w271: Homework 1 (Due: Week 2) with Suggested Solutions

Professor Jeffrey Yau

Due: Before the Live Session of Week 2

Instructions (Please Read it Carefully!):

- **Page limit of the pdf report: None, but be reasonable**
- Page setup:
- Do not play around with the margin, linespace, and font size;
- Use the one specified below:
  - fontsize=11pt
  - margin=1in
  - line_spacing=single
- Submission:
  - Homework needs to be completed individually; this is not a group project. Each student needs to submit his/her homework to the course github repo by the deadline; submission and revision made after the deadline will not be graded
  - Submit 2 files:
    1. A pdf file that details your answers. Include all the R codes used to produce the answers. *Please do not suppress the codes in your pdf file.*
    2. R markdown file used to produce the pdf file
  - Use the following file-naming convensation; fail to do so will receive 10% reduction in the grade:
    - StudentFirstNameLastName_HWNumber.fileExtension
    - For example, if the student's name is Kyle Cartman for homework 1, name your files as the follow
      - KyleCartman_HW1.Rmd
      - KyleCartman_HW1.pdf
  - Although it sounds obvious, please write your name on page 1 of your pdf and Rmd files.
  - For statistical methods that we cover in this course, use only the R libraries and functions that are covered in this course. If you use libraries and functions for statistical modeling that we have not covered, you have to (1) provide an explanation of why such libraries and functions are used instead and (2) reference to the library documentation. **Lacking the explanation and reference to the documentation will result in a score of zero for the corresponding question.** For data wrangling and data visualization, you are free to use other libraries, such as dplyr, ggplot2, etc.

- For mathematical formulae, type them in your R markdown file. **Do not write them on a piece of paper, snap a photo, and either insert the image file or sumbit the image file separately. Doing so will receive a** 0 **for that whole question.**
- Students are expected to act with regards to UC Berkeley Academic Integrity.

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) {

  # Objective:
  #    Calculate the true confidence level of a Wald Confidence (given pi, alpha, and n)

  # Input:
  #    pi: the true parameter value
  #    alpha: significance level
  #    n: the number of trials

  # Return:
  #    wald.df: a data.frame containing
  #    (1) observed number of success, w
  #    (2) MLE of pi, pi.hat
  #    (3) Binomial probability of obtaining the number of successes from n trials, pmf
  #    (4) lower bound of the Wald confidence interval, wald.CI_lower.bound
  #    (5) upper bound of the Wald confidence interval, wald.CI_upper.bound
  #    (6) whether or not an interval contains the true parameter, covered.pi

  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,
```

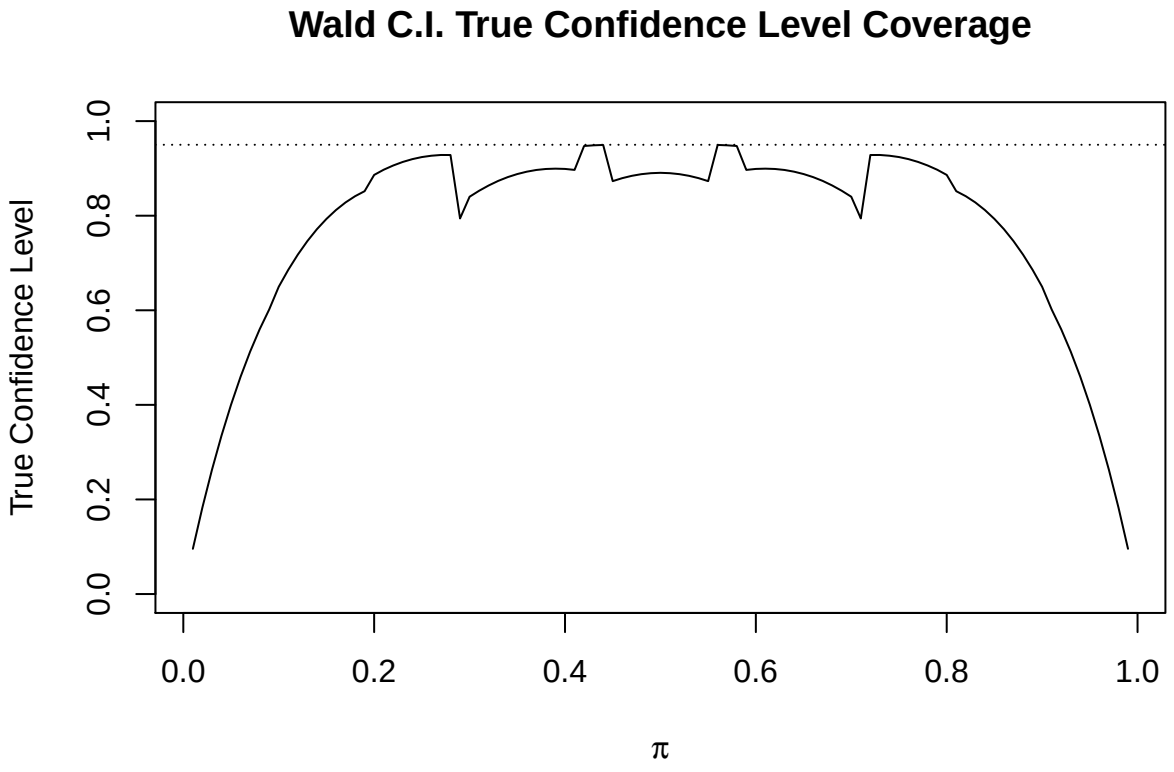
```
                      yes = ifelse(test = pi<wald.CI_upper.bound, yes=1, no=0), no=0)

  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)

  return(wald.df)
}

# Call the function with user-provided arguments (pi, alpha, n) to
# generate the data.frame that contains
# (1) the observed number of success, w
# (2) MLE of pi, pi.hat
# (3) Binomial probability of obtaining the number of successes from n trials, pmf
# (4) the lower bound of the Wald confidence interval, wald.CI_lower.bound
# (5) the upper bound of the Wald confidence interval, wald.CI_upper.bound
# (6) whether or not an interval contains the true parameter, covered.pi

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

# Obtain the true confidence level from the Wald Confidence,
# given pi, alpha, and n
wald.CI.true.coverage.level = sum(wald.df$covered.pi*wald.df$pmf)

# Generalize the above computation to a sequence of pi's

# Generate an example sequence of pi (feel free to make the increment smaller)
pi.seq = seq(0.01, 0.99, by=0.01)

# Create a matrix to store (1) pi and (2) the true confidence level of
# the Wald Confidence Interval corresponding to the specific pi
wald.CI.true.matrix = matrix(data=NA,nrow=length(pi.seq),ncol=2)

# Loop through the sequence of pi's to obtain the true confidence level of
# the Wald Confidence Interval corresponding to the specific pi
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 ...
```

```
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
```

```
# 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")
```



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.

Answer:

To redo the exercise for alternative n, it is easy to first pack all the codes into a function.

** 1a.1 Answer: Redo the exercise for $n = 50, n = 100, n = 500$** We could create separate data frames to store the result for a specific π and α.

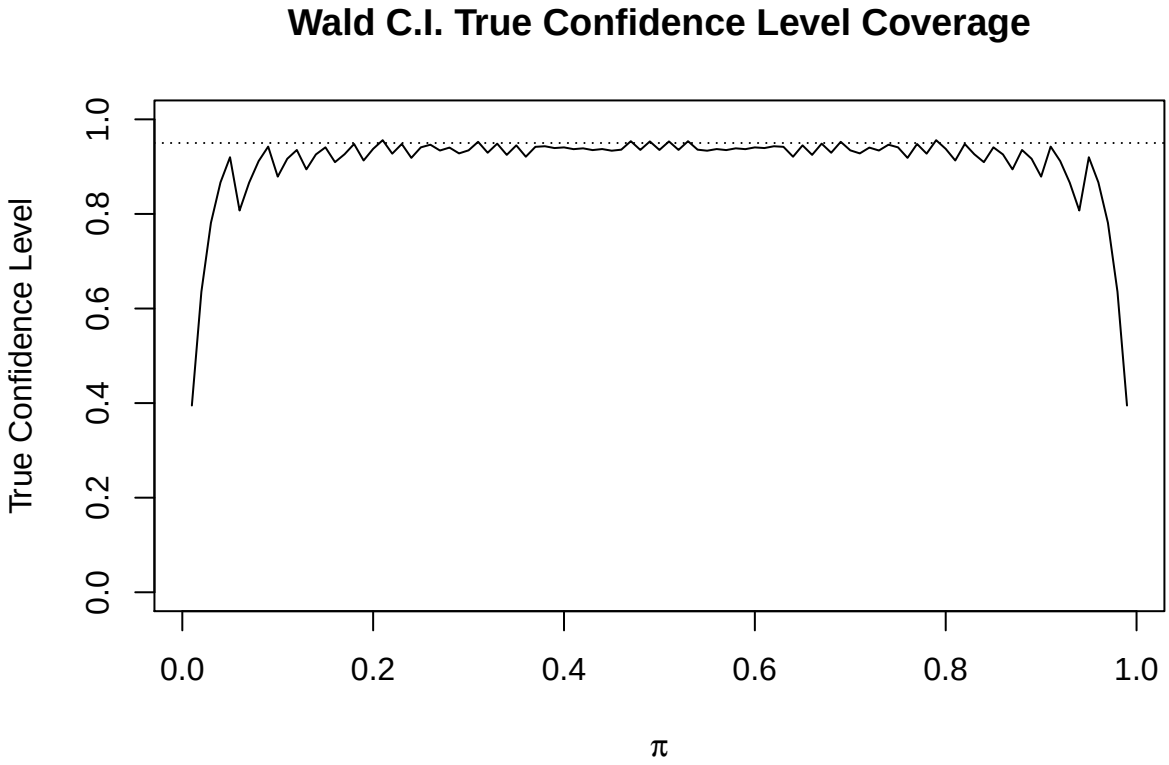
```r
# define a list containing the 3 n's loop over the n's and compute the confidence
# intervals

df.name = list("wald.df.n50", "wald.df.n100", "wald.df.n500")
n_seq = list(50, 100, 500)

for (i in 1:3) {
    print(df.name[[i]])
    print(n_seq[[i]])
    df.name[[i]] = wald.CI.true.coverage(pi = 0.6, alpha = 0.05, n = n_seq[[i]])
}
```

```
## [1] "wald.df.n50"
## [1] 50
## [1] "wald.df.n100"
## [1] 100
## [1] "wald.df.n500"
## [1] 500
```

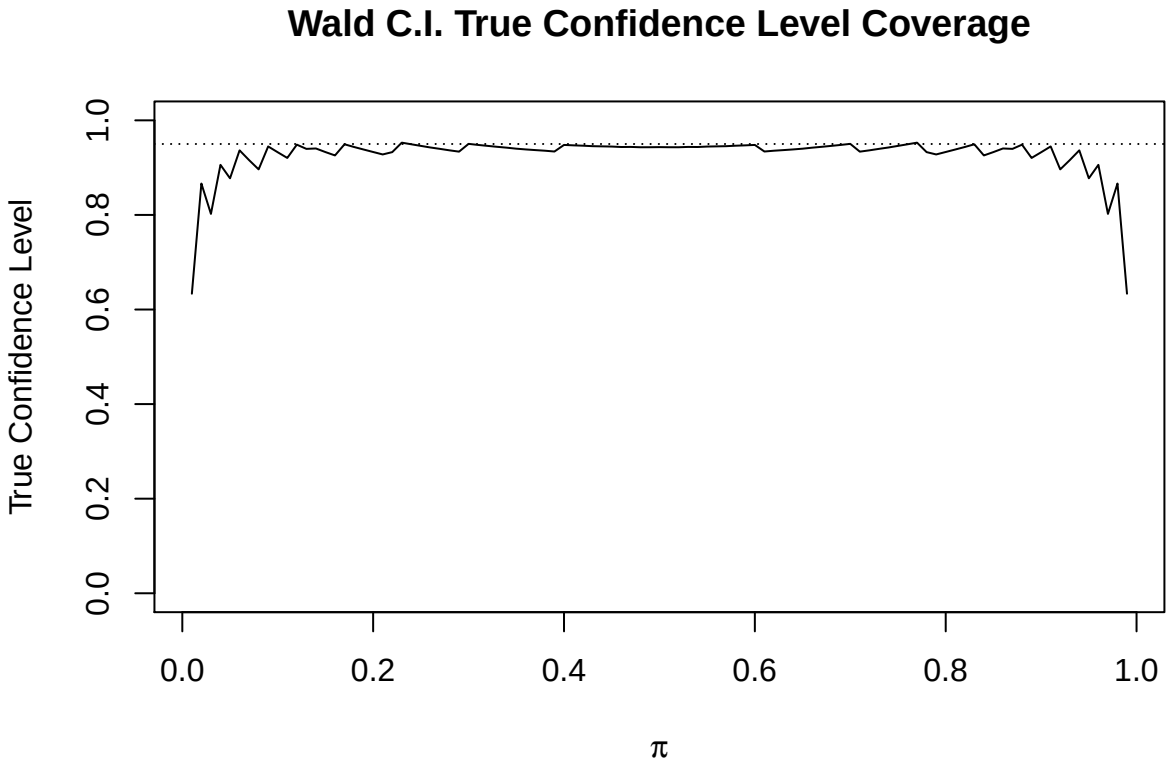
1a.2 Answer: Plot the graphs

```r
pi.seq = seq(0.01, 0.99, by = 0.01)
n_seq = list(50, 100, 500)
wald.CI.true.matrix = matrix(data = NA, nrow = length(pi.seq), ncol = 2)

for (i in 1:3) {
    counter = 1
    for (pi in pi.seq) {
        wald.df2 = wald.CI.true.coverage(pi = pi, alpha = 0.05, n = n_seq[[i]])
        wald.CI.true.matrix[counter, ] = c(pi, sum(wald.df2$covered.pi * wald.df2$pmf))
        counter = counter + 1
    }
    str(wald.CI.true.matrix)
    wald.CI.true.matrix[1:5, ]

    # 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")
}
```

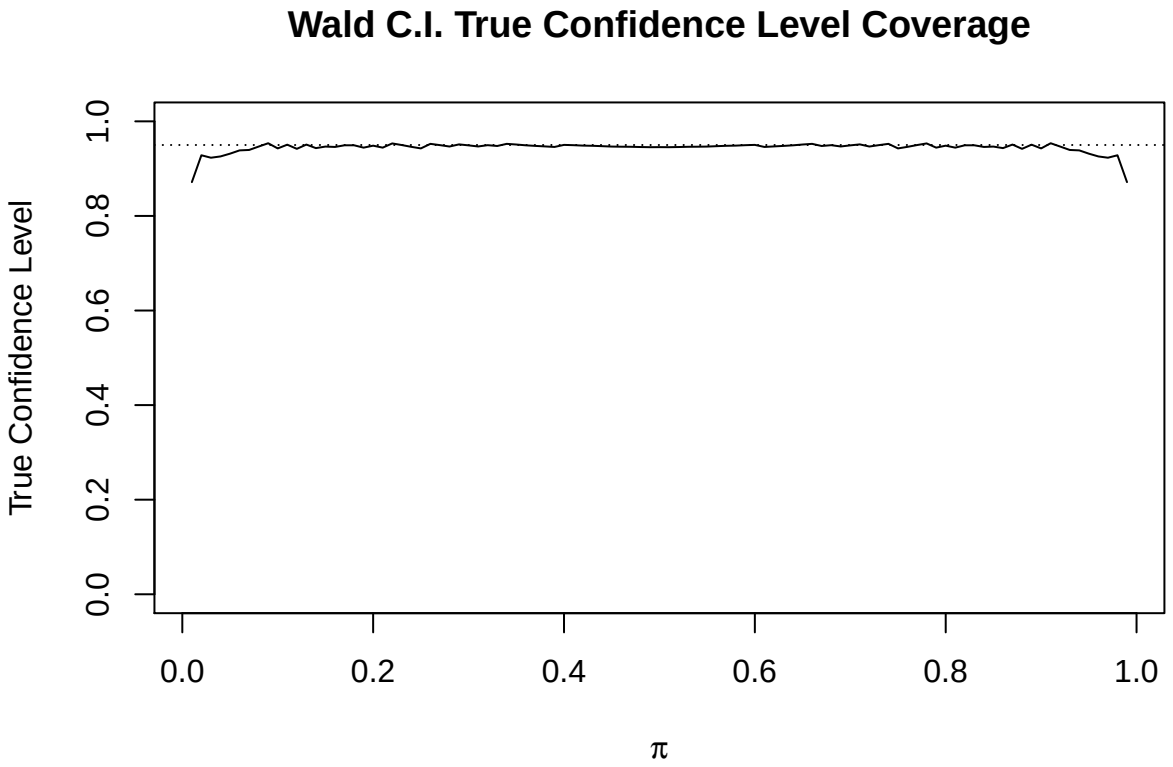
```
##  num [1:99, 1:2] 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1 ...
```




```
##  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 ...
```




```
##  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 ...
```



Q1a.3. Answer: Your observation. As the number of trials, n, increases, so is the Wald confidence interval approximation. In fact, as $n = 500$, the stated confidence levels are very close to the true confidence level (i.e. $\alpha = 0.95$) except when π's are close to the two extremes.

Question 1b: (1) Modify the code above for the Wilson Confidence 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.

Wilson Confidence Interval

Question 1b.1 Answer: Redo the exercise for Wilson Confidence Interval

The *Wilson Confidence Interval* takes the following functional form:

$$\tilde{\pi} \pm \frac{Z_{1-\frac{\alpha}{2}} n^{1/2}}{n + Z^2_{1-\frac{\alpha}{2}}} \sqrt{\hat{\pi}(1-\hat{\pi}) + \frac{Z^2_{1-\frac{\alpha}{2}}}{4n}}$$

where $\tilde{\pi} = \frac{w + \frac{1}{2} Z^2_{1-\frac{\alpha}{2}}/2}{n + Z^2_{1-\frac{\alpha}{2}}}$, which can be considered as an "adjusted" estimate of π.

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

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

    # Objective: Calculate the true confidence level of a Wilson Confidence Interval
```

```
    # (given pi, alpha, and n)

    # Input: pi: the true parameter value alpha: significance level n: the number of
    # trials

    # Return: wilson.df: a data.frame containing (1) observed number of success, w
    # (2) pi.tilde (can be considered as adjusted MLE of pi) (3) Binomial probability
    # of obtaining the number of successes from n trials, pmf (4) lower bound of the
    # Wilson confidence interval, wilson.CI_lower.bound (5) upper bound of the Wilson
    # confidence interval, wilson.CI_upper.bound (6) whether or not an interval
    # contains the true parameter, covered.pi

    w = 0:n
    pi.hat = w/n

    pmf = dbinom(x = w, size = n, prob = pi)
    z = qnorm(p = 1 - alpha/2)
    pi.tilde = (w + z^2/2)/(n + z^2)

    wilson.CI_lower.bound = pi.tilde - ((z * sqrt(n))/(n + z^2)) * sqrt(pi.hat *
        (1 - pi.hat) + (z^2)/(4 * n))
    wilson.CI_upper.bound = pi.tilde + ((z * sqrt(n))/(n + z^2)) * sqrt(pi.hat *
        (1 - pi.hat) + (z^2)/(4 * n))

    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.tilde, round(data.frame(pmf, wilson.CI_lower.bound,
        wilson.CI_upper.bound), 4), covered.pi)

    return(wilson.df)
}
```

We could create separate data frames to store the result for a specific π and α.

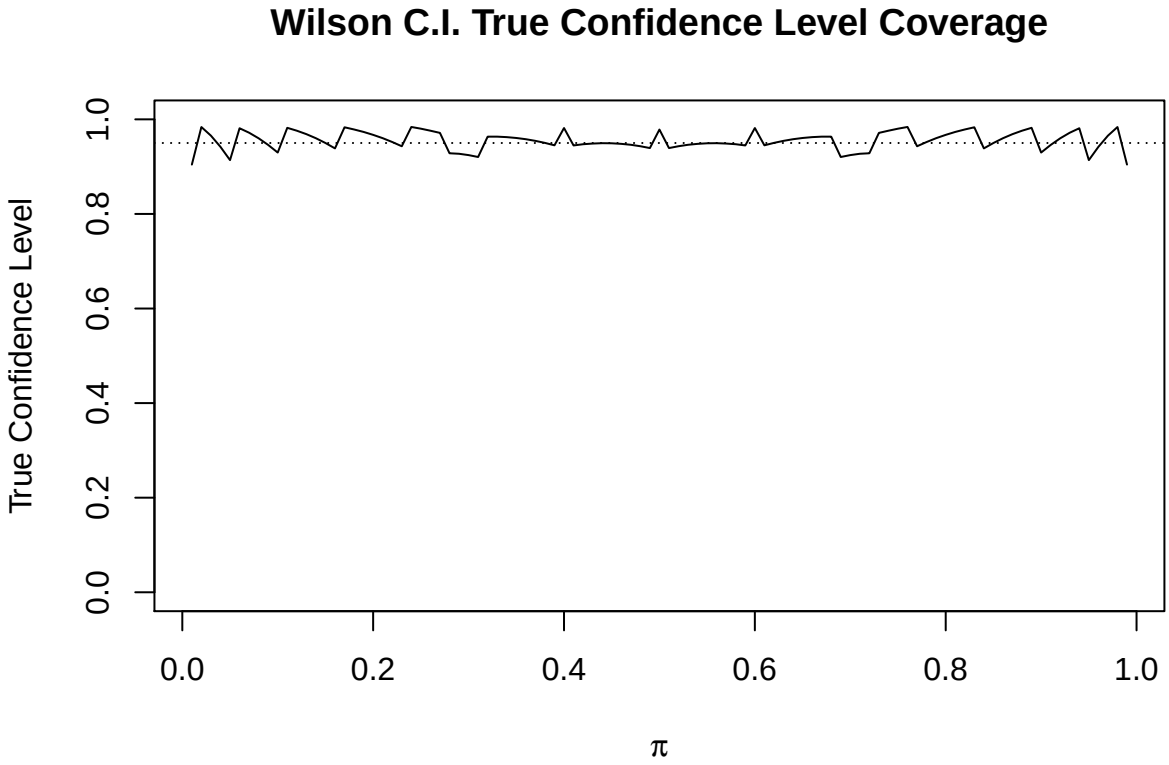
```
# define a list containing the 3 n's loop over the n's and compute the confidence
# intervals

df.wilson.CI = list("wilson.df.n10", "wilson.df.n50", "wilson.df.n100", "wilson.df.n500")
n_seq = list(10, 50, 100, 500)

for (i in 1:4) {
    print(df.wilson.CI[[i]])
    print(n_seq[[i]])
    df.wilson.CI[[i]] = wilson.CI.true.coverage(pi = 0.6, alpha = 0.05, n = n_seq[[i]])
}
```

```
## [1] "wilson.df.n10"
## [1] 10
## [1] "wilson.df.n50"
## [1] 50
## [1] "wilson.df.n100"
## [1] 100
## [1] "wilson.df.n500"
## [1] 500
```

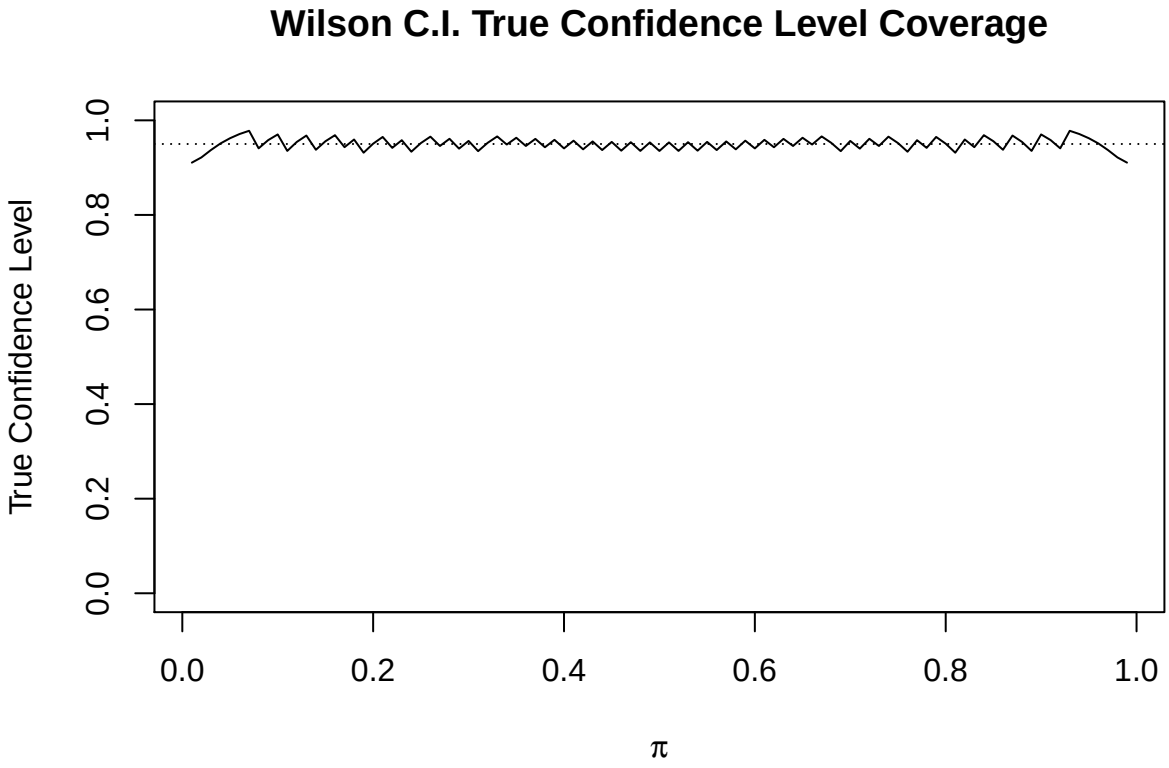
1b.2 Plot the graphs

```r
pi.seq = seq(0.01, 0.99, by = 0.01)
n_seq = list(10, 50, 100, 500)
wilson.CI.true.matrix = matrix(data = NA, nrow = length(pi.seq), ncol = 2)

for (i in 1:4) {
    counter = 1
    for (pi in pi.seq) {
        wilson.df2 = wilson.CI.true.coverage(pi = pi, alpha = 0.05, n = n_seq[[i]])
        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, ]

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

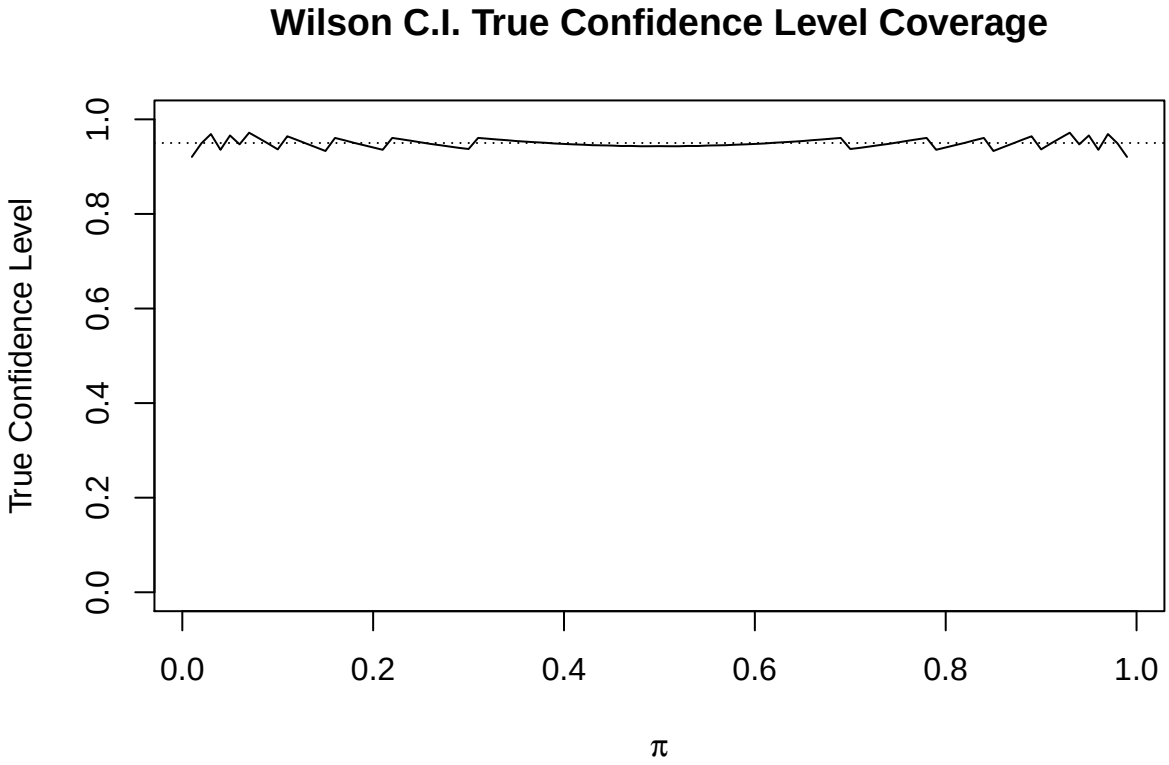
```
##  num [1:99, 1:2] 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1 ...
```



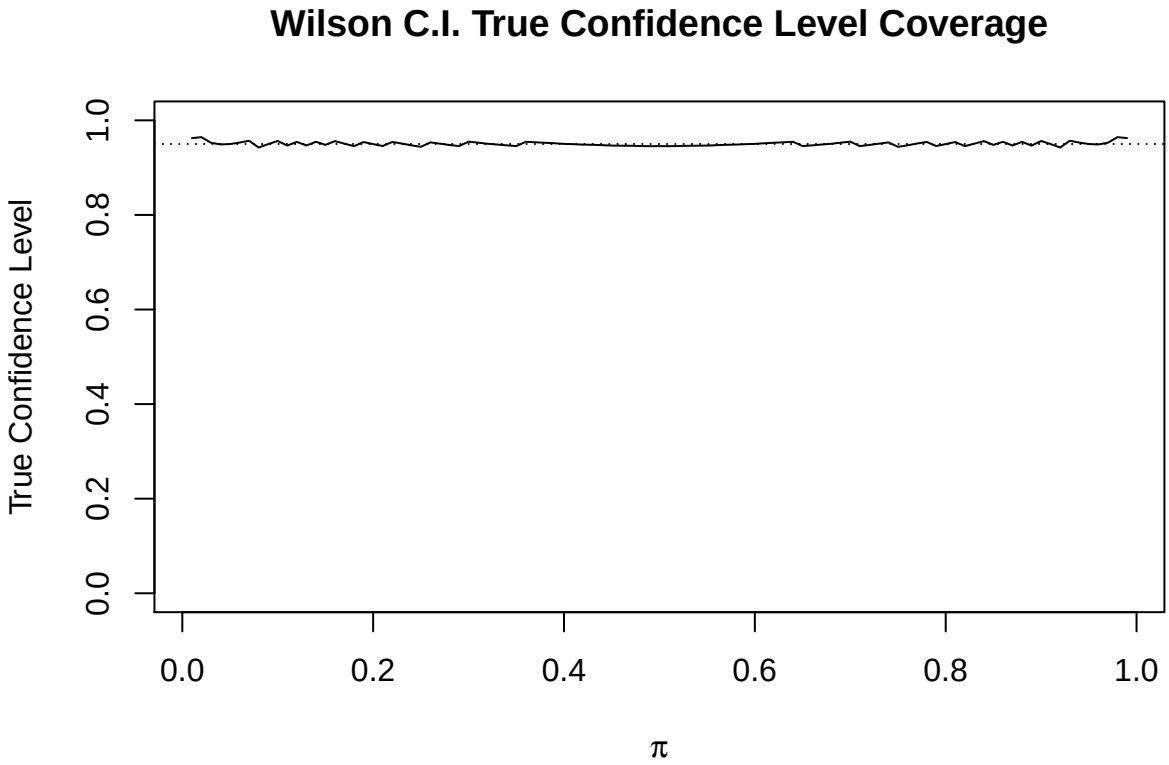

```
##  num [1:99, 1:2] 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1 ...
```




```
##  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 ...
```




```
##  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 ...
```



Note: The discussion of the Wilson confidence interval is in the book page 11 and 12.

Q1b.3. Your observation. Wilson confidence interval gives much better approximation than Wald confidence interval does, even for small n and when π is either very small or very large. As

the number of trials, n, increases, so is the Wilson confidence interval approximation. In fact, as $n = 500$, the stated confidence levels are very close to the true confidence level (i.e. $\alpha = 0.95$) even when π's are close to the two extremes.

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).

Answers:

No, we cannot say that the estimated confidence interval, such as $[-2, 2]$, continue the true parameter, θ, 95% of the time because an estiamted confidence interval with specific sample values plugged in either contain or does not contain the true parameter, which is simply an unknown constant.

From a frequentist perspective, a 95% confidence interval means that if many samples are drawn from an underlying population of interest and each of the samples is used to estimate a confidence interval, then 95% of such large number of estimated confidence intervals contain the true parameter, θ. However, any one specific estimate confidence interval either contain or does not contain the true parameter.

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?

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

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

Answer:

The odds of success is equal to $\frac{\pi}{(1-\pi)} = \frac{0.1}{0.9} = 0.11$. The odds of success is 0.11.

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

Answer:

This is also referred to as "9-to-1 against", meaning that 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.

Answer:

The odds (of success) in group 1 is 3 times as large as that of group 2.

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 $Weather$.

Exercise 8a: Background:

Exercise 17 of Chapter 1 examined data from Berry and Wood (2004) to determine if if an "icing the kicker" strategy implemented by the opposing team would reduce the probability of success for a field goal. Additional data collected for this investigation are included in the placekick.BW.csv file. Below are descriptions of the variables available in this file:

- GameNum: Identifies the year and game
- Kicker: Last name of kicker
- Good: Response variable ("Y" = success, "N" = failure)
- Distance: Length in yards of the field goal
- Weather: Levels of "Clouds", "Inside", "SnowRain", and "Sun"
- Wind15: 1 if wind speed is ≥ 15 miles per hour and the placekick is outdoors, 0 otherwise.
- Temperature: Levels of "Nice" ($40^oF <$ temperature $< 80^oF$ or inside a dome), "Cold" (temperature $\leq 40^o$ and outdoors), and "Hot" (temperature $\geq 80^o$ and outdoors)
- Grass: 1 if kicking on a grass field, 0 otherwise
- Pressure: "Y" if attempt is in the last 3 minutes of a game and a successful field goal causes a lead change, "N" otherwise
- Ice: 1 if Pressure = 1 and a time-out is called prior to the attempt, 0 otherwise

Notice that these variables are similar but not all are exactly the same as given for the placekicking data described in Section 2.2.1 (e.g., information was collected on field goals only, so there is no PAT variable).

Continuing Exercise 7, use the *Distance, Weather, Wind15, Temperature, Grass, Pressure, and Ice* explanatory variables as linear terms in a new logistic regression model and complete the following:

a: Estimate the model and properly define the indicator variables used within it.

Answer:

```
# Import libraries
library(Hmisc)
```

```
## Loading required package: lattice
```

```
## Loading required package: survival
```

```
## Loading required package: Formula
```

```
## Loading required package: ggplot2
##
## Attaching package: 'Hmisc'
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
# Set working directory setwd('~/Documents/Teach/Cal/w271/_2018.03_Fall/hw/hw01')

# Load Data
df = read.csv(file = "../placekick.BW.csv", header = TRUE, sep = ",")

# Examine the data structure
str(df)
```

```
## 'data.frame':    2003 obs. of  10 variables:
##  $ GameNum    : Factor w/ 523 levels "2002-0101","2002-0102",..: 1 1 1 1 1 1 1 2 2 2 ...
##  $ Kicker     : Factor w/ 52 levels "Akers","Andersen",..: 8 8 13 13 13 13 13 27 27 27 ...
##  $ Good       : Factor w/ 2 levels "N","Y": 2 2 1 2 1 2 2 2 2 1 ...
##  $ Distance   : int  29 33 25 23 48 33 36 34 45 48 ...
##  $ Weather    : Factor w/ 4 levels "Clouds","Inside",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ Wind15     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Temperature: Factor w/ 3 levels "Cold","Hot","Nice": 3 3 3 3 3 3 3 2 2 2 ...
##  $ Grass      : int  1 1 1 1 1 1 1 0 0 0 ...
##  $ Pressure   : Factor w/ 2 levels "N","Y": 1 1 1 1 1 1 2 1 1 1 ...
##  $ Ice        : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
describe(df)
```

```
## df
##
##  10  Variables      2003  Observations
## ---------------------------------------------------------------------------
## GameNum
##        n  missing distinct
##     2003        0      523
##
## lowest : 2002-0101 2002-0102 2002-0103 2002-0104 2002-0105
## highest: 2003-P203 2003-P204 2003-P301 2003-P302 2003-P401
## ---------------------------------------------------------------------------
## Kicker
##        n  missing distinct
##     2003        0       52
##
## lowest : Akers      Andersen   Anderson   Boyd       Brien
## highest: Stover     Tuthill    Vanderjagt Vinatieri  Wilkins
## ---------------------------------------------------------------------------
## Good
```

```
##        n  missing distinct
##     2003        0        2
##
## Value          N     Y
## Frequency    438  1565
## Proportion 0.219 0.781
## ---------------------------------------------------------------------------
## Distance
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     2003        0       43    0.999    36.35    11.11       22       23
##      .25      .50      .75      .90      .95
##       28       37       44       49       52
##
## lowest : 18 19 20 21 22, highest: 56 57 58 60 62
## ---------------------------------------------------------------------------
## Weather
##        n  missing distinct
##     2003        0        4
##
## Value        Clouds   Inside SnowRain      Sun
## Frequency       717      385      171      730
## Proportion    0.358    0.192    0.085    0.364
## ---------------------------------------------------------------------------
## Wind15
##        n  missing distinct     Info      Sum     Mean      Gmd
##     2003        0        2    0.343      264   0.1318    0.229
##
## ---------------------------------------------------------------------------
## Temperature
##        n  missing distinct
##     2003        0        3
##
## Value       Cold   Hot  Nice
## Frequency    259   198  1546
## Proportion 0.129 0.099 0.772
## ---------------------------------------------------------------------------
## Grass
##        n  missing distinct     Info      Sum     Mean      Gmd
##     2003        0        2     0.68     1308    0.653   0.4534
##
## ---------------------------------------------------------------------------
## Pressure
##        n  missing distinct
##     2003        0        2
##
## Value          N     Y
## Frequency   1864   139
## Proportion 0.931 0.069
```

```
## ------------------------------------------------------------------------
## Ice
##        n  missing distinct     Info      Sum     Mean      Gmd
##     2003        0        2    0.056       38  0.01897  0.03724
##
## ------------------------------------------------------------------------
```

```
mod.glm1 <- glm(formula = Good ~ Distance + Weather + Wind15 + Temperature + Grass +
    Pressure + Ice, family = binomial(link = logit), data = df)
summary(mod.glm1)
```

```
##
## Call:
## glm(formula = Good ~ Distance + Weather + Wind15 + Temperature +
##     Grass + Pressure + Ice, family = binomial(link = logit),
##     data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6804   0.2599   0.4360   0.7148   1.8698
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)      5.740185   0.369597  15.531   <2e-16 ***
## Distance        -0.109600   0.007188 -15.249   <2e-16 ***
## WeatherInside   -0.083030   0.214711  -0.387   0.6990
## WeatherSnowRain -0.444193   0.217852  -2.039   0.0415 *
## WeatherSun      -0.247582   0.139642  -1.773   0.0762 .
## Wind15          -0.243777   0.175527  -1.389   0.1649
## TemperatureHot   0.250013   0.247540   1.010   0.3125
## TemperatureNice  0.234932   0.181461   1.295   0.1954
## Grass           -0.328435   0.160050  -2.052   0.0402 *
## PressureY        0.270174   0.262809   1.028   0.3039
## Ice             -0.876133   0.451251  -1.942   0.0522 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2104.0  on 2002  degrees of freedom
## Residual deviance: 1791.3  on 1992  degrees of freedom
## AIC: 1813.3
##
## Number of Fisher Scoring iterations: 5
```

Note that all of the factors variables are already coded as a factor in the dataset. Therefore, they are properly encoded when included in the logistic regression. The estimate binary logistic regression is

$logit(\hat{\pi}(Good)) = 5.74 - 0.11 Distance - 0.08 WeatherInside - 0.44 WeatherSnowRain - 0.25 WeatherSun - 0.24 Wi$

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

Answer: Note that the current ordering is "Clouds", "Inside","SnowRain", "Sun". To make it the base level category, we will have to relevel this factor variable.

```
# Examine the current level in the Weather variable
levels(df$Weather)
```

```
## [1] "Clouds"   "Inside"   "SnowRain" "Sun"
```

```
# Relevel the variable using factor() function
df$Weather = factor(df$Weather, labels = c("Sun", "Clouds", "Inside", "SnowRain"))

# Re-estimate the logistic regression, calling it mod.glm1b
mod.glm1b <- glm(formula = Good ~ Distance + Weather + Wind15 + Temperature + Grass +
    Pressure + Ice, family = binomial(link = logit), data = df)
summary(mod.glm1b)
```

```
##
## Call:
## glm(formula = Good ~ Distance + Weather + Wind15 + Temperature +
##     Grass + Pressure + Ice, family = binomial(link = logit),
##     data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6804   0.2599   0.4360   0.7148   1.8698
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)      5.740185   0.369597  15.531   <2e-16 ***
## Distance        -0.109600   0.007188 -15.249   <2e-16 ***
## WeatherClouds   -0.083030   0.214711  -0.387   0.6990
## WeatherInside   -0.444193   0.217852  -2.039   0.0415 *
## WeatherSnowRain -0.247582   0.139642  -1.773   0.0762 .
## Wind15          -0.243777   0.175527  -1.389   0.1649
## TemperatureHot   0.250013   0.247540   1.010   0.3125
## TemperatureNice  0.234932   0.181461   1.295   0.1954
## Grass           -0.328435   0.160050  -2.052   0.0402 *
## PressureY        0.270174   0.262809   1.028   0.3039
## Ice             -0.876133   0.451251  -1.942   0.0522 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 2104.0  on 2002  degrees of freedom
## Residual deviance: 1791.3  on 1992  degrees of freedom
## AIC: 1813.3
##
## Number of Fisher Scoring iterations: 5
```

c: Perform LRTs for all explanatory variables to evaluate their importance within the model. Discuss the results.

Let's use our original model, *mod.glm1*, for thie exercise.

```
library(car)
```

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

```
# Conduct LRTs on all of the explanatory variables
Anova(mod.glm1, 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
```

```
# Note that the following three lines of code are not necessary for this
# exercise.

# Estimate the Profile likelihood C.I.
system.time(mod.glm1.ci <- confint(object = mod.glm1, level = 0.95))
```

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

```
##    user  system elapsed
##   0.466   0.075   0.547
```

```
# Print Profile likelihood C.I. for the original estimated coefficients
mod.glm1.ci
```

```
##                      2.5 %      97.5 %
## (Intercept)      5.0301567  6.47971272
## Distance        -0.1239459 -0.09575447
## WeatherInside   -0.5041730  0.33833063
## WeatherSnowRain -0.8675598 -0.01228988
## WeatherSun      -0.5220896  0.02563297
```

```
## Wind15          -0.5845063  0.10437447
## TemperatureHot  -0.2336594  0.73801758
## TemperatureNice -0.1251926  0.58704498
## Grass           -0.6462665 -0.01830276
## PressureY       -0.2315463  0.80227235
## Ice             -1.7591953  0.01710798
```

```
# Print Profile likelihood C.I. for the estimated odds ratios
exp(mod.glm1.ci)
```

```
##                       2.5 %      97.5 %
## (Intercept)     152.9569791 651.7836742
## Distance          0.8834276   0.9086871
## WeatherInside     0.6040049   1.4026042
## WeatherSnowRain   0.4199751   0.9877853
## WeatherSun        0.5932796   1.0259643
## Wind15            0.5573810   1.1100160
## TemperatureHot    0.7916314   2.0917846
## TemperatureNice   0.8823269   1.7986655
## Grass             0.5239985   0.9818637
## PressureY         0.7933060   2.2306039
## Ice               0.1721834   1.0172552
```

From a statistical significance perspective, the variables *Distance*, *Grass*, and *Ice* are all significant, thought *Ice* is only marginally significant.

The other p-values are all greater than 0.10, where Weather and Wind15 are somewhat closer to 0.10 than Pressure and Temperature. We can say for these four variables that there is not sufficient evidence that they affect the probability of success for a field goal when $\alpha = 0.05$.

It is important to note that each hypothesis test is conditional on the other variables remaining in the model.

d: Estimate an appropriate odds ratio for Distance, and compute the corresponding confidence interval. Interpret the odds ratio.

```
# Estimated coefficient associated with Distance
mod.glm1$coefficients[2]
```

```
##   Distance
## -0.1095996
```

```
# Estimated Odds ratio for Distance
exp(-10 * mod.glm1$coefficients[2])
```

```
## Distance
## 2.992162
```

```
# Confidence Interval for the Disatace Odds Ratios
Distance.CI = confint(object = mod.glm1, parm = "Distance", level = 0.95)
```

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

```r
exp(-10 * Distance.CI)
```

```
##    2.5 %   97.5 %
## 3.453744 2.605292
```

For a 10-yard decrease in distance, the estimated odds ratio is 2.99, meaning that a 10-yard decrease

The corresponding 95% profile LR interval is $(2.61, 3.45)$. With 95% confidence, the odds of a success change by an amount between 2.61 and 3.45 times for every 10-yard decrease in distance, holding the other variables in the model constant.