# SCIPY-MODULE

## Data Science statistics, we mean the foundational concepts and techniques from statistics that help you:

✅**Understand data** (descriptive statistics)
✅**Draw conclusions** (inferential statistics)
✅**Test hypotheses**
✅**Build predictive models**

Here's a **quick overview** of the most important statistics topics for data science:

---

### ? 1. Descriptive Statistics

Summarize and describe data.

- **Mean**: Average of values.
- **Median**: Middle value of sorted data.
- **Mode**: Most frequent value.
- **Variance**: Spread around the mean.
- **Standard Deviation**: Square root of variance.
- **Range, Quartiles, IQR** (Interquartile Range).
- **Skewness & Kurtosis**: Shape of the distribution.

## `scipy.stats` — Statistical Functions

The `scipy.stats` module contains many **probability distributions**, **hypothesis tests**, and **descriptive statistics**.

```
from scipy import stats
import numpy as np

# Sample data
data = np.array([2, 3, 5, 7, 11, 13, 17])

# Descriptive statistics
mean = np.mean(data)
std_dev = np.std(data)
skewness = stats.skew(data)
kurtosis = stats.kurtosis(data)
```

```
print(f"Mean: {mean}, Std: {std_dev}, Skewness: {skewness}, Kurtosis: {kurtosis}")
```

# Mean

**Definition:** Average value

**Example in Python:**

```
import numpy as np
data = [2, 3, 5, 7, 11]
mean = np.mean(data)
print(mean)  # Output: 5.6
```

# Median

**Definition:** Middle value when the data is sorted.
**Formula:**

- If nnn is odd: middle value
- If nnn is even: average of the two middle values

**Example in Python:**

```
median = np.median(data)
print(median)  # Output: 5
```

# Mode

**Definition:** Value that appears most frequently.

**Example in Python:**

```
from scipy import stats
mode_result = stats.mode(data)
print(mode_result.mode[0])  # most frequent value
print(mode_result.count[0])  # frequency
```

# Variance

**Definition:** Average squared deviation from the mean.

**Example in Python:**

```
variance = np.var(data, ddof=1)  # ddof=1 for sample variance
print(variance)
```

**#ddof = 1**:
Stands for *"delta degrees of freedom"*, set to 1.
This tells the function to divide by N−1, where N is the number of elements.

# Range, Quartiles, IQR

**Range** = max − min
**Quartiles** = 25%, 50%, 75% percentiles
**IQR** = Q3 − Q1

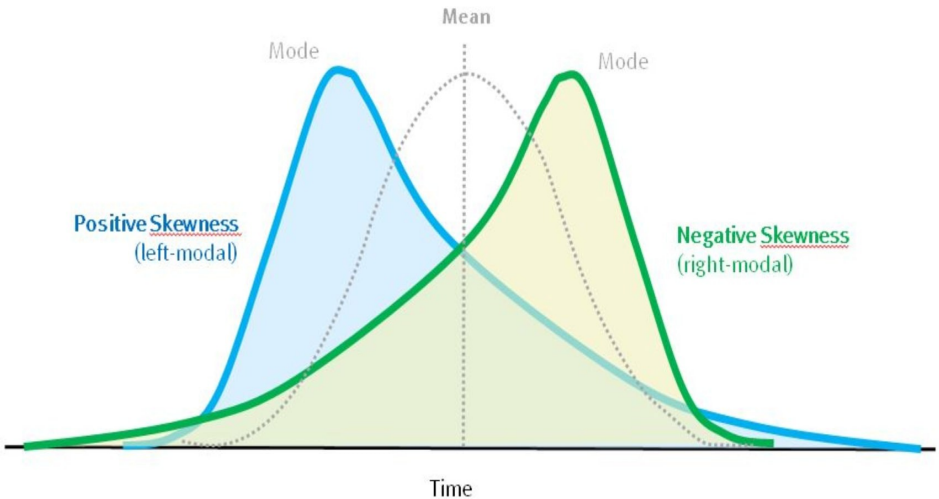
**Example in Python:**

```
range_ = np.ptp(data)
q1 = np.percentile(data, 25)
q3 = np.percentile(data, 75)
iqr = q3 - q1
print(range_, q1, q3, iqr)
```

# Skewness

**Definition:** Measure of asymmetry of the distribution.

- Skewness < 0 → left-skewed
- Skewness = 0 → symmetric
- Skewness > 0 → right-skewed



```
skewness = stats.skew(data)
print(skewness)
```

```
from scipy import stats
import numpy as np

data = [2, 3, 5, 7, 11, 13, 17]
skewness = stats.skew(data)

print(f"Skewness: {skewness:.3f}")
```

# Kurtosis

**Definition:** Measure of "tailedness" of the distribution.

- Positive kurtosis → heavy tails
- Negative kurtosis → light tails

Example in Python:

```
kurt = stats.kurtosis(data)
print(kurt)
```

```
kurtosis = stats.kurtosis(data)
print(f"Kurtosis (excess): {kurtosis:.3f}")
```

# Range

- **Definition**: The difference between the **maximum** and **minimum** value of the data.

Range=max(data)−min(data)

---

**Example**:

```
import numpy as np

data = [2, 3, 5, 7, 11, 13, 17]
range_ = np.ptp(data)  # ptp = peak-to-peak
print(range_)  # Output: 15
```

# Quartiles

Quartiles split your sorted data into **4 equally sized parts**.

- **Q1 (25th percentile)**: The value below which 25% of data lies
- **Q2 (50th percentile)**: The **median** — 50% of data lies below
- **Q3 (75th percentile)**: The value below which 75% of data lies

---

```
data = [2, 3, 5, 7, 11, 13, 17]

q1 = np.percentile(data, 25)
q2 = np.percentile(data, 50)
q3 = np.percentile(data, 75)

print(q1, q2, q3)  # Output: 3.5 7.0 13.0
```

# Interquartile Range (IQR)

- **Definition**: Difference between Q3 and Q1

$$IQR = Q3 - Q1$$

✅ IQR is **robust** against extreme outliers and a better measure of data **spread** than range.

---

### ? Example:

$$IQR = 13 - 3.5 = 9.5$$

```
iqr = q3 - q1
print(iqr)  # Output: 9.5
```

## scipy.optimize

### 1) Minimize a function

```
from scipy.optimize import minimize

# Define f(x) = (x-3)^2
f = lambda x: (x-3)**2

# Minimize starting at x0 = 0
result = minimize(f, x0=0)
```

```
print(result.x)    # → close to 3
print(result.fun)  # → value at the optimum (~0)
```

## Scipy.linalg

```
import numpy as np
from scipy import stats, optimize, linalg, integrate

# 3. scipy.linalg - solve a linear system
A = np.array([[3,2],[1,2]])
b = np.array([5,5])
x = linalg.solve(A, b)
print(f"Solution to Ax=b is {x}")
```

## scipy.integrate - definite integration

```
g = lambda x: x**2
area, err = integrate.quad(g, 0, 3)
print(f"Integral of x^2 from 0 to 3 is {area}, Error = {err}")
```