ST 502 Project 1

Taylor Cesarski and Jie Chen

2024-10-05

Introduction and Goals of Simulation Study

For this study, we will utilize simulation to analyze different confidence interval methods for estimating a true probability of success (p) from a binomial random sample. We will look at the following confidence interval methods:

1. Wald Interval
2. Adjusted Wald Interval
3. Clopper-Pearson Interval
4. Score Interval
5. Raw Percentile Interval using a Parametric Bootstrap
6. Bootstrap T Interval using a Parametric Bootstrap

To determine the best method, we will use simulation to generate many confidence intervals at varying sample sizes and probabilities of success. From there, we will look at the average length of these intervals, the standard errors of these lengths, the observed coverage probability, and the percentage of intervals that are missing below the true proportion and the percentage of intervals that are missing above the true proportion. Since conducting a confidence interval is "one of the most basic analyses in statistical inference (Agresti & Coull, 1998, p. 119), we want to form intervals that are both accurate, meaningful, and do not have excess variability.

Methods Used & Calculations

Method 1: Wald Interval

The first method we will examine is the Wald Interval. The Wald interval is often presented in introductory statistics courses as the method used to calculate a confidence interval for a sample proportion. It relies on the Central Limit Theorem and estimates the standard error with use of the sample proportion instead of the true proportion (since this in theory would be unknown). We generate this interval by taking the sample proportion and adding and subtracting the critical value from the standard normal distribution multiplied by the estimated standard error.

The Wald Interval can be found using the following function:

```r
#Create a Wald Interval function.

#Let the default for alpha be 0.05.

wald_int <- function(y, n, alpha = 0.05){

  #Generate the lower and upper endpoints using sample proportion,
  #critical value from standard normal distribution,
  #and estimated standard error.

  c(y/n -qnorm(1-alpha/2)*sqrt(((y/n)*(1-y/n))/n),

    y/n +qnorm(1-alpha/2)*sqrt(((y/n)*(1-y/n))/n))

}
```

Method 2: Adjusted Wald Interval

Because of the poor performance of the Wald interval for small samples or extreme p values, Adjusted Wald Interval is introduced as an improvement for estimating a Binomial proportion. The Adjusted Wald interval can be calculated by 'adding two successes and two failures and then use the Wald formula'. We can define the adjusted proportion: $\hat{p} = (X + 2)/(n + 4)$ and the adjusted sample size: n+4.

Function for Adjusted Wald Interval Calculation

```r
adjusted_waldCI <- function (y, n, alpha=0.05){
  c((y+2)/(n+4) - qnorm(1-alpha/2)*sqrt(((y+2)/(n+4))*(1-((y+2)/(n+4)))/(n+4)),
    (y+2)/(n+4) + qnorm(1-alpha/2)*sqrt(((y+2)/(n+4))*(1-((y+2)/(n+4)))/(n+4)))
}
```

Method 3: Clopper-Pearson Interval

The third method we will examine is the Clopper-Pearson Interval. This is referred to as an exact interval because it doesn't rely on the asymptotic normality seen in the Wald Interval. The Clopper-Pearson Interval is based on the idea of "inverting equal-tailed binomial tests" and it ensures that the coverage probability is at least (1-α) (Agresti & Coull, 1988, p. 119). However, although this is treated as the best method by many textbooks, it actually can often overshoot the desired coverage probability unless the sample size is large. The endpoints for this interval come from the $F_{a,b,c}$ distribution where a represents the numerator degrees of freedom, b represents the denominator degrees of freedom, and c represents the probability/area to the right.

The Clopper-Pearson Interval can be found using the following function:

```r
#Create a Clopper-Pearson Interval function.

#Let the default for alpha be 0.05.

CP_int <- function(y, n, alpha = 0.05){

  #Consider cases where y =0 or n and assign intervals.

  if(y==0){
```

```
    return(c(0,0))
  }
  else if(y==n){
    return(c(1,1))
  }
  #If y does not equal 0 or n, generate Clopper-Pearson interval based on F distribution.
  else{
  c((1 + (n-y+1)/(y*qf(alpha/2,2*y, 2*(n-y+1))))^-1,
    (1 + (n-y)/((y+1)*qf(1-alpha/2, 2*(y+1), 2*(n-y))))^-1)
  }
}
```

Method 4: Score Interval

Score interval is an improvement over the Wald interval, providing better performance, especially for small sample sizes or proportions near 0 or 1. The center of score interval is the weighted average $\hat{p} + \frac{z^2_{\frac{\alpha}{2}}}{2n}$. The estimated standard error will be $\sqrt{\frac{\hat{p}(1-\hat{p})}{n} + \frac{z^2_{\frac{\alpha}{2}}}{4n^2}}$.

The Score Interval can be found with the following function.

```
# Score Interval Function
scoreCI <- function(y, n, alpha = 0.05) {
  # Sample proportion
  p_hat <- y / n

  # Critical value for the desired confidence level (alpha/2)
  z <- qnorm(1 - alpha / 2)

  # center of the interval
  center <- (p_hat + (z^2 / (2 * n)))

  # Calculate the margin of error
  error_term <- z * sqrt((p_hat * (1 - p_hat) + (z^2 / (4 * n)))/n)

  # Calculate the lower and upper bounds
  lower_bound <- (center - error_term)/ (1 + (z^2 / n))
  upper_bound <- (center + error_term) / (1 + (z^2 / n))

  # Return the score interval
  return(c(score_lower<-lower_bound, score_upper<-upper_bound))

}
```

Method 5: Raw Percentile Interval using a Parametric Bootstrap

The fifth method we will examine is a raw percentile interval using a parametric bootstrap. The idea of a parametric bootstrap is to resample from a fitted distribution to get many samples of the same size (that correspond to the original sample size). Then find the bootstrap estimate (in this case the sample proportion) and generate the bootstrap distribution. Finally, we will use the (1-$\alpha/2$) and $\alpha/2$ quantiles from this bootstrap distribution as the confidence interval.

Method 6: Bootstrap t interval using a parametric bootstrap

The parametric bootstrap t-interval is based on resampling from this specific parametric distribution to estimate the statistic's variability and the confidence interval. We will need to first fit a parametric model to our data (here is a binomial distribution). Then, we generate bootstrap samples from the fitted distribution (rather than resampling directly from the observed data). For each bootstrap sample, compute the bootstrap estimate (e.g., sample proportion) and bootstrap standard error of the statistic. Use the bootstrap t-statistic, to compute the percentiles of the t-statistics from the bootstrap samples to find the appropriate cutoff points for the confidence interval.

We will create one function that finds the two bootstrap based intervals for saving computational time.

```
bootstrap_int <- function(y, n, alpha = 0.05, B = 100) {

  # Calculate the sample proportion.
  phat <- y / n

   if (y == 0) {return(list(bootstrap_per_interval = c(0, 0), bootstrap_t_interval = c(0, 0)))
   }
  else if (y == n) {return(list(bootstrap_per_interval = c(1, 1), bootstrap_t_interval = c(1, 1))) }
  else{

  # Get B resamples of size n and compute the sample proportion for each.
  boot_estimates <- replicate(B, {

    boot_sample <- rbinom(1, size = n, prob = phat)
    boot_phat <- boot_sample / n

    return(boot_phat)
  })

  # Calculate the raw percentile interval from the bootstrapped proportions
  lower_percentile <- quantile(boot_estimates, alpha / 2)
  upper_percentile <- quantile(boot_estimates, 1 - alpha / 2)

  # Secondary bootstrap for standard error estimation and t-statistics
  secondary_boot <- replicate(B, {
    boot_sample2 <- rbinom(1, size = n, prob = phat)
    boot_phat2 <- boot_sample2 / n
    return(boot_phat2)
  })

    estimated_SE_phat_boot <- sd(secondary_boot)

  #If the standard error is too small (close to zero), return NA for the t-interval
  if (estimated_SE_phat_boot < 1e-10) {
```

```
    lower_boot_t <- NA
    upper_boot_t <- NA
  } else {
    #Calculate the t-statistics
    boot_phat_t <- (boot_estimates - phat) / estimated_SE_phat_boot

    #Calculate the critical values (percentiles of the t-statistics)
    lower_t_percentile <- quantile(boot_phat_t, alpha / 2, na.rm=TRUE)
    upper_t_percentile <- quantile(boot_phat_t, 1 - alpha / 2, na.rm = TRUE)

    # Calculate the bootstrap t-interval
    lower_boot_t <- phat - upper_t_percentile * estimated_SE_phat_boot
    upper_boot_t <- phat - lower_t_percentile * estimated_SE_phat_boot
  }

}
  # Return both intervals as a list
  return(list(
    bootstrap_per_interval = c(lower_percentile, upper_percentile),
    bootstrap_t_interval = c(lower_boot_t, upper_boot_t)
  ))
}
```

Creation of Data Process and Code

We want to generate 1000 random samples from a binomial where n varies across n=15, 30, and 100 and p values from 0.01 to 0.99 for 15 total values of p for a total of 45 combinations. We will then generate 1000 95% confidence intervals for each of our combinations and find properties of these confidence intervals. In order for a confidence interval to be "good", we want the observed coverage probability to be equal to or close to (1-α). Some intervals will inherently miss where the true probability falls below or above the interval. Ideally, we would want these proportions that miss on both sides to be equal to $\alpha/2$. We also want a relatively narrow interval so that it gives us helpful information. For example, it is not helpful to say the true probability is between 0 and 1 or even 0.05 and 0.95 because the interval is too wide to even be beneficial. Therefore, we want to compare the average lengths of our intervals.

We will start by generating the 1000 confidence intervals for each method.

```
#Set the seed for reproducibility.

set.seed(200)

# set to N=1000 random samples
N<-1000

#Create a vector of sample sizes.

sample_sizes <- c(15, 30, 100)

# Create a vector of 15 values of p ranging from 0.01 to 0.99
p <- seq(0.01, 0.99, length.out = 15)

# Print the vector
print(p)
```

```
##  [1] 0.01 0.08 0.15 0.22 0.29 0.36 0.43 0.50 0.57 0.64 0.71 0.78 0.85 0.92 0.99
```

```
# Initialize lists to store each data frame of CIs.
wald_confidence_int <- list()
adjusted_wald_CI <- list()
cp_confidence_int <- list()
score_CI <-list()
boot_per_CI <- list()
boot_t_CI <- list()
#Iterate through all sample sizes.
for(size in sample_sizes){
  #Iterate through all probabilities.
  for(prob in p){
  #Intialize data frames to store the confidence intervals.
   wald_CIs <- data.frame(lower=numeric(N), upper=numeric(N))
   adjusted_waldCIs <- data.frame(lower=numeric(N), upper=numeric(N))
   cp_CIs <- data.frame(lower=numeric(N), upper=numeric(N))
   scoreCIs <- data.frame(lower=numeric(N), upper=numeric(N))
   boot_perCIs <- data.frame(lower=numeric(N), upper=numeric(N))
   boot_tCIs <- data.frame(lower=numeric(N), upper=numeric(N))

   #Generate the random data based on n and p.
   data <- rbinom(n=N, size, prob)
#Get 1000 samples.
for(i in 1:N){
  y<-data[i]

  #Apply the wald_int function and store the lower & upper endpoint (95% CI).
```

```
  wald_interval <- wald_int(y, size)

  wald_CIs$lower[i] <- wald_interval[1]

  wald_CIs$upper[i] <- wald_interval[2]

  # Apply the adjusted wald interval function
  adjusted_wald_interval <- adjusted_waldCI(y, size)

  adjusted_waldCIs$lower[i] <- adjusted_wald_interval[1]

  adjusted_waldCIs$upper[i] <- adjusted_wald_interval[2]


  #Apply the CP_int function and store the lower & upper endpoint.

  cp_interval <- CP_int(y, size)

  cp_CIs$lower[i] <- cp_interval[1]

  cp_CIs$upper[i] <- cp_interval[2]

  # Apply score interval function and store the lower and upper endpoint

  score_interval <- scoreCI(y,size)

  scoreCIs$lower[i] <- score_interval[1]

  scoreCIs$upper[i] <- score_interval[2]


  #Apply the bootstrap_int function and store the lower & upper endpoint for both methods.

  boot_interval <- bootstrap_int(y, size)

  boot_perCIs$lower[i] <- boot_interval$bootstrap_per_interval[1]

  boot_perCIs$upper[i] <- boot_interval$bootstrap_per_interval[2]

  boot_tCIs$lower[i] <-boot_interval$bootstrap_t_interval[1]

  boot_tCIs$upper[i] <-boot_interval$bootstrap_t_interval[2]
}

  #Store each data frame in the list with names.

  wald_confidence_int[[paste0("Wald", "size_", size, "prob_", prob)]] <- wald_CIs

  adjusted_wald_CI[[paste0("Adjusted Wald","size_", size,"prob_", prob)]]<-adjusted_waldCIs

  cp_confidence_int[[paste0("Clopper-Pearson", "size_", size, "prob_", prob)]] <- cp_CIs
```

```
  score_CI[[paste0("Score", "size_", size, "prob_", prob)]] <- scoreCIs

  boot_per_CI[[paste0("Bootstrap Percentile", "size_", size, "prob_", prob)]] <- boot_perCIs

  boot_t_CI[[paste0("Bootstrap t interval", "size_", size, "prob_", prob)]] <- boot_tCIs

  }
}
```

We will create a data frame for the each type of interval that contains the sample size (n), the probability of success (p), the average length of the intervals, the standard errors of the average lengths, the coverage probability, the proportion of intervals that miss above, and the proportion of intervals that miss below. We will also produce plots to show the average lengths and coverage probabilities across different sample sizes.

```
#Initialize a lengths numeric vector to hold average lengths of size 45.

wald_lengths <- numeric(45)

adjusted_wald_lengths <- numeric(45)

cp_lengths <- numeric(45)

score_lengths <-numeric(45)

boot_per_lengths <- numeric(45)

boot_t_lengths <-numeric(45)

wald_se <- numeric(45)


#Use a for loop to calculate the average lengths for each set of 1000 intervals.

for(i in 1:45){

  wald_lengths[i] <- mean(wald_confidence_int[[i]]$upper - wald_confidence_int[[i]]$lower)

  wald_se[i] <- sd(wald_confidence_int[[i]]$upper - wald_confidence_int[[i]]$lower)

  adjusted_wald_lengths[i] <- mean(adjusted_wald_CI[[i]]$upper - adjusted_wald_CI[[i]]$lower)

  cp_lengths[i] <- mean(cp_confidence_int[[i]]$upper - cp_confidence_int[[i]]$lower)

  score_lengths[i] <- mean(score_CI[[i]]$upper - score_CI[[i]]$lower)

  boot_per_lengths[i] <- mean(boot_per_CI[[i]]$upper - boot_per_CI[[i]]$lower)

  boot_t_lengths[i] <- mean(boot_t_CI[[i]]$upper -  boot_t_CI[[i]]$lower)

}


# Create a vector that contains the probabilities 3 times.
```

```r
repeated_p <- rep(p, 3)



#Initialize a numeric vector of length 45 to hold the coverage probabilities.

wald_coverage_probabilities <- numeric(45)

adjusted_wald_coverage_probabilities <- numeric(45)

cp_coverage_probabilities <- numeric(45)

score_coverage_probabilities <-numeric(45)

boot_per_coverage_probabilities <- numeric(45)

boot_t_coverage_probabilities <-numeric(45)



#Use a for loop to calculate the coverage probability (% containing true p) for each set of 1000 interv

for (i in 1:45) {

  wald_coverage_probabilities[i] <-
    mean(wald_confidence_int[[i]]$upper > repeated_p[i] &
           wald_confidence_int[[i]]$lower < repeated_p[i])

  adjusted_wald_coverage_probabilities[i] <-
    mean(adjusted_wald_CI[[i]]$upper > repeated_p[i] &
           adjusted_wald_CI[[i]]$lower < repeated_p[i])

  cp_coverage_probabilities[i] <-
    mean(cp_confidence_int[[i]]$upper > repeated_p[i] &
           cp_confidence_int[[i]]$lower < repeated_p[i])

  score_coverage_probabilities[i] <-
    mean(score_CI[[i]]$upper > repeated_p[i] &
           score_CI[[i]]$lower < repeated_p[i])

  boot_per_coverage_probabilities[i] <-
    mean(boot_per_CI[[i]]$upper > repeated_p[i] &
           boot_per_CI[[i]]$lower < repeated_p[i])

  boot_t_coverage_probabilities[i] <-
    mean(boot_t_CI[[i]]$upper > repeated_p[i] &
           boot_t_CI[[i]]$lower < repeated_p[i])

}


#Initalize numeric vectors of length 45 to store the percentage of intervals missing too low.
```

```r
wald_miss_low <- numeric(45)

adjusted_wald_miss_low <- numeric(45)

cp_miss_low <- numeric(45)

score_miss_low <- numeric(45)

boot_per_miss_low <- numeric(45)

boot_t_miss_low <-numeric(45)


#Find the percentage where the true p is above the interval.

for (i in 1:45) {

  wald_miss_low[i] <- mean(wald_confidence_int[[i]]$upper < repeated_p[i])

  adjusted_wald_miss_low[i] <- mean(adjusted_wald_CI[[i]]$upper < repeated_p[i])

  cp_miss_low[i] <- mean(cp_confidence_int[[i]]$upper < repeated_p[i])

  score_miss_low[i] <- mean(score_CI[[i]]$upper < repeated_p[i])

  boot_per_miss_low[i] <- mean(boot_per_CI[[i]]$upper < repeated_p[i])

  boot_t_miss_low[i]<-mean(boot_t_CI[[i]]$upper < repeated_p[i])
}


# Initalize numeric vectors of length 45 to store the percentage of intervals missing too high.

wald_miss_high <- numeric(45)

adjusted_wald_miss_high <-numeric(45)

cp_miss_high <- numeric(45)

score_miss_high <-numeric(45)

boot_per_miss_high <- numeric(45)

boot_t_miss_high <- numeric(45)


# Find the percentage where the true p is below the interval.

for (i in 1:45) {

  wald_miss_high[i] <- mean(wald_confidence_int[[i]]$lower > repeated_p[i])
```

```
  adjusted_wald_miss_high[i] <- mean(adjusted_wald_CI[[i]]$lower > repeated_p[i])

  cp_miss_high[i] <- mean(cp_confidence_int[[i]]$lower > repeated_p[i])

  score_miss_high[i] <- mean(score_CI[[i]]$lower > repeated_p[i])

  boot_per_miss_high[i] <- mean(boot_per_CI[[i]]$lower > repeated_p[i])

  boot_t_miss_high[i] <- mean(boot_t_CI[[i]]$lower > repeated_p[i])

}


#Create a vector of the values of n.

n_values <- c(rep(15,15), rep(30, 15), rep(100,15))

#Use the repeated p values created above.

repeated_p <- rep(p, 3)



#Store the n, p, lengths, coverage probabilities, and % missing above and below.

wald_summary <- data.frame(n = n_values,
                           p = repeated_p,
                           avg_int_lengths = wald_lengths,
                           coverage_probs = wald_coverage_probabilities,
                           miss_low = wald_miss_low,
                           miss_high = wald_miss_high)

#Display first few rows of wald_summary.
head(wald_summary)
```

```
##    n    p avg_int_lengths coverage_probs miss_low miss_high
## 1 15 0.01      0.03134204          0.121    0.879     0.000
## 2 15 0.08      0.22143421          0.715    0.281     0.004
## 3 15 0.15      0.32712162          0.897    0.085     0.018
## 4 15 0.22      0.39282286          0.865    0.130     0.005
## 5 15 0.29      0.43886956          0.950    0.040     0.010
## 6 15 0.36      0.46729810          0.935    0.054     0.011
```

```
adjusted_wald_summary <- data.frame(n = n_values,
                                    p = repeated_p,
                                    avg_int_lengths = adjusted_wald_lengths,
                                    coverage_probs = adjusted_wald_coverage_probabilities,
                                    miss_low = adjusted_wald_miss_low,
                                    miss_high = adjusted_wald_miss_high)

#Display first few rows of adjusted_wald_summary.
head(adjusted_wald_summary)
```

```
##    n    p avg_int_lengths coverage_probs miss_low miss_high
## 1 15 0.01       0.2826091          0.992    0.000     0.008
## 2 15 0.08       0.3307491          0.972    0.000     0.028
## 3 15 0.15       0.3665456          0.946    0.000     0.054
## 4 15 0.22       0.3938505          0.966    0.000     0.034
## 5 15 0.29       0.4148032          0.960    0.003     0.037
## 6 15 0.36       0.4292020          0.975    0.014     0.011
```

```
cp_summary <- data.frame(n = n_values,
                         p = repeated_p,
                         avg_int_lengths = cp_lengths,
                         coverage_probs = cp_coverage_probabilities,
                         miss_low = cp_miss_low,
                         miss_high = cp_miss_high)

#Display first few rows of cp_summary.
head(cp_summary)
```

```
##    n    p avg_int_lengths coverage_probs miss_low miss_high
## 1 15 0.01      0.03906496          0.113    0.879     0.008
## 2 15 0.08      0.25952997          0.715    0.281     0.004
## 3 15 0.15      0.36631871          0.897    0.085     0.018
## 4 15 0.22      0.42586266          0.968    0.027     0.005
## 5 15 0.29      0.46613016          0.987    0.003     0.010
## 6 15 0.36      0.48988040          0.975    0.014     0.011
```

```
score_summary <- data.frame(n = n_values,
                            p = repeated_p,
                            avg_int_lengths = score_lengths,
                            coverage_probs = score_coverage_probabilities,
                            miss_low = score_miss_low,
                            miss_high = score_miss_high)

#Display first few rows of score_summary.
head(score_summary)
```

```
##    n    p avg_int_lengths coverage_probs miss_low miss_high
## 1 15 0.01       0.2143367          0.879    0.000     0.121
## 2 15 0.08       0.2876705          0.972    0.000     0.028
## 3 15 0.15       0.3393162          0.946    0.000     0.054
## 4 15 0.22       0.3772056          0.939    0.027     0.034
## 5 15 0.29       0.4056858          0.960    0.003     0.037
## 6 15 0.36       0.4247951          0.975    0.014     0.011
```

```
boot_per_summary <- data.frame(n = n_values,
                               p = repeated_p,
                               avg_int_lengths = boot_per_lengths,
                               coverage_probs = boot_per_coverage_probabilities,
                               miss_low = boot_per_miss_low,
                               miss_high = boot_per_miss_high)

#Display first few rows of boot_per_summary.
head(boot_per_summary)
```

```
##    n    p avg_int_lengths coverage_probs miss_low miss_high
## 1 15 0.01      0.02535333          0.120    0.879     0.001
## 2 15 0.08      0.19486167          0.712    0.281     0.007
## 3 15 0.15      0.29636333          0.890    0.091     0.019
## 4 15 0.22      0.36064833          0.864    0.109     0.027
## 5 15 0.29      0.40873000          0.910    0.066     0.024
## 6 15 0.36      0.43854167          0.912    0.054     0.034
```

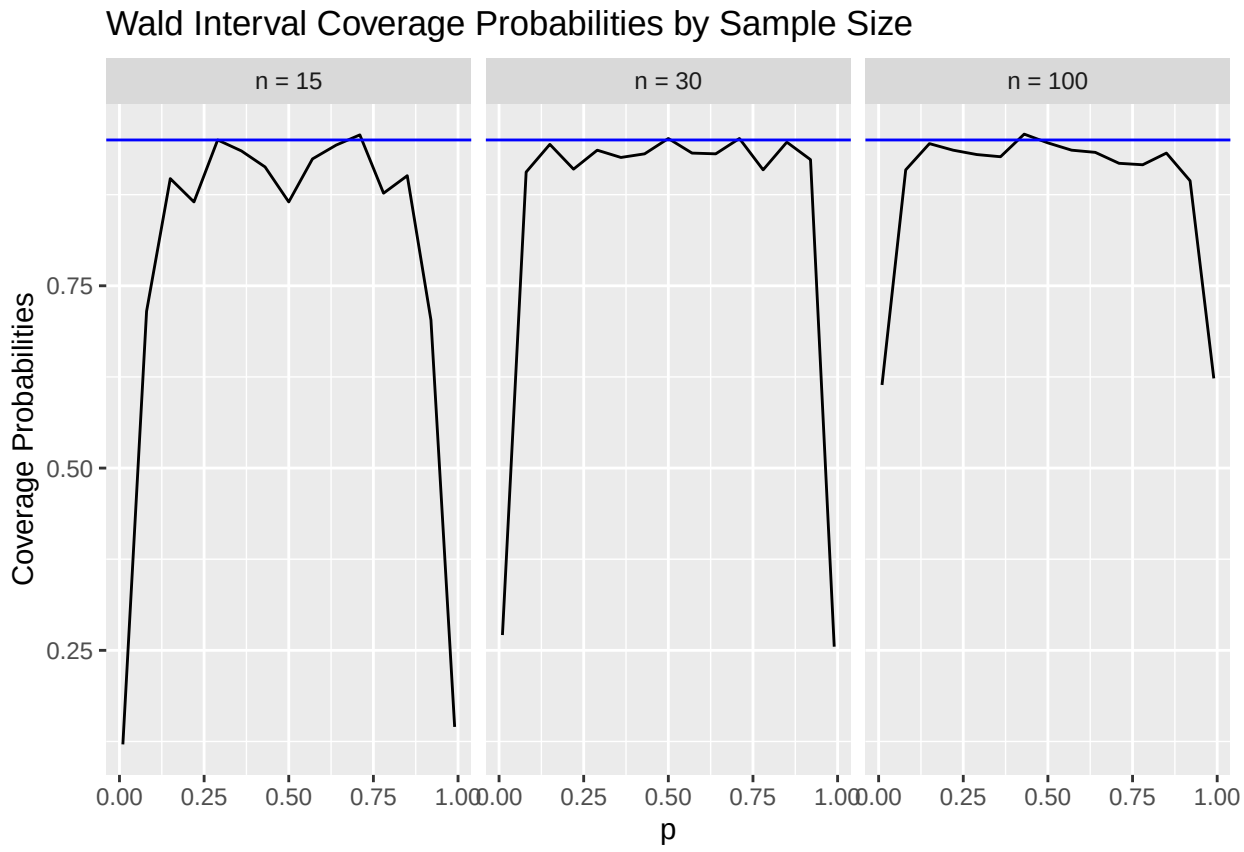
```
boot_t_summary <- data.frame(n = n_values,
                             p = repeated_p,
                             avg_int_lengths = boot_t_lengths,
                             coverage_probs = boot_t_coverage_probabilities,
                             miss_low = boot_t_miss_low,
                             miss_high = boot_t_miss_high)

#Display first few rows of boot_t_summary
head(boot_t_summary)
```

```
##    n    p avg_int_lengths coverage_probs miss_low miss_high
## 1 15 0.01      0.02535333          0.121    0.879     0.000
## 2 15 0.08      0.19486167          0.716    0.281     0.003
## 3 15 0.15      0.29636333          0.664    0.325     0.011
## 4 15 0.22      0.36064833          0.849    0.130     0.021
## 5 15 0.29      0.40873000          0.831    0.145     0.024
## 6 15 0.36      0.43854167          0.877    0.085     0.038
```

Wald Convergence Probability Plot

The plot for the Wald Interval shows that the convergence probabilities are missing under the desired 95% for almost all values of p when n=15. When n=30, five values appear very close to the 95% convergence probability, but most are still too low on convergence, with only one having more than 95% of the intervals capture the true p. When n=100, the convergence probability seems good for values of p between approximately 0.2 and 0.9, but still doesn't perform well for extreme values of p towards 0 or 1. This concept makes sense because the Wald Interval is based on the CLT which holds for large samples.

```
ggplot(wald_summary, aes(x=p, y=wald_coverage_probabilities)) +

  geom_line()+

  #Facet wrap over sample size to compare plots for different sample sizes.
  facet_wrap(~n, labeller = labeller(n = function(x) paste("n =", x))) +

  #Add title for overall plot and y axis.
  labs(title = "Wald Interval Coverage Probabilities by Sample Size",
       y = "Coverage Probabilities") +

  #Add a line at the idea convergence probability for clarity.
  geom_hline(yintercept = 0.95, color = "blue")
```

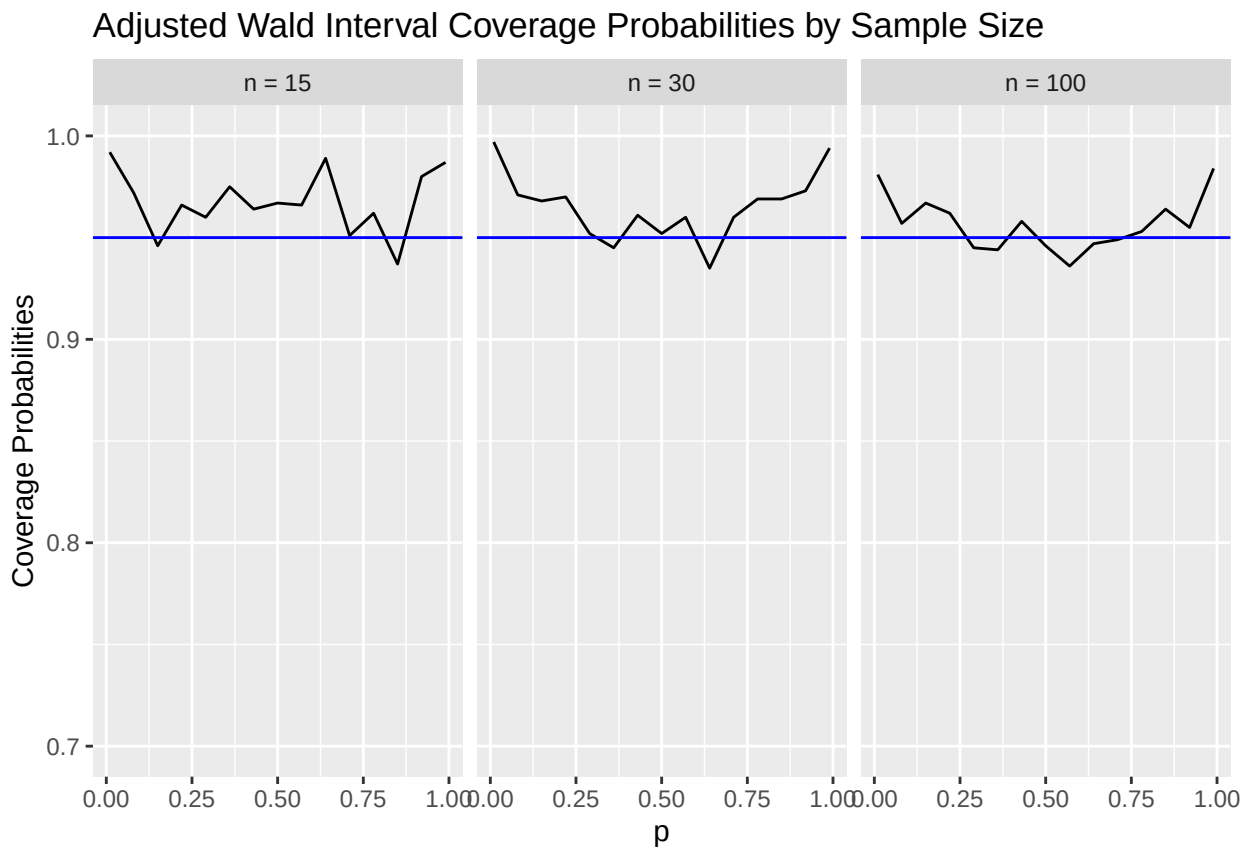


Adjusted Wald Coverage Probability Plot

The plot below for Adjusted Wald coverage probability shows that as the sample size increases from 15 to 100, the curve moves closer to the 0.95 reference line, indicating better performance. Even for the small sample size of 15, the Adjusted Wald method still outperforms the standard Wald method. Overall, the Adjusted Wald confidence intervals perform adequately for practical applications across essentially any sample size (Agresti & Coull, 1998, p. 124).

```
ggplot(adjusted_wald_summary, aes(x = p, y = adjusted_wald_coverage_probabilities)) +

  # Use a line graph to create a curve
  geom_line(color = "black") +

  # Facet wrap over sample size to compare plots for different sample sizes,
  # use labeller to show n = n_values in the strip labels
  facet_wrap(~n, labeller = labeller(n = function(x) paste("n =", x))) +

  # Add title for overall plot and y axis.
  labs(title = "Adjusted Wald Interval Coverage Probabilities by Sample Size",
       y = "Coverage Probabilities",
       x = "p") +

  ylim (0.7,1) +

  # Add a line at the ideal coverage probability (0.95) for clarity.
```

```
  geom_hline(yintercept = 0.95, color = "blue")
```

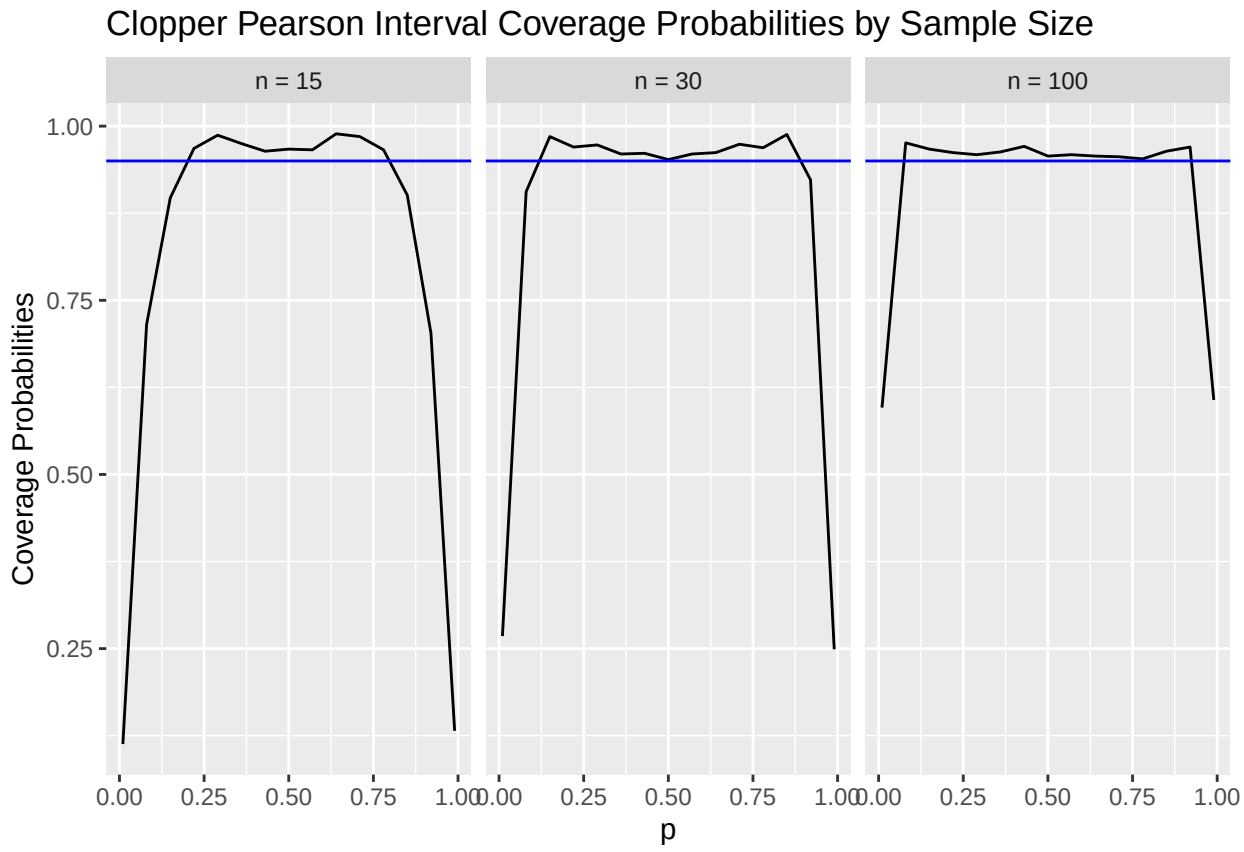


Clopper Pearson Coverage Probability Plot

The plot for the Clopper-Pearson Interval shows that the convergence probabilities perform much better for smaller sample sizes compared to the Wald Interval. This is due to the fact that the Clopper-Pearson is an exact interval and does not rely on asymptotic normality. Even at n=100, the Clopper-Pearson Interval is still not ideal for extreme values of p (0.01 and 0.99) and more conservative values of p when n is smaller. The Clopper-Pearson Interval also "overhits" the target in most cases, with convergence probabilities above the 95% ideal convergence probability, which is actually not preferred.

```
ggplot(cp_summary, aes(x=p, y=cp_coverage_probabilities)) +

  #Use point graph to make easier to see differences.

  geom_line()+

  #Facet wrap over sample size to compare plots for different sample sizes.

  facet_wrap(~n, labeller = labeller(n = function(x) paste("n =", x))) +

  #Add title for overall plot and y axis.

  labs(title = "Clopper Pearson Interval Coverage Probabilities by Sample Size",
```

```
      y = "Coverage Probabilities") +

  #Add a line at the idea convergence probability for clarity.

  geom_hline(yintercept = 0.95, color = "blue")
```

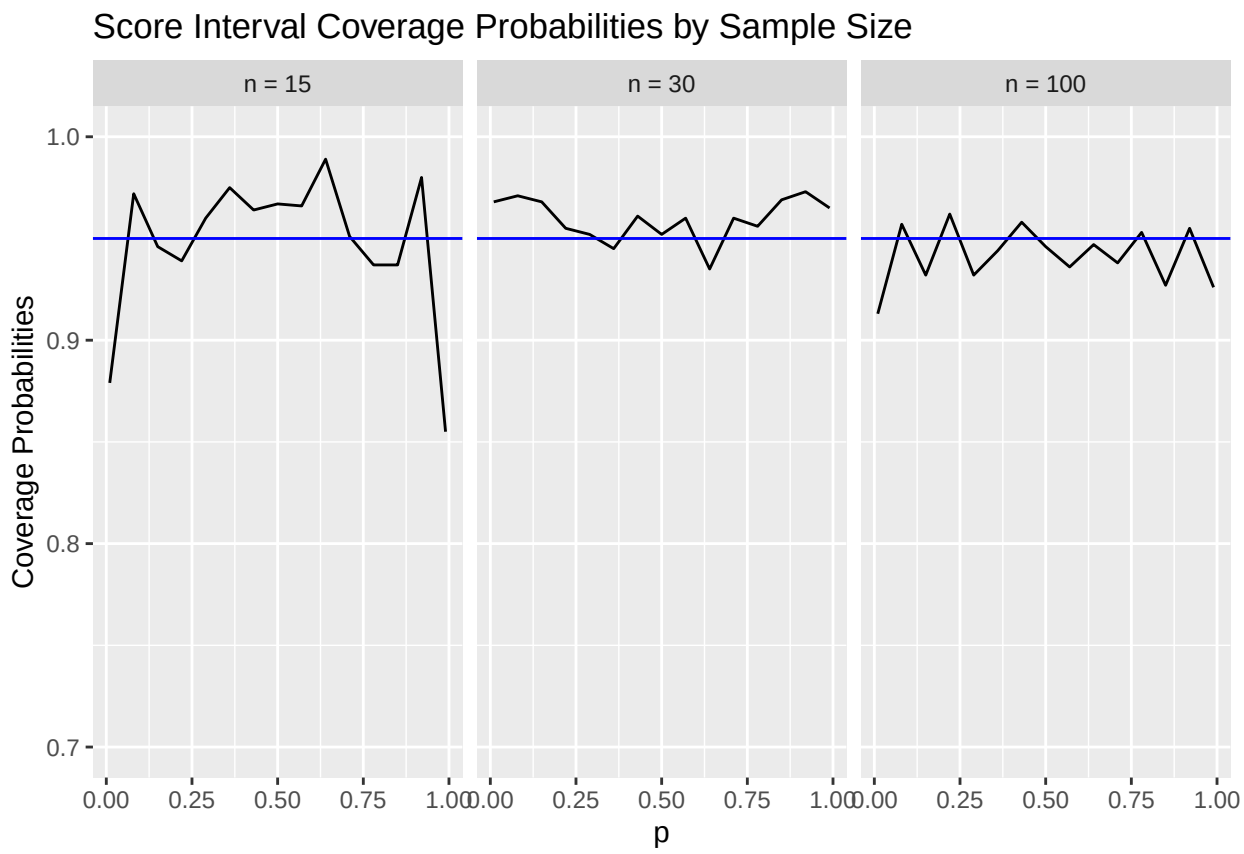


Score Interval

The curve dips below 0.95 when p gets closer to 0 or 1, but not as drastically as it does for the Wald interval. The score interval tends to avoid the severe under-coverage that the Wald interval suffers from in these regions. In addition, for smaller sample sizes, the plot may show more variability, but the Score interval still performs better than the Wald interval.

```
ggplot(score_summary, aes(x = p, y = score_coverage_probabilities)) +

  # Use a line graph to create a curve
  geom_line(color = "black") +

  # Facet wrap over sample size to compare plots for different sample sizes,
  # use labeller to show n = n_values in the strip labels
  facet_wrap(~n, labeller = labeller(n = function(x) paste("n =", x))) +

  # Add title for overall plot and y axis.
  labs(title = "Score Interval Coverage Probabilities by Sample Size",
```

```
      y = "Coverage Probabilities",
      x = "p") +
ylim(0.7,1) +

# Add a line at the ideal coverage probability (0.95) for clarity.
geom_hline(yintercept = 0.95, color = "blue")
```

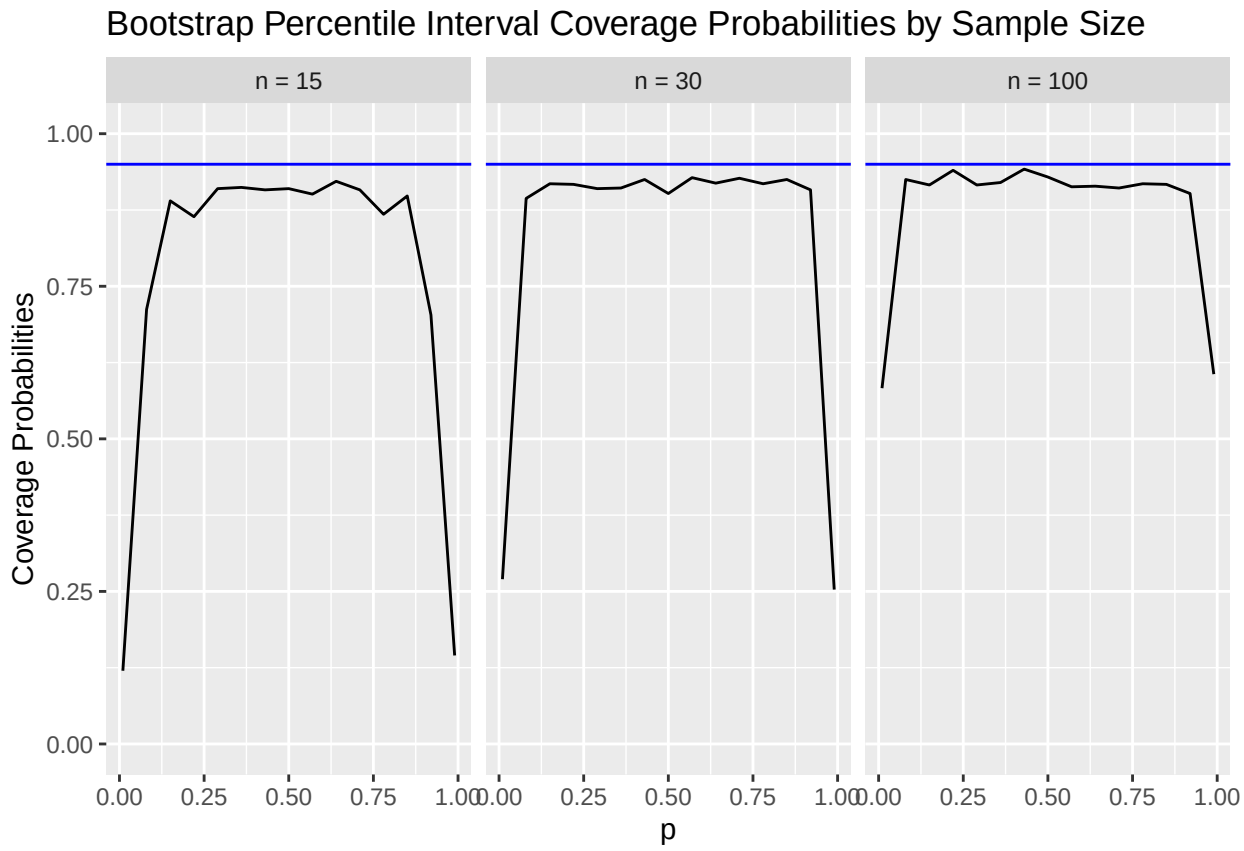


Raw Percentile Interval using a Parametric Bootstrap

The raw percentile interval using a parametric bootstrap misses the ideal convergence probability for every sample size and value of p. When n=100, it gets relatively close, but all of the convergence probabilities still lie under 95%. Over all values of n, the raw percentile interval method performs very poorly for extreme values of p towards 0 and 1.

```
#Bootstrap Percentile Coverage Probability Plot

ggplot(boot_per_summary, aes(x=p, y=boot_per_coverage_probabilities)) +

  #Use point graph to make easier to see differences.

  geom_line()+

  #Facet wrap over sample size to compare plots for different sample sizes.
```

```
  facet_wrap(~n, labeller = labeller(n = function(x) paste("n =", x))) +

  #Add title for overall plot and y axis.

  labs(title = "Bootstrap Percentile Interval Coverage Probabilities by Sample Size",

       y = "Coverage Probabilities") +

  #Add a line at the idea convergence probability for clarity.

  geom_hline(yintercept = 0.95, color = "blue") +ylim(0,1)
```

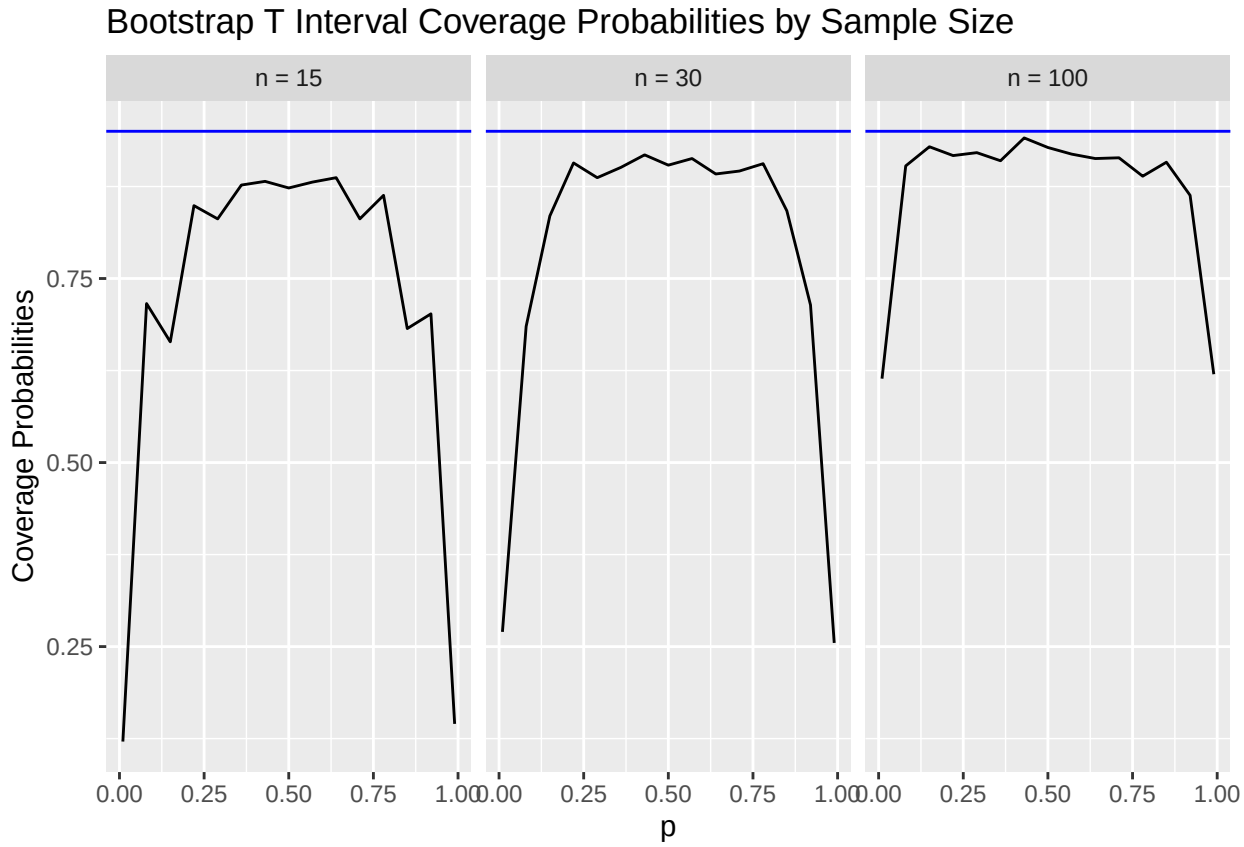


Bootstrap T Interval using a Parametric Bootstrap

From the plot below, we observe that the coverage probabilities are below the 95% reference line. The coverage probabilities improve as n increases and get very close to the ideal coverage probability for n=100. However, it is not performing very well for small sample sizes at all.

```
ggplot(boot_t_summary, aes(x = p, y = boot_t_coverage_probabilities)) +

  # Use a line graph to create a curve
  geom_line(color = "black") +

  # Facet wrap over sample size to compare plots for different sample sizes,
```

```
  # use labeller to show n = n_values in the strip labels
  facet_wrap(~n, labeller = labeller(n = function(x) paste("n =", x))) +

  # Add title for overall plot and y axis.
  labs(title = "Bootstrap T Interval Coverage Probabilities by Sample Size",
       y = "Coverage Probabilities",
       x = "p") +

  # Add a line at the ideal coverage probability (0.95) for clarity.
  geom_hline(yintercept = 0.95, color = "blue")
```

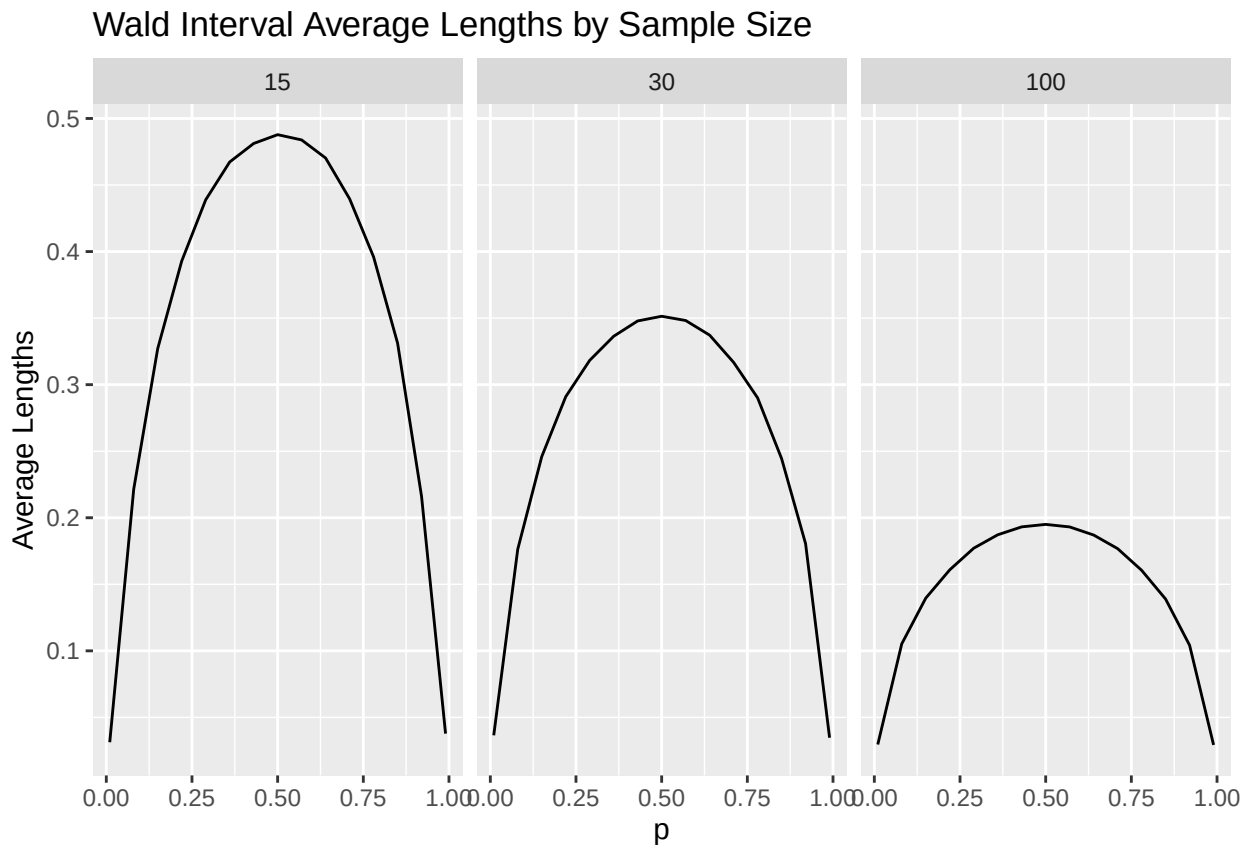


Wald Interval Average Lengths Plot

As the sample size increases for the wald interval, the average lengths of the intervals decreases. For n=15, the average length peaks when p is around 0.5 to an average length of about 0.5 For n=30, the average length peaks when p is around 0.5 to an average length of about 0.35. For n=100, the average length peaks when p is around 0.5 to an average length of about 0.2.

```
ggplot(wald_summary, aes(x=p, y=wald_lengths)) +
  #Use a line graph
  geom_line()+
  #Facet wrap over the sample sizes.
  facet_wrap(~n) +
  #Add a title and label to the y axis.
```

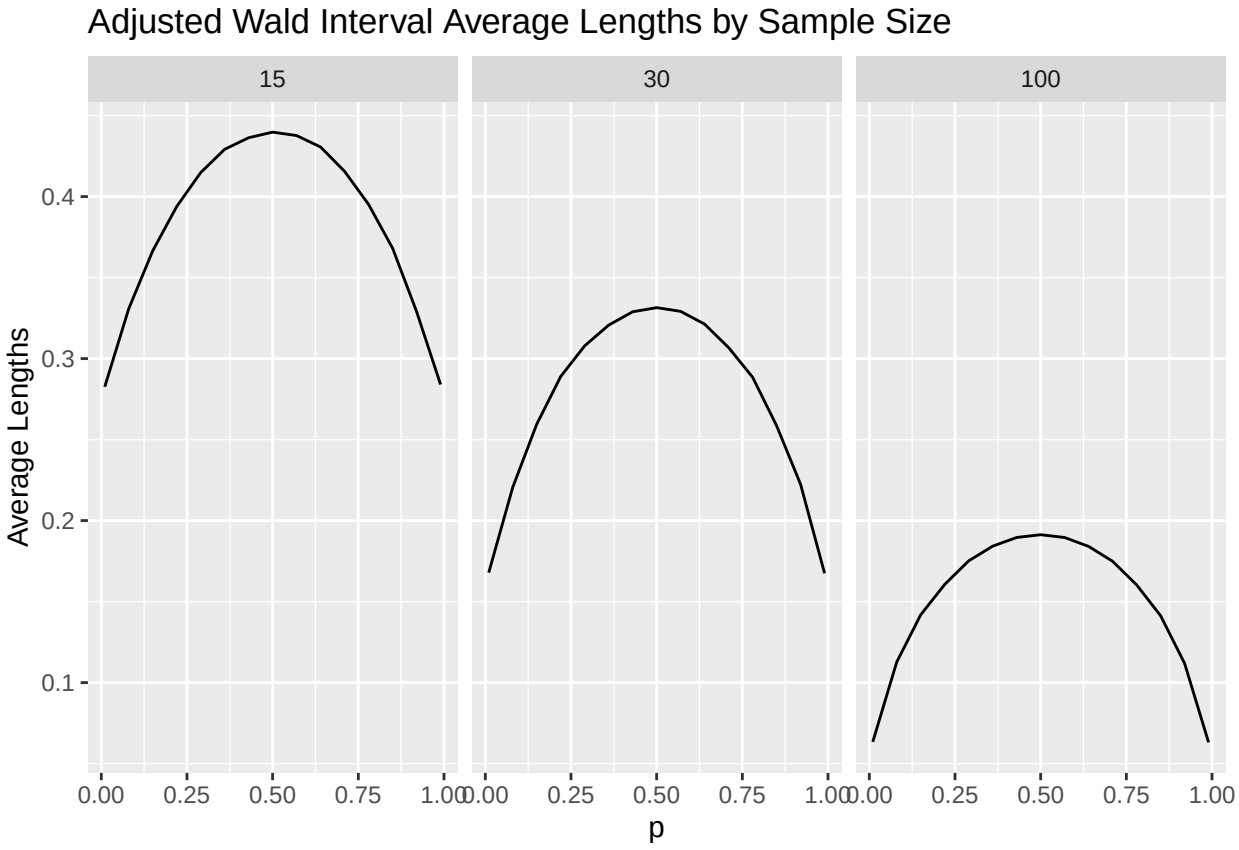
```
  labs(title = "Wald Interval Average Lengths by Sample Size",

       y = "Average Lengths")
```



Adjusted Wald Average Lengths Plots

The plots below indicating that as the sample size increases, the intervals become much tighter, and its length will decrease. The average length peaks when p is around 0.5 to an average length of about 0.44 (n=15), 0.33 (n=30), 0.19 (n=100). The minimum values for average lengths are 0.28 (n=15), 0.16(n=30), 0.03 (n=100).

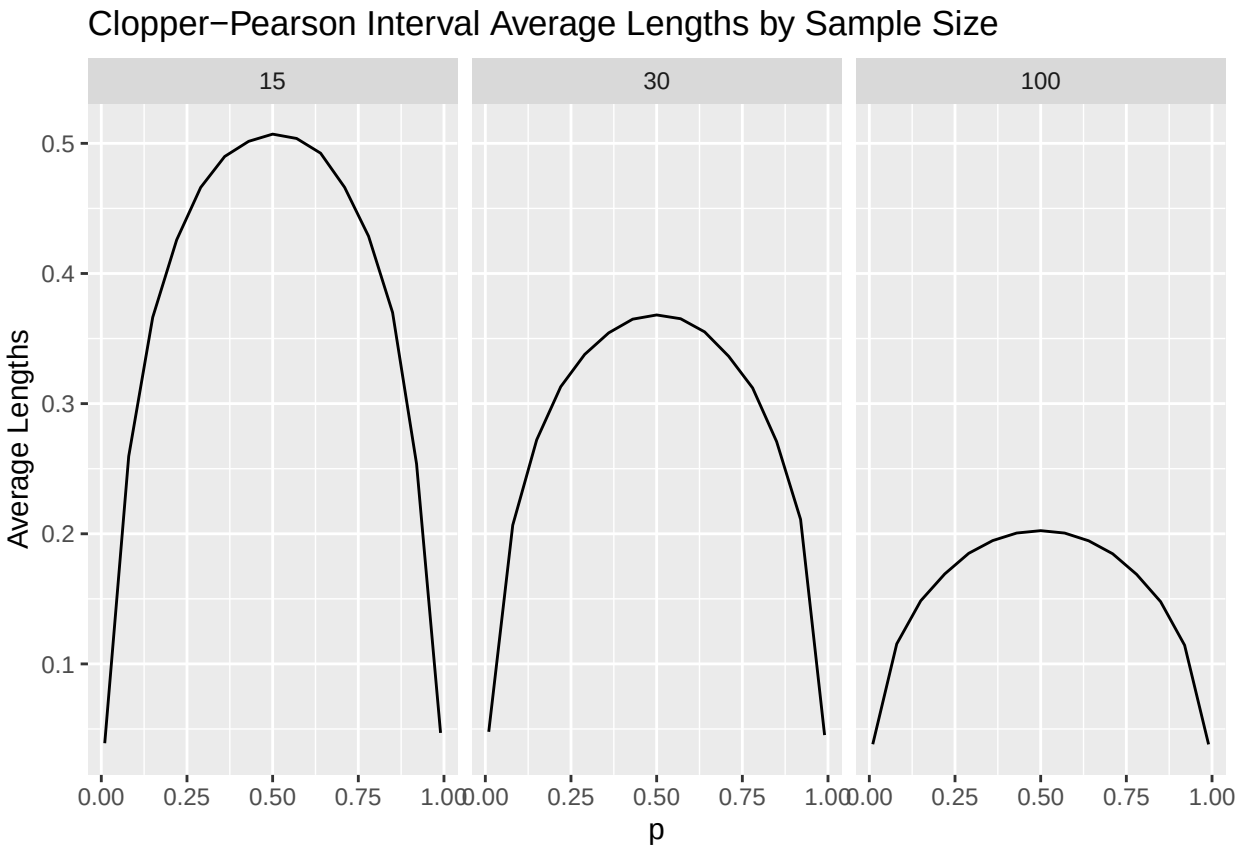
```
ggplot(adjusted_wald_summary, aes(x=p, y=adjusted_wald_lengths)) +
  #Use a line graph.
  geom_line()+

  #Facet wrap over the sample sizes.
  facet_wrap(~n) +

  #Add a title and label to the y axis.
  labs(title = "Adjusted Wald Interval Average Lengths by Sample Size",

       y = "Average Lengths")
```



Clopper-Pearson Average Lengths Plots

As the sample size increases for the Clopper-Pearson interval, the average lengths of the intervals decreases. For n=15, the average length peaks when p is around 0.5 to an average length of about 0.52 For n=30, the average length peaks when p is around 0.5 to an average length of about 0.36. For n=100, the average length peaks when p is around 0.5 to an average length of about 0.2.

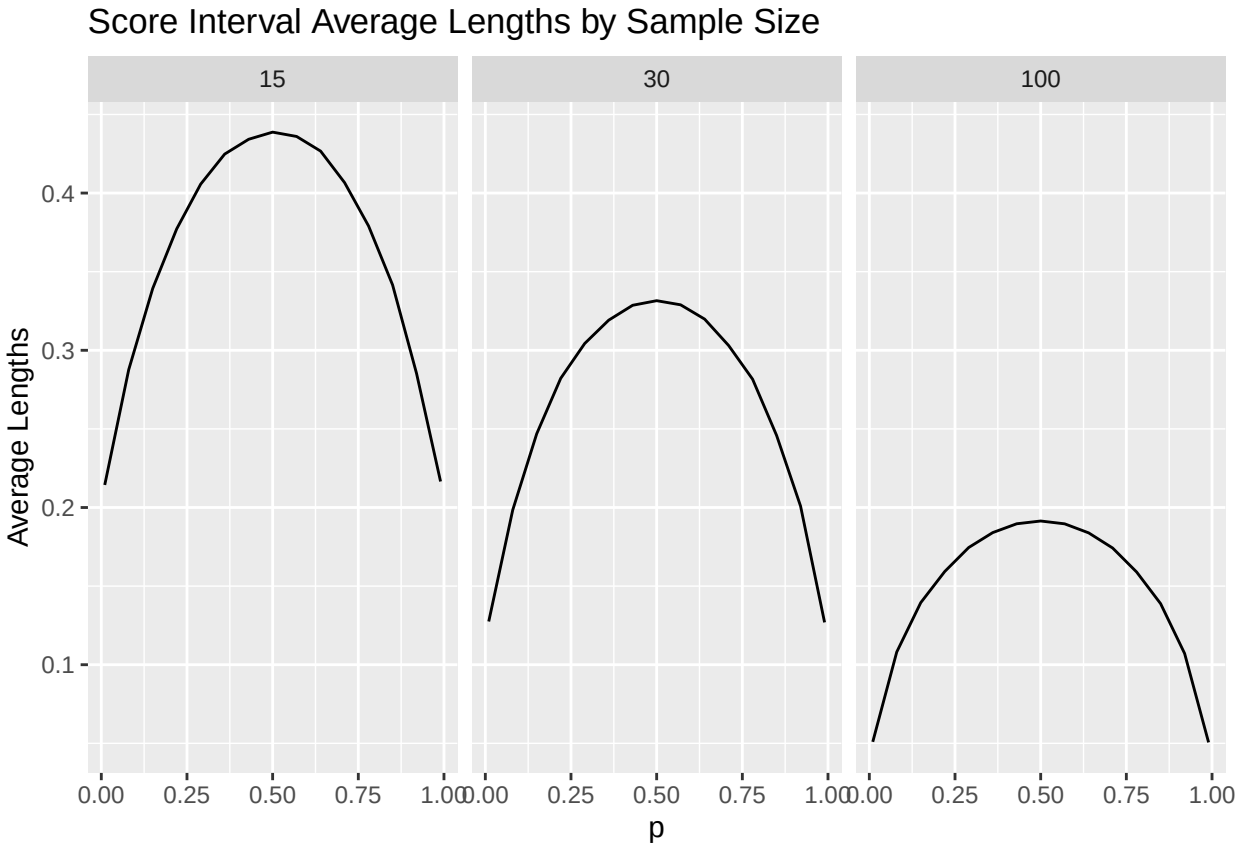
```
#Clopper-Pearson Average Lengths Plot

ggplot(cp_summary, aes(x=p, y=cp_lengths)) +
  #Use a line graph.
  geom_line()+

  #Facet wrap over the sample sizes.
  facet_wrap(~n) +

  #Add a title and label to the y axis.
  labs(title = "Clopper-Pearson Interval Average Lengths by Sample Size",

       y = "Average Lengths")
```



Score Interval Average Lengths Plots

As the sample size increases , the average lengths of the intervals decreases. The average length peaks when p is around 0.5 is about 0.44 (n=15) ,0.33 (n=30), 0.18 (n=100). The minimum values for average lengths are 0.21 (n=15), 0.13(n=30), 0.04 (n=100).

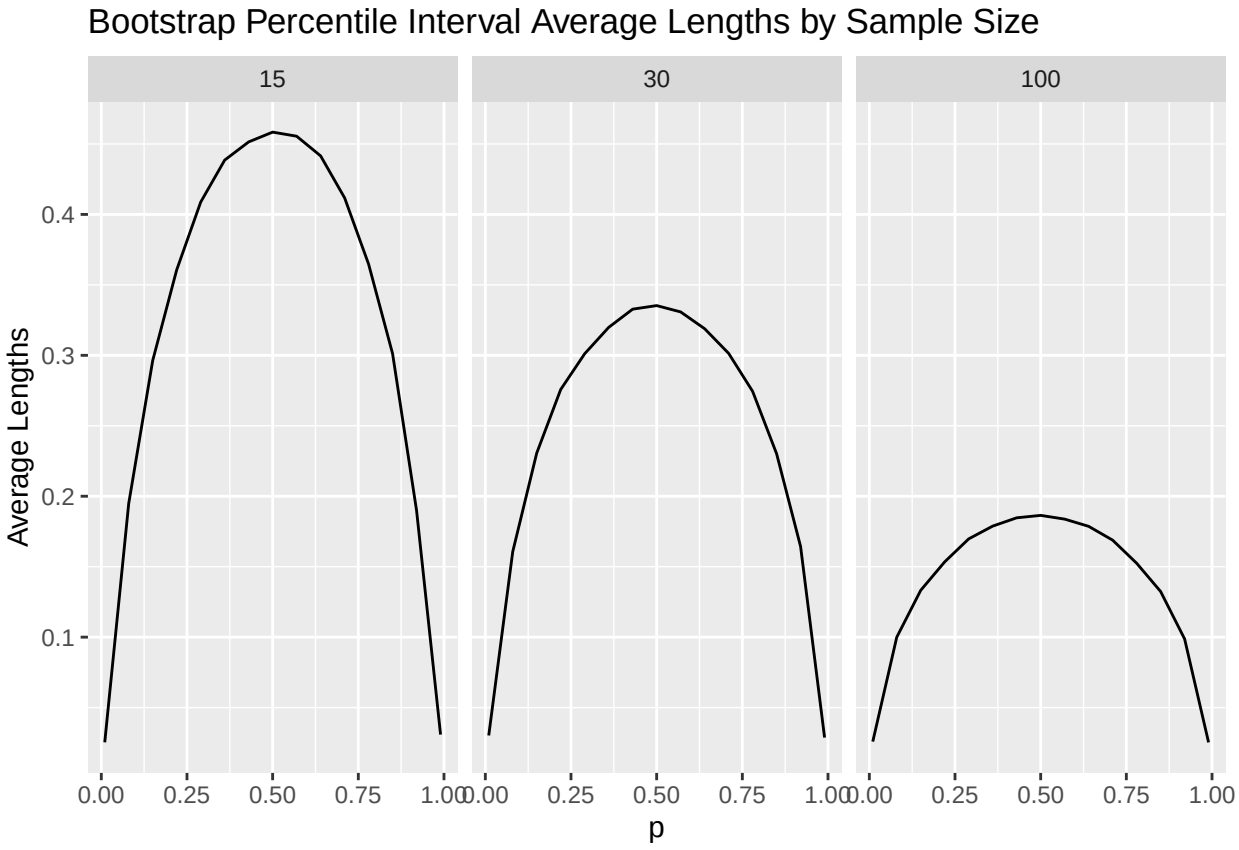
```
#Score Interval Average Lengths Plot

ggplot(score_summary, aes(x=p, y=score_lengths)) +
  #Use a line graph.
  geom_line(na.rm = TRUE)+

  #Facet wrap over the sample sizes.
  facet_wrap(~n) +

  #Add a title and label to the y axis.
  labs(title = "Score Interval Average Lengths by Sample Size",

       y = "Average Lengths")
```



Raw Percentile Bootstrap Interval Average Lengths Plots

As the sample size increases for the Raw Percentile Interval, the average lengths of the intervals decreases. For n=15, the average length peaks when p is around 0.5 to an average length of about 0.46. For n=30, the average length peaks when p is around 0.5 to an average length of about 0.33. For n=100, the average length peaks when p is around 0.5 to an average length of about 0.18.

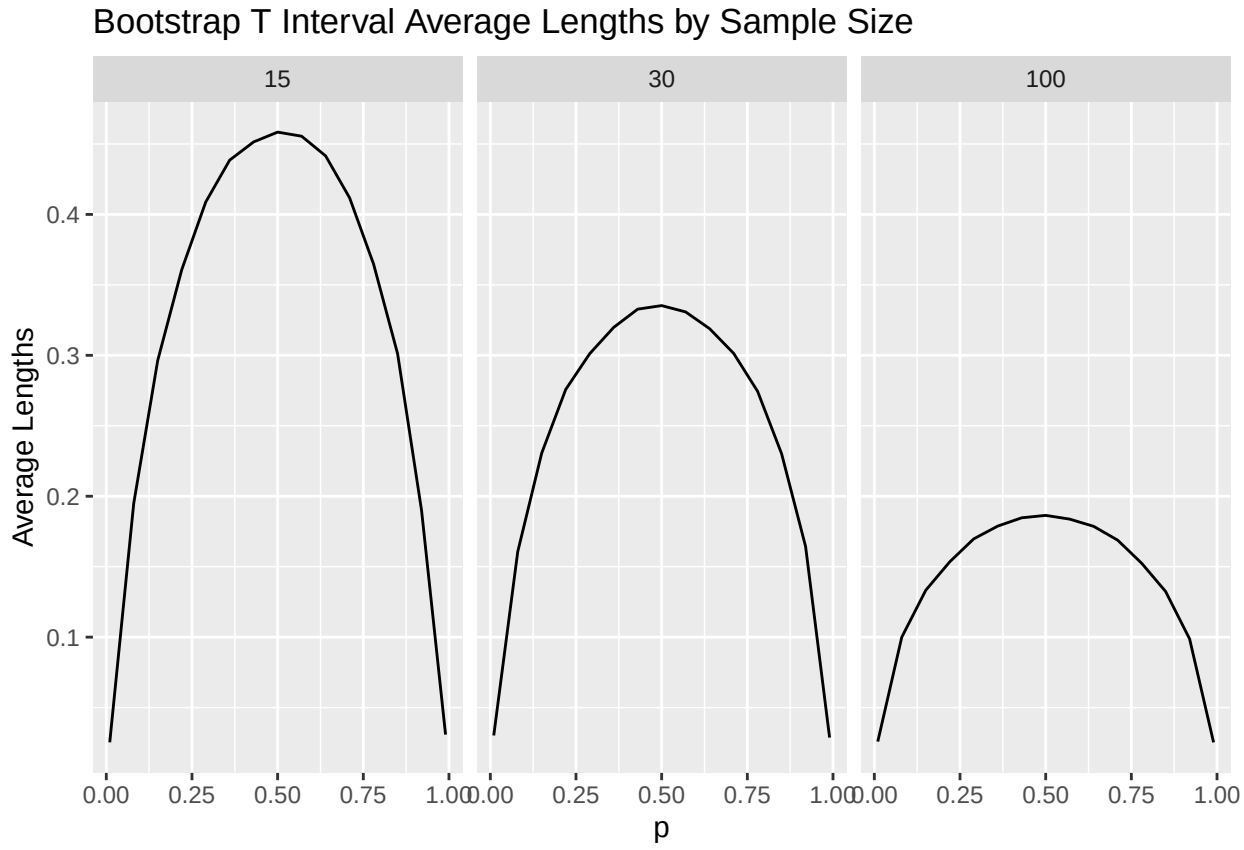
```
#Bootstrap Percentile Interval Average Lengths Plot

ggplot(boot_per_summary, aes(x=p, y=boot_per_lengths)) +
  #Use a line graph.
  geom_line(na.rm = TRUE)+

  #Facet wrap over the sample sizes.
  facet_wrap(~n) +

  #Add a title and label to the y axis.
  labs(title = "Bootstrap Percentile Interval Average Lengths by Sample Size",

       y = "Average Lengths")
```



Bootstrap T interval Average Lengths

The following plots also show that as the sample size increases, the average interval length decreases. The average length peaks when p is around 0.5 is about 0.46 (n=15) ,0.33 (n=30), 0.18 (n=100). The minimum values for average lengths are around 0.44 (n=15), 0.28(n=30), 0.13 (n=100).

```
#Bootstrap T Interval Average Lengths Plot

ggplot(boot_t_summary, aes(x=p, y=boot_t_lengths)) +
  #Use a line graph.
  geom_line(na.rm = TRUE)+

  #Facet wrap over the sample sizes.
  facet_wrap(~n) +

  #Add a title and label to the y axis.
  labs(title = "Bootstrap T Interval Average Lengths by Sample Size",

       y = "Average Lengths")
```



Standard Errors of Average lengths

Below are the standard errors of the average lengths computed in the previous section. The Score Intervals and Adjusted Wald Intervals display the least standard error, whereas the Clopper-Pearson shows the greatest standard error of the average lengths.

```
sd_wald <-sd(wald_lengths)
paste("The SE of the average lengths for the Wald Interval is ", sd_wald)
```

```
## [1] "The SE of the average lengths for the Wald Interval is  0.138859041403988"
```

```
sd_adjusted_wald<- sd(adjusted_wald_lengths)
paste("The SE of the average lengths for the Adjusted Wald Interval is ", sd_adjusted_wald)
```

```
## [1] "The SE of the average lengths for the Adjusted Wald Interval is  0.109211550387495"
```

```
sd_cp <- sd(cp_lengths)
paste("The SE of the average lengths for the Clopper Pearson Interval is ", sd_cp)
```

```
## [1] "The SE of the average lengths for the Clopper Pearson Interval is  0.144525552128912"
```

```r
sd_score <- sd(score_lengths)
paste("The SE of the average lengths for the Score Interval is ", sd_score)
```

```
## [1] "The SE of the average lengths for the Score Interval is  0.109803547753001"
```

```r
sd_boot_per <-sd(boot_per_lengths, na.rm = TRUE)
paste("The SE of the average lengths for the Bootstrap Percentile Interval is ", sd_boot_per)
```

```
## [1] "The SE of the average lengths for the Bootstrap Percentile Interval is  0.130760101864441"
```

```r
sd_boot_t <-sd(boot_t_lengths, na.rm=TRUE)
paste("The SE of the average lengths for the Bootstrap T Interval is ", sd_boot_t)
```

```
## [1] "The SE of the average lengths for the Bootstrap T Interval is  0.130760101864441"
```

Results

Coverage Probability

We can see from the Coverage Proability Plots that as the sample size grows, the coverage probabilities of all the intervals gets closer to the ideal coverage probability of $1-\alpha$. The coverage probabilities also appear to be closer to the $1-\alpha$ when p is close to 0.5, rather than extreme values of p towards 0 and 1. The Wald Interval coverage probabilities display a lot of variability, with most of the combinations of n and p falling below the 95% threshold. On the other hand, the Clopper Pearson Interval appears to be above the 95% mark for most combinations, but this comes at the expense of larger intervals which we will discuss in the following paragraph. The Average Wald Interval and the Score Intervals do a fairly good job, although the Adjusted Wald is not doing great for a small sample size of 15. The Bootstrap Percentile Interval and the Bootstrap T Interval are consistently below the ideal convergence probability, with the Bootstrap T Interval showing a significant discrepancy for a small sample size.

Miss Low and Miss High

In addition, the proportions that miss above and miss below, tell us how often the confidence intervals are too low or too high, respectively, relative to the true value. From the wald_summary, the intervals severely miss low, with almost no cases of missing high. As p increases, the miss low and miss high values become more balanced, especially with larger sample sizes. At high p, the intervals often miss high, reflecting overestimation.

Adjusted_wald_summary: These intervals rarely miss low with a slight overestimation (miss high is small) when p is very small (0.01). There are balanced miss low and miss high rates, for mid-range p indicating well-centered intervals. There is a tendency to underestimate (miss low increases), with almost no overestimation (miss high is close to 0) for high p. The Adjusted Wald method performs well overall but shows a slight conservative bias at high proportions.

From cp_summary, low p tend to be underestimated (high rate of miss low). Mid-range p show a better balance of miss low and miss high. High values of p tend to be overestimated (high miss high), especially for small sample sizes.

From the score_summary, for small values of p, especially around 0.01, intervals rarely miss low (0.000) but often miss high (e.g., 0.121 for n=15). This shows a tendency to overestimate at the lower boundary of the

proportion scale. For larger values of p, like 0.99, the intervals tend to miss low (e.g., 0.145 for n=15) and rarely miss high, indicating underestimation for values near 1.

From boot_per_summary, low proportions are often underestimated (high miss low), while high proportions tend to be overestimated (high miss high). Mid-range proportions (p=0.50) show relatively balanced miss low and miss high rates, indicating good centering of the intervals.For high proportions, bootstrap percentile method shows a tendency to miss high, meaning it often overestimates the true value.

From boot_t_summary, miss low is very significant, especially for smaller sample sizes. For instance, at p=0.01 and n=15, miss low is 87.9%, indicating a strong tendency to underestimate the true proportion. The miss low and miss high are more balanced in this range, but there's still a tendency to miss low at smaller sample sizes. At high values of p, the intervals often miss high, reflecting overestimation of the true proportion.

Average Lengths

We can see from all of the average lengths plots that as the sample size grows, the average lengths of the intervals increases. This means that as our sample size increases, we are able to get a more accurate picture (a narrower interval) of where the true value of p is. For all intervals, we can also see that there is a peak in the average length around p=0.5. This makes sense because p=0.5 maximizes the standard error. A probability of 0.5 means that it is essentially a toss-up - there is an equal probability of observing a success or a failure. When comparing p=0.5 for varying sample sizes:

- At n = 15, the Score Interval has the smallest average length and the Clopper-Pearson has the largest average length.
- At n = 30, the Adjusted Wald Interval has the smallest average length (closely followed by the Score Interval) and the Clopper-Pearson has the largest average length.
- At n = 100, the Adjusted Wald Interval has the smallest average length (closely followed by the Score Interval) and the Clopper-Pearson has the largest average length.

The standard errors for these average lengths show similar patterns with the Adjusted Wald and the Score Intervals showing the least amount of variability and the Clopper-Pearson showing the greatest standard error.

Conclusions

Based on the coverage probabilities and the average lengths, we would prefer either the Adjusted Wald Interval or the Score Interval to estimate a true population proportion. Although the Wald is the method typically introduced in many Introductory Statistics courses, it shows much variability and relies on asymptotic normality, making it unfit for smaller sample sizes. On the other hand, the Clopper-Pearson does better for smaller sample sizes but actually overshoots the coverage probability with larger average lengths which isn't ideal either. Both of the bootstrap intervals come at a higher computational cost and also do not do an great job in terms of coverage probabilities. The Adjusted Wald Intervals and Score Intervals have the smallest average lengths and standard errors, but are still capturing the true population with a coverage probability close to $1 - \alpha$. The Adjusted Wald appears to do better than the Score in terms of coverage probabilities for more extreme values of p since the score method at the extremes shows "the actual coverage probability falls seriously below the nominal confidence level, and this badly affects the actual confidence coefficient" (Agresti & Coull, 1998, p. 122). This actually doesn't improve as much n increases for the extreme values of p, but still performs reasonably well for most values of p and is still to be preferred compared to four of the other methods presented (Clopper-Pearson, Wald, and Bootstrap Intervals). The Adjusted Wald Interval does not have the disadvantage of large dips at extreme values of p, but "the proportion of the parameter space for which the actual coverage probability falls within 0.02 of 0.95" (Agresti & Coull, 1998, p. 124) is slightly less than the Score Interval.

Based on all of the findings and results above, we would recommend using either the Adjusted Wald or Score Intervals for calculating a confidence interval for a population proportion. If there is reason to believe based on background knowledge of the topic that the true probability might be close to 0 or 1 we should opt for the Adjusted Wald but, for more moderate values of p, both the Adjusted Wald and Score Intervals give fairly narrow intervals that are accurate across various sample sizes.