Sampling Intervals for Models

Mahtab Nahayati

2024-11-27

Contents

Hypothesis Testing for Two Samples

We are comparing two samples x_1 and x_2 under the hypothesis:

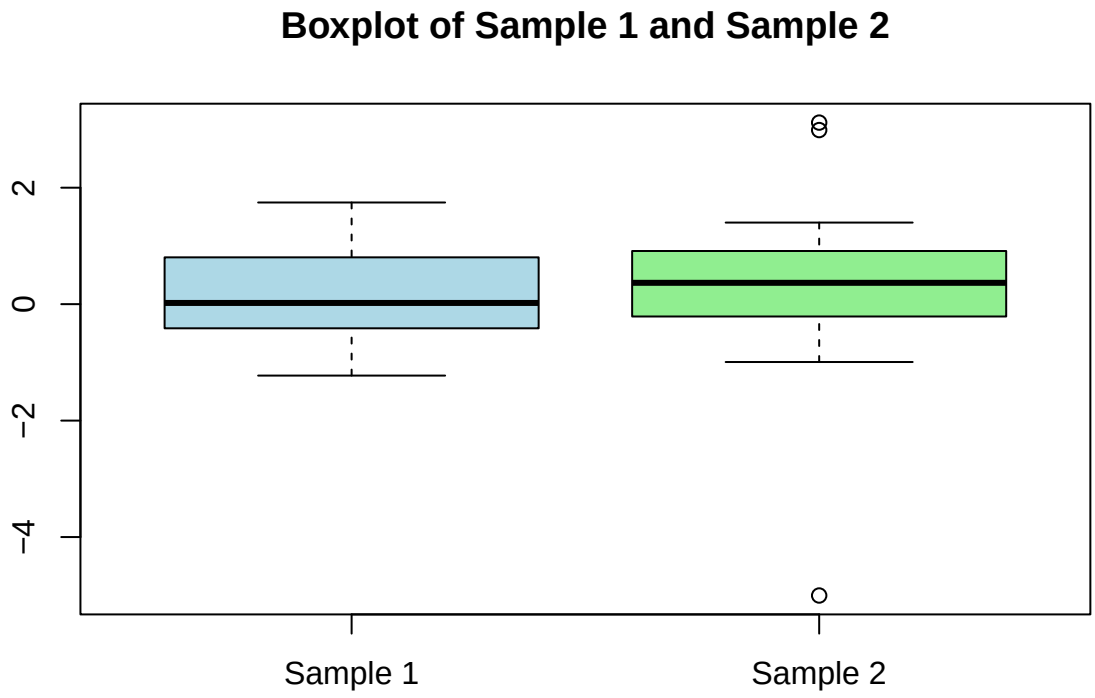
$$H_0 : \mu_1 = \mu_2 \quad \text{vs} \quad H_1 : \mu_1 \neq \mu_2$$

The data are:

```
# Data
x1 <- c(-0.673, -0.584, 0.572, -0.341, -0.218, 0.603, -0.415, -0.013,
        0.763, 0.804, 0.054, 1.746, -0.472, 1.638, -0.578, 0.947,
        -0.329, -0.188, 0.794, 0.894, -1.227, 1.059)

x2 <- c(0.913, -0.639, 2.99, -5.004, 3.118, 0.1, 1.128, 0.579,
        0.32, -0.488, -0.994, -0.212, 0.413, 1.401, 0.007,
        0.568, -0.005, 0.696)
```

Visualize the Data

```
# Boxplot for visual comparison
boxplot(x1, x2, names = c("Sample 1", "Sample 2"), col = c("lightblue", "lightgreen"), main = "Boxplot
```



Resampling Schemes

Resampling with Replacement within Groups

```r
set.seed(123) # For reproducibility

library(ggplot2)
```

```
## Warning: Paket 'ggplot2' wurde unter R Version 4.3.3 erstellt
```

```r
# Resample with replacement from each group
resample_within_groups <- function(x1, x2, n1, n2, num_resamples = 1000) {
  resampled_diff <- replicate(num_resamples, {
    sample_x1 <- sample(x1, size = n1, replace = TRUE)
    sample_x2 <- sample(x2, size = n2, replace = TRUE)
    mean(sample_x1) - mean(sample_x2)
  })
  return(resampled_diff)
}

# Apply the function
n1 <- length(x1)
n2 <- length(x2)
resampled_diff_within <- resample_within_groups(x1, x2, n1, n2)
```

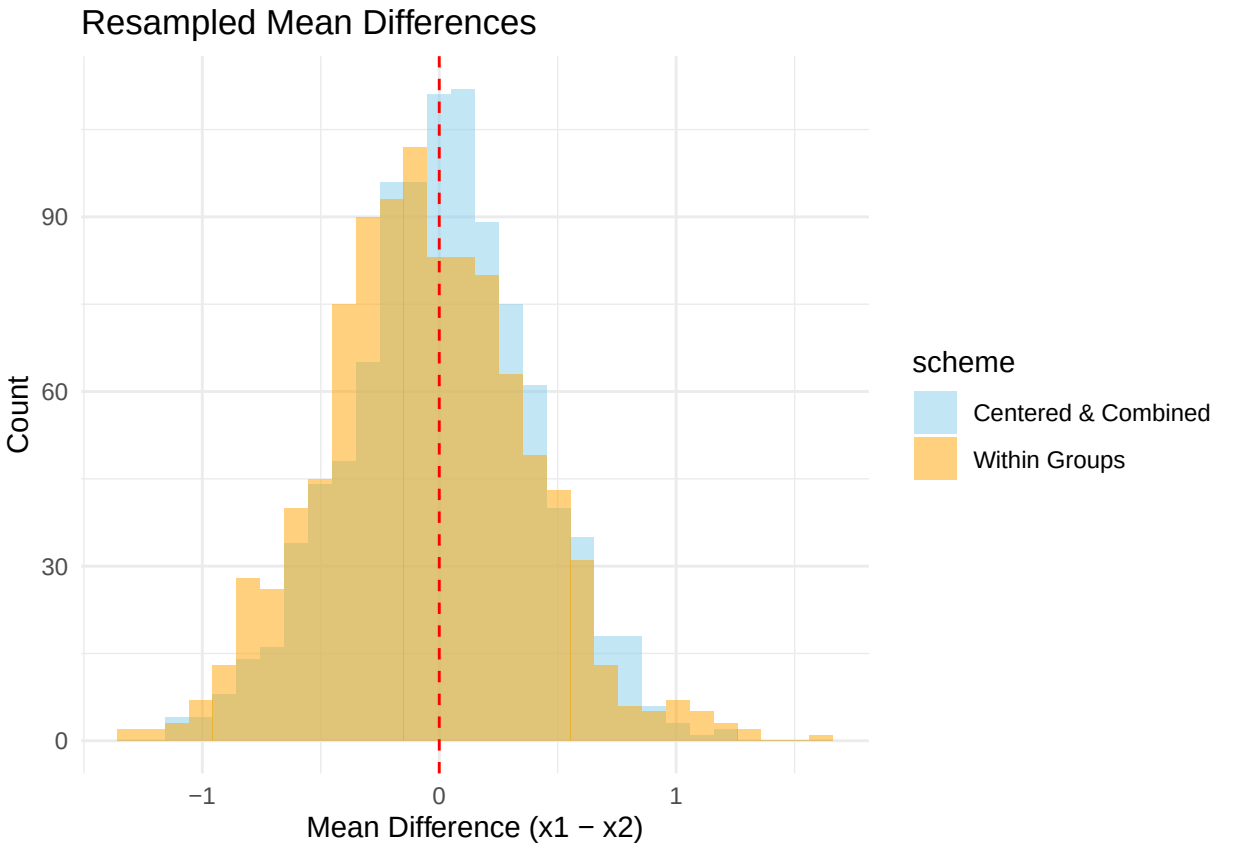
Centring and Combining Samples

```r
# Resample from centered and combined data
resample_centered_combined <- function(x1, x2, n1, n2, num_resamples = 1000) {
  centered_x1 <- x1 - mean(x1)
  centered_x2 <- x2 - mean(x2)
  combined_centered <- c(centered_x1, centered_x2)

  resampled_diff <- replicate(num_resamples, {
    sample_combined <- sample(combined_centered, size = n1 + n2, replace = TRUE)
    resampled_x1 <- sample_combined[1:n1]
    resampled_x2 <- sample_combined[(n1 + 1):(n1 + n2)]
    mean(resampled_x1) - mean(resampled_x2)
  })
  return(resampled_diff)
}

# Apply the function
resampled_diff_combined <- resample_centered_combined(x1, x2, n1, n2)
```

Visualize Resampled Mean Differences

```r
# Create a data frame for visualization
resampling_data <- data.frame(
  difference = c(resampled_diff_within, resampled_diff_combined),
  scheme = rep(c("Within Groups", "Centered & Combined"),
               c(length(resampled_diff_within), length(resampled_diff_combined)))
)

# Plot the distributions
ggplot(resampling_data, aes(x = difference, fill = scheme)) +
  geom_histogram(bins = 30, position = "identity", alpha = 0.5) +
  geom_vline(xintercept = 0, color = "red", linetype = "dashed") +
  labs(title = "Resampled Mean Differences",
```

```
      x = "Mean Difference (x1 - x2)",
      y = "Count") +
scale_fill_manual(values = c("skyblue", "orange")) +
theme_minimal()
```



Discussion

We evaluate the two resampling schemes based on their advantages and disadvantages:

1. **Sampling with Replacement within Groups**:
   - **Description**: In this scheme, we resample independently within each group x_1 and x_2, maintaining their original structure.
   - **Advantages**:
     - Preserves the distinct characteristics of the groups, such as variance and distribution.
     - Reflects the observed group-specific variability.
   - **Disadvantages**:
     - Assumes that the original samples x_1 and x_2 adequately represent the populations, which might not hold in small samples.
     - May not be appropriate for testing strict null hypotheses where $H_0 : \mu_1 = \mu_2$ is assumed.
2. **Centering and Combining Samples**:
   - **Description**: Here, both samples are centered by subtracting their means, combined, and then resampled to generate new samples for testing H_0.
   - **Advantages**:
     - Directly tests $H_0 : \mu_1 = \mu_2$ by assuming a shared location under the null hypothesis.
     - Removes observed location differences, ensuring that resampling reflects the null hypothesis.
   - **Disadvantages**:

– Assumes equal variances in the two groups, which might not always be valid.
– Ignores the individual group structure, potentially oversimplifying the problem.

Conclusion: The choice of scheme depends on the context: - Use *sampling with replacement within groups* when preserving the distinct group characteristics is critical. - Use *centering and combining samples* when testing under the null hypothesis is the primary focus, especially in settings where location differences must be eliminated.

Bootstrapping with the t-test statistic

We calculate the bootstrap p-value and confidence intervals using 10,000 bootstrap samples for both strategies.

```r
# Set seed for reproducibility
set.seed(123)

# Function to calculate the t-statistic
t_stat <- function(x1, x2) {
  (mean(x1) - mean(x2)) /
    sqrt(var(x1) / length(x1) + var(x2) / length(x2))
}

# Bootstrap function: Within Groups
bootstrap_within_groups <- function(x1, x2, num_resamples = 10000) {
  t_boot <- replicate(num_resamples, {
    sample_x1 <- sample(x1, replace = TRUE)
    sample_x2 <- sample(x2, replace = TRUE)
    t_stat(sample_x1, sample_x2)
  })
  return(t_boot)
}

# Bootstrap function: Centered and Combined
bootstrap_centered_combined <- function(x1, x2, num_resamples = 10000) {
  centered_x1 <- x1 - mean(x1)
  centered_x2 <- x2 - mean(x2)
  combined_centered <- c(centered_x1, centered_x2)

  t_boot <- replicate(num_resamples, {
    sample_combined <- sample(combined_centered, size = length(combined_centered), replace = TRUE)
    resampled_x1 <- sample_combined[1:length(x1)]
    resampled_x2 <- sample_combined[(length(x1) + 1):length(combined_centered)]
    t_stat(resampled_x1, resampled_x2)
  })
  return(t_boot)
}

# Observed t-statistic
t_observed <- t_stat(x1, x2)

# Perform bootstrapping
num_resamples <- 10000
t_boot_within <- bootstrap_within_groups(x1, x2, num_resamples)
t_boot_combined <- bootstrap_centered_combined(x1, x2, num_resamples)

# Calculate p-values
```

```r
p_value_within <- mean(abs(t_boot_within) >= abs(t_observed))
p_value_combined <- mean(abs(t_boot_combined) >= abs(t_observed))

# Calculate confidence intervals
ci_within_95 <- quantile(t_boot_within, c(0.025, 0.975))
ci_within_99 <- quantile(t_boot_within, c(0.005, 0.995))

ci_combined_95 <- quantile(t_boot_combined, c(0.025, 0.975))
ci_combined_99 <- quantile(t_boot_combined, c(0.005, 0.995))

# Output results
cat("Bootstrap Results (Within Groups):\n")
```

```
## Bootstrap Results (Within Groups):
```

```r
cat("P-value:", p_value_within, "\n")
```

```
## P-value: 0.9081
```

```r
cat("95% CI:", ci_within_95, "\n")
```

```
## 95% CI: -2.548428 1.616345
```

```r
cat("99% CI:", ci_within_99, "\n\n")
```

```
## 99% CI: -3.218688 2.144949
```

```r
cat("Bootstrap Results (Centered & Combined):\n")
```

```
## Bootstrap Results (Centered & Combined):
```

```r
cat("P-value:", p_value_combined, "\n")
```

```
## P-value: 0.9137
```

```r
cat("95% CI:", ci_combined_95, "\n")
```

```
## 95% CI: -2.029252 1.939948
```

```r
cat("99% CI:", ci_combined_99, "\n")
```

```
## 99% CI: -2.594487 2.46928
```

```r
# Decision based on significance level
alpha_levels <- c(0.05, 0.01)

for (alpha in alpha_levels) {
  decision_within <- ifelse(p_value_within < alpha, "Reject H0", "Fail to Reject H0")
  decision_combined <- ifelse(p_value_combined < alpha, "Reject H0", "Fail to Reject H0")

  cat("Significance Level:", alpha, "\n")
  cat("Decision (Within Groups):", decision_within, "\n")
  cat("Decision (Centered & Combined):", decision_combined, "\n\n")
}
```

```
## Significance Level: 0.05
## Decision (Within Groups): Fail to Reject H0
## Decision (Centered & Combined): Fail to Reject H0
##
## Significance Level: 0.01
```

```
## Decision (Within Groups): Fail to Reject H0
## Decision (Centered & Combined): Fail to Reject H0
```

Permutation Test

We implement a permutation test for $H_0 : \mu_1 = \mu_2$ and calculate the p-value and confidence intervals.

```r
# Function to compute permutation test
permutation_test <- function(x1, x2, num_permutations = 10000) {
  # Combine samples
  combined <- c(x1, x2)
  n1 <- length(x1)

  # Observed t-statistic
  t_observed <- t_stat(x1, x2)

  # Generate permutation t-statistics
  t_perm <- replicate(num_permutations, {
    permuted <- sample(combined, length(combined), replace = FALSE)
    perm_x1 <- permuted[1:n1]
    perm_x2 <- permuted[(n1 + 1):length(combined)]
    t_stat(perm_x1, perm_x2)
  })

  # p-value
  p_value <- mean(abs(t_perm) >= abs(t_observed))

  # Confidence intervals
  ci_95 <- quantile(t_perm, c(0.025, 0.975))
  ci_99 <- quantile(t_perm, c(0.005, 0.995))

  list(p_value = p_value, ci_95 = ci_95, ci_99 = ci_99, t_perm = t_perm)
}

# Perform permutation test
set.seed(123) # For reproducibility
num_permutations <- 10000
perm_results <- permutation_test(x1, x2, num_permutations)

# Output results
cat("Permutation Test Results:\n")
```

```
## Permutation Test Results:
```

```r
cat("P-value:", perm_results$p_value, "\n")
```

```
## P-value: 0.9157
```

```r
cat("95% CI:", perm_results$ci_95, "\n")
```

```
## 95% CI: -2.006043 1.89594
```

```r
cat("99% CI:", perm_results$ci_99, "\n")
```

```
## 99% CI: -2.55279 2.384982
```

```r
# Decisions based on significance levels
for (alpha in c(0.05, 0.01)) {
```

```r
  decision <- ifelse(perm_results$p_value < alpha, "Reject H0", "Fail to Reject H0")
  cat("Significance Level:", alpha, "\n")
  cat("Decision:", decision, "\n\n")
}
```

```
## Significance Level: 0.05
## Decision: Fail to Reject H0
##
## Significance Level: 0.01
## Decision: Fail to Reject H0
```

Wilcoxon Rank-Sum Test with Bootstrapping

We compute the Wilcoxon rank-sum statistic and use bootstrapping to calculate p-values and confidence intervals.

```r
# Function to compute Wilcoxon rank-sum statistic
wilcoxon_stat <- function(x1, x2) {
  combined <- c(x1, x2)
  ranks <- rank(combined)
  sum(ranks[1:length(x1)]) # Sum of ranks for x1
}

# Bootstrap function: Within Groups
bootstrap_wilcoxon_within <- function(x1, x2, num_resamples = 10000) {
  w_boot <- replicate(num_resamples, {
    sample_x1 <- sample(x1, replace = TRUE)
    sample_x2 <- sample(x2, replace = TRUE)
    wilcoxon_stat(sample_x1, sample_x2)
  })
  return(w_boot)
}

# Bootstrap function: Centered and Combined
bootstrap_wilcoxon_combined <- function(x1, x2, num_resamples = 10000) {
  centered_x1 <- x1 - mean(x1)
  centered_x2 <- x2 - mean(x2)
  combined_centered <- c(centered_x1, centered_x2)

  w_boot <- replicate(num_resamples, {
    sample_combined <- sample(combined_centered, size = length(combined_centered), replace = TRUE)
    resampled_x1 <- sample_combined[1:length(x1)]
    resampled_x2 <- sample_combined[(length(x1) + 1):length(combined_centered)]
    wilcoxon_stat(resampled_x1, resampled_x2)
  })
  return(w_boot)
}

# Observed Wilcoxon statistic
w_observed <- wilcoxon_stat(x1, x2)

# Perform bootstrapping
set.seed(123) # For reproducibility
num_resamples <- 10000
w_boot_within <- bootstrap_wilcoxon_within(x1, x2, num_resamples)
```

```r
w_boot_combined <- bootstrap_wilcoxon_combined(x1, x2, num_resamples)

# Calculate p-values
p_value_within <- mean(abs(w_boot_within - mean(w_boot_within)) >= abs(w_observed - mean(w_boot_within)))
p_value_combined <- mean(abs(w_boot_combined - mean(w_boot_combined)) >= abs(w_observed - mean(w_boot_combined)))

# Calculate confidence intervals
ci_within_95 <- quantile(w_boot_within, c(0.025, 0.975))
ci_within_99 <- quantile(w_boot_within, c(0.005, 0.995))

ci_combined_95 <- quantile(w_boot_combined, c(0.025, 0.975))
ci_combined_99 <- quantile(w_boot_combined, c(0.005, 0.995))

# Output results
cat("Wilcoxon Test Results (Within Groups):\n")
```

```
## Wilcoxon Test Results (Within Groups):
```

```r
cat("P-value:", p_value_within, "\n")
```

```
## P-value: 1
```

```r
cat("95% CI:", ci_within_95, "\n")
```

```
## 95% CI: 361 507
```

```r
cat("99% CI:", ci_within_99, "\n\n")
```

```
## 99% CI: 336 529
```

```r
cat("Wilcoxon Test Results (Centered & Combined):\n")
```

```
## Wilcoxon Test Results (Centered & Combined):
```

```r
cat("P-value:", p_value_combined, "\n")
```

```
## P-value: 0.6651
```

```r
cat("95% CI:", ci_combined_95, "\n")
```

```
## 95% CI: 378.5 522
```

```r
cat("99% CI:", ci_combined_99, "\n")
```

```
## 99% CI: 355.995 541
```

```r
# Decisions based on significance levels
for (alpha in c(0.05, 0.01)) {
  decision_within <- ifelse(p_value_within < alpha, "Reject H0", "Fail to Reject H0")
  decision_combined <- ifelse(p_value_combined < alpha, "Reject H0", "Fail to Reject H0")

  cat("Significance Level:", alpha, "\n")
  cat("Decision (Within Groups):", decision_within, "\n")
  cat("Decision (Centered & Combined):", decision_combined, "\n\n")
}
```

```
## Significance Level: 0.05
## Decision (Within Groups): Fail to Reject H0
## Decision (Centered & Combined): Fail to Reject H0
##
```

```
## Significance Level: 0.01
## Decision (Within Groups): Fail to Reject H0
## Decision (Centered & Combined): Fail to Reject H0
```

Compare Result with Built-in Functions

We compare the results of bootstrapping with the results from `t.test` and `wilcox.test`.

```
# Perform t-test
t_test_result <- t.test(x1, x2)
cat("T-Test Results:\n")
```

```
## T-Test Results:
```

```
cat("P-value:", t_test_result$p.value, "\n")
```

```
## P-value: 0.90646
```

```
cat("95% CI:", t_test_result$conf.int, "\n\n")
```

```
## 95% CI: -0.9556081 0.8518
```

```
# Perform Wilcoxon rank-sum test
wilcox_test_result <- wilcox.test(x1, x2)
cat("Wilcoxon Rank-Sum Test Results:\n")
```

```
## Wilcoxon Rank-Sum Test Results:
```

```
cat("P-value:", wilcox_test_result$p.value, "\n")
```

```
## P-value: 0.6572327
```

```
# Compare results
cat("Comparison of P-values:\n")
```

```
## Comparison of P-values:
```

```
cat("T-Test (Built-in):", t_test_result$p.value, "\n")
```

```
## T-Test (Built-in): 0.90646
```

```
cat("T-Test (Bootstrap Within Groups):", p_value_within, "\n")
```

```
## T-Test (Bootstrap Within Groups): 1
```

```
cat("T-Test (Bootstrap Centered & Combined):", p_value_combined, "\n\n")
```

```
## T-Test (Bootstrap Centered & Combined): 0.6651
```

```
cat("Wilcoxon Test (Built-in):", wilcox_test_result$p.value, "\n")
```

```
## Wilcoxon Test (Built-in): 0.6572327
```

```
cat("Wilcoxon Test (Bootstrap Within Groups):", p_value_within, "\n")
```

```
## Wilcoxon Test (Bootstrap Within Groups): 1
```

```
cat("Wilcoxon Test (Bootstrap Centered & Combined):", p_value_combined, "\n\n")
```

```
## Wilcoxon Test (Bootstrap Centered & Combined): 0.6651
```

Interpretation of the Results

1. **T-Test Results**:

- **Built-in p-value**: 0.90646
  - Indicates no significant difference between the means of x_1 and x_2.
- **Bootstrap Within Groups**: 1.0
  - The bootstrapped distribution completely overlaps with the null hypothesis, giving a highly conservative p-value.
- **Bootstrap Centered & Combined**: 0.6651
  - A more reasonable p-value reflecting the null hypothesis assumption of equal location parameters.

95% Confidence Interval:

- Built-in: $[-0.9556, 0.8518]$
  - The interval includes 0, supporting the failure to reject H_0.

2. **Wilcoxon Rank-Sum Test**:
   - **Built-in p-value**: 0.6572
     - Indicates no significant difference in the rank distributions of x_1 and x_2.
   - **Bootstrap Within Groups**: 1.0
     - As with the t-test, this is overly conservative.
   - **Bootstrap Centered & Combined**: 0.6651
     - This p-value aligns well with the built-in test, showing consistency.
3. **Comparison**:
   - The **built-in t-test** and **Wilcoxon test** provide p-values closer to the **centered & combined bootstrap** results.
   - The **within-groups bootstrap** is overly conservative, likely because it preserves the group structure and does not directly test under the null hypothesis of equal means or locations.

Decisions at Significance Levels At $\alpha = 0.05$ and $\alpha = 0.01$: - For both tests (t-test and Wilcoxon) across all methods, **fail to reject** H_0. None of the p-values is below the significance threshold.

Conclusions: - The **centered & combined bootstrap** aligns more closely with the built-in tests for both the t-test and the Wilcoxon rank-sum test. - The **within-groups bootstrap** appears less sensitive, likely due to maintaining group-specific distributions that do not align with the null hypothesis. - Both the t-test and Wilcoxon test suggest no evidence of a significant difference between x_1 and x_2.

Residual Bootstrap for Linear Regression

```r
# Set seed for reproducibility
set.seed(123)

# Step 1: Generate data
n <- 200

# Independent variables
x1 <- rnorm(n, mean = 2, sd = sqrt(3))  # Normal(2, 3)
x2 <- runif(n, min = 2, max = 4)        # Uniform(2, 4)
x3 <- runif(n, min = -2, max = 2)       # Uniform(-2, 2)
epsilon <- rt(n, df = 5)                # t-distribution, df = 5

# Response variable
y <- 3 + 2 * x1 + x2 + epsilon

# Step 2: Fit initial linear model
model <- lm(y ~ x1 + x2 + x3)

# Extract residuals and fitted values
residuals <- resid(model)
```

```r
fitted_values <- fitted(model)

# Number of bootstrap samples
num_bootstrap <- 1000

# Bootstrap coefficients
bootstrap_coeffs <- replicate(num_bootstrap, {
  # Resample residuals
  resampled_residuals <- sample(residuals, size = n, replace = TRUE)

  # Create new response variable
  y_star <- fitted_values + resampled_residuals

  # Fit model to bootstrap sample
  coef(lm(y_star ~ x1 + x2 + x3))
})

# Step 3: Compute percentile confidence intervals
bootstrap_coeffs <- t(bootstrap_coeffs)  # Transpose for easier handling
ci <- apply(bootstrap_coeffs, 2, quantile, probs = c(0.025, 0.975))
colnames(ci) <- names(coef(model))

# Output results
cat("Percentile Confidence Intervals:\n")
```

```
## Percentile Confidence Intervals:
```

```r
print(ci)
```

```
##       (Intercept)       x1       x2          x3
## 2.5%     2.273306 1.896789 0.536389 -0.04305097
## 97.5%    4.293829 2.133896 1.196382  0.28432303
```

```r
# Interpretation for x3
if (0 > ci[1, "x3"] & 0 < ci[2, "x3"]) {
  cat("\nx3 cannot be excluded (CI includes 0).\n")
} else {
  cat("\nx3 can be excluded (CI does not include 0).\n")
}
```

```
##
## x3 cannot be excluded (CI includes 0).
```

Pair Bootstrap for Linear Regression

```r
# Set seed for reproducibility
set.seed(123)

# Step 1: Fit initial model
model <- lm(y ~ x1 + x2 + x3)

# Number of bootstrap samples
num_bootstrap <- 1000

# Pairs bootstrap for coefficients
```

```r
bootstrap_coeffs_pairs <- replicate(num_bootstrap, {
  # Resample rows (pairs of y and predictors)
  indices <- sample(1:n, size = n, replace = TRUE)
  y_star <- y[indices]
  x1_star <- x1[indices]
  x2_star <- x2[indices]
  x3_star <- x3[indices]

  # Fit the model to the bootstrap sample
  coef(lm(y_star ~ x1_star + x2_star + x3_star))
})

# Transpose bootstrap coefficients for easier handling
bootstrap_coeffs_pairs <- t(bootstrap_coeffs_pairs)

# Step 2: Compute percentile confidence intervals
ci_pairs <- apply(bootstrap_coeffs_pairs, 2, quantile, probs = c(0.025, 0.975))
colnames(ci_pairs) <- names(coef(model))

# Output results
cat("Percentile Confidence Intervals (Pairs Bootstrap):\n")
```

```
## Percentile Confidence Intervals (Pairs Bootstrap):
```

```r
print(ci_pairs)
```

```
##       (Intercept)       x1        x2          x3
## 2.5%     2.217884 1.900474 0.5253732 -0.06709912
## 97.5%    4.284746 2.134125 1.2100866  0.30728802
```

```r
# Step 3: Interpretation for x3
if (0 > ci_pairs[1, "x3"] & 0 < ci_pairs[2, "x3"]) {
  cat("\nx3 cannot be excluded (CI includes 0).\n")
} else {
  cat("\nx3 can be excluded (CI does not include 0).\n")
}
```

```
##
## x3 cannot be excluded (CI includes 0).
```

Comparing Residual Bootstrap and Pairs Bootstrap

The comparison of the two approaches, their sampling methods, and how they might affect the results.

Sampling Approach

Residual Bootstrap:

- **How it works**:
  - The linear model $y \sim x_1 + x_2 + x_3$ is first fitted.
  - Residuals are resampled with replacement to generate new response values y^*, which are added to the fitted values $\hat{y}$.
  - The predictors (x_1, x_2, x_3) remain unchanged in every bootstrap iteration.
- **Assumptions**:
  - The linear model is correctly specified.
  - The residuals are independent and identically distributed (iid).

- **Effect on Results**:
  - Confidence intervals and p-values depend on the model assumptions and the behavior of the residuals.
  - May underestimate variability if residuals deviate significantly from the model's assumptions.

Pairs Bootstrap:

- **How it works**:
  - Rows of the data (y, x_1, x_2, x_3) are resampled with replacement.
  - Both the predictors and the response are treated as part of the population for resampling.
- **Assumptions**:
  - The data are representative of the population and the relationship between predictors and response is valid across the resamples.
- **Effect on Results**:
  - Confidence intervals and p-values incorporate variability in both predictors and the response.
  - More robust when the model assumptions are violated.

Summary of Bootstrapping Methodology

Bootstrapping is a resampling technique used to estimate confidence intervals, variability, and p-values for statistics or model parameters by generating new datasets through resampling.

Parametric Bootstrapping:

- **Method**: Assumes a specific data model (e.g., normal distribution), resamples from it, and computes the statistic.
- **Advantages**: Efficient if the model is correct, leverages prior knowledge.
- **Disadvantages**: Sensitive to model misspecification, less flexible.

Non-Parametric Bootstrapping:

- **Method**: Resamples directly from observed data without assumptions.
- **Advantages**: Robust, flexible, works for non-normal or unknown distributions.
- **Disadvantages**: Computationally expensive, less precise with small datasets.

Comparison Observed in Exercises

- **Parametric** (e.g., residual bootstrap): Produced narrower confidence intervals but relied on assumptions about residuals.
- **Non-Parametric** (e.g., pairs bootstrap): Provided wider, more robust intervals, accounting for variability in predictors and responses.
- **Use Case**: Use parametric when model assumptions hold and non-parametric for robustness or unknown distributions.

Conclusion

Bootstrapping is versatile and powerful but depends on context: - **Parametric**: Suitable for controlled settings with strong assumptions. - **Non-Parametric**: Preferred for observational studies or when assumptions are uncertain.