w271: Homework 1

Krysten Thompson

To person grading this: I removed code comments in original code block to reduce the number of pages in output. If this is wrong, please let me know. I was trying to be sensitive to page length.

Question 1: True Confidence Level of Various Confidence Intervals for One Binary Random Variable

During the live session in week 1, I explained why the Wald confidence interval does not always have the stated confidence level, $1 - \alpha$, where α, which is the probability of rejecting the null hypothesis when it is true, often is set to 0.05%, and I walked through the code below to explain the concept.

```
require(knitr)
```

```
## Loading required package: knitr
```

```
# Wrap long lines in R:
opts_chunk$set(tidy.opts=list(width.cutoff=80),tidy=TRUE)


pi = 0.6 # true parameter value of the probability of success
alpha = 0.05 # significane level
n = 10
w = 0:n

wald.CI.true.coverage = function(pi, alpha=0.05, n) {
  w = 0:n

  pi.hat = w/n
  pmf = dbinom(x=w, size=n, prob=pi)

  var.wald = pi.hat*(1-pi.hat)/n
  wald.CI_lower.bound = pi.hat - qnorm(p = 1-alpha/2)*sqrt(var.wald)
  wald.CI_upper.bound = pi.hat + qnorm(p = 1-alpha/2)*sqrt(var.wald)

  covered.pi = ifelse(test = pi>wald.CI_lower.bound,
                      yes = ifelse(test = pi<wald.CI_upper.bound, yes=1, no=0), no=0)

  wald.CI.true.coverage = sum(covered.pi*pmf)

  wald.df = data.frame(w, pi.hat,
                       round(data.frame(pmf, wald.CI_lower.bound,wald.CI_upper.bound),4),
                       covered.pi)
```

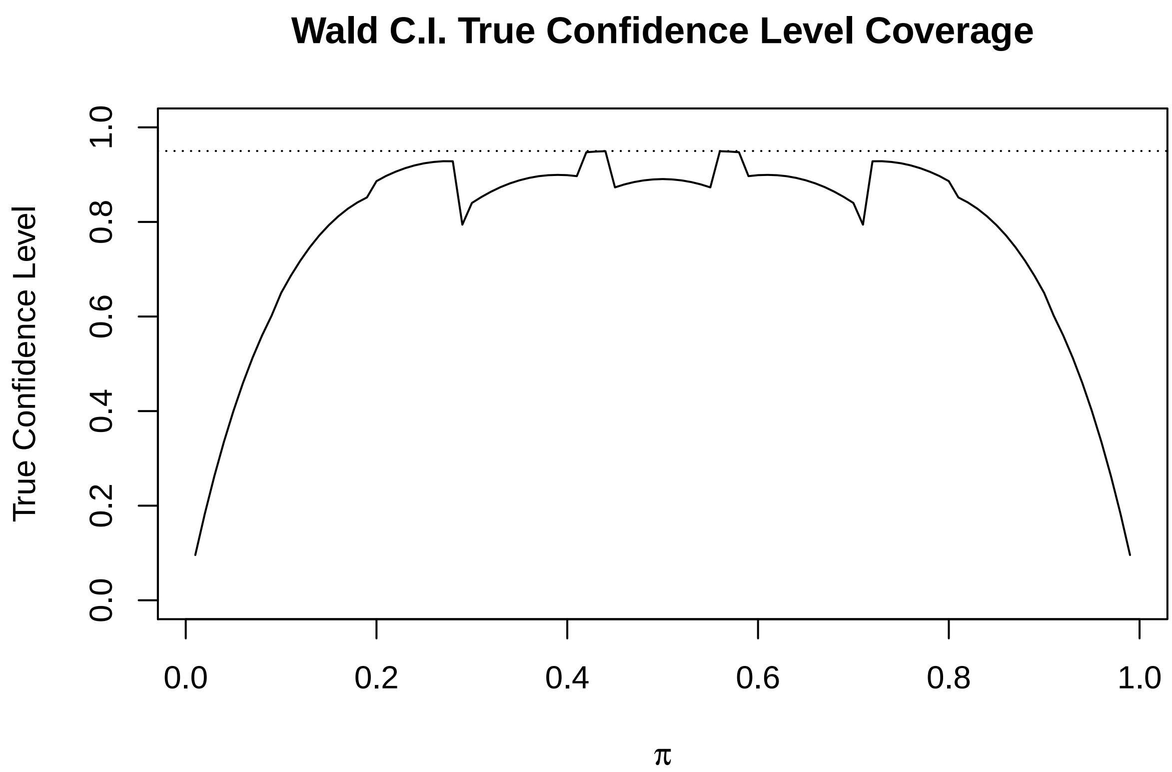
```r
  return(wald.df)
}

wald.df = wald.CI.true.coverage(pi=0.6, alpha=0.05, n=10)

wald.CI.true.coverage.level = sum(wald.df$covered.pi*wald.df$pmf)

pi.seq = seq(0.01, 0.99, by=0.01)

wald.CI.true.matrix = matrix(data=NA,nrow=length(pi.seq),ncol=2)

counter=1
for (pi in pi.seq) {
    wald.df2 = wald.CI.true.coverage(pi=pi, alpha=0.05, n=10)
    #print(paste('True Coverage is', sum(wald.df2$covered.pi*wald.df2$pmf)))
    wald.CI.true.matrix[counter,] = c(pi,sum(wald.df2$covered.pi*wald.df2$pmf))
    counter = counter+1
}
str(wald.CI.true.matrix)
```

```
##  num [1:99, 1:2] 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1 ...
```

```r
wald.CI.true.matrix[1:5,]
```

```
##      [,1]   [,2]
## [1,] 0.01 0.0956
## [2,] 0.02 0.1828
## [3,] 0.03 0.2624
## [4,] 0.04 0.3347
## [5,] 0.05 0.4002
```

```r
# Plot the true coverage level (for given n and alpha)
plot(x=wald.CI.true.matrix[,1],
     y=wald.CI.true.matrix[,2],
     ylim=c(0,1),
     main = "Wald C.I. True Confidence Level Coverage", xlab=expression(pi),
     ylab="True Confidence Level",
     type="l")
abline(h=1-alpha, lty="dotted")
```




```
wald.CI.true.coverage.level   #n=10
```

```
## [1] 0.8989
```

Question 1a: Use the code above and (1) redo the following exercise for $n = 50, n = 100, n = 500$, (2) plot the graphs, and (3) describe what you have observed from the results. Use the same $pi.seq$ as I used in the code above.

#This is $n = 50$

```
require(knitr)
# Wrap long lines in R:
opts_chunk$set(tidy.opts = list(width.cutoff = 80), tidy = TRUE)


pi = 0.6  # true parameter value of the probability of success
alpha = 0.05  # significance level
n = 50
w = 0:n

wald.CI.true.coverage = function(pi, alpha = 0.05, n) {

    w = 0:n

    pi.hat = w/n
    pmf = dbinom(x = w, size = n, prob = pi)

    var.wald = pi.hat * (1 - pi.hat)/n
```

```
    wald.CI_lower.bound = pi.hat - qnorm(p = 1 - alpha/2) * sqrt(var.wald)
    wald.CI_upper.bound = pi.hat + qnorm(p = 1 - alpha/2) * sqrt(var.wald)

    covered.pi = ifelse(test = pi > wald.CI_lower.bound, yes = ifelse(test = pi <
        wald.CI_upper.bound, yes = 1, no = 0), no = 0)

    wald.CI.true.coverage = sum(covered.pi * pmf)

    wald.df = data.frame(w, pi.hat, round(data.frame(pmf, wald.CI_lower.bound, wald.CI_upper.bo
        4), covered.pi)

    return(wald.df)
}

wald.df = wald.CI.true.coverage(pi = 0.6, alpha = 0.05, n = 50)

wald.CI.true.coverage.level = sum(wald.df$covered.pi * wald.df$pmf)

pi.seq = seq(0.01, 0.99, by = 0.01)

wald.CI.true.matrix = matrix(data = NA, nrow = length(pi.seq), ncol = 2)

counter = 1
for (pi in pi.seq) {
    wald.df2 = wald.CI.true.coverage(pi = pi, alpha = 0.05, n = 50)
    # print(paste('True Coverage is', sum(wald.df2$covered.pi*wald.df2$pmf)))
    wald.CI.true.matrix[counter, ] = c(pi, sum(wald.df2$covered.pi * wald.df2$pmf))
    counter = counter + 1
}
str(wald.CI.true.matrix)
```

```
##  num [1:99, 1:2] 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1 ...
```

```
wald.CI.true.matrix[1:5, ]
```

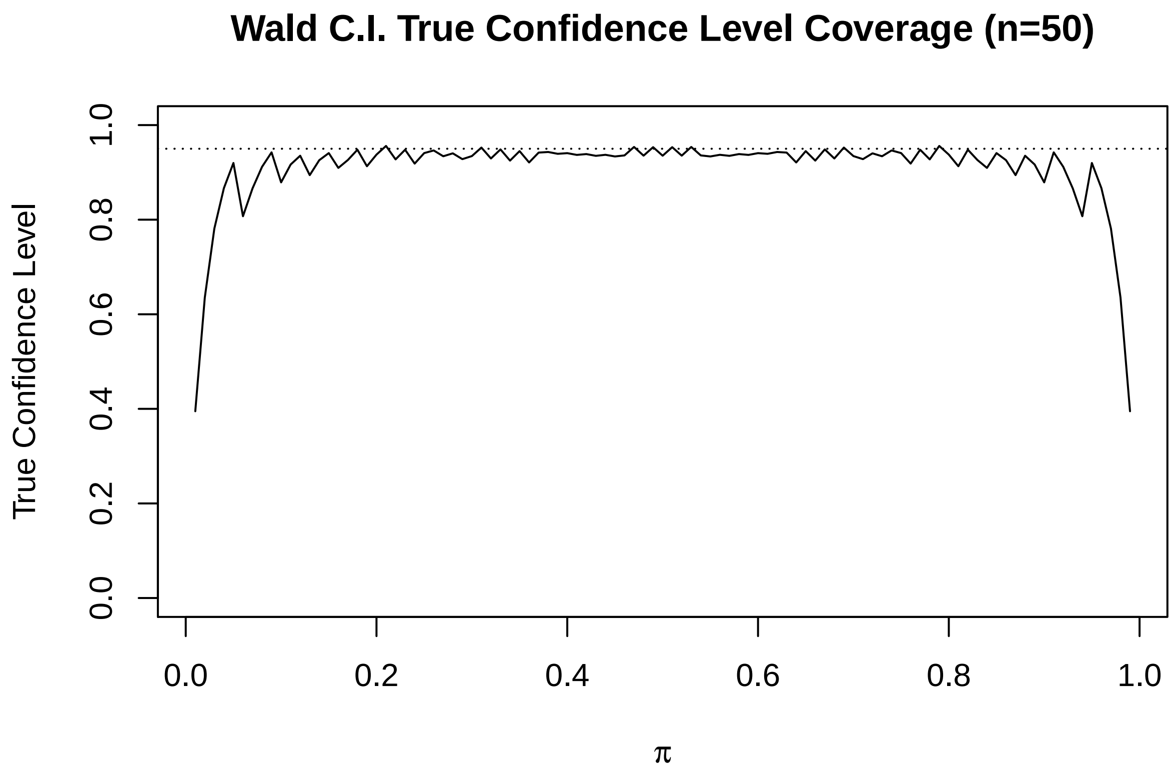
```
##      [,1]   [,2]
## [1,] 0.01 0.3949
## [2,] 0.02 0.6353
## [3,] 0.03 0.7811
## [4,] 0.04 0.8666
## [5,] 0.05 0.9199
```

```
wald.CI.true.coverage.level  #I wanted to see true CL for each level of n
```

```
## [1] 0.9407
```

```
# Plot the true coverage level (for given n and alpha)
plot(x = wald.CI.true.matrix[, 1], y = wald.CI.true.matrix[, 2], ylim = c(0, 1),
    main = "Wald C.I. True Confidence Level Coverage (n=50)", xlab = expression(pi),
```

```
    ylab = "True Confidence Level", type = "l")
abline(h = 1 - alpha, lty = "dotted")
```



#This is $n = 100$

```
require(knitr)
# Wrap long lines in R:
opts_chunk$set(tidy.opts = list(width.cutoff = 80), tidy = TRUE)


pi = 0.6  # true parameter value of the probability of success
alpha = 0.05  # significane level
n = 100
w = 0:n

wald.CI.true.coverage = function(pi, alpha = 0.05, n) {

    w = 0:n

    pi.hat = w/n
    pmf = dbinom(x = w, size = n, prob = pi)

    var.wald = pi.hat * (1 - pi.hat)/n
    wald.CI_lower.bound = pi.hat - qnorm(p = 1 - alpha/2) * sqrt(var.wald)
    wald.CI_upper.bound = pi.hat + qnorm(p = 1 - alpha/2) * sqrt(var.wald)
```

```r
    covered.pi = ifelse(test = pi > wald.CI_lower.bound, yes = ifelse(test = pi <
        wald.CI_upper.bound, yes = 1, no = 0), no = 0)

    wald.CI.true.coverage = sum(covered.pi * pmf)

    wald.df = data.frame(w, pi.hat, round(data.frame(pmf, wald.CI_lower.bound, wald.CI_upper.bo
        4), covered.pi)

    return(wald.df)
}

wald.df = wald.CI.true.coverage(pi = 0.6, alpha = 0.05, n = 100)

wald.CI.true.coverage.level = sum(wald.df$covered.pi * wald.df$pmf)

pi.seq = seq(0.01, 0.99, by = 0.01)

wald.CI.true.matrix = matrix(data = NA, nrow = length(pi.seq), ncol = 2)

counter = 1
for (pi in pi.seq) {
    wald.df2 = wald.CI.true.coverage(pi = pi, alpha = 0.05, n = 100)
    # print(paste('True Coverage is', sum(wald.df2$covered.pi*wald.df2$pmf)))
    wald.CI.true.matrix[counter, ] = c(pi, sum(wald.df2$covered.pi * wald.df2$pmf))
    counter = counter + 1
}
str(wald.CI.true.matrix)
```

```
##  num [1:99, 1:2] 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1 ...
```

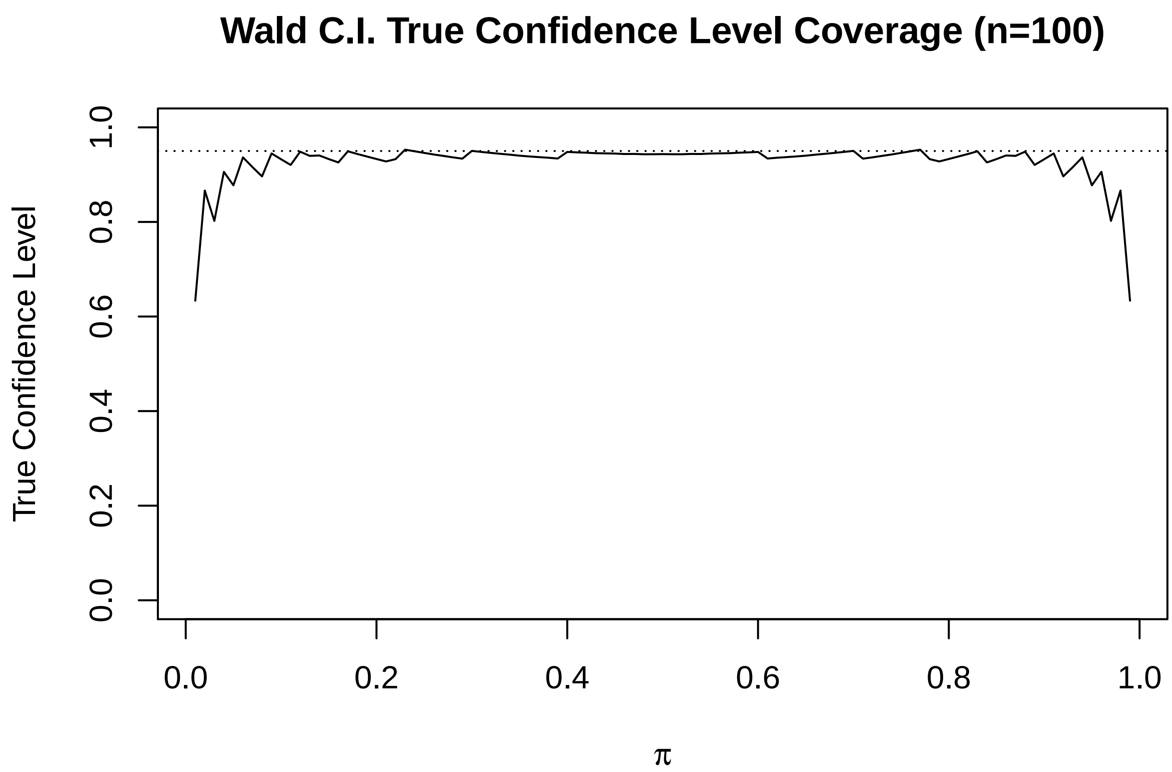
```r
wald.CI.true.matrix[1:5, ]
```

```
##      [,1]   [,2]
## [1,] 0.01 0.6334
## [2,] 0.02 0.8664
## [3,] 0.03 0.8022
## [4,] 0.04 0.9059
## [5,] 0.05 0.8774
```

```r
wald.CI.true.coverage.level
```

```
## [1] 0.948
```

```r
# Plot the true coverage level (for given n and alpha)
plot(x = wald.CI.true.matrix[, 1], y = wald.CI.true.matrix[, 2], ylim = c(0, 1),
    main = "Wald C.I. True Confidence Level Coverage (n=100)", xlab = expression(pi),
    ylab = "True Confidence Level", type = "l")
abline(h = 1 - alpha, lty = "dotted")
```



#This is $n = 500$

```
require(knitr)
# Wrap long lines in R:
opts_chunk$set(tidy.opts = list(width.cutoff = 80), tidy = TRUE)


pi = 0.6  # true parameter value of the probability of success
alpha = 0.05  # significane level
n = 500
w = 0:n

wald.CI.true.coverage = function(pi, alpha = 0.05, n) {

    w = 0:n

    pi.hat = w/n
    pmf = dbinom(x = w, size = n, prob = pi)

    var.wald = pi.hat * (1 - pi.hat)/n
    wald.CI_lower.bound = pi.hat - qnorm(p = 1 - alpha/2) * sqrt(var.wald)
    wald.CI_upper.bound = pi.hat + qnorm(p = 1 - alpha/2) * sqrt(var.wald)

    covered.pi = ifelse(test = pi > wald.CI_lower.bound, yes = ifelse(test = pi <
        wald.CI_upper.bound, yes = 1, no = 0), no = 0)
```

```
    wald.CI.true.coverage = sum(covered.pi * pmf)

    wald.df = data.frame(w, pi.hat, round(data.frame(pmf, wald.CI_lower.bound, wald.CI_upper.bo
        4), covered.pi)

    return(wald.df)
}


wald.df = wald.CI.true.coverage(pi = 0.6, alpha = 0.05, n = 500)

wald.CI.true.coverage.level = sum(wald.df$covered.pi * wald.df$pmf)

pi.seq = seq(0.01, 0.99, by = 0.01)

wald.CI.true.matrix = matrix(data = NA, nrow = length(pi.seq), ncol = 2)

counter = 1
for (pi in pi.seq) {
    wald.df2 = wald.CI.true.coverage(pi = pi, alpha = 0.05, n = 500)
    # print(paste('True Coverage is', sum(wald.df2$covered.pi*wald.df2$pmf)))
    wald.CI.true.matrix[counter, ] = c(pi, sum(wald.df2$covered.pi * wald.df2$pmf))
    counter = counter + 1
}
str(wald.CI.true.matrix)
```

```
##  num [1:99, 1:2] 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1 ...
```

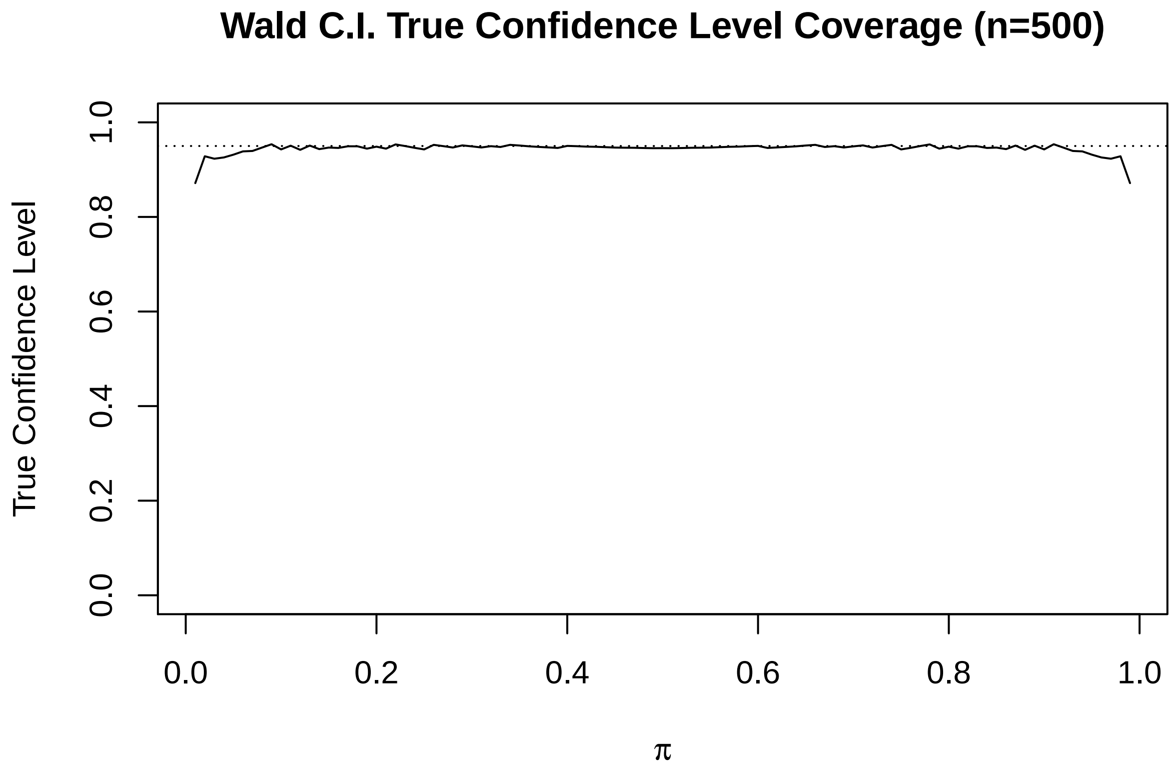
```
wald.CI.true.matrix[1:5, ]
```

```
##      [,1]   [,2]
## [1,] 0.01 0.8715
## [2,] 0.02 0.9283
## [3,] 0.03 0.9231
## [4,] 0.04 0.9258
## [5,] 0.05 0.9317
```

```
wald.CI.true.coverage.level
```

```
## [1] 0.9502
```

```
# Plot the true coverage level (for given n and alpha)
plot(x = wald.CI.true.matrix[, 1], y = wald.CI.true.matrix[, 2], ylim = c(0, 1),
    main = "Wald C.I. True Confidence Level Coverage (n=500)", xlab = expression(pi),
    ylab = "True Confidence Level", type = "l")
abline(h = 1 - alpha, lty = "dotted")
```



The Wald test is known for not working well on small sizes of n. This is evident not only in the graphs where accuracy improves as n increases from 50 to 500, but also in the true Confidence Level. At $n = 50$, the true CL equals 94.07% which is not even 95%. At $n = 100$, the CL gets closer to 95% at 94.8% (and rounding up hits 95%). At $n = 500$, we finally see a CL at 95.02%.

#

Question 1b: (1) Modify the code above for the Wilson Interval. (2) Do the exercise for $n = 10, n = 50, n = 100, n = 500$. (3) Plot the graphs. (4) Describe what you have observed from the results and compare the Wald and Wilson intervals based on your results. Use the same *pi.seq* as in the code above.

I could not get the Wilson confidence interval to work after struggling for 1hr+. I commented out my code to show that I tried. Because of some code errors, it was not knitting correctly so I had to comment it all out.

#This is $n = 50$

```
# require(knitr) opts_chunk$set(tidy.opts=list(width.cutoff=80),tidy=TRUE) n = 50
# w = 18 alpha = 0.05 # significance level pi.hat <- w/n

# wilson.CI.true.coverage = function(alpha=0.05, n) { w = 0:n pi.hat = w/n pmf =
# dbinom(x=w, size=n, prob=pi.hat) var.wilson = pi.hat*(1-pi.hat)/n
# wilson.CI_lower.bound = pi.hat - qnorm(p = 1-alpha/2)*sqrt(var.wilson)
# wald.CI_upper.bound = pi.hat + qnorm(p = 1-alpha/2)*sqrt(var.wilson) covered.pi
# = ifelse(test = pi>wilson.CI_lower.bound, yes = ifelse(test = pi
# <wilson.CI_upper.bound, yes=1, no=0), no=0) wilson.CI.true.coverage =
# sum(covered.pi*pmf) wilson.df = data.frame(w, pi.hat, round(data.frame(pmf,
# wilson.CI_lower.bound,wilson.CI_upper.bound),4), covered.pi) return(wilson.df)
# } #Adjusted estimate of pi pi.tilde <- (w + qnorm(p = 1-alpha/2)^2 /2) / (n +
# qnorm(p = 1-alpha/2)^2) #Wilson Confidence Internval wilson.CI <-
# round(pi.tilde + qnorm(p = c(alpha/2, 1-alpha/2)) * sqrt(n) / (n + qnorm(p =
# 1-alpha/2)^2) * sqrt(pi.hat*(1-pi.hat) + qnorm(p = 1-alpha/2)^2/(4*n)), 4)
# pi.seq = seq(0.01, 0.99, by=0.01) # Create a matrix to store (1) pi.hat and (2)
# the true confidence level wilson.CI.true.matrix =
# matrix(data=NA,nrow=length(pi.seq),ncol=2) counter=1 for (pi in pi.seq) {
# wilson.df2 = wilson.CI.true.coverage(alpha=0.05, n=50) #print(paste('True
# Coverage is', sum(wald.df2$covered.pi*wald.df2$pmf)))
# wilson.CI.true.matrix[counter,] =
# c(pi,sum(wilson.df2$covered.pi*wilson.df2$pmf)) counter = counter+1 }
# str(wilson.CI.true.matrix) wilson.CI.true.matrix[1:5,] pi.tilde wilson.CI
```

DO OVER ***********************************

```
# require(knitr) opts_chunk$set(tidy.opts=list(width.cutoff=80),tidy=TRUE) n = 50
# w = 18 alpha = 0.05 # significance level pi.hat <- w/n p.tilde<-(w + qnorm(p =
# 1-alpha/2)^2 /2) / (n+qnorm(1-alpha/2)^2) lower.wilson<-p.tilde - qnorm(p =
# 1-alpha/2) * sqrt(n) / (n+qnorm(1-alpha/2)^2) * sqrt(pi.hat*(1-pi.hat) +
# qnorm(1-alpha/2)^2/(4*n)) upper.wilson<-p.tilde + qnorm(p = 1-alpha/2) *
# sqrt(n) / (n+qnorm(1-alpha/2)^2) * sqrt(pi.hat*(1-pi.hat) +
# qnorm(1-alpha/2)^2/(4*n)) pi.seq<- seq(0.01, 0.99, by=0.01) # Save true
# confidence levels in a matrix save.true.conf<-matrix(data = NA, nrow =
# length(pi.seq), ncol = 2) #wilson.CI.true.matrix =
# matrix(data=NA,nrow=length(pi.seq),ncol=2) # Create counter for the loop
# counter<-1 # Loop over each pi that the true confidence level is calculated on
# for(pi in pi.seq) { pmf<-dbinom(x = w, size = n, prob = pi) # Wilson
# save.wilson<-ifelse(test = pi>lower.wilson, yes = ifelse(test =
```

```
# pi<upper.wilson, yes = 1, no = 0), no = 0) wilson<-sum(save.wilson*pmf) } #
# Plots plot(x = save.true.conf[,1], y = save.true.conf[,4], main = 'Wilson',
# xlab = expression(pi), ylab = 'True confidence level', type = 'l', ylim =
# c(0.85,1)) abline(h = 1-alpha, lty = 'dotted')
```

Question 2: Confidence Interval Interpretation

Is it okay to say that the "estimated" confidence interval has $(1-\alpha)100\%$ probability of containing the ture parameter, named θ?

For instance, suppose we have a sample of data, and we use that sample to estimate a parameter, θ, of a statistical model and the confidence interval of the estimate. Suppose the resulting estimated 95% confidence interval is $[-2, 2]$. From a frequentist perspective, can we say that this estimated confidence interval contains the true parameter, θ, 95% of the time?

Please answer (1) Yes or No, and (2) give the reasoning of your answer provided in (1).

No. What we can say is one of two things about θ and 95% confidence interval:

1. With 95% confidence, the true probability of success is between -2 and 2.
2. Given that the 95% confidence interval is between -2 and 2, we would expect that 95% of all similarly constructed intervals to contain *pi*.

Question 3: Odds Ratios

When studying the multiple binary random varibles, we often use the notion of odds. The "odds" is simply the probability of a success divided by the probability of a failure: $\frac{\pi}{1-\pi}$

Suppose $\pi = 0.1$

Question 3a: What are the corresponding odds?

$\frac{\pi}{1-\pi} = \frac{0.1}{1-0.1} = \frac{0.1}{0.9}$

The odds are 1 : 9.

Question 3b: Interpret it in the following two types of statements

- **1. The odds of success are X. (Fill in X)**

The odds of success are 1.

- **2. The probability of failure is X times the probability of success. (Fill in X)**

The probability of failure is 9 times the probability of success.

The notion of odds ratio becomes relevant when there are more than one groups and we to compare their odds.

The odds ratio is the ratio of two odds. Mathematically, it is

$$OR = \frac{odds_1}{odds_2} = \frac{\pi_1/(1-\pi_1)}{\pi_2/(1-\pi_2)} = \frac{\pi_1(1-\pi_2)}{\pi_2(1-\pi_1)}$$

where

- π_i denotes the probability of success of Group i, $i \in \{1, 2\}$
- $odds_i$ represents the odds of a success of group i, $i \in \{1, 2\}$

Question 3c: Suppose the $OR = 3$. Write down the odds of success of group 1 in relation to the odds of success of group 2.

The odds of success for Group 1 equal $9:1$. The odds of success for Group 2 equal $3:1$.

Question 4: Binary Logistic Regression

Do **Exercise 8 a, b, c, and d (on page 131 of Bilder and Loughin's textbook)**. Please write down each of the questions. The dataset for this question is stored in the file *"placekick.BW.csv"*. The dataset is provided to you. In general, all the R codes and datasets used in Bilder and Loughin's book are provided on the book's website: chrisbilder.com

For **question 8b**, change it to the following: Re-estimate the model in part (a) using "*Sun*" as the base level category for *Weather*.

Using Distance, Weather, Wind15, Temperature, Grass, Pressure, and Ice explanatory variables as linear terms in a logistic regression model as follows:

```
placekick <- read.table(file = "placekick.BW.csv", header = TRUE, sep = ",")
head(placekick)  #This was to make sure it worked correctly
```

```
##     GameNum Kicker Good Distance Weather Wind15 Temperature Grass Pressure
## 1 2002-0101 Bryant    Y       29     Sun      0        Nice     1        N
## 2 2002-0101 Bryant    Y       33     Sun      0        Nice     1        N
## 3 2002-0101 Cortez    N       25     Sun      0        Nice     1        N
## 4 2002-0101 Cortez    Y       23     Sun      0        Nice     1        N
## 5 2002-0101 Cortez    N       48     Sun      0        Nice     1        N
## 6 2002-0101 Cortez    Y       33     Sun      0        Nice     1        N
##   Ice
## 1   0
## 2   0
## 3   0
## 4   0
## 5   0
## 6   0
```

```r
# tail(placekick)
```

(a) Estimate the model and properly define the indicator variables used within it

The indicator variables in this model are: Distance, Weather, Wind, Temperature, Grass, Pressure, and Ice.

```r
# Estimate the model

mod.fit.4a <- glm(formula = Good ~ Distance + Weather + Wind15 + Temperature + Grass +
    Pressure + Ice, family = binomial(link = logit), data = placekick)
mod.fit.4a
```

```
##
## Call:  glm(formula = Good ~ Distance + Weather + Wind15 + Temperature +
##     Grass + Pressure + Ice, family = binomial(link = logit),
##     data = placekick)
##
## Coefficients:
##     (Intercept)         Distance    WeatherInside  WeatherSnowRain
##         5.74018         -0.10960         -0.08303         -0.44419
##      WeatherSun           Wind15   TemperatureHot  TemperatureNice
##        -0.24758         -0.24378          0.25001          0.23493
##           Grass        PressureY              Ice
##        -0.32843          0.27017         -0.87613
##
## Degrees of Freedom: 2002 Total (i.e. Null);  1992 Residual
## Null Deviance:       2104
## Residual Deviance: 1791  AIC: 1813
```

(b) Re-estimate the model in part (a) using "*Sun*" as the base level category for *Weather*.

```
sun.var <- placekick$Weather == "Sun"

mod.fit.4b <- glm(formula = Good ~ Distance + sun.var + Wind15 + Temperature + Grass +
    Pressure + Ice, family = binomial(link = logit), data = placekick)

mod.fit.4b
```

```
##
## Call:  glm(formula = Good ~ Distance + sun.var + Wind15 + Temperature +
##     Grass + Pressure + Ice, family = binomial(link = logit),
##     data = placekick)
##
## Coefficients:
##     (Intercept)         Distance      sun.varTRUE           Wind15
##          5.6087          -0.1094          -0.1606          -0.2701
##  TemperatureHot  TemperatureNice            Grass        PressureY
##          0.3055           0.2841          -0.3340           0.2774
##             Ice
##         -0.8679
##
## Degrees of Freedom: 2002 Total (i.e. Null);  1994 Residual
## Null Deviance:       2104
## Residual Deviance: 1795  AIC: 1813
```

(c) Perform LRT's for all explanatory variables to evaluate their importance within the model. Discuss the results.

Using Anova():

```
library(car)
```

```
## Loading required package: carData
```

```
Anova(mod.fit.4a, test = "LR")
```

```
## Analysis of Deviance Table (Type II tests)
##
## Response: Good
##             LR Chisq Df Pr(>Chisq)
## Distance     294.341  1    < 2e-16 ***
## Weather        5.670  3    0.12884
## Wind15         1.898  1    0.16833
## Temperature    1.723  2    0.42254
## Grass          4.314  1    0.03781 *
## Pressure       1.088  1    0.29682
## Ice            3.698  1    0.05448 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

For the Anova() test, the extremely low p-value for distance indicates its importance when the model includes Weather, Wind Speed, Temp, Grass, Pressure, and Ice. Additionally, the indicator variable Grass is also important for a successful field goal given its p-value of 0.0378.

Using anova()

```r
anova(mod.fit.4a, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: Good
##
## Terms added sequentially (first to last)
##
##
##             Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                         2002     2104.0
## Distance     1  287.047      2001     1817.0 < 2.2e-16 ***
## Weather      3   13.424      1998     1803.6  0.003804 **
## Wind15       1    2.090      1997     1801.5  0.148228
## Temperature  2    1.831      1995     1799.7  0.400249
## Grass        1    4.659      1994     1795.0  0.030884 *
## Pressure     1    0.003      1993     1795.0  0.954510
## Ice          1    3.698      1992     1791.3  0.054479 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The anova() function differs from Anova() by first calculating the null-hypothesis, then building on itself according to the order of the explanatory variables.

Using anova() and specifying NULL hypothesis model because I wanted to see the difference between calling anova() on one model versus creating NULL-hypothesis model and "all in" model.

```r
model.fit.Ho <- glm(formula = Good ~ 1, family = binomial(link = logit), data = placekick)

model.fit.Ha <- glm(formula = Good ~ Distance + Weather + Wind15 + Temperature +
    Grass + Pressure + Ice, family = binomial(link = logit), data = placekick)

anova(model.fit.Ho, model.fit.Ha, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: Good ~ 1
## Model 2: Good ~ Distance + Weather + Wind15 + Temperature + Grass + Pressure +
##     Ice
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1      2002     2104.0
## 2      1992     1791.3 10   312.75 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

(d) Estimate an appropriate odds ratio for distance, and compute the corresponding confidence interval. Interpret the odds ratio.

```
mod.fit.4d <- glm(formula = Good ~ Distance, family = binomial(link = logit), data = placekick)

exp(mod.fit.4d$coefficients[2])
```

```
##  Distance
## 0.8991814
```

```
mod.fit.4d.CI <- confint(object = mod.fit.4d, parm = "Distance", level = 0.95)
```

```
## Waiting for profiling to be done...
```

```
mod.fit.4d.CI
```

```
##       2.5 %     97.5 %
## -0.1202893 -0.0927308
```

```
# The book decreased yardage by 10 for calculations; I wanted to see what would
# happen if yardage increased by 10

rev(exp(10 * mod.fit.4d.CI))
```

```
##    97.5 %     2.5 %
## 0.3956173 0.3003243
```

```
rev(1/exp(mod.fit.4d.CI * -10))
```

```
##    97.5 %     2.5 %
## 0.3956173 0.3003243
```

For the explanatory variable "Distance", $\hat{\beta} = 0.8992$ meaning that each 1-yard change in fieldkick ball placement yields a 0.8992 change in success.

We can also say that with 95% confidence, for every 10-yard increase in field kick length, the odds of success decrease between 0.30 and 0.40.