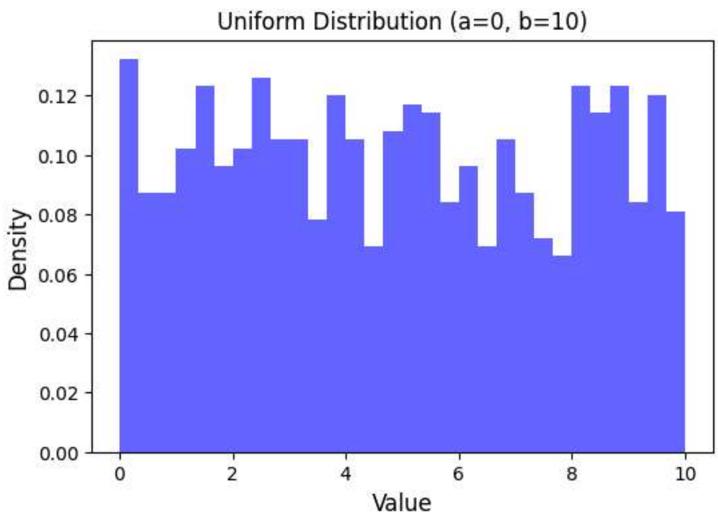
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Uniform Distribution

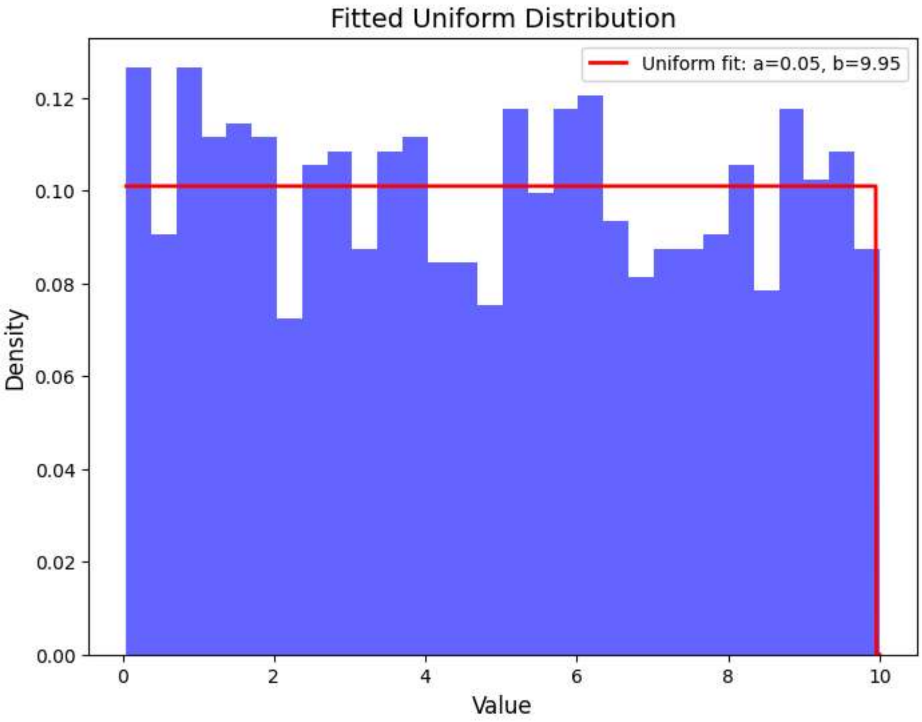
Uniform Distribution is the probability distribution that represents equal likelihood of all outcomes within a specific range. i.e. the probability of each outcome occurring is the same.

```
import numpy as np
import matplotlib.pyplot as plt

# Parameters for the uniform distribution
a, b = 0, 10 # range of the distribution
size = 1000  # Number of samples

# Generate random samples from a uniform distribution
data = np.random.uniform(a, b, size)

# Plot the histogram
plt.figure(figsize=(6, 4))
plt.hist(data, bins=30, density=True, alpha=0.6, color='b')

# Add labels and title
plt.title('Uniform Distribution (a=0, b=10)', fontsize=12)
plt.xlabel('Value', fontsize=12)
plt.ylabel('Density', fontsize=12)

# Show the plot
plt.show()
```




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

# Generate random data from a uniform distribution for demonstration
np.random.seed(42)
data = np.random.uniform(0, 10, 1000)

# Fit a uniform distribution to the data
a, b = uniform.fit(data)

# Create a range of values for plotting the PDF
x = np.linspace(min(data), max(data), 1000)
pdf = uniform.pdf(x, a, b - a)  # PDF of the uniform distribution

# Plot the histogram of the data
plt.figure(figsize=(8, 6))
plt.hist(data, bins=30, density=True, alpha=0.6, color='b')

# Plot the fitted uniform distribution
plt.plot(x, pdf, 'r-', label=f'Uniform fit: a={a:.2f}, b={b:.2f}', linewidth=2)

# Add labels and title
plt.title('Fitted Uniform Distribution', fontsize=14)
```

```
plt.xlabel('Value', fontsize=12)
plt.ylabel('Density', fontsize=12)
plt.legend()

# Show the plot
plt.show()
```



Exponential Distribution

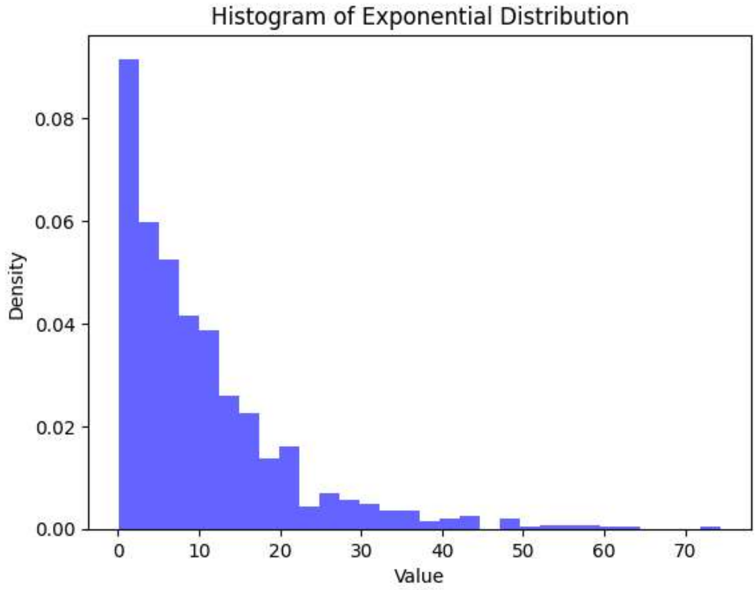
The probability density function (PDF) of the exponential distribution is given by the following equation:

$$f(x; \lambda) = \lambda e^{-\lambda x} \quad \text{for} \quad x \geq 0$$

Where:

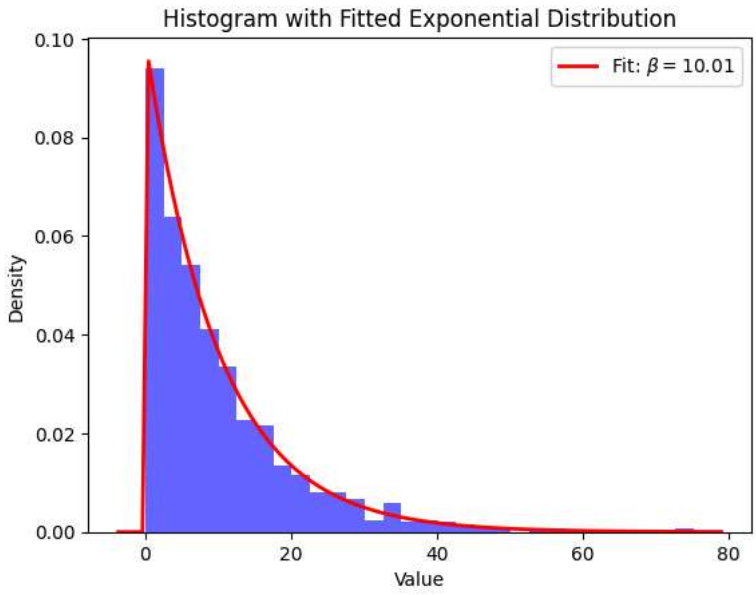
- (x) is the random variable (e.g., time between events),
- λ is the **rate parameter**, which is the inverse of the mean β = 1/λ, where β is the **location parameter** (mean time between events).

```
import numpy as np
import matplotlib.pyplot as plt

# Parameters for the exponential distribution
lamda = 0.1        #rate parameter
beta = 1/lamda          # Mean of the distribution (beta)
n_samples = 1000   # Sample size

# Generate random samples from the exponential distribution
samples = np.random.exponential(beta, n_samples)

# Plot histogram
plt.hist(samples, bins=30, density=True, alpha=0.6, color='b')

# Add titles and labels
plt.title('Histogram of Exponential Distribution')
plt.xlabel('Value')
plt.ylabel('Density')
plt.show()
```



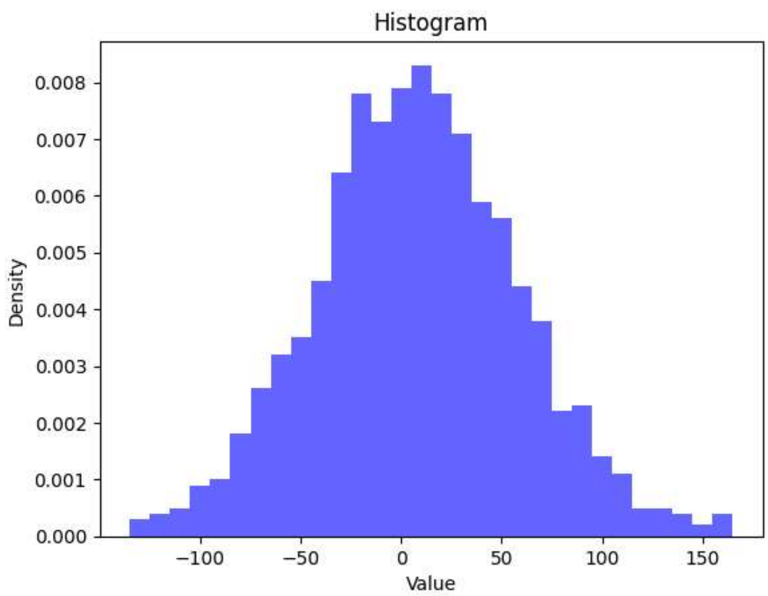

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import expon

# Parameters for the exponential distribution
lamda = 0.1 #rate parameter
beta = 1/lamda           # Mean
n_samples = 1000   # Sample size

# Generate random samples from the exponential distribution
samples = np.random.exponential(beta, n_samples)

# Fit an exponential distribution to the data
loc_fit, beta_fit = expon.fit(samples, floc=0)  # Forcing location (loc) to be 0 for simplicity

# Calculate 95% confidence interval for the mean
n = len(samples)
se = beta_fit / np.sqrt(n)
z = 1.96
muci_lower = beta_fit - z * se
muci_upper = beta_fit + z * se
muci = (muci_lower, muci_upper)

print(f"Mean (fitted beta): {beta_fit}")
print(f"95% confidence interval: {muci}")

# Plot histogram
plt.hist(samples, bins=30, density=True, alpha=0.6, color='b')

# Plot the PDF of the fitted distribution
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
pdf_fit = expon.pdf(x, loc_fit, beta_fit)
plt.plot(x, pdf_fit, 'r-', linewidth=2, label=f'Fit: $\\beta={beta_fit:.2f}$')

# Add titles and labels
plt.title('Histogram with Fitted Exponential Distribution')
plt.xlabel('Value')
plt.ylabel('Density')
plt.legend()
plt.show()
```

```
Mean (fitted beta): 10.010249457748287
95% confidence interval: (9.389807768382466, 10.630691147114108)
```



Normal Distribution

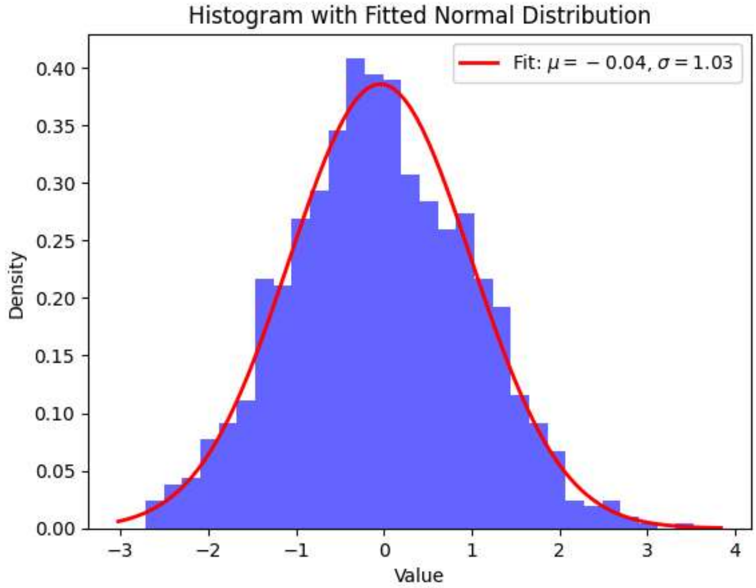
```
import numpy as np
import matplotlib.pyplot as plt

#parameters of the normal distribution
mu = 10            #mean
sigma = 50         #Standard deviation
n_samples = 1000   #sample size

#Generate random samples from the normal distribution
samples = np.random.normal(mu, sigma, n_samples)

#Plot histogram
plt.hist(samples, bins=30, density=True, alpha=0.6, color='b')

# Add titles and labels
plt.title('Histogram')
plt.xlabel('Value')
plt.ylabel('Density')
plt.show()
```




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

# Parameters of the normal distribution
```

```
mu = 0            # mean
sigma = 1         # standard deviation
n_samples = 1000  # sample size

# Generate random samples from the normal distribution
samples = np.random.normal(mu, sigma, n_samples)

# Fit a normal distribution to the data
mu_fit, sigma_fit = norm.fit(samples)

# Plot histogram
plt.hist(samples, bins=30, density=True, alpha=0.6, color='b')

# Plot the PDF of the fitted distribution
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
pdf_fit = norm.pdf(x, mu_fit, sigma_fit)
plt.plot(x, pdf_fit, 'r-', linewidth=2, label=f'Fit: $\mu={mu_fit:.2f}$, $\sigma={sigma_fit:.2f}$')

# Confidence interval for the mean
alpha = 0.05  # 95% confidence interval
z_value = norm.ppf(1 - alpha / 2)
mean_ci = (mu_fit - z_value * sigma_fit / np.sqrt(n_samples),
           mu_fit + z_value * sigma_fit / np.sqrt(n_samples))

print("Confidence interval for the mean:",mean_ci)

# Add titles and labels
plt.title('Histogram with Fitted Normal Distribution')
plt.xlabel('Value')
plt.ylabel('Density')
plt.legend()
plt.show()
```

```
Confidence interval for the mean: (-0.10294796596423095, 0.025198694190764113)
```



Hypotheses:

Question 4: The average number of collisions on a stretch of the motorway is 17 per year. The speed limit is reduced and the number of collisions in the following year is 11. Test at a 5% significance level if there is evidence to support the claim that lowering the speed limit reduces the number of collisions.

- **Null Hypothesis (H_0)**: $\mu = 17$
- **Alternative Hypothesis (H_1)**: $\mu < 17$

```
import scipy.stats as stats
import math

# Given data
mean_before = 17  # previous average number of collisions
mean_after = 11   # number of collisions after reducing the speed limit
alpha = 0.05      # significance level

# Perform the z-test
std_before = math.sqrt(mean_before)
```

```
# Calculate the z-score
z_score = (mean_after - mean_before) / (std_before)

# Find the critical z-value for a one-tailed test (since we're testing if the number reduced)
z_critical = stats.norm.ppf(alpha)

# Calculate the p-value
p_value = stats.norm.cdf(z_score)

# Output the results
print("Z-score: z_score}")
print(f"Z-critical value at 5% significance: {z_critical}")
print(f"P-value: {p_value}")

if p_value < alpha:
    print("Reject the null hypothesis: There is evidence that lowering the speed limit reduces the number of collisions.")
else:
    print("Fail to reject the null hypothesis: There is no sufficient evidence that lowering the speed limit reduces the number of coll:
```

```
Z-score: z_score}
Z-critical value at 5% significance: -1.6448536269514729
P-value: 0.07280504769843348
Fail to reject the null hypothesis: There is no sufficient evidence that lowering the speed limit reduces the number of collisions.
```

Hypothesis Testing

Question: Actual content (in ml) of tomato sauces in 10 randomly selected 250ml bottles in a bottling plant is as follows:

[248, 249.5, 250.5, 247.5, 251, 250, 250.5, 249.5, 250, 249]

a) Find a 95% confidence interval for the mean content.

b) Is it safe to assume that the mean content of the bottles is maintained at 250 ml?

c) The bottling process is said to be "out of control" if the variance of the contents exceeds 1 ml^2. Is there strong evidence suggesting that the bottling process has gone "out of control"?

```
import numpy as np
import scipy.stats as stats

# Given data
data = [248, 249.5, 250.5, 247.5, 251, 250, 250.5, 249.5, 250, 249]
n = len(data)
alpha = 0.05  # 95% confidence level

# a) Calculate the 95% confidence interval for the mean
mean = np.mean(data)
std_dev = np.std(data, ddof=1)  # Sample standard deviation

# t-critical value for 95% confidence interval with n-1 degrees of freedom
t_critical = stats.t.ppf(1 - alpha/2, df=n-1)

# Confidence interval calculation
margin_of_error = t_critical * (std_dev / np.sqrt(n))
lower_bound = mean - margin_of_error
upper_bound = mean + margin_of_error
print("a)")
print(f"95% Confidence Interval for the mean: ({lower_bound:.2f}, {upper_bound:.2f})")


# b) t-test to check if the mean is significantly different from 250 ml
hypothesized_mean = 250
t_statistic = abs((mean - hypothesized_mean) / (std_dev / np.sqrt(n)))

# Calculate the critical t-value for a two-tailed test
t_critical_b = stats.t.ppf(1 - alpha / 2, df=n-1)

# Calculate the p-value for the two-tailed test
p_value = 2 * (1 - stats.t.cdf(abs(t_statistic), df=n-1))

print("b)")
print(f"T-statistic: {t_statistic:.2f}")
print(f"Critical t-value for 95% confidence level: {t_critical_b:.2f}")
print(f"P-value: {p_value:.4f}")

if p_value < alpha:
    print("Reject the null hypothesis: The mean content is significantly different from 250 ml.")
else:
    print("Fail to reject the null hypothesis: There is no significant evidence to suggest the mean content is different from 250 ml.")
```

```
# c) Test if the variance is greater than 1 ml² (out of control)
sample_variance = np.var(data, ddof=1)
chi_square_stat = (n-1) * sample_variance / 1  # Variance = 1 for the null hypothesis

# Critical chi-square value for a one-tailed test at alpha = 0.05
chi_square_critical = stats.chi2.ppf(1 - alpha, df=n-1)

print("c)")
print(f"Sample variance: {sample_variance:.3f}")
print(f"Chi-square statistic: {chi_square_stat:.3f}")
print(f"Chi-square critical value: {chi_square_critical:.3f}")

if chi_square_stat > chi_square_critical:
    print("There is strong evidence that the bottling process is 'out of control'.")
else:
    print("There is No strong evidence that the bottling process is 'out of control'.")
```

```
a)
95% Confidence Interval for the mean: (248.75, 250.35)
b)
T-statistic: 1.27
Critical t-value for 95% confidence level: 2.26
P-value: 0.2345
Fail to reject the null hypothesis: There is no significant evidence to suggest the mean content is different from 250 ml.
c)
Sample variance: 1.247
Chi-square statistic: 11.225
Chi-square critical value: 16.919
There is No strong evidence that the bottling process is 'out of control'.
```

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.