Exercise 5 - Sampling Intervals for Models

12433732 - Stefan Merdian

2024-11-24

Task 1

Given : 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)

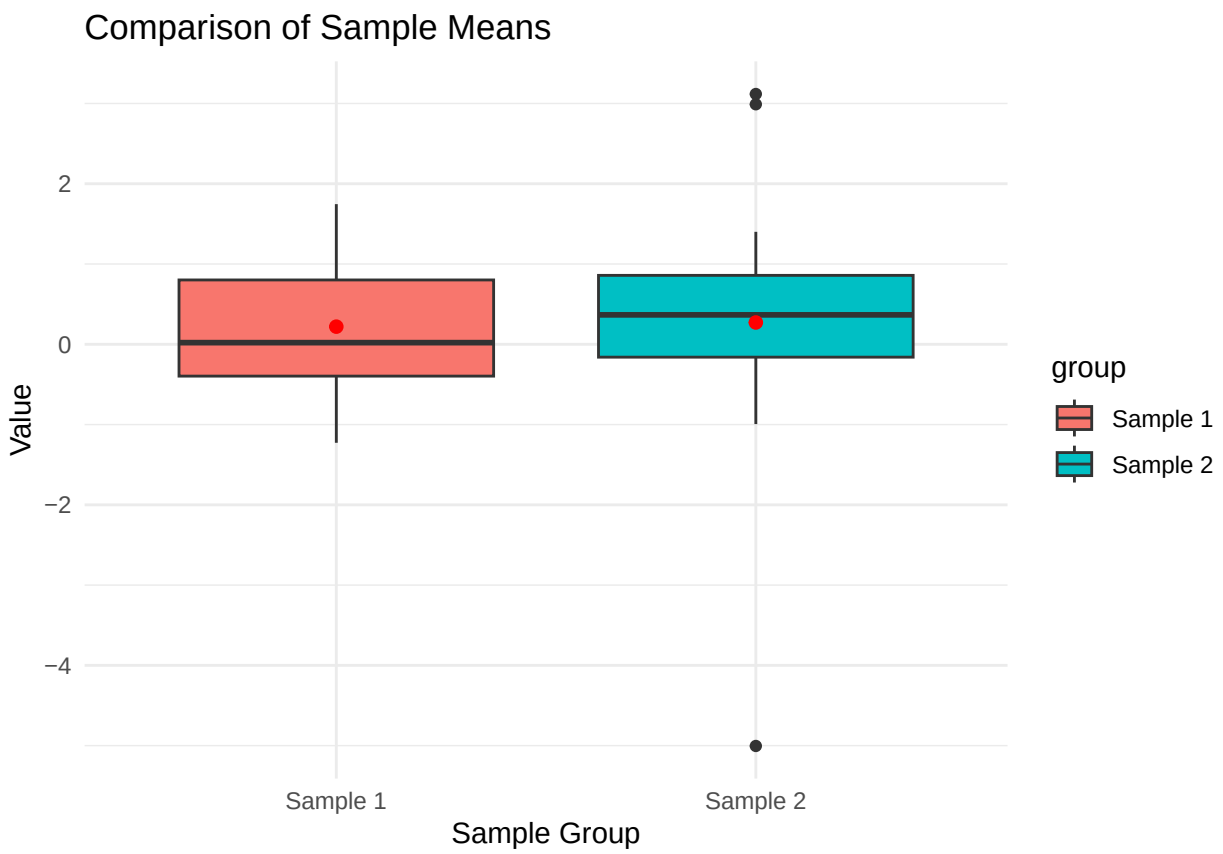
1) Plot the Data

```
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, -0.47
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, 1.4

data <- data.frame(
  value = c(x1, x2),
  group = factor(rep(c("Sample 1", "Sample 2"), times = c(length(x1), length(x2))))
)


library(ggplot2)


ggplot(data, aes(x = group, y = value, fill = group)) +
  geom_boxplot() +
  stat_summary(fun = mean, geom = "point", shape = 20, size = 3, color = "red", fill = "red") +
  labs(title = "Comparison of Sample Means",
       x = "Sample Group",
       y = "Value") +
  theme_minimal()
```



2) Different sampling schemes:

Sampling with replacement from each group

```
set.seed(123)
x1_sample <- sample(x1, size = length(x1), replace = TRUE)
x2_sample <- sample(x2, size = length(x2), replace = TRUE)

print("Sampled x1:")
```

```
## [1] "Sampled x1:"
```

```
print(x1_sample)
```

```
##  [1] -0.578  0.794  1.638  0.572  0.804 -0.188  1.059  0.054 -0.218  0.894
## [11]  1.638  1.059 -0.218  0.794  0.763  0.572 -0.013 -0.415  0.804  0.763
## [21]  0.794 -0.341
```

```
print("Sampled x2:")
```

```
## [1] "Sampled x2:"
```

```
print(x2_sample)
```

```
##  [1]  1.401 -0.005 -0.994  1.128 -0.212  0.007 -0.488  0.413  1.128  0.320
## [11]  0.320 -0.488  1.128  0.100 -0.639  3.118  0.579 -0.212
```

Centering both samples and then resample from the combined samples x1 and x2 for n1 and n2 times.

```
x1_centered <- x1 - mean(x1)
x2_centered <- x2 - mean(x2)

combined_samples <- c(x1_centered, x2_centered)

n1 <- length(x1)
n2 <- length(x2)

resampled_x1 <- sample(combined_samples, size = n1, replace = TRUE)
resampled_x2 <- sample(combined_samples, size = n2, replace = TRUE)

print("Resampled x1:")
```

```
## [1] "Resampled x1:"
```

```
print(resampled_x1)
```

```
##  [1] -0.69181818 -0.40781818 -1.26572222  2.84627778  2.71827778  0.29627778
##  [7] -1.44681818 -0.79781818 -5.27572222  0.04827778  0.72718182  0.30727778
## [13]  0.38318182 -0.23281818  0.83918182  0.83918182 -0.27672222  0.04827778
## [19] -0.54881818 -0.48372222 -0.56081818 -0.69181818
```

```
print("Resampled x2:")
```

```
## [1] "Resampled x2:"
```

```
print(resampled_x2)
```

```
##  [1] -0.43781818  2.71827778  0.83918182  2.71827778 -0.75972222  2.71827778
##  [7]  0.64127778  0.14127778  0.42427778  0.30727778  1.52618182  0.04827778
## [13]  0.30727778  0.14127778  1.41818182  0.85627778 -0.75972222 -0.63481818
```

The first scheme (Replacement from Each Group), is treated for each Group independently and samples are drawn with replacement from each group separately. So it maintains the structure of the original data for each group. It is more natural, because it directly respects the hypothesis structure: - testing whether the means (1 and 2) of the two distinct samples differ

The adventage here is:

The distributions of the two groups remain distinct, which is ideal for evaluating group differences. Since our H0 is H0: 1= 2, so in this case it checks if the means of the two independent groups are diffrent. We can see diffrences in resampling process directly for each group. In this case we assume that both groups are independent. However, if the two groups are not independent, this approach may not be valid because it ignores the relationship or pairing between the groups. Treating dependent groups as independent can lead to:

- the variability between the groups may be overestimated, leading to reduced statistical power. This means you are less likely to detect a real difference between the group

For the second scheme (Centering Both Samples and Resampling from the Combined Samples), both samples are centered by subtracting their means, removing location differences. The centered samples are combined, and resampling is done from this combined dataset.

This approach assumes the null hypothesis H0: 1= 2) is true. It focuses on the variability of the data under the null hypothesis, testing whether the observed differences could arise by chance. The advantage here is, it matches more the null hypothesis for testing whether the observed differences are due to random variation. It explores the variability, which can capture shared features between the two groups.

However here we assume the groups come from a shared distribution, which may not be valid if group specific differences are important. If 1 2, centering artificially removes this difference, potentially underestimating true variability or effect size.

Centering and Resampling from Combined Samples is better suited for testing under H0, particularly when variability is of interest or group distinctions are less critical.

3

Sampling with replacement from each group

```
library(boot)
```

```
## Warning: package 'boot' was built under R version 4.4.2
```

```
observed_t <- t.test(x1, x2, var.equal = TRUE)$statistic
data_combined <- c(x1, x2)
group_indicator <- c(rep(1, length(x1)), rep(2, length(x2)))

# function to calculate t-statistic for resampled data
t_stat <- function(data, indices) {
  group1 <- data[indices[group_indicator == 1]]
  group2 <- data[indices[group_indicator == 2]]
  t.test(group1, group2, var.equal = TRUE)$statistic
}

boot_result_1 <- boot(data = data_combined, statistic = t_stat, R = 10000)

p_value_1 <- mean(abs(boot_result_1$t) >= abs(observed_t))

ci_95_1 <- quantile(boot_result_1$t, probs = c(0.025, 0.975))
ci_99_1 <- quantile(boot_result_1$t, probs = c(0.005, 0.995))

cat("Strategy 1: Sampling with Replacement from Each Group\n")
```

```
## Strategy 1: Sampling with Replacement from Each Group
```

```
cat("Bootstrap p-value:", p_value_1, "\n")
```

```
## Bootstrap p-value: 0.9011
```

```
cat("95% Confidence Interval:", ci_95_1, "\n")
```

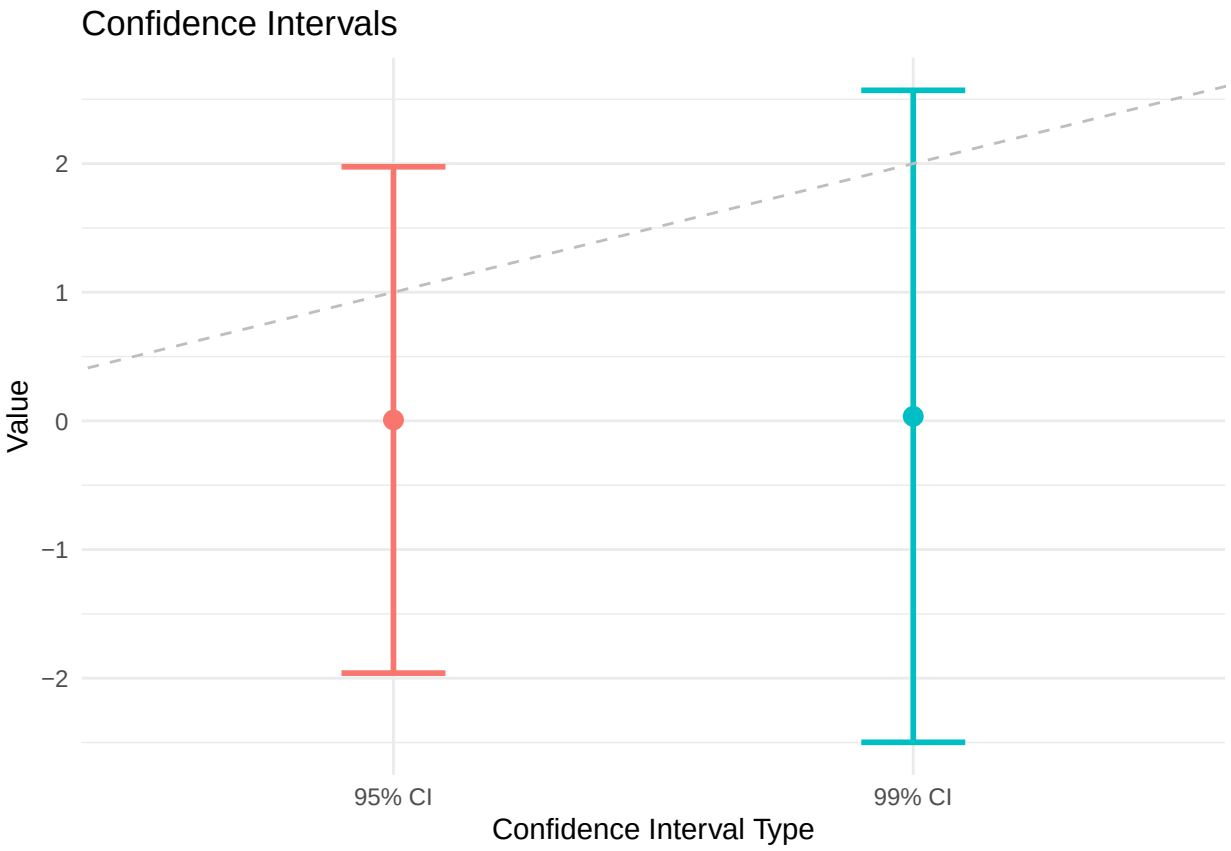
```
## 95% Confidence Interval: -1.96043 1.975561
```

```
cat("99% Confidence Interval:", ci_99_1, "\n\n")
```

```
## 99% Confidence Interval: -2.498415 2.569764
```

```
plot_confidence_intervals(ci_95_1, ci_99_1)
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



Based on the bootstrap results for the first scheme, we are testing whether the means of two samples (1 and 2) are equal (H0: 1= 2) or not (H1: 1 2). At both the 0.05 and 0.01 significance levels, the results indicate that we fail to reject the null hypothesis. This means we do not have enough evidence to conclude that the two means are different.

the p-value from the bootstrap test is 0.905, which is much larger than both 0.05 and 0.01. A large p-value suggests that the observed difference between the means is likely due to random chance. Additionally, the 95% confidence interval for the difference in means is -1.946631 to 1.971155, and the 99% confidence interval

is -2.621404 to 2.491732. Both intervals include 0, which means that no difference in means is a plausible outcome.

In simple terms, this test shows that the two sample means are statistically similar. While this does not prove that the means are exactly the same, it does show that there is no strong evidence to suggest a significant difference between them based on the given data and the bootstrap approach. So p 0.05 and p 0.01, that means we fail to reject H0 –> not enough evidence to suggest the means are different.

Centering both samples and then resample from the combined samples

```
library(boot)

# Center and combine data
x1_centered <- x1 - mean(x1)
x2_centered <- x2 - mean(x2)
data_combined_centered <- c(x1_centered, x2_centered)

#function to calculate t-statistic for resampled data
t_stat_centered <- function(data, indices) {
  group1 <- data[indices[1:length(x1)]]
  group2 <- data[indices[(length(x1) + 1):length(data)]]
  t.test(group1, group2, var.equal = TRUE)$statistic
}

boot_result_2 <- boot(data = data_combined_centered, statistic = t_stat_centered, R = 10000)

p_value_2 <- mean(abs(boot_result_2$t) >= abs(observed_t))

ci_95_2 <- quantile(boot_result_2$t, probs = c(0.025, 0.975))
ci_99_2 <- quantile(boot_result_2$t, probs = c(0.005, 0.995))

cat("Strategy 2: Centering Both Samples and Resampling from Combined Data\n")
```

```
## Strategy 2: Centering Both Samples and Resampling from Combined Data
```

```
cat("Bootstrap p-value:", p_value_2, "\n")
```

```
## Bootstrap p-value: 0.9012
```

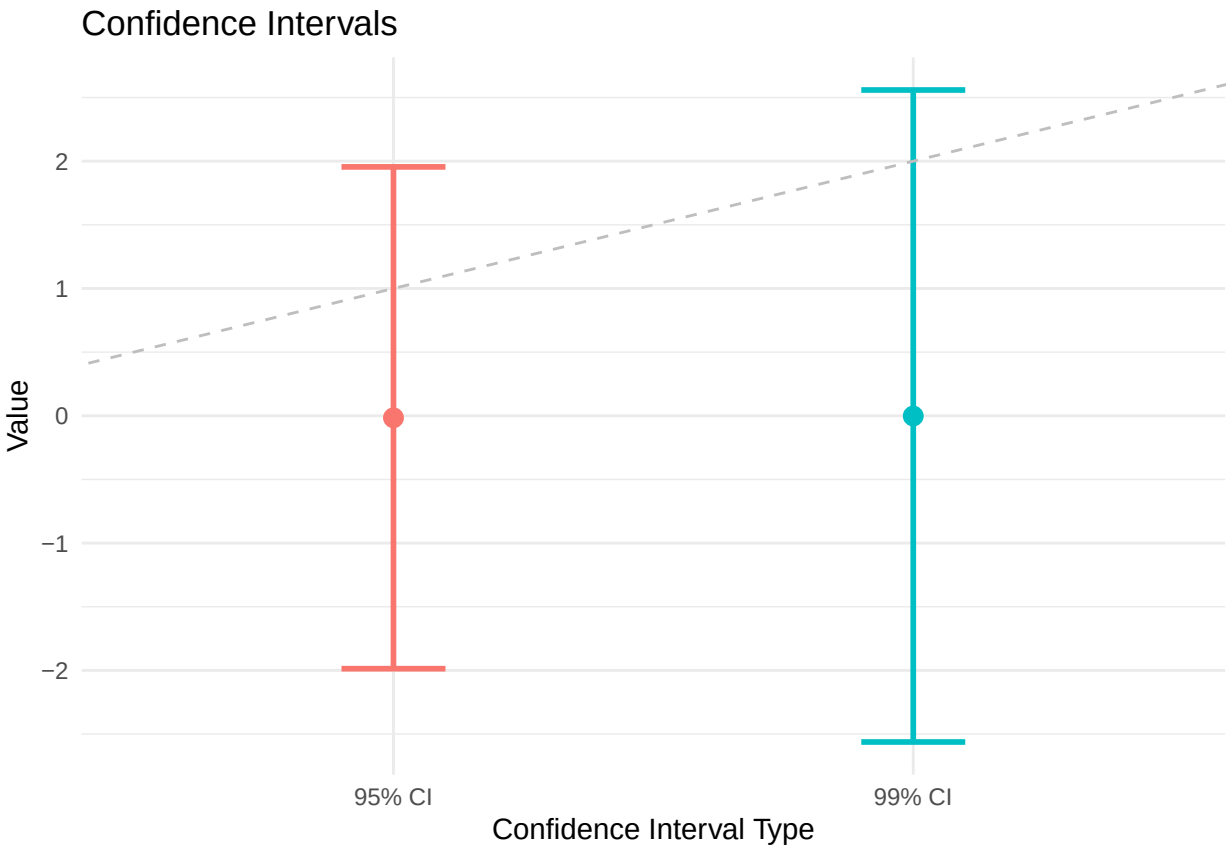
```
cat("95% Confidence Interval:", ci_95_2, "\n")
```

```
## 95% Confidence Interval: -1.986538 1.95493
```

```
cat("99% Confidence Interval:", ci_99_2, "\n\n")
```

```
## 99% Confidence Interval: -2.562472 2.558924
```

```
plot_confidence_intervals(ci_95_2, ci_99_2)
```



We have similar output like the first approach:

P-value of 0.902 is much larger than typical significance levels (=0.05 or =0.01), meaning we fail to reject H0. There is no strong evidence to suggest that 1 2 For both Ci, 95% and 99% 0 is included in the interval, it is possible that the true difference between the means is 0. This further supports the conclusion that we cannot reject H0.

Conclusion:

We use the first approach when our goal is to test under H1 (e.g. finding differences between the two groups). This approach is better for detecting differences because it keeps the group-specific characteristics intact. On the other hand, we use the second approach when it's more important to test under H0H (e.g checking if the two groups really come from the same distribution). This approach combines the data and enforces the null hypothesis during resampling.

In this case, since the null hypothesis is likely true (1 2), both approaches give similar results, as neither finds strong evidence to reject H0.

If the null hypothesis were false (H1), the first approach would probably detect it better because it preserves the differences between the groups. The second approach, however, is more focused on simulating H0 and testing the idea that the two groups are from the same distribution. This is why the second approach works well when validating H0.

4)

```r
data_combined <- c(x1, x2)
n1 <- length(x1)
n2 <- length(x2)


n_permutations <- 10000
t_perm <- numeric(n_permutations)

for (i in 1:n_permutations) {
  # Shuffle the combined data
  permuted_indices <- sample(length(data_combined))

  # Split into two groups
  group1 <- data_combined[permuted_indices[1:n1]]
  group2 <- data_combined[permuted_indices[(n1 + 1):(n1 + n2)]]

  # t-statistic
  t_perm[i] <- t.test(group1, group2, var.equal = TRUE)$statistic
}


p_value_perm <- mean(abs(t_perm) >= abs(observed_t))

# Confidence intervals
ci_95_perm <- quantile(t_perm, probs = c(0.025, 0.975))
ci_99_perm <- quantile(t_perm, probs = c(0.005, 0.995))

# Output results
cat("Two-Sample Permutation Test Results\n")
```

```
## Two-Sample Permutation Test Results
```

```r
cat("Permutation p-value:", p_value_perm, "\n")
```

```
## Permutation p-value: 0.909
```

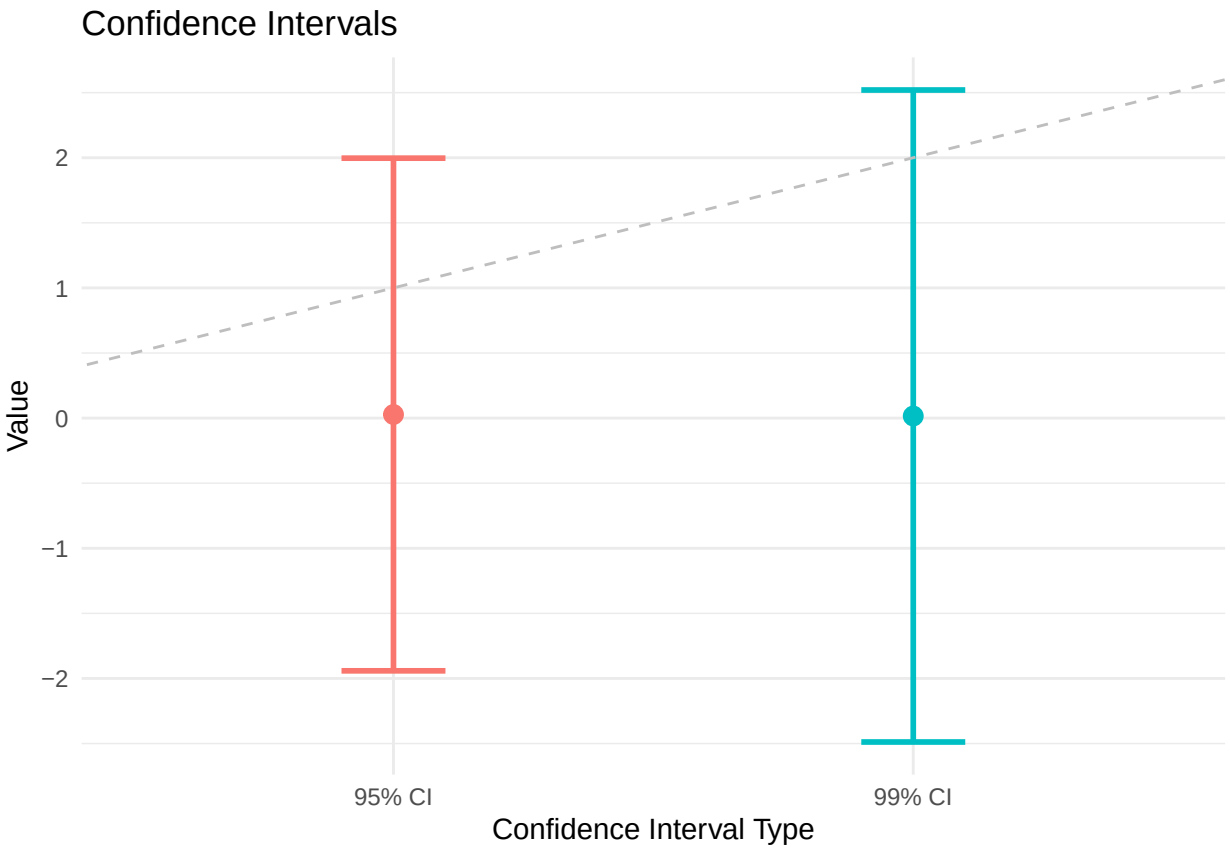
```r
cat("95% Confidence Interval:", ci_95_perm, "\n")
```

```
## 95% Confidence Interval: -1.941424 1.996688
```

```r
cat("99% Confidence Interval:", ci_99_perm, "\n")
```

```
## 99% Confidence Interval: -2.48822 2.520293
```

```r
plot_confidence_intervals(ci_95_perm, ci_99_perm)
```



All three methods (sampling with replacement, centering and pooling, and the permutation test) reached the same conclusion:

- There is no significant evidence to reject the null hypothesis H0: 1= 2.
- The observed differences between the two groups are small and likely due to random chance.

The permutation test gives a similar result to the other methods because all three are testing the same null hypothesis (H0: 1= 2) and the data strongly supports this hypothesis. In the permutation test, the valuess are shuffled to simulate what the data would look like if there truly were no difference between the groups. The observed t-statistic falls well within the range of the null distribution created by the permutations, leading to a large p-value and confidence intervals that include 0.

5) The Wilxocon rank sum test

```
observed_stat <- sum(rank(c(x1, x2))[1:length(x1)])


n1 <- length(x1)
n2 <- length(x2)
n_bootstrap <- 10000
bootstrap_stats_1 <- numeric(n_bootstrap)

for (i in 1:n_bootstrap) {
  sample_x1 <- sample(x1, replace = TRUE)
  sample_x2 <- sample(x2, replace = TRUE)
```

```
  bootstrap_stats_1[i] <- sum(rank(c(sample_x1, sample_x2))[1:length(sample_x1)])
}

p_value_1 <- mean(abs(bootstrap_stats_1 - mean(bootstrap_stats_1)) >= abs(observed_stat - mean(bootstra

ci_95_1 <- quantile(bootstrap_stats_1, probs = c(0.025, 0.975))
ci_99_1 <- quantile(bootstrap_stats_1, probs = c(0.005, 0.995))

cat("Bootstrap with Sampling with Replacement\n")
```

```
## Bootstrap with Sampling with Replacement
```

```
cat("Observed Wilcoxon rank-sum statistic:", observed_stat, "\n")
```

```
## Observed Wilcoxon rank-sum statistic: 434
```

```
cat("Bootstrap p-value:", p_value_1, "\n")
```

```
## Bootstrap p-value: 1
```

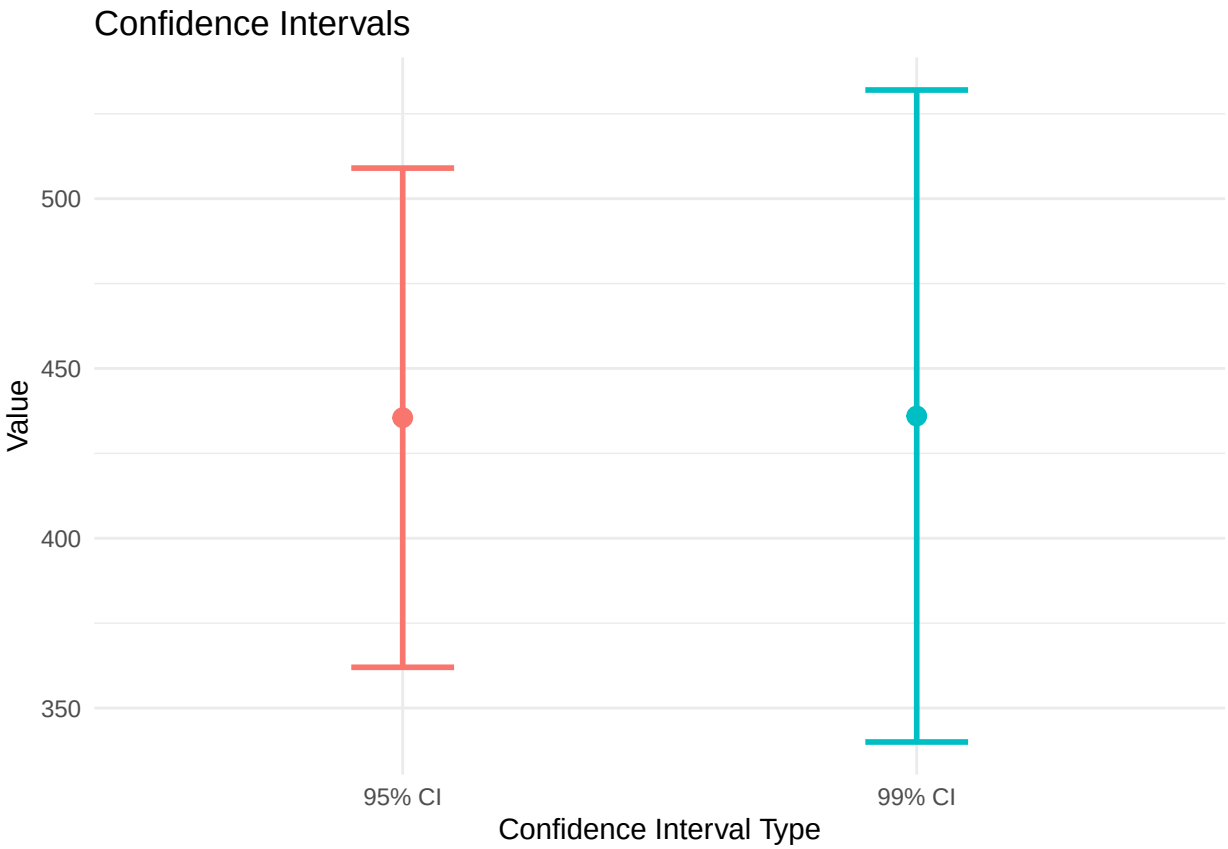
```
cat("95% Confidence Interval:", ci_95_1, "\n")
```

```
## 95% Confidence Interval: 362 509
```

```
cat("99% Confidence Interval:", ci_99_1, "\n\n")
```

```
## 99% Confidence Interval: 340 532
```

```
plot_confidence_intervals(ci_95_1,ci_99_1)
```



The observed Wilcoxon rank-sum statistic is 434, with a bootstrap p-value of 1. This indicates no evidence to reject the null hypothesis, suggesting no significant difference between the two groups. The confidence intervals include the observed statistic, further confirming the results.

The null hypothesis assumes there is no difference between the two groups. The observed statistic is not more extreme than any value in the bootstrap distribution, so the observed rank-sum aligns perfectly with the null hypothesis.

6) Compare results with t.test and wilcox.test.

Using t.test (Parametric Test)

```
t_test_result <- t.test(x1, x2, var.equal = TRUE)

cat("t.test Results\n")
```

```
## t.test Results
```

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

```
## p-value: 0.8996773
```

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

```
## 95% Confidence Interval: -0.8798181 0.77601
```

The p-value is large, indicating no evidence to reject the null hypothesis. It aligns with the results from other methods; however, the 95% confidence interval is slightly narrower here because the t-test assumes normality and calculates the confidence interval for the difference in means directly based on the t-distribution.

Using wilcox.test (Non-Parametric Test)

```
wilcox_test_result <- wilcox.test(x1, x2, exact = FALSE)

cat("wilcox.test Results\n")
```

```
## wilcox.test Results
```

```
cat("Wilcoxon rank-sum statistic:", wilcox_test_result$statistic, "\n")
```

```
## Wilcoxon rank-sum statistic: 181
```

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

```
## p-value: 0.6537383
```

The p-value is smaller than those from the t-test and bootstrap methods but remains non-significant. This difference is notable because the Wilcoxon test does not assume normality and instead uses ranks, making it more robust to outliers or non-normal data. However, it produces slightly different statistics. Additionally, this test does not provide confidence intervals because it does not directly estimate parameters like means or medians, it only tests whether the two distributions differ significantly in location.

The difference in Wilcoxon rank-sum statistics compared to the earlier exercise arises because our manually calculated observed statistic is the sum of ranks for the first sample (x1) in the combined dataset. However the wilcox.test function in R reports the Mann-Whitney U statistic, which is closely related to but not identical to the rank sum

Task 2

1)

```
n <- 200

x1 <- rnorm(n, mean = 2, sd = sqrt(3))
x2 <- runif(n, min = 2, max = 4)
epsilon <- rt(n, df = 5)

y <- 3 + 2 * x1 + x2 + epsilon

x3 <- runif(n, min = -2, max = 2)

data <- data.frame(x1 = x1, x2 = x2, x3 = x3, y = y)
head(data)
```

```
##            x1       x2         x3         y
## 1  1.02802928 2.167701  1.6985842  7.065753
## 2  1.60367346 3.927857  1.0143554 10.184316
## 3  1.29752562 3.224663  1.9143110 10.265203
## 4 -0.36682524 2.120234 -1.1552491  2.983108
## 5  0.03453025 3.827838  0.5669425  6.156578
## 6  1.07119888 3.793380  0.2766124  5.916871
```

2)

```r
fit <- lm(y ~ x1 + x2 + x3, data = data)

n_boot <- 1000
bootstrap_coefs <- matrix(NA, nrow = n_boot, ncol = 4)

for (i in 1:n_boot) {
  resampled_residuals <- sample(residuals(fit), replace = TRUE)

  y_boot <- fitted(fit) + resampled_residuals

  boot_fit <- lm(y_boot ~ x1 + x2 + x3, data = data)

  bootstrap_coefs[i, ] <- coef(boot_fit)
}

ci_percentile <- apply(bootstrap_coefs, 2, quantile, probs = c(0.025, 0.975))
colnames(ci_percentile) <- names(coef(fit))

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

```
## Percentile Confidence Intervals for Coefficients:
```

```r
print(ci_percentile)
```

```
##       (Intercept)       x1        x2          x3
## 2.5%     1.840173 1.953732 0.6932523 -0.06787996
## 97.5%    3.888275 2.161365 1.3324503  0.26716130
```

```r
# Check if x3 can be excluded
if (ci_percentile[1, "x3"] <= 0 && ci_percentile[2, "x3"] >= 0) {
  cat("\nBased on the confidence interval, x3 can be excluded as it includes 0.\n")
} else {
  cat("\nBased on the confidence interval, x3 cannot be excluded as it does not include 0.\n")
}
```

```
##
## Based on the confidence interval, x3 can be excluded as it includes 0.
```

3)

```r
# Pairs bootstrap
n_boot <- 1000
bootstrap_coefs <- matrix(NA, nrow = n_boot, ncol = 4)  # Store coefficients

for (i in 1:n_boot) {
  resampled_data <- data[sample(1:n, replace = TRUE), ]

  boot_fit <- lm(y ~ x1 + x2 + x3, data = resampled_data)

  bootstrap_coefs[i, ] <- coef(boot_fit)
}

# Calculate percentile confidence intervals
ci_percentile <- apply(bootstrap_coefs, 2, quantile, probs = c(0.025, 0.975))
colnames(ci_percentile) <- names(coef(fit))

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

```
## Percentile Confidence Intervals for Coefficients:
```

```r
print(ci_percentile)
```

```
##       (Intercept)       x1        x2         x3
## 2.5%     1.910667 1.954507 0.6903668 -0.0711222
## 97.5%    3.831312 2.158810 1.3228032  0.2794531
```

```r
# Check if x3 can be excluded
ci_x3 <- ci_percentile[, "x3"]
if (ci_x3[1] <= 0 && ci_x3[2] >= 0) {
  cat("\nBased on the confidence interval [", round(ci_x3[1], 4), ", ", round(ci_x3[2], 4),
      "], x3 can be excluded as it includes 0.\n", sep = "")
} else {
  cat("\nBased on the confidence interval [", round(ci_x3[1], 4), ", ", round(ci_x3[2], 4),
      "], x3 cannot be excluded as it does not include 0.\n", sep = "")
}
```

```
##
## Based on the confidence interval [-0.0711, 0.2795], x3 can be excluded as it includes 0.
```

4)

Residual Bootstrap:

After fitting the regression model to the original data, this method involves calculating the residuals (the differences between observed and predicted values), randomly sampling these residuals with replacement, adding the resampled residuals back to the model's fitted values to create new response variables, and keeping the predictor variables unchanged. This approach assumes that the model is correctly specified and that the residuals are identically distributed. It tends to produce narrower confidence intervals for the

coefficients, reflecting higher precision under the assumption that the model captures the true relationship between predictors and the response variable.

Pairs Bootstrap:

This method involves resampling entire data points (pairs of response and predictor variables) with replacement from the original dataset and fitting the regression model to each resampled dataset. The pairs bootstrap does not assume that the model is correctly specified. It is more robust to model misspecification and can handle situations where the relationship between variables is complex or unknown. However, this robustness often results in wider confidence intervals for the coefficients, indicating greater uncertainty in the estimates.

So the residual bootstrap are more efficient with narrower confidence intervals when the model is correctly specified. However pairs bootstrap are more robust with wider confidence intervals, especially useful when the model may be misspecified.

The confidence intervals for the coefficient of x3 are:

- Residual Bootstrap: -0.1868, 0.212
- Pairs Bootstrap: -0.2136, 0.244

Both intervals include zero, suggesting that x3 may not be a significant predictor.

Task 3

Bootstrapping is a statistical technique that involves repeatedly sampling from your data to create an empirical distribution of a statistic. This method allows for the estimation of confidence intervals, p-values, and other metrics without relying heavily on traditional parametric assumptions.

Like already discussed last week

Parametric vs. Nonparametric Bootstrapping:

• Parametric bootstrapping assumes that the data comes from a specific distribution (like in our case normal distribution). It generates bootstrap samples based on the scale (like mean and standard deviation) of that assumed distribution. This makes parametric bootstrapping more stable since it relies on an underlying model to filter out noise, which makes it less affected by extreme values or outliers.

• Nonparametric bootstrapping, on the other hand, makes no assumptions about the data distribution. It directly resamples the original data points. This makes it more flexible but also more sensitive to outliers or irregularities, as it uses whatever values are present in the data without any smoothing assumptions

In hypothesis testing, bootstrapping entails generating datasets under the null hypothesis and computing the test statistic for each sample. For instance, bootstrapped t-tests and rank-sum tests can be employed to determine confidence intervals and p-values. By comparing the observed statistic to the bootstrapped distribution, one can assess significance and construct confidence intervals.

In regression analysis, bootstrapping is utilized to estimate confidence intervals for model coefficients. This can be achieved through residual bootstrapping, which involves resampling residuals to maintain the model's structure, or pairs bootstrapping, which involves resampling entire observation rows.