.randint(1, 7, 1000)
# experments
event = np.sum(experments == 6)
event

p = event / len(experments)
print(f"P(rolling 6) = {p}")
```

```
P(rolling 6) = 0.155
```

[70]:
```
# Conditional probability
# Find the probability of even number that are greater then 5 form the
# sample space of 1000 having numbers in range 1 to 10
```

```
data = np.random.randint(1, 11, 1000)
A = data % 2 == 0
B = data > 5
# print(A)
# print(B)

P_A_given_B = np.sum(A & B) / np.sum(B)

print(f"P(Even | >5) = {P_A_given_B}")
```

```
P(Even | >5) = 0.611764705882353
```

[ ]: