Assignment-5

Ashish Verma

2024-02-13

Part 1

Section 7.4 Problem #44 Solution:

```r
# Given values
s <- 2.81
n <- 9
df <- n - 1
alpha <- 0.05

# Critical values for chi-square distribution
chi_upper <- qchisq(1 - alpha/2, df)
chi_lower <- qchisq(alpha/2, df)

# Confidence interval for variance
variance_CI <- c((df * s^2) / chi_upper, (df * s^2) / chi_lower)

# Confidence interval for standard deviation
sigma_CI <- c(sqrt((df * s^2) / chi_upper), sqrt((df * s^2) / chi_lower))

# Print the results
cat("Confidence interval for variance:", variance_CI, "\n")
```

```
## Confidence interval for variance: 3.602534 28.98009
```

```r
cat("Confidence interval for standard deviation:", sigma_CI, "\n")
```

```
## Confidence interval for standard deviation: 1.898034 5.383316
```

Section 7.4 Problem #46 Solution:

```r
# Given data
data <- c(33.2, 41.8, 37.3, 40.2, 36.7, 39.1, 36.2, 41.8, 36, 35.2, 36.7, 38.9, 35.8, 35.2, 40.1)
n <- length(data)
alpha <- 0.05

# Kolmogorov-Smirnov test
ks_test <- ks.test(data, "pnorm")
```

```
## Warning in ks.test.default(data, "pnorm"): ties should not be present for the
## Kolmogorov-Smirnov test
```

```
D_stat <- ks_test$statistic
D_critical <- qnorm(1 - alpha/2) / sqrt(n)

# Print the results
cat("Kolmogorov-Smirnov Test:\n")
```

```
## Kolmogorov-Smirnov Test:
```

```
cat("D_stat:", D_stat, "\n")
```

```
## D_stat: 1
```

```
cat("D_critical:", D_critical, "\n")
```

```
## D_critical: 0.5060605
```

```
cat("H0 Rejected: Data dont follow normal distribution pattern", D_stat > D_critical, "\n\n")
```

```
## H0 Rejected: Data dont follow normal distribution pattern TRUE
```

```
# Given values for confidence interval
s <- 2.57
sample_variance <- s^2
confidence_level <- 0.95
df <- n - 1

# Critical value for chi-square distribution
chi_critical <- qchisq((1 - confidence_level)/2, df)

# Confidence interval for population standard deviation
upper_bound <- sqrt((n - 1) * sample_variance / chi_critical)

# Print the results
cat("Confidence Interval for Population Standard Deviation:\n")
```

```
## Confidence Interval for Population Standard Deviation:
```

```
cat("Upper Bound:", upper_bound, "\n")
```

```
## Upper Bound: 4.053144
```

Part 2

```r
# Load the data from URL
url <- "https://archive.ics.uci.edu/static/public/1/abalone.zip"

# Specify the destination folder for the downloaded ZIP file
zip_file <- "abalone.zip"

# Download the ZIP file
download.file(url, zip_file)

# Unzip the file
unzip(zip_file)

column_names <- c("Sex", "Length", "Diameter", "Height", "Whole weight", "Shucked weight",
                   "Viscera weight", "Shell weight", "Rings")
# Read CSV with schema
data <- read.csv("abalone.data", header = FALSE, col.names = column_names)
# Load necessary libraries
library(boot)

# Assuming you have downloaded the data and saved it as 'abalone.data'
# Adjust the file path accordingly

url <- "https://archive.ics.uci.edu/static/public/1/abalone.zip"

# Specify the destination folder for the downloaded ZIP file
zip_file <- "abalone.zip"

# Download the ZIP file
download.file(url, zip_file)

# Unzip the file
unzip(zip_file)

column_names <- c("Sex", "Length", "Diameter", "Height", "Whole weight", "Shucked weight",
                   "Viscera weight", "Shell weight", "Rings")
# Read CSV with schema
abalone_data <- read.csv("abalone.data", header = FALSE, col.names = column_names)

# Calculate the population median
# Calculate the population median
population_median <- median(abalone_data$Length, na.rm = TRUE)
population_median
```

```
## [1] 0.545
```

```r
num_replications <- 5000

# Standard Normal Bootstrap Confidence Interval
standard_normal_boot <- boot(abalone_data$Length, statistic = function(x, i) median(x[i]), R = num_repl
standard_normal_ci <- boot.ci(standard_normal_boot, type = "norm", conf = 0.9)
standard_normal_ci
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 5000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = standard_normal_boot, conf = 0.9, type = "norm")
##
## Intervals :
## Level      Normal
## 90%   ( 0.5411,  0.5520 )
## Calculations and Intervals on Original Scale
```

```
# Basic Bootstrap Confidence Interval
basic_boot <- boot(abalone_data$Length, statistic = function(x, i) median(x[i]), R = num_replications)
basic_ci <- boot.ci(basic_boot, type = "basic", conf = 0.9)
basic_ci
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 5000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = basic_boot, conf = 0.9, type = "basic")
##
## Intervals :
## Level      Basic
## 90%   ( 0.54,  0.55 )
## Calculations and Intervals on Original Scale
```

```
# Percentile Bootstrap Confidence Interval
percentile_boot <- boot(abalone_data$Length, statistic = function(x, i) median(x[i]), R = num_replicati
percentile_ci <- boot.ci(percentile_boot, type = "perc", conf = 0.9)
percentile_ci
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 5000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = percentile_boot, conf = 0.9, type = "perc")
##
## Intervals :
## Level     Percentile
## 90%   ( 0.54,  0.55 )
## Calculations and Intervals on Original Scale
```

```
# t Bootstrap Confidence Interval
t_boot <- boot(abalone_data$Length, statistic = function(x, i) median(x[i]), R = num_replications)
t_ci <- boot.ci(t_boot, type = "bca", conf = 0.9)
t_ci
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 5000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = t_boot, conf = 0.9, type = "bca")
```

```
##
## Intervals :
## Level       BCa
## 90%   ( 0.535,  0.545 )
## Calculations and Intervals on Original Scale
```

```
# BCa Bootstrap Confidence Interval
bca_boot <- boot(abalone_data$Length, statistic = function(x, i) median(x[i]), R = num_replications)
bca_ci <- boot.ci(bca_boot, type = "bca", conf = 0.9)
bca_ci
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 5000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = bca_boot, conf = 0.9, type = "bca")
##
## Intervals :
## Level       BCa
## 90%   ( 0.535,  0.545 )
## Calculations and Intervals on Original Scale
```

```
# Function to determine if the true median is inside the interval
true_median_inside_interval <- function(ci, true_median) {
  all(!is.na(ci)) && ci[1] <= true_median && true_median <= ci[2]
}

# Function to compute the fraction of samples inside the interval
fraction_inside_interval <- function(ci, true_median) {
  mean(sapply(ci, function(x) true_median_inside_interval(x, true_median)))
}

# True population median
true_median <- median(abalone_data$Length)

# Number of replications
num_replications <- 5000  # Adjust as needed

# Standard Normal Bootstrap Confidence Interval
standard_normal_boot <- boot(abalone_data$Length, statistic = function(x, i) median(x[i]), R = num_repli
standard_normal_ci <- boot.ci(standard_normal_boot, type = "norm", conf = 0.9)
standard_normal_fraction_inside <- fraction_inside_interval(standard_normal_ci$normal, true_median)

# Basic Bootstrap Confidence Interval
basic_boot <- boot(abalone_data$Length, statistic = function(x, i) median(x[i]), R = num_replications)
basic_ci <- boot.ci(basic_boot, type = "basic", conf = 0.9)
basic_fraction_inside <- fraction_inside_interval(basic_ci$basic, true_median)

# Percentile Bootstrap Confidence Interval
percentile_boot <- boot(abalone_data$Length, statistic = function(x, i) median(x[i]), R = num_replicati
percentile_ci <- boot.ci(percentile_boot, type = "perc", conf = 0.9)
percentile_fraction_inside <- fraction_inside_interval(percentile_ci$percent, true_median)
```

```r
# t Bootstrap Confidence Interval
t_boot <- boot(abalone_data$Length, statistic = function(x, i) median(x[i]), R = num_replications)
t_ci <- boot.ci(t_boot, type = "bca", conf = 0.9)
t_fraction_inside <- fraction_inside_interval(t_ci$bca, true_median)

# BCa Bootstrap Confidence Interval
bca_boot <- boot(abalone_data$Length, statistic = function(x, i) median(x[i]), R = num_replications)
bca_ci <- boot.ci(bca_boot, type = "bca", conf = 0.9)
bca_fraction_inside <- fraction_inside_interval(bca_ci$bca, true_median)

# Display fraction of samples where true median is inside each interval
results <- data.frame(
  Method = c("Standard Normal", "Basic", "Percentile", "t", "BCa"),
  FractionInside = c(standard_normal_fraction_inside, basic_fraction_inside, percentile_fraction_inside
)
results
```

```
##            Method FractionInside
## 1 Standard Normal             NA
## 2           Basic             NA
## 3      Percentile             NA
## 4               t             NA
## 5             BCa             NA
```