Shishir Agarwal - W271 Assignment 1

Due 11:59pm Pacific Time, Sunday February 14, 2021

```
# Start with a clean R environment
rm(list = ls())

library(knitr)
opts_chunk$set(tidy.opts=list(width.cutoff=60),tidy=TRUE)

# Check and install missing packages
list.of.packages <- c("car", "dplyr", "Hmisc", "skimr", "ggplot2", "stargazer",
                      "mcprofile", "gridExtra", "binom","grid")
new.packages <- list.of.packages[!(list.of.packages %in%
                                     installed.packages()[,"Package"])]
if(length(new.packages)) install.packages(new.packages)

# Load Libraries
lapply(c("car", "dplyr", "Hmisc", "skimr", "ggplot2", "stargazer",
         "mcprofile","gridExtra", "binom", "grid"),
       require, character.only = TRUE)
```

```
## [[1]]
## [1] TRUE
##
## [[2]]
## [1] TRUE
##
## [[3]]
## [1] TRUE
##
## [[4]]
## [1] TRUE
##
## [[5]]
## [1] TRUE
##
## [[6]]
## [1] TRUE
##
## [[7]]
## [1] TRUE
##
```

```
## [[8]]
## [1] TRUE
##
## [[9]]
## [1] TRUE
##
## [[10]]
## [1] TRUE
```

```
setwd("/home/jovyan/r_bridge/student_work/shagarwa/Assignment#1")
```

1. Confidence Intervals (2 points)

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%. This was demonstrated with code in the week 1 live session file.

Question 1.1: Use the code from the week 1 live session file and: (1) redo the 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 in the live session code.

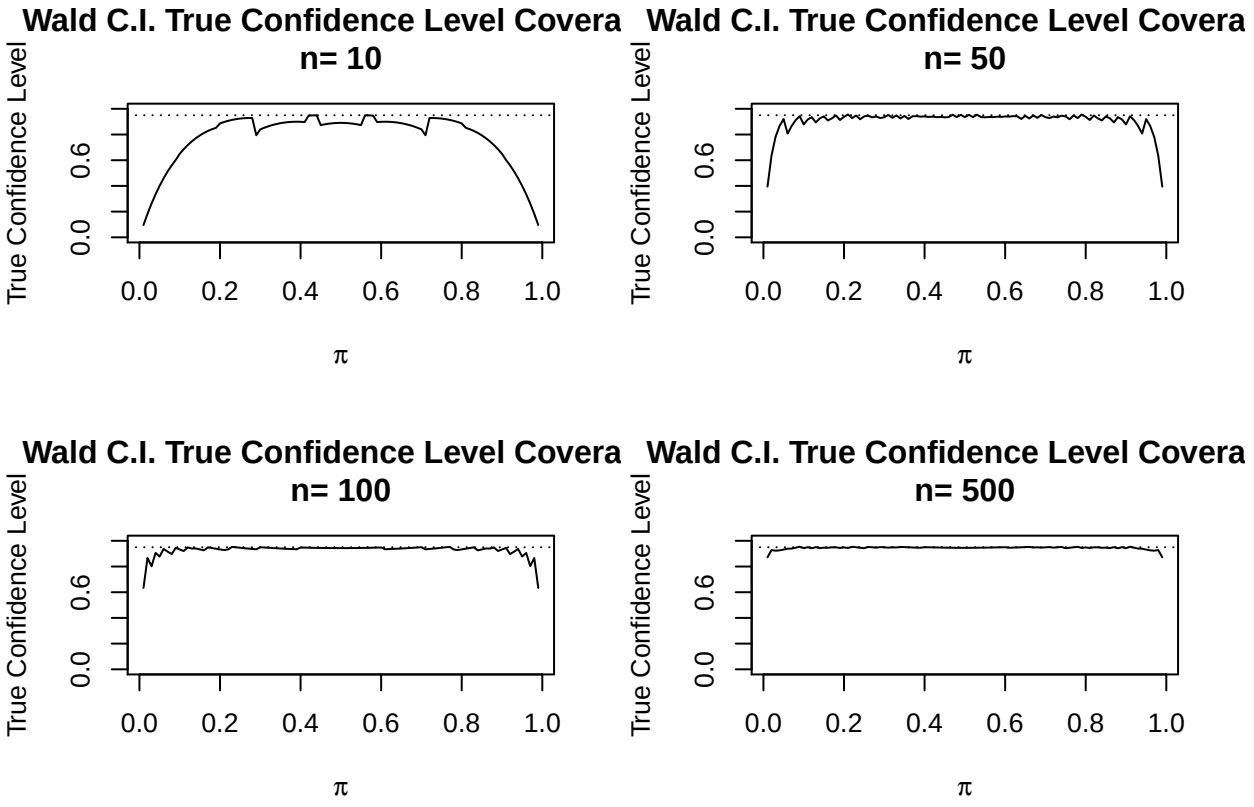
Observation: The Wald confidence interval rely on underlying normal distribution approximation for the estimators. Also the underlying distribution from which we are drawing samples is a discrete distribution as opposed to a continuous distribution. Because of this we find Wald confidence interval to be not accurate. When **n=10** Wald confidence interval **liberal** especially for π values less than 0.2 or more than 0.8. For values between 0.2 and 0.8 the confidence interval estimation is spiky and achieves the true confidence interval for a few values of 'π. However as the sample size n grows and due to asymptotics the sample distribution gets closer to a Normal distribution. Thus, we observe Wald interval as a good approximating to the true confidence interval at higher values of n. Thus, at **n=500** we see the Wald confidence interval approximating to the true confidence interval for majority of π values.

```
alpha <- 0.05
wald.CI.true.coverage = function(pi, alpha=0.05, n=n) {
  w = 0:n
  pi.hat = w/n

  pmf = dbinom(x=w, size=n, prob=pi)
  var.wald = pi.hat*(1-pi.hat)/n
  wald.CI_lower.bound = pi.hat - qnorm(p = 1-alpha/2)*sqrt(var.wald)
  wald.CI_upper.bound = pi.hat + qnorm(p = 1-alpha/2)*sqrt(var.wald)
  covered.pi = ifelse(test = pi>wald.CI_lower.bound,
                      yes = ifelse(test = pi<wald.CI_upper.bound,
                                   yes=1, no=0),
                      no=0)
  wald.CI.true.coverage = sum(covered.pi*pmf)
  wald.df = data.frame(w, pi.hat,
                       round(data.frame(pmf,
                       wald.CI_lower.bound,wald.CI_upper.bound),4),
```

```
                        covered.pi)
  return(wald.df)
}
par(mfrow = c(2,2))
for (n in c(10,50,100,500)) {
  # Let's compute the true coverage for a sequence of pi
  pi.seq = seq(0.01,0.99, by=0.01)
  (wald.CI.true.matrix = matrix(data=NA,nrow=length(pi.seq),ncol=2))
  counter=1
  for (pi in pi.seq) {
    wald.df2 = wald.CI.true.coverage(pi=pi, alpha=0.05, n=n)
    wald.CI.true.matrix[counter,] = c(pi,sum(wald.df2$covered.pi*wald.df2$pmf))
    counter = counter+1
  }

  # 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 = paste("Wald C.I. True Confidence Level Coverage\nn=",n),
       xlab=expression(pi),
       ylab="True Confidence Level",
       type="l")
  abline(h=1-alpha, lty="dotted")
}
```




```
par(mfrow = c(1,1))
```

Question 1.2: (1) Modify the code for the Wilson Interval. (2) Do the exercise for `n=10, n=50, n=100, n=500`. (3) Plot the graphs. (4) Describe what you have observed from the results and compare the Wald and Wilson intervals based on your results. Use the same `pi.seq` as in the live session code.

Observation: We notice the Wilson confidence interval is much more closer to the true confidence interval as well as consistent across different values of **n**. Thus, compared to Wald interval, Wilson interval is lot more accurate especially at values of $\mathbf{n < 100}$. For $\mathbf{n > 500}$, we notice Wald interval is not a bad approximation however Wilson confidence interval is even better. Thus, the important take away is to use Wilson confidence interval and not the Wald confidence interval for $n < 100$. For $n > 100$, one can use Wilson or Wald however Wilson is more accurate.

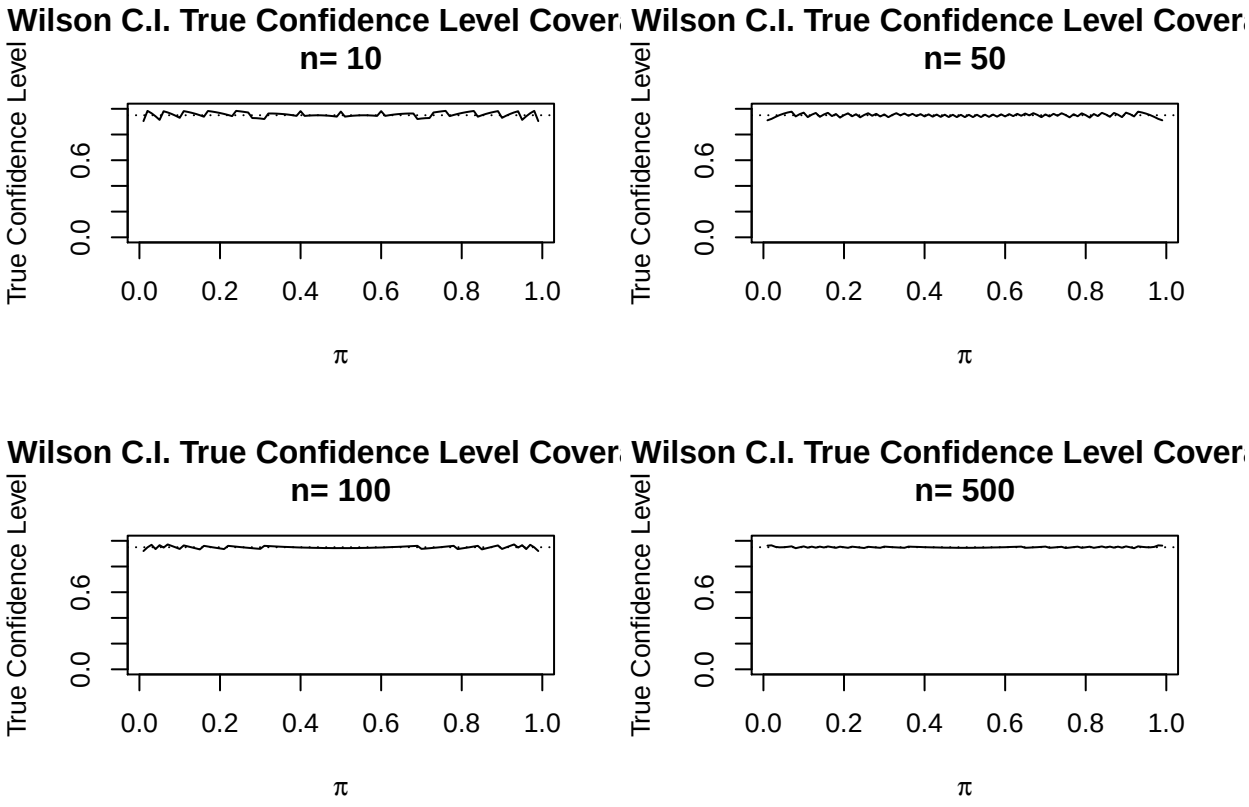
```
alpha <- 0.05
wilson.CI.true.coverage = function(pi, alpha=0.05, n=n) {
  w = 0:n
  pi.hat = w/n
  p.tilde<-(w + qnorm(p = 1-alpha/2)^2 /2) / (n+qnorm(1-alpha/2)^2)

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

  lower.wilson<-p.tilde - qnorm(p = 1-alpha/2) * sqrt(n) / (n+qnorm(1-alpha/2)^2) * sqrt(pi.hat
  upper.wilson<-p.tilde + qnorm(p = 1-alpha/2) * sqrt(n) / (n+qnorm(1-alpha/2)^2) * sqrt(pi.hat
```

```
  covered.pi = ifelse(test = pi>lower.wilson,
                      yes = ifelse(test = pi<upper.wilson,
                                   yes=1, no=0),
                      no=0)
  wilson.CI.true.coverage = sum(covered.pi*pmf)
  wilson.df = data.frame(w, pi.hat,
                         round(data.frame(pmf, lower.wilson,upper.wilson),4),
                         covered.pi)
  return(wilson.df)
}
par(mfrow = c(2,2))
for (n in c(10,50,100,500)) {
  # Let's compute the true coverage for a sequence of pi
  pi.seq = seq(0.01,0.99, by=0.01)
  (wilson.CI.true.matrix = matrix(data=NA,nrow=length(pi.seq),ncol=2))
  counter=1
  for (pi in pi.seq) {
    wilson.df2 = wilson.CI.true.coverage(pi=pi, alpha=0.05, n=n)
    wilson.CI.true.matrix[counter,] = c(pi,sum(wilson.df2$covered.pi*wilson.df2$pmf))
    counter = counter+1
  }

  # 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 = paste("Wilson C.I. True Confidence Level Coverage\nn=",n),
       xlab=expression(pi),
       ylab="True Confidence Level",
       type="l")
  abline(h=1-alpha, lty="dotted")
}
```




```
par(mfrow = c(1,1))
```

2: Binary Logistic Regression (2 points)

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"* which 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**, in addition to answering the question, re-estimate the model in part (a) using "*Sun*" as the base level category for *Weather*.

Question 8 a: Estimate the model and properly define the indicator variables used within it.

```
placekick.data <- read.table(file = "placekick.BW.csv", header = TRUE, sep = ",")
placekick.glm <- glm(factor(Good) ~ Distance + Weather + Wind15 + Temperature +
                       Grass + Pressure + Ice, family = binomial(link = logit),
                       data = placekick.data)
summary(placekick.glm)
```

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

```
beta <- round(placekick.glm$coefficients,2)
contrasts(factor(placekick.data$Weather))
```

```
##          Inside SnowRain Sun
## Clouds        0        0   0
## Inside        1        0   0
## SnowRain      0        1   0
## Sun           0        0   1
```

```
contrasts(factor(placekick.data$Temperature))
```

```
##      Hot Nice
## Cold   0    0
## Hot    1    0
## Nice   0    1
```

Model

$\text{logit}(\text{Probability of Good}) = (5.74) + (\text{-}0.11)\text{Distance} + (\text{-}0.08)\text{WeatherInside} + (\text{-}0.44)\text{WeatherSnowRain} + (\text{-}0.25)\text{WeatherSun} + (\text{-}0.24)\text{Wind15} + (0.25)\text{TemperatureHot} + (0.23)\text{TemperatureNice} + (\text{-}0.33)\text{Grass} + (0.27)\text{PressureY} + (\text{-}0.88)\text{Ice}$

Categorical Variables as Indicator Variables

- WeatherInside - 1 indicates playing inside and 0 indicates playing in cloudy conditions
- WeatherSnowRain - 1 indicates playing in snow/rain and 0 indicates playing in cloudy conditions
- WeatherSun - 1 indicates playing when sunny and 0 indicates playing in cloudy conditions
- TemperatureHot - 1 indicates playing when it is hot and 0 indicates playing when it is cold
- TemperatureNice - 1 indicates playing when it is Nice 0 indicates playing when it is cold

Numerical Variables with 2 distinct values are like Indicator Variables

- Wind15 - 1 indicates playing when windy and 0 indicates playing when not windy
- Grass - 1 indicates playing in grass and 0 indicates playing not in grass
- PressureY - 1 indicates playing under pressure and 0 indicates playing not under pressure
- Ice - 1 indicates playing when under pressure and timeout and 0 otherwise

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.

```
placekick.data$Weather <- relevel(factor(placekick.data$Weather), ref = "Sun")
placekick.glm <- glm(factor(Good) ~ Distance + Weather + Wind15 + Temperature +
                       Grass + Pressure + Ice, family = binomial(link = logit),
```

```
                       data = placekick.data)
summary(placekick.glm)
```

```
##
## Call:
## glm(formula = factor(Good) ~ Distance + Weather + Wind15 + Temperature +
##     Grass + Pressure + Ice, family = binomial(link = logit),
##     data = placekick.data)
##
## 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 ***
## WeatherClouds    0.247582   0.139642   1.773   0.0762 .
## WeatherInside    0.164553   0.215062   0.765   0.4442
## WeatherSnowRain -0.196611   0.219015  -0.898   0.3693
## 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
```

```
beta <- round(placekick.glm$coefficients,2)
contrasts(factor(placekick.data$Weather))
```

```
##          Clouds Inside SnowRain
## Sun           0      0        0
## Clouds        1      0        0
## Inside        0      1        0
## SnowRain      0      0        1
```

Model

$$\text{logit}(\text{Probability of Good}) = (5.49) + (\text{-}0.11)\text{Distance} + (0.25)\text{WeatherClouds} + (0.16)\text{WeatherInside}$$

+ (-0.2)WeatherSnowRain + (-0.24)Wind15 + (0.25)TemperatureHot + (0.23)TemperatureNice + (-0.33)Grass + (0.27)PressureY + (-0.88)Ice

By default R orders the levels of categorical variable numerically and alphabetically. This ordering is used to create the base level and indicator variables. The **relevel()** function can be used to manually re-order the **levels** and thereby select a specific level as the base level.

Question 8 c: Perform LRT for all the explanatory variables to evaluate their importance within the model. Discuss the results.

```
Anova(placekick.glm, test = "LR")
```

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

We know the **Anova LRT test** has null hypothesis as $H_0 : \beta = 0$. Thus, we find **Distance** variable to be immensely important in estimating if the kick will be good or not, given all the other variables are in the model. Also, given the LRT test results we know the **Distance** variable is not zero. We also find **Grass** indicator variable to be marginally important in estimating if the kick will be good or not, given all the variables are in the model. We are surprised we do not have weather variable nor the temperature variable as statistically significant variables in estimating the probability of kick being good or not.

Question 8 d: Estimate an appropriate odds ratio for **distance** and compute the corresponding confidence interval. Interpret the odds ratio.

```
Distance.coeff <- round(as.numeric(placekick.glm$coefficients[2]),2)
Distance.CI = round(rev(as.numeric(confint(placekick.glm, parm = "Distance",
                                           level = 0.95))),2)
c = -10
Distance.OR <- round(exp(c*Distance.coeff),2)
Distance.OR.CI = round(exp(c*Distance.CI),2)
```

The 95% profile LR confidence interval for the **Distance** parameter in the linear predictor is (-0.1, -0.12) where the coefficient itself is -0.11.

Using **c=-10**, the 95% profile LR interval for the odds ratio of **Distance** is (2.72, 3.32) where the odds ratio is 3.

With 95% confidence, the odds of success increases by amount between (2.72, 3.32) for every **10 yards decrease** in the distance of the placekick, holding other variable constant.

3: Binary Logistic Regression (2 points)

The dataset *"admissions.csv"* contains a small sample of graduate school admission data from a university. The variables are specificed below:

1. admit - the depenent variable that takes two values: $0, 1$ where 1 denotes *admitted* and 0 denotes *not admitted*
2. gre - GRE score
3. gpa - College GPA
4. rank - rank in college major

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. We will conduct this analysis by answering the follwing questions:

Question 3.1: Examine the data and conduct EDA

We have 400 observations and 5 variables. We ignore the first variable X since it just represent the serial number. *admit* is our response variable. It takes 2 distinct values of $(0, 1)$ with 31.75 students as admitted. The *gre* scores range from 220 to 800 with a mean score of 587. Similarly, the *gpa* varies from 2.26 to 4.0 with mean gpa of 3.39. The *rank* variable has 4 distinct values from 1 to 4 with 15.2 of students with rank 1, 37.8 of students as rank 2, 30.2 of students as rank 3 and 16.8 of students as rank 4.

```
admission.data <- read.table(file = "admissions.csv", header = TRUE, sep = ",")
names(admission.data)
```

```
## [1] "X"     "admit" "gre"   "gpa"   "rank"
```

```
dim(admission.data)
```

```
## [1] 400   5
```

```
str(admission.data)
```

```
## 'data.frame':    400 obs. of  5 variables:
##  $ X    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ admit: int  0 1 1 1 0 1 1 0 1 0 ...
##  $ gre  : int  380 660 800 640 520 760 560 400 540 700 ...
##  $ gpa  : num  3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...
##  $ rank : int  3 3 1 4 4 2 1 2 3 2 ...
```

```
summary(admission.data)
```

```
##        X             admit             gre             gpa
##  Min.   :  1.0   Min.   :0.0000   Min.   :220.0   Min.   :2.260
##  1st Qu.:100.8   1st Qu.:0.0000   1st Qu.:520.0   1st Qu.:3.130
##  Median :200.5   Median :0.0000   Median :580.0   Median :3.395
##  Mean   :200.5   Mean   :0.3175   Mean   :587.7   Mean   :3.390
##  3rd Qu.:300.2   3rd Qu.:1.0000   3rd Qu.:660.0   3rd Qu.:3.670
##  Max.   :400.0   Max.   :1.0000   Max.   :800.0   Max.   :4.000
##       rank
```

```
##  Min.   :1.000
##  1st Qu.:2.000
##  Median :2.000
##  Mean   :2.485
##  3rd Qu.:3.000
##  Max.   :4.000
```

```
#glimpse(admission.data)
describe(admission.data)
```

```
## admission.data
##
##  5  Variables      400  Observations
## --------------------------------------------------------------------------------
## X
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      400        0      400        1    200.5    133.7    20.95    40.90
##      .25      .50      .75      .90      .95
##   100.75   200.50   300.25   360.10   380.05
##
## lowest :   1   2   3   4   5, highest: 396 397 398 399 400
## --------------------------------------------------------------------------------
## admit
##        n  missing distinct     Info      Sum     Mean      Gmd
##      400        0        2     0.65      127   0.3175   0.4345
##
## --------------------------------------------------------------------------------
## gre
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      400        0       26    0.997    587.7    131.2      399      440
##      .25      .50      .75      .90      .95
##      520      580      660      740      800
##
## lowest : 220 300 340 360 380, highest: 720 740 760 780 800
## --------------------------------------------------------------------------------
## gpa
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      400        0      132        1     3.39   0.4351    2.758    2.900
##      .25      .50      .75      .90      .95
##    3.130    3.395    3.670    3.940    4.000
##
## lowest : 2.26 2.42 2.48 2.52 2.55, highest: 3.95 3.97 3.98 3.99 4.00
## --------------------------------------------------------------------------------
## rank
##        n  missing distinct     Info     Mean      Gmd
##      400        0        4     0.91    2.485    1.038
##
## Value          1     2     3     4
```

```
## Frequency     61   151   121    67
## Proportion 0.152 0.378 0.302 0.168
## ------------------------------------------------------------------------------
```

```
table(admission.data$rank)
```

```
##
##   1   2   3   4
##  61 151 121  67
```

```
prop.table(table(admission.data$rank))
```

```
##
##      1      2      3      4
## 0.1525 0.3775 0.3025 0.1675
```

```
table(admission.data$admit)
```

```
##
##   0   1
## 273 127
```

```
prop.table(table(admission.data$admit))
```

```
##
##      0      1
## 0.6825 0.3175
```

The data seems generally normally distributed however we see a large number of students that have GRE score of 800. Also, we have a large number of students with 4.0 GPA.

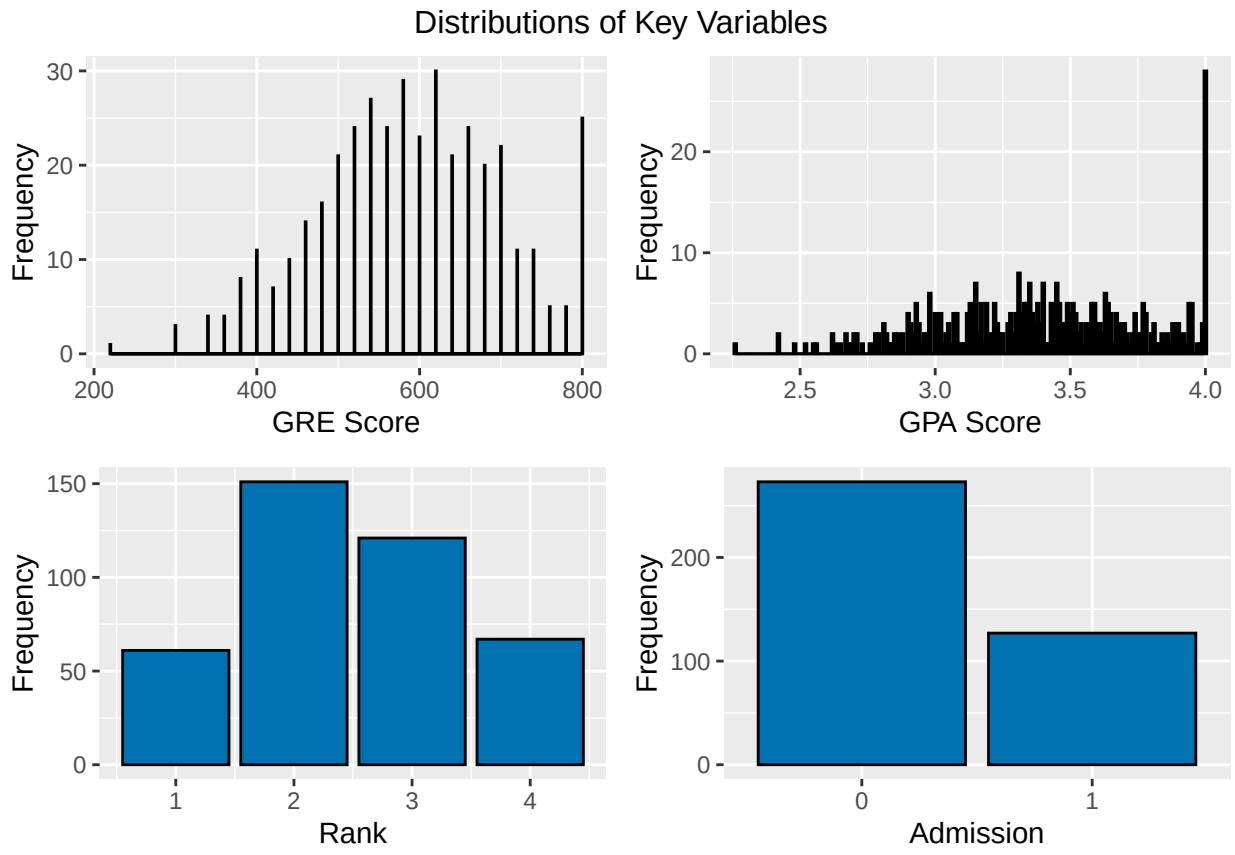
```
# GRE plot
gre.plot <- ggplot(admission.data, aes(x = gre)) +
  geom_histogram(aes(x = gre), binwidth = 1, fill="#0072B2", colour="black") +
  xlab("GRE Score") +
  ylab("Frequency") + theme(plot.title = element_text(lineheight=1, face="bold",
                                                      color ="dark blue"))

# GPA Plot
gpa.plot <- ggplot(admission.data, aes(x = gpa)) +
  geom_histogram(aes(x = gpa), binwidth = 0.01, fill="#0072B2", colour="black") +
  xlab("GPA Score") +
  ylab("Frequency") + theme(plot.title = element_text(lineheight=1, face="bold",
                                                      color ="dark blue"))

# Rank Plot
rank.plot <- ggplot(admission.data, aes(x = rank)) +
  geom_bar(aes(x = rank), fill="#0072B2", colour="black") +
  xlab("Rank") +
  ylab("Frequency") + theme(plot.title = element_text(lineheight=1, face="bold",
                                                      color ="dark blue"))
```

```
# admit Plot
admit.plot <- ggplot(admission.data, aes(x = factor(admit))) +
  geom_bar(aes(x = factor(admit)), fill="#0072B2", colour="black") +
  xlab("Admission") +
  ylab("Frequency") + theme(plot.title = element_text(lineheight=1, face="bold",
                                                      color ="dark blue"))

grid.arrange(gre.plot, gpa.plot, rank.plot, admit.plot,
             top="Distributions of Key Variables" ,
            ncol=2)
```



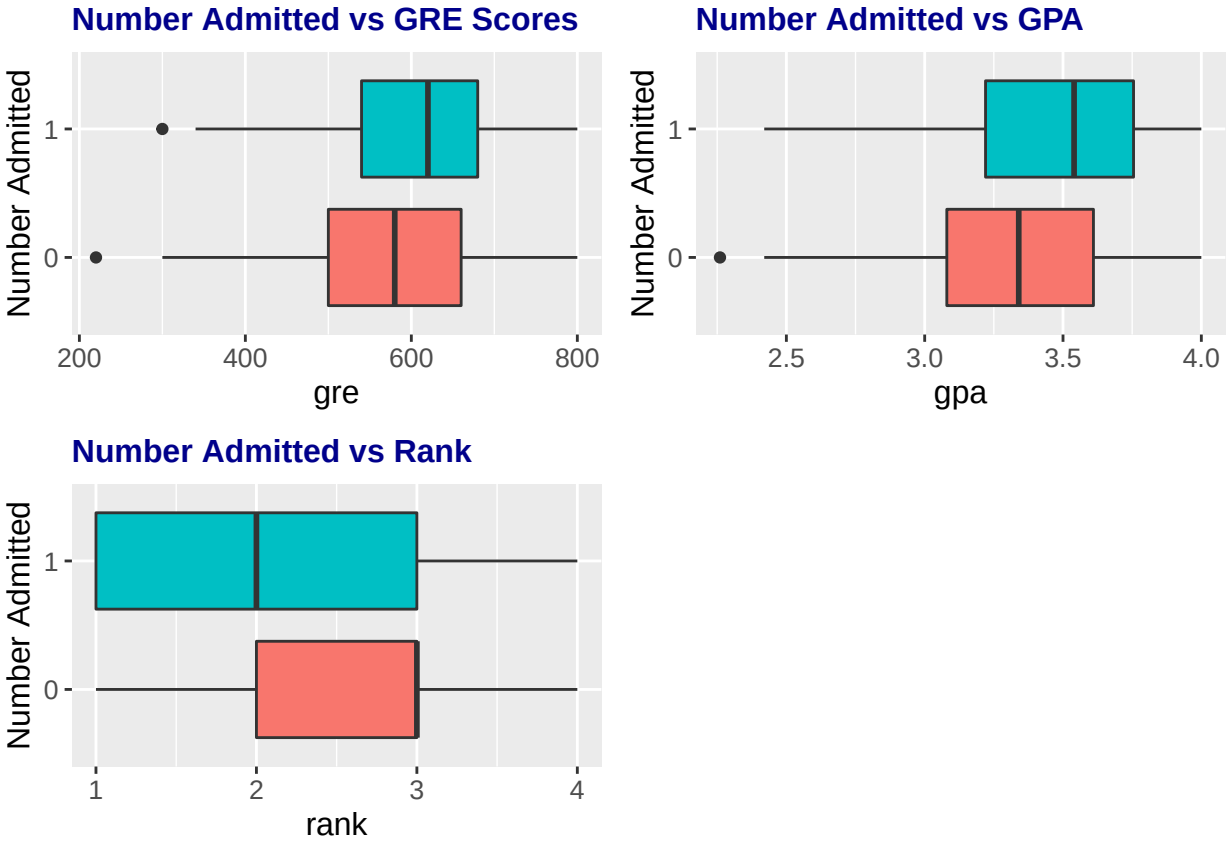
Looking at the box plot below we notice the GRE scores do not make much difference in terms of admission though higher GRE scores provides higher association with admission. We notice higher GPA association with admissions compared to GRE scores. Lastly, Rank seems like the most important variable and shows rank 1 has higher assciation with admission than rank 4

```
admit_gre.box <- ggplot(admission.data, aes(gre, factor(admit))) +
  geom_boxplot(aes(fill = factor(admit))) +
  guides(fill=FALSE) + ggtitle("Number Admitted vs GRE Scores") +
  ylab("Number Admitted") +
  theme(axis.text = element_text(size=10),
        axis.title = element_text(size=12),
        plot.title = element_text(lineheight=1, size =12, face="bold", color ="dark blue"))
```

```
admit_gpa.box <- ggplot(admission.data, aes(gpa, factor(admit))) +
  geom_boxplot(aes(fill = factor(admit))) +
  guides(fill=FALSE) + ggtitle("Number Admitted vs GPA") +
  ylab("Number Admitted") +
  theme(axis.text = element_text(size=10),
        axis.title = element_text(size=12),
        plot.title = element_text(lineheight=1, size =12, face="bold", color ="dark blue"))

admit_rank.box <- ggplot(admission.data, aes(rank, factor(admit))) +
  geom_boxplot(aes(fill = factor(admit))) +
  guides(fill=FALSE) + ggtitle("Number Admitted vs Rank") +
  ylab("Number Admitted") +
  theme(axis.text = element_text(size=10),
        axis.title = element_text(size=12),
        plot.title = element_text(lineheight=1, size =12, face="bold", color ="dark blue"))

grid.arrange(admit_gre.box, admit_gpa.box, admit_rank.box, nrow=2)
```



Question 3.2: Estimate a binary logistic regression using the following set of explanatory variables: gre, gpa, $rank$, gre^2, gpa^2, and $gre \times gpa$, where $gre \times gpa$ denotes the interaction between gre and gpa variables

```
admission.glm <- glm(factor(admit) ~ gre + gpa + rank + I(gre^2) + I(gpa^2) +
              gre:gpa, family = binomial(link = logit), data = admission.data)
summary(admission.glm)
```

```
##
## Call:
## glm(formula = factor(admit) ~ gre + gpa + rank + I(gre^2) + I(gpa^2) +
##     gre:gpa, family = binomial(link = logit), data = admission.data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4928  -0.8958  -0.6192   1.1436   2.2211
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -7.092e+00  9.024e+00  -0.786   0.4319
## gre          1.845e-02  1.169e-02   1.578   0.1146
## gpa         -7.960e-03  4.933e+00  -0.002   0.9987
## rank        -5.643e-01  1.278e-01  -4.414 1.01e-05 ***
## I(gre^2)     3.495e-06  8.156e-06   0.429   0.6683
## I(gpa^2)     6.511e-01  7.605e-01   0.856   0.3919
## gre:gpa     -5.987e-03  3.186e-03  -1.879   0.0602 .
## ---
## 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: 455.72  on 393  degrees of freedom
## AIC: 469.72
##
## Number of Fisher Scoring iterations: 4
```

Question 3.3: Test the hypothesis that GRE has no effect on admission using the likelihood ratio test

Looking at the results, we notice the higher order GRE and GPA variables as well as interaction between GRE and GPA are not important in affecting admissions. The only variable that is significant in determining admission is *rank*. GRE does not have effect on admission because from the LRT it is clear we are unable to reject the null hypothesis of $H_0 : \beta gre = 0$

```
Anova(admission.glm, test = "LR")
```

```
## Analysis of Deviance Table (Type II tests)
##
## Response: factor(admit)
##          LR Chisq Df Pr(>Chisq)
## gre        0.3451  1    0.55690
## gpa        0.0127  1    0.91033
```

```
## rank       20.9972  1    4.6e-06 ***
## I(gre^2)    0.1817  1    0.66989
## I(gpa^2)    0.7239  1    0.39487
## gre:gpa     3.5511  1    0.05951 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Question 3.4: What is the estimated effect of college GPA on admission?

Like GRE scores, GPA does not have a affect on the admissions because we are unable to reject the null hypothesis of $H_0 : \beta gpa = 0$.

Question 3.5: Construct the confidence interval for the admission probability for the students with $GPA = 3.3$, $GRE = 720$, and $rank = 1$

```
testdata = data.frame(gre = 720, gpa = 3.3, rank = 1)
linear.pred <- predict(admission.glm, newdata = testdata, type = "link", se = TRUE)
alpha = 0.05
pi.hat <- exp(linear.pred$fit)/(1+exp(linear.pred$fit))
CI.lin.pred <- linear.pred$fit + qnorm(p = c(alpha/2, 1-alpha/2)) * linear.pred$se
CI.pi <- exp(CI.lin.pred)/(1 + exp(CI.lin.pred))
data.frame(alpha = alpha, pi.hat, lower = CI.pi[1], upper = CI.pi[2])
```

```
##   alpha    pi.hat     lower     upper
## 1  0.05 0.5692897 0.4366982 0.6926379
```

4. Binary Logistic Regression (2 points)

Load the `Mroz` data set that comes with the *car* library (this data set is used in the week 2 live session file).

```
data("Mroz")
str(Mroz)
```

```
## 'data.frame':    753 obs. of  8 variables:
##  $ lfp : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ k5  : int  1 0 1 0 1 0 0 0 0 0 ...
##  $ k618: int  0 2 3 3 2 0 2 0 2 2 ...
##  $ age : int  32 30 35 34 31 54 37 54 48 39 ...
##  $ wc  : Factor w/ 2 levels "no","yes": 1 1 1 1 2 1 2 1 1 1 ...
##  $ hc  : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ lwg : num  1.2102 0.3285 1.5141 0.0921 1.5243 ...
##  $ inc : num  10.9 19.5 12 6.8 20.1 ...
```

Question 4.1: Estimate a linear probability model using the same specification as in the binary logistic regression model estimated in the week 2 live session. Interpret the model results. Conduct model diagnostics. Test the CLM model assumptions.

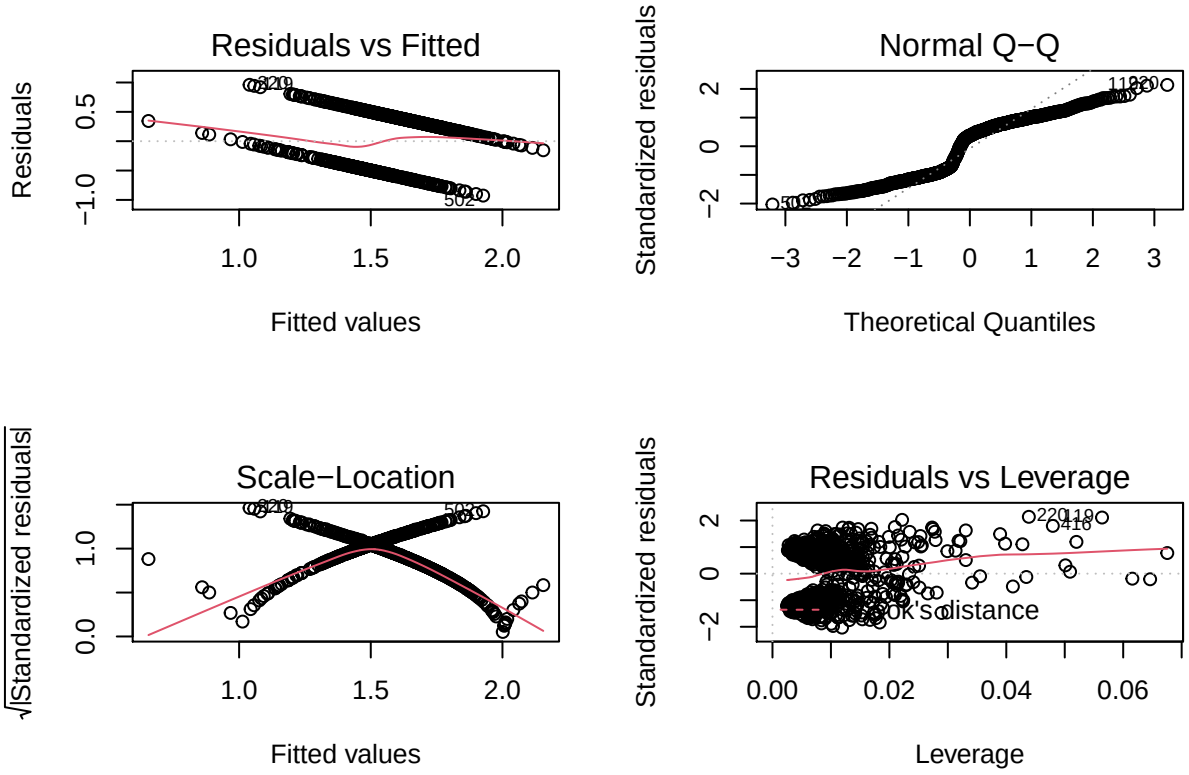
Below is the linear regression model with the same specification as in week 2 live session. I would consider the results of the model as a suspect because some of the basic assumptions for linear regression are violated especially the zero conditional mean, normality of error terms and homoskedasticity. For example when we look at "Residual vs Fitted" graph we see the mean of residuals are not centered around zero which violates the zero conditional mean. When we look at Normal Q-Q plat we see a divergence of residuals from normal distribution. So we further look at the residuals within a histogram and we see a bi-modal distribution of the residuals. Now having a large sample size we may be able to ignore the two violation however we are unable to ignore the violation of homoskedasticity assumption. Lastly, when we look at the response variable that takes values of either 1 and 2 we know the linear model will not be valid especially for prediction as it may predict results greater than 2 and less than 1.

```
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
```

```
par(mfrow=c(2,2))
plot(mroz.lm)
```




```
hist(mroz.lm$residuals)
cov(mroz.lm$residuals, mroz.lm$fitted.values)
```

```
## [1] -4.299894e-16
```

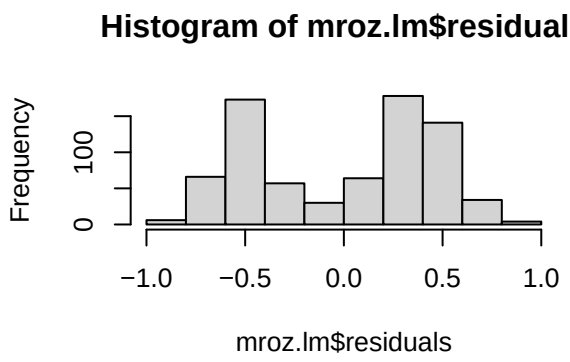
```
shapiro.test(mroz.lm$residuals)
```

```
##
##  Shapiro-Wilk normality test
```

```
##
## data:  mroz.lm$residuals
## W = 0.91081, p-value < 2.2e-16
```

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

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



Question 4.2: Estimate a binary logistic regression with `lfp`, which is a binary variable recoding the participation of the females in the sample, as the dependent variable. The set of explanatory variables includes `age`, `inc`, `wc`, `hc`, `lwg`, `totalKids`, and a quadratic term of `age`, called `age_squared`, where `totalKids` is the total number of children up to age 18 and is equal to the sum of `k5` and `k618`.

```
mroz.data <- Mroz
mroz.data$totalKids <- mroz.data$k5 + mroz.data$k618
mroz.data$age_squared <- mroz.data$age^2
str(mroz.data)
```

```
## 'data.frame':    753 obs. of  10 variables:
##  $ lfp        : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
```

```
##  $ k5         : int  1 0 1 0 1 0 0 0 0 0 ...
##  $ k618       : int  0 2 3 3 2 0 2 0 2 2 ...
##  $ age        : int  32 30 35 34 31 54 37 54 48 39 ...
##  $ wc         : Factor w/ 2 levels "no","yes": 1 1 1 1 2 1 2 1 1 1 ...
##  $ hc         : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ lwg        : num  1.2102 0.3285 1.5141 0.0921 1.5243 ...
##  $ inc        : num  10.9 19.5 12 6.8 20.1 ...
##  $ totalKids  : int  1 2 4 3 3 0 2 0 2 2 ...
##  $ age_squared: num  1024 900 1225 1156 961 ...
```

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

```
##
## Call:
## glm(formula = lfp ~ totalKids + age + age_squared + wc + hc +
##     lwg + inc, family = binomial, data = mroz.data)
##
## 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 *
## totalKids   -0.222490   0.063849  -3.485 0.000493 ***
## age          0.318014   0.109463   2.905 0.003670 **
## age_squared -0.004114   0.001272  -3.233 0.001224 **
## 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 ***
## inc         -0.034561   0.007922  -4.363 1.28e-05 ***
## ---
## 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
```

Question 4.3: Is the age effect statistically significant?

Yes, age effect is statistically significant for both age and age_squared. We see in both cases we are able to reject the null hypothesis.

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

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

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.

```
c = -5
age = 45
lfp.OR = exp(c*mroz.glm$coefficients[3] + c*mroz.glm$coefficients[4]*(2*age+c))
as.numeric(lfp.OR)
```

```
## [1] 1.171602
```

The odds of participation increase by 1.1716016 with every decrease in age by 5 years for a female who is 45 years of age.

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 nor had a husband who attended college, had log wage equal to 1, and did not have children.

```
alpha = 0.05
testdata = data.frame(age = 40, inc = 20.0, wc = "no", hc = "no", lwg = 1,
                      totalKids = 0, age_squared = 1600)
linear.pred <- predict(mroz.glm, type = "link", se = TRUE, newdata = testdata)
pi.hat <- exp(linear.pred$fit)/(1+exp(linear.pred$fit))
linear.pred.CI <- linear.pred$fit + (linear.pred$se * qnorm(c(alpha/2,1-alpha/2)))
pi.hat.CI <- exp(linear.pred.CI)/(1+exp(linear.pred.CI))
data.frame(alpha, pi.hat, lower = pi.hat.CI[1], upper = pi.hat.CI[2])
```

```
##   alpha   pi.hat     lower     upper
## 1  0.05 0.668822 0.5861286 0.7422584
```

5: Maximum Likelihood (2 points)

Question 18 a and b of Chapter 3 (page 192,193)

For the wheat kernel data (*wheat.csv*), consider a model to estimate the kernel condition using the density explanatory variable as a linear term.

Question 5.1 Write an R function that computes the log-likelihood function for the multinomial regression model. Evaluate the function at the parameter estimates produced by multinom(), and verify that your computed value is the same as that produced by logLik() (use the object saved from multinom() within this function).

```
library(nnet)
wheat.data <- read.table(file = "wheat.csv", header = TRUE, sep = ",")
str(wheat.data)
```

```
## 'data.frame':    275 obs. of  7 variables:
##  $ class   : chr  "hrw" "hrw" "hrw" "hrw" ...
##  $ density : num  1.35 1.29 1.23 1.34 1.26 ...
##  $ hardness: num  60.3 56.1 44 53.8 44.4 ...
##  $ size    : num  2.3 2.73 2.51 2.27 2.35 ...
##  $ weight  : num  24.6 33.3 31.8 32.7 26.1 ...
##  $ moisture: num  12 12.2 11.9 12.1 12.1 ...
##  $ type    : chr  "Healthy" "Healthy" "Healthy" "Healthy" ...
```

```
table(wheat.data$type)
```

```
##
## Healthy    Scab  Sprout
##      96      83      96
```

```
wheat.mult <- multinom(formula = type ~ density, data = wheat.data)
```

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

```
summary(wheat.mult)
```

```
## Call:
## multinom(formula = type ~ density, data = wheat.data)
##
## 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
```

```
logLik(wheat.mult)
```

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

```
coef(wheat.mult)
```

```
##        (Intercept)   density
## Scab      29.37827 -24.56215
## Sprout    19.12165 -15.47633
```

```
vcov(wheat.mult)
```

```
##                     Scab:(Intercept) Scab:density Sprout:(Intercept)
## Scab:(Intercept)          13.519536   -11.076940          10.325093
## Scab:density             -11.076940     9.107371          -8.328101
## Sprout:(Intercept)        10.325093    -8.328101          11.136180
## Sprout:density            -8.272576     6.681955          -8.970700
##                     Sprout:density
## Scab:(Intercept)         -8.272576
## Scab:density              6.681955
## Sprout:(Intercept)       -8.970700
## Sprout:density            7.243792
```

```
logL_Mult <- function(mult_object) {
  coeff <- coef(mult_object)
  x <- model.frame(mult_object)$density
  Y <- model.frame(mult_object)$type
  pi_healthy <- 1/(1 + (exp(coeff[1,1]+coeff[1,2]*x))
                   + (exp(coeff[2,1]+coeff[2,2]*x)))
  pi_scab <- pi_healthy * (exp(coeff[1,1]+coeff[1,2]*x))
  pi_sprout <- pi_healthy * (exp(coeff[2,1]+coeff[2,2]*x))
  y_healthy <- ifelse(Y =="Healthy", 1, 0)
  y_scab <- ifelse(Y =="Scab", 1, 0)
  y_sprout <- ifelse(Y =="Sprout", 1, 0)
  sum(log(pi_healthy)*y_healthy + log(pi_scab)*y_scab + log(pi_sprout)*y_sprout)
}

# The results of this function matched the results of logLik function
logL_Mult(wheat.mult)
```

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

Question 5.2 Maximize the log-likelihood function using optim() to obtain the MLEs and the estimated covariance matrix. Compare your answers to what is obtained by multinom(). Note that to obtain starting values for optim(), one approach is to estimate separate logistic regression models

for $log\left(\frac{\pi_2}{\pi_1}\right)$ and $log\left(\frac{\pi_3}{\pi_1}\right)$. These models are estimated only for those observations that have the corresponding responses (e.g., a $Y = 1$ or $Y = 2$ for $log\left(\frac{\pi_2}{\pi_1}\right)$).

```
logL_Mult_opt <- function(beta, x, Y) {
  pi_healthy <- 1/(1 + (exp(beta[1]+beta[2]*x)) + (exp(beta[3]+beta[4]*x)))
  pi_scab <- pi_healthy * (exp(beta[1]+beta[2]*x))
  pi_sprout <- pi_healthy * (exp(beta[3]+beta[4]*x))
  y_healthy <- ifelse(Y =="Healthy", 1, 0)
  y_scab <- ifelse(Y =="Scab", 1, 0)
  y_sprout <- ifelse(Y =="Sprout", 1, 0)
  sum(log(pi_healthy)*y_healthy + log(pi_scab)*y_scab + log(pi_sprout)*y_sprout)
}

wheat.data.healthy.scab <- wheat.data %>%
  filter(type == "Healthy" | type == "Scab" )
wheat.data.healthy.scab.glm <- glm(factor(type) ~ density,
        family = binomial(link = logit), data = wheat.data.healthy.scab)

wheat.data.healthy.sprout <- wheat.data %>%
  filter(type == "Healthy" | type == "Sprout" )
wheat.data.healthy.sprout.glm <- glm(factor(type) ~ density,
        family = binomial(link = logit), data = wheat.data.healthy.sprout)

start_coeff <- matrix(nrow = 2, ncol = 2)
start_coeff[1,] <- wheat.data.healthy.scab.glm$coefficients
start_coeff[2,] <- wheat.data.healthy.sprout.glm$coefficients

wheat.optim <- optim(par = as.vector(t(start_coeff)),
                         fn = logL_Mult_opt,
                         hessian = TRUE,
                         x = wheat.data$density,
                         Y = wheat.data$type,
                         control = list(fnscale = -1),
                         method = "BFGS")

# Results of this matches the results of multinom function
wheat.optim$par
```

```
## [1]  29.39074 -24.57235  19.13215 -15.48479
```

```
wheat.optim$value
```

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

```
-solve(wheat.optim$hessian)
```

```
##             [,1]       [,2]      [,3]      [,4]
## [1,]  13.528402 -11.084135 10.332701 -8.278667
## [2,] -11.084135   9.113217 -8.334221  6.686858
```

```
## [3,]  10.332701  -8.334221 11.143631 -8.976677
## [4,]  -8.278667   6.686858 -8.976677  7.248590
```