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.

#just paste the code alp = NULL n = NULL Xlist = NULL sd = NULL

se <- qt(1 - alp/2, n-1)*(apply(Xlist, 1, sd) / sqrt(n)) se <- qnorm(1 - alp/2)*(apply(Xlist, 1, sd) / sqrt(n)) moe <- qnorm(1 - alp/2)*(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/

```
set.seed(5400)
CovProb <- function(n, mu=2, alp=0.05, dist, ...) {
Xlist <- matrix(do.call(paste0("r", dist), c(10000*n, list(...))), 10000, n)
Xbarlist <- rowMeans(Xlist)
moe <- qnorm(1 - alp/2)*(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)
```

```
}

# Example Usage:
set.seed(5400)
CovProb(n = 10, dis = "exp", rate = 1/2)  # exponential distribution

## [1] 0.8687

CovProb(n = 10, dis = "unif", min = 0, max = 10)  # uniform distribution

## [1] 0.088

CovProb(n = 10,  dis = "gamma", shape = 2, scale = 2)  # gamma distribution

## [1] 0.3324
```

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

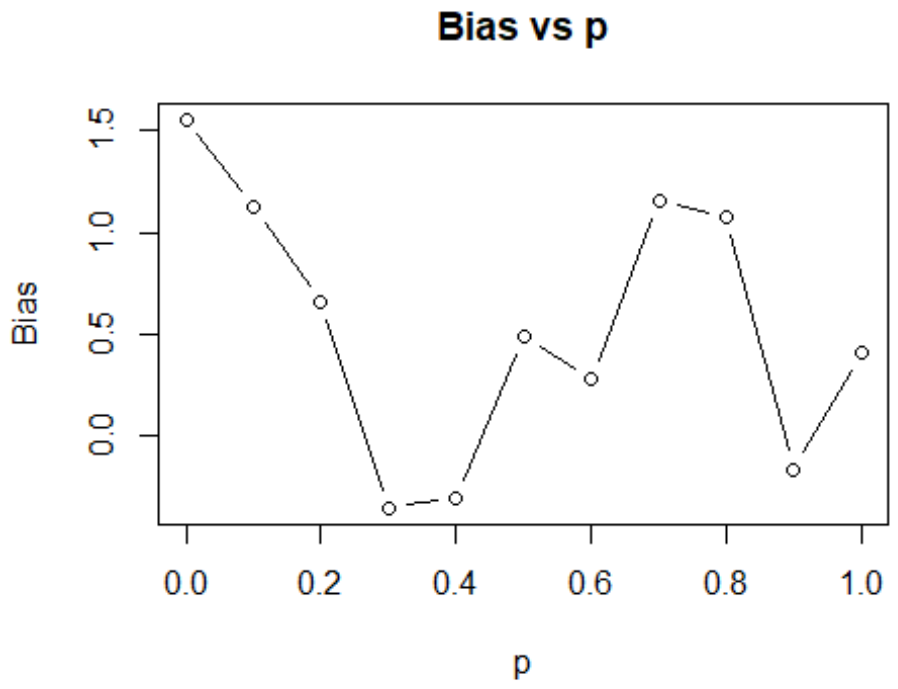
Suppose $\hat{\theta}$ is an estimator of a population parameter θ. The bias is defined as $\mathrm{E}(\hat{\theta} - \theta)$, and the mean squared error (MSE) is defined as $\mathrm{E}(\hat{\theta} - \theta)^2$.

Suppose $X_1, \dots, X_{15}$ 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.

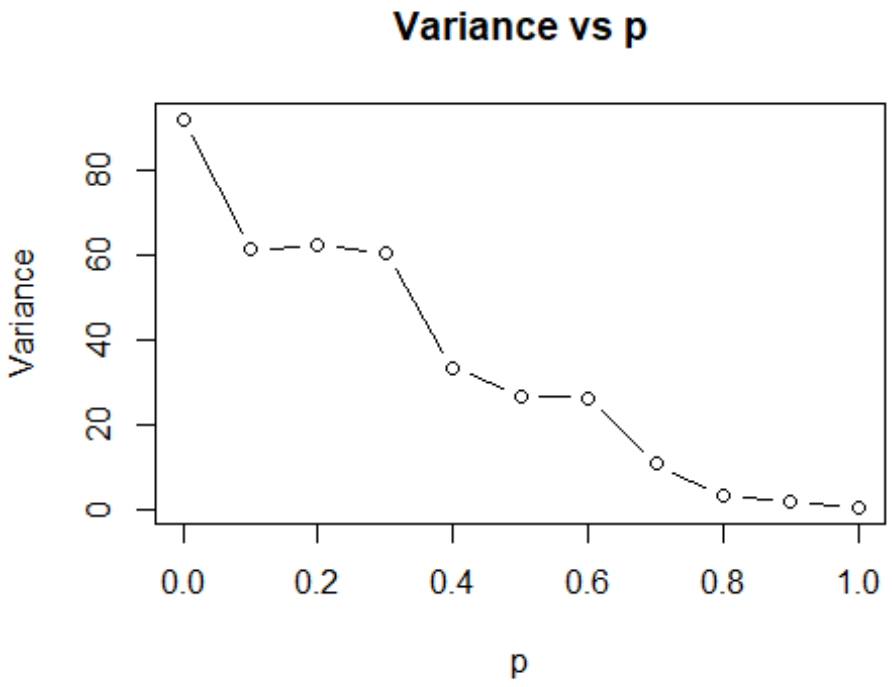
- Estimate the bias, variance, and MSE of the trimmed mean using simulations.
- Now suppose the data-generating model is a mixture normal distribution: $p\mathrm{N}(0,1) + (1-p)\mathrm{N}(0, 10^2)$. Plot the estimate of the bias, variance, and MSE against p, where $p = (0, 0.1, 0.2, \dots, 1)$.

```
n <- 15
X <- rt(n, df = 4)
trimmed_mean <- mean(sort(X)[-c(1, n)])
true_mean <- 0
bias_t <- trimmed_mean - true_mean
var_t <- var(X)
mse_t <- mean((trimmed_mean - true_mean)^2)
print(bias_t)

## [1] 0.2172351

print(var_t)

## [1] 2.67141

print(mse_t)

## [1] 0.04719107

simulate_mixture <- function(p) {
Z <- rnorm(n, mean = 0, sd = sqrt(1)) # Simulate N(0, 1) component
```

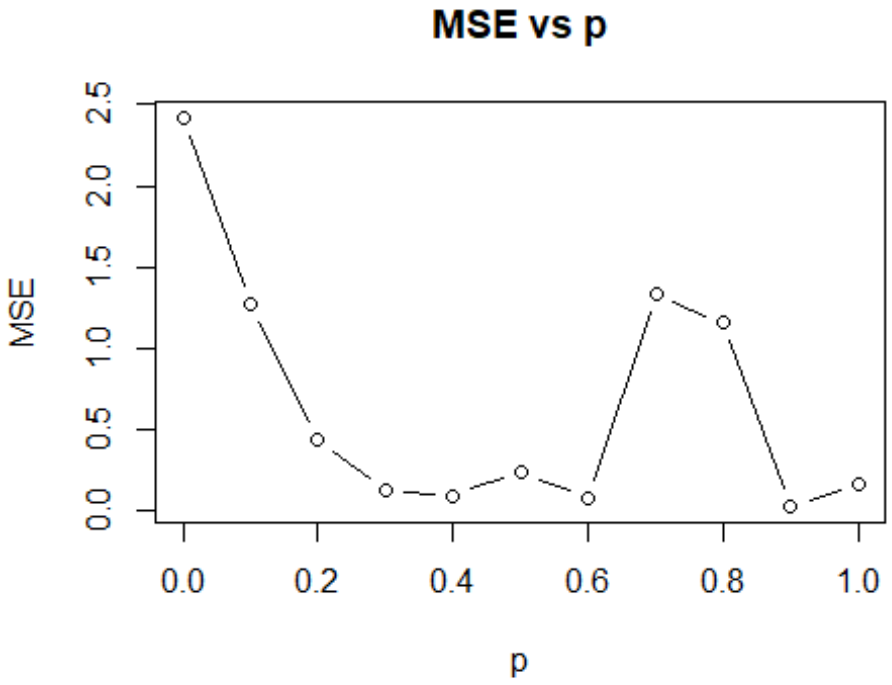
```r
W <- rnorm(n, mean = 0, sd = sqrt(100)) # Simulate N(0, 10^2) component
X <- p * Z + (1 - p) * W # Mixture
return(X)
}
p_values <- seq(0, 1, by = 0.1)
results <- data.frame(p = p_values, bias = numeric(length(p_values)), var =
numeric(length(p_values)), mse = numeric(length(p_values)))
for (i in seq_along(p_values)) {
X <- simulate_mixture(p_values[i])
trimmed_mean <- mean(sort(X)[-c(1, n)])
bias <- trimmed_mean
var <- var(X)
mse <- mean((trimmed_mean - true_mean)^2)
results[i, c("bias", "var", "mse")] <- c(bias, var, mse)
}
plot(p_values, results$bias, type = "b", xlab = "p", ylab = "Bias", main =
"Bias vs p")
```




```r
plot(p_values, results$var, type = "b", xlab = "p", ylab = "Variance", main =
"Variance vs p")
```




```r
plot(p_values, results$mse, type = "b", xlab = "p", ylab = "MSE", main = "MSE vs p")
```



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)
mean_dat = mean(dat)
sd_dat = sd(dat)
n_dat = length(dat)
ci_approx = c(mean_dat - qnorm(0.975)*sd_dat/sqrt(n_dat), mean_dat +
qnorm(0.975)*sd_dat/sqrt(n_dat))
print(ci_approx)
```

```
## [1] 4.32291 6.77709
```

- Based on this sample, construct an approximated 95% confidence interval for the mean λ using central limit theorem.
- Estimate the coverage probability of this approximated confidence interval using simulations with 10^5 replications.
- 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.
- Estimate the coverage probability of the exact confidence interval using simulations with 10^5 replications.

```
set.seed(5400)

# Number of data points
n_dat = 20

# Run simulations to calculate approximate coverage probability
simulations = replicate(10^5, {
  dat_sim = rpois(n_dat, 5)  # Simulate Poisson data
  mean_sim = mean(dat_sim)  # Calculate mean
  sd_sim = sd(dat_sim)  # Calculate standard deviation
  # Approximate confidence interval using normal approximation
  ci_sim = c(mean_sim - qnorm(0.975)*sd_sim/sqrt(n_dat), mean_sim +
qnorm(0.975)*sd_sim/sqrt(n_dat))
  # Check if the true mean (5) is within the CI
  ci_sim[1] <= 5 && ci_sim[2] >= 5
})

# Calculate approximate coverage probability
coverage_prob_approx = mean(simulations)
print(coverage_prob_approx)
```

```
## [1] 0.93226

# Calculate exact confidence interval based on the sum of the data
dat = rpois(n_dat, 5)  # Example data
s = sum(dat)  # Sum of the data
# Exact confidence interval for Poisson parameter
ci_exact = c(qchisq(0.025, 2*s)/(2*n_dat), qchisq(0.975, 2*(s+1))/(2*n_dat))
print(ci_exact)

## [1] 4.023116 6.026447

set.seed(5400)
simulations = replicate(10^5, {
dat_sim = rpois(20, 5)
s_sim = sum(dat_sim)
ci_sim = c(qchisq(0.025, 2*s_sim)/(2*n_dat), qchisq(0.975,
2*(s_sim+1))/(2*n_dat))
ci_sim[1] <= 5 && ci_sim[2] >= 5
})
coverage_prob_exact = mean(simulations)
print(coverage_prob_exact)

## [1] 0.95419
```

5. Confidence interval for proportions

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

```
library(binom)

## Warning: package 'binom' was built under R version 4.2.3

n_sims <- 10000    # Number of simulations
n_trials <- 100    # Sample size
true_prop <- 0.5   # True proportion

simple_coverage <- numeric(n_sims)
wald_coverage <- numeric(n_sims)
score_coverage <- numeric(n_sims)
cp_coverage <- numeric(n_sims)
ac_coverage <- numeric(n_sims)
bayes_coverage <- numeric(n_sims)

for (i in 1:n_sims) {
    sample_data <- rbinom(n_trials, 1, true_prop)
    est_prop <- mean(sample_data)

    simple_ci <- binom.confint(sum(sample_data), n_trials, methods =
"asymptotic")
```

```r
    simple_coverage[i] <- (simple_ci$lower <= true_prop & true_prop <=
simple_ci$upper)

    se_est <- sqrt(est_prop * (1 - est_prop) / n_trials)
    z_val <- qnorm(1 - (1 - 0.95) / 2)
    wald_lower <- est_prop - z_val * se_est
    wald_upper <- est_prop + z_val * se_est
    wald_coverage[i] <- (wald_lower <= true_prop & true_prop <= wald_upper)

    score_ci <- binom.confint(sum(sample_data), n_trials, methods = "wilson")
    score_coverage[i] <- (score_ci$lower <= true_prop & true_prop <=
score_ci$upper)

    cp_ci <- binom.confint(sum(sample_data), n_trials, methods = "exact")
    cp_coverage[i] <- (cp_ci$lower <= true_prop & true_prop <= cp_ci$upper)

    ac_ci <- binom.confint(sum(sample_data), n_trials, methods = "ac")
    ac_coverage[i] <- (ac_ci$lower <= true_prop & true_prop <= ac_ci$upper)

    bayes_ci <- binom.bayes(sum(sample_data), n_trials)
    bayes_coverage[i] <- (bayes_ci$lower <= true_prop & true_prop <=
bayes_ci$upper)
}

print(paste("Simple CI Coverage:", mean(simple_coverage)))
```

```
## [1] "Simple CI Coverage: 0.9439"
```

```r
print(paste("Wald CI Coverage:", mean(wald_coverage)))
```

```
## [1] "Wald CI Coverage: 0.9439"
```

```r
print(paste("Score CI Coverage:", mean(score_coverage)))
```

```
## [1] "Score CI Coverage: 0.9439"
```

```r
print(paste("CP CI Coverage:", mean(cp_coverage)))
```

```
## [1] "CP CI Coverage: 0.9624"
```

```r
print(paste("AC CI Coverage:", mean(ac_coverage)))
```

```
## [1] "AC CI Coverage: 0.9439"
```

```r
print(paste("Bayes CI Coverage:", mean(bayes_coverage)))
```

```
## [1] "Bayes CI Coverage: 0.9439"
```