ST 502 Project 1

Taylor Cesarski and Jie Chen

Goals of Simulation Study

Methods Used & Calculations

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).

The Wald Interval can be found using the following function:

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

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:

```
CP_int <- function(y, n, alpha = 0.05){
  c((1 + (n-y+1)/(y*qf(1-alpha/2,2*y, 2*(n-y+1))))^-1,
    (1 + (n-y+1)/((y+1)*qf(alpha/2, 2*(y+1), 2*(n-y))))^-1)
```

```
  help(qf)

}
```

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 with replacement to get many samples of the same size. Then find the bootstrap estimate, and

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 considering the Wald Interval.

```
sample_sizes <- c(15, 30, 100)
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, 0.9

wald_confidence_int <- list()


for(size in sample_sizes){
  for(prob in p){
   wald_CIs <- data.frame(lower=numeric(1000), upper=numeric(1000))
    data <- rbinom(n=1000, size, prob)
for(i in 1:1000){
  y<-data[i]

  wald_interval <- wald_int(y, size)

  wald_CIs$lower[i] <- wald_interval[1]
  wald_CIs$upper[i] <- wald_interval[2]
}
```

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

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

We will create a data frame for the Wald Interval that contains the sample size (n), the probability of success (p), the average length of the intervals, the coverage probability, the proportion of intervals that miss above, and the proportion of intervals that miss below.

```
#Initialize a "lengths" numeric vector to hold average lengths of size 45.
lengths <- numeric(45)

#Use a for loop to calculate the average lengths for each set of 1000 intervals.
for(i in 1:45){
  lengths[i] <- mean(wald_confidence_int[[i]]$upper - wald_confidence_int[[i]]$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.
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) {
  coverage_probabilities[i] <- mean(wald_confidence_int[[i]]$upper > repeated_p[i] &
                    wald_confidence_int[[i]]$lower < repeated_p[i])
}

#Find the percentage where the true p is above the interval.
miss_low <- numeric(45)

for (i in 1:45) {
  miss_low[i] <- mean(wald_confidence_int[[i]]$upper < repeated_p[i])
}

miss_low
```

```
 [1] 0.863 0.477 0.212 0.192 0.086 0.059 0.051 0.076 0.029 0.045 0.022 0.018
[13] 0.002 0.001 0.000 0.733 0.190 0.172 0.045 0.047 0.062 0.032 0.025 0.022
```

```
[25] 0.031 0.024 0.011 0.007 0.004 0.000 0.354 0.119 0.068 0.054 0.042 0.026
[37] 0.026 0.032 0.024 0.023 0.017 0.013 0.008 0.004 0.000
```

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

miss_high <- numeric(45)

for (i in 1:45) {
  miss_high[i] <- mean(wald_confidence_int[[i]]$lower > repeated_p[i])
}

miss_high
```

```
 [1] 0.000 0.001 0.001 0.019 0.023 0.047 0.022 0.048 0.029 0.071 0.070 0.159
[13] 0.186 0.474 0.834 0.000 0.006 0.006 0.016 0.021 0.032 0.043 0.026 0.039
[25] 0.055 0.034 0.036 0.176 0.232 0.742 0.000 0.001 0.004 0.017 0.013 0.025
[37] 0.029 0.026 0.030 0.032 0.034 0.044 0.057 0.115 0.395
```

```r
#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 = lengths, covera
wald_summary
```

```
    n    p avg_int_lengths coverage_probs miss_low miss_high
1   15 0.01      0.03596184          0.137    0.863     0.000
2   15 0.05      0.14955947          0.522    0.477     0.001
3   15 0.10      0.25295200          0.787    0.212     0.001
4   15 0.20      0.37183827          0.789    0.192     0.019
5   15 0.25      0.41167417          0.891    0.086     0.023
6   15 0.35      0.46296126          0.894    0.059     0.047
7   15 0.45      0.48491243          0.927    0.051     0.022
8   15 0.50      0.48786690          0.876    0.076     0.048
9   15 0.60      0.47786696          0.942    0.029     0.029
10  15 0.65      0.46266876          0.884    0.045     0.071
```

```
11  15 0.75      0.41540378          0.908    0.022     0.070
12  15 0.80      0.37734912          0.823    0.018     0.159
13  15 0.90      0.26032118          0.812    0.002     0.186
14  15 0.95      0.15121763          0.525    0.001     0.474
15  15 0.99      0.04288625          0.166    0.000     0.834
16  30 0.01      0.03612520          0.267    0.733     0.000
17  30 0.05      0.13762810          0.804    0.190     0.006
18  30 0.10      0.20196806          0.822    0.172     0.006
19  30 0.20      0.27968413          0.939    0.045     0.016
20  30 0.25      0.30095295          0.932    0.047     0.021
21  30 0.35      0.33442332          0.906    0.062     0.032
22  30 0.45      0.34974136          0.925    0.032     0.043
23  30 0.50      0.35157175          0.949    0.025     0.026
24  30 0.60      0.34525061          0.939    0.022     0.039
25  30 0.65      0.33514009          0.914    0.031     0.055
26  30 0.75      0.30169289          0.942    0.024     0.034
27  30 0.80      0.27668373          0.953    0.011     0.036
28  30 0.90      0.20245953          0.817    0.007     0.176
29  30 0.95      0.13001885          0.764    0.004     0.232
30  30 0.99      0.03514146          0.258    0.000     0.742
31 100 0.01      0.03076036          0.646    0.354     0.000
32 100 0.05      0.08201684          0.880    0.119     0.001
33 100 0.10      0.11541423          0.928    0.068     0.004
34 100 0.20      0.15492461          0.929    0.054     0.017
35 100 0.25      0.16844619          0.945    0.042     0.013
36 100 0.35      0.18613109          0.949    0.026     0.025
37 100 0.45      0.19401228          0.945    0.026     0.029
38 100 0.50      0.19499296          0.942    0.032     0.026
39 100 0.60      0.19084405          0.946    0.024     0.030
40 100 0.65      0.18596768          0.945    0.023     0.032
41 100 0.75      0.16849354          0.949    0.017     0.034
42 100 0.80      0.15532542          0.943    0.013     0.044
43 100 0.90      0.11559181          0.935    0.008     0.057
44 100 0.95      0.08210201          0.881    0.004     0.115
45 100 0.99      0.02873285          0.605    0.000     0.395
```

```r
#Summary with just n, p, and coverage probabilities.
simplified_summary <- data.frame(sample_size = n_values, p = repeated_p, coverage_probs =
simplified_summary
```

```
   sample_size    p coverage_probs
1           15 0.01          0.137
```

2	15	0.05	0.522
3	15	0.10	0.787
4	15	0.20	0.789
5	15	0.25	0.891
6	15	0.35	0.894
7	15	0.45	0.927
8	15	0.50	0.876
9	15	0.60	0.942
10	15	0.65	0.884
11	15	0.75	0.908
12	15	0.80	0.823
13	15	0.90	0.812
14	15	0.95	0.525
15	15	0.99	0.166
16	30	0.01	0.267
17	30	0.05	0.804
18	30	0.10	0.822
19	30	0.20	0.939
20	30	0.25	0.932
21	30	0.35	0.906
22	30	0.45	0.925
23	30	0.50	0.949
24	30	0.60	0.939
25	30	0.65	0.914
26	30	0.75	0.942
27	30	0.80	0.953
28	30	0.90	0.817
29	30	0.95	0.764
30	30	0.99	0.258
31	100	0.01	0.646
32	100	0.05	0.880
33	100	0.10	0.928
34	100	0.20	0.929
35	100	0.25	0.945
36	100	0.35	0.949
37	100	0.45	0.945
38	100	0.50	0.942
39	100	0.60	0.946
40	100	0.65	0.945
41	100	0.75	0.949
42	100	0.80	0.943
43	100	0.90	0.935
44	100	0.95	0.881

```
45          100 0.99           0.605
```

```
library(ggplot2)
ggplot(simplified_summary, aes(x=p, y=coverage_probs)) +
  geom_line()+
  facet_wrap(~sample_size) +
  labs(title = "Wald Interval Coverage Probabilities by Sample Size",
       y = "Coverage Probabilities") +
  geom_hline(slope = 0, yintercept = 0.95, color = "blue")
```

```
Warning in geom_hline(slope = 0, yintercept = 0.95, color = "blue"): Ignoring
unknown parameters: `slope`
```