HW1

2023-03-30

```r
# Load packages
pacman::p_load(tidyverse, boot, car)
```

Question 1 Monte Carlo Simulation

```r
# Set true values of parameters
y_true = c(1, 1)
sigma = matrix(diag(c(0.25^2,1)), ncol = 2)
theta = y_true[1]/y_true[2]

# Simulating parameters
n = 50
B = 10000

# Set seed
set.seed(123)

# Initialize simulation vectors
sim.theta = rep(0, B)
sim.mu1 = rep(0,B)
sim.mu2 = rep(0, B)
sim.sigma = matrix(0, B, 3)

# Monte Carlo loop
for (sim in 1:B) {
  # Simulate data
  sim.y = tibble(y1 = rnorm(n, mean = y_true[1], sd = sigma[1,1]),
                 y2 = rnorm(n, mean = y_true[2], sd = sigma[2,2]))
  mu_hat = c(mean(sim.y$y1), mean(sim.y$y2))
  sigma_hat = with(sim.y, c(sd(y1), cov(y1, y2), sd(y2)))/n

  # Parameter of interest
  theta_hat = mu_hat[1]/mu_hat[2]

  # Store the simulated values
  sim.theta[sim] = theta_hat
  sim.mu1[sim] = mu_hat[1]
  sim.mu2[sim] = mu_hat[2]
  sim.sigma[sim,] = sigma_hat
}

# Calculate the mean and standard error of theta_hat
mean_theta_hat = mean(sim.theta); mean_theta_hat
```

```
## [1] 1.020602
```

```
se_theta_hat = sd(sim.theta); se_theta_hat
```

```
## [1] 0.1531086
```

```
# Confidence interval of theta_hat
mean_theta_hat + c(-se_theta_hat, se_theta_hat)*1.96
```

```
## [1] 0.7205087 1.3206943
```

```
# Var cov matrix
sigma_hat = c(mean(sim.sigma[,1]), rep(mean(sim.sigma[,2]), 2), mean(sim.sigma[,3])) |> matrix(nrow = 2
```

Question 2

Theta is defined as $\theta = \frac{\mu_1}{\mu_2} = f(\mu)$. To use the delta method, we take the derivative of θ in therms of μ_1 and μ_2.

$$g(\mu) = \frac{\partial f(\mu)}{\partial \mu} = \begin{pmatrix} \frac{\partial f(\mu)}{\partial \mu_1} \\ \frac{\partial f(\mu)}{\partial \mu_2} \end{pmatrix} = \begin{pmatrix} \frac{1}{\mu_2} \\ \frac{-\mu_1}{\mu_2^2} \end{pmatrix}$$

Using this, we can calculate the variance as

$$\widehat{\text{var}}(\hat{\theta}) = g(\hat{\theta})' \hat{V}_{\hat{\theta}} g(\hat{\theta})$$

So the standard error is

$$\widehat{\text{se}}(\hat{\mu}) = \sqrt{\widehat{\text{var}}(\hat{\theta})}$$

We already calculated the $\hat{V}_{\hat{\theta}}$ in part 1, so we just need to enumerate $g(\theta)$

```
g_mu_hat = c(1/y_true[1], -y_true[1]/y_true[2]) |> matrix()
varhat_theta_hat = t(g_mu_hat)%*%sigma_hat%*%g_mu_hat
sehat_theta_hat = sqrt(varhat_theta_hat)
```

Now check with the delta_method function from the car package.

```
# Set names of g_mu_hat
names(g_mu_hat) = c("mu1", "mu2")
rownames(g_mu_hat) = names(g_mu_hat)

# String for the expression of interest
f_string = "mu1 / mu2"

# Delta method
dm = car::deltaMethod(g_mu_hat, f_string, sigma_hat)
dm$SE # Exact match for the manual computation
```

```
## [1] 0.145384
```

Question 3 Jackknife Method

```
# Simulated data
y_sim = tibble(y1 = rnorm(n, mean = y_true[1], sd = sigma[1,1]),
               y2 = rnorm(n, mean = y_true[2], sd = sigma[2,2]))

# Initialize values
n = nrow(y_sim)
jack_theta = rep(0, n)

# Set seed
set.seed(123)

# Jackknife loop
for (i in 1:n){
  # Selects all the data except the one observation we're leaving out
  temp_data = y_sim[-i,]

  # Calculate the parameters
  mu_hat = c(mean(temp_data$y1), mean(temp_data$y2))

  # Parameter of interest
  theta_hat = mu_hat[1]/mu_hat[2]

  # Store the parameter
  jack_theta[i] = theta_hat
}

# Jackknife standard error
jack_mean = mean(jack_theta)
jack_se = ((n-1)/n)*sum((jack_theta - theta)^2);jack_se
```

```
## [1] 0.1110255
```

```
# Confidence interval
jack_mean + c(-jack_se, jack_se)*1.96
```

```
## [1] 0.8238768 1.2590968
```

Question 4 NonParametric Bootstrap

```
# Bootstrap parameters
n = 50
B = 10000

# Initialize necessary vectors
boot_theta = rep(0, B)
list_boot_theta = vector('list', 5) # empty list to put results from different seeds into
boot_test = vector('list', 5)
# Bootstrap loop
# Outside loop is to loop over the different seeds
for (seed in 123:127){
  set.seed(seed)
  y_sim = tibble(y1 = rnorm(n, mean = y_true[1], sd = sigma[1,1]),
                 y2 = rnorm(n, mean = y_true[2], sd = sigma[2,2]))

  # Inside loop is bootstrapping
  for (i in 1:B){
    # Get bootstrap sample
    idx = sample(n, replace = TRUE) # idk stands for index
    boot_samp = y_sim[idx,]

    # Get parameters
    mu_hat = c(mean(boot_samp$y1), mean(boot_samp$y2))

    # Parameter of interest
    boot_theta[i] = mu_hat[1]/mu_hat[2]
    boot_test[[i]] = boot_samp
  }
  list_boot_theta[[seed-122]] = boot_theta # Subtract 122 so the list indexes from 1
}

# Parameter of interest for each seed
boot_theta_hat = sapply(1:5, function(i) {mean(list_boot_theta[[i]])})
boot_se_theta_hat = sapply(1:5, function(i) {sd(list_boot_theta[[i]])});boot_se_theta_hat
```

```
## [1] 0.10379276 0.12699370 0.15338724 0.09707892 0.10356049
```

```
# Normal CI
sapply(1:5, function(i){boot_theta_hat[[i]] + c(-boot_se_theta_hat[[i]], boot_se_theta_hat[[i]])*1.96})
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 0.682592 0.7006524 0.6695321 0.6554387 0.6584561
## [2,] 1.089460 1.1984678 1.2708101 1.0359881 1.0644132
```

To calculate the other CIs, we will use the boot package

```
# This is a function to get the parameter given a bootstrapped sample
fun_theta = function(x, i){
  temp_data = x[i,]
  boot_theta = mean(temp_data$y1)/mean(temp_data$y2)
  return(boot_theta)
}

# Initialize empty list for the result from the five seeds
list_boot_theta_ci = vector('list', 5)

# Loop over the seeds again
for (seed in 123:127) {
  set.seed(seed)
  y_sim = tibble(y1 = rnorm(n, mean = y_true[1], sd = sigma[1,1]),
                 y2 = rnorm(n, mean = y_true[2], sd = sigma[2,2]))
  theta.boot = boot(y_sim, statistic = fun_theta, R = 10000)

  list_boot_theta_ci[[seed-122]] = theta.boot
}

# All the confidence intervals
# Im not doing this in a loop because it takes away the nice formatting of the CIs
boot.ci(list_boot_theta_ci[[1]], conf=0.95, type=c("norm", "perc", "bca"))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = list_boot_theta_ci[[1]], conf = 0.95, type = c("norm",
##     "perc", "bca"))
##
## Intervals :
## Level      Normal             Percentile            BCa
## 95%   ( 0.6586,  1.0661 )   ( 0.7173,  1.1215 )   ( 0.7171,  1.1205 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(list_boot_theta_ci[[2]], conf=0.95, type=c("norm", "perc", "bca"))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = list_boot_theta_ci[[2]], conf = 0.95, type = c("norm",
##     "perc", "bca"))
##
## Intervals :
## Level      Normal             Percentile            BCa
## 95%   ( 0.6720,  1.1742 )   ( 0.7468,  1.2499 )   ( 0.7536,  1.2634 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(list_boot_theta_ci[[3]], conf=0.95, type=c("norm", "perc", "bca"))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = list_boot_theta_ci[[3]], conf = 0.95, type = c("norm",
##     "perc", "bca"))
##
## Intervals :
## Level      Normal             Percentile            BCa
## 95%   ( 0.6324,  1.2260 )   ( 0.7438,  1.3291 )   ( 0.7542,  1.3851 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(list_boot_theta_ci[[4]], conf=0.95, type=c("norm", "perc", "bca"))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = list_boot_theta_ci[[4]], conf = 0.95, type = c("norm",
##     "perc", "bca"))
##
## Intervals :
## Level      Normal              Percentile            BCa
## 95%   ( 0.6365,  1.0130 )   ( 0.6906,  1.0618 )   ( 0.6949,  1.0731 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(list_boot_theta_ci[[5]], conf=0.95, type=c("norm", "perc", "bca"))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = list_boot_theta_ci[[5]], conf = 0.95, type = c("norm",
##     "perc", "bca"))
##
## Intervals :
## Level      Normal              Percentile            BCa
## 95%   ( 0.6304,  1.0419 )   ( 0.6883,  1.0938 )   ( 0.6852,  1.0841 )
## Calculations and Intervals on Original Scale
```

```
# The normal CI is not quite the same as my manual one, but close enough
```

Question 5 Parametric Bootstrap

```
# Set parameters
B = 10000
n = 50

# Initialize empty vectors
boot_theta = rep(0, B)
list_boot_theta = vector('list', 5) # empty list to put results from different seeds into


for (seed in 123:127) {
  set.seed(seed)
  y_sim = tibble(y1 = rnorm(n, mean = y_true[1], sd = sigma[1,1]),
                 y2 = rnorm(n, mean = y_true[2], sd = sigma[2,2]))

  # Get bootstrap se's
  se_y1 = sd(y_sim$y1)
  se_y2 = sd(y_sim$y2)

  # Get bootstrap means
  mu1 = mean(y_sim$y1)
  mu2 = mean(y_sim$y2)
```

```r
  # Run bootstrap
  for (i in 1:B) {
    y1_b = rnorm(n, mu1, se_y1)
    y2_b = rnorm(n, mu2, se_y2)

    # Parameter of interest
    theta_hat = mean(y1_b) / mean(y2_b)

    # Store parameters
    boot_theta[i] = theta_hat

  }
  list_boot_theta[[seed-122]] = boot_theta # Subtract 122 so the list indexes from 1
}

# Parameter of interest for each seed
boot_theta_hat = sapply(1:5, function(i) {mean(list_boot_theta[[i]])})
boot_se_theta_hat = sapply(1:5, function(i) {sd(list_boot_theta[[i]])});boot_se_theta_hat
```

```
## [1] 0.10251781 0.13002571 0.15416749 0.09712885 0.10730163
```

```r
# Normal CI
sapply(1:5, function(i){boot_theta_hat[[i]] + c(-boot_se_theta_hat[[i]], boot_se_theta_hat[[i]])*1.96})
```

```
##           [,1]     [,2]      [,3]      [,4]      [,5]
## [1,] 0.6841243 0.697833 0.6708458 0.6542447 0.6522617
## [2,] 1.0859941 1.207534 1.2751823 1.0349898 1.0728841
```

Question 6 Table Comparing Results