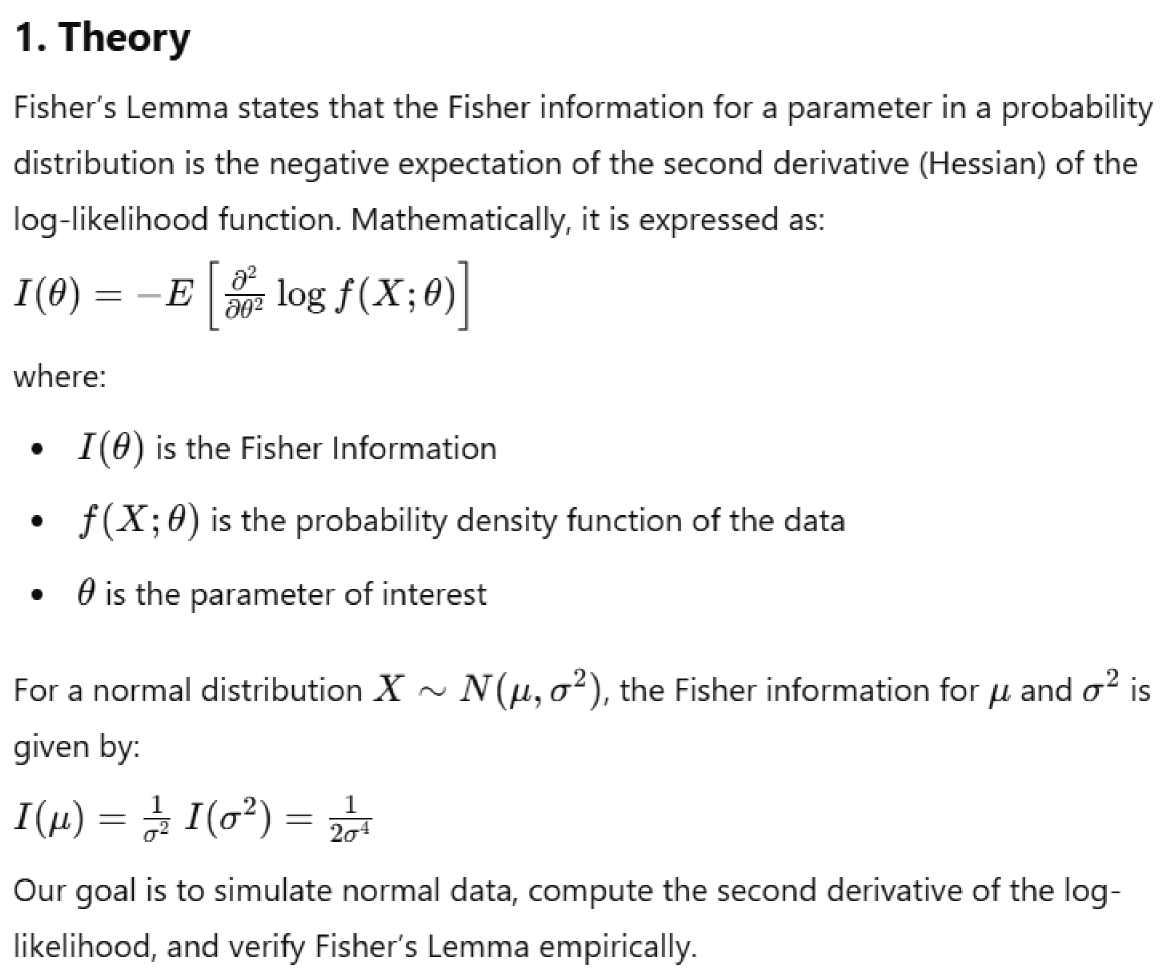
Lab 1: Verification of Fisher’s Lemma Using Simulated Data from Normal Distributions



1. Theory

Fisher’s Lemma states that the Fisher information for a parameter in a probability distribution is the negative expectation of the second derivative (Hessian) of the log-likelihood function. Mathematically, it is expressed as:

$$I(\theta) = -E\left[\frac{\partial^2}{\partial\theta^2}\log f(X;\theta)\right]$$

where:

- $I(\theta)$ is the Fisher Information
- $f(X;\theta)$ is the probability density function of the data
- θ is the parameter of interest

For a normal distribution $X \sim N(\mu, \sigma^2)$, the Fisher information for μ and σ^2 is given by:

$$I(\mu) = \frac{1}{\sigma^2}\ I(\sigma^2) = \frac{1}{2\sigma^4}$$

Our goal is to simulate normal data, compute the second derivative of the log-likelihood, and verify Fisher’s Lemma empirically.

2. Objective

The main objective of this experiment is to verify Fisher’s Lemma by comparing the empirical estimation of Fisher information with its theoretical value.

3. Pseudocode

```
1. Set parameters: mean (), standard deviation (), sample size (), and number of simulations ().
2. Initialize empty vectors for sample means and sample variances.
3. Repeat for to :
      o Generate a random sample from .
      o Compute the sample mean and variance.
      o Store values in respective vectors.
4. Compute the correlation between sample means and sample variances.
5. Display correlation result.
6. Set up a 2×2 graphical layout.
7. Generate histograms for sample means and sample variances with theoretical curves.
```

8. Create a scatter plot of sample means vs. sample variances with a reference line.
9. Generate a Q-Q plot for sample means with a reference line.

R code

```
mu <- 5

sigma <- 2

n <- 30

N_sim <- 1000


# Initialize vectors

sample_means <- numeric(N_sim)

sample_vars <- numeric(N_sim)

# Creates two empty numeric vectors of length N_sim to store sample means and sample variances.


# Simulation

set.seed(123)

for (i in 1:N_sim)

{

  data <- rnorm(n, mean = mu, sd = sigma)

  sample_means[i] <- mean(data)

  sample_vars[i] <- var(data)

}


# Check correlation

correlation <- cor(sample_means, sample_vars)

print(paste("Correlation between sample mean and sample variance:", correlation))

# Computes the correlation between sample_means and sample_vars using Pearson’s correlation coefficient.

# Displays the correlation value.


# Graphical Output

par(mfrow = c(2, 2))

# Divides the plotting area into a 2×2 grid for four plots.
```

```r
# Histogram of sample means

hist(sample_means, breaks = 30, col = "lightblue", main = "Distribution of Sample Means", xlab = "Sample Mean", border = "white")

curve(dnorm(x, mean = mu, sd = sigma/sqrt(n)), add = TRUE, col = "red", lwd = 2)


# Histogram of sample variances

hist(sample_vars, breaks = 30, col = "lightgreen", main = "Distribution of Sample Variances", xlab = "Sample Variance", border = "white")

curve(dchisq((x * (n-1)) / sigma^2, df = n-1) * (n-1)/sigma^2, add = TRUE, col = "blue", lwd = 2)


# Scatterplot of sample means vs. sample variances

plot(sample_means, sample_vars, pch = 19, col = rgb(0, 0, 1, 0.5), main = "Sample Mean vs. Sample Variance", xlab = "Sample Mean", ylab = "Sample Variance")

abline(h = sigma^2, col = "red", lwd = 2)


# Q-Q plot for sample means

qqnorm(sample_means, main = "Q-Q Plot for Sample Means", col = "blue")

qqline(sample_means, col = "red", lwd = 2)
```

Sample Input and Output

Input Parameters:

- Mean (): 5
- Standard deviation (): 2
- Sample size (): 30
- Number of simulations (): 1000

Output Parameters:

Correlation between sample mean and sample variance: -0.0288960157244368

Lab 2: Generation and Analysis of χ2-Distributed Data

1. Theory

The chi-squared (χ^2) distribution is widely used in statistics, particularly in hypothesis testing and confidence interval estimation. It is defined as the sum of the squares of k independent standard normal random variables:

$$X = Z_1^2 + Z_2^2 + ... + Z_k^2$$

where $Z_i \sim N(0,1)$. The mean and variance of a χ^2 distribution with k degrees of freedom are:

- $E[X] = k$
- $Var(X) = 2k$

2. Objective

The objective of this lab is to generate χ²-distributed data, compute empirical mean and variance, and compare them with theoretical values. Additionally, we visualize the distribution using histograms, density plots, and Q-Q plots.

3. Pseudocode

```
1. Set parameters: degrees of freedom and number of simulations .
2. Generate random samples from a χ² distribution.
3. Compute empirical mean and variance.
4. Print computed mean and variance.
5. Generate and display:
    o Histogram with theoretical χ² density curve.
    o Density plot with theoretical χ² curve.
    o Q-Q plot comparing sample quantiles to theoretical quantiles.
```

R code

```
# Parameters
k <- 5  # Degrees of freedom
N_sim <- 1000  # Number of simulations

# Generate chi-squared data
set.seed(123)
```

```r
chi2_data <- rchisq(N_sim, df = k)

# set.seed(123) ensures reproducibility, meaning running the code multiple times will yield the same random numbers.

rchisq(N_sim, df = k)

# generates 1000 random numbers from a χ2 distribution with 5 degrees of freedom.


# Compute mean and variance

mean_chi2 <- mean(chi2_data)  # Compute mean

var_chi2 <- var(chi2_data)  # Compute variance


print(paste("Mean:", mean_chi2))

print(paste("Variance:", var_chi2))


# Graphical Output

par(mfrow = c(1, 3))

# Set plotting layout to 1 row, 3 columns


# Histogram

hist(chi2_data, breaks = 30, col = "lightblue", probability = TRUE, main = "Chi-Squared Distribution", xlab = "Value", border = "white")

# lots a histogram with density on the y-axis.

curve(dchisq(x, df = k), add = TRUE, col = "red", lwd = 2)

# overlays the theoretical χ2 density curve in red.


# Density plot

plot(density(chi2_data), col = "blue", lwd = 2, main = "Density Plot", xlab = "Value")

#  creates a smooth density plot of the simulated data.

curve(dchisq(x, df = k), add = TRUE, col = "red", lwd = 2)

# overlays the theoretical χ2 density function.


# Q-Q plot (Quantile-Quantile Plot)

qqplot(qchisq(ppoints(N_sim), df = k), chi2_data, main = "Q-Q Plot for Chi-Squared Data", col = "blue", xlab = "Theoretical Quantiles", ylab = "Sample Quantiles")
```

```
abline(0, 1, col = "red", lwd = 2)
```

Sample Input/ Output

Input Parameters:

k = 5

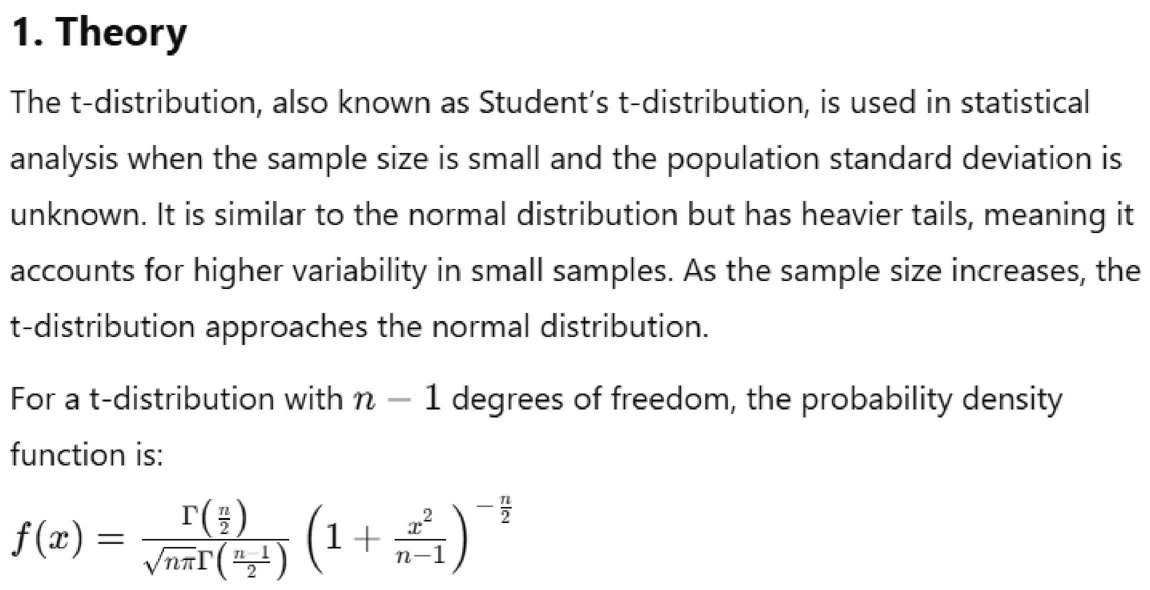
N_sim = 1000

Output Parameters:

Mean: 4.79389985708151

Variance: 8.4017924901745

Lab 3: Comparison of t-Distribution with Normal Distribution for Small Sample Sizes



1. Theory

The t-distribution, also known as Student's t-distribution, is used in statistical analysis when the sample size is small and the population standard deviation is unknown. It is similar to the normal distribution but has heavier tails, meaning it accounts for higher variability in small samples. As the sample size increases, the t-distribution approaches the normal distribution.

For a t-distribution with $n-1$ degrees of freedom, the probability density function is:

$$f(x) = \frac{\Gamma\left(\frac{n}{2}\right)}{\sqrt{n\pi}\,\Gamma\left(\frac{n-1}{2}\right)} \left(1 + \frac{x^2}{n-1}\right)^{-\frac{n}{2}}$$

Objective

The objective of this experiment is to compare the t-distribution with the normal distribution for small sample sizes by:

- Generating t-distributed and normal data.
- Visualizing their histograms and density functions.
- Comparing quantiles using Q-Q plots.

3. Pseudocode

1. Set parameters: sample size and number of simulations .
2. Generate random values from a t-distribution with degrees of freedom.
3. Generate random values from a standard normal distribution.

4. Plot:
    - Histogram of t-distribution with theoretical density.
    - Histogram of normal distribution with theoretical density.
    - Density plots of both distributions for comparison.
    - Q-Q plot for t-distribution against theoretical quantiles.

R code

```r
# Parameters
n <- 10  # Sample size
N_sim <- 1000  # Number of simulations

# Generate data
set.seed(123)  # Ensures reproducibility
t_data <- rt(N_sim, df = n-1)  # Generate t-distributed data with df = n-1
normal_data <- rnorm(N_sim)  # Generate standard normal data

# Graphical Output
par(mfrow = c(2, 2)) # Set layout: 2 rows, 2 columns

# Histogram of t-distribution
hist(t_data, breaks = 30, col = "lightblue", probability = TRUE, main = "t-Distribution", xlab = "Value", border = "white")
curve(dt(x, df = n-1), add = TRUE, col = "red", lwd = 2) #overlays the theoretical t-distribution density.

# Histogram of normal distribution
hist(normal_data, breaks = 30, col = "lightgreen", probability = TRUE, main = "Normal Distribution", xlab = "Value", border = "white")
curve(dnorm(x), add = TRUE, col = "blue", lwd = 2)  # overlays the theoretical normal density.

# Density plot comparison
plot(density(t_data), col = "red", lwd = 2, main = "Density Comparison", xlab = "Value", ylim = c(0, 0.4))
lines(density(normal_data), col = "blue", lwd = 2)
```

```
legend("topright", legend = c("t-Distribution", "Normal Distribution"), col = c("red", "blue"), lwd = 2)


# Q-Q plot for t-distribution

qqplot(qt(ppoints(N_sim), df = n-1), t_data, main = "Q-Q Plot for t-Distribution", col = "red", xlab = "Theoretical Quantiles", ylab = "Sample Quantiles")

abline(0, 1, col = "blue", lwd = 2)
```

Sample Input/Output:

Input parameter:

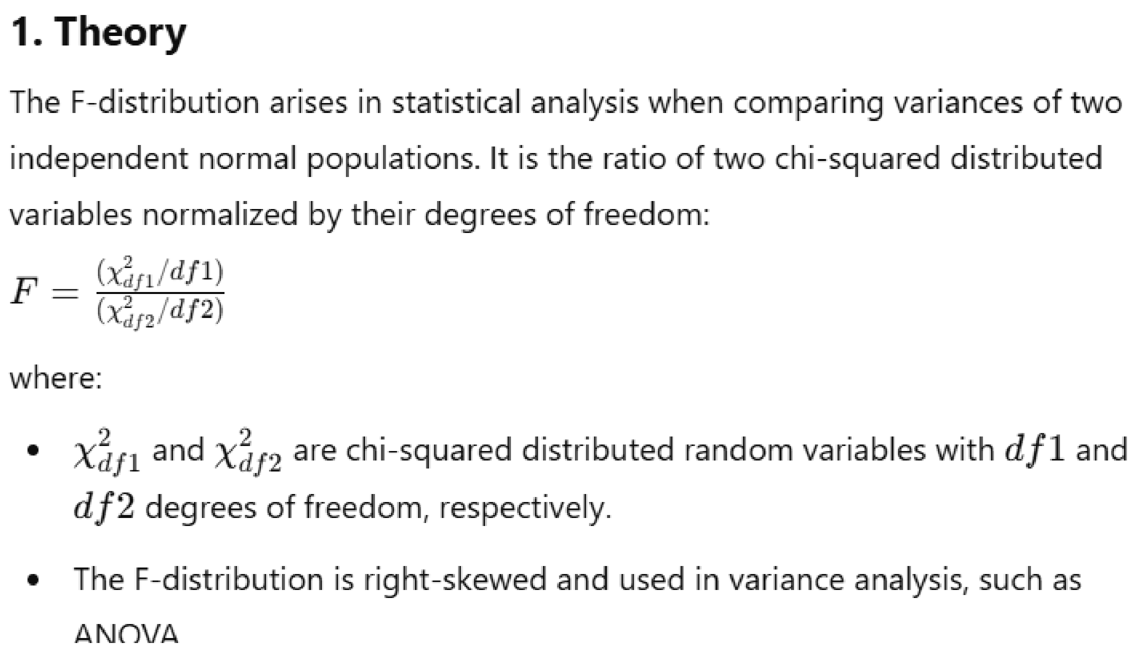
n = 10

N_sim = 1000

Output (Graphical Plots)

1. **Histogram of t-Distribution** (Slightly heavier tails than normal).
2. **Histogram of Normal Distribution** (Bell-shaped curve).
3. **Density Comparison Plot** (Shows wider tails for t-distribution).
4. **Q-Q Plot** (Compares sample t-data to theoretical quantiles).

# Lab 4: Simulation of F-Distributed Data and Its Relationship with χ2-Distributions



1. Theory

The F-distribution arises in statistical analysis when comparing variances of two independent normal populations. It is the ratio of two chi-squared distributed variables normalized by their degrees of freedom:

$$F = \frac{(\chi^2_{df1}/df1)}{(\chi^2_{df2}/df2)}$$

where:

- χ^2_{df1} and χ^2_{df2} are chi-squared distributed random variables with $df1$ and $df2$ degrees of freedom, respectively.
- The F-distribution is right-skewed and used in variance analysis, such as ANOVA

Objective

The objective of this experiment is to simulate F-distributed data using chi-squared distributions and analyze its properties using:

- Histogram and density plots.
- Q-Q plot for distribution verification.
- Boxplot to observe spread and skewness.

3. Pseudocode

1. Set parameters: numerator degrees of freedom , denominator degrees of freedom , and number of simulations .
2. Generate chi-squared distributed random numbers for both numerator and denominator.
3. Compute the F-distributed values using their ratio.
4. Generate graphical outputs:
    - Histogram with theoretical density curve.
    - Density plot overlaying the theoretical curve.
    - Q-Q plot comparing sample quantiles to theoretical quantiles.
    - Boxplot to visualize spread and skewness.

4. R Code

```r
# Parameters
df1 <- 5  # Numerator degrees of freedom
df2 <- 10  # Denominator degrees of freedom
N_sim <- 1000  # Number of simulations

# Generate F-distributed data
set.seed(123)  # Ensures reproducibility
chi2_1 <- rchisq(N_sim, df = df1)  # Generate Chi-Squared data with df1
chi2_2 <- rchisq(N_sim, df = df2)  # Generate Chi-Squared data with df2
f_data <- (chi2_1 / df1) / (chi2_2 / df2)  # Compute F-distributed values

# Graphical Output
par(mfrow = c(2, 2))       # Set layout: 2 rows, 2 columns

# Histogram
hist(f_data, breaks = 30, col = "lightblue", probability = TRUE, main = "F-Distribution", xlab = "Value", border = "white")

curve(df(x, df1 = df1, df2 = df2), add = TRUE, col = "red", lwd = 2)
```

```
# Density plot

plot(density(f_data), col = "blue", lwd = 2, main = "Density Plot", xlab = "Value")


curve(df(x, df1 = df1, df2 = df2), add = TRUE, col = "red", lwd = 2)



# Q-Q plot

qqplot(qf(ppoints(N_sim), df1 = df1, df2 = df2), f_data, main = "Q-Q Plot for F-Distribution", col = "blue", xlab = "Theoretical Quantiles", ylab = "Sample Quantiles")

abline(0, 1, col = "red", lwd = 2)


# Boxplot

boxplot(f_data, col = "lightgreen", main = "Boxplot of F-Distributed Data", ylab = "Value")


# The boxplot shows the spread and skewness of the F-distributed data.

# Since the F-distribution is right-skewed, the upper whisker is typically longer.
```

Sample Input and Output

Input parameter:

df1 = 5

df2 = 10

N_sim = 1000

Output:

- Histogram of F-distribution with theoretical density curve.
- Density plot showing empirical and theoretical distributions.
- Q-Q plot indicating the goodness of fit.
- Boxplot illustrating the skewness of the F-distribution.

Lab 5: Distribution of Medians and Ranges from Sampled Populations

Theory

In this lab, the goal is to examine the **distributions of the medians** and **ranges** computed from multiple samples drawn from a normal distribution. The **median** is the middle value when the data is sorted, and the **range** is the difference between the maximum and minimum values in the sample.

- The **median** is a robust measure of central tendency, especially in the presence of outliers.
- The **range** provides a measure of the spread or variability in the sample data, but it can be heavily influenced by outliers.

By simulating multiple samples from a normal distribution, we can investigate how the distributions of medians and ranges behave across repeated sampling. For each sample, we calculate both the median and the range, and then analyze their distributions.

Objective

1. **Simulate 1000 random samples** from a normal distribution with mean (mu = 0) and standard deviation (sigma = 1), each containing 20 values.
2. **Compute the median** and **range** for each sample.
3. **Visualize the distributions** of medians and ranges through histograms, density plots, and boxplots.
4. Compare the distributions of medians and ranges with the normal distribution and inspect their shapes.

Pseudocode

```
1. Set parameters:
    o mu = 0 (mean of the normal distribution)
    o sigma = 1 (standard deviation of the normal distribution)
    o n = 20 (sample size for each simulation)
    o N_sim = 1000 (number of simulations)
2. Initialize vectors to store the results of the medians and ranges for each simulation.
3. Simulation Loop:
    o For each of the 1000 simulations:
        ▪ Generate a random sample of size n from the normal distribution.
        ▪ Compute and store the median of the sample.
        ▪ Compute and store the range of the sample (difference between max and min values).
4. Set graphical layout to a 2×2 grid.
5. Create histograms for the distribution of medians and ranges:
    o Overlay the theoretical normal distribution curve for medians.
```

6. **Create a density plot** for the distribution of medians.
    - Overlay the theoretical normal distribution curve.
7. **Create a boxplot** comparing the distributions of medians and ranges.

R code

```
# Parameters

mu <- 0

sigma <- 1

n <- 20

N_sim <- 1000


# Initialize vectors

medians <- numeric(N_sim)

ranges <- numeric(N_sim)


# Simulation

set.seed(123)

for (i in 1:N_sim) {

  data <- rnorm(n, mean = mu, sd = sigma)

  medians[i] <- median(data)

  ranges[i] <- max(data) - min(data)

}


# Graphical Output

par(mfrow = c(2, 2))


# Histogram of medians

hist(medians, breaks = 30, col = "lightblue", probability = TRUE, main = "Distribution of Medians", xlab = "Median",
border = "white")

curve(dnorm(x, mean = mu, sd = sigma/sqrt(n)), add = TRUE, col = "red", lwd = 2)


# Histogram of ranges
```

```
hist(ranges, breaks = 30, col = "lightgreen", probability = TRUE, main = "Distribution of Ranges", xlab = "Range", border =
"white")


# Density plot of medians

plot(density(medians), col = "blue", lwd = 2, main = "Density Plot of Medians", xlab = "Median")

curve(dnorm(x, mean = mu, sd = sigma/sqrt(n)), add = TRUE, col = "red", lwd = 2)


# Boxplot of medians and ranges

boxplot(list(Medians = medians, Ranges = ranges), col = c("lightblue", "lightgreen"), main = "Boxplot of Medians and
Ranges", ylab = "Value")
```

Sample Input & Output

Since this code generates random data, the exact numerical results will vary. However, the graphical outputs will have a clear pattern.

# Lab 6: Estimate Population Parameters (Mean, Variance) from Sample Data

Theory

When we draw a sample from a population, we use the sample data to **estimate population parameters** such as the **mean** and **variance**. The **sample mean** and **sample variance** are point estimates, but they come with some uncertainty. Therefore, we can calculate **confidence intervals** to estimate the range in which the true population parameters likely fall.

- The **sample mean** is an estimate of the population mean.
- The **sample variance** is an estimate of the population variance, though it is biased, and needs to be adjusted when constructing confidence intervals.
- A **confidence interval** provides a range of values within which the population parameter is likely to lie, given the sample data.

Objective

1. **Generate sample data** from a normal distribution using the known population parameters (mean = 5, standard deviation = 2).
2. **Estimate the population parameters** (mean and variance) from the sample.
3. Calculate the **95% confidence interval** for both the sample mean and the sample variance.
4. **Visualize** the sample data with:

- A histogram showing the sample data and highlighting the sample mean and the confidence interval for the mean.
- A boxplot to summarize the distribution of the sample data.

Pseudocode

```
1. Set Parameters:
    o mu = 5 (population mean)
    o sigma = 2 (population standard deviation)
    o n = 30 (sample size)
    o N_sim = 1000 (number of simulations, though not used directly here)
2. Generate Sample Data:
    o Draw n = 30 random samples from a normal distribution with mean mu = 5 and standard deviation sigma = 2.
3. Point Estimates:
    o Calculate the sample mean: sample_mean = mean(sample_data)
    o Calculate the sample variance: sample_var = var(sample_data)
4. Confidence Intervals:
    o Compute the 95% confidence interval for the mean using the t.test function.
    o Compute the 95% confidence interval for the variance using the chi-squared distribution formula.
5. Output Results:
    o Print the sample mean, sample variance, confidence interval for the mean, and confidence interval for the variance.
6. Graphical Output:
    o Histogram: Display a histogram of the sample data with vertical lines indicating the sample mean and the confidence interval for the mean.
    o Boxplot: Display a boxplot of the sample data.
```

R code:

```r
# Parameters

mu <- 5

sigma <- 2

n <- 30

N_sim <- 1000
```

```
# Generate sample data

set.seed(123)

sample_data <- rnorm(n, mean = mu, sd = sigma)

# Point estimates

sample_mean <- mean(sample_data)

sample_var <- var(sample_data)

# Confidence intervals

conf_int_mean <- t.test(sample_data)$conf.int

conf_int_var <- c((n-1)*sample_var / qchisq(0.975, df = n-1),

(n-1)*sample_var / qchisq(0.025, df = n-1))

# Output

print(paste("Sample Mean:", sample_mean))

print(paste("Sample Variance:", sample_var))

print(paste("95% CI for Mean:", conf_int_mean))

print(paste("95% CI for Variance:", conf_int_var))

# Graphical Output

par(mfrow = c(1, 2))

# Histogram with mean and CI

hist(sample_data, breaks = 30, col = "lightblue", main = "Sample Data with Mean", xlab = "Value", border = "white")

abline(v = sample_mean, col = "red", lwd = 2)

abline(v = conf_int_mean, col = "blue", lty = 2)

# Boxplotboxplot(sample_data, col = "lightgreen", main = "Boxplot of Sample Data", ylab = "Value")
```

Sample Input/Output:

Input parameter:

```
Sample Mean: 4.90579248793639

Sample Variance: 3.84968498055034

95% CI for Mean: (4.1731467163713, 5.63843825950147)

95% CI for Variance: (2.44171660432967, 6.95708641148571)
```

Output (Graphical Plots):

A histogram of the sample data with Mean

Lab 7: Demonstrate Consistency by Increasing Sample Size

1. Theory

Consistency of an estimator means that as the sample size increases, the estimator converges to the true parameter value. The sample mean $\bar{X}$ is a consistent estimator of the population mean μ, and its variance decreases as the sample size increases, following:

$$\mathrm{Var}(\bar{X}) = \frac{\sigma^2}{n}$$

2. Objective

To demonstrate consistency by showing that:

- The sample mean converges to the population mean (μ).
- The variance of the sample mean decreases with increasing sample size.

3. Pseudocode

```
1. Define parameters: population mean ($\mu$), standard deviation ($\sigma$), sample sizes, and number of
   simulations ($N_{sim}$).
2. Initialize vectors for sample means and variances.
3. For each sample size:
    • Generate $N_{sim}$ sample means.
    • Compute and store the average sample mean and variance.
4. Plot the sample means and variances against sample sizes with theoretical reference lines.
```

R code:

```
mu <- 5

sigma <- 2

sample_sizes <- c(10, 30, 100, 500, 1000)

N_sim <- 1000

# Initialize vectors

means <- numeric(length(sample_sizes))

vars <- numeric(length(sample_sizes))

# Simulation
```

```
set.seed(123)

for (i in 1:length(sample_sizes)) {

 n <- sample_sizes[i]

 sample_means <- replicate(N_sim, mean(rnorm(n, mean = mu, sd = sigma)))

 means[i] <- mean(sample_means)

 vars[i] <- var(sample_means)

}

# Graphical Output

par(mfrow = c(1, 2))

# Plot sample means

plot(sample_sizes, means, type = "b", col = "blue", main = "Convergence of Sample Mean", xlab = "Sample Size", ylab = "Sample Mean")

abline(h = mu, col = "red", lwd = 2)

# Plot sample variances

plot(sample_sizes, vars, type = "b", col = "green", main = "Convergence of Sample Variance", xlab = "Sample Size", ylab = "Sample Variance")

abline(h = sigma^2, col = "red", lwd = 2)
```

Sample Input and Output:

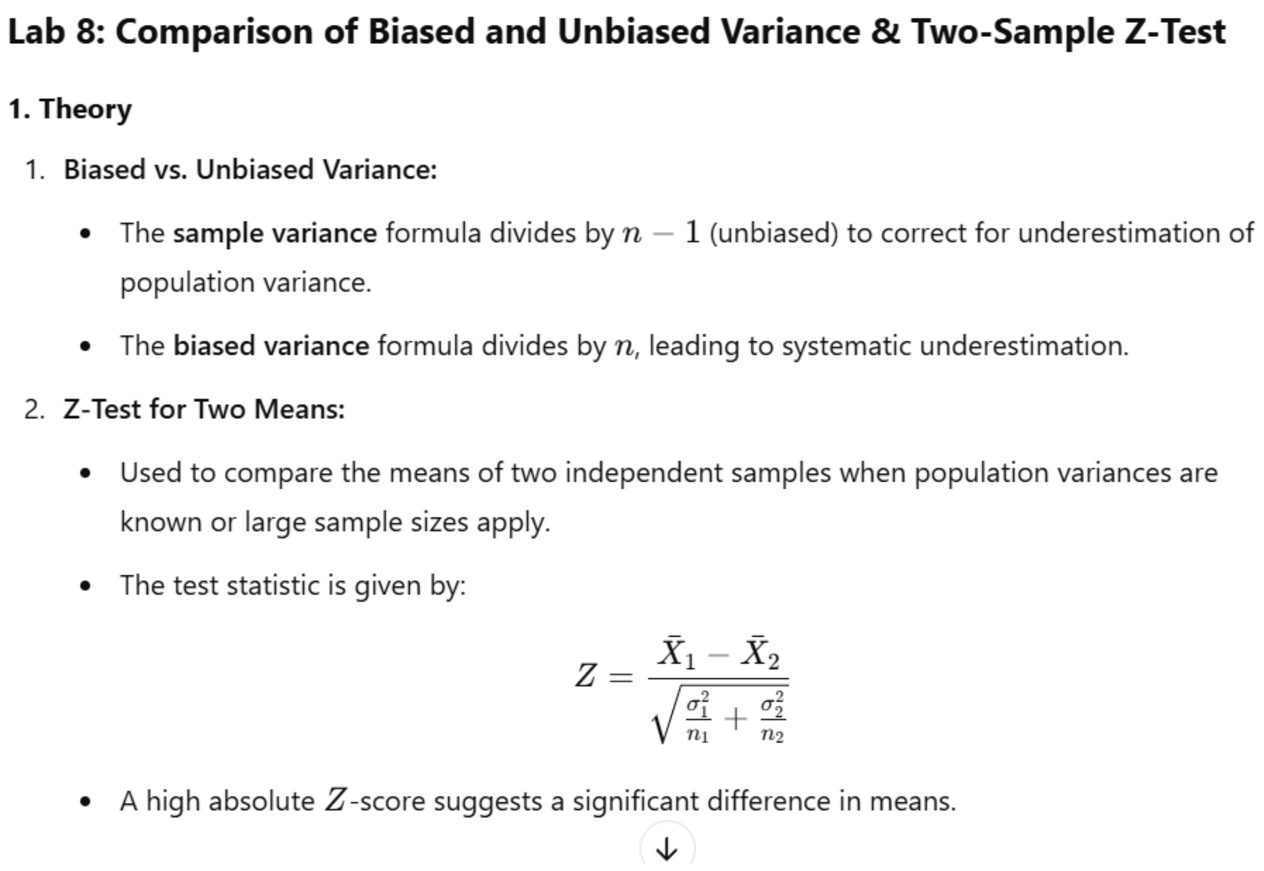
Input:

mu = 5

sigma = 2

sample_sizes = [10, 30, 100, 500, 1000]

N_sim = 1000



Lab 8: Comparison of Biased and Unbiased Variance & Two-Sample Z-Test

1. Theory

1. **Biased vs. Unbiased Variance:**
   - The **sample variance** formula divides by $n-1$ (unbiased) to correct for underestimation of population variance.
   - The **biased variance** formula divides by n, leading to systematic underestimation.
2. **Z-Test for Two Means:**
   - Used to compare the means of two independent samples when population variances are known or large sample sizes apply.
   - The test statistic is given by:

$$Z = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$

   - A high absolute Z-score suggests a significant difference in means.

2. Objective

1. To compare biased vs. unbiased sample variance using simulation.
2. To perform a two-sample **Z-test** to determine if there is a significant difference between two independent groups.

---

3. Pseudocode

1. **Variance Comparison:**
   - Set parameters: μ,σ,n,Nsim\mu, \sigma, n, N_{sim}μ,σ,n,Nsim.
   - Initialize vectors for sample variance (unbiased) and biased variance.
   - For each simulation:
     - Generate a normal sample.
     - Compute unbiased and biased variance.
   - Plot histograms of both variances.
2. **Two-Sample Z-Test:**
   - Generate two normal samples with different means.
   - Perform a **Z-test** using z.test().
   - Plot histograms of both groups.

R code:

```
# Load necessary library
library(BSDA)

# Parameters
mu <- 5
sigma <- 2
n <- 30
N_sim <- 1000

# Initialize vectors
sample_vars <- numeric(N_sim)
biased_vars <- numeric(N_sim)

# Simulation
set.seed(123)
for (i in 1:N_sim) {
  data <- rnorm(n, mean = mu, sd = sigma)
  sample_vars[i] <- var(data)  # Unbiased variance
  biased_vars[i] <- sum((data - mean(data))^2) / n  # Biased variance
}

# Graphical Output
par(mfrow = c(1, 2))

# Histogram of unbiased variances
hist(sample_vars, breaks = 30, col = "lightblue",
     main = "Unbiased Sample Variance", xlab = "Variance", border = "white")
abline(v = sigma^2, col = "red", lwd = 2)
```

```
# Histogram of biased variances
hist(biased_vars, breaks = 30, col = "lightgreen",
     main = "Biased Sample Variance", xlab = "Variance", border = "white")
abline(v = sigma^2, col = "red", lwd = 2)

# Define groups for z-test
set.seed(123)
group1 <- rnorm(30, mean = 50, sd = 10)
group2 <- rnorm(30, mean = 55, sd = 10)

# Perform z-test
z_test_result <- z.test(group1, group2, sigma.x = sd(group1), sigma.y = sd(group2))

# Output test result
print(z_test_result)

# Graphical Output
par(mfrow = c(1, 2))

# Histogram of groups
hist(group1, breaks = 30, col = "lightblue",
     main = "Histogram of Group 1", xlab = "Value", border = "white")
hist(group2, breaks = 30, col = "lightgreen",
     main = "Histogram of Group 2", xlab = "Value", border = "white")
```

Sample Input and Output

Input:

```
mu = 5
sigma = 2
n = 30
N_sim = 1000
group1 <- rnorm(30, mean = 50, sd = 10)
group2 <- rnorm(30, mean = 55, sd = 10)
```

Lab 9: Efficiency of Mean vs. Median

1. Theory

- The **sample mean** is the most efficient estimator for normally distributed data, having the smallest variance.
- The **sample median** is more robust to outliers but has higher variance.
- **Efficiency** is measured as the ratio of variances:

$$\text{Efficiency} = \frac{\text{Var}(\text{Median})}{\text{Var}(\text{Mean})}$$

A lower value (<1) indicates that the mean is more efficient.

2. Objective

To compare the efficiency of the sample mean and median using simulation.

3. Pseudocode

```
1. Set parameters: μ,σ,n,Nsim\mu, \sigma, n, N_{\text{sim}}μ,σ,n,Nsim.
2. Initialize vectors for sample means and medians.
3. For each simulation:
     o Generate normal data.
     o Compute sample mean and median.
4. Compute efficiency as the ratio of variances.
5. Plot histograms of means and medians.
```

R code:

```
# Parameters

mu <- 5

sigma <- 2

n <- 30

N_sim <- 1000


# Initialize vectors

means <- numeric(N_sim)

medians <- numeric(N_sim)
```

```
# Simulation
set.seed(123)
for (i in 1:N_sim) {
  data <- rnorm(n, mean = mu, sd = sigma)
  means[i] <- mean(data)
  medians[i] <- median(data)
}

# Efficiency (ratio of variances)
efficiency <- var(medians) / var(means)
print(paste("Efficiency (Median/Mean):", efficiency))

# Graphical Output
par(mfrow = c(1, 2))

# Histogram of means
hist(means, breaks = 30, col = "lightblue",
     main = "Distribution of Sample Means",
     xlab = "Value", border = "white")

# Histogram of medians
hist(medians, breaks = 30, col = "lightgreen",
     main = "Distribution of Sample Medians",
     xlab = "Value", border = "white")
```

Sample Input & Output

Input:

```
mu = 5
sigma = 2
n = 30
N_sim = 1000
```

Output: Efficiency (Median/Mean): 1.5085760936887

Lab 10: MLE for Binomial, Poisson & Normal Distributions

1. Theory

- Binomial MLE: $\hat{p} = \frac{\sum X}{n}$
- Poisson MLE: $\hat{\lambda} = \frac{\sum X}{N}$
- Normal MLE: $\hat{\mu} = \frac{\sum X}{N}, \quad \hat{\sigma} = \sqrt{\frac{\sum (X-\hat{\mu})^2}{N}}$

2. Objective

Estimate parameters using Maximum Likelihood Estimation (MLE).

3. Pseudocode

```
1. Generate Binomial, Poisson, and Normal data.
2. Compute MLE estimates for $p, \lambda, \mu, \sigma$.
3. Print results.
```

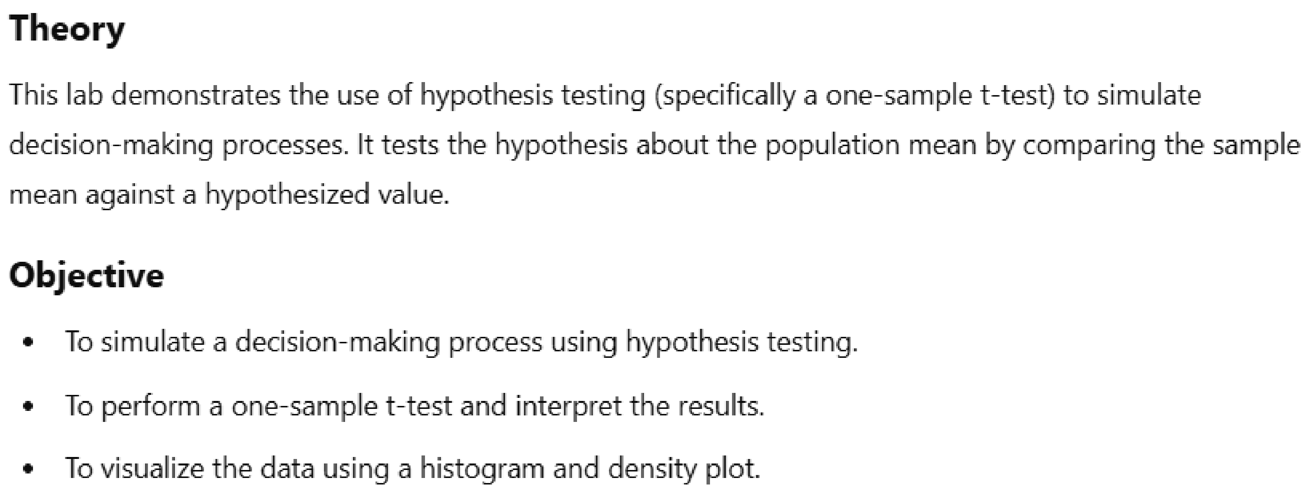
R code:

```r
# Lab 10: Derive MLEs for Binomial, Poisson, and Normal Distributions

# Binomial MLE

n_binom <- 20

p_true <- 0.6

data_binom <- rbinom(100, size = n_binom, prob = p_true)

p_mle <- mean(data_binom) / n_binom

# Poisson MLE

lambda_true <- 3

data_pois <- rpois(100, lambda = lambda_true)

lambda_mle <- mean(data_pois)

# Normal MLE

mu_true <- 5

sigma_true <- 2

data_norm <- rnorm(100, mean = mu_true, sd = sigma_true)

mu_mle <- mean(data_norm)

sigma_mle <- sqrt(mean((data_norm - mu_mle)^2))

# Output

print(paste("Binomial MLE for p:", p_mle))
print(paste("Poisson MLE for lambda:", lambda_mle))

print(paste("Normal MLE for mu:", mu_mle))

print(paste("Normal MLE for sigma:", sigma_mle))
```

Sample output:

```
Binomial MLE for p: 0.603

Poisson MLE for lambda: 2.89

Normal MLE for mu: 5.35420581393028

Normal MLE for sigma: 1.8574597852958
```

# Lab 11: Simulate Decision-Making Processes Using Hypothesis Testing

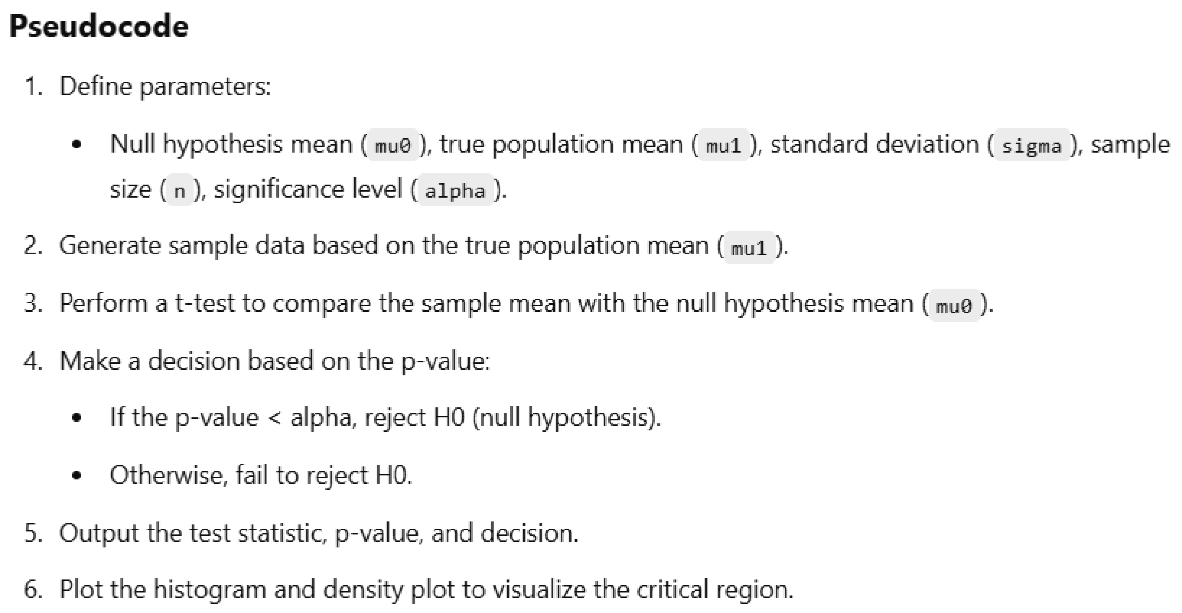


Theory

This lab demonstrates the use of hypothesis testing (specifically a one-sample t-test) to simulate decision-making processes. It tests the hypothesis about the population mean by comparing the sample mean against a hypothesized value.

Objective

- To simulate a decision-making process using hypothesis testing.
- To perform a one-sample t-test and interpret the results.
- To visualize the data using a histogram and density plot.



Pseudocode

```
1. Define parameters:
    • Null hypothesis mean (mu0), true population mean (mu1), standard deviation (sigma), sample
      size (n), significance level (alpha).
2. Generate sample data based on the true population mean (mu1).
3. Perform a t-test to compare the sample mean with the null hypothesis mean (mu0).
4. Make a decision based on the p-value:
    • If the p-value < alpha, reject H0 (null hypothesis).
    • Otherwise, fail to reject H0.
5. Output the test statistic, p-value, and decision.
6. Plot the histogram and density plot to visualize the critical region.
```

R code:

```
mu0 <- 5 # Null hypothesis mean

mu1 <- 6 # True population mean

sigma <- 2 # Population standard deviation

n <- 30 # Sample size

alpha <- 0.05 # Significance level

# Generate sample data

set.seed(123)

sample_data <- rnorm(n, mean = mu1, sd = sigma)

# Perform t-test

t_test_result <- t.test(sample_data, mu = mu0, alternative = "greater")

# Decision

if (t_test_result$p.value < alpha) {

 decision <- "Reject H0"

} else {

 decision <- "Fail to reject H0"

}

# Output

print(paste("Test Statistic:", t_test_result$statistic))

print(paste("P-value:", t_test_result$p.value))

print(paste("Decision:", decision))

# Graphical Output

par(mfrow = c(1, 2))

# Histogram with critical region

hist(sample_data, breaks = 30, col = "lightblue", main = "Sample Data", xlab = "Value", border = "white")

abline(v = mu0, col = "red", lwd = 2)

abline(v = mean(sample_data), col = "blue", lwd = 2)

# Density plot with critical region

plot(density(sample_data), col = "blue", lwd = 2, main = "Density Plot", xlab = "Value")

abline(v = qt(1 - alpha, df = n-1), col = "red", lty = 2)
```

Sample Input/output:

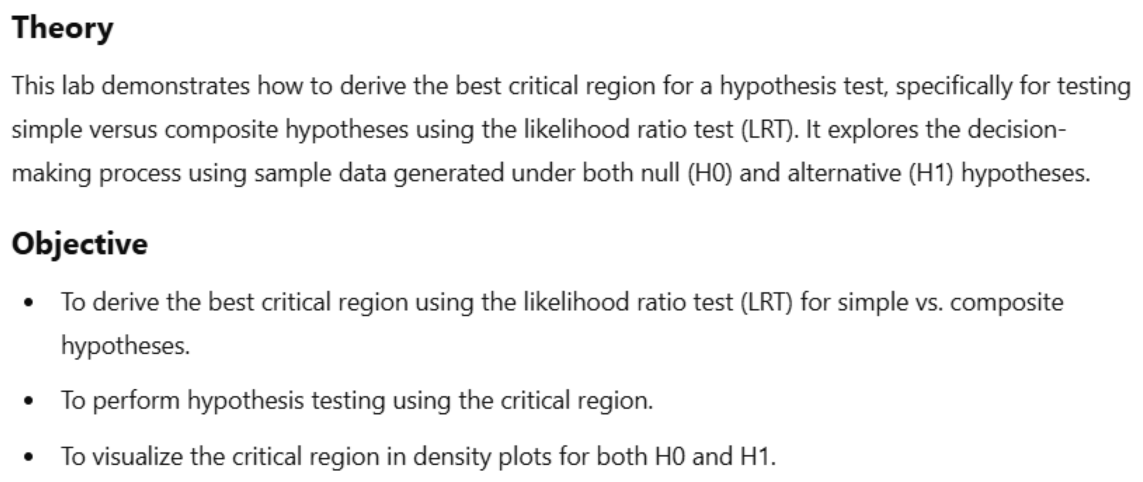
Input:

```
mu0 <- 5       # Null hypothesis mean

mu1 <- 6       # True population mean

sigma <- 2     # Population standard deviation

n <- 30        # Sample size

alpha <- 0.05  # Significance level
```

Sample Output:

```
Test Statistic: 2.52858027419059

P-value: 0.00857566021520242

Decision: Reject H0
```

# Lab 12: Derive the Best Critical Region for Simple vs. Composite Hypotheses



Theory

This lab demonstrates how to derive the best critical region for a hypothesis test, specifically for testing simple versus composite hypotheses using the likelihood ratio test (LRT). It explores the decision-making process using sample data generated under both null (H0) and alternative (H1) hypotheses.

Objective

- To derive the best critical region using the likelihood ratio test (LRT) for simple vs. composite hypotheses.
- To perform hypothesis testing using the critical region.
- To visualize the critical region in density plots for both H0 and H1.

Pseudocode:

1. **Set Parameters**: `mu0`, `mu1`, `sigma`, `n`, `alpha`
2. **Generate Data**:
   - `sample_data_H0` from `mu0`
   - `sample_data_H1` from `mu1`
3. **Likelihood Ratio Test**:
   - Calculate likelihood ratio for data
4. **Critical Value**: `critical_value = qnorm(1 - alpha, mu0, sigma / sqrt(n))`
5. **Decision**:
   - `decision_H0 = mean(sample_data_H0) > critical_value`
   - `decision_H1 = mean(sample_data_H1) > critical_value`
6. **Output**: Print critical value, decisions for H0 and H1
7. **Plot**: Plot density for `sample_data_H0` and `sample_data_H1` with critical value

R code:

```
mu0 <- 5 # Null hypothesis mean

mu1 <- 6 # Alternative hypothesis mean

sigma <- 2 # Population standard deviation

n <- 30 # Sample size

alpha <- 0.05 # Significance level

# Generate sample data under H0

set.seed(123)

sample_data_H0 <- rnorm(n, mean = mu0, sd = sigma)

# Generate sample data under H1

sample_data_H1 <- rnorm(n, mean = mu1, sd = sigma)

# Likelihood ratio test

likelihood_ratio <- function(data, mu0, mu1, sigma) {

 exp(sum(dnorm(data, mean = mu1, sd = sigma, log = TRUE)) -

 sum(dnorm(data, mean = mu0, sd = sigma, log = TRUE)))

}

# Critical region

critical_value <- qnorm(1 - alpha, mean = mu0, sd = sigma / sqrt(n))
```

```
# Decision
decision_H0 <- mean(sample_data_H0) > critical_value
decision_H1 <- mean(sample_data_H1) > critical_value
# Output
print(paste("Critical Value:", critical_value))
print(paste("Decision under H0:", decision_H0))
print(paste("Decision under H1:", decision_H1))
# Graphical Output
par(mfrow = c(1, 2))
# Density plot under H0
plot(density(sample_data_H0), col = "blue", lwd = 2, main = "Density under H0", xlab = "Value")
abline(v = critical_value, col = "red", lty = 2)
# Density plot under H1
plot(density(sample_data_H1), col = "green", lwd = 2, main = "Density under H1", xlab = "Value")
abline(v = critical_value, col = "red", lty = 2)
```

Sample input/output:

Input:

```
mu0 <- 5       # Null hypothesis mean
mu1 <- 6       # Alternative hypothesis mean
sigma <- 2     # Population standard deviation
n <- 30        # Sample size
alpha <- 0.05  # Significance level
```

output:

```
Critical Value: 5.643234
Decision under H0: FALSE
Decision under H1: TRUE
```

# Lab 13: Simulate Type I and Type II Errors in Hypothesis Testing

Theory

This lab simulates **Type I** and **Type II** errors in hypothesis testing by performing multiple simulations of t-tests. The goal is to estimate the error rates associated with rejecting the null hypothesis when it is true (Type I error) and failing to reject the null hypothesis when the alternative hypothesis is true (Type II error). The **power** of the test is also calculated as $1 - \text{Type II error rate}$.

Objective

- Simulate the occurrence of Type I and Type II errors in hypothesis testing.
- Estimate and display the **Type I error rate**, **Type II error rate**, and **power** of a hypothesis test.
- Visualize the distributions of p-values under the null hypothesis (H0) and alternative hypothesis (H1).

Pseudocode

```
1. Set Parameters: mu0, mu1, sigma, n, alpha, N_sim
2. Initialize Counters: type_I_errors = 0, type_II_errors = 0
3. Run Simulations (N_sim times):
    • Simulate data under H0 and perform t-test:
        • If p-value < alpha, increment type_I_errors.
    • Simulate data under H1 and perform t-test:
        • If p-value >= alpha, increment type_II_errors.
4. Output: Calculate and print Type I error rate, Type II error rate, and Power.
5. Plot:
    • Plot histograms for p-values under H0 (Type I error) and H1 (Type II error) with alpha threshold.
```

R code:

```
mu0 <- 5 # Null hypothesis mean

mu1 <- 6 # Alternative hypothesis mean

sigma <- 2 # Population standard deviation

n <- 30 # Sample size

alpha <- 0.05 # Significance level

N_sim <- 1000 # Number of simulations
```

```
# Initialize counters
type_I_errors <- 0
type_II_errors <- 0
# Simulation
set.seed(123)
for (i in 1:N_sim) {
 # Simulate data under H0
 data_H0 <- rnorm(n, mean = mu0, sd = sigma)
 t_test_H0 <- t.test(data_H0, mu = mu0, alternative = "greater")
 if (t_test_H0$p.value < alpha) {
 type_I_errors <- type_I_errors + 1
 }
 # Simulate data under H1
 data_H1 <- rnorm(n, mean = mu1, sd = sigma)
 t_test_H1 <- t.test(data_H1, mu = mu0, alternative = "greater")
 if (t_test_H1$p.value >= alpha) {
 type_II_errors <- type_II_errors + 1
 }
}
# Output
print(paste("Type I Error Rate:", type_I_errors / N_sim))
print(paste("Type II Error Rate:", type_II_errors / N_sim))
print(paste("Power:", 1 - (type_II_errors / N_sim)))
# Graphical Output
par(mfrow = c(1, 2))
# Type I Error Distribution
hist(replicate(N_sim, t.test(rnorm(n, mean = mu0, sd = sigma), mu = mu0, alternative = "greater")$p.value),
 breaks = 30, col = "lightblue", main = "P-values under H0", xlab = "P-value", border = "white")
abline(v = alpha, col = "red", lwd = 2)
# Type II Error Distribution
hist(replicate(N_sim, t.test(rnorm(n, mean = mu1, sd = sigma), mu = mu0, alternative = "greater")$p.value),
 breaks = 30, col = "lightgreen", main = "P-values under H1", xlab = "P-value", border = "white")
```

```
abline(v = alpha, col = "red", lwd = 2)
```

Sample input/output:

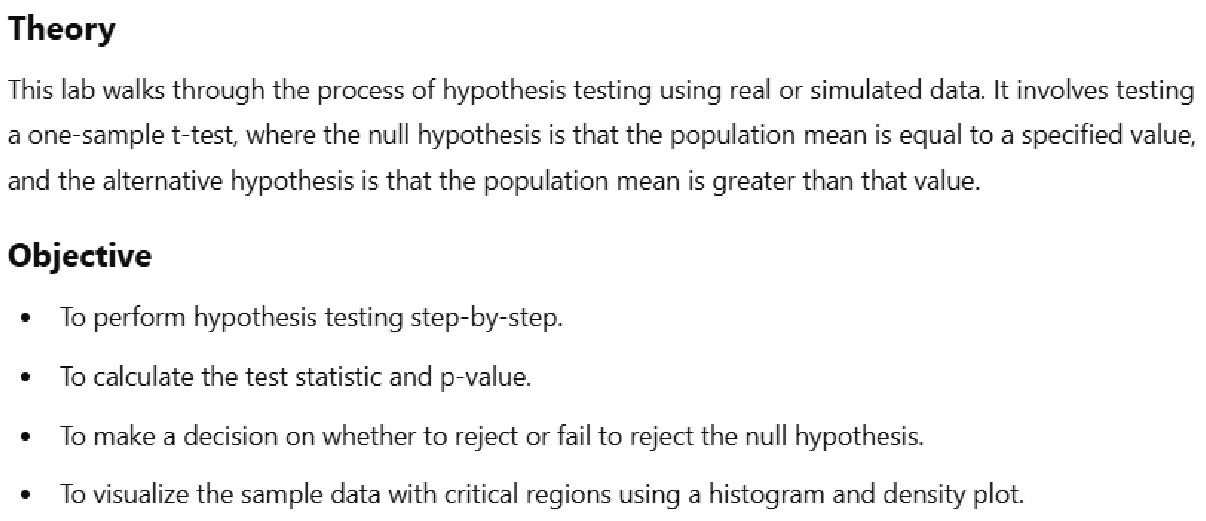
Input:

```
mu0 <- 5       # Null hypothesis mean

mu1 <- 6       # Alternative hypothesis mean

sigma <- 2      # Population standard deviation

n <- 30        # Sample size

alpha <- 0.05   # Significance level

N_sim <- 1000   # Number of simulations
```

Output:

```
Type I Error Rate: 0.039

Type II Error Rate: 0.154

Power: 0.846
```

# Lab 14: Perform Hypothesis Testing Step-by-Step Using Real or Simulated Data



Theory

This lab walks through the process of hypothesis testing using real or simulated data. It involves testing a one-sample t-test, where the null hypothesis is that the population mean is equal to a specified value, and the alternative hypothesis is that the population mean is greater than that value.

Objective

- To perform hypothesis testing step-by-step.
- To calculate the test statistic and p-value.
- To make a decision on whether to reject or fail to reject the null hypothesis.
- To visualize the sample data with critical regions using a histogram and density plot.

Pseudocode

```
1. Set Parameters: mu0, mu1, sigma, n, alpha
2. Generate Sample Data: Simulate data based on mu1.
3. State Hypotheses:
     • H0: mu = 5
     • H1: mu > 5
4. Choose Significance Level: Set alpha = 0.05.
5. Calculate Test Statistic:
     • $t = \frac{\bar{X} - \mu_0}{s/\sqrt{n}}$
6. Determine Critical Value and p-value:
     • Critical value: $t_{\alpha,df}$
     • p-value: Calculate using the pt() function.
7. Make Decision:
     • If t_stat > critical_value, reject H0.
     • Otherwise, fail to reject H0.
8. Output: Display test statistic, p-value, critical value, and decision.
9. Plot:
     • Histogram and density plot with critical region.
```

R code:

```
mu0 <- 5 # Null hypothesis mean

mu1 <- 6 # True population mean

sigma <- 2 # Population standard deviation

n <- 30 # Sample size

alpha <- 0.05 # Significance level

# Generate sample data

set.seed(123)

sample_data <- rnorm(n, mean = mu1, sd = sigma)

# Step 1: State hypotheses

print("H0: mu = 5")
print("H1: mu > 5")

# Step 2: Choose significance level

print(paste("Significance level (alpha):", alpha))

# Step 3: Calculate test statistic

t_stat <- (mean(sample_data) - mu0) / (sd(sample_data) / sqrt(n))

print(paste("Test Statistic (t):", t_stat))

# Step 4: Determine critical value or p-value

critical_value <- qt(1 - alpha, df = n-1)

p_value <- pt(t_stat, df = n-1, lower.tail = FALSE)

print(paste("Critical Value:", critical_value))

print(paste("P-value:", p_value))

# Step 5: Make a decision

if (t_stat > critical_value) {

 decision <- "Reject H0"

} else {

 decision <- "Fail to reject H0"

}

print(paste("Decision:", decision))

# Graphical Output

par(mfrow = c(1, 2))

# Histogram with critical region

hist(sample_data, breaks = 30, col = "lightblue", main = "Sample Data", xlab = "Value", border = "white")

abline(v = mu0, col = "red", lwd = 2)

abline(v = mean(sample_data), col = "blue", lwd = 2)

# Density plot with critical region

plot(density(sample_data), col = "blue", lwd = 2, main = "Density Plot", xlab = "Value")

abline(v = critical_value, col = "red", lty = 2)
```

sample input/output:

input:

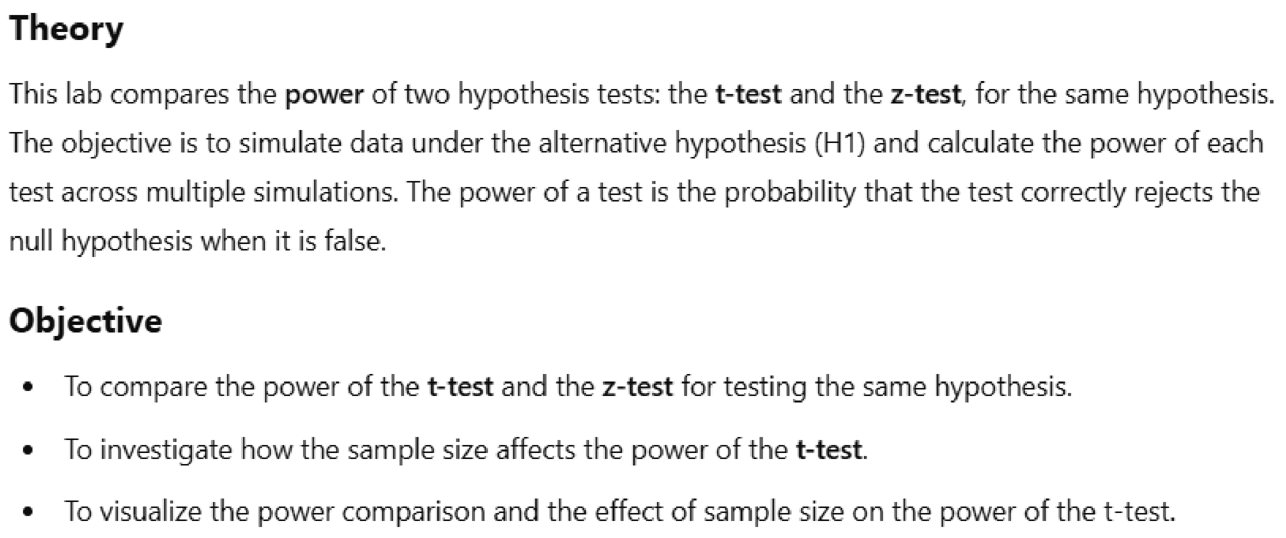
```
mu0 <- 5       # Null hypothesis mean

mu1 <- 6       # True population mean
```

```
sigma <- 2      # Population standard deviation

n <- 30         # Sample size

alpha <- 0.05   # Significance level
```

output:

```
H0: mu = 5

H1: mu > 5

Significance level (alpha): 0.05

Test Statistic (t): 2.52858027419059

Critical Value: 1.6991270265335

P-value: 0.00857566021520242

Decision: Reject H0
```

# Lab 15: Compare the Power of Different Tests for the Same Hypothesis



Theory

This lab compares the **power** of two hypothesis tests: the **t-test** and the **z-test**, for the same hypothesis. The objective is to simulate data under the alternative hypothesis (H1) and calculate the power of each test across multiple simulations. The power of a test is the probability that the test correctly rejects the null hypothesis when it is false.

Objective

- To compare the power of the **t-test** and the **z-test** for testing the same hypothesis.
- To investigate how the sample size affects the power of the **t-test**.
- To visualize the power comparison and the effect of sample size on the power of the t-test.

Pseudocode

```
1. Set Parameters: mu0, mu1, sigma, n, alpha, N_sim
2. Initialize Power Counters: power_t_test = 0, power_z_test = 0
3. Run Simulations (N_sim times):
    • Simulate data under H1.
    • Perform a t-test and check if p-value < alpha. If true, increment power_t_test.
    • Perform a z-test and check if test statistic > critical value. If true, increment power_z_test.
4. Output: Calculate and print the power of the t-test and z-test.
5. Plot:
    • Barplot comparing the power of both tests.
    • Plot the effect of sample size on the power of the t-test.
```

R code:

```
mu0 <- 5 # Null hypothesis mean
mu1 <- 6 # Alternative hypothesis mean
sigma <- 2 # Population standard deviation
n <- 30 # Sample size
alpha <- 0.05 # Significance level
N_sim <- 1000 # Number of simulations
# Initialize power counters
power_t_test <- 0
power_z_test <- 0
# Simulation
set.seed(123)
for (i in 1:N_sim) {
 # Simulate data under H1
 data <- rnorm(n, mean = mu1, sd = sigma)
 # Perform t-test
 t_test <- t.test(data, mu = mu0, alternative = "greater")
 if (t_test$p.value < alpha) {
 power_t_test <- power_t_test + 1
 }
 # Perform z-test
 z_stat <- (mean(data) - mu0) / (sigma / sqrt(n))
 z_critical <- qnorm(1 - alpha)
 if (z_stat > z_critical) {
 power_z_test <- power_z_test + 1
 }
}
# Output
print(paste("Power of t-test:", power_t_test / N_sim))
print(paste("Power of z-test:", power_z_test / N_sim))
# Graphical Output
par(mfrow = c(1, 2))
# Power comparison
barplot(c(power_t_test / N_sim, power_z_test / N_sim), names.arg = c("t-test", "z-test"), col = c("lightblue", "lightgreen"), main = "Power Comparison", ylab = "Power")
# Effect of sample size on power
sample_sizes <- seq(10, 100, by = 10)
power_t <- sapply(sample_sizes, function(n) {
 sum(replicate(N_sim, t.test(rnorm(n, mean = mu1, sd = sigma), mu = mu0, alternative = "greater")$p.value < alpha)) / N_sim
})
plot(sample_sizes, power_t, type = "b", col = "blue", main = "Power vs. Sample Size", xlab = "Sample")
```

sample input/output:

input:

```
mu0 <- 5       # Null hypothesis mean
mu1 <- 6       # Alternative hypothesis mean
sigma <- 2     # Population standard deviation
n <- 30        # Sample size
alpha <- 0.05  # Significance level
N_sim <- 1000  # Number of simulations
```

Output:

```
Power of t-test: 0.844
Power of z-test: 0.854
```

Lab 16. Apply Bartlett's test to compare variances across multiple groups.

Theory

Bartlett's test is used to test if the variances of multiple groups are equal (homogeneity of variances). It is based on the assumption that the data are normally distributed. If the p-value of the test is below the significance level, we reject the null hypothesis of equal variances.

Objective

- Perform Bartlett's test to check the homogeneity of variances across three groups.
- Visualize the data using boxplots and density plots to better understand the distribution and variability of each group.

Pseudocode

```
1. Generate Sample Data: Create three groups with different standard deviations (group1, group2,
   group3).
2. Combine Data: Create a combined data frame for performing Bartlett's test.
3. Perform Bartlett's Test:
    • Test the null hypothesis that the variances are equal across the groups.
4. Output: Print the result of Bartlett's test.
5. Plot:
    • Create a boxplot to visualize the spread and variability across groups.
    • Create density plots to compare the distributions of the groups.
    • Add a legend to distinguish between groups.
```

R code:

```
set.seed(123)

group1 <- rnorm(30, mean = 5, sd = 2)

group2 <- rnorm(30, mean = 5, sd = 3)

group3 <- rnorm(30, mean = 5, sd = 4)


# Combine data into a data frame for Bartlett's test

data_values <- c(group1, group2, group3)

group_labels <- rep(c("Group 1", "Group 2", "Group 3"), each = 30)
```

```r
# Perform Bartlett's test
bartlett_test_result <- bartlett.test(data_values ~ group_labels)

# Output test result
print(bartlett_test_result)

# Graphical Output
par(mfrow = c(1, 2))

# Boxplot of groups
boxplot(group1, group2, group3,
        col = c("lightblue", "lightgreen", "lightcoral"),
        main = "Boxplot of Groups",
        names = c("Group 1", "Group 2", "Group 3"),
        xlab = "Group", ylab = "Value")

# Density plots of groups
plot(density(group1), col = "blue", lwd = 2, main = "Density Plots", xlab = "Value", ylim = c(0, 0.25))
lines(density(group2), col = "green", lwd = 2)
lines(density(group3), col = "red", lwd = 2)

# Add legend
legend("topright", legend = c("Group 1", "Group 2", "Group 3"),
       col = c("blue", "green", "red"), lwd = 2)
```

Sample Input

```r
set.seed(123)
group1 <- rnorm(30, mean = 5, sd = 2)
group2 <- rnorm(30, mean = 5, sd = 3)
group3 <- rnorm(30, mean = 5, sd = 4)
```

output:

```
Bartlett test of homogeneity of variances

data:  data_values by group_labels

Bartlett's K-squared = 9.4313, df = 2, p-value = 0.008954
```

# Lab 17: Perform Fisher's Exact Test on 2×2 Contingency Tables

Theory

Fisher's Exact Test is used to determine if there are nonrandom associations between two categorical variables in a **2x2 contingency table**. It calculates the exact probability of obtaining a distribution of values in a table given the marginal sums, making it particularly useful when sample sizes are small.

Objective

- Perform Fisher's Exact Test on a **2x2 contingency table** to assess if there is a significant relationship between two categorical variables.
- Visualize the data using **barplots** and **mosaic plots** to gain further insights into the distribution of the variables.

Pseudocode

```
1. Create a 2x2 Contingency Table with values for two variables (e.g., Group A vs Group B, Success vs Failure).
2. Perform Fisher's Exact Test on the contingency table to test for associations.
3. Output:
    o Print the contingency table and Fisher's test result.
4. Graphical Output:
    o Create a barplot to visualize the counts for each outcome.
    o Create a mosaic plot to visualize the relationship between the two categorical variables.
```

R code:

```
# Create a 2x2 contingency table

data <- matrix(c(10, 5, 2, 8), nrow = 2, byrow = TRUE)

rownames(data) <- c("Group A", "Group B")

colnames(data) <- c("Success", "Failure")

# Perform Fisher's exact test

fisher_test_result <- fisher.test(data)

# Output

print(data)

print(fisher_test_result)
```

```
# Graphical Output

par(mfrow = c(1, 2))

# Barplot of the contingency table

barplot(data, beside = TRUE, col = c("lightblue", "lightgreen"), main = "2x2 Contingency Table", xlab = "Outcome", ylab =
"Count")

legend("topright", legend = rownames(data), fill = c("lightblue", "lightgreen"))

# Mosaic plot

mosaicplot(data, main = "Mosaic Plot", color = TRUE)
```

Sample Input

```
data <- matrix(c(10, 5, 2, 8), nrow = 2, byrow = TRUE)

rownames(data) <- c("Group A", "Group B")

colnames(data) <- c("Success", "Failure")
```

sample output:

```
        Success Failure
Group A      10       5
Group B       2       8


        Fisher's Exact Test for Count Data


data:  data
p-value = 0.04141
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
  0.9540368 96.2686947
sample estimates:
odds ratio
  7.281573
```

Lab 18. Analyze three-way contingency tables using log-linear models.

Theory

A **three-way contingency table** is used to examine the relationship between three categorical variables. A **log-linear model** is often used to assess interactions between these variables. The log-linear model fits a statistical model to the data by modeling the log of expected cell counts as a linear combination of the variables' effects and their interactions.

Objective

- Create a **three-way contingency table** with categorical variables (Gender, Treatment, Outcome).
- Fit a **log-linear model** to assess interactions between these variables.
- Visualize the data using a **heatmap** and an **interaction plot** to understand relationships between the variables.

Pseudocode

```
1. Create a 3D contingency table with dimensions for Gender, Treatment, and Outcome.
2. Fit a log-linear model to the table to identify the relationships between the variables.
3. Output:
     o Print the three-way contingency table and the results of the log-linear model.
4. Graphical Output:
     o Create a heatmap to visualize the joint distribution of Gender and Treatment, summing over Outcome.
     o Create an interaction plot to visualize the counts of outcomes by Treatment and Gender.
```

R code:

```
# Create a three-way contingency table

data <- array(c(10, 5, 2, 8, 3, 6, 4, 7), dim = c(2, 2, 2))

dimnames(data) <- list(Gender = c("Male", "Female"),

               Treatment = c("Yes", "No"),

               Outcome = c("Success", "Failure"))


# Fit a log-linear model

log_linear_model <- loglin(data, margin = list(1, 2, 3), fit = TRUE)


# Output

print("Three-Way Contingency Table:")

print(data)

print("Log-Linear Model Output:")

print(log_linear_model)
```

```
# Graphical Output
par(mfrow = c(1, 2))

# Convert to 2D matrix for heatmap
heatmap_matrix <- apply(data, c(1, 2), sum)  # Summing over Outcome
heatmap(heatmap_matrix, main = "Heatmap of Gender vs Treatment", col = heat.colors(100))

# Convert data into vectors for interaction plot
Gender <- rep(c("Male", "Female"), times = 4)
Treatment <- rep(c("Yes", "No"), each = 2, times = 2)
Outcome <- rep(c("Success", "Failure"), each = 4)
Counts <- as.vector(data)

# Create an interaction plot
interaction.plot(Treatment, Gender, Counts,
        main = "Interaction Plot: Treatment & Gender",
        xlab = "Treatment", ylab = "Counts", col = c("red", "blue"), lwd = 2)
```

Sample Input

```
data <- array(c(10, 5, 2, 8, 3, 6, 4, 7), dim = c(2, 2, 2))
dimnames(data) <- list(Gender = c("Male", "Female"),
           Treatment = c("Yes", "No"),
           Outcome = c("Success", "Failure"))
```

Output:

```
Three-Way Contingency Table:
, , Outcome = Success

    Treatment
Gender   Yes No
 Male    10  5
 Female   2  8
```

```
, , Outcome = Failure

       Treatment
Gender    Yes No
  Male     3  6
  Female   4  7
```

Lab 19: Conduct Non-Parametric Tests

Theory

Non-parametric tests are statistical methods that do not assume a specific distribution for the data. The **Wilcoxon rank-sum test** (also known as the **Mann-Whitney U test**) is a non-parametric test used to compare two independent samples to determine if one tends to have higher values than the other.

Objective

- Generate two independent groups of data.
- Conduct the **Wilcoxon rank-sum test** to compare the distributions of the two groups.
- Visualize the data using **boxplots** and **density plots**.

Pseudocode

1. **Generate random sample data** for two independent groups.
2. **Perform Wilcoxon rank-sum test** to check if there is a significant difference between the two groups.
3. **Output** the test results.
4. **Graphical Output**:
    - **Boxplot** to compare distributions of the two groups.
    - **Density plot** to visualize the probability distributions of the two groups.

R code:

```
# Generate sample data

set.seed(123)

group1 <- rnorm(20, mean = 5, sd = 2)

group2 <- rnorm(20, mean = 7, sd = 2)

# Perform Wilcoxon rank-sum test

wilcox_test_result <- wilcox.test(group1, group2)

# Output
```

```
print(wilcox_test_result)

# Graphical Output

par(mfrow = c(1, 2))

# Boxplot of groups

boxplot(list(group1, group2), col = c("lightblue", "lightgreen"), main = "Boxplot of Groups", xlab = "Group", ylab =
"Value")

# Density plots of groups

plot(density(group1), col = "blue", lwd = 2, main = "Density Plots", xlab = "Value", ylim = c(0, 0.25))

lines(density(group2), col = "green", lwd = 2)

legend("topright", legend = c("Group 1", "Group 2"), col = c("blue", "green"), lwd = 2)
```

Sample Input

```
group1 <- rnorm(20, mean = 5, sd = 2)

group2 <- rnorm(20, mean = 7, sd = 2)
```

Sample Output

```
Wilcoxon rank sum exact test

data:  group1 and group2

W = 104, p-value = 0.008712

alternative hypothesis: true location shift is not equal to 0
```

# Lab 20: Perform z-Tests for Large Sample Sizes

Theory

A **z-test** is a statistical test used to compare the means of two large samples (typically n>30n > 30n>30) when the population standard deviations are known. It assumes that the data follows a **normal distribution**.

Objective

- Generate two independent large sample groups.
- Perform a **z-test** to determine if there is a significant difference between the means of the two groups.
- Visualize the data distribution using histograms.

Pseudocode

1. **Import necessary library** (BSDA for z-test).
2. **Generate random sample data** for two groups.
3. **Perform z-test**:

   - Compare the means of the two groups.
   - Use the **standard deviation** of each group.
4. **Output test results** (z-statistic and p-value).
5. **Graphical Output**:
   - **Histogram** for Group 1.
   - **Histogram** for Group 2.

R code:

```
# Generate sample data

library(BSDA)

set.seed(123)

group1 <- rnorm(100, mean = 5, sd = 2)

group2 <- rnorm(100, mean = 6, sd = 2)

# Perform z-test

z_test_result <- z.test(group1, group2, sigma.x = sd(group1), sigma.y = sd(group2))

# Output

print(z_test_result)

# Graphical Output

par(mfrow = c(1, 2))

# Histogram of groups

hist(group1, breaks = 30, col = "lightblue", main = "Histogram of Group 1", xlab = "Value", border = "white")

hist(group2, breaks = 30, col = "lightgreen", main = "Histogram of Group 2", xlab = "Value", border = "white")
```

Sample Input

```
group1 <- rnorm(100, mean = 5, sd = 2)

group2 <- rnorm(100, mean = 6, sd = 2)
```

Sample Output:

```
	Two-sample z-Test

data:  group1 and group2

z = -2.2714, p-value = 0.02312

alternative hypothesis: true difference in means is not equal to 0
```

```
95 percent confidence interval:

 -1.12535598 -0.08283319

sample estimates:

mean of x mean of y

 5.180812  5.784906
```