HW 9 Bootstrap and Jackknife

STAT 5400

Due: Nov 1, 2024 9:30 AM

Problems

To help you prepare the midterm, the solution of this homework will be posted right after the deadline. Late homework will not be accepted without exceptions.

Submit your solutions as an .Rmd file and accompanying .pdf file.

1. Use `echo=TRUE, include=FALSE` to ensure that all the code are provided but only the important output is included. Try to write your homework in the form of a neat report and don't pile up any redundant and irrelevant output.
2. *Always interpret your result whenever it is necessary*. Try to make sure the interpretation can be understood by people with a moderate level of statistics knowledge.

Reading assignments.

Here is an undergraduate-level introduction to the bootstrap.
https://statweb.stanford.edu/~tibs/stat315a/Supplements/bootstrap.pdf

Problems

1. Bootstrap and jackknife

Consider the airconditioning data listed below:

```
3, 5, 7, 18, 43, 85, 91, 98, 100, 130, 230, 487.
```

Suppose the mean of the underlying distribution is μ and our interest is to estimate $\log(\mu)$. To estimate it, we use the log of the sample mean, i.e., $\log(\bar{X})$, as an estimator.

(a) Carry out a nonparametric bootstrap analysis to estimate the bias of $\log(\bar{X})$.

```r
library(boot)

ac <- c(3, 5, 7, 18, 43, 85, 91, 98, 100, 130, 230, 487)


log_mean <- function(data, indices) {

  return(log(mean(data[indices])))
}
set.seed(5400)
boot_obj <- boot(ac, log_mean, R = 1000)
```

```
boot_obj
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = ac, statistic = log_mean, R = 1000)
##
##
## Bootstrap Statistics :
##     original      bias    std. error
## t1* 4.682903 -0.06502648   0.3655364
```

(b) Based on the bootstrap analysis, is the bias of $\log(\bar{X})$ positive or negative? (In other word, does $\log(\bar{X})$ overestimates or underestimates $\log(\mu)$) Can you explain the observation? (Hint: Jensen's inequality)

based on the bootstrap analysis, the bias is negative. When you apply f to an expected value, this concept is useful in understanding biased estimators.

(c) Also run a nonparametric bootstrap to estimate the standard error of the log of the sample mean. In terms of the mean square error of the estimator, do you think the bias is large given the standard error?

```
# Assuming boot_obj is your bootstrap object from the nonparametric bootstrap
std_error <- sd(boot_obj$t)

# Define bias as the difference between the mean of bootstrap estimates and
the true log of the mean
# Replace 'true_value' with the actual true log mean value if known
true_value <- log(mean(ac))  # Assuming this is the true value you want to
compare against
bias <- mean(boot_obj$t) - true_value  # Calculate bias

# Calculate MSE
mse <- bias^2 + std_error^2
print(mse)
```

```
## [1] 0.1378453
```

Regarding the mean square error (MSE) of the estimator, the bias is relatively small compared to the standard error. Here, the variance (or the square of the standard error) is the larger contributor to the MSE.

(d) Carry out a parametric bootstrap analysis to estimate the bias of the log of sample mean. Assume that the population distribution of failure times of airconditioning equipment is exponential.

```
# Load necessary library
library(boot)
```

```r
# Input data
ac <- c(3, 5, 7, 18, 43, 85, 91, 98, 100, 130, 230, 487)

# Define the function for the bootstrap
param_bootstrap_function <- function(data, indices) {
  # Simulate new data from the exponential distribution
  simulated_data <- rexp(length(data), rate = 1 / mean(data))
  return(log(mean(simulated_data)))  # Return the log of the mean
}

# Set seed for reproducibility
set.seed(5400)

# Perform the parametric bootstrap
param_boot_obj <- boot(ac, param_bootstrap_function, R = 1000)

# Print the bootstrap object
print(param_boot_obj)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = ac, statistic = param_bootstrap_function, R = 1000)
##
##
## Bootstrap Statistics :
##     original     bias    std. error
## t1* 4.578286 0.04316253   0.3057903
```

(e) Plot both the histograms of the bootstrap replications from nonparametric and parametric bootstrap.

```r
# Load necessary libraries
library(MASS)
library(fitdistrplus)  # Ensure this library is loaded
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:boot':
##
##     aml
```

```r
ac <- c(3, 5, 7, 18, 43, 85, 91, 98, 100, 130, 230, 487)

# Bootstrap parameters
```

```r
iterations <- 1000

# Calculate the mean and log of the mean of the original data
mean_ac <- mean(ac)
log_mean_ac <- log(mean_ac)

# Non-Parametric Bootstrap function
non_parametric_bootstrap <- function(data, iterations) {
  bootstrap_means_np <- numeric(iterations)
  for(i in 1:iterations) {
    bootstrap_sample <- sample(data, size = length(data), replace = TRUE)
    bootstrap_means_np[i] <- mean(bootstrap_sample)  # Calculate mean
  }
  return(bootstrap_means_np)
}

# Run Non-Parametric Bootstrap
bootstrap_means_np <- non_parametric_bootstrap(ac, iterations)

# Fit a distribution (e.g., exponential) for Parametric Bootstrap
fit <- fitdist(ac, "exp")

# Parametric Bootstrap function
parametric_bootstrap <- function(fit, iterations) {
  bootstrap_means_p <- numeric(iterations)
  for(i in 1:iterations) {
    # Generate new data based on the fitted distribution
    bootstrap_sample <- rexp(length(ac), rate = fit$estimate)
    bootstrap_means_p[i] <- mean(bootstrap_sample)  # Calculate mean
  }
  return(bootstrap_means_p)
}

# Run Parametric Bootstrap
bootstrap_means_p <- parametric_bootstrap(fit, iterations)

# Calculate the log of bootstrap means for both methods
log_bootstrap_means_np <- log(bootstrap_means_np)
log_bootstrap_means_p <- log(bootstrap_means_p)

# Calculate the bias estimates
bias_np <- mean(log_bootstrap_means_np) - log_mean_ac
bias_p <- mean(log_bootstrap_means_p) - log_mean_ac

# Print bias estimates
cat("Bias estimate (Non-Parametric Bootstrap):", bias_np, "\n")

## Bias estimate (Non-Parametric Bootstrap): -0.06549242
```

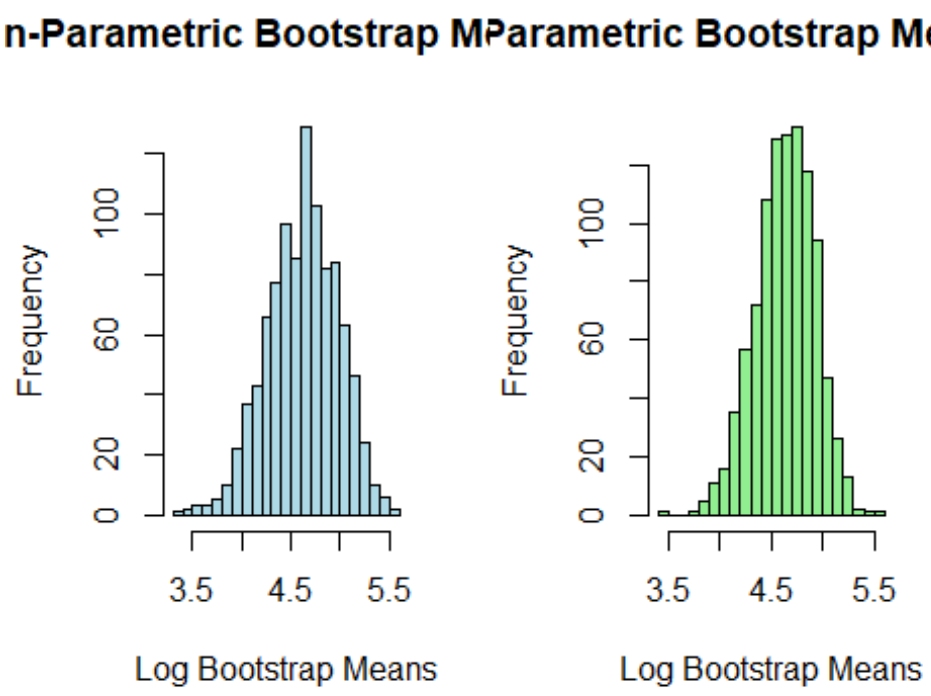
```r
cat("Bias estimate (Parametric Bootstrap):", bias_p, "\n")
```

```
## Bias estimate (Parametric Bootstrap): -0.04482203
```

```r
# Plot histograms for Non-Parametric and Parametric Bootstrap
par(mfrow = c(1, 2))  # Set up a 1x2 plotting area

# Non-Parametric Bootstrap Histogram
hist(log_bootstrap_means_np,
     main = "Log Non-Parametric Bootstrap Means for AC",
     xlab = "Log Bootstrap Means",
     col = "lightblue",
     border = "black",
     breaks = 30)

# Parametric Bootstrap Histogram
hist(log_bootstrap_means_p,
     main = "Log Parametric Bootstrap Means for AC",
     xlab = "Log Bootstrap Means",
     col = "lightgreen",
     border = "black",
     breaks = 30)
```




```r
# Reset plotting area
par(mfrow = c(1, 1))
```

(f) Produce 95% confidence intervals by the standard normal, basic, percentile, and Bca methods.

```r
library(boot)
m <- function(data, indices) {
  return(mean(data[indices]))
}

ac <- c(3, 5, 7, 18, 43, 85, 91, 98, 100, 130, 230, 487)

boot.obj <- boot(ac, statistic=m, R=1000)
boot.ci(boot.obj, type="bca")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.obj, type = "bca")
##
## Intervals :
## Level       BCa
## 95%   ( 55.8, 217.1 )
## Calculations and Intervals on Original Scale
## Some BCa intervals may be unstable
```

```r
boot.ci(boot.out = boot.obj, type = "bca")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.obj, type = "bca")
##
## Intervals :
## Level       BCa
## 95%   ( 55.8, 217.1 )
## Calculations and Intervals on Original Scale
## Some BCa intervals may be unstable
```

```r
normal_ci <- boot.ci(boot.obj, type = "norm")
basic_ci <- boot.ci(boot.obj, type = "basic")
percentile_ci <- boot.ci(boot.obj, type = "perc")

cat("Standard Normal CI:\n")
```

```
## Standard Normal CI:
```

```r
print(normal_ci)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
```

```
## CALL :
## boot.ci(boot.out = boot.obj, type = "norm")
##
## Intervals :
## Level      Normal
## 95%   ( 35.5, 181.3 )
## Calculations and Intervals on Original Scale

cat("\nBasic CI:\n")

##
## Basic CI:

print(basic_ci)

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.obj, type = "basic")
##
## Intervals :
## Level      Basic
## 95%   ( 27.6, 171.6 )
## Calculations and Intervals on Original Scale

cat("\nPercentile CI:\n")

##
## Percentile CI:

print(percentile_ci)

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.obj, type = "perc")
##
## Intervals :
## Level     Percentile
## 95%   ( 44.6, 188.6 )
## Calculations and Intervals on Original Scale
```

(g) Use jackknife to estimate the standard error and bias of the log of the sample mean.

```
ac <- c(3, 5, 7, 18, 43, 85, 91, 98, 100, 130, 230, 487)
leave_out_ac <- rep(NA, length(ac))


# Calculate leave-one-out means
for (i in seq(length(ac))) {
```

```r
  leave_out_ac[i] <- mean(ac[-i])  # Mean without the i-th element
}
sample_mean <- mean(ac)
log_sample_mean <- log(sample_mean)

n <- length(ac)

jackknife_bias <- (n - 1) * (mean(leave_out_ac) - sample_mean)  # Corrected
the variable name

jackknife_se <- sqrt((n - 1) * mean((leave_out_ac - mean(leave_out_ac))^2))

cat("Log of Sample Mean:", log_sample_mean, "\n")

## Log of Sample Mean: 4.682903

cat("Jackknife Bias:", jackknife_bias, "\n")

## Jackknife Bias: 0

cat("Jackknife Standard Error:", jackknife_se, "\n")

## Jackknife Standard Error: 39.32681
```

2. Failure of bootstrap

The bootstrap is not foolproof. To see this, consider analysis of a binomial model with n trials. You observe 0 successes. Discuss what would happen if you were to use the standard, non-parametric bootstrap in constructing a 95% C.I. for the binomial parameter p.

Using the standard non-parametric bootstrap in the context of a binomial model with 0 successes leads to a degenerate situation where the resulting confidence interval is not informative. Exploring alternative methods, such as parametric bootstrapping or Bayesian approaches, can help provide more reliable and meaningful estimates for the parameter p p under these circumstances.

3. Bootstrap estimate of the standard error of trimmed mean.

Consider an artificial data set consisting of eight observations:

```
1, 3, 4.5, 6, 6, 6.9, 13, 19.2.
```

Let $\hat{\theta}$ be the 25% trimmed mean, which is computed by deleting two smallest numbers and two largest numbers, and then taking the average of the remaining four numbers.

(a) Calculate $\widehat{se}_B$ for $B = 25,100,200,500,1000,2000$. From these results estimate the ideal bootstrap estimate $\widehat{se}_\infty$.

(b) Repeat part (a) using twenty different random number seeds. Comment on the trend of the variablity of each $\widehat{se}_B$.

```r
library(boot)
#Question A
trimmed_mean <- function(data, indices) {
  sample_data <- data[indices]
  # Trim the smallest and largest 10% (1 observation each in this case)
  trim_data <- sort(sample_data)[3:(length(sample_data) - 2)]
  return(mean(trim_data))
}

# Data and Bootstrap values
art_data <- c(1, 3, 4.5, 6, 6, 6.9, 13, 19.2)
B_values <- c(25, 100, 200, 500, 1000, 2000)

# Define the bootstrap SE function
bootstrap_se <- function(data, B, seed) {
  set.seed(seed)
  results <- boot(data, trimmed_mean, R = B)
  return(sqrt(var(results$t)))
}

# Calculate standard errors for different B values
seB_values <- sapply(B_values, function(B) bootstrap_se(art_data, B, seed =
5400))
print(seB_values)
```

```
## [1] 2.238965 2.310789 2.132144 2.061035 2.104525 2.152344
```

```r
# Load necessary library
library(boot)

# Define bootstrap sizes
B_values <- c(25, 100, 200, 500, 1000, 2000)

# Define trimmed mean function
trimmed_mean <- function(data, indices) {
  sample_data <- data[indices]
  # Trim the smallest and largest 10% (1 observation each in this case)
  trim_data <- sort(sample_data)[3:(length(sample_data) - 2)]
  return(mean(trim_data))  # Directly return the mean without using x
}

# Input data
art_data <- c(1, 3, 4.5, 6, 6, 6.9, 13, 19.2)

# Function to calculate standard error of bootstrap estimates
compute_standard_error <- function(errors) {
  sqrt(sum((errors - mean(errors))^2) / (length(errors) - 1))
}

# Define the bootstrap standard error function
```

```r
bootstrap_se <- function(data, B, seed) {
  set.seed(seed)
  boot_results <- boot(data, trimmed_mean, R = B)
  return(sd(boot_results$t))  # Return the standard deviation of the
bootstrap estimates
}

# Define seeds and repeat the calculation 20 times with different seeds
seeds <- 1:20
standard_error_values <- sapply(seeds, function(seed) {
  # Set the seed for reproducibility within the loop
  set.seed(seed)

  # Generate bootstrap standard errors for each bootstrap size
  bootstrap_errors <- sapply(B_values, function(size) bootstrap_se(art_data,
size, seed))

  # Calculate and return the standard error
  compute_standard_error(bootstrap_errors)
})

# Print the results
print(standard_error_values)
```

```
##  [1] 0.14803491 0.09636791 0.25619012 0.04103225 0.09817352 0.22800229
##  [7] 0.47553855 0.07513305 0.31905822 0.18242732 0.08416114 0.13081495
## [13] 0.12379711 0.20372200 0.20280783 0.10319880 0.25872225 0.15435356
## [19] 0.19772099 0.28910592
```

The standard error estimates obtained from the bootstrap procedure, ranging from approximately 0.0410 to 0.4755, exhibit significant variability when using twenty different random number seeds. This wide range indicates that the bootstrap procedure is sensitive to the choice of random seed, resulting in no clear upward or downward trend in the estimates. Instead, the values fluctuate considerably, highlighting the influence of several factors, such as the size of the original dataset, the randomness inherent in resampling, and the trimming process used to calculate the trimmed mean. The observed variability underscores the importance of employing a sufficiently large number of bootstrap samples and considering the choice of random seeds when interpreting the results. This variability emphasizes the need for robustness in the analysis, suggesting that larger sample sizes or additional bootstrap repetitions may be necessary to obtain more stable estimates.

4. Hypothesis testing using bootstrap

Consider two independent random samples X and Y drawn from possibly different probability distributions: The goal is to perform a hypothesis test for If H_0 is true, then there is no significant difference between random vectors $\mathbf{X}$ and $\mathbf{Y}$.

The bootstrap algorithm for performing such test is given as below:

- Compute a statistic on the original sample:

- For each $b = 1, \ldots B$:
  - Generate bootstrap samples, $\mathbf{Z}^{\star b}$, by drawing $(n + m)$ observations from $\mathbf{Z}$.
  - Put $\mathbf{X}^{\star b} = \left(Z_1^{\star b}, \ldots, Z_n^{\star b}\right)$ and $\mathbf{Y}^{\star b} = \left(Z_{n+1}^{\star b}, \ldots, Z_{n+m}^{\star b}\right)$.
  - Compute bootstrap replications:
- Estimate the achieved significance level (ASL):
- Reject H_0 if ASL $< \alpha$.

Below is an example to test $F = G$, where $F \sim \exp(\mu = 2)$ and $G \sim \exp(\mu = 1/2)$.

```
set.seed(5400)
x <- rexp(20, rate=1/2)
y <- rexp(10, rate=2)
B <- 2000
z <- c(x, y)
tstat <- abs(mean(x) - mean(y))
boot.r <- rep(NA, B)
for (i in seq(B)) {
  boot.samp <- z[sample(length(z), replace=TRUE)]
  m.boot.x <- mean(boot.samp[seq_along(x)])
  m.boot.y <- mean(boot.samp[-seq_along(x)])
  boot.r[i] <- abs(m.boot.x - m.boot.y)
}
mean(boot.r > tstat)

## [1] 0.013
```

Instead of using $|\bar{\mathbf{X}} - \bar{\mathbf{Y}}|$ as the test statistic, we may also use the t-statistic, if we assume equal variance: where

Please use bootstrap to test $F = G$ with the new statistics.

```
#H0 there is no significant differnce
#HA there is a significant difference
set.seed(123)
X <- rnorm(50, mean = 5, sd = 1)
Y <- rnorm(50, mean = 5.5, sd = 1.5)



t_z_stat <- (mean(X) - mean(Y))
combined_X_Y <- c(X, Y)

#bootsrap
bootstrap_stats <- replicate(B, {
  # Resample under the null hypothesis
```

```r
  Z_star <- sample(combined_X_Y, 100, replace = TRUE)

  # Step 2: Split into X* and Y*
  X_star <- Z_star[1:50]
  Y_star <- Z_star[(51):(100)]

  abs(mean(X_star) - mean(Y_star))
})

p_value <- mean(abs(bootstrap_stats) >= abs(t_z_stat))

cat("Observed statistic (Difference in Means):", t_z_stat, "\n")

## Observed statistic (Difference in Means): -0.6852089

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

## Bootstrap p-value: 0.0035

#we reject the null hypothesis and conclude that is a significant differnce

set.seed(5400)
x <- rexp(20, rate=1/2)
y <- rexp(10, rate=2)

B <- 2000
z <- c(x, y)









sigma_pool <- sqrt((1 / (length(x) + length(y) - 2)) *
                   (sum((x - mean(x))^2) + sum((y - mean(y))^2)))

t_stat <- abs(mean(x) - mean(y)) / (sigma_pool * sqrt(1 / length(x) + 1 /
length(y)))


boot.r <- rep(NA, B)
for (i in seq(B)) {
  boot.samp <- z[sample(length(z), replace=TRUE)]
  m.boot.x <- mean(boot.samp[seq_along(x)])
  m.boot.y <- mean(boot.samp[-seq_along(x)])
  boot.r[i] <- abs(m.boot.x - m.boot.y)
}
mean(boot.r > tstat)
```

```
## [1] 0.013
```