Practical Guide to Probability and Statistics in Python

---

1. Bernoulli Distribution Simulation

Theory: A Bernoulli distribution has only two possible outcomes: 1 (success) with probability p, and 0 (failure) with probability $1 - p$. It's useful for modeling binary events.

Formula: $P(X = 1) = p, \quad P(X = 0) = 1 - p$

```
from scipy.stats import bernoulli
import numpy as np
import matplotlib.pyplot as plt

p = 0.5
sample = bernoulli.rvs(p, size=1000)
plt.hist(sample, bins=2)
plt.title('Bernoulli Distribution')
plt.show()
```

---

2. Binomial Distribution - Number of Heads in Coin Flips

Theory: The binomial distribution represents the number of successes in n independent Bernoulli trials with probability p.

Formula: $P(X = k) = \binom{n}{k} p^k (1 - p)^{n-k}$

```
from scipy.stats import binom
n, p = 10, 0.5
x = np.arange(0, n+1)
pmf = binom.pmf(x, n, p)
plt.bar(x, pmf)
plt.title('Binomial PMF (n=10, p=0.5)')
plt.xlabel('Number of Heads')
plt.ylabel('Probability')
plt.show()
```

---

3. Dice Roll Simulation and Histogram

Theory: When rolling a dice, the outcomes follow a discrete distribution. We simulate biased rolls here.

```
rolls = np.random.choice([1,2,2,3,3,3], size=10000)
plt.hist(rolls, bins=[1,2,3,4], rwidth=0.8)
plt.title("Dice Roll Distribution")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()
```

---

4. Uniform Distribution with Seaborn

Theory: In a continuous uniform distribution, every interval of the same length has equal probability.

PDF Formula: $f(x) = \frac{1}{b-a}, \quad \text{for } a \leq x \leq b$

```
from scipy.stats import uniform
import seaborn as sns
sns.set(style="whitegrid")
data = uniform.rvs(loc=5, scale=7, size=10000)
sns.histplot(data, kde=True)
plt.title("Uniform Distribution")
plt.show()
```

---

5. Normal Distribution (Standard and Comparison)

Theory: The normal (Gaussian) distribution is symmetric and described by its mean μ and std deviation σ .

PDF Formula: $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

```
from scipy.stats import norm
sample = norm.rvs(loc=0, scale=1, size=10000)
sns.histplot(sample, kde=True)
plt.title("Standard Normal Distribution")
plt.show()
```

---

6. Shapiro-Wilk Test for Normality

Theory: The Shapiro-Wilk test tests the null hypothesis that a sample comes from a normal distribution.

```
from scipy.stats import shapiro
sample = np.random.normal(0, 1, 100)
stat, p = shapiro(sample)
print(f"Statistic: {stat}, p-value: {p}")
```

---

7. T-Test on Iris Petal Width

Theory: A one-sample t-test checks if the sample mean differs from a given population mean.

Formula: $t = \frac{\bar{x} - \mu}{s/\sqrt{n}}$

```
from scipy.stats import ttest_1samp
from sklearn.datasets import load_iris
iris = load_iris()
setosa_width = iris.data[iris.target == 0, 3]
stat, p = ttest_1samp(setosa_width, 0.2)
print(f"T-statistic: {stat}, p-value: {p}")
```

---

8. Boxplot for Sepal Width by Class

Theory: Boxplots show the distribution of numerical data and identify outliers and quartiles.

```
import pandas as pd
import seaborn as sns
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)
iris_df['species'] = iris.target
sns.boxplot(x='species', y='sepal width (cm)', data=iris_df)
plt.title("Boxplot of Sepal Width")
plt.show()
```

---

9. Comparing Distributions of Iris Setosa and Virginica

Theory: We compare distributions using overlapping histograms and KDEs.

```
setosa = iris.data[iris.target == 0, 0]
virginica = iris.data[iris.target == 2, 0]
sns.histplot(setosa, kde=True, color='blue', label='Setosa')
sns.histplot(virginica, kde=True, color='red', label='Virginica')
plt.legend()
plt.title("Sepal Length Distribution")
plt.show()
```

---

10. Bayes Theorem - Test Accuracy Problem

Theory: Bayes' theorem calculates conditional probability based on prior knowledge.

Formula: $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$

```
p_disease = 0.005
p_pos_given_disease = 0.95
p_neg_given_healthy = 0.95
p_healthy = 1 - p_disease

p_pos = (p_pos_given_disease * p_disease) + ((1 - p_neg_given_healthy) *
p_healthy)
p_disease_given_pos = (p_pos_given_disease * p_disease) / p_pos
print(f"Probability patient is sick given positive test: {p_disease_given_pos:.
4f}")
```

---

11. Central Limit Theorem Simulation

Theory: The CLT states that the sampling distribution of the sample mean approaches a normal distribution as the sample size increases.

```
samples = [np.mean(np.random.uniform(0, 1, 100)) for _ in range(1000)]
sns.histplot(samples, kde=True)
plt.title("CLT Simulation")
plt.show()
```

---

12. Confidence Interval Calculation

Theory: A confidence interval gives a range of values that is likely to contain a population parameter.

Formula (95% CI): $CI = \bar{x} \pm z \cdot \frac{s}{\sqrt{n}}$

```
import scipy.stats as stats
mean = np.mean(sample)
sem = stats.sem(sample)
ci = stats.t.interval(0.95, len(sample)-1, loc=mean, scale=sem)
print(f"95% confidence interval: {ci}")
```

---

13. Correlation and Covariance

Theory: Correlation measures linear relationship between variables. Covariance shows directional relationship.

```
x = np.random.normal(0, 1, 100)
y = 2*x + np.random.normal(0, 1, 100)
print("Correlation:", np.corrcoef(x, y))
print("Covariance matrix:\n", np.cov(x, y))
```

---

14. Hypothesis Testing (Two-Sample T-Test)

Theory: Test if two independent samples have different means.

```
x1 = np.random.normal(0, 1, 100)
x2 = np.random.normal(0.5, 1, 100)
stat, p = stats.ttest_ind(x1, x2)
print(f"T-statistic: {stat}, p-value: {p}")
```

---

15. Chi-Square Test for Independence

Theory: Tests if two categorical variables are independent.

```
from scipy.stats import chi2_contingency
observed = np.array([[10, 20], [20, 40]])
stat, p, dof, expected = chi2_contingency(observed)
print(f"Chi2: {stat}, p-value: {p}")
```

---

16. Poisson Distribution

Theory: Models the number of events occurring in a fixed interval of time or space.

Formula: $P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$

```
from scipy.stats import poisson
x = np.arange(0, 15)
mu = 3
plt.bar(x, poisson.pmf(x, mu))
plt.title("Poisson Distribution (mu=3)")
plt.xlabel("Events")
plt.ylabel("Probability")
plt.show()
```

---

17. Exponential Distribution

Theory: Models the time between events in a Poisson process.

PDF Formula: $f(x; \lambda) = \lambda e^{-\lambda x}$ for $x \geq 0$

```
from scipy.stats import expon
sample = expon.rvs(scale=1, size=10000)
sns.histplot(sample, kde=True)
plt.title("Exponential Distribution")
plt.show()
```

---

18. Z-Test for Population Mean

Theory: Used when population std deviation is known.

```
mu = 100
sigma = 15
sample = np.random.normal(mu, sigma, 50)
z = (np.mean(sample) - mu) / (sigma / np.sqrt(len(sample)))
p_value = 2 * (1 - stats.norm.cdf(abs(z)))
print(f"Z-score: {z:.2f}, p-value: {p_value:.4f}")
```

---

19. ANOVA (Analysis of Variance)

Theory: Tests whether there are significant differences between means of 3+ groups.

```
group1 = np.random.normal(10, 1, 30)
group2 = np.random.normal(11, 1, 30)
group3 = np.random.normal(12, 1, 30)
stat, p = stats.f_oneway(group1, group2, group3)
print(f"ANOVA statistic: {stat:.2f}, p-value: {p:.4f}")
```

---

20. Simple Linear Regression

Theory: Models relationship between two variables by fitting a linear equation.

Formula: $y = a + bx$

```
import seaborn as sns
from sklearn.linear_model import LinearRegression
x = np.random.rand(100, 1)
y = 3 * x + 2 + np.random.normal(0, 0.1, (100, 1))
model = LinearRegression().fit(x, y)
plt.scatter(x, y)
plt.plot(x, model.predict(x), color='red')
plt.title("Simple Linear Regression")
plt.show()
```
