HW 7 Sampling distribution and confidence interval

STAT 5400

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.

```
# slide 46
se <- qt(1 - alp/2, df = n - 1) * sqrt(apply(Xlist, 1, var) / n)
# slide 48
moe <- qnorm(alp/2, lower.tail = FALSE)*sdlist/sqrt(n)
# slide 49
moe <- qnorm(alp/2, lower.tail = FALSE)*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/

```r
CovProb <- function(n, mu=2, alp=0.05, dist, ...) {
  argsString <- paste(10000 * n, ..., sep=",")
  rdistText <- parse(text = paste0('r',dist,'(',argsString,')'))
  Xvals = eval(rdistText)
  Xlist <- matrix(Xvals, 10000, n)
  Xbarlist <- rowMeans(Xlist)
  moe <- qnorm(alp/2, lower.tail = FALSE)*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)
}

CovProb(300, mu =2, alp=0.05, 'exp',1/2)
```

```
## [1] 0.9478
```

```r
CovProb(300, mu =7.5, alp=0.05, 'unif',5,10)
```

```
## [1] 0.9509
```

```r
CovProb(300, mu =1.25, alp=0.05, 'gamma',5,4)
```

```
## [1] 0.9453
```

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.

```r
bias <- function (predicted, actual) {
  return (mean(predicted - actual))
}

mse <- function(predicted, actual) {
    return (mean((actual - predicted)^2))
}

trimmed_row_means <- function (values) {
  values <- t(apply(values,1,  function(x) x[-which.max(x)]))
  values <- t(apply(values,1,  function(x) x[-which.min(x)]))

  return(rowMeans(values))
}

num_samples <- 10000
```

```r
XList <- matrix(rt(15 * num_samples,4), num_samples, 15)

Xbars <- rowMeans(XList)
XbarCaps <- trimmed_row_means(XList)

mse(XbarCaps, Xbars )
```

```
## [1] 0.01627951
```

```r
bias(XbarCaps, Xbars )
```

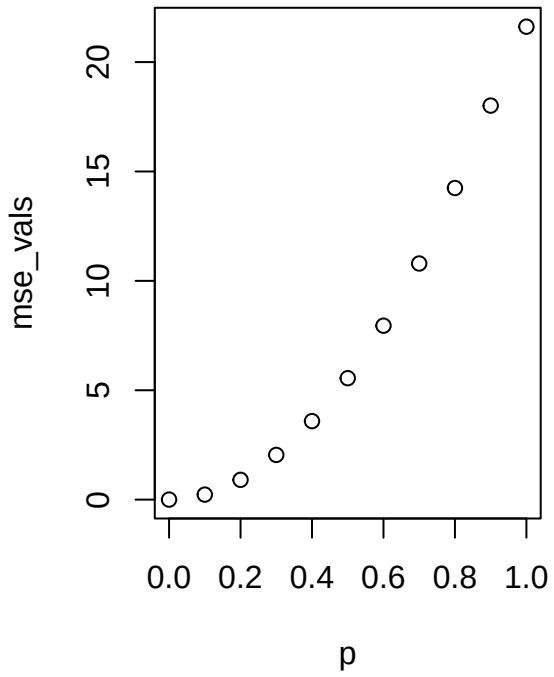
```
## [1] 0.0005256023
```

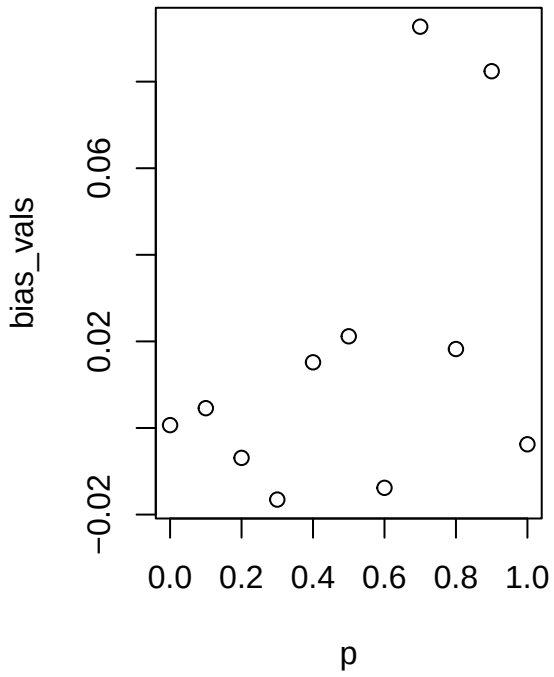
```r
var(XbarCaps)
```

```
## [1] 0.102868
```

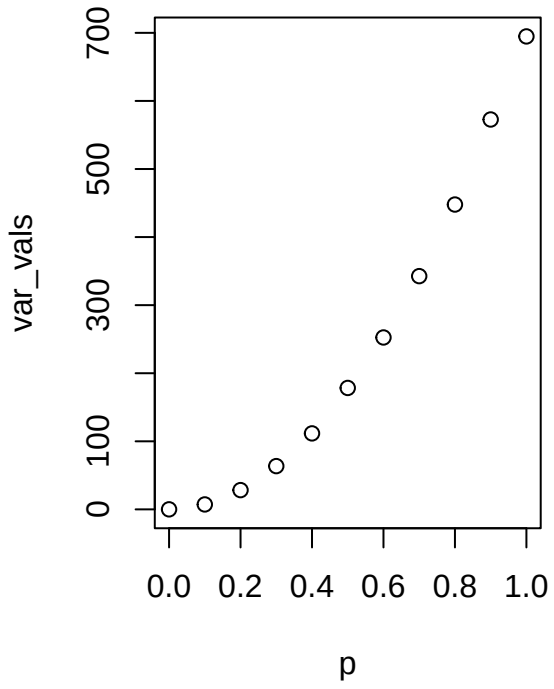
- Now suppose the data-generating model is a mixture normal distribution: $p\mathrm{N}(0,1) + (1-p)\mathrm{N}(0,10^2)$. Plot the estimate of the bias, variance, and MSE against p, where $p = (0, 0.1, 0.2, \ldots, 1)$.

```r
p <- seq(0,1,0.1)
mse_vals = vector(length = 11)
bias_vals = vector(length = 11)
var_vals = vector(length = 11)

for(i in 1:length(p)) {
  XList <- matrix((i*rnorm(15 * num_samples)) + ((1 - i)*rnorm(15 * num_samples,0,10)), num_samples, 15)

  Xbars <- rowMeans(XList)
  XbarCaps <- trimmed_row_means(XList)

  mse_vals[i] <- mse(XbarCaps, Xbars )
  bias_vals[i] <- bias(XbarCaps, Xbars )
  var_vals[i] <- var(XbarCaps)

}
par(mfrow=c(1,2))
plot(p, mse_vals)
plot(p, bias_vals)
```






```
plot(p, var_vals)
par(mfrow=c(1,1))
```



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.

```r
set.seed(5400)
dat <- rpois(20, 5)

x_bar <- mean(dat)

n <- length(dat)

se <- sqrt(x_bar / n)

z_alpha <- qnorm(1 - 0.05 / 2)

lower_bound <- x_bar - z_alpha * se
upper_bound <- x_bar + z_alpha * se

cat("95% Confidence Interval for lambda: [", lower_bound, ", ", upper_bound, "]\n")
```

```
## 95% Confidence Interval for lambda: [ 4.517525 ,  6.582475 ]
```

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

```r
set.seed(5400)
lambda <- 5
n <- 20
num_simulations <- 10^5
coverage_count <- 0

for (i in 1:num_simulations) {
  # Generate a random sample from Poisson distribution
  dat <- rpois(n, lambda)

  # Sample mean
  x_bar <- mean(dat)

  # Standard error of the sample mean
  se <- sqrt(x_bar / n)

  # Calculate the critical value using qnorm for a 95% confidence level
  z_alpha <- qnorm(1 - 0.05 / 2)

  # Confidence interval
  lower_bound <- x_bar - z_alpha * se
  upper_bound <- x_bar + z_alpha * se

  # Check if the true lambda falls within the confidence interval
  if (lower_bound <= lambda && upper_bound >= lambda) {
    coverage_count <- coverage_count + 1
  }
}
```

```r
# Estimate the coverage probability
coverage_probability <- coverage_count / num_simulations

cat("Estimated coverage probability of the 95% confidence interval:", coverage_probability, "\n")
```

```
## Estimated coverage probability of the 95% confidence interval: 0.94421
```

- 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
set.seed(123)

# Step 1: Generate sample data
n <- 30  # Sample size
lambda_true <- 5  # True mean
sample_data <- rpois(n, lambda_true)

# Step 2: Calculate the sum s
s <- sum(sample_data)

# Step 3: Calculate degrees of freedom
dof_lower <- 2 * s
dof_upper <- 2 * (s + 1)

# Step 4: Calculate critical values for the Chi-squared distribution
alpha <- 0.05
chi_lower <- qchisq(1 - alpha / 2, dof_lower)
chi_upper <- qchisq(alpha / 2, dof_upper)

# Step 5: Construct the confidence interval
confidence_interval <- c(1 / (2 * n) * chi_lower, 1 / (2 * n) * chi_upper)

# Print results
cat("Exact 95% Confidence Interval for lambda:", confidence_interval, "\n")
```

```
## Exact 95% Confidence Interval for lambda: 6.549613 4.877668
```

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

```r
# Set seed for reproducibility
set.seed(123)

# Parameters
n <- 30                # Sample size
lambda_true <- 5      # True mean
replications <- 10^5 # Number of replications
```

```r
# Initialize a counter for coverage
coverage_count <- 0

# Simulation
for (i in 1:replications) {
  # Step 1: Generate sample data
  sample_data <- rpois(n, lambda_true)

  # Step 2: Calculate the sum s
  s <- sum(sample_data)

  # Step 3: Calculate degrees of freedom
  dof_lower <- 2 * s
  dof_upper <- 2 * (s + 1)

  # Step 4: Calculate critical values for the Chi-squared distribution
  alpha <- 0.05
  chi_lower <- qchisq(1 - alpha / 2, dof_lower)
  chi_upper <- qchisq(alpha / 2, dof_upper)

  # Step 5: Construct the confidence interval
  ci_lower <- 1 / (2 * n) * chi_lower
  ci_upper <- 1 / (2 * n) * chi_upper

  # Step 6: Check if the true lambda is within the confidence interval
  if (lambda_true >= ci_lower && lambda_true <= ci_upper) {
    coverage_count <- coverage_count + 1
  }
}

# Calculate coverage probability
coverage_probability <- coverage_count / replications

# Print results
cat("Estimated Coverage Probability of the Exact Confidence Interval:", coverage_probability, "\n")
```

```
## Estimated Coverage Probability of the Exact Confidence Interval: 0
```

5. Confidence interval for proportions

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

```r
# Load necessary library
library(binom)

# Set seed for reproducibility
set.seed(123)

# Parameters
p_true <- 0.5        # True proportion
n <- 30              # Sample size
replications <- 10^5 # Number of replications
```

```
# Initialize coverage counters for each method
coverage_count_wald <- 0
coverage_count_score <- 0
coverage_count_cp <- 0
coverage_count_ac <- 0
coverage_count_bayes <- 0
coverage_count_wilson <- 0

# Simulation
# Load necessary library
library(binom)

# Set the true proportion and sample size
p_true <- 0.5
n <- 30
replications <- 10^5

# Function to calculate metrics for each CI method
WidthCov <- function(method) {
  coverage_count <- 0
  widths <- numeric(replications)
  se_values <- numeric(replications)

  for (i in 1:replications) {
    # Generate sample data
    sample_data <- rbinom(n, 1, p_true)
    x <- sum(sample_data)  # Number of successes
    p_hat <- x / n         # Sample proportion
    ci =''

    # Calculate confidence intervals based on the method
    if (method == "simple") {
      ci <- c(p_hat - qnorm(0.975) * sqrt((p_hat * (1 - p_hat)) / n),
              p_hat + qnorm(0.975) * sqrt((p_hat * (1 - p_hat)) / n))
    } else if (method == "Wald") {
      ci <- c(p_hat - qnorm(0.975) * sqrt((p_hat * (1 - p_hat)) / n),
              p_hat + qnorm(0.975) * sqrt((p_hat * (1 - p_hat)) / n))
    } else if (method == "score") {
      ci <- binom.confint(x, n, conf.level = 0.95, methods = "score")$lower[1:2]
    } else if (method == "CP") {
      ci <- binom.confint(x, n, conf.level = 0.95, methods = "exact")$lower[1:2]
    } else if (method == "AC") {
      ci <- binom.confint(x, n, conf.level = 0.95, methods = "ac")$lower[1:2]
    } else if (method == "Bayes") {
      ci <- binom.confint(x, n, conf.level = 0.95, methods = "bayes")$lower[1:2]
    }

    # Calculate metrics
    widths[i] <- diff(ci)
    se_values[i] <- sqrt((p_hat * (1 - p_hat)) / n)

    # Check coverage
    if (p_true >= ci[1] && p_true <= ci[2]) {
```

```
      coverage_count <- coverage_count + 1
    }
  }

  # Calculate average width, average SE, and coverage probability
  average_width <- mean(widths)
  average_se <- mean(se_values)
  coverage_probability <- coverage_count / replications

  return(c(exp.width = average_width, se = average_se, cov.prob = coverage_probability))
}

# List of confidence interval methods
col.nm <- paste0("CI_", c("simple", "Wald", "score", "CP", "AC", "Bayes"))

# Calculate metrics for each method and store in a matrix
res <- sapply(col.nm, WidthCov)

# Set column and row names
colnames(res) <- col.nm
rownames(res) <- c("exp.width", "se", "cov.prob")

# Round the results and display
res <- round(res, 3)
print(res)
```