102b_hw03_Daren_Sathasivam

Daren Sathasivam

2024-04-30

Problem 1:

```
# Load packages, data, and helper functions
library(MASS)
data("cars")
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

```
source("bootsample.R")
source("bootstats.R")
```

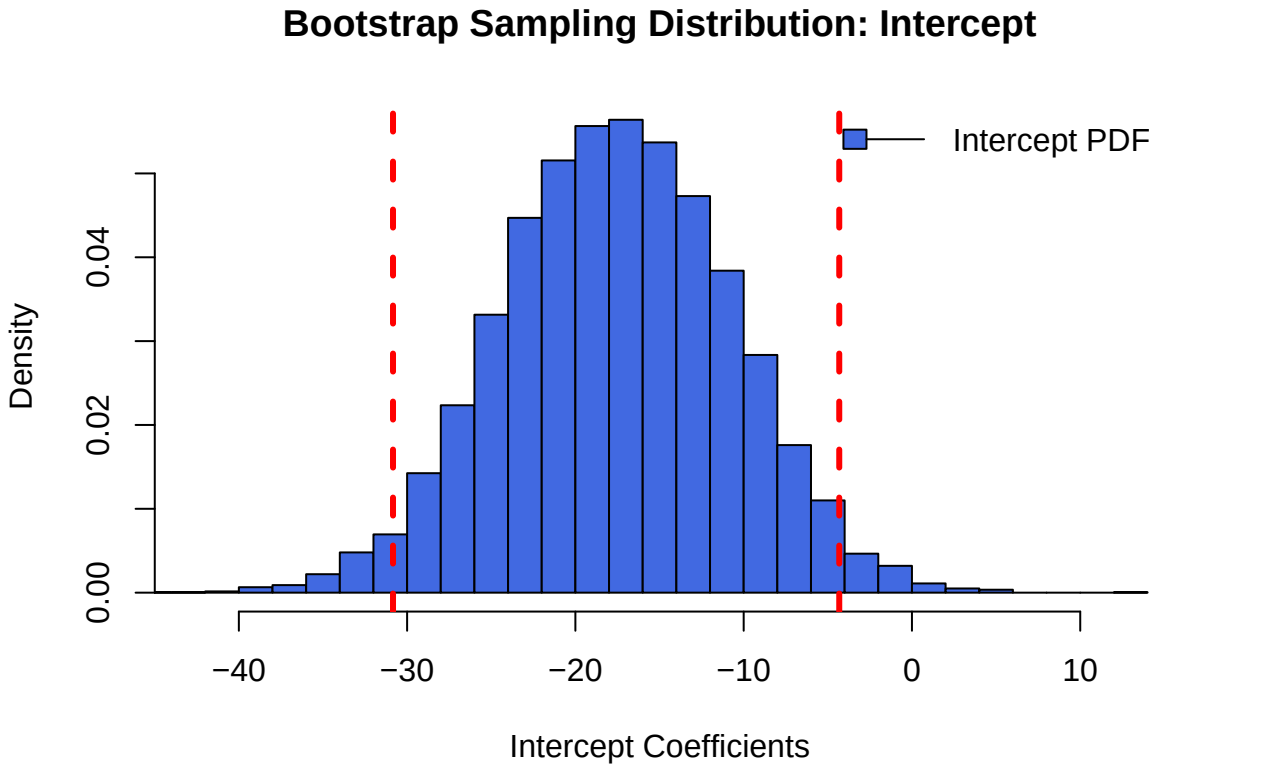
- Consider the simple linear regression model:
  - $y_i = \beta_0 + \beta_1 x_i + \epsilon_i, i = 1, ..., 50$, where x = speed and y = braking distance

Part(a) Parametric Bootstrap:

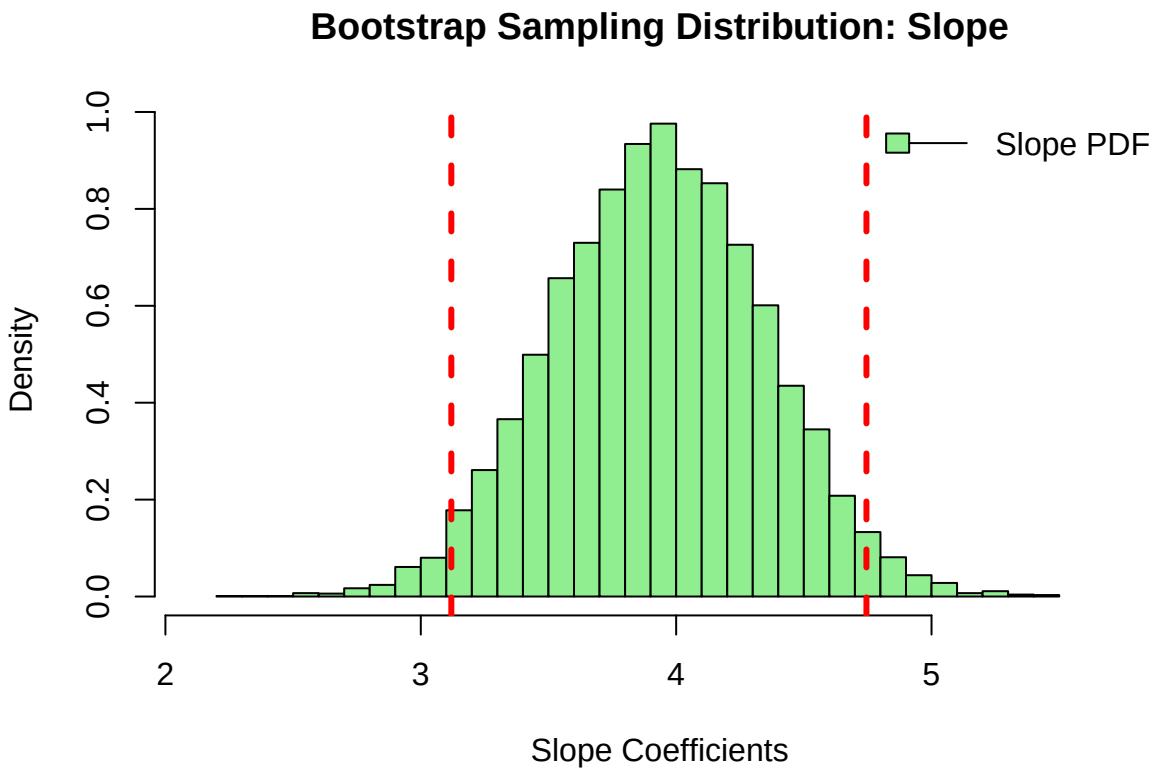
1. Normal Bootstrap CI

```
# Set seed for reproducibility
set.seed(100)
# Fit the initial linear model
lm_initial <- lm(dist ~ speed, data = cars)
# Extract coefficients and residuals
beta_initial <- coef(lm_initial)
residuals_initial <- residuals(lm_initial)
# Number of bootstrap replicates
B <- 10000
# Define the alpha level and calculate the z-value for the
# Normal CI
alpha <- 0.05
z_val <- qnorm(1 - alpha/2)
# Initialize matrices to store bootstrap results
bootstrap_betas <- matrix(nrow = B, ncol = 2)

# Perform bootstrap resampling
```

```r
for (i in 1:B) {
    # Simulate new residuals
    simulated_residuals <- rnorm(n = length(residuals_initial),
        mean = 0, sd = sd(residuals_initial))
    # Generate new response variable
    y_new <- fitted(lm_initial) + simulated_residuals
    # Fit new model to the bootstrapped dataset
    lm_bootstrap <- lm(y_new ~ speed, data = cars)
    # Store the coefficients
    bootstrap_betas[i, ] <- coef(lm_bootstrap)
}
# Calculate the standard errors of bootstrap estimates
bootstrap_se <- apply(bootstrap_betas, 2, sd)


# Calculate Normal Confidence Intervals
normal_ci_intercept <- beta_initial[1] + c(-1, 1) * z_val * bootstrap_se[1]
normal_ci_slope <- beta_initial[2] + c(-1, 1) * z_val * bootstrap_se[2]

# par(mfrow = c(1, 2)) Plotting the histograms for
# intercept
hist(bootstrap_betas[, 1], breaks = 30, freq = FALSE, xlim = range(bootstrap_betas[,
    1]) * c(0.95, 1.05), col = "royalblue", border = "black",
    main = "Bootstrap Sampling Distribution: Intercept", xlab = "Intercept Coefficients",
    ylab = "Density")
abline(v = normal_ci_intercept, col = "red", lwd = 3, lty = 2)
legend("topright", "Intercept PDF", lty = 1, bty = "n", fill = "royalblue")
```



```r
# Plotting the histograms for slope
hist(bootstrap_betas[, 2], breaks = 30, freq = FALSE, xlim = range(bootstrap_betas[,
    2]) * c(0.95, 1.05), col = "lightgreen", border = "black",
    main = "Bootstrap Sampling Distribution: Slope", xlab = "Slope Coefficients",
    ylab = "Density")
abline(v = normal_ci_slope, col = "red", lwd = 3, lty = 2)
legend("topright", "Slope PDF", lty = 1, bty = "n", fill = "lightgreen")
```




```r
# Output results
cat("Normal Bootstrap CI for Intercept: [", normal_ci_intercept[1],
    ", ", normal_ci_intercept[2], "]\n")
```

```
## Normal Bootstrap CI for Intercept: [ -30.83843 ,  -4.319757 ]
```

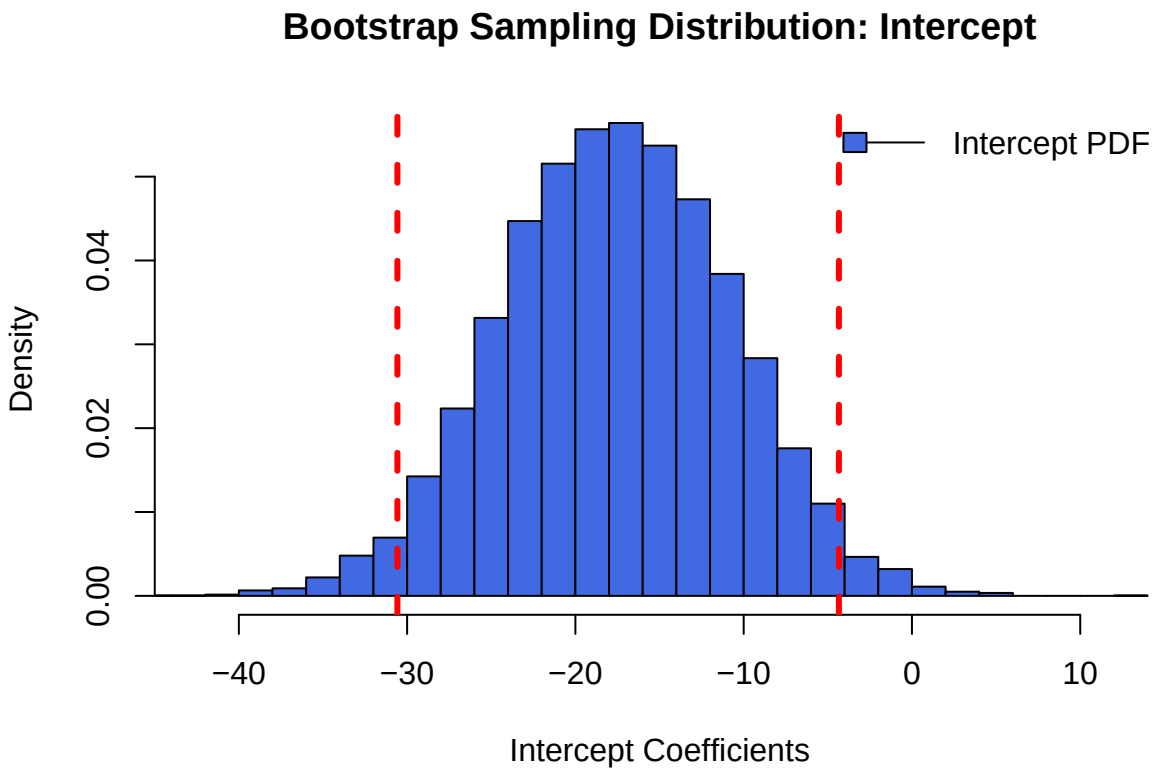
```r
cat("Normal Bootstrap CI for Slope: [", normal_ci_slope[1], ", ",
    normal_ci_slope[2], "]\n")
```

```
## Normal Bootstrap CI for Slope: [ 3.119599 ,  4.745218 ]
```

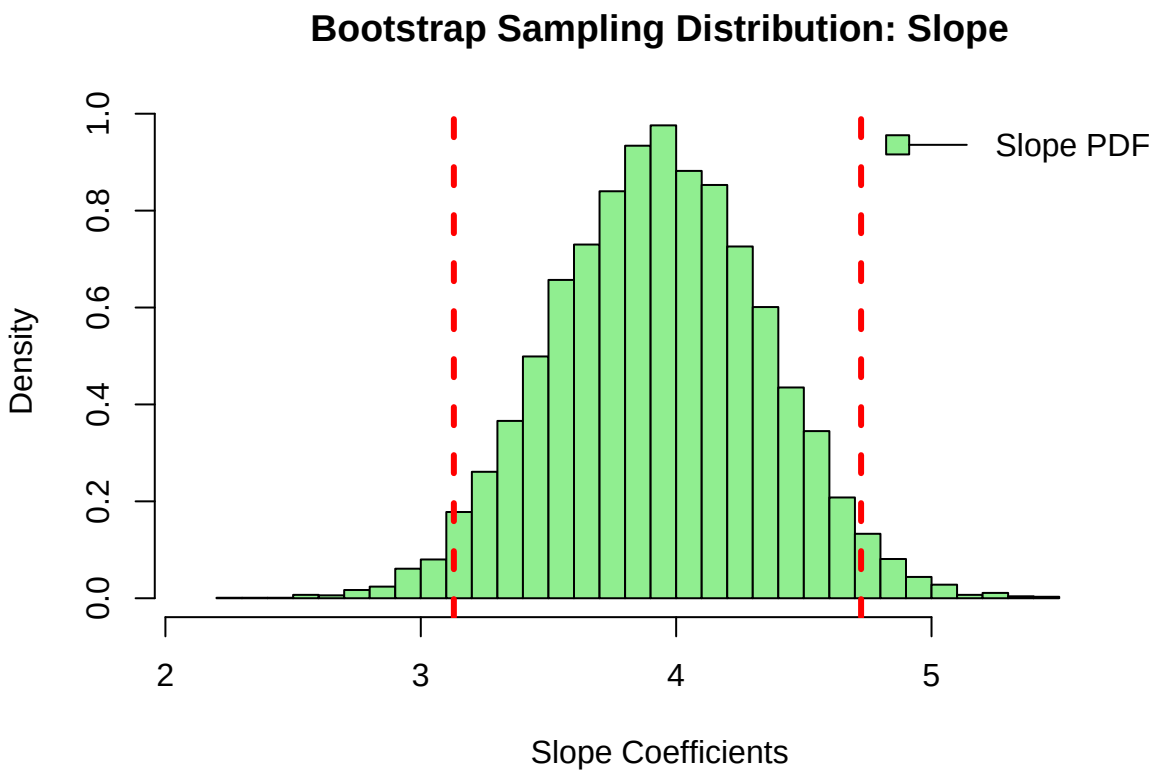
2. Basic Bootstrap CI

```r
# Calculate Basic Bootstrap Confidence Intervals
basic_ci_intercept <- 2 * beta_initial[1] - quantile(bootstrap_betas[,
    1], probs = c(1 - alpha/2, alpha/2))
basic_ci_slope <- 2 * beta_initial[2] - quantile(bootstrap_betas[,
    2], probs = c(1 - alpha/2, alpha/2))

# Plotting the histograms for intercept
hist(bootstrap_betas[, 1], breaks = 30, freq = FALSE, xlim = range(bootstrap_betas[,
    1]) * c(0.95, 1.05), col = "royalblue", border = "black",
```

```
    main = "Bootstrap Sampling Distribution: Intercept", xlab = "Intercept Coefficients",
    ylab = "Density")
abline(v = basic_ci_intercept, col = "red", lwd = 3, lty = 2)
legend("topright", "Intercept PDF", lty = 1, bty = "n", fill = "royalblue")
```




```
# Plotting the histograms for slope
hist(bootstrap_betas[, 2], breaks = 30, freq = FALSE, xlim = range(bootstrap_betas[,
    2]) * c(0.95, 1.05), col = "lightgreen", border = "black",
    main = "Bootstrap Sampling Distribution: Slope", xlab = "Slope Coefficients",
    ylab = "Density")
abline(v = basic_ci_slope, col = "red", lwd = 3, lty = 2)
legend("topright", "Slope PDF", lty = 1, bty = "n", fill = "lightgreen")
```




```
# Output results
cat("Basic Bootstrap CI for Intercept: [", basic_ci_intercept[1],
    ", ", basic_ci_intercept[2], "]\n")
```

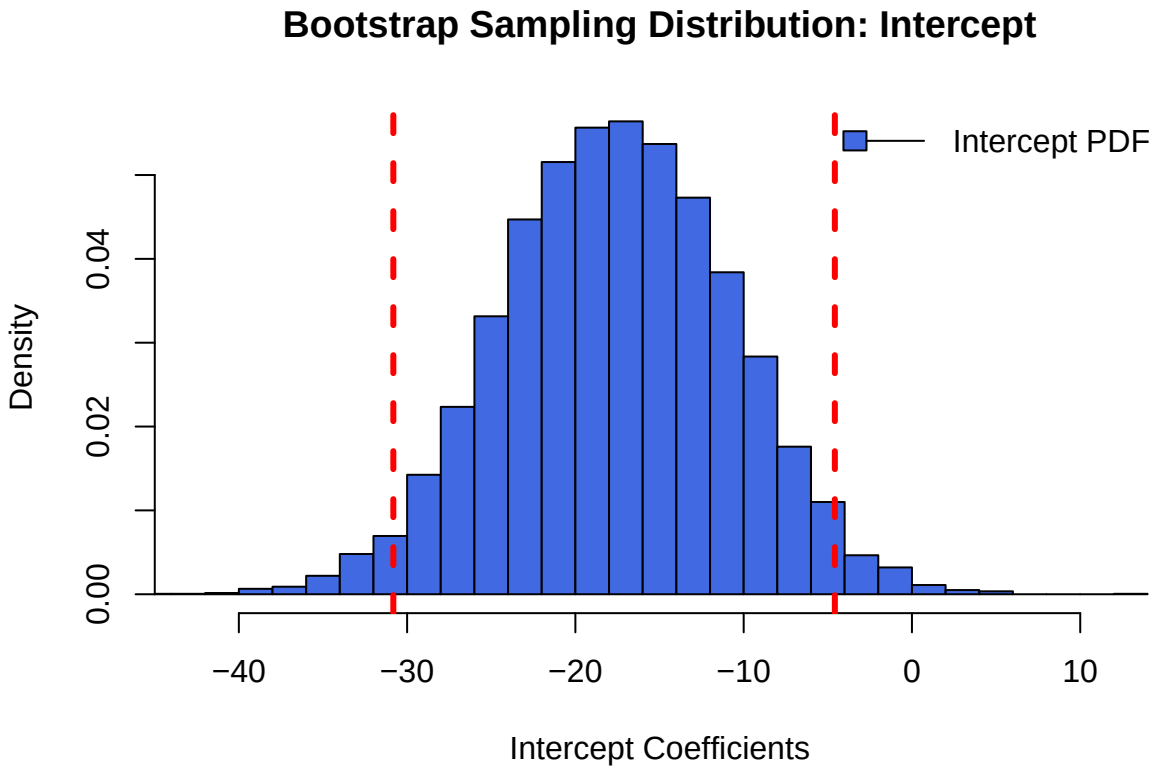
```
## Basic Bootstrap CI for Intercept: [ -30.57642 ,  -4.339112 ]
```

```
cat("Basic Bootstrap CI for Slope: [", basic_ci_slope[1], ", ",
    basic_ci_slope[2], "]\n")
```

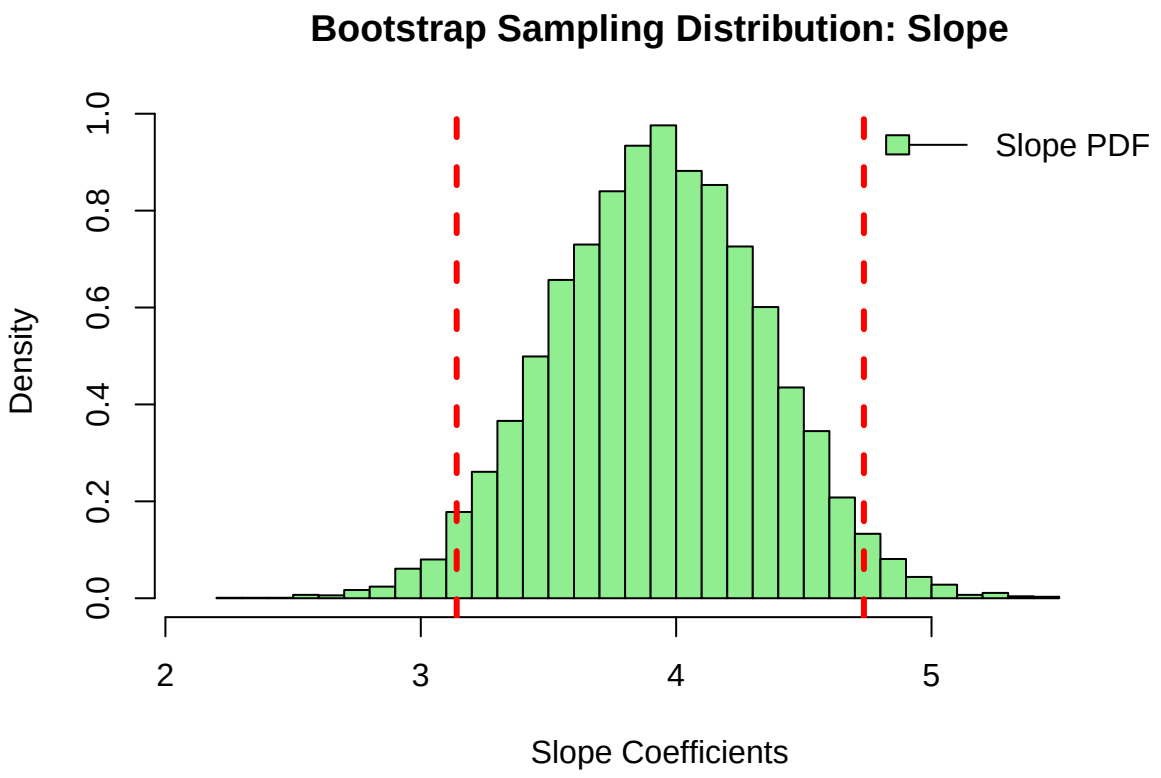
```
## Basic Bootstrap CI for Slope: [ 3.129563 ,  4.724175 ]
```

3. Percentile Bootstrap CI

```
# Calculate Percentile Bootstrap Confidence Intervals
percentile_ci_intercept <- quantile(bootstrap_betas[, 1], probs = c(alpha/2,
    1 - alpha/2))
percentile_ci_slope <- quantile(bootstrap_betas[, 2], probs = c(alpha/2,
    1 - alpha/2))

# Plotting the histograms for intercept
hist(bootstrap_betas[, 1], breaks = 30, freq = FALSE, xlim = range(bootstrap_betas[,
    1]) * c(0.95, 1.05), col = "royalblue", border = "black",
    main = "Bootstrap Sampling Distribution: Intercept", xlab = "Intercept Coefficients",
    ylab = "Density")
abline(v = percentile_ci_intercept, col = "red", lwd = 3, lty = 2)
legend("topright", "Intercept PDF", lty = 1, bty = "n", fill = "royalblue")
```




```r
# Plotting the histograms for slope
hist(bootstrap_betas[, 2], breaks = 30, freq = FALSE, xlim = range(bootstrap_betas[,
    2]) * c(0.95, 1.05), col = "lightgreen", border = "black",
    main = "Bootstrap Sampling Distribution: Slope", xlab = "Slope Coefficients",
    ylab = "Density")
abline(v = percentile_ci_slope, col = "red", lwd = 3, lty = 2)
legend("topright", "Slope PDF", lty = 1, bty = "n", fill = "lightgreen")
```




```
# Output results
cat("Percentile Bootstrap CI for Intercept: [", percentile_ci_intercept[1],
    ", ", percentile_ci_intercept[2], "]\n")
```

```
## Percentile Bootstrap CI for Intercept: [ -30.81908 ,  -4.581767 ]
```

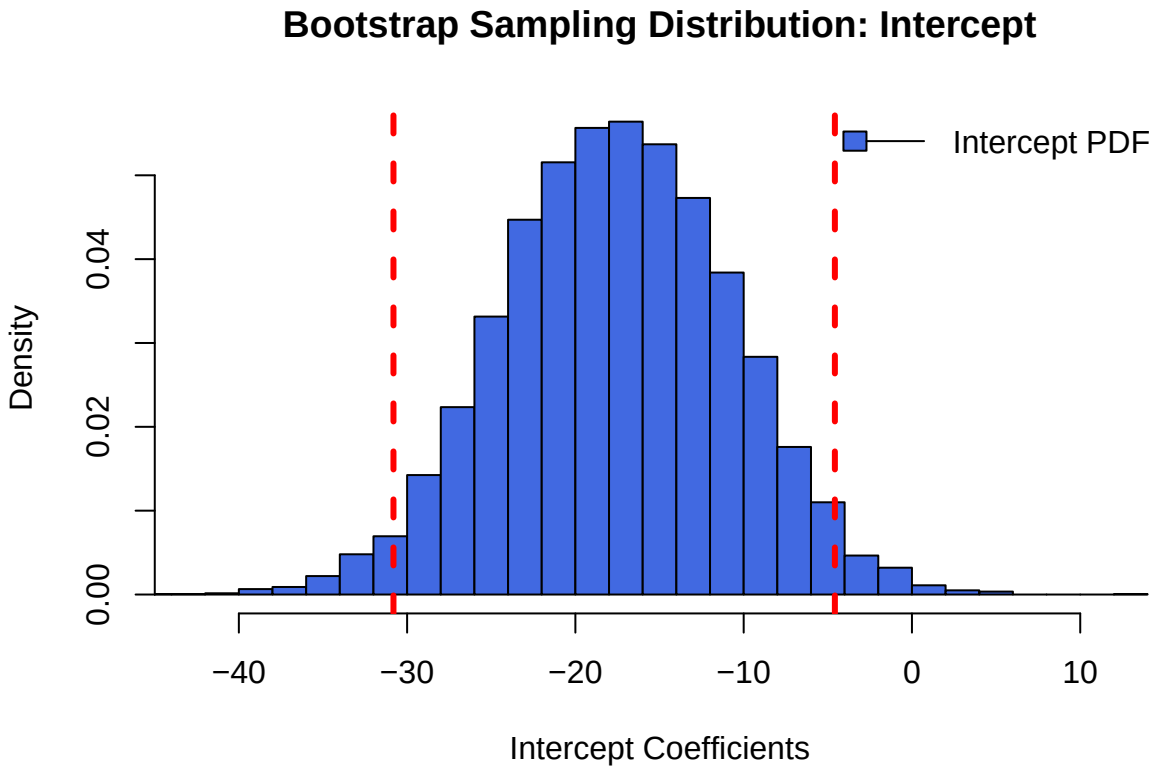
```
cat("Percentile Bootstrap CI for Slope: [", percentile_ci_slope[1],
    ", ", percentile_ci_slope[2], "]\n")
```

```
## Percentile Bootstrap CI for Slope: [ 3.140642 ,  4.735255 ]
```

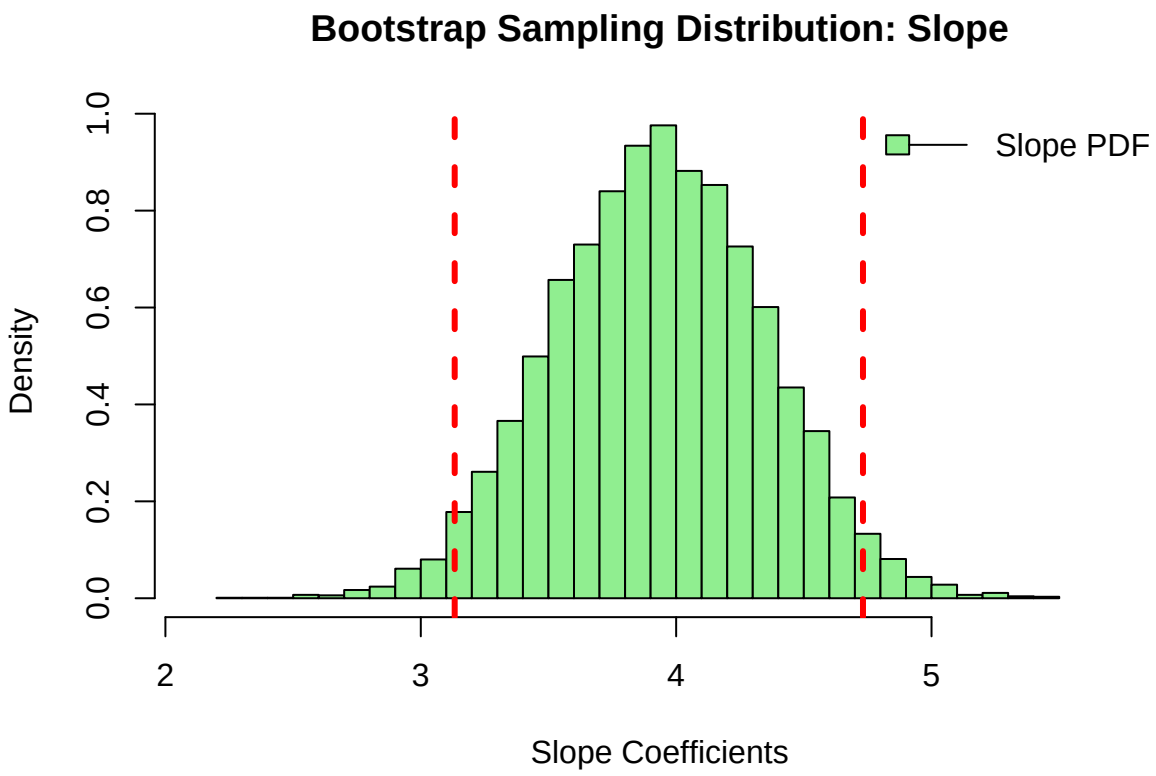
4. Bias corrected Bootstrap CI

```
# Calculate Bias Corrected Bootstrap Confidence Intervals
z0int <- qnorm(sum(bootstrap_betas[, 1] < beta_initial[1])/B)
z0slope <- qnorm(sum(bootstrap_betas[, 2] < beta_initial[2])/B)
A1int <- pnorm(2 * z0int + qnorm(alpha/2))
A2int <- pnorm(2 * z0int + qnorm(1 - alpha/2))
A1slope <- pnorm(2 * z0slope + qnorm(alpha/2))
A2slope <- pnorm(2 * z0slope + qnorm(1 - alpha/2))

bc_ci_intercept <- quantile(bootstrap_betas[, 1], probs = c(A1int,
    A2int))
bc_ci_slope <- quantile(bootstrap_betas[, 2], probs = c(A1slope,
    A2slope))

# Plotting the histograms for intercept
hist(bootstrap_betas[, 1], breaks = 30, freq = FALSE, xlim = range(bootstrap_betas[,
    1]) * c(0.95, 1.05), col = "royalblue", border = "black",
    main = "Bootstrap Sampling Distribution: Intercept", xlab = "Intercept Coefficients",
```

```
    ylab = "Density")
abline(v = bc_ci_intercept, col = "red", lwd = 3, lty = 2)
legend("topright", "Intercept PDF", lty = 1, bty = "n", fill = "royalblue")
```




```
# Plotting the histograms for slope
hist(bootstrap_betas[, 2], breaks = 30, freq = FALSE, xlim = range(bootstrap_betas[,
    2]) * c(0.95, 1.05), col = "lightgreen", border = "black",
    main = "Bootstrap Sampling Distribution: Slope", xlab = "Slope Coefficients",
    ylab = "Density")
abline(v = bc_ci_slope, col = "red", lwd = 3, lty = 2)
legend("topright", "Slope PDF", lty = 1, bty = "n", fill = "lightgreen")
```




```
# Output results
cat("Bias Corrected Bootstrap CI for Intercept: [", bc_ci_intercept[1],
    ", ", bc_ci_intercept[2], "]\n")
```

```
## Bias Corrected Bootstrap CI for Intercept: [ -30.81128 ,  -4.580708 ]
```

```
cat("Bias Corrected Bootstrap CI for Slope: [", bc_ci_slope[1],
    ", ", bc_ci_slope[2], "]\n")
```

```
## Bias Corrected Bootstrap CI for Slope: [ 3.132752 ,  4.731557 ]
```

```

# Function to calculate CI length and shape
calculate_length_shape <- function(ci, original_estimate) {
    length_ci <- diff(ci)
    # Calculate the shape using the provided formula
    shape_ci <- (ci[2] - original_estimate)/(original_estimate -
        ci[1])
    return(list(length = length_ci, shape = shape_ci))
}

# Calculate Length and Shape for each Bootstrap Type
normal_stats_intercept <- calculate_length_shape(normal_ci_intercept,
    beta_initial[1])
normal_stats_slope <- calculate_length_shape(normal_ci_slope,
    beta_initial[2])

basic_stats_intercept <- calculate_length_shape(basic_ci_intercept,
    beta_initial[1])
basic_stats_slope <- calculate_length_shape(basic_ci_slope, beta_initial[2])
```

```r
percentile_stats_intercept <- calculate_length_shape(percentile_ci_intercept,
    beta_initial[1])
percentile_stats_slope <- calculate_length_shape(percentile_ci_slope,
    beta_initial[2])

bc_stats_intercept <- calculate_length_shape(bc_ci_intercept,
    beta_initial[1])
bc_stats_slope <- calculate_length_shape(bc_ci_slope, beta_initial[2])

# Printed results below. No R code to display since it is
# just print functions and takes up too much space.
```

Calculate the length and shape of each type of Bootstrap CI and report them as well. (Shape $= \frac{C_U - \hat{\theta}}{\hat{\theta} - C_L}$**)**

```
## Normal Bootstrap CI for Intercept:
## Length: 26.51868
## Shape: 1
## Normal Bootstrap CI for Slope:
## Length: 1.625619
## Shape: 1
## Basic Bootstrap CI for Intercept:
## Length: 26.23731
## Shape: 1.01867
## Basic Bootstrap CI for Slope:
## Length: 1.594612
## Shape: 0.9862002
## Percentile Bootstrap CI for Intercept:
## Length: 26.23731
## Shape: 0.9816726
## Percentile Bootstrap CI for Slope:
## Length: 1.594612
## Shape: 1.013993
## Bias Corrected Bootstrap CI for Intercept:
## Length: 26.23057
## Shape: 0.982331
## Bias Corrected Bootstrap CI for Slope:
## Length: 1.598805
## Shape: 0.999364
```

- The shape value is calculated using the confidence intervals lower and upper bounds and the values of the estimates. A shape value of 1 indicates that the confidence interval is symmetric around the original estimate, suggesting that it consists of unbiased and evenly distributed set of bootstrap estimates. A shape value less than 1 indicates a skew towards lower values and a value higher than 1 indicates a skew towards higher values which can signal potential biases or issues with the distribution in the data.

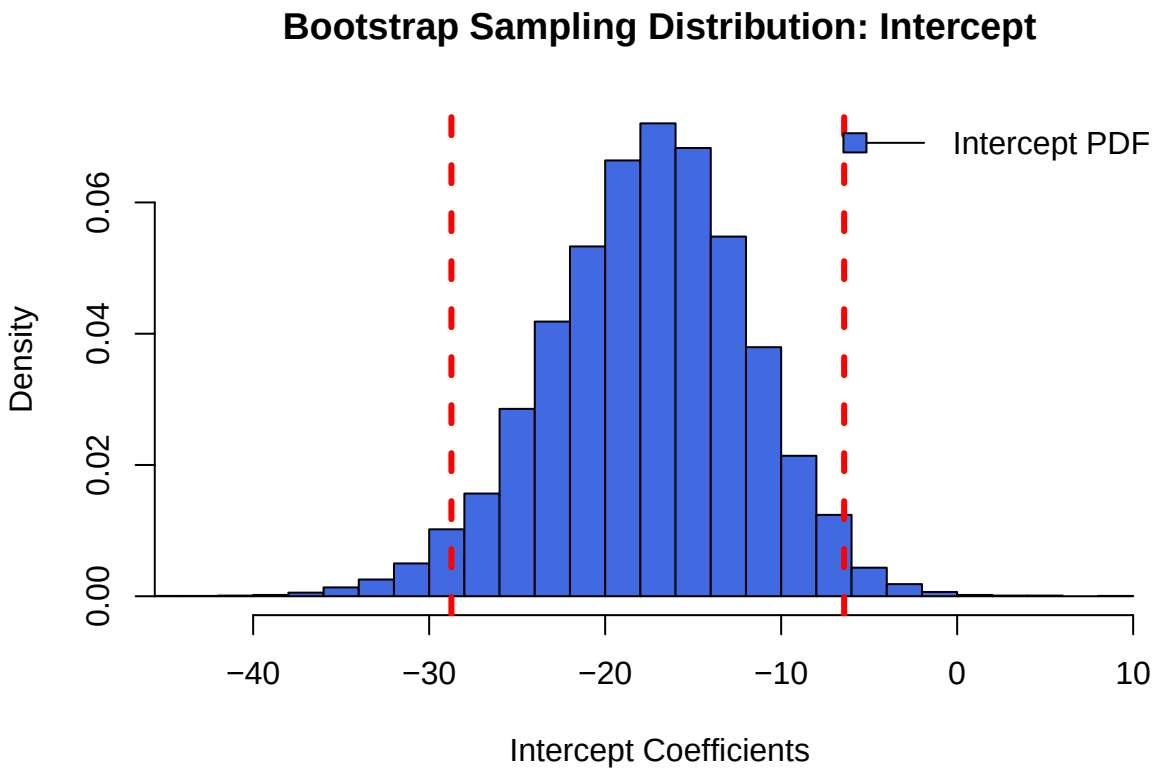
Discuss how you selected the number of bootstrap B replicates and comment on the results.

- I chose to use 10000 as the number of bootstrap replicates as it would maintain computational efficiency and also provide accurate results due to the number of replications being relatively large. This value helps mitigate the effects of random sampling error and assists with the credibility of the bootstrap results.

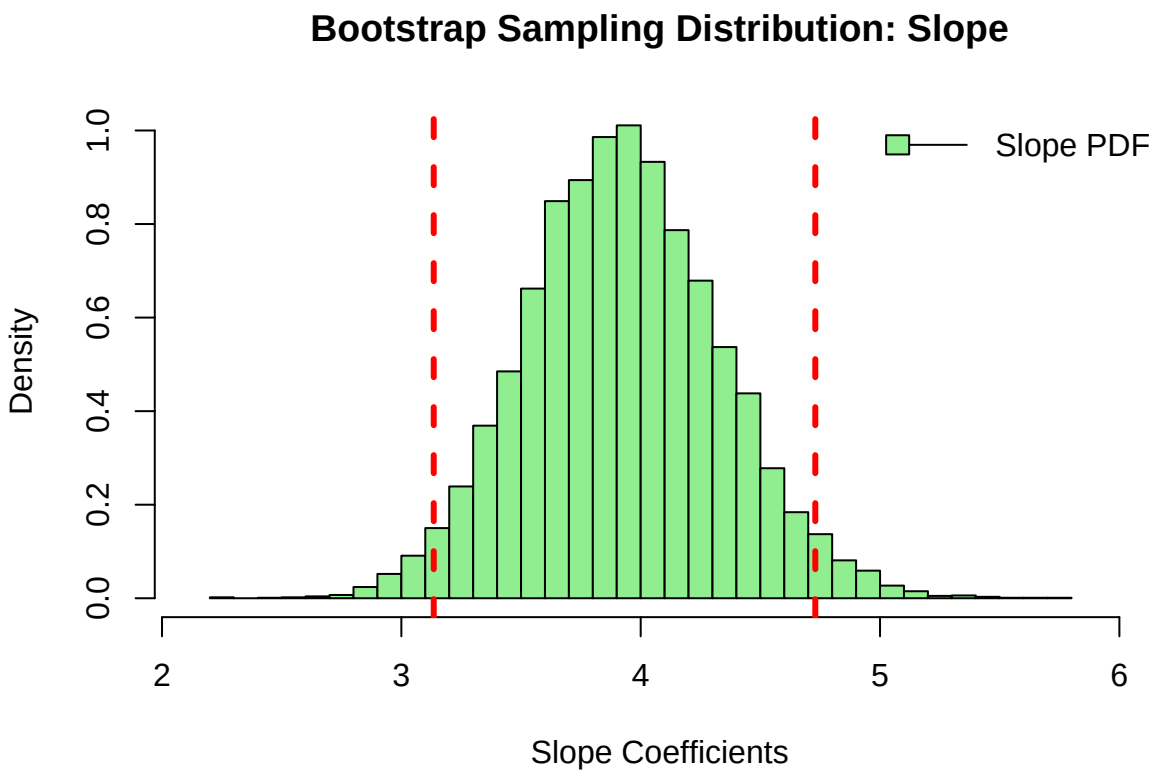
Part(b) Nonparametric Bootstrap:

1. Normal Bootstrap CI

```
# Set seed for reproducibility
set.seed(100)
# Fit the initial linear model
lm_initial <- lm(dist ~ speed, data = cars)
# Extract original estimates
beta_initial <- coef(lm_initial)
# Number of bootstrap replicates
B <- 10000
alpha <- 0.05
z_val <- qnorm(1 - alpha/2)

# Initialize matrices to store bootstrap results
bootstrap_betas <- matrix(nrow = B, ncol = 2)

# Perform non-parametric bootstrap resampling
for (i in 1:B) {
    # Resample the indices of the data
    resampled_indices <- sample(nrow(cars), replace = TRUE)
    resampled_data <- cars[resampled_indices, ]
    # Fit model to the resampled data
    lm_resampled <- lm(dist ~ speed, data = resampled_data)
    # Store the coefficients
    bootstrap_betas[i, ] <- coef(lm_resampled)
}
# Calculate the standard errors of bootstrap estimates
bootstrap_se <- apply(bootstrap_betas, 2, sd)

# Calculate Normal Confidence Intervals
normal_ci_intercept <- beta_initial[1] + c(-1, 1) * z_val * bootstrap_se[1]
normal_ci_slope <- beta_initial[2] + c(-1, 1) * z_val * bootstrap_se[2]

# Plotting the histograms for intercept
hist(bootstrap_betas[, 1], breaks = 30, freq = FALSE, xlim = range(bootstrap_betas[,
    1]) * c(0.95, 1.05), col = "royalblue", main = "Bootstrap Sampling Distribution: Intercept",
    xlab = "Intercept Coefficients", ylab = "Density")
abline(v = normal_ci_intercept, col = "red", lwd = 3, lty = 2)
legend("topright", "Intercept PDF", lty = 1, bty = "n", fill = "royalblue")
```




```
# Plotting the histograms for slope
hist(bootstrap_betas[, 2], breaks = 30, freq = FALSE, xlim = range(bootstrap_betas[,
    2]) * c(0.95, 1.05), col = "lightgreen", main = "Bootstrap Sampling Distribution: Slope",
    xlab = "Slope Coefficients", ylab = "Density")
abline(v = normal_ci_slope, col = "red", lwd = 3, lty = 2)
legend("topright", "Slope PDF", lty = 1, bty = "n", fill = "lightgreen")
```




```
# Output results
cat("Normal Bootstrap CI for Intercept: [", normal_ci_intercept[1],
    ", ", normal_ci_intercept[2], "]\n")
```

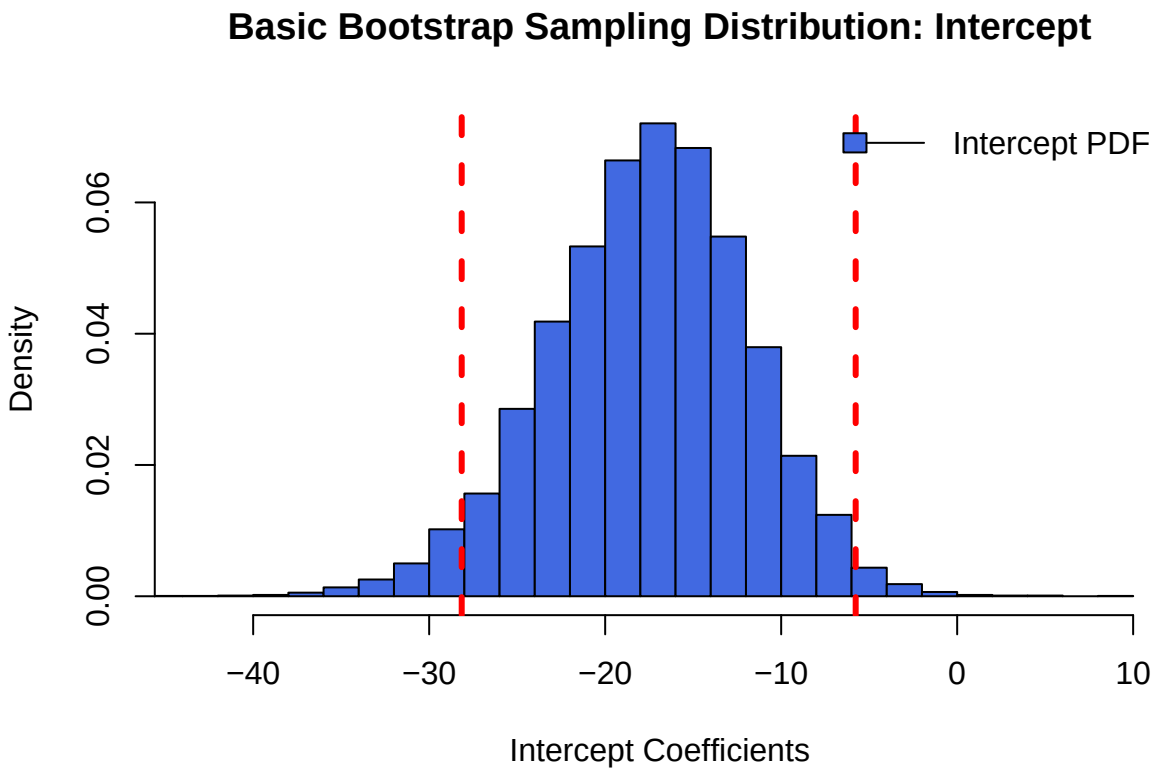
```
## Normal Bootstrap CI for Intercept: [ -28.73566 ,  -6.422525 ]
```

```
cat("Normal Bootstrap CI for Slope: [", normal_ci_slope[1], ", ",
    normal_ci_slope[2], "]\n")
```

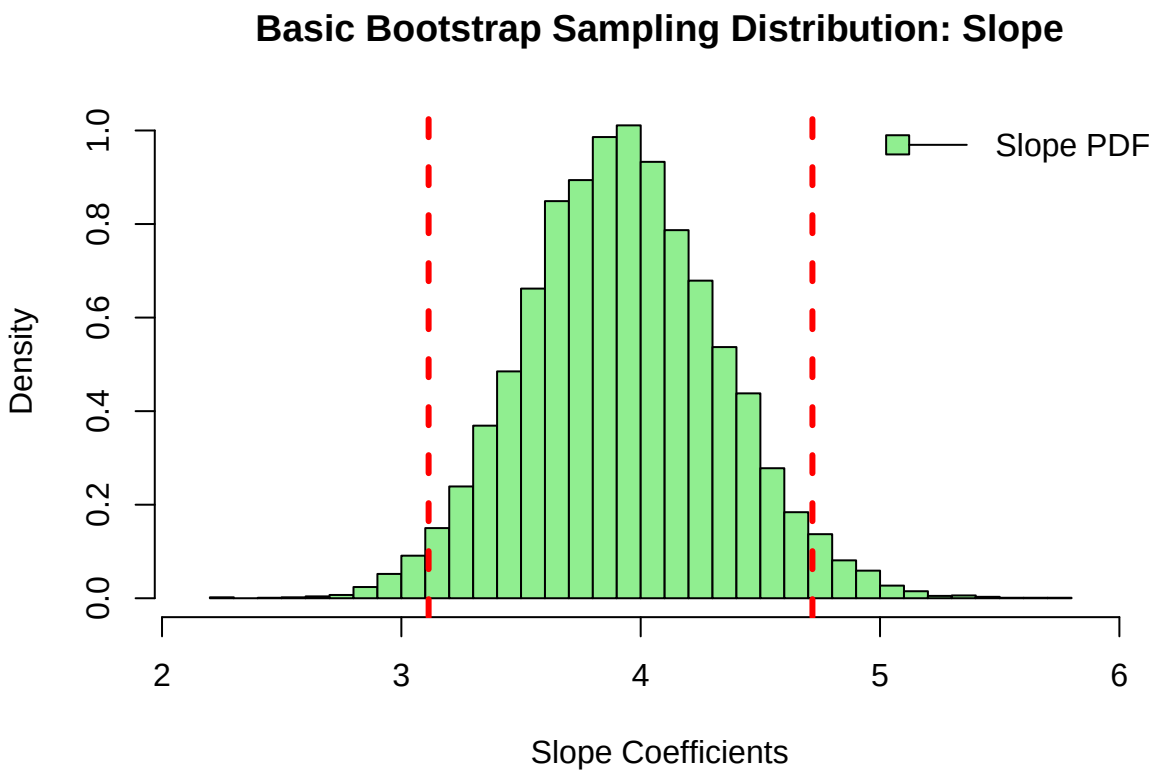
```
## Normal Bootstrap CI for Slope: [ 3.135305 ,  4.729513 ]
```

2. Basic Bootstrap CI

```
# Calculate Basic Bootstrap Confidence Intervals
basic_ci_intercept <- 2 * beta_initial[1] - quantile(bootstrap_betas[,
    1], probs = c(1 - alpha/2, alpha/2))
basic_ci_slope <- 2 * beta_initial[2] - quantile(bootstrap_betas[,
    2], probs = c(1 - alpha/2, alpha/2))

# Plotting the histograms for intercept
hist(bootstrap_betas[, 1], breaks = 30, freq = FALSE, xlim = range(bootstrap_betas[,
    1]) * c(0.95, 1.05), col = "royalblue", main = "Basic Bootstrap Sampling Distribution: Intercept",
    xlab = "Intercept Coefficients", ylab = "Density")
abline(v = basic_ci_intercept, col = "red", lwd = 3, lty = 2)
legend("topright", "Intercept PDF", lty = 1, bty = "n", fill = "royalblue")
```




```
# Plotting the histograms for slope
hist(bootstrap_betas[, 2], breaks = 30, freq = FALSE, xlim = range(bootstrap_betas[,
    2]) * c(0.95, 1.05), col = "lightgreen", main = "Basic Bootstrap Sampling Distribution: Slope",
    xlab = "Slope Coefficients", ylab = "Density")
abline(v = basic_ci_slope, col = "red", lwd = 3, lty = 2)
legend("topright", "Slope PDF", lty = 1, bty = "n", fill = "lightgreen")
```




```
# Output results
cat("Basic Bootstrap CI for Intercept: [", basic_ci_intercept[1],
    ", ", basic_ci_intercept[2], "]\n")
```

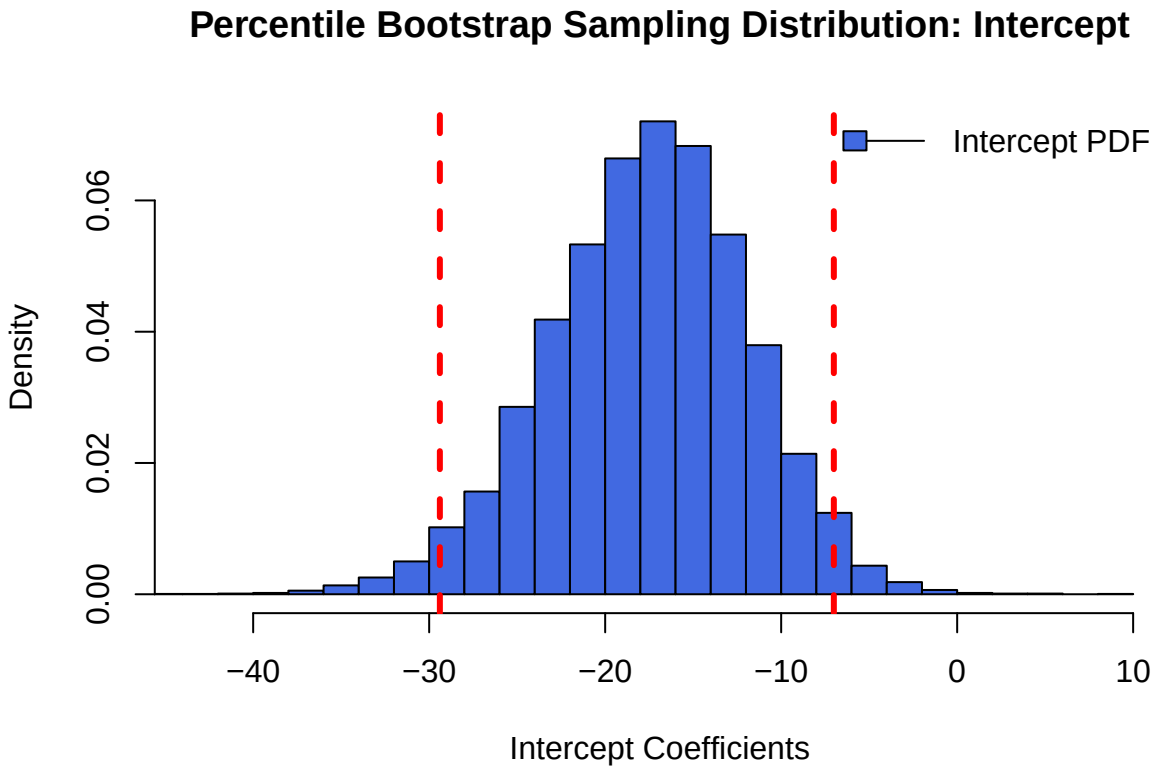
```
## Basic Bootstrap CI for Intercept: [ -28.15066 ,  -5.766208 ]
```

```
cat("Basic Bootstrap CI for Slope: [", basic_ci_slope[1], ", ",
    basic_ci_slope[2], "]\n")
```

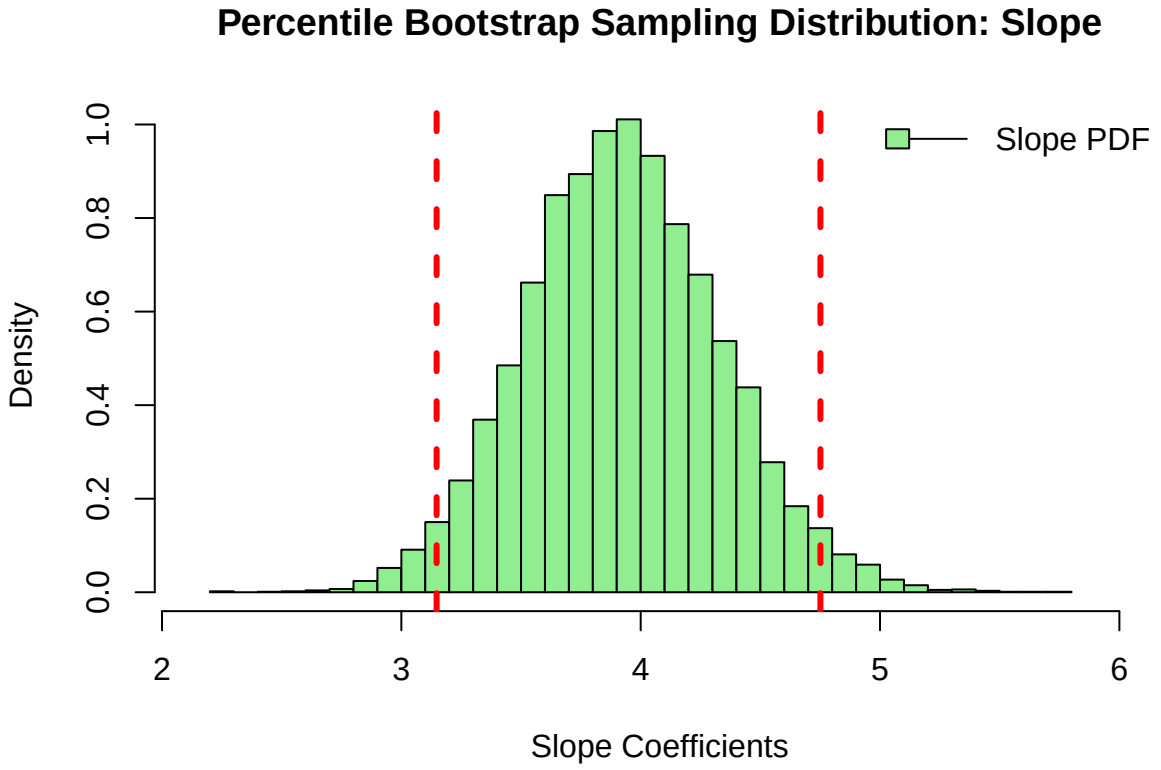
```
## Basic Bootstrap CI for Slope: [ 3.113828 ,  4.71747 ]
```

3. Percentile Bootstrap CI

```
# Calculate Percentile Bootstrap Confidence Intervals
percentile_ci_intercept <- quantile(bootstrap_betas[, 1], probs = c(alpha/2,
    1 - alpha/2))
percentile_ci_slope <- quantile(bootstrap_betas[, 2], probs = c(alpha/2,
    1 - alpha/2))

# Plotting the histograms for intercept
hist(bootstrap_betas[, 1], breaks = 30, freq = FALSE, xlim = range(bootstrap_betas[,
    1]) * c(0.95, 1.05), col = "royalblue", main = "Percentile Bootstrap Sampling Distribution: Intercep
    xlab = "Intercept Coefficients", ylab = "Density")
abline(v = percentile_ci_intercept, col = "red", lwd = 3, lty = 2)
legend("topright", "Intercept PDF", lty = 1, bty = "n", fill = "royalblue")
```




```
# Plotting the histograms for slope
hist(bootstrap_betas[, 2], breaks = 30, freq = FALSE, xlim = range(bootstrap_betas[,
    2]) * c(0.95, 1.05), col = "lightgreen", main = "Percentile Bootstrap Sampling Distribution: Slope"
    xlab = "Slope Coefficients", ylab = "Density")
abline(v = percentile_ci_slope, col = "red", lwd = 3, lty = 2)
legend("topright", "Slope PDF", lty = 1, bty = "n", fill = "lightgreen")
```




```
# Output results
cat("Percentile Bootstrap CI for Intercept: [", percentile_ci_intercept[1],
    ", ", percentile_ci_intercept[2], "]\n")
```

```
## Percentile Bootstrap CI for Intercept: [ -29.39198 ,  -7.007529 ]
```

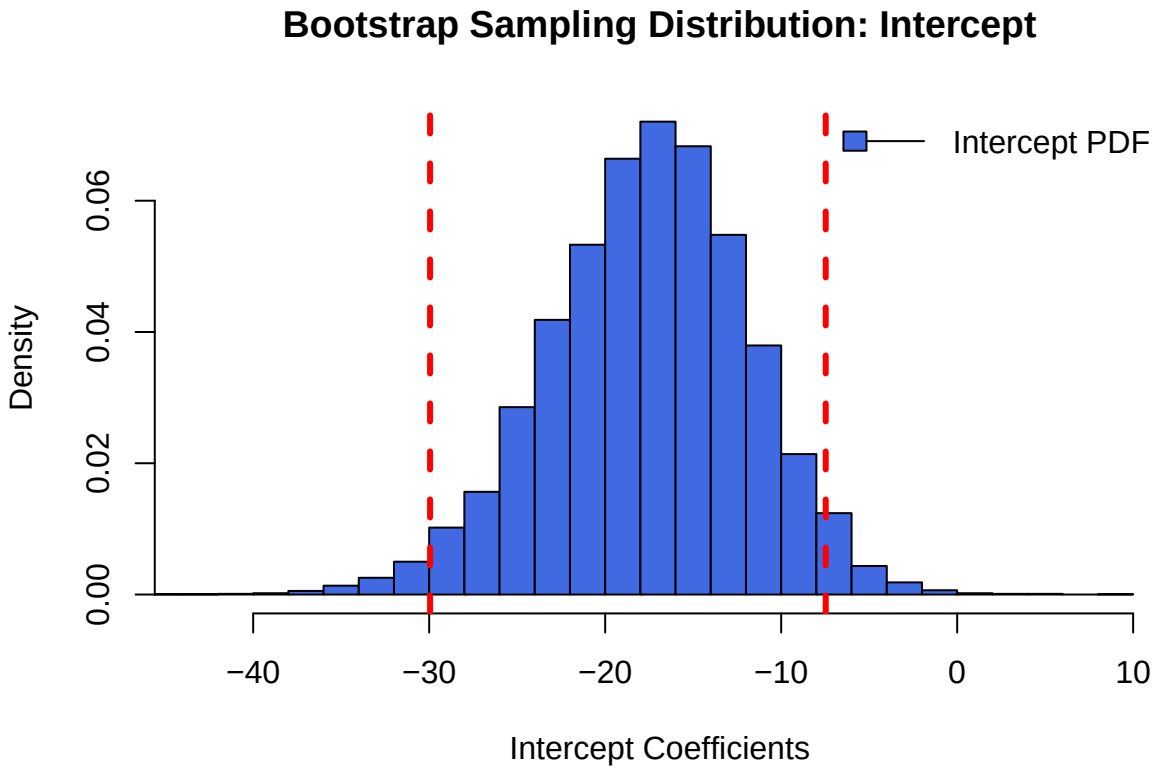
```
cat("Percentile Bootstrap CI for Slope: [", percentile_ci_slope[1],
    ", ", percentile_ci_slope[2], "]\n")
```

```
## Percentile Bootstrap CI for Slope: [ 3.147348 ,  4.75099 ]
```

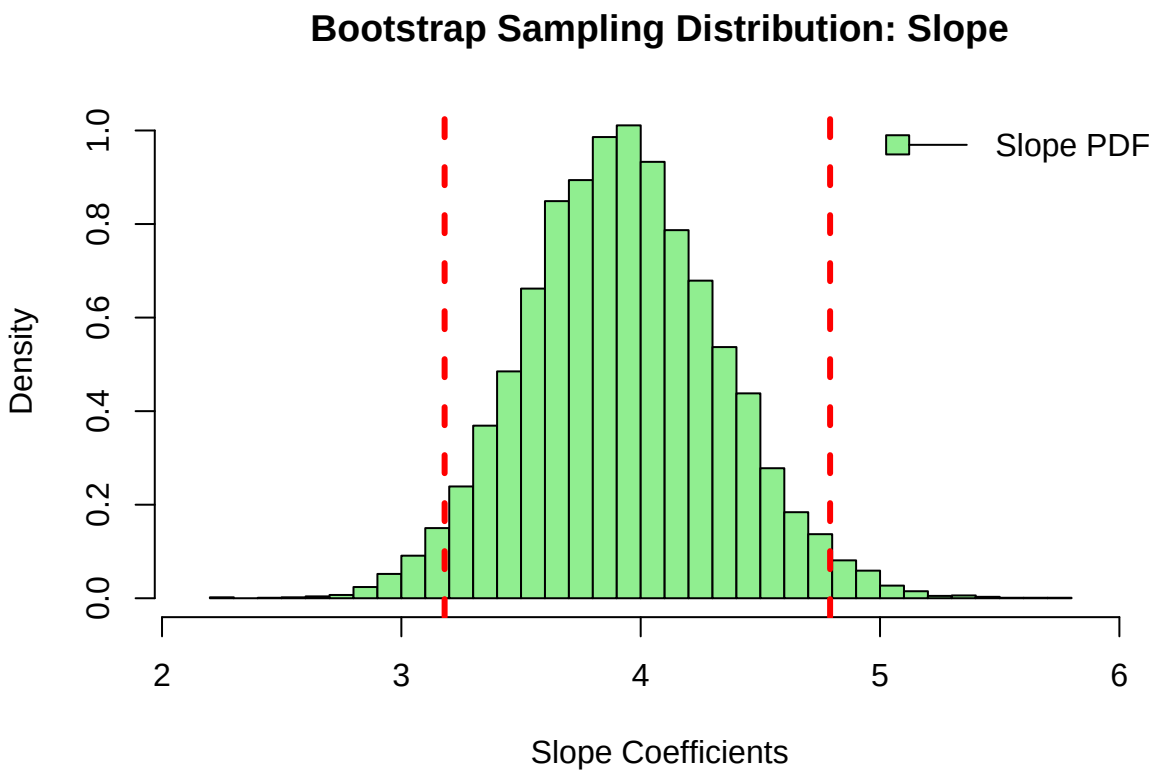
4. Bias corrected Bootstrap CI

```
# Calculate Bias Corrected Bootstrap Confidence Intervals
z0int <- qnorm(sum(bootstrap_betas[, 1] < beta_initial[1])/B)
z0slope <- qnorm(sum(bootstrap_betas[, 2] < beta_initial[2])/B)
A1int <- pnorm(2 * z0int + qnorm(alpha/2))
A2int <- pnorm(2 * z0int + qnorm(1 - alpha/2))
A1slope <- pnorm(2 * z0slope + qnorm(alpha/2))
A2slope <- pnorm(2 * z0slope + qnorm(1 - alpha/2))

bc_ci_intercept <- quantile(bootstrap_betas[, 1], probs = c(A1int,
    A2int))
bc_ci_slope <- quantile(bootstrap_betas[, 2], probs = c(A1slope,
    A2slope))

# Plotting the histograms for intercept
hist(bootstrap_betas[, 1], breaks = 30, freq = FALSE, xlim = range(bootstrap_betas[,
    1]) * c(0.95, 1.05), col = "royalblue", border = "black",
    main = "Bootstrap Sampling Distribution: Intercept", xlab = "Intercept Coefficients",
```

```
    ylab = "Density")
abline(v = bc_ci_intercept, col = "red", lwd = 3, lty = 2)
legend("topright", "Intercept PDF", lty = 1, bty = "n", fill = "royalblue")
```




```
# Plotting the histograms for slope
hist(bootstrap_betas[, 2], breaks = 30, freq = FALSE, xlim = range(bootstrap_betas[,
    2]) * c(0.95, 1.05), col = "lightgreen", border = "black",
    main = "Bootstrap Sampling Distribution: Slope", xlab = "Slope Coefficients",
    ylab = "Density")
abline(v = bc_ci_slope, col = "red", lwd = 3, lty = 2)
legend("topright", "Slope PDF", lty = 1, bty = "n", fill = "lightgreen")
```




```
# Output results
cat("Bias Corrected Bootstrap CI for Intercept: [", bc_ci_intercept[1],
    ", ", bc_ci_intercept[2], "]\n")
```

```
## Bias Corrected Bootstrap CI for Intercept: [ -29.94774 ,  -7.467932 ]
```

```
cat("Bias Corrected Bootstrap CI for Slope: [", bc_ci_slope[1],
    ", ", bc_ci_slope[2], "]\n")
```

```
## Bias Corrected Bootstrap CI for Slope: [ 3.180714 ,  4.791361 ]
```

```

# Function to calculate CI length and shape
calculate_length_shape <- function(ci, original_estimate) {
    length_ci <- diff(ci)
    # Calculate the shape using the provided formula
    shape_ci <- (ci[2] - original_estimate)/(original_estimate -
        ci[1])
    return(list(length = length_ci, shape = shape_ci))
}

# Calculate Length and Shape for each Bootstrap Type
normal_stats_intercept <- calculate_length_shape(normal_ci_intercept,
    beta_initial[1])
normal_stats_slope <- calculate_length_shape(normal_ci_slope,
    beta_initial[2])

basic_stats_intercept <- calculate_length_shape(basic_ci_intercept,
    beta_initial[1])
basic_stats_slope <- calculate_length_shape(basic_ci_slope, beta_initial[2])
```

```r
percentile_stats_intercept <- calculate_length_shape(percentile_ci_intercept,
    beta_initial[1])
percentile_stats_slope <- calculate_length_shape(percentile_ci_slope,
    beta_initial[2])

bc_stats_intercept <- calculate_length_shape(bc_ci_intercept,
    beta_initial[1])
bc_stats_slope <- calculate_length_shape(bc_ci_slope, beta_initial[2])

# Printed results below. No R code to display since it is
# just print functions and takes up too much space.
```

Calculate the length and shape of each type of Bootstrap CI and report them as well. (Shape $= \frac{C_U - \hat{\theta}}{\hat{\theta} - C_L}$**)**

```
## Normal Bootstrap CI for Intercept:
## Length: 22.31314
## Shape: 1
## Normal Bootstrap CI for Slope:
## Length: 1.594208
## Shape: 1
## Basic Bootstrap CI for Intercept:
## Length: 22.38445
## Shape: 1.117421
## Basic Bootstrap CI for Slope:
## Length: 1.603642
## Shape: 0.959051
## Percentile Bootstrap CI for Intercept:
## Length: 22.38445
## Shape: 0.8949181
## Percentile Bootstrap CI for Slope:
## Length: 1.603642
## Shape: 1.042697
## Bias Corrected Bootstrap CI for Intercept:
## Length: 22.4798
## Shape: 0.8174837
## Bias Corrected Bootstrap CI for Slope:
## Length: 1.610647
## Shape: 1.142688
```

- A value of 1 indicates that the bootstrap data is not skewed whereas a value that is less than 1 or greater than 1 will indicate a skew towards lower or higher values.

Discuss how you selected the number of bootstrap B replicates and comment on the results.

- I selected 10000 bootstrap replicates for the analysis as it maintains a balance between computational efficiency and statistical reliability. This number is large enough to ensure tha tthe bootstrap distribution approximates the true sampling distribution and reduces variability while also increasing the accuracy of the confidence intervals calculated.

Part(c): Compare lengths and shapes for each type of bootstrap CI you constructed in Parts(a) and (b) and comments on the results.

- For part a, the parametric bootstrap estimates have a length of about 26.4 for intercept and varies for slope where as the nonparametric bootstrap estimates from part b have a length of about 22.3 for intercept and varies for slope. As observed, the parametric bootstrap CIs tend to have a slightly longer lengths in comparison to non-parametric CIs. Additionally, shape values are relatively close to 1 in part a when compared to the shape values of part b. This may suggest that the parametric assumptions may be appropriate for the intercept. Both part a and part b have relatively close value to 1 in shape, indicating that most of these bootstrap confidence intervals are mostly symmetric. The non-parametric bootstrap tends to show more variation in the shape metrics, which can indicate that it might be more sensitive to the actual data distribution which is advantageous when the assumptions of the parametric models do not hold.