Python

Theoretical Concepts

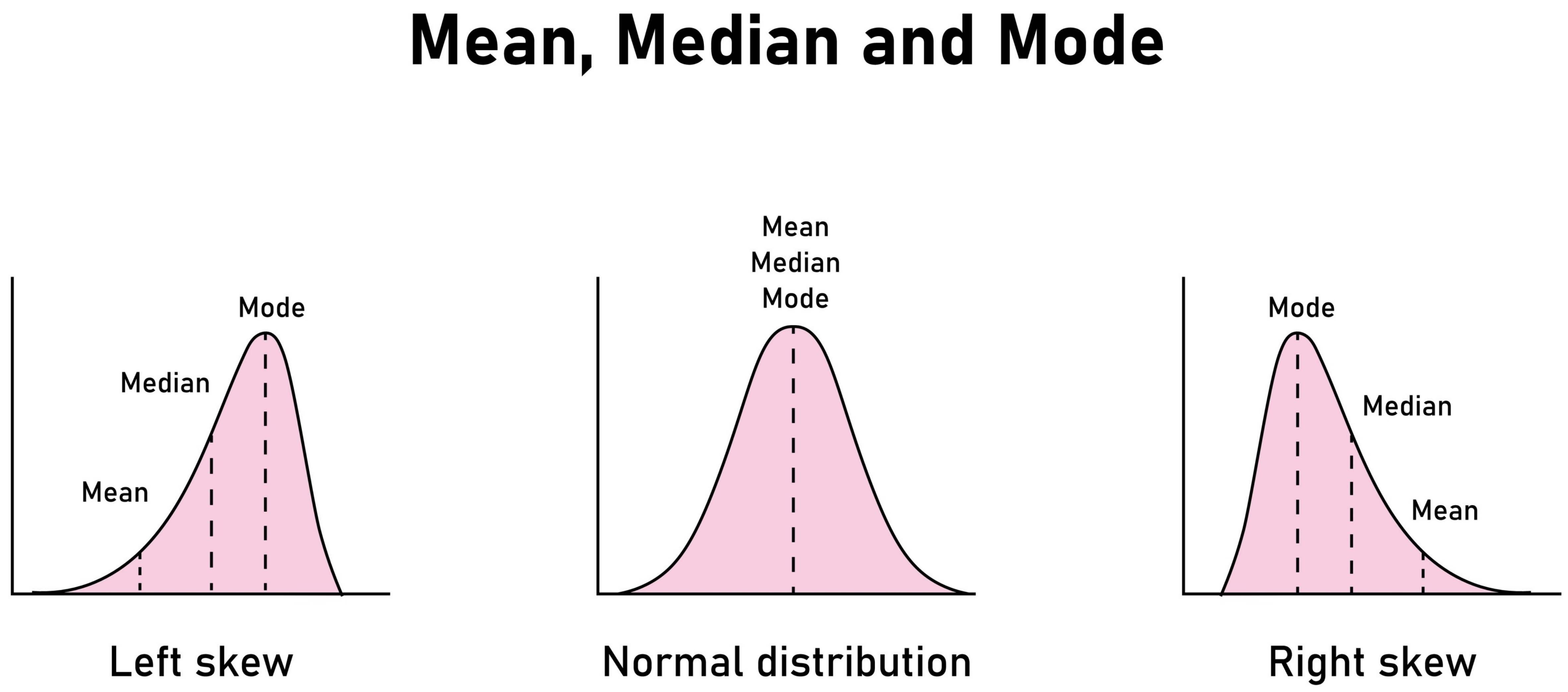
Alish Bista - 11/11/2025

Measures of Center & Spread

Measures of Center

Where Is The Middle?

- **Mean**
  - Arithmetic average (sum / count)
  - Best for symmetrical data
- **Median**
  - Exact middle value when sorted (50th percentile)
  - Used for skewed data or outliers
- **Mode**
  - Most frequently occurring data
  - Used for categorical data



Skewness

Impact of Skewness

- Outliers drag the mean towards the tail
  - **Right Skew**: Mean > Median (e.g., income, most people earn moderate amounts, but a few millionaires pull the mean upward)
  - **Left Skew**: Mean < Median (e.g., easy exam scores, most students score high, but a few low scores pull the mean downward)
  - Use the Median when data is skewed to avoid misleading conclusions

Calculating Measures of Center

Measures of Center in Python

```
import seaborn as sns
import numpy as np

df = sns.load_dataset('tips')
bill_data = df['total_bill']

mean_val = np.mean(bill_data)
median_val = np.median(bill_data)

print(f"Mean Bill: {mean_val:.2f}")
print(f"Median Bill: {median_val:.2f}")
```

Standard Deviation

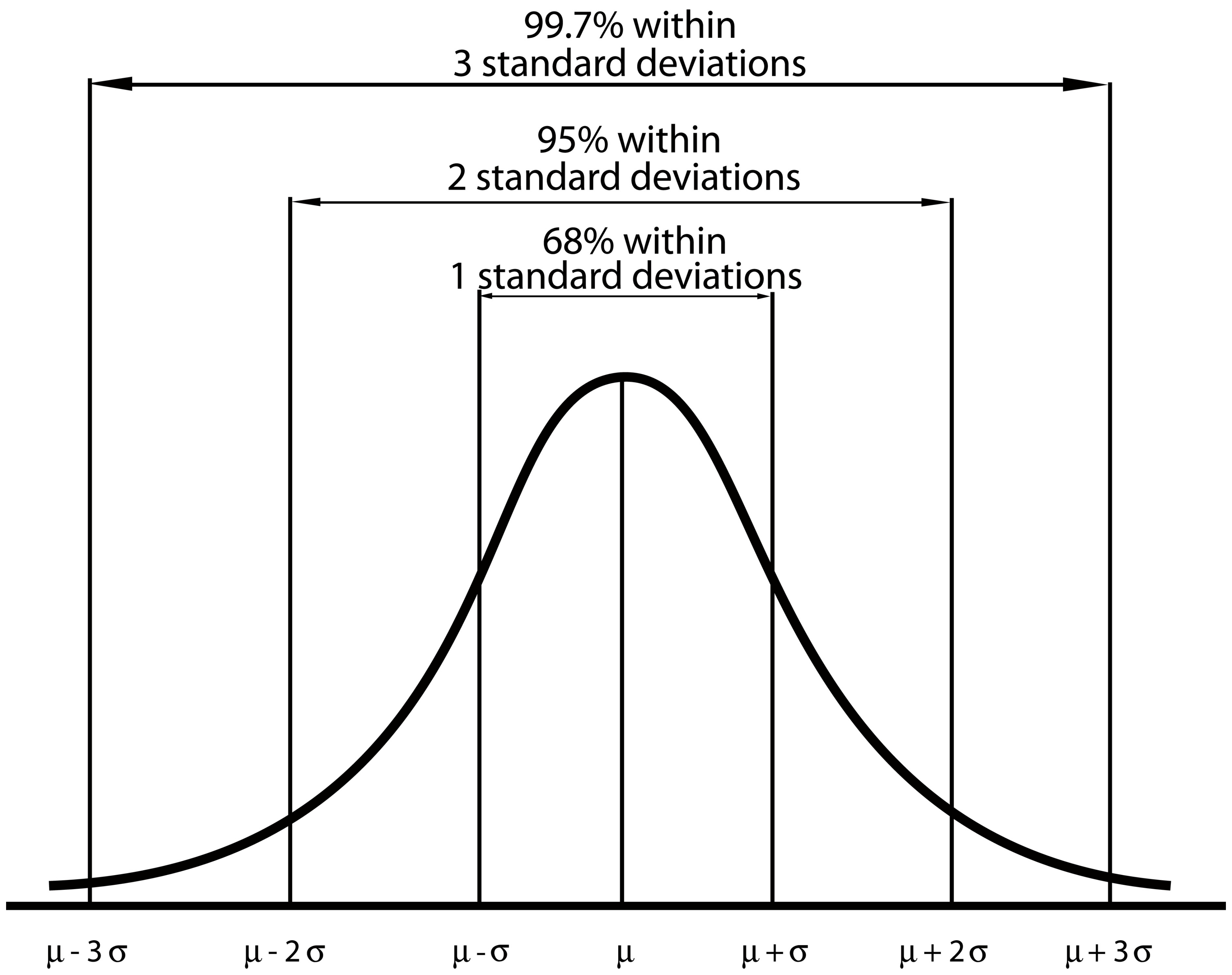
Measures of Spread

- SD (σ) is the average distance of data points from the Mean.
    - **Small SD**: Low variability, high consistency. Low risk
    - **Large SD**: High variability, scattered data. High risk
- SD measures the volatility or risk of a metric

Normal Distribution

The Bell Curve

- Many natural processes follow this distribution (defined by μ and σ).
- Empirical Rule (68-95-99.7):
  - 68% of data falls within ±1 SD of the Mean
  - 95% of data falls within ±2 SDs of the Mean
  - 99.7% of data falls within ±3 SDs of the Mean



Calculating Measures of Spread

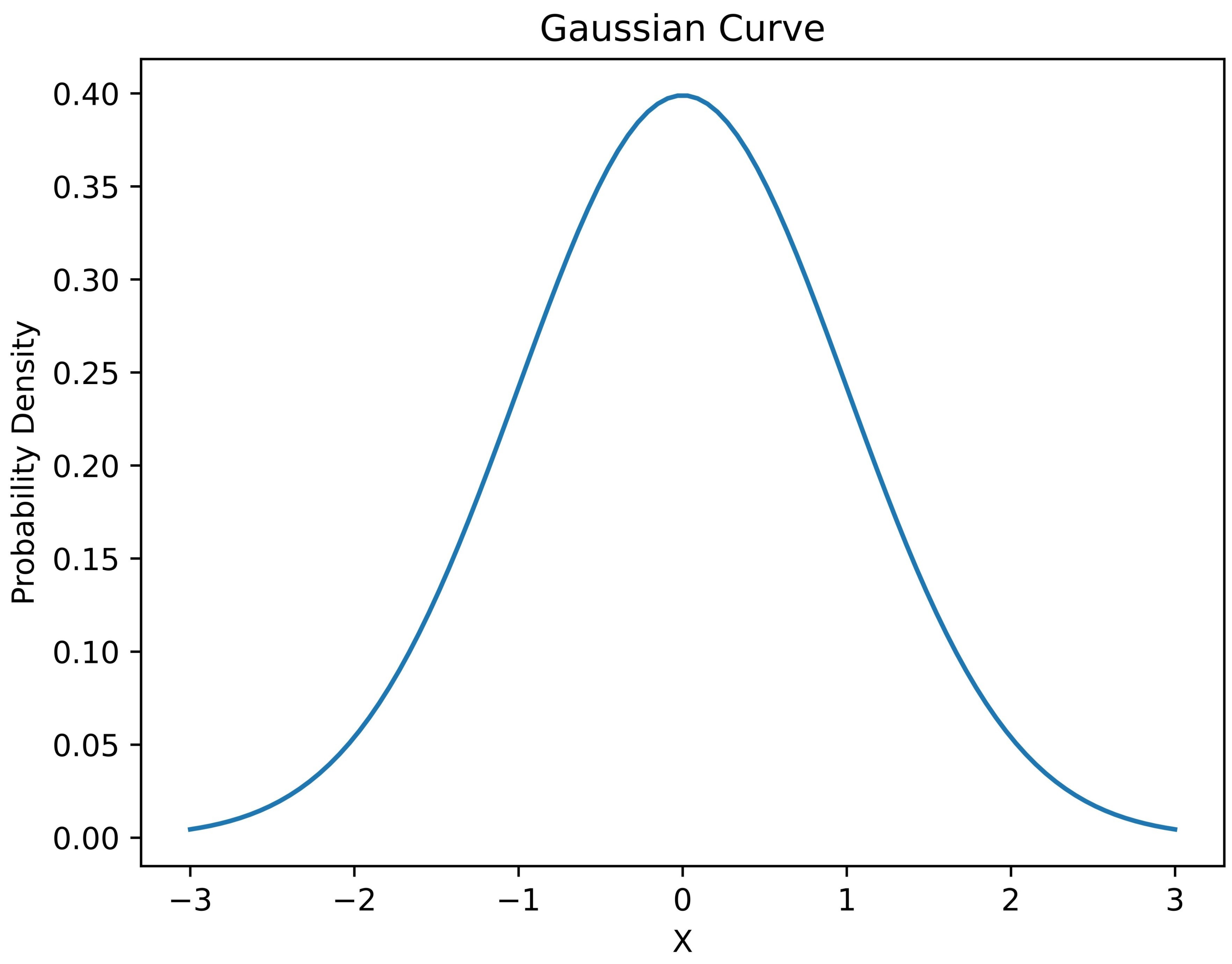
Standard Deviation in Python

```
import numpy as np
sales_a = [100, 105, 95, 100, 100] # Low SD
sales_b = [50, 150, 100, 20, 180]  # High SD


print(f"Mean A: {np.mean(sales_a)}, SD A:
{np.std(sales_a):.2f}")


print(f"Mean B: {np.mean(sales_b)}, SD B:
{np.std(sales_b):.2f}")
```

Density Plots

Visualizing Distribution

- Provides a smoothed, continuous curve of the distribution
- Better than histograms for seeing the overall shape
- Ideal for comparing continuous distributions (e.g., age groups)



Bar And Box

Visualizing Group Differences

- **Bar Plots**: Compare a measure (e.g., Mean Sales) across distinct categories (e.g., Regions)
- **Box Plots**: Display the full distribution (spread, median, and outliers) for comparison

Creating a Density Plot

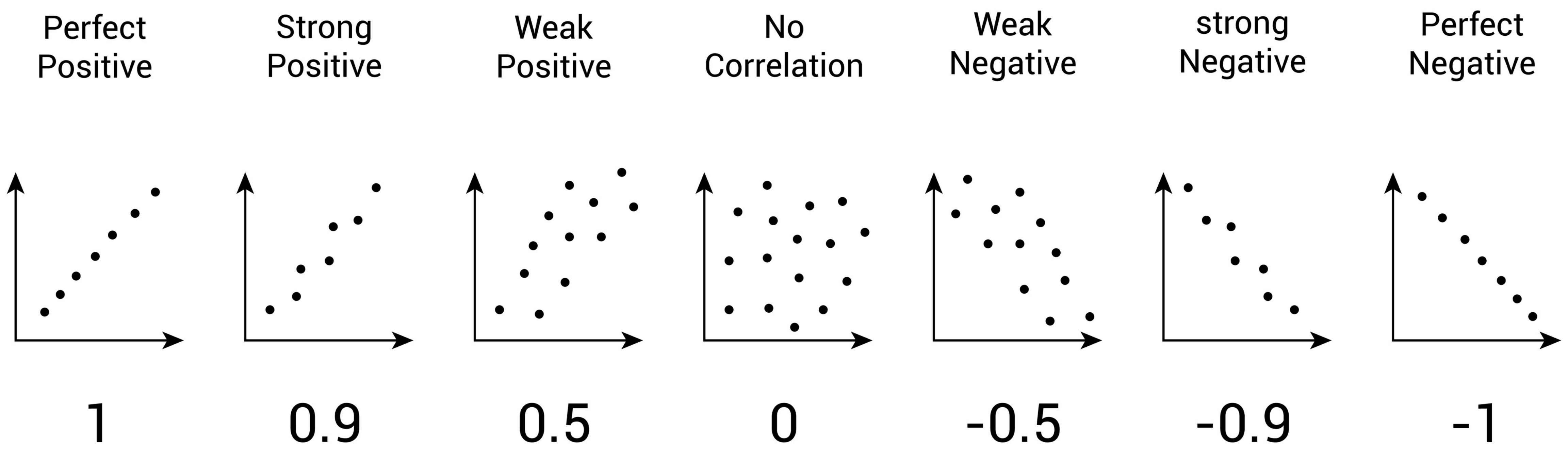
Plotting Density Plot

```
import seaborn as sns
import matplotlib.pyplot as plt

df = sns.load_dataset('titanic').dropna(subset=['age'])

sns.kdeplot(df['age'], fill=True)
plt.title('Age Distribution of Titanic Passengers')
plt.show()
```

Correlation

Relationship Analysis

- The strength and direction of a linear relationship (r ∈ [−1, +1])
  - r = +0.8 to +1.0: Strong positive (as X increases, Y increases strongly)
  - r = +0.3 to +0.7: Weak to moderate positive
  - r ≈ 0: No linear relationship
  - r = −0.3 to −0.7: Weak to moderate negative
  - r = −0.8 to −1.0: Strong negative (as X increases, Y decreases strongly)
- **Correlation does NOT imply causation.**



Displaying Correlation

Plotting Heatmap Plot

```
import seaborn as sns
import matplotlib.pyplot as plt

# Load data
df = sns.load_dataset('iris')

# Calculate correlation between two variables
correlation = df['petal_length'].corr(df['petal_width'])
print(f"Correlation (r): {correlation:.3f}")

# Create correlation heatmap for all numeric variables
plt.figure(figsize=(8, 6))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', center=0)
plt.title('Correlation Heatmap')
plt.show()
```

Regression

Predicting Outcomes

- Linear Regression finds the Line of Best Fit to predict a dependent variable (Y)
- **Model**: Y = mX + c
  - **m** (Slope): The predicted change in Y for a 1-unit change in X
  - **c** (Intercept): The value of Y when X = 0

Slope

Meaning of m

- The slope is the expected return for increasing your input (X)
- **Example**: Slope m = 3.5 means $1 spent will equal $3.50 gained in Sales
- Allows data-backed resource allocation decisions

Calculating Regression

Simple Regression Calculation

```
from scipy import stats

import numpy as np

ad_spend = np.array([10, 20, 30, 40, 50])

sales = np.array([100, 120, 150, 160, 190])

slope, intercept, r_value, p_value, std_err =
stats.linregress(ad_spend, sales)

print(f"Predicted Line: Y = {slope:.2f}X +
{intercept:.2f}")
```

Sample vs Population

Sample of a Population

- **Population**: The entire group you want to understand (e.g., all 50,000 customers)
- **Sample**: A smaller subset you actually measure (e.g., 500 surveyed customers)
- It's usually too expensive to measure the entire population so we use the sample to estimate population parameters
- Sample statistics ≠ Exact population values (Sampling Error exists)

Statistical Inference

From Sample to Population

- Measure the sample (e.g., sample mean = $50,000)
- Quantify uncertainty using Standard Error (SE)
- Build a Confidence Interval around the estimate
- Make decisions using Hypothesis Testing
- If you have the full population, you don't need inference. You already have the exact answer.

Statistical Inference

From Sample to Population

- Measure the sample (e.g., sample mean = $50,000)
- Quantify uncertainty using Standard Error (SE)
- Build a Confidence Interval around the estimate
- Make decisions using Hypothesis Testing
- If you have the full population, you don't need inference. You already have the exact answer.

Standard Error

Standard Error and Error Bars

- Standard Error (SE) measures how much our sample mean might differ from the true population mean
- Formula: $SE = \sigma / \sqrt{n}$
- Bar Charts: Error bars (vertical “I” marks) show uncertainty around each mean
- Line Charts: Faint shaded bands show where the true relationship likely falls
- Short bars/narrow bands = High precision; Long bars/wide bands = Low precision

Confidence Interval

How Confident are you?

- A range likely to contain the true population parameter
- 95% CI Interpretation:
  - We're 95% confident the true value falls within this range
  - Example: Mean = $50K, CI = [$47K - $53K]
- The error bars on charts are the confidence intervals showing the range of plausible values.

Calculating CI

Computing Confidence Interval

```
import seaborn as sns
import numpy as np
from scipy import stats
df = sns.load_dataset('tips')
tip_data = df['tip']
n = len(tip_data)
ci = stats.t.interval(
    confidence=0.95,
    df=n-1,
    loc=np.mean(tip_data),
    scale=stats.sem(tip_data)
)
print(f"95% CI for Mean Tip: ({ci[0]:.2f}, {ci[1]:.2f})")
```

Hypothesis Testing

Hypothetically Speaking

- Null Hypothesis (H_0): Status quo or "no effect" (e.g., "Training has no impact")
- Alternative Hypothesis (H_1): What we're testing (e.g., "Training increases sales")
- ***The Process:***
  - Collect sample data
  - Calculate test statistic and P-value
  - Make decision based on evidence

P-Value

Interpreting p-value

- Probability of observing our data if H_0 were true
- Decision Rule:
- P-value ≤ 0.05: Reject H_0 (Statistically significant, effect is real)
- P-value > 0.05: Fail to reject H_0 (Insufficient evidence)
- Example: $P = 0.02$ means only 2% chance this result happened by random chance. Strong evidence the effect is real.

Q&A