Homework 7 - STAT5400

Dev Narayan Baiju

Due: Oct 18, 2024. 9:30 AM

Problems

Submit your solutions as an .Rmd file and accompanying .pdf file. Include all the **relevant** R code and output. Always interpret your result whenever it is necessary.

Reading assignments.

Below is a tutorial on confidence interval. Read it if you have unfamiliar with the topic. https://online.stat.psu.edu/statprogram/reviews/statistical-concepts/confidence-intervals

Problems

1. Filling in the missing pieces on slides

- Fill in the missing piece on slides 46, 48, and 49 of S3P2.pdf. You only need to **submit the three missing lines** not the whole code. If you use the same seed as on the side, namely 5400, you should get the same estimates of the coverage probabilities.

Missing code for slide 46:

```
se <- 1.96*sqrt(sig.sq/n)
```

Missing code for slide 48:

```
moe <- 1.96*sdlist/sqrt(n) or qt(0.025, 19, lower.tail = FALSE)*sdlist/sqrt(n)
```

Missing code for slide 49:

```
moe <- 1.96*apply(Xlist, 1, sd)/sqrt(n)
```

2. A generic CovProb function

- Write a new generic CovProb function on slide 49.

The function has five arguments: `n`, `mu`, `alp`, `dis`, and the three-dot argument, where `dist` is the distribution names, such as `exp`, `unif`, `gamma`, and the three-dot argument specifies the input for the corresponding functions: `rexp`, `runif`, `rgamma`, etc.

A tutorial on the three-dot argument can be found in https://www.r-bloggers.com/2020/11/some-notes-when-using-dot-dot-dot-in-r/

```
new_CovProb <- function(n, mu=2, alp=0.05, dist,...) {

  set.seed(4356)
  if(dist == "exp")
    Xlist <- matrix(rexp(10000 * n,...), 10000, n)

  else if(dist == "unif")
    Xlist <- matrix(runif(10000 * n,...), 10000, n)

  else if(dist == "gamma")
    Xlist <- matrix(rgamma(10000*n,...), 10000, n)

  Xbarlist <- rowMeans(Xlist)
  moe <- 1.96*apply(Xlist, 1, sd)/sqrt(n)
  CI <- cbind(Xbarlist - moe, Xbarlist + moe)
  is_cover <- apply(CI, 1, function(x) mu > x[1] & mu < x[2])
  mean(is_cover)
}
mu = 2
new_CovProb(n = 20, mu = mu, alp = 0.05, dist = "exp", rate = 1/mu)
```

```
## [1] 0.9056
```

```
new_CovProb(n = 20, mu = mu, alp = 0.05, dist = "unif", min = 0, max = 2*mu-0)
```

```
## [1] 0.9329
```

```
new_CovProb(n = 20, mu = mu, alp = 0.05, dist = "gamma", shape = 2, rate = 2/mu)
```

```
## [1] 0.9165
```

3. Estimate bias, variance, and MSE of the trimmed mean

Suppose $\hat{\theta}$ is an estimator of a population parameter θ. The bias is defined as $\mathrm{E}(\hat{\theta} - \theta)$, and the mean squared error (MSE) is defined as $\mathrm{E}(\hat{\theta} - \theta)^2$.

Suppose $X_1, \ldots, X_{15}$ is a random sample from the $t(4)$ distribution. We consider the trimmed mean to estimate the population mean, where the trimmed mean is the average of all the sample observations except for the largest and smallest ones.

- Estimate the bias, variance, and MSE of the trimmed mean using simulations.

```
nsimulate <- 2000
bias_MSE <- matrix(NA, 2000, 2)
set.seed(4739)
for(i in 1:nsimulate){
  X <- rt(15, 4)
  trimmean <- mean(X[X<max(X) & X>min(X)])
  bias_MSE[i,1] <- trimmean-0
  bias_MSE[i,2] <- (trimmean-0)*(trimmean-0)
}
```

```r
bias <- mean(bias_MSE[,1])
MSE <- mean(bias_MSE[,2])
variance <- MSE - bias*bias
print(c(bias, variance, MSE))
```

```
## [1] 0.006656307 0.101750815 0.101795121
```

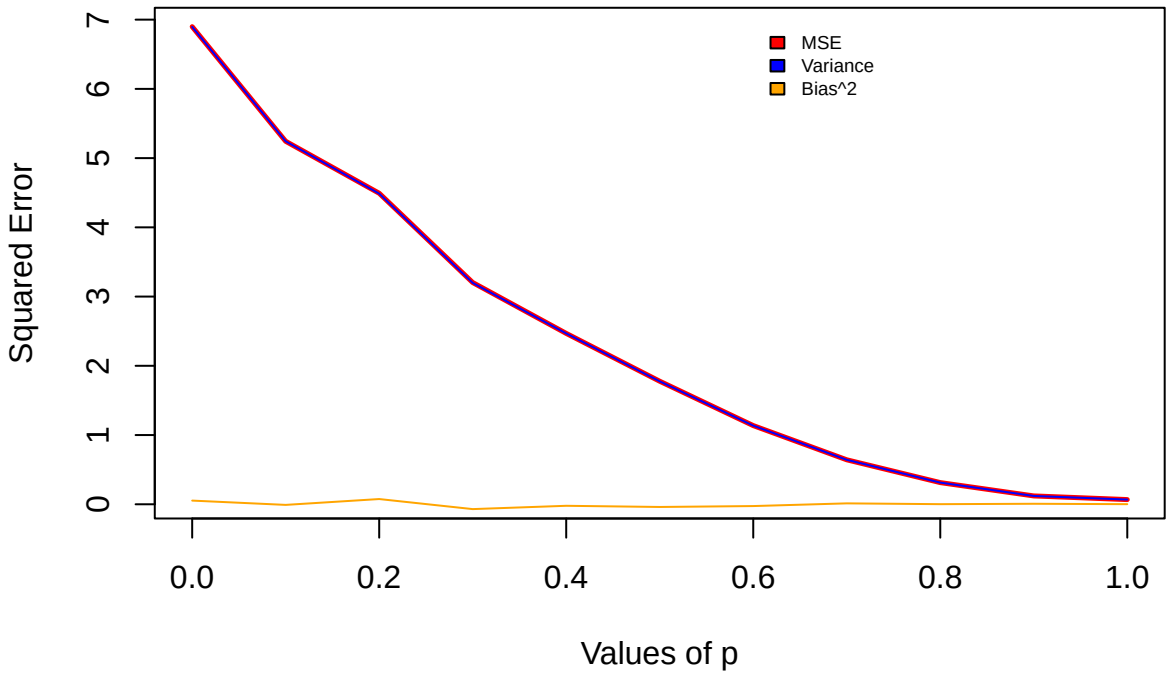
- Now suppose the data-generating model is a mixture normal distribution: $p\mathrm{N}(0,1)+(1-p)\mathrm{N}(0,10^2)$. Plot the estimate of the bias, variance, and MSE against p, where $p=(0,0.1,0.2,\ldots,1)$.

```r
nsimulate <- 2000
bias_mixed <- rep(NA, length(seq(0, 1, 0.1)))
MSE_mixed <- rep(NA, length(seq(0, 1, 0.1)))
j <- 1
set.seed(4739)
for(p in seq(0, 1, 0.1)){
  bias_MSE <- matrix(NA, 2000, 2)
  for(i in 1:nsimulate){

    X_mixed <- p*rnorm(15, 0, 1) + (1-p)*rnorm(15, 0, 10)
    trimmean <- mean(X_mixed[X_mixed<max(X_mixed) & X_mixed>min(X_mixed)])

    bias_MSE[i,1] <- trimmean-0
    bias_MSE[i,2] <- (trimmean-0)*(trimmean-0)
  }
  bias_mixed[j] <- mean(bias_MSE[,1])
  MSE_mixed[j] <- mean(bias_MSE[,2])
  j= j + 1
}
variance_mixed <- MSE_mixed - bias_mixed*bias_mixed

plot(seq(0, 1, 0.1), MSE_mixed, type = 'l', col = "red", lwd = 2.5,
     xlab = "Values of p", ylab = "Squared Error")
lines(seq(0, 1, 0.1), variance_mixed, col = "blue")
lines(seq(0, 1, 0.1), bias_mixed, col = "orange")
legend(0.6, 7, legend = c("MSE", "Variance", "Bias^2"), fill = c("red", "blue", "orange"),
       cex = 0.6, bty = "n")
```



4. Confidence interval for Poisson distributions

Suppose $X_1, \ldots, X_{20}$ is a random sample from Poisson distribution with mean $\lambda = 5$.

```
set.seed(5400)
dat = rpois(20, 5)
```

- Based on this sample, construct an approximated 95% confidence interval for the mean λ using central limit theorem.

```
n = 20
lamda = 5
alp = 0.05
dat = rpois(n, lamda)
meandat = mean(dat)
moe = qnorm(alp/2, lower.tail = FALSE)*sd(dat)/sqrt(n)
print(lamda)
```

```
## [1] 5
```

```
print(meandat-moe)
```

```
## [1] 3.781879
```

```
print(meandat+moe)
```

```
## [1] 5.518121
```

- Estimate the coverage probability of this approximated confidence interval using simulations with 10^5 replications.

```r
CovProb_Poisson <- function(n, lamda=5, alp=0.05) {
  dat <- matrix(rpois(100000 * n, lambda = lamda), 100000, n)
  datbarlist <- rowMeans(dat)
  #print(mean(apply(dat, 1, var)))
  moe <- qnorm(alp/2, lower.tail = FALSE)*apply(dat, 1, sd)/sqrt(n)
  CI <- cbind(datbarlist - moe, datbarlist + moe)
  is_cover <- apply(CI, 1, function(x) lamda > x[1] & lamda < x[2])
  mean(is_cover)
}
CovProb_Poisson(20, 5)
```

```
## [1] 0.93369
```

- In theory, an exact $100(1-\alpha)\%$ confidence interval for λ is given by

$$\left[\frac{1}{2n}\chi^2_{2s,1-\alpha/2}, \frac{1}{2n}\chi^2_{2(s+1),\alpha/2}\right],$$

  where $s = \sum_{i=1}^{n} x_i$, $\chi^2_{v,u}$ can be obatined by `qchisq(1-u, v)`. When $v = 0$, we always have $\chi^2_{0,u} = 0$. Based on the generated sample, construct an exact 95% confidence interval for the mean λ.

```r
#Confidence Interval for a sample of 20 values from Poisson Distribution
dat = rpois(20, 5)
datbar = mean(dat)#Find the mean of the data
sumdat = sum(dat)#Find the sum of the data
print(qchisq(alp/2, 2*sumdat)/(2*n))
print(qchisq(1-alp/2, 2*(sumdat+1))/(2*n))
```

- Estimate the coverage probability of the exact confidence interval using simulations with 10^5 replications.

```r
CovProb_Poisson_exact <- function(n, lamda=5, alp=0.05) {
  dat <- matrix(rpois(100000 * n, lambda = lamda), 100000, n)
  datbarlist <- rowMeans(dat)#Find the row-wise mean of the data
  sum_list <- rowSums(dat)#Find the row-wise sum of the data
  #Confidence Interval using Chi-Square formula
  CI <- cbind(qchisq(alp/2, 2*sum_list)/(2*n), qchisq(1-alp/2, 2*(sum_list+1))/(2*n))
  is_cover <- apply(CI, 1, function(x) lamda > x[1] & lamda < x[2])
  mean(is_cover)
}
CovProb_Poisson_exact(20, 5)
```

```
## [1] 0.95545
```

5. Confidence interval for proportions

Design simulation examples to compare the six confidence intervals for proportions introduced in S3P2.pdf, say Slide 68.

```r
library(binom)
set.seed(5400)
theta <- 0.6
```

```r
n <- 40
B <- 10000
alp <- 0.05
# generate data
Xbarlist <- rbinom(B, size=n, prob=theta)/n
## Wald interval
moe_Wald <- qnorm(1-alp/2) * sqrt(Xbarlist * (1 - Xbarlist)/n)
CI_Wald <- cbind(Xbarlist - moe_Wald,
                 Xbarlist + moe_Wald)
## simple conservative interval
moe_simple <- qnorm(1-alp/2) * sqrt(0.5 * (1 - 0.5)/n)
CI_simple <- cbind(Xbarlist - moe_simple,
                   Xbarlist + moe_simple)
## score, CP, AC, and the Jeffery's Bayesian CIs
CI_score <- CI_CP <- CI_AC <- CI_Bayes <- matrix(NA, B, 2)
for (i in seq(B)) {
  CI_score[i, ] <- prop.test(x=(n*Xbarlist[i]), n=n,
                             correct=FALSE, conf.level=(1-alp))$conf.int
  CI_CP[i, ] <- binom.test(x=(n*Xbarlist[i]), n=n,
                           conf.level=(1-alp))$conf.int
  ret <- binom.confint(x=(n*Xbarlist[i]), n=n,
                       conf.level=(1-alp), methods="ac")
  CI_AC[i,] <- c(ret$lower, ret$upper)
  ret2 <- binom.bayes(x=(n*Xbarlist[i]), n=n,
                      type="central", conf.level=(1-alp))
  CI_Bayes[i,] <- c(ret2$lower, ret2$upper)
}

## get mean and se of width and cov prob
WidthCov <- function(CI) {

    # get coverage probability
    is_cov <- function(x) theta > x[1] & theta < x[2]
    # get standard error
    se <- function(x) sd(x)/sqrt(nrow(CI))
    allwidth <- CI[,2] - CI[,1]     # width
    allcov <- apply(CI, 1, is_cov)  # cov prob
    # returns:
    c(mean(allwidth), se(allwidth),
      mean(allcov), se(allcov))
}
## display results
col.nm <- paste0("CI_", c('simple', 'Wald', 'score', 'CP', 'AC', 'Bayes'))
res <- sapply(lapply(col.nm, get), WidthCov)
colnames(res) <- col.nm
rownames(res) <- c('exp.width', 'se', 'cov.prob', 'se')
print(round(res, 3))
```

```
##           CI_simple CI_Wald CI_score CI_CP CI_AC CI_Bayes
## exp.width     0.310   0.300    0.287 0.314 0.288    0.291
## se            0.000   0.000    0.000 0.000 0.000    0.000
## cov.prob      0.964   0.946    0.964 0.964 0.964    0.946
## se            0.002   0.002    0.002 0.002 0.002    0.002
```