ST 502 Project 1

Taylor Cesarski and Jie Chen

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.

Methods Used & Calculations

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:

```
#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))
}
```

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) (INSERT CITATION). 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:

```
#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)
  }
}
```

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.

```
#Create a bootstrap interval function.
#Let the default for alpha be 0.05.
bootstrap_int <- function(y, n, alpha =0.05){
  #Calculate the sample proportion.
  phat <- y/n

  #Get 100 resamples of size n and compute the sample proportion for each.
  boot_estimates <- replicate(100,{
    boot_sample <- rbinom(1, size = n, prob = phat)
    boot_phat <- boot_sample/n
  })
  #Return the 2.5th quantile and 97.5th quantile of bootstrap distribution.
return(c(quantile(boot_estimates, 0.025), quantile(boot_estimates, 0.975)))
}
```

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)
#Create a vector of sample sizes.
```

```r
sample_sizes <- c(15, 30, 100)
#Create a vector of 15 values of p ranging from 0.01 to 0.99.
p <- c(0.01, 0.05, 0.10, 0.20, 0.25, 0.35, 0.45, 0.5, 0.6,
       0.65, 0.75, 0.8, 0.9, 0.95, 0.99)

#Initalize lists to store each data frame of CIs.
wald_confidence_int <- list()
cp_confidence_int <- list()
boot_percentile_confidence_int <- 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(1000), upper=numeric(1000))
   cp_CIs <- data.frame(lower=numeric(1000), upper=numeric(1000))
   boot_CIs <- data.frame(lower=numeric(1000), upper=numeric(1000))

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

  #Apply the wald_int function and store the lower & upper endpoint.
  wald_interval <- wald_int(y, size)

  wald_CIs$lower[i] <- wald_interval[1]
  wald_CIs$upper[i] <- 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 the bootstrap_int function and store the lower & upper endpoint.
  boot_interval <- bootstrap_int(y, size)

  boot_CIs$lower[i] <- boot_interval[1]
```

```
  boot_CIs$upper[i] <- boot_interval[2]

}
  #Store each data frame in the list with names.
  wald_confidence_int[[paste0("Wald", "size_", size, "prob_", prob)]] <- wald_CIs
  cp_confidence_int[[paste0("Clopper-Pearson", "size_", size, "prob_", prob)]] <- cp_CIs
  boot_percentile_confidence_int[[paste0("Bootstrap Percentile", "size_", size,
                                         "prob_", prob)]] <- boot_CIs
  }
}
```

Code to Execute Monte Carlo Study (maybe join with above)

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)
cp_lengths <- numeric(45)
boot_lengths <- 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)
  cp_lengths[i] <- mean(cp_confidence_int[[i]]$upper - cp_confidence_int[[i]]$lower)
  boot_lengths[i] <- mean(boot_percentile_confidence_int[[i]]$upper - boot_percentile_conf
}

#Initialize numeric vectors of size 45 to hold standard errors.
wald_se <- numeric(45)
cp_se <- numeric(45)
boot_se <- numeric(45)

#Use a for loop to calculate the standard errors of average lengths
#for each set of 1000 intervals.
for(i in 1:45){
  wald_se[i] <- sd((wald_confidence_int[[i]]$upper -
```

```
                      wald_confidence_int[[i]]$lower))
  cp_se[i] <- sd((cp_confidence_int[[i]]$upper -
                    cp_confidence_int[[i]]$lower))
  boot_se[i] <- sd((boot_percentile_confidence_int[[i]]$upper -
                      boot_percentile_confidence_int[[i]]$lower))
}

#Create a vector that contains the probabilities 3 times.
repeated_p <- rep(p, 3)

#Initialize a numeric vector of length 45 to hold the coverage probabilities.
wald_coverage_probabilities <- numeric(45)
cp_coverage_probabilities <- numeric(45)
boot_coverage_probabilities <- numeric(45)


#Use a for loop to calculate the coverage probability (% containing true p) for each set o
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])

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

  boot_coverage_probabilities[i] <-
    mean(boot_percentile_confidence_int[[i]]$upper > repeated_p[i] &
           boot_percentile_confidence_int[[i]]$lower < repeated_p[i])
}



#Initialize numeric vectors to store the percentage of intervals missing low.
wald_miss_low <- numeric(45)
cp_miss_low <- numeric(45)
boot_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])
```

```r
  cp_miss_low[i] <- mean(cp_confidence_int[[i]]$upper < repeated_p[i])
  boot_miss_low[i] <- mean(boot_percentile_confidence_int[[i]]$upper < repeated_p[i])
}

#Initalize numeric vectors to store the percentage of intervals missing high.
wald_miss_high <- numeric(45)
cp_miss_high <- numeric(45)
boot_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]]$lower > repeated_p[i])
  cp_miss_high[i] <- mean(cp_confidence_int[[i]]$lower > repeated_p[i])
  boot_miss_high[i] <- mean(boot_percentile_confidence_int[[i]]$lower > repeated_p[i])
}


#Create a data frame to hold information about the Wald Interval.

#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,
                           st_errors = wald_se,
                           coverage_probs = wald_coverage_probabilities,
                           miss_low = wald_miss_low,
                           miss_high = wald_miss_high)

head(wald_summary)
```

```
  n    p avg_int_lengths  st_errors coverage_probs miss_low miss_high
1 15 0.01      0.03134204 0.08497063          0.121    0.879     0.000
2 15 0.05      0.15467328 0.14899811          0.536    0.463     0.001
3 15 0.10      0.26141487 0.13995745          0.810    0.188     0.002
4 15 0.20      0.37882408 0.10102675          0.827    0.160     0.013
5 15 0.25      0.41529211 0.08324903          0.894    0.091     0.015
6 15 0.35      0.46642987 0.04854708          0.897    0.061     0.042
```

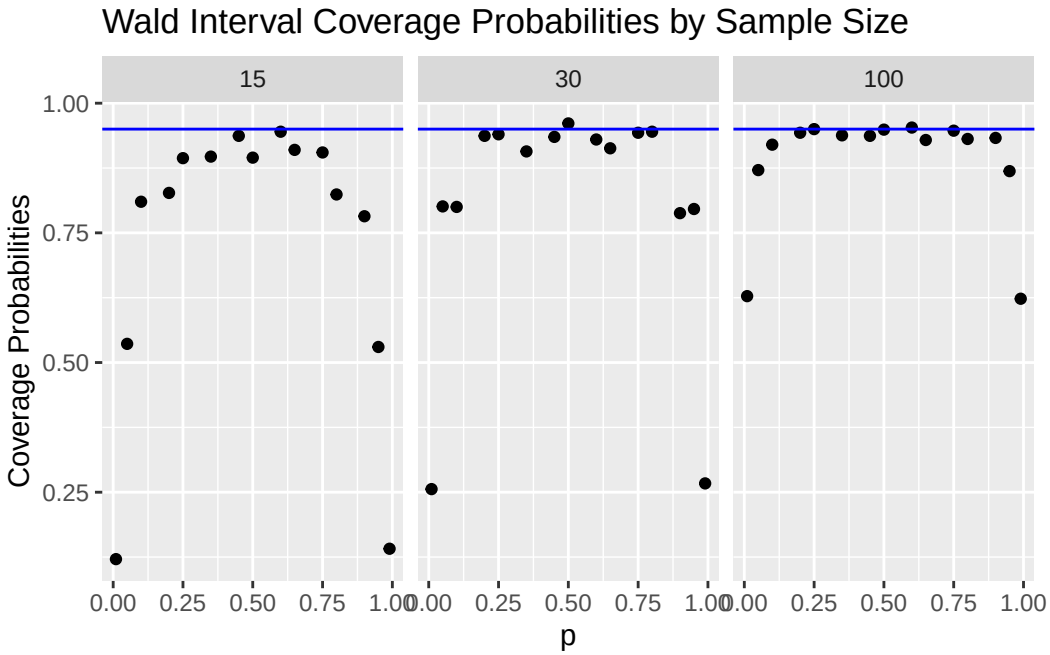
```
cp_summary <- data.frame(n = n_values,
                         p = repeated_p,
                         avg_int_lengths = cp_lengths,
                         st_errors = cp_se,
                         coverage_probs = cp_coverage_probabilities,
                         miss_low = cp_miss_low,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

boot_summary <- data.frame(n = n_values,
                           p = repeated_p,
                           avg_int_lengths = boot_lengths,
                           st_errors = boot_se,
                           coverage_probs = boot_coverage_probabilities,
                           miss_low = boot_miss_low,
                           miss_high = boot_miss_high)
```

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.

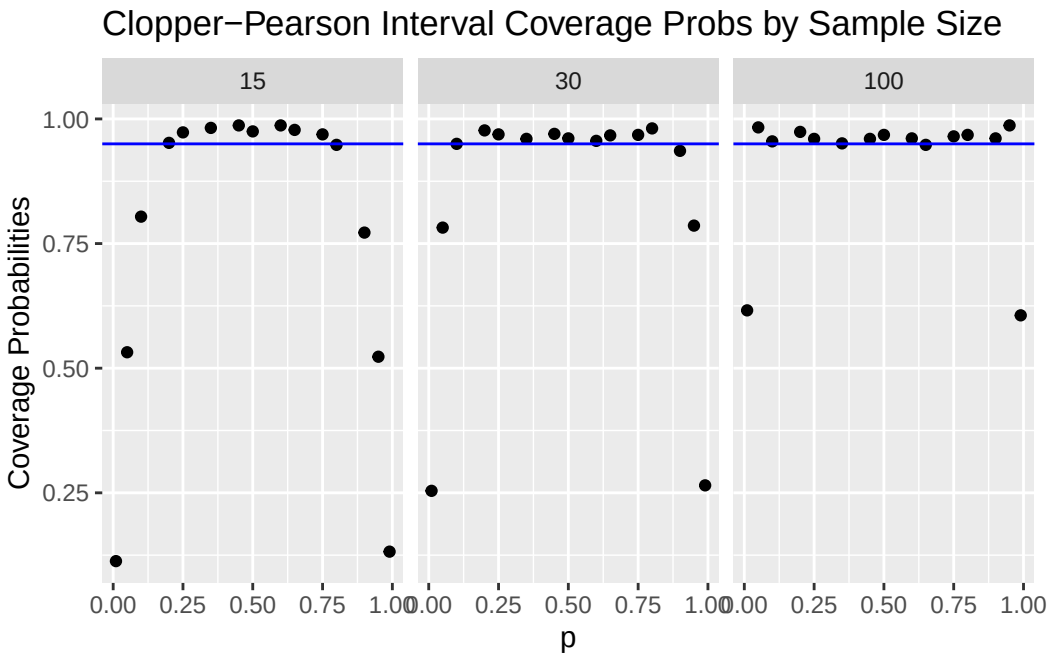
```
ggplot(wald_summary, aes(x=p, y=wald_coverage_probabilities)) +
  #Use point graph to make easier to see differences.
  geom_point()+
  #Facet wrap over sample size to compare plots for different sample sizes.
  facet_wrap(~n) +
  #Add title for overall plot and y axis.
  labs(title = "Wald Interval 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")
```



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.

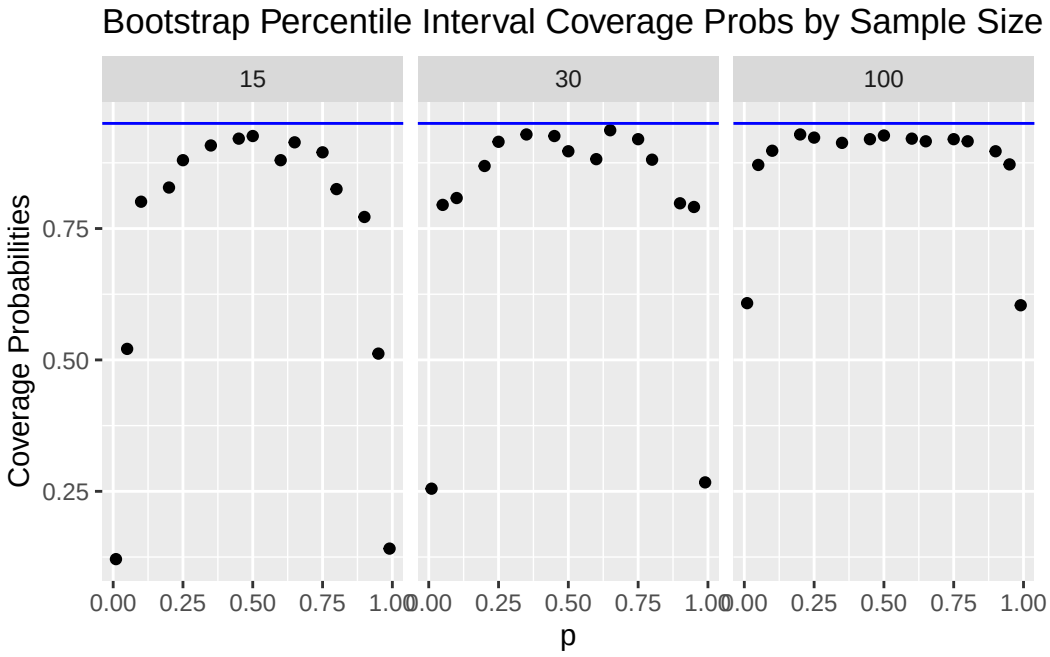
```
ggplot(cp_summary, aes(x=p, y=cp_coverage_probabilities)) +
  #Use point graph to make easier to see differences.
  geom_point()+
  #Facet wrap over sample size to compare plots for different sample sizes.
  facet_wrap(~n) +
  #Add title for overall plot and y axis.
  labs(title = "Clopper-Pearson Interval Coverage Probs by Sample Size",
       y = "Coverage Probabilities") +
  #Add a line at the idea convergence probability 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.

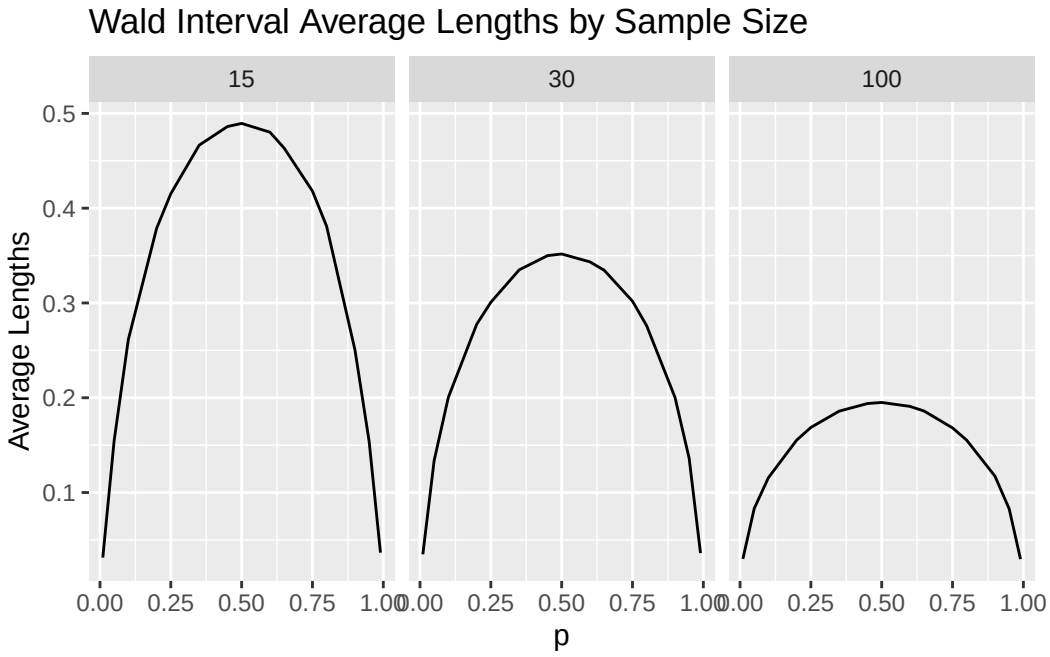
```
#Bootstrap Percentile Coverage Probability Plot
ggplot(boot_summary, aes(x=p, y=boot_coverage_probabilities)) +
  #Use point graph to make easier to see differences.
  geom_point()+
  #Facet wrap over sample size to compare plots for different sample sizes.
  facet_wrap(~n) +
  #Add title for overall plot and y axis.
  labs(title = "Bootstrap Percentile Interval Coverage Probs by Sample Size",
       y = "Coverage Probabilities") +
  #Add a line at the idea convergence probability 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.

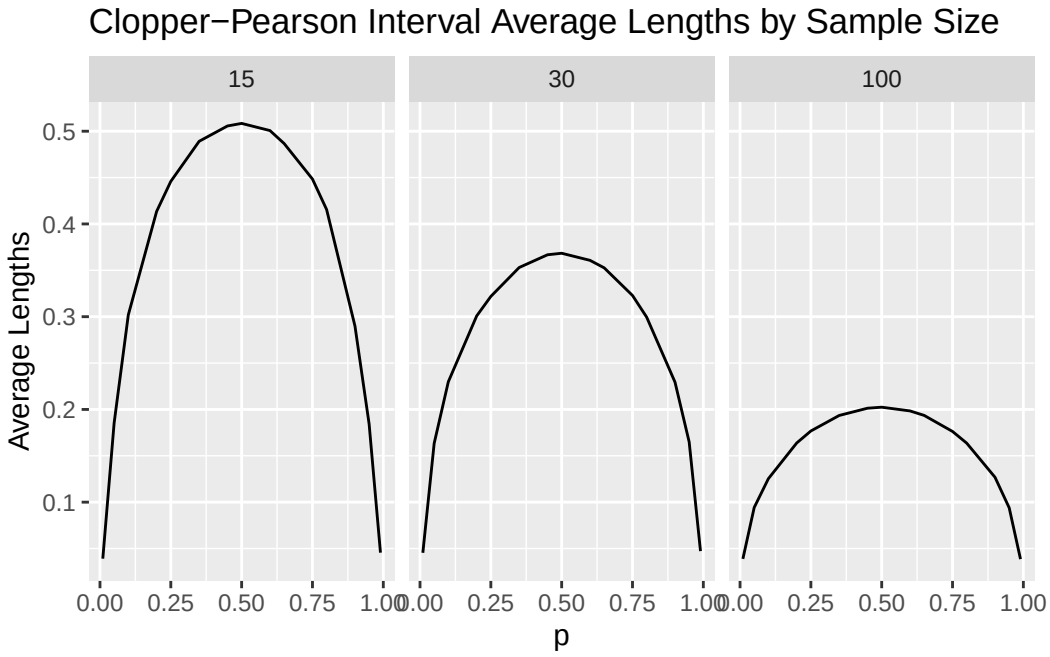
```
ggplot(wald_summary, aes(x=p, y=wald_lengths)) +
  geom_line()+
  facet_wrap(~n) +
  labs(title = "Wald Interval Average Lengths by Sample Size",
       y = "Average Lengths")
```



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 0.5 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.

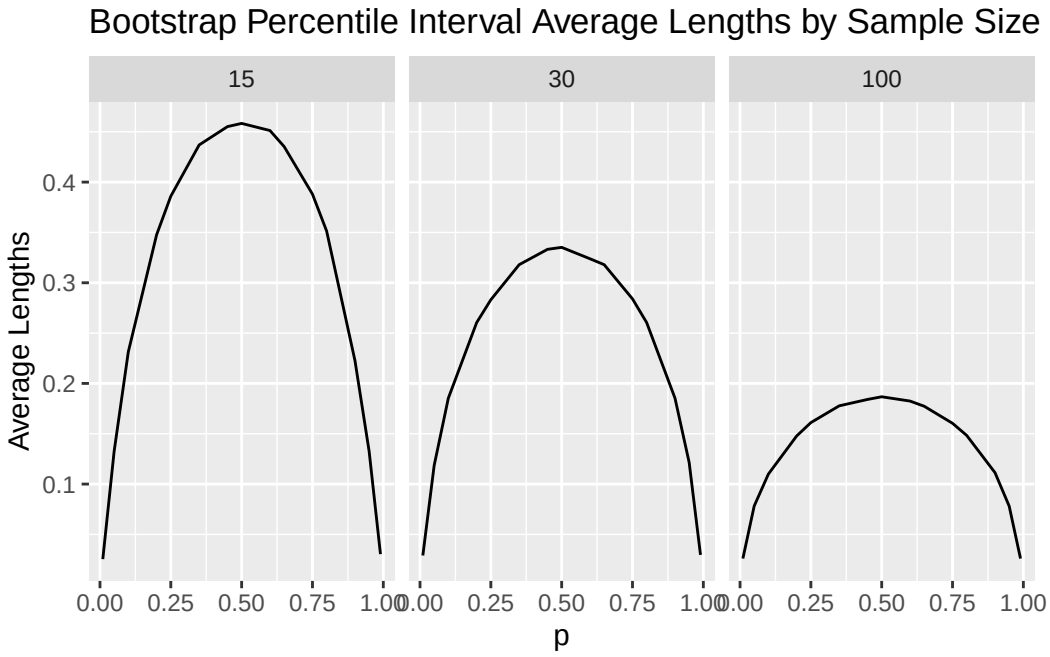
```
#Clopper-Pearson Average Lengths Plot
ggplot(cp_summary, aes(x=p, y=cp_lengths)) +
  geom_line()+
  facet_wrap(~n) +
  labs(title = "Clopper-Pearson Interval Average Lengths by Sample Size",
       y = "Average Lengths")
```



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_summary, aes(x=p, y=boot_lengths)) +
  geom_line()+
  facet_wrap(~n) +
  labs(title = "Bootstrap Percentile Interval Average Lengths by Sample Size",
       y = "Average Lengths")
```


Bootstrap Percentile Interval Average Lengths by Sample Size
15
30
100
Average Lengths
0.4
0.3
0.2
0.1
0.00
0.25
0.50
0.75
1.00
p