Homework-SMDS

LEC Exercises

Exercise 1

1. Compute the bootstra-based confidence interval for the **score** dataset using the studentized method.

Solution

```
set.seed(1)

# Checking computational time required for the naive version;
start_time <- Sys.time()

score <- read.table("student_score.txt", header = TRUE)

# B: number of total bootstrap samples;
# s_vect: vector of parameters calculated on bootstrap samples, theta_{b};
# jack_se: vector of standard errors calculated by implementing the jackknife method;
# z_vect: vector of standardized values require for the "studentized method";
B <- 10^2; s_vect <- rep(NA, B); jack_se <- rep(NA, B); z_vect <- rep(NA, B);

# Calculates eigenratio statistic;
psi_fun <- function(x)
{
  eig = eigen(cor(x))$values
  return(max(eig) / sum(eig))
}

# Calculates standard error of an estimator by implementing the jackknife method;
smpl_se <- function(x)
{
  for(i in 1:nrow(x))
  {
    if(i == 1) {jack_eratio = rep(NA, nrow(x));}

    jack_eratio[i] = psi_fun(x[-c(i),])

    if(i == nrow(x)) {return(sqrt(((nrow(x)-1)/nrow(x)) * sum((jack_eratio - mean(jack_eratio))^2)))}
  }
}

# Sample estimator + index;
psi_obs <- psi_fun(score); n <- nrow(score);
```

```r
# Main function: derives the necessary data for computing the CI;
for(i in 1:B)
{
  ind = sample(1:n, n, replace = TRUE)
  s_vect[i] = psi_fun(score[ind,]);
  jack_se[i] = smpl_se(score[ind,]);
  z_vect[i] = (s_vect[i] - psi_obs)/jack_se[i];
}

SE_boot <- sd(s_vect);
CI      <- psi_obs - quantile(z_vect, probs = c(0.975, 0.025)) * SE_boot

cat(" BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS\n",
    "Based on 1000 bootstrap replicates\n\n",
    "Intervals :\n", "Level      Studentized\n", "95%    (",
    round(CI[1], 3), ", ", round(CI[2], 3), ")\n\n")
```

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

 Intervals :
 Level      Studentized
 95%    ( 0.566 ,  0.805 )
```

```r
end_time <- Sys.time()
end_time - start_time
```

```
Time difference of 0.9121261 secs
```

Exercise 2

Compute bootstrap-based confidence intervals for the **score** dataset using the **boot** package.

Solution

```r
set.seed(1)

require(boot)
require(dplyr)
require(purrr)

start_time <- Sys.time()

# Calculates eigenratio statistic (similar to the
# original but taking also an index for sampling);
psi_fun <- function(data, indices)
{
  d = data[indices,];
  eig = eigen(cor(d))$values; return(max(eig) / sum(eig));
```

```r
}

# Calculates variance of an estimator by implementing a second-level bootstrap;
smpl_var <- function(data, indices, its)
{
  d = data[indices,]; n = nrow(d);
  eig = eigen(cor(d))$values; eratio = max(eig) / sum(eig);

  v = boot(R = 100, data = d, statistic = psi_fun, parallel = "multicore") %>%
            pluck("t") %>%
            var(na.rm = TRUE);
  c(eratio, v)
}

boot_t_out <- boot(R = 100, data = score, statistic = smpl_var, parallel = "multicore")
CI_vec     <- boot.ci(boot_t_out, type = "stud");
CI_vec
```

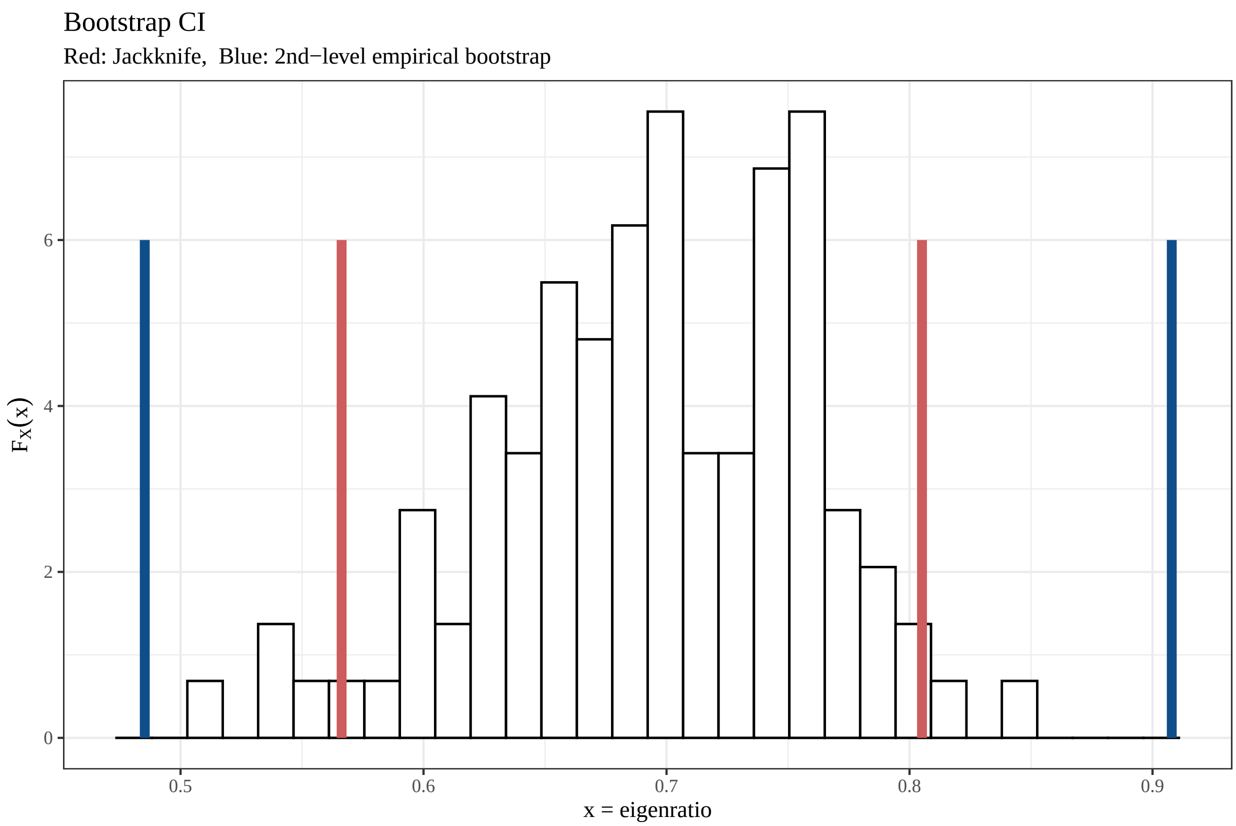
```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 100 bootstrap replicates

CALL :
boot.ci(boot.out = boot_t_out, type = "stud")

Intervals :
Level    Studentized
95%   ( 0.4853,  0.9079 )
Calculations and Intervals on Original Scale
Some studentized intervals may be unstable
```

```r
end_time <- Sys.time()
end_time - start_time
```

```
Time difference of 3.642625 secs
```

```r
set.seed(1)

require(ggplot2)
require(gridExtra)

boot_sample <- rep(NA, B);
for(i in 1:B)
{
  ind = sample(1:n, n, replace = TRUE)
  boot_sample[i] = psi_fun(score[ind,]);
}

ggplot() +

  # Bootstrap Sampling Distribution
  geom_histogram(aes(x = boot_sample, y = ..density..), color = "black", fill  = "white") +
```

```r
  # CI for Naive with Jackknife
  geom_line(aes(x = rep(CI[1], 2), y = c(0, 6)), color = "indianred", size = 2) +
  geom_line(aes(x = rep(CI[2], 2), y = c(0, 6)), color = "indianred", size = 2) +

  # CI for boot() with 2nd-level bootstrap
  geom_line(aes(x = rep(CI_vec$student[4], 2), y = c(0, 6)), color = "dodgerblue4", size = 2) +
  geom_line(aes(x = rep(CI_vec$student[5], 2), y = c(0, 6)), color = "dodgerblue4", size = 2) +

  # Custom Label
  labs(title = "Bootstrap CI",
       subtitle = "Red: Jackknife,  Blue: 2nd-level empirical bootstrap",
       x = "x = eigenratio",
       y = expression(F[X](x))) +
  theme_bw(base_size = 10, base_family = "Times")
```




```r
set.seed(1)

require(boot)
require(dplyr)
require(purrr)

start_time <- Sys.time()

# Calculates eigenratio statistic (similar to the
# original but taking also an index for sampling);
```

```r
psi_fun <- function(data, indices)
{
  d = data[indices,];
  eig = eigen(cor(d))$values; return(max(eig) / sum(eig));
}

# Calculates variance of an estimator by implementing a second-level bootstrap;
smpl_var <- function(data, indices, its)
{
  d = data[indices,]; n = nrow(d);
  eig = eigen(cor(d))$values; eratio = max(eig) / sum(eig);

  v = boot(R = 100, data = d, statistic = psi_fun, parallel = "multicore") %>%
            pluck("t") %>%
            var(na.rm = TRUE);
  c(eratio, v)
}

boot_t_out <- boot(R = 100, data = score, statistic = smpl_var, parallel = "multicore")
CI_vec     <- boot.ci(boot_t_out, type = "stud");
CI_vec
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 100 bootstrap replicates

CALL :
boot.ci(boot.out = boot_t_out, type = "stud")

Intervals :
Level    Studentized
95%   ( 0.4853,  0.9079 )
Calculations and Intervals on Original Scale
Some studentized intervals may be unstable
```

```r
end_time <- Sys.time()
end_time - start_time
```

```
Time difference of 3.722736 secs
```

Lab Exercises

Exercise 5

In **sim** in the code above, you find the MCMC output which allows to approximate histogram for these random draws $\theta^{(1)}, \dots, \theta^{(S)}$, compute the empirical quantiles, and overlap the true posterior distribution.

```r
set.seed(1)

require(rstanarm)
require(bayesplot)
```

```r
require(ggplot2)
require(gridExtra)
require(rstan)

# True mean;
theta_sample <- 2;
# Likelihood variance;
sigma2 <- 2;
# Sample size;
n <- 10;
# Prior mean;
mu <- 7;
# Prior variance;
tau2 <- 2

# Generate some data;
y <- rnorm(n, theta_sample, sqrt(sigma2))

# Posterior mean;
mu_star <- ((1/tau2) * mu + (n/sigma2) * mean(y))/((1/tau2) + (n/sigma2))
# Posterior standard deviation;
sd_star <- sqrt(1/((1/tau2) + (n/sigma2)))

# Launch Stan Model;
data <- list(N = n, y = y, sigma = sqrt(sigma2), mu = mu, tau = sqrt(tau2))
fit  <- stan(file = "normal.stan", data = data, chains = 4, iter = 2000)
sim  <- data.frame(extract(fit))
```

```r
plot_mcmc <- ggplot() +

  # theta - MCMC;
  geom_histogram(data = sim, aes(x = theta, y = ..density..),
                 colour = "black",
                 fill   = "white",
                 alpha  = 0.5) +

  # True Posterior;
  geom_line(aes(x = seq(1, 4.5, 0.01), y = dnorm(seq(1, 4.5, 0.01), mu_star, sd_star)),
            color = "indianred",
            size  = 1) +

  # Custom label;
  labs(title = "MCMC approximation of the posterior distribution",
       x     = "x",
       y     = expression(F[X](x))) +
  theme_bw(base_size = 10, base_family = "Times")

emp_quantiles <- ggplot() +

  # True Posterior quantiles;
  geom_line(aes(x = qnorm(seq(0.01, 0.99, 0.01), mu_star, sd_star),
                y = qnorm(seq(0.01, 0.99, 0.01), mu_star, sd_star)),
            color = "red2") +
```

```r
  # theta - MCMC quantiles;
  geom_point(aes(x = qnorm(seq(0.01, 0.99, 0.01), mu_star, sd_star),
                 y = as.vector(quantile(sim$theta, seq(0.01, 0.99, 0.01)))),
             size  = 0.5,
             color = "black") +

  # Custom label;
  labs(title = "Q-Q plot for MCMC distribution",
       x     = "True quantiles",
       y     = "Sample quantiles") +
  theme_bw(base_size = 10, base_family = "Times")


grid.arrange(plot_mcmc, emp_quantiles, nrow = 1, ncol = 2)
```