HW 9 Bootstrap and Jackknife

Dev Narayan Baiju and Yu Chao Huang

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.

```
acdata <- c(3, 5, 7, 18, 43, 85, 91, 98, 100, 130, 230, 487)
samp_estim <- log(mean(acdata))
print(samp_estim)
```

```
## [1] 4.682903
```

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

```r
set.seed(5400)
B <- 1000
n <- length(acdata)
bootestim_nonp <- numeric(B)
for(i in 1:B){
  bootsample <- sample(acdata, length(acdata), replace = TRUE)
  bootestim_nonp[i] <- log(mean(bootsample))
}
boot_bias <- mean(bootestim_nonp) - samp_estim
print(boot_bias)
```

```
## [1] -0.06495026
```

(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). The bias that was obtained was negative. In other words, $\log(\bar{X})$ overestimates $\log(\mu)$.

(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?

```r
boot_se <- sd(bootestim_nonp)
print(paste("Standard error is", boot_se, sep = " "))
```

```
## [1] "Standard error is 0.364934248480577"
```

```r
print(paste("Squared bias is", boot_bias, sep = " "))
```

```
## [1] "Squared bias is -0.0649502622200897"
```

Comparing the both, the standard error is larger than the bias.

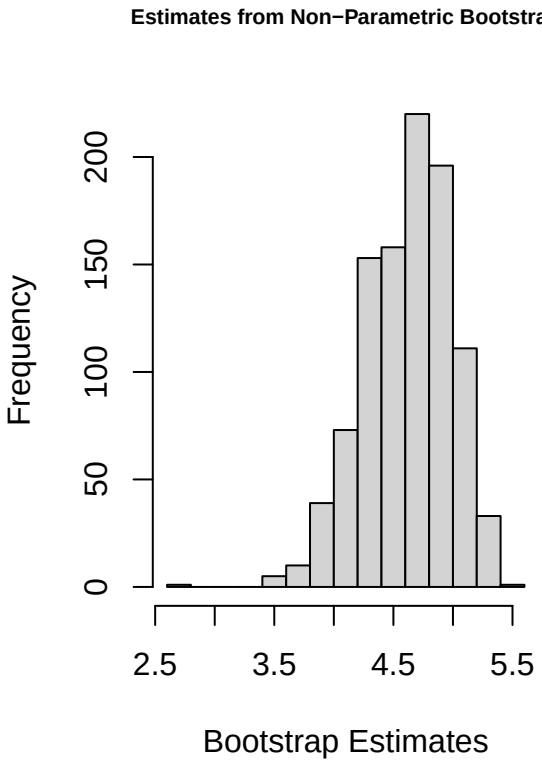
(d) Carry out a parametric bootstrap analysis to estimate the bias of the log of sample mean. Assume that the population distribution of failure times of airconditioning equipment is exponential.

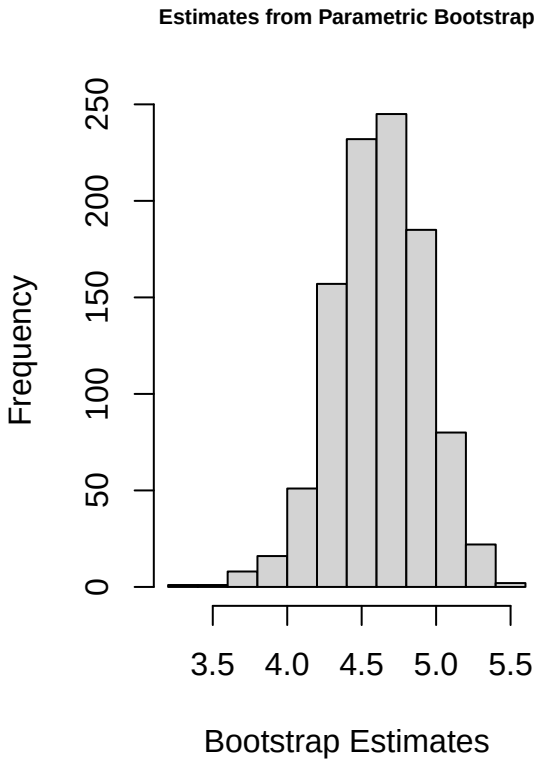
```r
acdata_mean <- mean(acdata)
B <- 1000
bootsample <- replicate(B, rexp(n, 1/acdata_mean))
bootestim_p <- apply(bootsample, 2, function(x) log(mean(x)))
boot_bias <- mean(bootestim_p) - samp_estim
print(boot_bias)
```

```
## [1] -0.06177025
```

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

```r
par(mfrow = c(1,2))
hist(bootestim_nonp, xlab = "Bootstrap Estimates", ylab = "Frequency",
     main = "Estimates from Non-Parametric Bootstrap", cex.main = 0.7)
hist(bootestim_p, xlab = "Bootstrap Estimates", ylab = "Frequency",
     main = "Estimates from Parametric Bootstrap", cex.main = 0.7)
```





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

```
set.seed(5400)
B <- 1000
n <- length(acdata)
bootestim_nonp <- numeric(B)
for(i in 1:B){
  bootsample <- sample(acdata, length(acdata), replace = TRUE)
  bootestim_nonp[i] <- log(mean(bootsample))
}
boot_se <- sd(bootestim_nonp)
#standard normal CI
print(samp_estim + c(-1, 1)*qnorm(0.975)*boot_se)
```

```
## [1] 3.967645 5.398161
```

```
#basic CI
boot.quant <- quantile(bootestim_nonp, c(0.975, 0.025), type = 6)
print(2*samp_estim - boot.quant)
```

```
##    97.5%     2.5%
## 4.113227 5.484198
```

```
#percentile CI
print(quantile(bootestim_nonp, c(0.025, 0.975), type = 6))
```

```
##     2.5%    97.5%
## 3.881607 5.252578
```

```r
#BCa CI
library(boot)
estim_func <- function(xlist, indices){
  return(log(mean(xlist[indices])))
}
boot.obj <- boot(data = acdata, statistic = estim_func, R = 1000)
print(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%   ( 4.035,  5.370 )
## Calculations and Intervals on Original Scale
## Some BCa intervals may be unstable
```

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

```r
jack.log <- rep(NA, n)
for (i in seq(n)){
  jack.log[i] <- log(mean(acdata[-i]))
}
(jack.bias <- (n-1)*(mean(jack.log) - samp_estim))
```

```
## [1] -0.07889564
```

```r
(jack.se <- sqrt((n-1)/n * sum((jack.log - mean(jack.log))^2)))
```

```
## [1] 0.4154832
```

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.

```r
set.seed(5400)
p <- 0
size_b <- 30
B <- 1000
binomdata <- rbinom(20, size_b, p)
binom_samp_estim <- mean(binomdata)/size_b
bootestim_binom <- numeric(B)
for(i in 1:B){
  bootsample <- sample(binomdata, replace = TRUE)
  bootestim_binom[i] <- mean(bootsample)/size_b
```

```
}
#Confidence Interval
print(binom_samp_estim + c(-1, 1)*qnorm(0.975)*sd(bootestim_binom))
```

```
## [1] 0 0
```

We can see that the confidence interval is not correctly found by bootstrap method ### 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 $\widehat{\text{se}}_B$ for $B = 25, 100, 200, 500, 1000, 2000$. From these results estimate the ideal bootstrap estimate $\widehat{\text{se}}_\infty$.

```
trimmed_mean <- function(Xlist){
  return(mean(Xlist, trim = 0.25))
}
X_list <- c(1, 3, 4.5, 6, 6, 6.9, 13, 19.2)
trim_samp_estim <- trimmed_mean(X_list)
bval <- c(25, 100, 200, 500, 1000, 2000)
se_b <- numeric(6)
jj <- 1
for(B in bval){
  bootsample <- replicate(B, sample(X_list, replace = TRUE))
  bootestim_trim <- apply(bootsample, 2, trimmed_mean)
  se_b[jj] <- sd(bootestim_trim)
  jj = jj+1
}
print(se_b)
```

```
## [1] 2.299052 2.086476 1.929409 2.046863 2.234090 2.038842
```

```
print(mean(se_b))
```

```
## [1] 2.105789
```

The best value of SE is close to 1.9 and 2 (b) Repeat part (a) using twenty different random number seeds. Comment on the trend of the variablity of each $\widehat{\text{se}}_B$.

```
a <- c(1:10000)
seed.list <- sample(a, 20)
se_list2 <- c()
for (k in 1:20){
  set.seed(seed.list[k])
  B <- c(25,100,200,500,1000,2000)
```

```
  se_list <- rep(NA, length(B))
  for (j in 1:length(B)){
    boot_list <- rep(NA, B[j])
    for (i in 1:B[j]){
      boot.data <- sample(X_list, n, replace = TRUE)
      boot.theta <- mean(boot.data, trim = 0.25)
      boot_list[i] <- boot.theta
    }
    se_list[j] <- sd(boot_list)
  }
  se_list2 <- cbind(se_list2, se_list)
}
se_list2
```

```
##       se_list  se_list  se_list  se_list   se_list  se_list  se_list  se_list
## [1,] 1.799719 1.039392 1.639804 2.118474 0.9823946 2.443371 1.530755 1.509159
## [2,] 1.595473 1.820110 1.629473 1.923970 1.3898197 1.720817 1.575057 1.442814
## [3,] 1.335694 1.708881 1.498085 1.341342 1.7758761 1.662940 1.627736 1.960640
## [4,] 1.661881 1.653198 1.696004 1.559091 1.5614688 1.534473 1.675958 1.528871
## [5,] 1.575740 1.618123 1.675162 1.595072 1.6894363 1.656835 1.592783 1.615241
## [6,] 1.624796 1.636941 1.611248 1.589196 1.6078106 1.682536 1.554593 1.626518
##       se_list  se_list  se_list  se_list  se_list  se_list  se_list  se_list
## [1,] 1.550721 1.930068 1.605782 2.005265 1.994563 1.835043 1.972273 1.349080
## [2,] 1.843020 1.792155 1.609886 1.926268 1.617992 1.748409 1.559566 1.558927
## [3,] 1.426005 1.556706 1.762188 1.679958 1.588445 1.624158 1.610980 1.618721
## [4,] 1.639429 1.632521 1.597601 1.563458 1.558667 1.666210 1.571160 1.706834
## [5,] 1.655275 1.665298 1.599556 1.534908 1.611594 1.644455 1.554522 1.606365
## [6,] 1.591788 1.620423 1.567389 1.568466 1.583289 1.612991 1.603368 1.599472
##       se_list  se_list  se_list  se_list
## [1,] 1.467234 1.599590 1.431400 1.305349
## [2,] 1.614028 1.700640 1.490419 1.456489
## [3,] 1.606800 1.627543 1.631109 1.671459
## [4,] 1.728357 1.791688 1.467809 1.580515
## [5,] 1.596839 1.585699 1.522395 1.677437
## [6,] 1.551653 1.558171 1.639155 1.568996
```

```
print(apply(se_list2, 2, sd))
```

```
##    se_list    se_list    se_list    se_list    se_list    se_list    se_list
## 0.15151958 0.27443475 0.06941709 0.28148765 0.28546195 0.32928593 0.05245950
##    se_list    se_list    se_list    se_list    se_list    se_list    se_list
## 0.18322682 0.13751914 0.13714661 0.06946327 0.20345199 0.16571562 0.08642022
##    se_list    se_list    se_list    se_list    se_list    se_list
## 0.16182392 0.12012823 0.08542541 0.08721070 0.08643057 0.14185884
```

NOTE:This problem was done mostly by another classmate. We can see that the variability of the SE's ranges from 0.049 to 0.338.

4. Hypothesis testing using bootstrap Consider two independent random samples X and Y drawn from possibly different probability distributions:

$$X_1, \ldots, X_n \overset{iid}{\sim} F,$$
$$Y_1, \ldots, Y_m \overset{iid}{\sim} G,$$
$$X_i \text{ is indepdent of } Y_j, \ \forall i, j.$$

The goal is to perform a hypothesis test for

$$H_0 : F = G \text{ vs } H_1 : F \neq G$$

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:

$$t(\mathbf{Z}) = |\bar{\mathbf{X}} - \bar{\mathbf{Y}}|.$$

- For each $b = 1, \ldots B$:
  - Generate bootstrap samples, $\mathbf{Z}^{\star b}$, by drawing $(n + m)$ observations from **Z with replacement**.
  - Put $\mathbf{X}^{\star b} = (Z_1^{\star b}, \ldots, Z_n^{\star b})$ and $\mathbf{Y}^{\star b} = (Z_{n+1}^{\star b}, \ldots, Z_{n+m}^{\star b})$.
  - Compute bootstrap replications:
$$t(\mathbf{Z}^{\star b}) = |\bar{\mathbf{X}}^{\star b} - \bar{\mathbf{Y}}^{\star b}|.$$

- Estimate the achieved significance level (ASL):

$$\widehat{\text{ASL}}_B = \#\{t(\mathbf{Z}^{\star b}) \geq t(\mathbf{Z})\}/B.$$

- 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:

$$t(\mathbf{Z}) = \frac{|\bar{\mathbf{X}} - \bar{\mathbf{Y}}|}{\hat{\sigma}_{\text{pool}}\sqrt{1/n + 1/m}},$$

where

$$\hat{\sigma}_{\text{pool}} = \sqrt{\frac{1}{n+m-2}\left[\sum_{i=1}^{n}(X_i - \bar{\mathbf{X}})^2 + \sum_{j=1}^{m}(Y_j - \bar{\mathbf{Y}})^2\right]}.$$

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

```
set.seed(5400)
x <- rexp(20, rate=1/2)
y <- rexp(10, rate=2)
B <- 2000
z <- c(x, y)
nx <- length(x)
my <- length(y)
s_pool <- sqrt( (1/( nx + my - 2 )) * (sum( (x - mean(x))^2 ) + sum( (y - mean(y))^2 )))
tstat <- abs(mean(x) - mean(y)) / (s_pool*sqrt(1/nx + 1/my))
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.s_pool <- sqrt( (1/( nx + my - 2 )) *
                         (sum( (boot.samp[seq_along(x)] - m.boot.x)^2 ) +
                            sum( (boot.samp[-seq_along(x)] - m.boot.y)^2 )))
  boot.r[i] <- abs(m.boot.x - m.boot.y) / (boot.s_pool*sqrt(1/nx + 1/my))
}
mean(boot.r > tstat)
```

```
## [1] 0.0115
```