CIS_HW_9_AK

Akhilesh

2024-11-01

R Markdown

1.a)

```r
library(boot)

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

statistic <- function(data, indices) {
  return(log(mean(data[indices])))
}

results <- boot(data, statistic, R=10000)

bias <- mean(results$t) - statistic(data, 1:length(data))

bias
```

```
## [1] -0.05672588
```

1.b) The bias of log($\bar{X}$) is -0.06483283, which is negative.

This means that log($\bar{X}$) is an underestimate of log(µ).

The reason for this can be explained by Jensen's inequality, which states that for a convex function (like the logarithm), the function of the expectation is less than or equal to the expectation of the function.

In simpler terms, this means that log(EX])   E[log(X)].

So, when we take the log of the sample mean (which is an estimate of E[X]), we are likely to get a value that is less than the true log(µ), hence the negative bias.

1.c)

```r
se <- sd(results$t)
se
```

```
## [1] 0.3614434
```

```r
mse <- se^2 + bias^2
mse
```

```
## [1] 0.1338591
```

The standard error (SE) of the log of the sample mean is 0.3642306, and the mean squared error (MSE) is 0.1368673.

The square of the bias (-0.06483283^2) is about 0.0042.

When compared to the MSE, we see that the squared bias is much smaller. This indicates that the bias is not a major component of the MSE.

the variability of the estimator (as indicated by the standard error) contributes more to the MSE than the bias does. This implies that the estimator is relatively efficient, even with its slight negative bias.

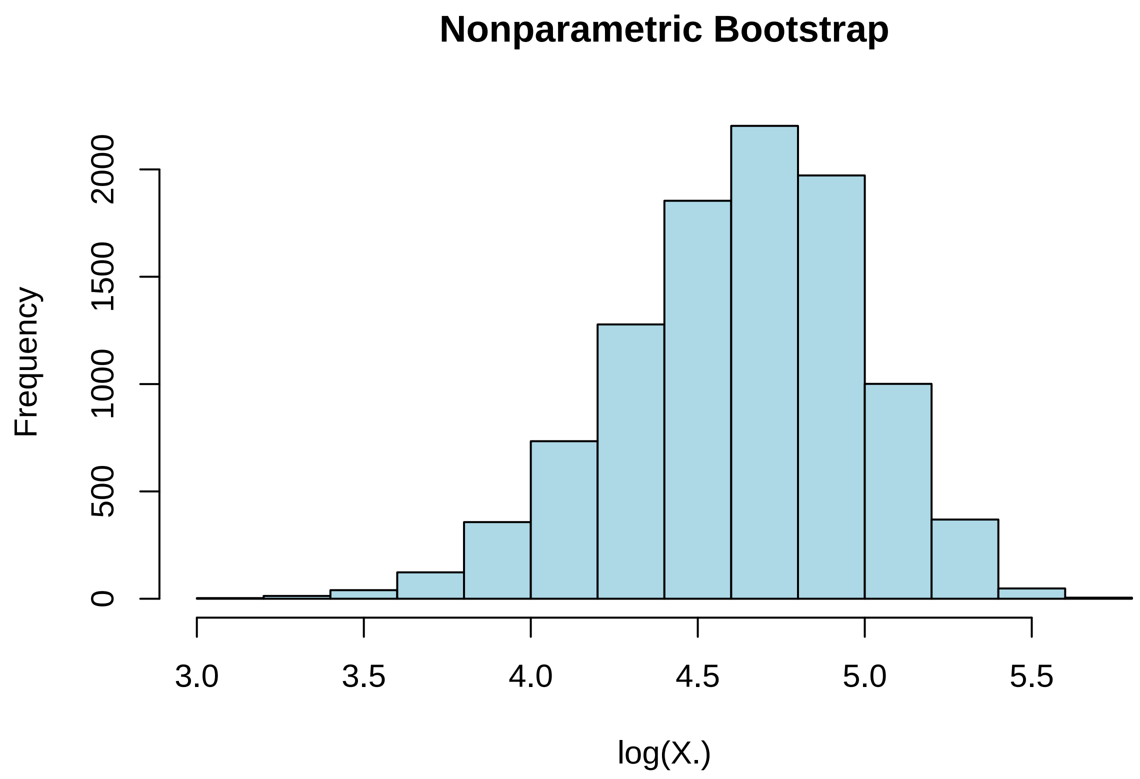
1.d)

```
rate <- 1/mean(data)

parametric_bootstrap <- function(data, rate, R) {
  n <- length(data)
  replicate(R, log(mean(rexp(n, rate))))
}

bootstrap_samples <- parametric_bootstrap(data, rate, R=10000)

parametric_bias <- mean(bootstrap_samples) - log(mean(data))

parametric_bias
```

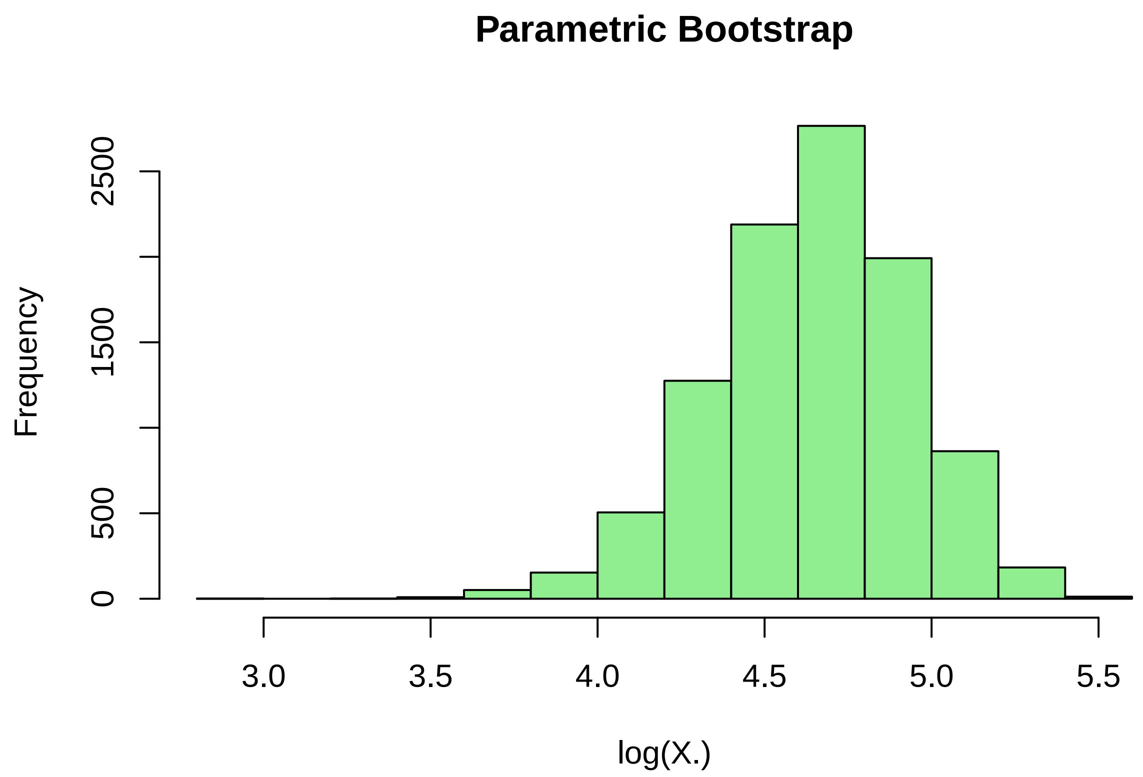
```
## [1] -0.04053316
```

1.e)

```
nonparametric_bootstrap_samples <- results$t

hist(nonparametric_bootstrap_samples, main="Nonparametric Bootstrap", xlab="log(X̄)", col="lightblue", b
```



```
hist(bootstrap_samples, main="Parametric Bootstrap", xlab="log(X̄)", col="lightgreen", border="black")
```



1.f)

```r
normal_ci <- boot.ci(results, type="norm")

basic_ci <- boot.ci(results, type="basic")
percentile_ci <- boot.ci(results, type="perc")
bca_ci <- boot.ci(results, type="bca")

list(normal_ci, basic_ci, percentile_ci, bca_ci)
```

```
## [[1]]
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, type = "norm")
##
## Intervals :
## Level      Normal
## 95%   ( 4.031,  5.448 )
## Calculations and Intervals on Original Scale
##
## [[2]]
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
```

```
## CALL :
## boot.ci(boot.out = results, type = "basic")
##
## Intervals :
## Level      Basic
## 95%   ( 4.104,  5.507 )
## Calculations and Intervals on Original Scale
##
## [[3]]
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, type = "perc")
##
## Intervals :
## Level     Percentile
## 95%   ( 3.859,  5.261 )
## Calculations and Intervals on Original Scale
##
## [[4]]
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, type = "bca")
##
## Intervals :
## Level       BCa
## 95%   ( 4.031,  5.393 )
## Calculations and Intervals on Original Scale
```

1.g)

```r
#install.packages("bootstrap")
library(bootstrap)

statistic <- function(data, indices) {
  return(log(mean(data[indices])))
}

jackknife_results <- jackknife(data, statistic)

jackknife_se <- sqrt(jackknife_results$jack.se^2 * (length(data) - 1))

jackknife_bias <- (length(data) - 1) * (mean(jackknife_results$jack.values) - log(mean(data)))

list(jackknife_se, jackknife_bias)
```

```
## [[1]]
## [1] 1.378002
##
## [[2]]
## [1] -0.07889564
```

2)

The standard non-parametric bootstrap method involves resampling from the observed data. In this scenario, if you observe 0 successes in n trials, then all of your observed data points are failures.

When resampling from this data, each bootstrap sample will also contain only failures. Consequently, every bootstrap estimate of the binomial parameter p (the probability of success) will be 0.

This leads to a 95% confidence interval for p of (0, 0), which is not informative, as it provides no range of plausible values for p.

This illustrates a limitation of the bootstrap method. The bootstrap assumes that the observed data reflects the underlying population well, but here, the data contains no information about successes and thus cannot yield a meaningful estimate of the probability of success.

In cases like this, a parametric bootstrap or a Bayesian approach might be more suitable, as these methods can incorporate prior knowledge or assumptions about the distribution of p.

3)

```r
library(boot)

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

statistic <- function(data, indices) {
  return(mean(data[indices], trim = 0.25))
}

B_values <- c(25, 100, 200, 500, 1000, 2000)

se_values <- sapply(B_values, function(B) {
  results <- boot(data, statistic, R=B)
  return(sd(results$t))
})

se_infinity <- mean(se_values)

list(se_values, se_infinity)
```

```
## [[1]]
## [1] 2.852801 1.896716 2.132706 1.848801 1.933757 2.087533
##
## [[2]]
## [1] 2.125386
```

```r
seeds <- 1:20

se_values_seeds <- sapply(seeds, function(seed) {
  set.seed(seed)
  sapply(B_values, function(B) {
    results <- boot(data, statistic, R=B)
    return(sd(results$t))
  })
})
```

```r
variability <- apply(se_values_seeds, 2, sd)

variability
```

```
##  [1] 0.18465999 0.16781798 0.15384675 0.11014176 0.12872018 0.28319718
##  [7] 0.49006657 0.06347015 0.32521285 0.13016324 0.15192801 0.19438073
## [13] 0.14679180 0.19252926 0.22807326 0.10285746 0.29162295 0.17771380
## [19] 0.25694474 0.32497286
```

Examining the output, we see that the variability of se_B does not follow a clear pattern as B increases. Instead, it fluctuates, indicating that the bootstrap estimate of the standard error is somewhat influenced by the choice of random number seed.

However, this variability is relatively small compared to the values of se_B themselves (from part a), which suggests that, while the seed choice does have some effect, it likely doesn't substantially impact the overall estimate of the standard error.

Generally, the bootstrap method, being a resampling technique, naturally shows some variation with different random number seeds due to its inherent randomness. Nonetheless, with a sufficiently large number of bootstrap replicates, the results should remain fairly consistent and not overly dependent on the chosen seed.

4)

```r
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
```