W271 Assignment 1 Solution

Contents


```
library(tidyverse)

library(sandwich)
library(lmtest)

library(car)
library(knitr)
```

1 Confidence Intervals (2 points for the whole question)

A Wald confidence interval for a binary response probability does not always have the stated confidence level, $1 - \alpha$, where α (the probability of rejecting the null hypothesis when it is true) is often set to 0.05%. The code below calculates the true confidence level of a Wald Confidence for given pi, alpha, and n.

```
# 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)
```

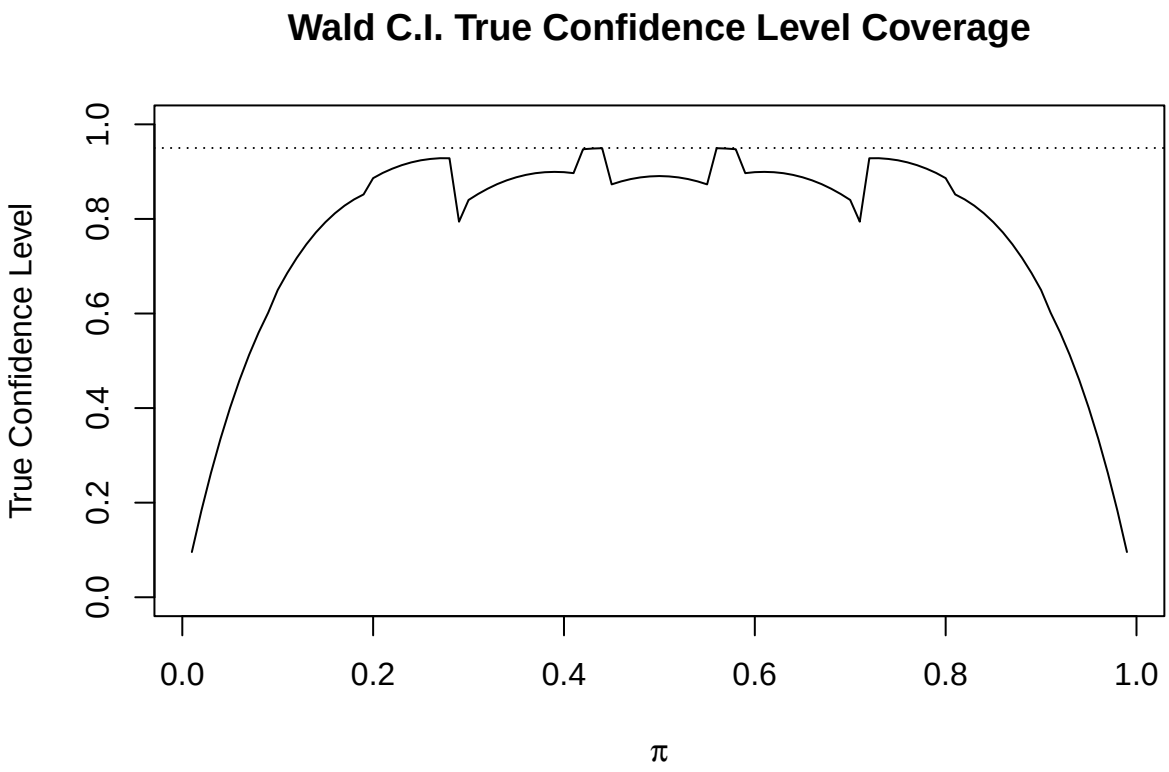
```r
# Loop through the sequence of pi's to obtain the true confidence level of
# the Wald Confidence Interval corresponding to the specific pi
counter=1
for (pi in pi.seq) {
    wald.df2 = wald.CI.true.coverage(pi=pi, alpha=0.05, n=10)
    #print(paste('True Coverage is', sum(wald.df2$covered.pi*wald.df2$pmf)))
    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 ...
```

```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")
```



1.1 Transfer learning

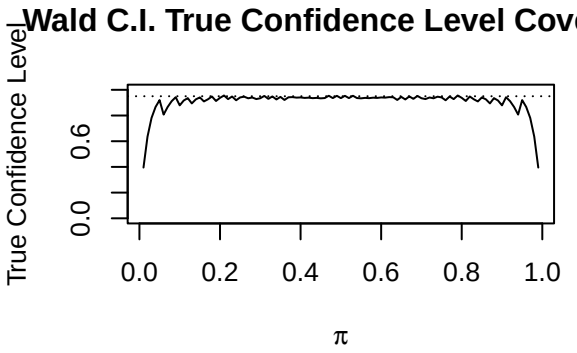
Use the code above to:

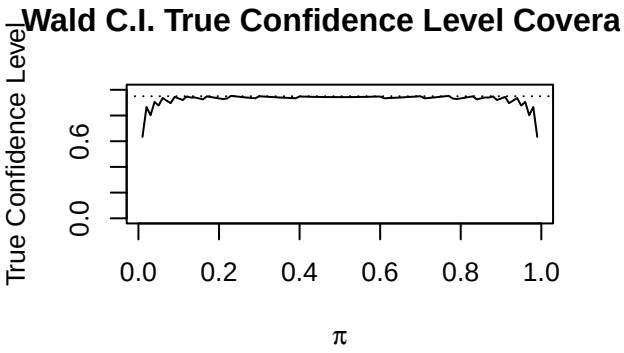
a. Redo the exercise for `n=50, n=100, n=500`;
b. Plot the graphs; and,
c. Describe what you have observed from the results. Use the same `pi.seq` as in the live session code.

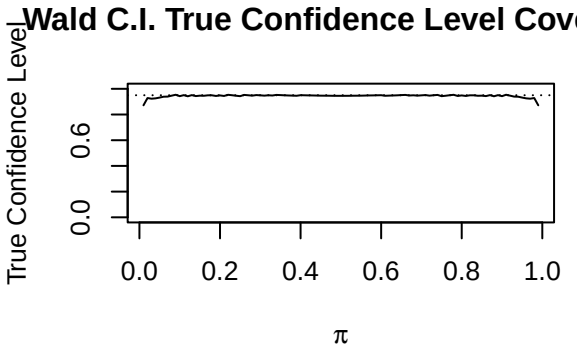
1.1.a and 1.1.b Solution

```
par(mfrow = c(2, 2))
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 ...
```







1.1.c Solution

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.

1.2 Modify the Wilson Interval

Use the code above to:

a. Modify the code for the Wilson Interval.
b. Do the exercise for `n=10, n=50, n=100, n=500`.
c. Plot the graphs.
d. 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 live session code.

1.2.a. and 1.2.b Solution

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
```

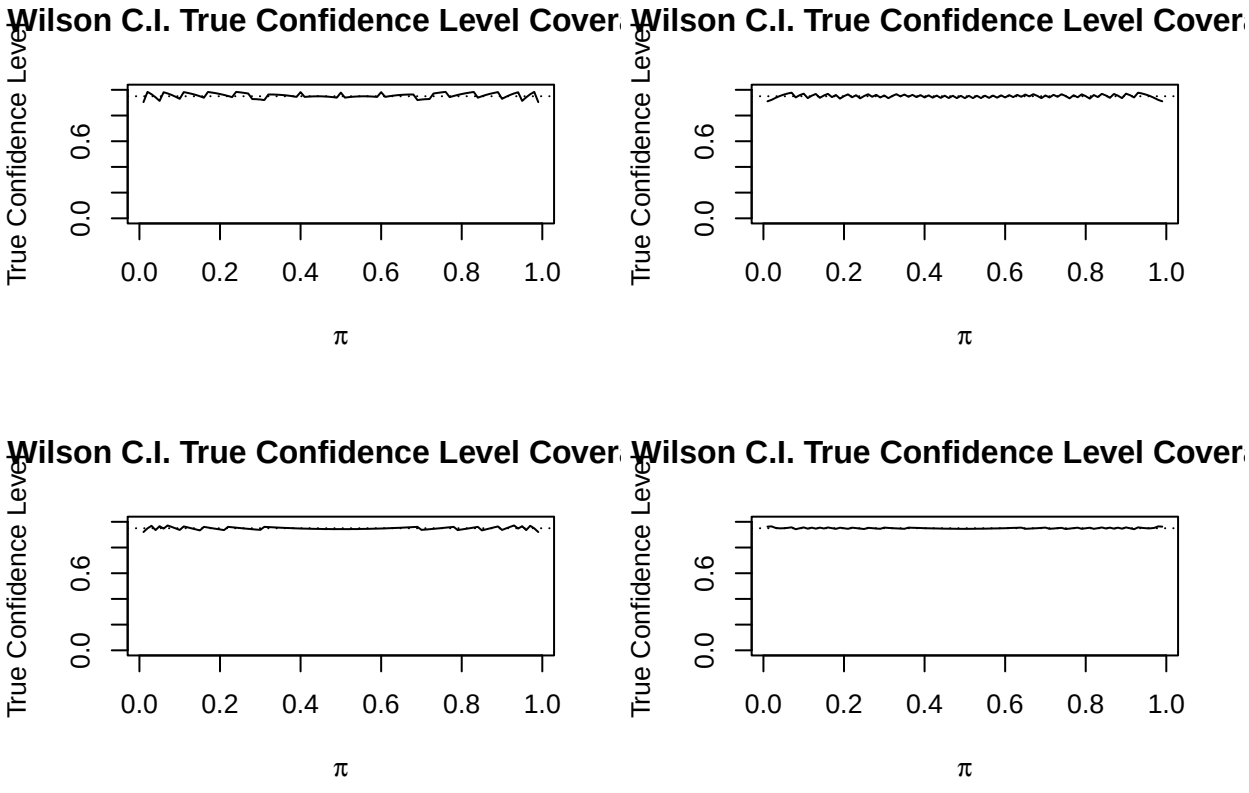
```r
#n = 10
#w = 0:n

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)
}
```

1.2.c. Solution

```r
par(mfrow = c(2, 2))
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.

1.2.d. Solution

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.

2 Maximum Likelihood (5 points for the whole question)

Let's build off of the maximum likelihood model of a binomial distribution from lecture and apply it to the wheat data set provided with the assignment.

```
wheat <- read.csv("./data/wheat.csv")
wheat$is_healthy <- wheat$type == "Healthy"
```

Suppose we want to estimate the probability of wheat being healthy as a function of density. In future lectures, we will do this using logistic regression but here we focus on maximum likelihood.

Suppose that we can express the probability of wheat being healthy as a function of density in the following form (you should recognize this as the connection between log odds and probability from the lecture):

$$P(Healthy) = P(\alpha, \beta) = \frac{e^{\alpha+\beta*Density}}{1 + e^{\alpha+\beta*Density}}$$

2.1 Write the likelihood function

Using this and assuming the number of healthy wheat in the data set follows a binomial distribution with parameters n and $p(\alpha, \beta)$, **write down the likelihood function** $L(\alpha, \beta|Data)$.

The quantity we want to estimate in a binary response model is probability of success π. In this question, probability of success or healthy kernel in not constant any more and depends on the density of wheat.

2.1. Solution

- The likelihood function is:

$$\begin{aligned} L(\pi_1, \pi_2.....\pi_n|y_1, ..., y_n) &= p(Y_1 = y_1, Y_2 = y_2...., Y_n = y_n) \\ &= p(Y_1 = y_1) * p(Y_2 = y_2)..... * p(Y_n = y_n) \\ &= \prod_{i=1}^{n} p(Y = y_i) \\ &= \prod_{i=1}^{n} (\pi_i)^{y_i}(1 - \pi_i)^{1-y_i} \end{aligned}$$

- From question prompt we know that:

$$\pi_i(\alpha, \beta) = \frac{e^{\alpha+\beta*Density}}{1 + e^{\alpha+\beta*Density}}$$

- Substitute π_i in the likelihood function:

$$\begin{aligned} &L(\alpha, \beta|y_1, ..., y_n) \\ &= \prod_{i=1}^{n} \left(\frac{e^{\alpha+\beta*Density_i}}{1 + e^{\alpha+\beta*Density_i}} \right)^{y_i} \left(1 - \frac{e^{\alpha+\beta*Density_i}}{1 + e^{\alpha+\beta*Density_i}} \right)^{(1-y_i)} \end{aligned}$$

2.2 Write and compute the log-likelihood

Find the **negative log likelihood** and write an R function to calculate it given inputs of alpha and beta and using the wheat data.

2.2. Solution

$$-Log(L(\alpha,\beta|y_1,.....,y_n) = -\sum_{i=1}^{n}\left(y_i log\left(\frac{e^{\alpha+\beta*Density_i}}{1+e^{\alpha+\beta*Density_i}}\right)+(1-y_i)log\left(1-\frac{e^{\alpha+\beta*Density_i}}{1+e^{\alpha+\beta*Density_i}}\right)\right)$$

```
n <- nrow(wheat)
y <- sum(wheat$is_healthy)
density <- wheat$density

log.lik<-function(param) {
  pi <- exp(param[1] + param[2] * density) / (1 + exp(param[1] + param[2] * density))
  log.lik <- -sum((wheat$is_healthy) * log(pi) + (1-wheat$is_healthy) * log(1 - pi))
  return(log.lik)
}
```

- We write a **negative log likelihood** in R beacause, the optim() function minimizes an R function with respect to a vector of parameters.

2.3 Compute the MLE of parameters

Use the optim function to **find the MLE of alpha and beta on the wheat data**. You can use starting values of 0 for both parameters. Note that optim by default finds the minimum, so you can use the negative log likelihood directly.

2.3. Solution

```
optim.results <- optim(c(0,0), log.lik)
optim.results
```

```
## $par
## [1] -22.90594  18.25561
##
## $value
## [1] 125.8313
##
## $counts
## function gradient
##       85       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

- Estimated model is:

$$logit(\pi) = -22.9 + 18.3 * density$$

2.4 Calculate a confidence interval

Again using the optim function, find the **variance of the MLE estimates** (hint use hessian = TRUE in optim) for alpha and beta. Calculate a **95% confidence interval** for each parameter. Are they statistically different than zero?

2.4. Solution

```
optim.results <- optim(c(0,0), log.lik, hessian = T)
optim.results
```

```
## $par
## [1] -22.90594  18.25561
##
## $value
## [1] 125.8313
##
## $counts
## function gradient
##       85       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
##
## $hessian
##          [,1]     [,2]
## [1,] 42.04745 51.98790
## [2,] 51.98790 64.42416
```

```
coef <- optim.results$par
var_cov= solve(optim.results$hessian)

alpha_lower <- coef[1] - qnorm(p = 1- alpha/2)*sqrt(var_cov[2,2])
alpha_upper <- coef[1] + qnorm(p = 1- alpha/2)*sqrt(var_cov[2,2])

beta_lower <- coef[2] - qnorm(p = 1- alpha/2)*sqrt(var_cov[1,1])
beta_upper <- coef[2] + qnorm(p = 1- alpha/2)*sqrt(var_cov[1,1])

data.frame(alpha_lower = alpha_lower, alpha_upper=alpha_upper,
           beta_lower=beta_lower, beta_upper=beta_upper )
```

```
##   alpha_lower alpha_upper beta_lower beta_upper
## 1   -28.03916   -17.77272   11.90166   24.60957
```

- We use solve() to inverse the estimated hessian matrix and find the estimated variance-covariance matrix.
- Both coefficients are statistically significant since their corresponding confidence intervals don't include zero.

2.5 Model comparison

Compare the **MLE of alpha and beta to the output of the logistic regression model code** that is provided below. What do you notice? Can you think of why this is the case? (Think about the connection between MLE of regression coefficients and linear regression)

2.5. Solution

```
mod.logit <- glm(is_healthy ~ density, data = wheat, family = binomial(link = "logit"))
summary(mod.logit)
```

```
##
## Call:
## glm(formula = is_healthy ~ density, family = binomial(link = "logit"),
##     data = wheat)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.7775  -0.7665  -0.2621   0.8179   2.1620
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -22.912      3.242  -7.066 1.59e-12 ***
## density       18.261      2.620   6.971 3.15e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 355.79  on 274  degrees of freedom
## Residual deviance: 251.66  on 273  degrees of freedom
## AIC: 255.66
##
## Number of Fisher Scoring iterations: 6
```

- The estimated coefficients are the same as the optim() results which indicates the glm() also compute the MLE of coefficients.