W271 Assignment 1

Erin Werner

October 4th, 2020

Contents

```r
library(ggplot2)
library(dplyr)
library(car)
library(readr)
library(expss)
library(nnet)
library(mcprofile)
library(gridExtra)
library(grid)
```

1 Confidence Intervals

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.

1.1 Question 1.1

Redo the Wald Confidence Interval exercise for `n=50`, `n=100`, `n=500`, plot the graphs, and describe what you have observed from the results.

I can construct a true Wald confidence level by running simulations and calculating the interval using the Wald Confidence Interval's formula.

$$\hat{\pi} \pm Z_{1-\frac{\alpha}{2}} \sqrt{\frac{\hat{\pi}(1-\hat{\pi})}{n}}$$

To start, I can initialize some functions and variables.

```r
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.b = pi.hat - qnorm(p = 1-alpha/2)*sqrt(var.wald)
  wald.CI_upper.b = pi.hat + qnorm(p = 1-alpha/2)*sqrt(var.wald)
  covered.pi = ifelse(test = pi>wald.CI_lower.b, yes = ifelse(test =
                                          pi<wald.CI_upper.b,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.b,
                                    wald.CI_upper.b),4),covered.pi)
  return(wald.df)
}
```

```r
wald_dataframe = function(n){
  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=my_alpha, n=n)
      wald.CI.true.matrix[counter,] = c(pi,sum(wald.df2$covered.pi*wald.df2$pmf))
      counter = counter+1
  }
  return(data.frame(wald.CI.true.matrix))
}
```

```
my_alpha <- 0.05
my_n10 <- 10
my_n50 <- 50
my_n100 <- 100
my_n500 <- 500
```

First, I can compute the true coverage for $\pi = 0.6$.

```
wald.df = wald.CI.true.coverage(pi=0.6, alpha=0.05, n=my_n50)
wald.CI.t.cov.lev = sum(wald.df$covered.pi*wald.df$pmf)
print(paste0("For n = ",my_n50,", True Confidence Level = ",wald.CI.t.cov.lev,"."))
```

```
## [1] "For n = 50, True Confidence Level = 0.9407."
```

```
wald.df = wald.CI.true.coverage(pi=0.6, alpha=0.05, n=my_n100)
wald.CI.t.cov.lev = sum(wald.df$covered.pi*wald.df$pmf)
print(paste0("For n = ",my_n100,", True Confidence Level = ",wald.CI.t.cov.lev,"."))
```

```
## [1] "For n = 100, True Confidence Level = 0.948."
```

```
wald.df = wald.CI.true.coverage(pi=0.6, alpha=0.05, n=my_n500)
wald.CI.t.cov.lev = sum(wald.df$covered.pi*wald.df$pmf)
print(paste0("For n = ",my_n500,", True Confidence Level = ",wald.CI.t.cov.lev,"."))
```

```
## [1] "For n = 500, True Confidence Level = 0.9502."
```

Based on the Wald true confidence level values, one can see that the true confidence level is slightly less than the theoretical 0.95 level that is expected, given different values of n. Additionally, one can see that increasing n does indeed improve the confidence value, as the confidence value incrementally approaches the theoretical level for greater values of n. As a result, for high values of n (such as n = 500), the Wald true confidence level is actually a true 95% confidence level.

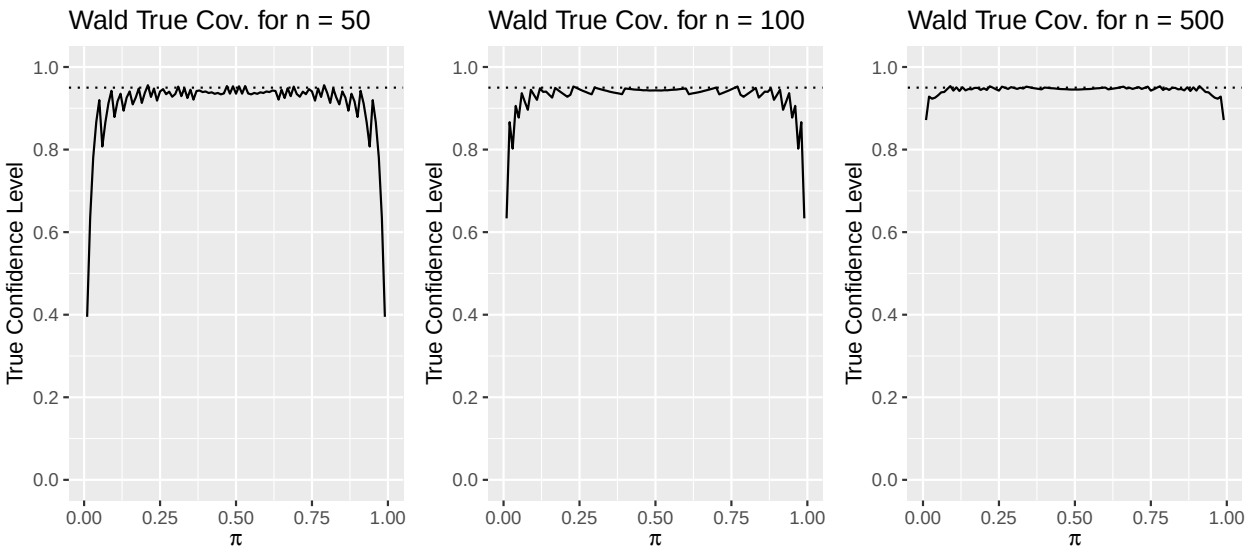
Next, I can get a more general understanding of the Wald true confidence level by computing the true coverage given a sequence of π in [0,1] and then plotting the true Wald coverage level (for given `n` and α).

```
my_wald <- wald_dataframe(my_n50)
plot50 <- ggplot(my_wald, aes(X1, X2)) + geom_line() + xlim(0,1) +
  scale_y_continuous(breaks=c(0,0.2, 0.4, 0.6, 0.8,1), limits = c(0,1)) +
  labs(title=paste0("Wald True Cov. for n = ",my_n50),
       y="True Confidence Level",x=expression(pi)) +
  geom_hline(yintercept=1-my_alpha, linetype="dotted")

my_wald <- wald_dataframe(my_n100)
plot100 <- ggplot(my_wald, aes(X1, X2)) + geom_line() + xlim(0,1) +
  scale_y_continuous(breaks=c(0,0.2, 0.4, 0.6, 0.8,1), limits = c(0,1)) +
  labs(title=paste0("Wald True Cov. for n = ",my_n100),
       y="True Confidence Level",x=expression(pi)) +
  geom_hline(yintercept=1-my_alpha, linetype="dotted")

my_wald <- wald_dataframe(my_n500)
plot500 <- ggplot(my_wald, aes(X1, X2)) + geom_line() + xlim(0,1) +
  scale_y_continuous(breaks=c(0,0.2, 0.4, 0.6, 0.8,1), limits = c(0,1)) +
  labs(title=paste0("Wald True Cov. for n = ",my_n500),
       y="True Confidence Level",x=expression(pi)) +
  geom_hline(yintercept=1-my_alpha, linetype="dotted")

egg::ggarrange(plot50, plot100, plot500, nrow = 1)
```



I can observe that the Wald interval with n = 50 tends to be somewhat far from the theoretical confidence level of 0.95, with values generally oscillating between 0.80 and 0.95. For values of π close to 0 or 1, the interval can be very liberal.

Then, I can observe that the Wald interval with n = 100 tends to be somewhat closer to the theoretical confidence level of 0.95 compared to n = 50, with values generally oscillating between 0.85 and 0.95. For values of π close to 0 or 1, the interval is still somewhat liberal.

Last, I can observe that the Wald interval with n = 500 tends to be very close to the theoretical confidence level of 0.95, with values mostly around 0.95, but more generally between 0.90 and 0.95. For values of π close to 0 or 1, the interval is more conservative.

Overall, I notice that as I increase the value of n, the true confidence level more closely represents the theoretical confidence level, with less interval oscillation and increasingly more conservative end intervals.

1.2 Question 1.2

Modify the code for the Wilson Interval and redo the confidence interval exercise for `n=10`, `n=50`, `n=100`, `n=500`, plot the graphs, describe what you have observed from the results, and compare the Wald and Wilson intervals based on your results.

I can now construct a true Wilson confidence level by running simulations and calculating the interval using the Wilson's Confidence Interval's formula.

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

To start, I can initialize some new Wilson functions.

```
wilson.CI.true.coverage = function(pi, alpha=0.05, n){
  w = 0:n
  pi.hat = w/n
  pmf = dbinom(x=w, size=n, prob=pi)
  my_Z = qnorm(p = 1-alpha/2)
  var.wilson1 = pi.hat*(1-pi.hat)
  var.wilson2 = ((my_Z)^2)/(4*n)
  var.wilson = var.wilson1 + var.wilson2
```

```
  Z_num = my_Z*sqrt(n)
  Z_den = n + (my_Z)^2
  Z_frac = Z_num/Z_den
  wil.CI_lower.b = pi.hat - Z_frac*sqrt(var.wilson)
  wil.CI_upper.b = pi.hat + Z_frac*sqrt(var.wilson)
  covered.pi = ifelse(test = pi > wil.CI_lower.b,yes = ifelse(test =
                                        pi<wil.CI_upper.b,yes=1, no=0), no=0)
  wil.CI.true.coverage = sum(covered.pi*pmf)
  wilson.df=data.frame(w, pi.hat,round(data.frame(pmf,
                                  wil.CI_lower.b,wil.CI_upper.b),4),covered.pi)
  return(wilson.df)
}
```

```
wilson_dataframe = function(n){
  pi.seq = seq(0.01,0.99, by=0.01)
  wil.CI.true.matrix = matrix(data=NA,nrow=length(pi.seq),ncol=2)
  counter=1
  for (pi in pi.seq) {
      wil.df2 = wilson.CI.true.coverage(pi=pi, alpha=my_alpha, n=n)
      wil.CI.true.matrix[counter,] = c(pi,sum(wil.df2$covered.pi*wil.df2$pmf))
      counter = counter+1
  }
  return(data.frame(wil.CI.true.matrix))
}
```

Once again, I can first compute the true coverage for $\pi = 0.6$.

```
wilson.df = wilson.CI.true.coverage(pi=0.6, alpha=0.05, n=my_n10)
wil.CI.t.cov.lev = sum(wilson.df$covered.pi*wilson.df$pmf)
print(paste0("For n = ",my_n10,", True Confidence Level = ",wil.CI.t.cov.lev,"."))
```

```
## [1] "For n = 10, True Confidence Level = 0.8989."
```

```
wilson.df = wilson.CI.true.coverage(pi=0.6, alpha=0.05, n=my_n50)
wil.CI.t.cov.lev = sum(wilson.df$covered.pi*wilson.df$pmf)
print(paste0("For n = ",my_n50,", True Confidence Level = ",wil.CI.t.cov.lev,"."))
```

```
## [1] "For n = 50, True Confidence Level = 0.9407."
```

```
wilson.df = wilson.CI.true.coverage(pi=0.6, alpha=0.05, n=my_n100)
wil.CI.t.cov.lev = sum(wilson.df$covered.pi*wilson.df$pmf)
print(paste0("For n = ",my_n100,", True Confidence Level = ",wil.CI.t.cov.lev,"."))
```

```
## [1] "For n = 100, True Confidence Level = 0.9329."
```

```
wilson.df = wilson.CI.true.coverage(pi=0.6, alpha=0.05, n=my_n500)
wil.CI.t.cov.lev = sum(wilson.df$covered.pi*wilson.df$pmf)
print(paste0("For n = ",my_n500,", True Confidence Level = ",wil.CI.t.cov.lev,"."))
```

```
## [1] "For n = 500, True Confidence Level = 0.9444."
```

Based on the Wilson true confidence level values for $\pi = 0.6$, one can see that the true confidence level is slightly less than the theoretical 0.95 level for different values of n. Yet, one can also see that increasing n doesn't always necessarily improve the given confidence value for every value of π. This varies from the observed Wald true coverage values, as n = 100 results in a slightly lower Wilson true confidence level than that of n = 50. Furthermore, the overall true confidences for values of n greater than 50 are slightly lower than the Wald true coverage values for $\pi = 0.6$. However, these results only represent a single instance of π and there is still a general trend that does approach the theoretical value of a 95% confidence level.

To get a more comprehensive understanding of the overall Wilson true confidence level, I can compute the true coverage given a sequence of π in [0,1] and then plot the true Wilson coverage level (for given `n` and α).

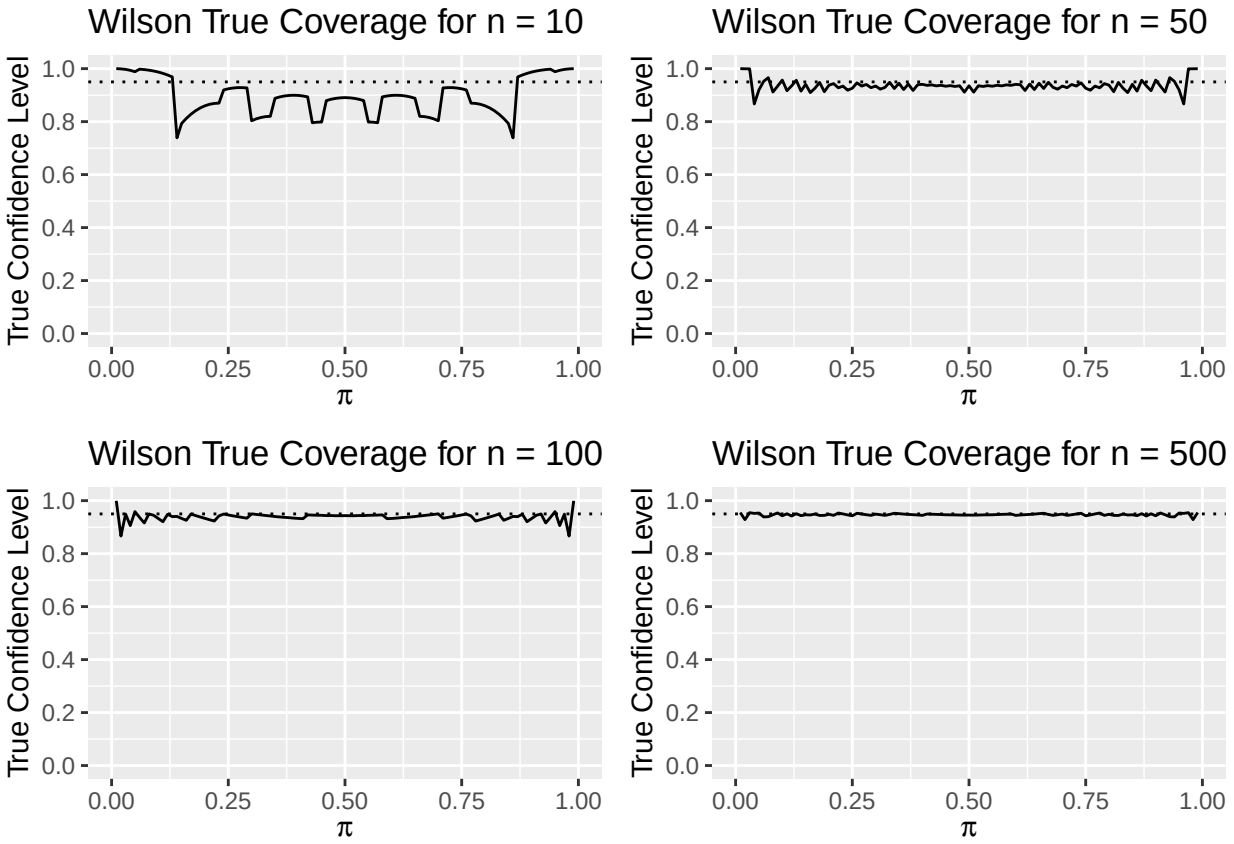
```
my_wilson  <- wilson_dataframe(my_n10)
wil10 <- ggplot(my_wilson, aes(X1, X2)) + geom_line() + xlim(0,1) +
  scale_y_continuous(breaks=c(0,0.2, 0.4, 0.6, 0.8,1), limits = c(0,1)) +
  labs(title=paste0("Wilson True Coverage for n = ",my_n10),
       y="True Confidence Level",x=expression(pi)) +
  geom_hline(yintercept=1-my_alpha, linetype="dotted")

my_wilson  <- wilson_dataframe(my_n50)
wil50 <- ggplot(my_wilson, aes(X1, X2)) + geom_line() + xlim(0,1) +
  scale_y_continuous(breaks=c(0,0.2, 0.4, 0.6, 0.8,1), limits = c(0,1)) +
  labs(title=paste0("Wilson True Coverage for n = ",my_n50),
       y="True Confidence Level",x=expression(pi)) +
  geom_hline(yintercept=1-my_alpha, linetype="dotted")

my_wilson  <- wilson_dataframe(my_n100)
wil100 <- ggplot(my_wilson, aes(X1, X2)) + geom_line() + xlim(0,1) +
  scale_y_continuous(breaks=c(0,0.2, 0.4, 0.6, 0.8,1), limits = c(0,1)) +
  labs(title=paste0("Wilson True Coverage for n = ",my_n100),
       y="True Confidence Level",x=expression(pi)) +
  geom_hline(yintercept=1-my_alpha, linetype="dotted")

my_wilson  <- wilson_dataframe(my_n500)
wil500 <- ggplot(my_wilson, aes(X1, X2)) + geom_line() + xlim(0,1) +
  scale_y_continuous(breaks=c(0,0.2, 0.4, 0.6, 0.8,1), limits = c(0,1)) +
  labs(title=paste0("Wilson True Coverage for n = ",my_n500),
       y="True Confidence Level",x=expression(pi)) +
  geom_hline(yintercept=1-my_alpha, linetype="dotted")

egg::ggarrange(wil10, wil50, wil100, wil500, nrow = 2)
```



First, I can observe that the Wilson interval with n = 10 tends to be somewhat far from the theoretical confidence level of 0.95, with values generally oscillating between 0.75 and 1.00. For values of π close to 0 or 1, the interval can be conservative, with intervals above the 0.95 confidence level. Overall, the Wilson confidence interval performs much better compared to the Wald confidence interval for n = 10, as it has a slightly smaller range of true confidence values and more conservative end intervals.

Next, I can then observe that the Wilson interval with n = 50 tends to be somewhat close to the theoretical confidence level of 0.95, with values generally oscillating between 0.85 and 0.95. For values of π close to 0 or 1, the interval can be very conservative, with intervals above the 0.95 confidence level. Overall, the Wilson confidence interval performs much better compared to the Wald confidence interval for n = 50, as it has a smaller range of true confidence values and more conservative end intervals.

Then, I can observe that the Wilson interval with n = 100 tends to be even closer to the theoretical confidence level of 0.95, with values generally between 0.90 and 0.95. For values of π close to 0 or 1, the interval can be very conservative, with intervals around the 0.95 confidence level. Overall, the Wilson confidence interval still performs much better compared to the Wald confidence interval for n = 100, as it has an even smaller range of true confidence values and conservative end intervals.

Last, I can observe that the Wilson interval with n = 500 tends to be very close to the theoretical confidence level of 0.95, with values mostly around 0.95 but generally between 0.92 and 0.95. For values of π close to 0 or 1, the interval is very conservative, with intervals around the 0.95 confidence level. Overall, the Wilson confidence interval still performs slightly better compared to the Wald confidence interval for n = 500, as it has a slightly smaller range of true confidence values and, the main difference being, more conservative end intervals.

As a result, I notice that as I increase the value of n for both Wald and Wilson confidence intervals, the true confidence level more closely represents the theoretical confidence level, with less interval oscillation and increasingly more conservative end intervals. However, I can see that the Wilson confidence interval generally does a much better job than the Wald interval.

2 Binary Logistic Regression

From the placekick.csv data set, 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 questions.

```
placekick_BW<-read_csv(paste0(here::here(),"/assignments/assignment_1/placekick.BW.csv"))
```

```
## Parsed with column specification:
## cols(
##   GameNum = col_character(),
##   Kicker = col_character(),
##   Good = col_character(),
##   Distance = col_double(),
##   Weather = col_character(),
##   Wind15 = col_double(),
##   Temperature = col_character(),
##   Grass = col_double(),
##   Pressure = col_character(),
##   Ice = col_double()
## )
```

```
head(placekick_BW)
```

```
## # A tibble: 6 x 10
##   GameNum  Kicker Good  Distance Weather Wind15 Temperature Grass Pressure   Ice
##   <chr>    <chr>  <chr>    <dbl> <chr>    <dbl> <chr>       <dbl> <chr>    <dbl>
## 1 2002-01~ Bryant Y           29 Sun          0 Nice            1 N            0
## 2 2002-01~ Bryant Y           33 Sun          0 Nice            1 N            0
## 3 2002-01~ Cortez N           25 Sun          0 Nice            1 N            0
## 4 2002-01~ Cortez Y           23 Sun          0 Nice            1 N            0
## 5 2002-01~ Cortez N           48 Sun          0 Nice            1 N            0
## 6 2002-01~ Cortez Y           33 Sun          0 Nice            1 N            0
```

2.1 Question 8.a

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

First, I need to transform the binary dependent variable, which consists of "Y" and "N" values, into zeros and ones for the model. So, for the variable `Good`, "Y" will convert to 1, indicating that the kick was good, and "N" will convert to 0, indicating that the kick was bad.

```
placekick_BW$Result <- ifelse(placekick_BW$Good == "Y", 1, 0)
```

Then, to estimate the model, I can run the regression $logit(\pi) = \beta_0 + \beta_1 Distance + \beta_2 Weather + \beta_3 Wind15 + \beta_4 Temperature + \beta_5 Grass + \beta_6 Pressure + \beta_7 Ice$ as follows.

```
mod.plk <- glm(formula=Result~Distance+Weather+Wind15+Temperature+Grass+Pressure+Ice,
               family = binomial(link = logit), data = placekick_BW)
```

```
summary(mod.plk)
```

```
##
## Call:
## glm(formula = Result ~ Distance + Weather + Wind15 + Temperature +
##     Grass + Pressure + Ice, family = binomial(link = logit),
##     data = placekick_BW)
##
```

```
## 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
```

As a result, $logit(\pi)$ = 5.7402 - 0.1096Distance - 0.0830WeatherInside - 0.4442WeatherSnowRain - 0.2476WeatherSun - 0.2438Wind15 + 0.2500TemperatureHot+ 0.2349TemperatureNice - 0.3284Grass + 0.2702PressureY - 0.8761Ice.

In general, indicator variables represent the various categorical binary 'dummy' variables. This means that for each categorical variable in the data set, all but one default category will get its own binary column to indicate the absence or presence of that categorical effect. The value of the indicator variables are equal to 1 when their corresponding condition is true and 0 otherwise. For example, in this context, `TemperatureHot` = 1 for field goals attempted at a hot temperature and 0 for non-hot temperatures. For this model, the indicator variables are `WeatherInside`, `WeatherSnowRain`, `WeatherSun`, `TemperatureHot`, `TemperatureNice`, and `PressureY`.

2.2 Question 8.b

The authors use `Sun` as the base level category for Weather, which is not the default level that R uses. Describe how `Sun` can be specified as the base level in R. Also, re-estimate the model in part (a) using `Sun` as the base level category for `Weather`.

In order to set the default level for a categorical indicator variable, in R, one can use the `relevel()` or `factor()` functions to change the base level. So in this context, one can use `factor()` to change the 'default' category for `Weather` to the sunny condition as follows.

```
placekick_BW$Weather_Sun <- factor(placekick_BW$Weather,
                                   levels = c("Sun","Clouds","SnowRain","Inside"))
```

Now, I can re-estimate the model from Part (a) with `Sun` as the base level for `Weather` by using the newly factored feature, named `Weather_Sun`, instead of `Weather`. Then, the new regression is $logit(\pi) = \beta_0 + \beta_1 Distance + \beta_2 Weather\_Sun + \beta_3 Wind15 + \beta_4 Temperature + \beta_5 Grass + \beta_6 Pressure + \beta_7 Ice$.

```
mod.plk.2<-glm(formula=Result~Distance+Weather_Sun+Wind15+Temperature+Grass+Pressure+Ice,
               family = binomial(link = logit), data = placekick_BW)
```

```
summary(mod.plk.2)
```

```
##
## Call:
## glm(formula = Result ~ Distance + Weather_Sun + Wind15 + Temperature +
##     Grass + Pressure + Ice, family = binomial(link = logit),
##     data = placekick_BW)
##
## 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)           5.492602   0.370141  14.839   <2e-16 ***
## Distance             -0.109600   0.007188 -15.249   <2e-16 ***
## Weather_SunClouds     0.247582   0.139642   1.773   0.0762 .
## Weather_SunSnowRain -0.196611   0.219015  -0.898   0.3693
## Weather_SunInside     0.164553   0.215062   0.765   0.4442
## 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
```

As a result, the updated regression is $logit(\pi)$ = 5.4926 - 0.1096Distance + 0.2476Weather_SunClouds - 0.1299Weather_SunSnowRain + 0.1646Weather_SunInside - 0.2438Wind15 + 0.2500TemperatureHot+ 0.2349TemperatureNice - 0.3284Grass + 0.2702PressureY - 0.8761Ice.

2.3 Question 8.c

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

```
Anova(mod = mod.plk, test = "LR")
```

```
## Analysis of Deviance Table (Type II tests)
##
## Response: Result
##             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
```

In this LRT, each hypothesis test is conditional on the inclusion of all the other variables in the model. Overall, most of the explanatory variables do not have statistically significant p-values for $\alpha = 0.05$. However, there are two variables that do, `Grass` and `Distance`. The p-value corresponding to `Distance` is 2e-16, which is very small and significant. This means that the variable is very important in this model. Furthermore, the p-values corresponding to `Grass` and `Ice` are both somewhat small (less than 0.06). This means that there is some evidence of their statistical importance in the model. Yet, the other p-values are all greater than 0.10, where `Pressure` and `Temperature` are much larger, as they are greater than 0.25. As a result, for these four variables, there is not sufficient evidence that they affect the probability of success for a field goal.

2.4 Question 8.d

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

The odds ratio for a change in distance by c units is expressed in the following formula:

$$OR = exp(c\beta_1)$$

which depends on the distance coefficient.

```
beta1 <- mod.plk$coefficients['Distance']
```

An important thing to consider in order to yield valuable information is the value of c. In this context, a one yard difference is not as significant compared to a 10 yard difference. So, I will set c to -10, representing a 10 yard decrease in distance. This will allow me to calculate an appropriate odds ratio for distance.

```
c10 <- -10
OR <- exp(beta1*c10)
print(paste0("For c = ",c10,", the Odds Ratio = ", round(OR,3)))
```

```
## [1] "For c = -10, the Odds Ratio = 2.992"
```

As a result, for a 10 yard decrease in distance, the estimated odds ratio is 2.99.

Next, I can compute the corresponding confidence interval

```
beta.ci <- confint(object = mod.plk, parm = "Distance", level = 0.95)
```

```
bci10de <- as.numeric(rev(exp(c10*beta.ci)))
print(paste0("For c = ",c10,", the CI is (",
             round(bci10de[1],3), ", ", round(bci10de[2],3), ")"))
```

```
## [1] "For c = -10, the CI is (2.605, 3.454)"
```

So, given a 10 yard decrease, the corresponding 95% interval is (2.61, 3.45). Thus, 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 all other variables in the model constant.

3 Binary Logistic Regression

Suppose you are hired by the University's Admission Committee and are charged to analyze this data to quantify the effect of GRE, GPA, and college rank on admission probability.

The data set `admissions.csv` contains a small sample of graduate school admission data from a university.

```
admissions <- read_csv(paste0(here::here(),"/assignments/assignment_1/admissions.csv"))
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   X1 = col_double(),
##   admit = col_double(),
##   gre = col_double(),
##   gpa = col_double(),
##   rank = col_double()
## )
```

```
head(admissions)
```

```
## # A tibble: 6 x 5
##      X1 admit   gre   gpa  rank
##   <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     1     0   380  3.61     3
## 2     2     1   660  3.67     3
## 3     3     1   800  4        1
## 4     4     1   640  3.19     4
## 5     5     0   520  2.93     4
## 6     6     1   760  3        2
```

3.1 Question 3.1

Examine the data and conduct EDA.

I first need to get an overall idea of the data I am working with.

```
summary(admissions[2:5])
```

```
##      admit             gre             gpa             rank
##  Min.   :0.0000   Min.   :220.0   Min.   :2.260   Min.   :1.000
##  1st Qu.:0.0000   1st Qu.:520.0   1st Qu.:3.130   1st Qu.:2.000
##  Median :0.0000   Median :580.0   Median :3.395   Median :2.000
##  Mean   :0.3175   Mean   :587.7   Mean   :3.390   Mean   :2.485
##  3rd Qu.:1.0000   3rd Qu.:660.0   3rd Qu.:3.670   3rd Qu.:3.000
##  Max.   :1.0000   Max.   :800.0   Max.   :4.000   Max.   :4.000
```

```
paste("Sample Size: ", nrow(admissions))
```

```
## [1] "Sample Size:  400"
```

From the printout of the Admissions data set, I see:

- It is a somewhat small data set with 400 observations and 4 columns. There does not appear to be any missing or erroneous data points.
- Column `admit` takes on two integer values, which indicate whether or not the individual was admitted. Zero indicates that the individual was not admitted and one indicates that the individual was admitted. This variable should be considered as the binomial outcome and will act as the dependent variable of

the exercise. Based on the mean of the column (0.3), one can tell that there are a greater amount of non-admitted individuals in the data set.

- Column `gre` is a numeric column, which typically consists of values between 200 and 800 with a mean of about 580 and a max of 800. This represents the individuals GRE Test Score. This will be one of the main features that I will use to build the model.
- Column `gpa` is a numeric column, which typically consists of values between 2 and 4 with a mean of about 3.4. This represents the individuals college GPA, with a maximum potential value of 4.0. This will be another one of the main features that I will use to build the model.
- Column `rank` is a numeric column yet it only takes on 4 values, thus one could also consider modeling with it being a categorical feature. Those four values are 1, 2, 3, and 4. In this context, 1 is considered to be the highest rank and 4 is the lowest on this scale. This represents the individual's rank in their college major. This column will act as an additional feature that I will consider in the model.

I will now conduct an exploratory data analysis through graphs and tables to gain a more in depth understanding of the data that I am working with in order to build the model. I added additional columns for visualization purposes.

```
admissions$admit_str <- ifelse(admissions$admit == 0, "Not Admitted", "Admitted")
```

To further understand the data, I made following tables and visualizations.

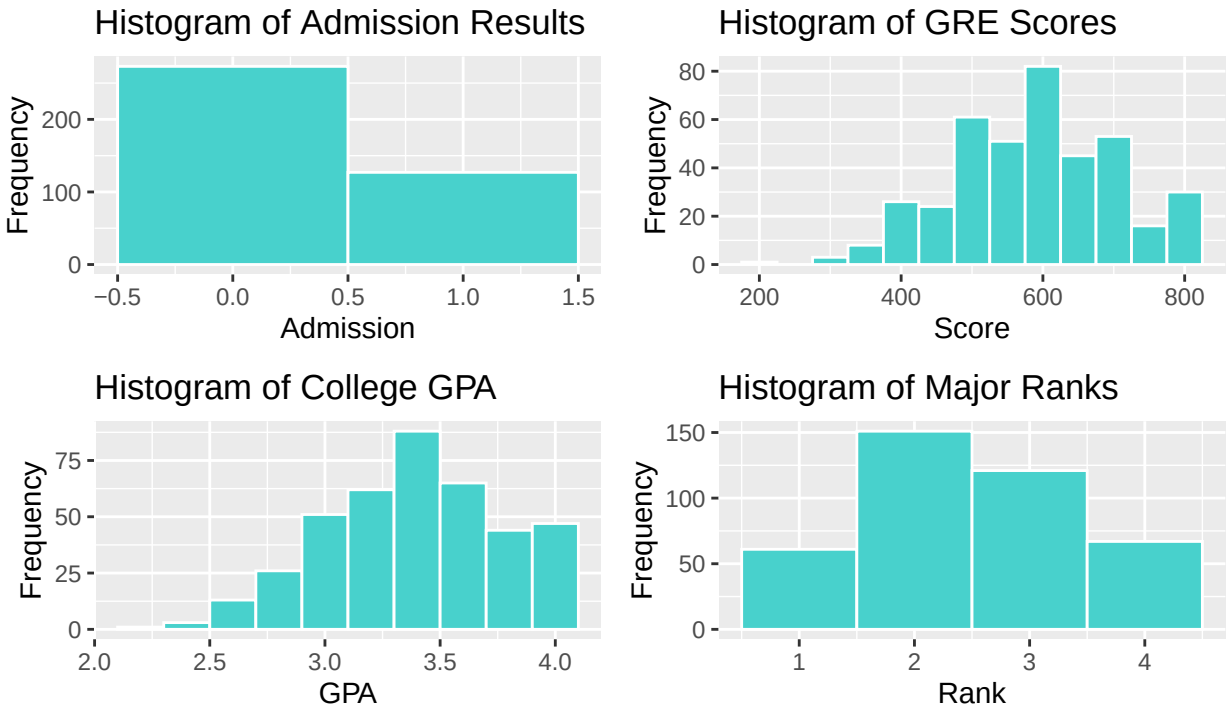
```
temp <- as.data.frame(cro(admissions$rank, admissions$admit))
colnames(temp) <- c("", "Not Admitted", "Admitted")
rownames(temp) <- c("Rank 1", "Rank 2", "Rank 3", "Rank 4", "Total")
temp$Total = coalesce(temp$`Not Admitted`, 0) + coalesce(temp$`Admitted`,0)
temp[2:4]
```

```
##        Not Admitted Admitted Total
## Rank 1           28       33    61
## Rank 2           97       54   151
## Rank 3           93       28   121
## Rank 4           55       12    67
## Total           273      127   400
```

This table shows the number of total admitted and non-admitted individuals for each college major rank. One can see that there are a majority of non-admitted individuals as well as a higher quantity of applicants with ranks of 2 and 3. An important observation is that admissions and non-admissions both occur at all rank levels. However, more individuals with Rank 1 and 2 are admitted than those of Rank 3 and 4.

Now that I have a general idea of what the data looks like, I can take a more detailed look into the individual distributions of the four key variables.

```
p <- ggplot(admissions, aes(admit)) +
  geom_histogram(binwidth = 1, fill = "mediumturquoise",col="white",size = 0.5)+
  labs(title="Histogram of Admission Results", x = "Admission",y = "Frequency")
p1 <- ggplot(admissions, aes(gre)) +
  geom_histogram(binwidth = 50, fill = "mediumturquoise",col="white",size = 0.5)+
  labs(title="Histogram of GRE Scores", x = "Score",y = "Frequency")
p2 <- ggplot(admissions, aes(gpa)) +
  geom_histogram(binwidth = 0.2, fill = "mediumturquoise",col="white",size = 0.5)+
  labs(title="Histogram of College GPA", x = "GPA",y = "Frequency")
p3 <- ggplot(admissions, aes(rank)) +
  geom_histogram(binwidth = 1, fill = "mediumturquoise",col="white",size = 0.5)+
  labs(title="Histogram of Major Ranks", x = "Rank",y = "Frequency")
egg::ggarrange(p, p1, p2, p3, nrow = 2)
```



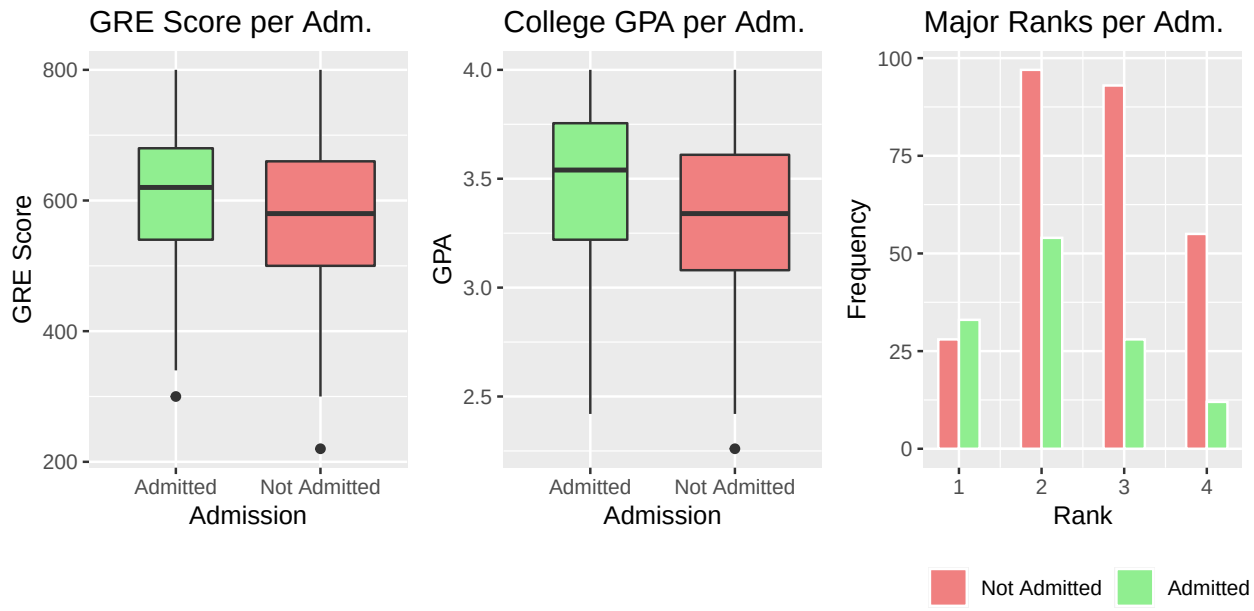
In the histograms above I see:

- Feature `admit` is a binary variable with a greater amount of zeros compared to ones. This confirms that more individuals in the data set are not admitted than admitted.
- Feature `gre` follows a mostly uni-modal, normal distribution with a slight negative skew towards higher GRE scores. The distribution peaks around the mean value of 587. There appears to be some outlying values towards the lower end of the score range, but there does not appear to be any fat tails. Yet, since I know that the sample size is 400, which is greater than 30, I can rely on asymptotic assumptions of normality for the data. This will be important for the model assumptions moving forward.
- Feature `gpa` appears to have a somewhat similar distribution to `gre`, possibly indicating that these variables could have a relationship in the model. The histogram appears to follow a uni-modal, normal distribution with a slight negative skew towards higher GPAs. The distribution peaks around the mean of 3.4. There does not appear to be any outliers or fat tails. Yet, I can also rely on asymptotic assumptions of normality.
- Feature `rank` can be considered as a categorical variable in this context, so the histogram serves to display the frequency of each category. One can see that the ranks are not uniformly distributed as most individuals fall in the second and third rank, creating an almost normal distribution. This will be important to consider for building the model.

Yet, these histograms only reveal the distribution of values, without any indication of how successful the individuals were in getting admitted. So, I can generate additional plots to help provide this insight.

```
b1 <- ggplot(admissions, aes(admit_str, gre)) + theme(legend.position="none") +
    geom_boxplot(varwidth=T, aes(fill = factor(admit))) +
    scale_fill_manual("", values = c("lightcoral", "lightgreen")) +
    labs(title="GRE Score per Adm.",x="Admission",y="GRE Score")
b2 <- ggplot(admissions, aes(admit_str, gpa)) + theme(legend.position="none") +
    geom_boxplot(varwidth=T, aes(fill = factor(admit))) +
    scale_fill_manual("", values = c("lightcoral", "lightgreen")) +
    labs(title="College GPA per Adm.",x="Admission",y="GPA")
b3 <- ggplot(admissions, aes(x=rank,group=admit,fill=factor(admit))) +
```

```
  theme(legend.position="bottom") +
  scale_fill_manual("", values = c("lightcoral", "lightgreen"),
                    labels=c("Not Admitted", "Admitted")) +
  geom_histogram(binwidth=0.5, position="dodge",col="white",size = 0.5)+
  labs(title="Major Ranks per Adm.", x = "Rank",y = "Frequency")
egg::ggarrange(b1, b2, b3, nrow = 1)
```

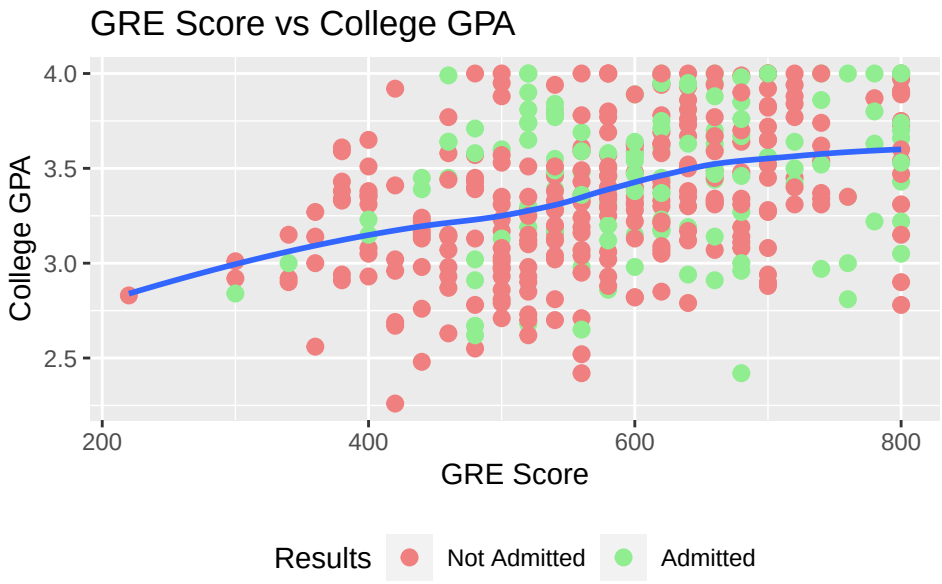


In the plots above I see:

- Feature `gre` has slightly different distributions for individuals that were admitted compared to not admitted. Individuals that were admitted tend to have higher scores, in general, than those who were not admitted. Both groups have outliers with lower scores. However, the two distributions overlap a fair amount and non-admitted individuals have a greater density of points than admitted.
- Feature `gpa`, again, appears to have a similar distribution to `gre`. The individual distributions for admitted and non-admitted are similar, but admitted individuals, on average, tend to have higher GPAs. The non-admitted group does have an outlier with a low score. Yet, these distributions do overlap and non-admitted individuals have a greater density of data points.
- Feature `rank` compares the quantity of admitted versus non-admitted individuals side-by-side with regards to college rank. The Rank 1 grouping is the only group in which the number of admitted individuals exceeds the number of non-admitted individuals. In the remaining ranks, the number of non-admitted individuals nearly doubles or triples the number of admitted individuals. This makes sense in this context, as Rank 1 is the highest rank, indicating a good potential admission.

Another way to look at a potential relationship between two features, more specifically between `gre` and `gpa`, is to focus on the scatter plot.

```
ggplot(admissions, aes(x=gre, y=gpa)) +
  geom_point(fill="gray", size = 2.5, aes(col=factor(admit))) +
  geom_smooth(method="loess", se=F) + theme(legend.position="bottom") +
  scale_colour_manual("Results",values=c("lightcoral","lightgreen"),
                      labels=c("Not Admitted", "Admitted")) +
  labs(title="GRE Score vs College GPA",y="College GPA",x="GRE Score")
```



Above, the scatter plot displays the GRE score vs. the college GPA as well as the predicted linear regression line, grouped by admission status. There appears to be a linear, positive relationship between the two different scores. This means that as an individuals GRE score increases, the GPA score is also likely to increase. However, there is no clear distinction between individuals that are admitted or not-admitted in regards to `gre` vs `gpa`.

3.2 Question 3.2

Estimate a binary logistic regression using the following set of explanatory variables: gre, gpa, $rank$, gre^2, gpa^2, **and** $gre \times gpa$**.**

First, I need to treat `rank` as a categorical variable.

```
admissions$rank <- factor(admissions$rank)
```

Then, I can run the logistic regression model $logit(\pi) = \beta_0 + \beta_1 gre + \beta_2 gpa + \beta_3 rank + \beta_4 gre^2 + \beta_5 gpa^2 + \beta_6 gre * gpa$.

```
mod.fit.admit <- glm(formula = admit ~ gre + gpa + rank + I(gre^2) + I(gpa^2) + gre:gpa,
                     family = binomial(link = logit),data = admissions)
```

```
summary(mod.fit.admit)
```

```
##
## Call:
## glm(formula = admit ~ gre + gpa + rank + I(gre^2) + I(gpa^2) +
##     gre:gpa, family = binomial(link = logit), data = admissions)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5502  -0.8754  -0.6297   1.1187   2.1888
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -7.325e+00  9.065e+00  -0.808 0.419012
## gre          1.860e-02  1.184e-02   1.571 0.116136
## gpa         -1.777e-01  4.952e+00  -0.036 0.971371
## rank2       -7.130e-01  3.202e-01  -2.227 0.025958 *
```

```
## rank3       -1.341e+00  3.474e-01  -3.861 0.000113 ***
## rank4       -1.595e+00  4.221e-01  -3.780 0.000157 ***
## I(gre^2)     3.070e-06  8.216e-06   0.374 0.708624
## I(gpa^2)     6.699e-01  7.625e-01   0.878 0.379690
## gre:gpa     -5.888e-03  3.196e-03  -1.842 0.065475 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 499.98  on 399  degrees of freedom
## Residual deviance: 454.90  on 391  degrees of freedom
## AIC: 472.9
##
## Number of Fisher Scoring iterations: 4
```

As a result, $logit(\pi)$ = -7.325e+00 + 1.860e-02`gre` - 1.777e-01`gpa` - 7.130e-01`rank2` - 1.341e+00`rank3` - 1.595e+00`rank4` + 3.070e-06`I(gre^2)` + 6.699e-01`I(gpa^2)` - 5.888e-03`gre:gpa`.

3.3 Question 3.3

Test the hypothesis that GRE has no effect on admission using the likelihood ratio test.

I can use the `Anova()` function from `car` package for the LRT. This function performs a Type II Error test, which is the non-rejection of a false null hypothesis.

All of the rows are based on the model $logit(\pi) = \beta_0 + \beta_1 gre + \beta_2 gpa + \beta_3 rank + \beta_4 gre^2 + \beta_5 gpa^2 + \beta_6 gre * gpa$, where each row gives a test of $H_0 : \beta_i = 0$ vs. $H_1 : \beta_i \neq 0$ for the respective variables.

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

```
## Analysis of Deviance Table (Type II tests)
##
## Response: admit
##          LR Chisq Df Pr(>Chisq)
## gre        0.3687  1    0.54373
## gpa        0.0238  1    0.87749
## rank      21.8244  3  7.095e-05 ***
## I(gre^2)   0.1383  1    0.70994
## I(gpa^2)   0.7620  1    0.38269
## gre:gpa    3.4119  1    0.06473 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The LRT test points to the results that `rank` is the only variable that has a significant p-value for $\alpha = 0.05$. This indicates that while college rank is important in explaining the admission status, the remaining variables are not. As a result, with regards to the GRE, I do not reject the null hypothesis. Therefore, I can assume that the GRE has no effect on admission.

3.4 Question 3.4

What is the estimated effect of college GPA on admission?

The odds ratio for an increase in GPA by c units is expressed in the following formula:

$$OR = exp(c\beta_2 + c\beta_4(2 \times gpa + c) + c\beta_6 gre)$$

which depends on both the GPA and the GRE score.

```
c <- 0.1 #an appropriate c-unit change for GPA
my_gre <- 720
my_gpa <- 3.3
gpa_OR <- c * exp(mod.fit.admit$coefficients['gpa'] +
                  mod.fit.admit$coefficients['I(gpa^2)']*(2*my_gpa + c) +
                  mod.fit.admit$coefficients['gre:gpa']*my_gre)
gpa_OR <- round(gpa_OR,4)
print(paste0("Estimated effect of college GPA = ", gpa_OR, " for c = ", c,
             ", GRE = ", my_gre, ", and GPA = ", my_gpa,"."))
```

```
## [1] "Estimated effect of college GPA = 0.1074 for c = 0.1, GRE = 720, and GPA = 3.3."
```

The odds of success change by $e^{c(\beta_2+\beta_4*(2*gpa+c)+\beta_g*gre)} = 0.1074$ times for a 0.1 unit increase in GPA when GPA is at a value of 3.3 with a GRE score of 720.

I can also make a table for different GRE and GPA values to see how the estimated effect of GPA changes, given a constant `c = 0.1` unit increase in GPA.

```
gre_scores <- c(200,300,400,500,550,600,650,700,800)
gpa_values <- c(1.0,2.0,2.5,2.8,3.0,3.3,3.5,3.8,4.0)
gpa_effect <- c()
for(the_gre in gre_scores){
  my_row <- c()
  for(the_gpa in gpa_values){
    gpa_OR_temp <- c * exp(mod.fit.admit$coefficients['gpa'] +
                           mod.fit.admit$coefficients['I(gpa^2)']*(2*the_gpa + c) +
                           mod.fit.admit$coefficients['gre:gpa']*the_gre)
    gpa_OR_temp <- round(gpa_OR_temp,3)
    my_row <- c(my_row,gpa_OR_temp)
  }
  gpa_effect <- rbind(gpa_effect,my_row)
}
gpa_impact <- data.frame(gpa_effect)
colnames(gpa_impact) <- c("GPA 1.0","GPA 2.0","GPA 2.5","GPA 2.8","GPA 3.0",
                          "GPA 3.3","GPA 3.5","GPA 3.8","GPA 4.0")
rownames(gpa_impact) <- c("GRE 200","GRE 300","GRE 400","GRE 500","GRE 550",
                          "GRE 600","GRE 650","GRE 700","GRE 800")
gpa_impact
```

```
##         GPA 1.0 GPA 2.0 GPA 2.5 GPA 2.8 GPA 3.0 GPA 3.3 GPA 3.5 GPA 3.8 GPA 4.0
## GRE 200   0.105   0.402   0.785   1.174   1.535   2.294   2.998   4.482   5.859
## GRE 300   0.058   0.223   0.436   0.651   0.852   1.273   1.664   2.487   3.252
## GRE 400   0.032   0.124   0.242   0.362   0.473   0.706   0.924   1.380   1.805
## GRE 500   0.018   0.069   0.134   0.201   0.262   0.392   0.513   0.766   1.002
## GRE 550   0.013   0.051   0.100   0.149   0.195   0.292   0.382   0.571   0.746
## GRE 600   0.010   0.038   0.075   0.111   0.146   0.218   0.284   0.425   0.556
## GRE 650   0.007   0.028   0.056   0.083   0.108   0.162   0.212   0.317   0.414
## GRE 700   0.006   0.021   0.041   0.062   0.081   0.121   0.158   0.236   0.309
## GRE 800   0.003   0.012   0.023   0.034   0.045   0.067   0.088   0.131   0.171
```

Now, the odds of success change by $e^{c(\beta_2+\beta_4*(2*gpa+c)+\beta_g*gre)}$ times for a 0.1 unit increase in GPA when GPA and GRE are set according to the rows and columns in the table. For example, the odds of success change by 0.785 times for a 0.1 unit increase in GPA when GPA is at a value of 2.5 with a GRE score of 200. Overall, I can see that a lower GRE score and a higher GPA value result in a high estimated effect for a 0.1 unit increase in GPA.

3.5 Question 3.5

Construct the confidence interval for the admission probability for the students with $GPA = 3.3$, $GRE = 720$, and $rank = 1$.

To start, I can initialize a general confidence interval calculation function.

```
ci.pi <- function(newdata, mod.fit.obj, alpha){
  linear.pred<-predict(object=mod.fit.obj,newdata=newdata,type="link",se=TRUE)
  CI.lin.pred.lower <- linear.pred$fit - qnorm(p = 1-alpha/2)*linear.pred$se
  CI.lin.pred.upper <- linear.pred$fit + qnorm(p = 1-alpha/2)*linear.pred$se
  CI.pi.lower <- exp(CI.lin.pred.lower)/(1 + exp(CI.lin.pred.lower))
  CI.pi.upper <- exp(CI.lin.pred.upper)/(1 + exp(CI.lin.pred.upper))
  list(lower = CI.pi.lower, upper = CI.pi.upper)
}
```

When the GRE Score is 720, the GPA is 3.3, and the college major rank is 1, the estimated probability of admission is

$$\pi = \frac{\exp(\hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \hat{\beta}_3 x_3 + \hat{\beta}_4 x_1^2 + \hat{\beta}_5 x_2^2 + \hat{\beta}_6 x_1 * x_2)}{1 + \exp(\hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \hat{\beta}_3 x_3 + \hat{\beta}_4 x_1^2 + \hat{\beta}_5 x_2^2 + \hat{\beta}_6 x_1 * x_2)}$$

with a corresponding confidence interval, which is calculated as follows:

```
newdata <- with(admissions, data.frame(gre = 720, gpa = 3.3, rank = factor(1)))
pred_prob <- round(predict(mod.fit.admit, newdata, type="response"),4)
print(paste0("Estimated Probability: ",pred_prob))
```

```
## [1] "Estimated Probability: 0.5935"
```

```
pred_low <- round(ci.pi(newdata,mod.fit.obj=mod.fit.admit,alpha=0.05)$lower,4)
pred_high <- round(ci.pi(newdata,mod.fit.obj=mod.fit.admit,alpha=0.05)$upper,4)
print(paste0("Lower CI: ",pred_low))
```

```
## [1] "Lower CI: 0.4345"
```

```
print(paste0("Upper CI: ",pred_high))
```

```
## [1] "Upper CI: 0.7351"
```

As a result, $\hat{\pi} = 0.5935$ and the 95% confidence interval is $0.4345 < \hat{\pi} < 0.7351$.

4 Binary Logistic Regression

Load the `Mroz` data set that comes with the *car* library.

```
head(Mroz)
```

```
##   lfp k5 k618 age  wc hc       lwg    inc
## 1 yes  1    0  32  no no 1.2101647 10.910
## 2 yes  0    2  30  no no 0.3285041 19.500
## 3 yes  1    3  35  no no 1.5141279 12.040
## 4 yes  0    3  34  no no 0.0921151  6.800
## 5 yes  1    2  31 yes no 1.5242802 20.100
## 6 yes  0    0  54  no no 1.5564855  9.859
```

4.1 Question 4.1

Estimate a linear probability model. Interpret the model results. Conduct model diagnostics. Test the CLM model assumptions.

First, I can run the linear regression model: $\beta_0 + \beta_1 k5 + \beta_2 k618 + \beta_3 age + \beta_4 wc + \beta_5 hc + \beta_6 lwg + \beta_7 inc$.

```
mroz.lm <- lm(as.numeric(lfp) ~ k5 + k618 + age + wc + hc + lwg + inc, data = Mroz)
summary(mroz.lm)
```

```
##
## Call:
## lm(formula = as.numeric(lfp) ~ k5 + k618 + age + wc + hc + lwg +
##     inc, data = Mroz)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.9268 -0.4632  0.1684  0.3906  0.9602
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.143548   0.127053  16.871  < 2e-16 ***
## k5          -0.294836   0.035903  -8.212 9.58e-16 ***
## k618        -0.011215   0.013963  -0.803 0.422109
## age         -0.012741   0.002538  -5.021 6.45e-07 ***
## wcyes        0.163679   0.045828   3.572 0.000378 ***
## hcyes        0.018951   0.042533   0.446 0.656044
## lwg          0.122740   0.030191   4.065 5.31e-05 ***
## inc         -0.006760   0.001571  -4.304 1.90e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.459 on 745 degrees of freedom
## Multiple R-squared:  0.1503, Adjusted R-squared:  0.1423
## F-statistic: 18.83 on 7 and 745 DF,  p-value: < 2.2e-16
```

As a result, the regression is: 2.144 - 0.295`k5` - 0.011`k618` - 0.013`age` + 0.126`wcyes` + 0.019`hcyes` + 0.123`lwg` - 0.007`inc`. The linear regression fitted a slightly downward sloping line between the probability of women in the work force and the explanatory variables. The intercept is 2.1435 and each coefficient represents a probability. For example, every additional year of age decreases the probability that the woman is in the labor force by about 1.3%. Many coefficients are statistically significant at 0.001 level, while the rest are not statistically significant for $\alpha = 0.05$. These significant coefficients are `(Intercept)`, `k5`, `age`, `wcyes`, `lwg`, and `inc`.

Now that I have interpreted the model results, I can conduct model diagnostics and test the 6 classical linear model (CLM) assumptions.

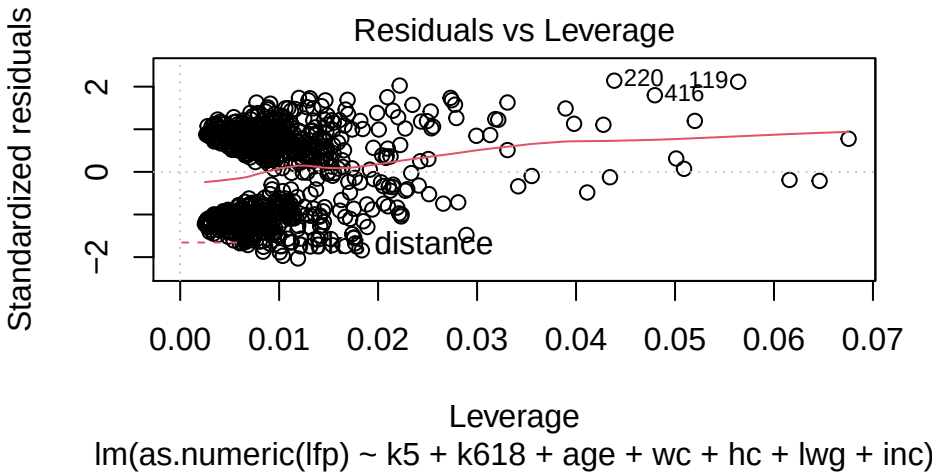
These assumptions are:

- Linearity
- Random Sampling
- No Perfect Collinearity
- Zero Conditional Mean
- Homoskedasticity
- Normality of Residuals

```
mroz.lm.res <- resid(mroz.lm)
Mroz$residuals <- mroz.lm.res
Mroz$index <- seq(1,nrow(Mroz))
Mroz$predicted <- predict(mroz.lm)
paste("Sample Size: ", nrow(Mroz))
```

```
## [1] "Sample Size:  753"
```

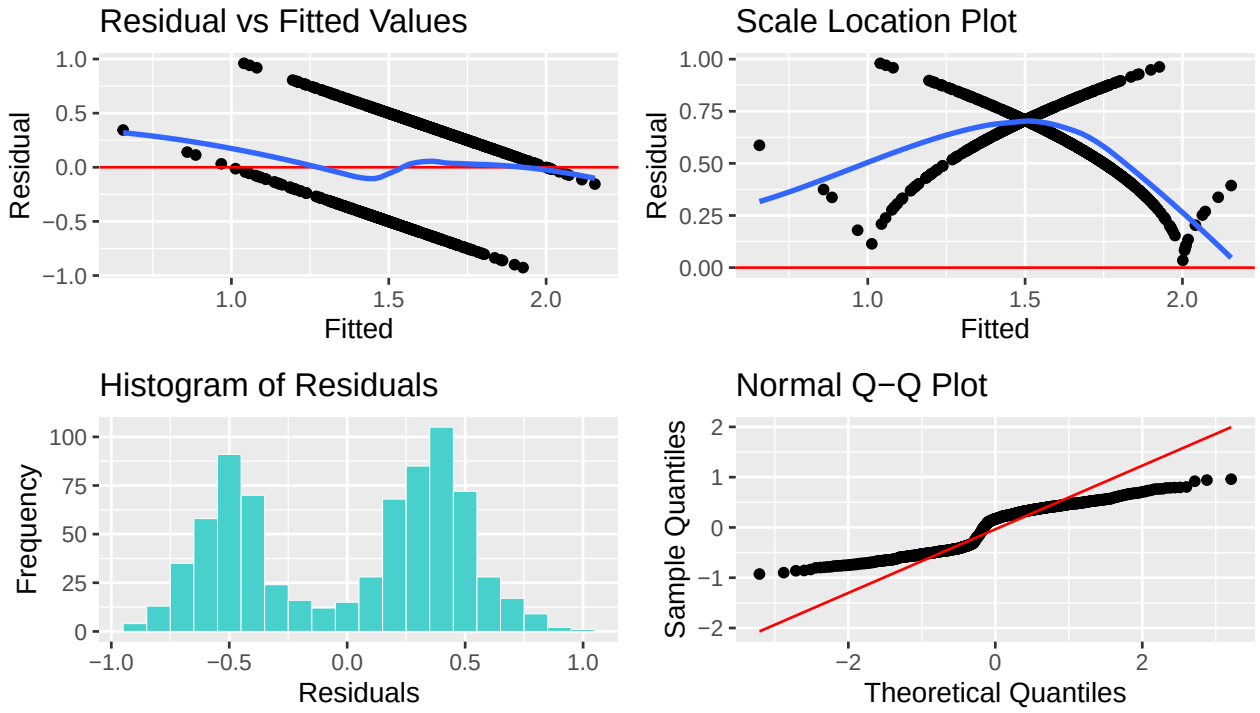
```
plot(mroz.lm, which = 5)
```



Here, the Residuals vs Leverage plot is used to identify points that are outliers and have a high influence on the model. Points that increase along the x-axis are increasing in leverage, which means that they have the most potential to affect coefficients. Yet, leverage is not the same as influence. I am mostly concerned with points that have a large Cook's distance. They could be extreme cases against the regression line and can alter the results if excluded from analysis. I can see that there are a few points that have a large Cook's distance, but they do not appear to be erroneous, so they can remain in the model.

```
resid <- ggplot(Mroz, aes(x=predicted, y=residuals)) + geom_point() +
  geom_hline(yintercept = 0, color = "red") + geom_smooth(method="loess", se=F)+
  labs(y="Residual",x="Fitted", title="Residual vs Fitted Values")
scale <- ggplot(Mroz, aes(x=predicted, y=sqrt(abs(residuals)))) + geom_point() +
  geom_hline(yintercept = 0, color = "red") + geom_smooth(method="loess", se=F)+
  labs(y="Residual", x="Fitted", title="Scale Location Plot")
resHist <- ggplot(Mroz, aes(residuals)) +
  geom_histogram(binwidth = 0.1, fill = "mediumturquoise",col="white",size=0.1)+
  labs(title="Histogram of Residuals",x = "Residuals", y = "Frequency")
my_qq <- ggplot(Mroz,aes(sample=residuals))+stat_qq()+stat_qq_line(color="red")+
```

```
  labs(title="Normal Q-Q Plot",x="Theoretical Quantiles",y="Sample Quantiles")
egg::ggarrange(resid, scale, resHist, my_qq, nrow = 2)
```



Linearity means that the model relies on variables that have a linear relationship, plus some amount of error. However, I can not confirm linearity in this model due to the Residuals vs Fitted plot, as there is indeed an apparent pattern among the data points. Also, the spline curve is not flat and fluctuates around the zero-line. These results mean that the model is not linear and does not satisfy the CLM Assumption of Linearity.

Random sampling is important as the data needs to follow the population distribution and be independent and identically distributed (iid). I know that the data maintains this assumption because of where the data came from and how it was collected. The observations, from the Panel Study of Income Dynamics (PSID), are married women. As these married women were randomly selected, the model does satisfy CLM Assumption of Random Sampling.

No perfect collinearity requires that no independent variables are constant and that there are no exact relationships among them. Multicollinearity can be assessed by examining tolerance and the VIF. Tolerance is a measure of collinearity represented by (1 - R^2). The VIF is 1/Tolerance and it's always greater than or equal to 1. Values of VIF that exceed 10 are often regarded as indicating multicollinearity.

```
vif(mroz.lm)
```

```
##       k5     k618      age       wc       hc      lwg      inc
## 1.263074 1.212232 1.497939 1.518339 1.540656 1.123176 1.192132
```

Each of the VIFs are small (less than 2), which indicates that there is no multicollinearity in the model. As a result, it's fair to claim that the CLM Assumption of No Perfect Collinearity is satisfied.

Zero conditional mean means that the value of explanatory variables contains no information about the mean of the unobserved factors, which is represented by $E(u_i|x_{i1}, x_{i2}, ..., x_{ik}) = 0$. This then enforces linearity. Once again, I will take a look at the Residuals vs Fitted plot. Here, I can see that the assumption of zero conditional mean is not satisfied. The data is not centered around the zero-line. The mean does change drastically from left to right. So, values are extremely different for different values of x. Therefore, the model does not satisfy the CLM Assumption of Zero Conditional Mean.

Homoskedasticity indicates that the variance of the error terms is constant. This is represented by $Var(u_i|x_{i1}, x_{i2}, ..., x_{ik}) = \sigma^2$. Referring back to the Residual vs Fitted plot, there is no uniform thickness of points around the zero-line. This means that the variance of errors is not constant, which does not satisfy the assumption of homoskedasticity. Additionally, in the Scale Location plot, I can see that the band of residuals is not remotely horizontal. This reinforces the violation of homoskedasticity. Homoskedasticity can also be tested statistically with the Breusch-Pagan test.

```
lmtest::bptest(mroz.lm)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  mroz.lm
## BP = 97.603, df = 7, p-value < 2.2e-16
```

As a result, the test reveals a p-value that is less than alpha = 0.05, meaning that it's statistically significant. So, it's fair to reject the null hypothesis of homoskedasticity, indicating that there is actually heteroskedasticity in the model. Thus, the model also does not satisfy the CLM Assumption of Homoskedasticity.

The last assumption for CLM is that the errors are normally distributed, represented by $u_i = N(0, \sigma^2)$. Although the distribution can be considered be normal for large sample sizes by a version of the Central Limit Theorem, it's still important to check this condition. This strong assumption indicates that the errors are independent of the x's. The histogram of residuals reveals that the errors are not remotely normally distributed, revealing that the sixth assumption of the CLM is actually untrue despite asymptotic assumptions. The histogram is bi-modal, with peaks on either end of distribution. This is further reinforced by the Q-Q plot as it shows that most of the data points do not fall on the diagonal, making it a good indication of non-normality. As a result, the model does not satisfy the CLM Assumption of Normality of Residuals.

As a result, a linear regression model is not necessarily suitable for this data set. The model does not satisfy four out of the six CLM Assumptions that are necessary for a proper linear regression, thus suggesting that a logistic regression model might perform better.

4.2 Question 4.2

Estimate a binary logistic regression.

First, I will have to create two new variables for the model, `age_squared` and `totalKids`.

```
Mroz$age_squared <- (Mroz$age)^2
Mroz$totalKids <- Mroz$k5 + Mroz$k618
```

Now, I can run the logistic regression model $logit(\pi) = \beta_0 + \beta_a ge + \beta_2 inc + \beta_3 wc + \beta_4 hc + \beta_5 lwg + \beta_6 totalKids + \beta_7 age\_squared$.

```
mroz.glm <- glm(lfp ~ age + inc + wc + hc + lwg + totalKids + age_squared,
                family = binomial, data = Mroz)
summary(mroz.glm)
```

```
##
## Call:
## glm(formula = lfp ~ age + inc + wc + hc + lwg + totalKids + age_squared,
##     family = binomial, data = Mroz)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8342  -1.1669   0.6773   1.0079   2.0614
##
## Coefficients:
```

```
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.294073   2.281551  -2.320 0.020320 *
## age          0.318014   0.109463   2.905 0.003670 **
## inc         -0.034561   0.007922  -4.363 1.28e-05 ***
## wcyes        0.666013   0.218074   3.054 0.002258 **
## hcyes        0.098260   0.198970   0.494 0.621417
## lwg          0.549976   0.145506   3.780 0.000157 ***
## totalKids   -0.222490   0.063849  -3.485 0.000493 ***
## age_squared -0.004114   0.001272  -3.233 0.001224 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1029.75  on 752  degrees of freedom
## Residual deviance:  952.02  on 745  degrees of freedom
## AIC: 968.02
##
## Number of Fisher Scoring iterations: 4
```

As a result, $logit(\pi)$ = -5.2941 + 0.3180age - 0.0346inc + 0.6660wcyes + 0.0983hcyes + 0.5500lwg - 0.2225totalKids - 0.0041age_squared.

4.3 Question 4.3

Is the age effect statistically significant?

```
Anova(mroz.glm)
```

```
## Analysis of Deviance Table (Type II tests)
##
## Response: lfp
##             LR Chisq Df Pr(>Chisq)
## age           8.6144  1  0.0033351 **
## inc          21.0740  1  4.419e-06 ***
## wc            9.5398  1  0.0020107 **
## hc            0.2439  1  0.6213914
## lwg          15.0213  1  0.0001063 ***
## totalKids    12.4267  1  0.0004232 ***
## age_squared  10.7487  1  0.0010435 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The LRT test points to the results that age does has a significant p-value for $\alpha = 0.05$. Additionally, age_squared is also statistically significant. This indicates that both age and the quadratic result of age are important in explaining the odds of labor force participation for women. As a result, I can reject the null hypothesis that there is no effect from age on the model as age is statistically significant.

4.4 Question 4.4

What is the effect of a decrease in age by 5 years on the odds of labor force participation for a female who was 45 years of age?

The odds ratio for a change in age by c years is expressed in the following formula:

$$OR = exp(c\beta_1 + c\beta_7(2 \times age + c))$$

which depends on the woman's age.

```
c <- -5
my_age <- 45
age_OR <- c * exp(mroz.glm$coefficients['age'] +
                  mroz.glm$coefficients['age_squared']*(2*my_age + c))
age_OR <- round(age_OR,4)
my_change <- ifelse(c < 0, "decrease", "increase")
cat("The estimated effect on the odds of larbor force participation is ",age_OR,"\n",
   " given a ",abs(c)," year ",my_change," in age for a ",my_age," year old woman.")
```

```
## The estimated effect on the odds of larbor force participation is  -4.8441
##   given a  5  year  decrease  in age for a  45  year old woman.
```

The odds of success change by $e^{c(\beta_1+\beta_7*(2*age+c))}$ = -4.8441 times for a 5 year decrease in age when a woman is 45 years old. As a result, the effect of a decrease in age by 5 years on the odds of labor force participation for a female who was 45 years of age is -4.8841.

4.5 Question 4.5

Estimate the profile likelihood confidence interval of the probability of labor force participation for females who were 40 years old, had income equal to 20, did not attend college, had log wage equal to 1, and did not have children.

First, I can calculate the estimated probability for the given input values. I will also set the variable `hc` to "yes", indicating that the woman's husband did attend college.

```
newdata <- with(Mroz,data.frame(age = 40, inc = 20, wc = "no", lwg = 1, hc = "yes",
                                totalKids = 0, age_squared = (40)^2))
pred_prob <- round(predict(mroz.glm, newdata, type="response"),4)
print(paste0("Estimated Probability: ",pred_prob))
```

```
## [1] "Estimated Probability: 0.6902"
```

Now, I can calculate the corresponding profile likelihood confidence interval.

In this context, "no" will be represented as 0 and "yes" will be represented by 1.

```
K <- matrix(data = c(1, 40, 20, 0, 1, 1, 0, (40)^2), nrow = 1, ncol = 8)
linear.combo <- mcprofile(object = mroz.glm, CM = K)
ci.logit.profile <- confint(object = linear.combo, level = 0.95)
exp(ci.logit.profile$confint)/(1 + exp(ci.logit.profile$confint))
```

```
##       lower     upper
## 1 0.5857033 0.7795871
```

As a result, $\hat{\pi} = 0.6902$ and the 95% profile likelihood confidence interval is $0.5857 < \hat{\pi} < 0.7796$.

5 Maximum Likelihood

Consider a model to estimate the kernel condition using the density explanatory variable as a linear term.

```
wheat <- read_csv(paste0(here::here(),"/assignments/assignment_1/wheat.csv"))
```

```
## Parsed with column specification:
## cols(
##   index = col_double(),
##   class = col_character(),
##   density = col_double(),
##   hardness = col_double(),
##   size = col_double(),
##   weight = col_double(),
##   moisture = col_double(),
##   type = col_character()
## )
```

I first need to get an overall idea of the data I am working with.

```
head(wheat)
```

```
## # A tibble: 6 x 8
##   index class density hardness  size weight moisture type
##   <dbl> <chr>   <dbl>    <dbl> <dbl>  <dbl>    <dbl> <chr>
## 1     1 hrw      1.35     60.3  2.30   24.6     12.0 Healthy
## 2     2 hrw      1.29     56.1  2.73   33.3     12.2 Healthy
## 3     3 hrw      1.23     44.0  2.51   31.8     11.9 Healthy
## 4     4 hrw      1.34     53.8  2.27   32.7     12.1 Healthy
## 5     5 hrw      1.26     44.4  2.35   26.1     12.1 Healthy
## 6     6 hrw      1.30     48.1  2.49   33.3     12.2 Healthy
```

```
summary(wheat[2:8])
```

```
##     class              density          hardness           size
##  Length:275         Min.   :0.7352   Min.   :-44.080   Min.   :0.5973
##  Class :character   1st Qu.:1.1358   1st Qu.:  0.689   1st Qu.:1.8900
##  Mode  :character   Median :1.2126   Median : 24.465   Median :2.2303
##                     Mean   :1.1885   Mean   : 25.564   Mean   :2.2047
##                     3rd Qu.:1.2687   3rd Qu.: 45.606   3rd Qu.:2.5125
##                     Max.   :1.6454   Max.   :111.934   Max.   :4.3100
##      weight          moisture          type
##  Min.   : 8.532   Min.   : 6.486   Length:275
##  1st Qu.:21.982   1st Qu.: 9.540   Class :character
##  Median :27.610   Median :11.909   Mode  :character
##  Mean   :27.501   Mean   :11.192
##  3rd Qu.:32.882   3rd Qu.:12.538
##  Max.   :46.334   Max.   :14.514
```

```
paste("Sample Size: ", nrow(wheat))
```

```
## [1] "Sample Size:  275"
```

The multinomial categorical variable that I am focused on will be `type`.

```
unique(wheat$type)
```

```
## [1] "Healthy" "Sprout"  "Scab"
```

As a result, there are three categories: *Healthy*, *Sprout*, and *Scab*.

Now, I can run the multinomial regression model.

```
mod.nominal <- multinom(type ~  density, data = wheat)
```

```
## # weights:  9 (4 variable)
## initial  value 302.118379
## iter  10 value 229.769334
## iter  20 value 229.712304
## final  value 229.712290
## converged
```

```
summary(mod.nominal)
```

```
## Call:
## multinom(formula = type ~ density, data = wheat)
##
## Coefficients:
##        (Intercept)   density
## Scab      29.37827 -24.56215
## Sprout    19.12165 -15.47633
##
## Std. Errors:
##        (Intercept)  density
## Scab      3.676892 3.017842
## Sprout    3.337092 2.691429
##
## Residual Deviance: 459.4246
## AIC: 467.4246
```

The `multinom()` function uses *Healthy* as the categorical base level. Thus, the estimated regressions are:

Equation 1: Scab vs. Healthy

$$log\left(\frac{\widehat{\pi}_{Scab}}{\widehat{\pi}_{Healthy}}\right) = 29.3783 - 24.5622density$$

Equation 2: Sprout vs. Healthy

$$log\left(\frac{\widehat{\pi}_{Sprout}}{\widehat{\pi}_{Healthy}}\right) = 19.1217 - 15.4763density$$

5.1 Question 5.1

Compute the log-likelihood function for the multinomial regression model.

I can determine the log-likelihood through its formula, where `m` represents the number of categories and `n` is the number of observations.

$$\Sigma_{i=1}^{n}\Sigma_{j=1}^{m} y_j * log(\pi_j)$$

In this context, y_i serves as a binary indicator as to whether that instance belongs to the given category or not. A value of "1" indicates that the instance belongs to that category and a value of "0" indicates that is does not. Collectively, these columns create a Y indicator matrix, where each column represents a category.

I can calculate π_j, given one explanatory variable and J categories, as follows:

$$\pi_1 = \frac{1}{1 + \Sigma_{j=2}^{J} exp(\beta_{j0} + \beta_{j1} * x_1)}$$

and

$$\pi_j = \frac{exp(\beta_{j0} + \beta_{j1} * x_1)}{1 + \Sigma_{j=2}^{J} exp(\beta_{j0} + \beta_{j1} * x_1)}$$

for j = 2,. . . , J.

First, I will set some variables.

```
n_cat <- 3
n_feat <- 1
```

Next, I will store the coefficients from the model.

```
betas <- c()
max_beta <- (n_cat-1) + (n_cat-1)*n_feat
for(i in 1:(n_cat-1)){
  temp <- i
  while(temp <= max_beta){
    my_beta <- summary(mod.nominal)$coefficients[temp]
    betas <- c(betas,my_beta)
    temp <- temp + (1+n_feat)
  }
}
```

I can also form the Y indicator matrix. The columns will be a binary response for each of the different categories of the dependent variable `type`.

```
y1 <- ifelse(wheat$type == "Healthy", 1,0)
y2 <- ifelse(wheat$type == "Scab", 1,0)
y3 <- ifelse(wheat$type == "Sprout", 1,0)
my_Y <- cbind(y1,y2,y3)
```

Then, I can write an R function that computes the log-likelihood function for the multinomial regression model. The function will require parameters for the model coefficients, the dependent variable, as well as the Y indicator matrix.

```
logL <- function(beta, x, Y){
  mysum1 <- exp(beta[1] + beta[2]*x)
  mysum2 <- exp(beta[3] + beta[4]*x)
  sumTot <- mysum1 + mysum2

  my_pi1 <- (1/(1+sumTot))
  my_pi2 <- (mysum1/(1+sumTot))
  my_pi3 <- (mysum2/(1+sumTot))

  sum(Y[,1]*log(my_pi1) + Y[,2]*log(my_pi2) + Y[,3]*log(my_pi3))
}
```

```
logL(x = wheat$density, Y = my_Y, beta = betas)
```

```
## [1] -229.7123
```

As a result, the MLE is calculated to be -229.7123.

Now, I can verify that the computed value is the same as that produced by the `logLik()` function by comparing the results.

```
logLik(mod.nominal)
```

```
## 'log Lik.' -229.7123 (df=4)
```

I can see that the two function have the same output. Therefore, the same MLE values are produced and the custom function is validated.

5.2 Question 5.2

Maximize the log-likelihood function using `optim()`. Compare your answers to what is obtained by `multinom()`.

I can also try to optimize the parameters of the log-likelihood calculation with the `optim()` function.

```
mle <- optim(betas, logL, x = wheat$density, Y = my_Y, hessian = TRUE,
             control = list(fnscale = -1))
mle
```

```
## $par
## [1]  29.38880 -24.57073  19.12977 -15.48281
##
## $value
## [1] -229.7123
##
## $counts
## function gradient
##       99       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
##
## $hessian
##           [,1]      [,2]      [,3]      [,4]
## [1,] -39.36470 -45.58558  27.48387  31.13035
## [2,] -45.58558 -53.14174  31.13035  35.51185
## [3,]  27.48387  31.13035 -57.76507 -68.86487
## [4,]  31.13035  35.51185 -68.86487 -82.62617
```

The components that result from the `optim()` function include the parameter estimates ($par), the log-likelihood function's maximum value ($value), and the estimated covariance matrix ($hessian). All of these values are practically the same as those produced by `multinom()`, where small differences are due to different convergence criteria for the iterative numerical procedures. The MLE value is exactly the same, while the parameter estimates are slightly different. Furthermore, I can see that the covariance matrix is symmetric along the diagonal. Overall, the `optim()` function reaffirms my earlier MLE results.