Topics in discrete response

Stone Jiang

The RMarkdown will explore 3 topics in the analysis of discrete response:

1. Confidence intervals for the parameter of a binomial distribution

For small sample sizes and/or values of π close to 0 or 1, the normal distribution is a poor approximation for the binomial. Due to the finite outcome space of discrete distributions, a finite number of confidence intervals for any calculation procedure exists for any given data set of fixed size n. The true confidence level of these intervals are rarely achieved under these circumstances. We will explore empirical confidence intervals for the binomial distribution, and compare their claim confidence level to their true confidence level.

2. Inference with Logistic Regression

Logistic regression is a commonly used classfier in machine learning. Here, rather than focusing on its predictive power as binary classifier, we will focus on inference and interpretibility of logistic regression models. We will demonstrate on three separate problems coefficient interpretation, various hypothesis testing procedures, confidence interval generation, and problem-solving.

3. Implementation of MLE for Multinomial Regression

Finally, we will implement maximum likelihood estimation (MLE) with multinomial regression. MLE is an important concept that is used for many parameter estimation procedures, and the theoretical outline presented, as well as MLE calculation code, can be generalized any generalized linear model regression problem where the model parameters are a function of a set of collected observables.

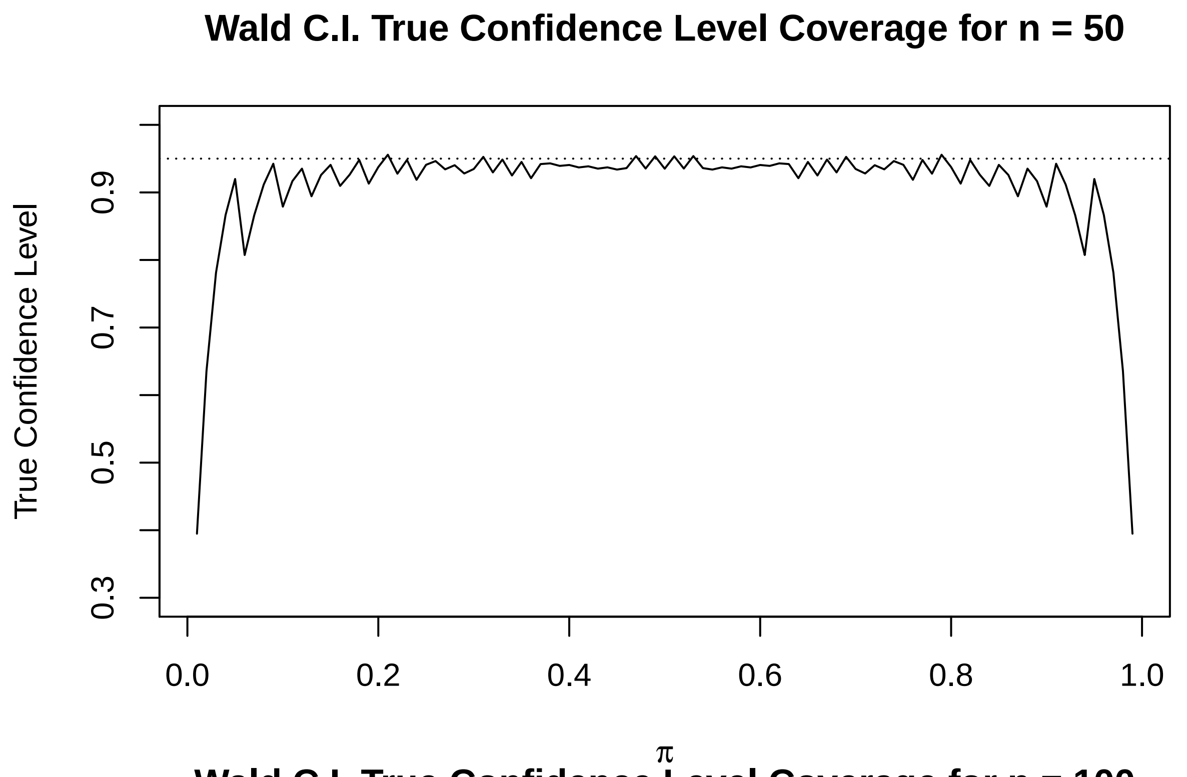
```r
#Set up global knitr option so that text does not go off of the page
library(knitr)
opts_chunk$set(tidy.opts=list(width.cutoff=60),tidy=TRUE)
```

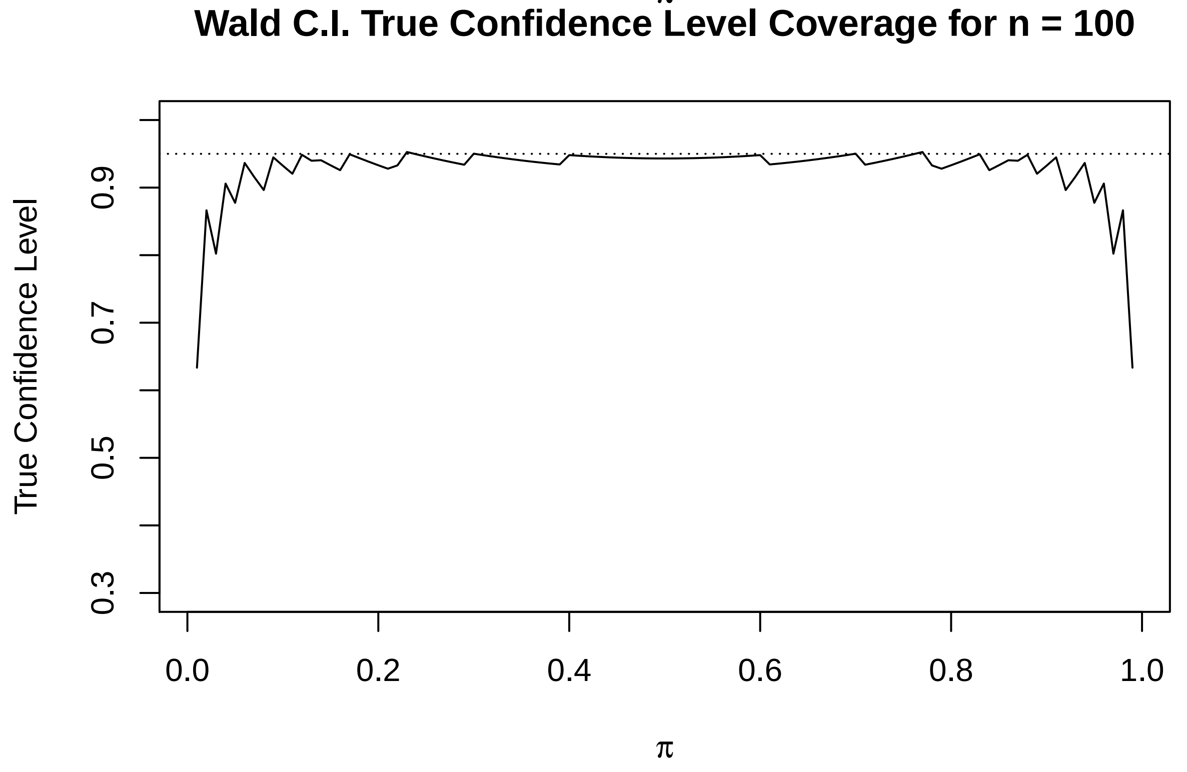
1. Confidence Intervals (2 points)

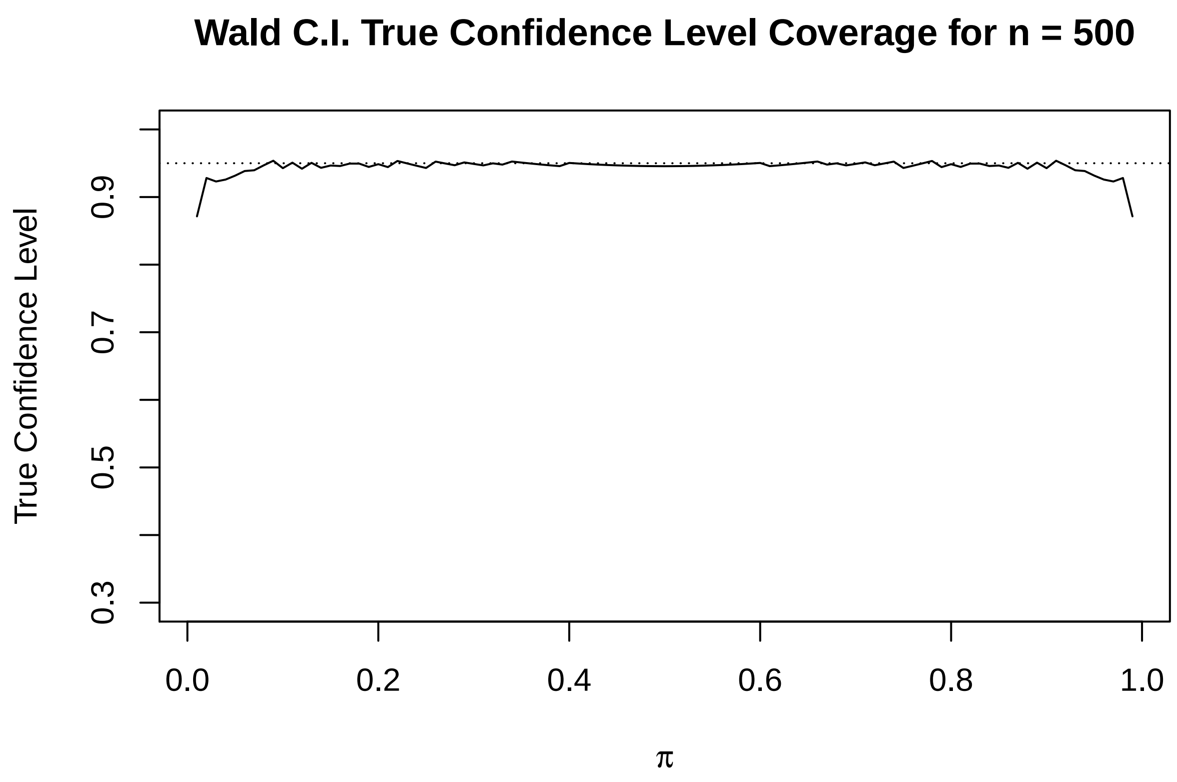
A Wald confidence interval for a binary response probability does not always have the stated confidence level, $1 - \alpha$, where α (the probability of rejecting the null hypothesis when it is true) is often set to 0.05%. This was demonstrated with code in the week 1 live session file.

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.

```r
# Code from week 1 live session file is reproduced below
# Additional comments have been added to explain what the
# code is doing

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

    # There is a finite number of values w, the number of
    # successes, can take This ranges from 0 up to the number of
    # trials
    w = 0:n

    # This is the estimated pi.hat given the number of successes
    # w
    pi.hat = w/n

    # calculate the probability of observing each w under the
    # true pi
    pmf = dbinom(x = w, size = n, prob = pi)

    # wald confidence interval calculated for each w
    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)

    # determine whether the CIs above contain the true pi
    covered.pi = ifelse(test = pi > wald.CI_lower.bound, yes = ifelse(test = pi <
        wald.CI_upper.bound, yes = 1, no = 0), no = 0)

    # the true confidence level is the sum of the probabilities
    # of the CIs which do contain the true pi
    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.CI.true.coverage)
}

wald.CI.true.coverage.pi.range <- function(n, pi.seq = seq(0.01,
    0.99, by = 0.01)) {
    # Computes the Wald CI true coverage for a range of pi values

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

    # calculate the wald CI true coverage for each true pi value
    # and store in matrix
    counter = 1
    for (pi in pi.seq) {
        true.coverage = wald.CI.true.coverage(pi = pi, alpha = 0.05,
            n = n)
        wald.CI.true.matrix[counter, ] = c(pi, sum(true.coverage))
        # Make sure to use sum of wald.df = data.frame(w, pi.hat,
        # round(data.frame(pmf,
        # wald.CI_lower.bound,wald.CI_upper.bound),4), covered.pi)

        counter = counter + 1
    }
    return(wald.CI.true.matrix)
}


# Plot the true coverage level (for given n and alpha)
plot.CI.true.coverage <- function(CI.true.matrix, alpha = 0.05,
    main = "", ylim = c(0.3, 1)) {

    plot(x = CI.true.matrix[, 1], y = CI.true.matrix[, 2], ylim = ylim,
        main = main, xlab = expression(pi), ylab = "True Confidence Level",
        type = "l")

    # Also plot a dotted line for the true confidence level
    abline(h = 1 - alpha, lty = "dotted")
}

# Generate the plots for desired n values
for (n in c(50, 100, 500)) {
    n.wald.CI.true.matrix <- wald.CI.true.coverage.pi.range(n)
    plot.CI.true.coverage(n.wald.CI.true.matrix, main = paste0("Wald C.I. ",
        "True Confidence Level Coverage for n = ", n))
}
```


Wald C.I. True Confidence Level Coverage for n = 50
True Confidence Level
0.3
0.5
0.7
0.9
0.0
0.2
0.4
0.6
0.8
1.0
π


Wald C.I. True Confidence Level Coverage for n = 100
True Confidence Level
0.3
0.5
0.7
0.9
0.0
0.2
0.4
0.6
0.8
1.0
π

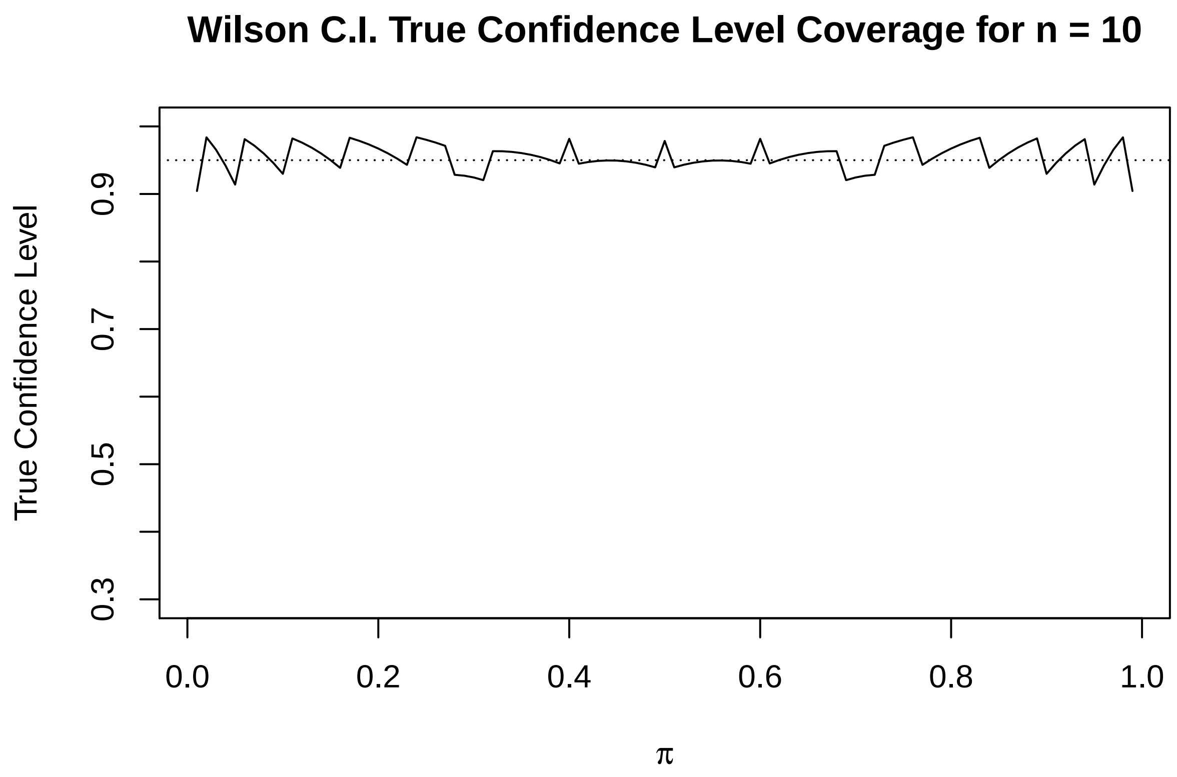


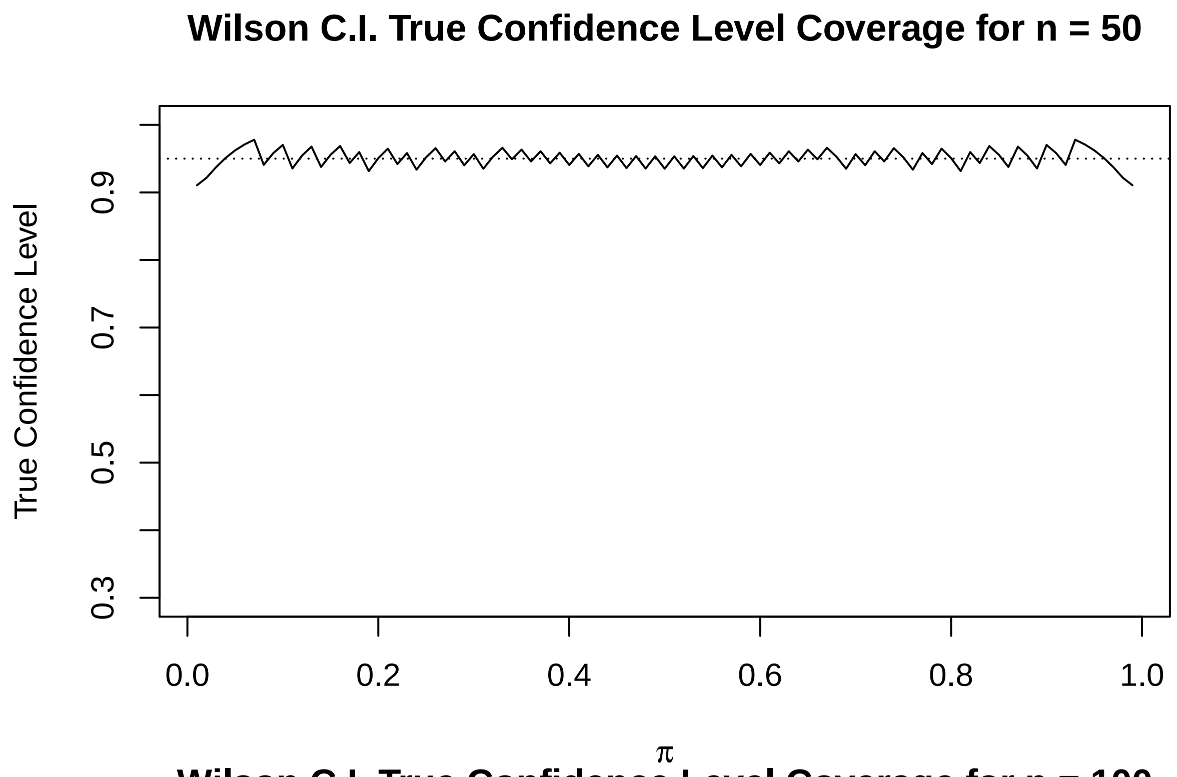
As n increases, the true confidence level gets closer and closer to the stated confidence level of 0.95. This is especially true for extreme values of π close to 0 or 1. The reason for this is that as n increases, the normal distribution becomes a better and better approximation to the binomial distribution, since the binomial random variable is a sum of Bernoulli random variables, which asymptotically converges to the normal distribution by CLT. The Wald CI is constructed assuming that $\frac{\hat{\pi}-\pi}{\sqrt{Var(\hat{\pi})}}$ is distributed as the standard normal, and this approximation becomes more and more accurate as n increases. This means that the stated confidence level gets closer to the true confidence level, and the true confidence level is achieved only if the statistic is exactly distributed as the standard normal.

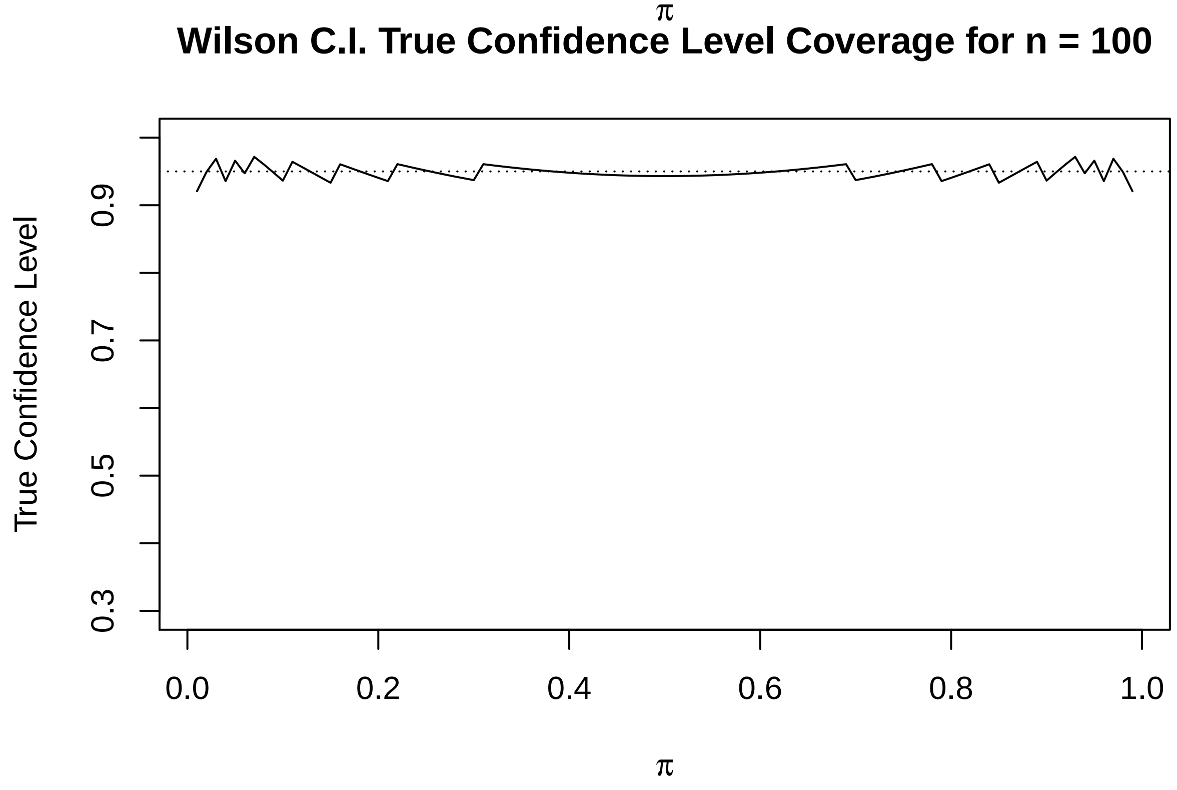
The reason that extreme values of π are affected more is that binomial distributions with π close to 0 and 1 are highly skewed to begin with compared to π values closer to 0.5, which means under CLT they require larger sample sizes to converge to the normal distribution than π values close to 0.5. Even at n=500, π values from 0-0.1 and 0.9-1 have true levels consistently lower than the stated level, and we expect this to get closer when n gets even larger.

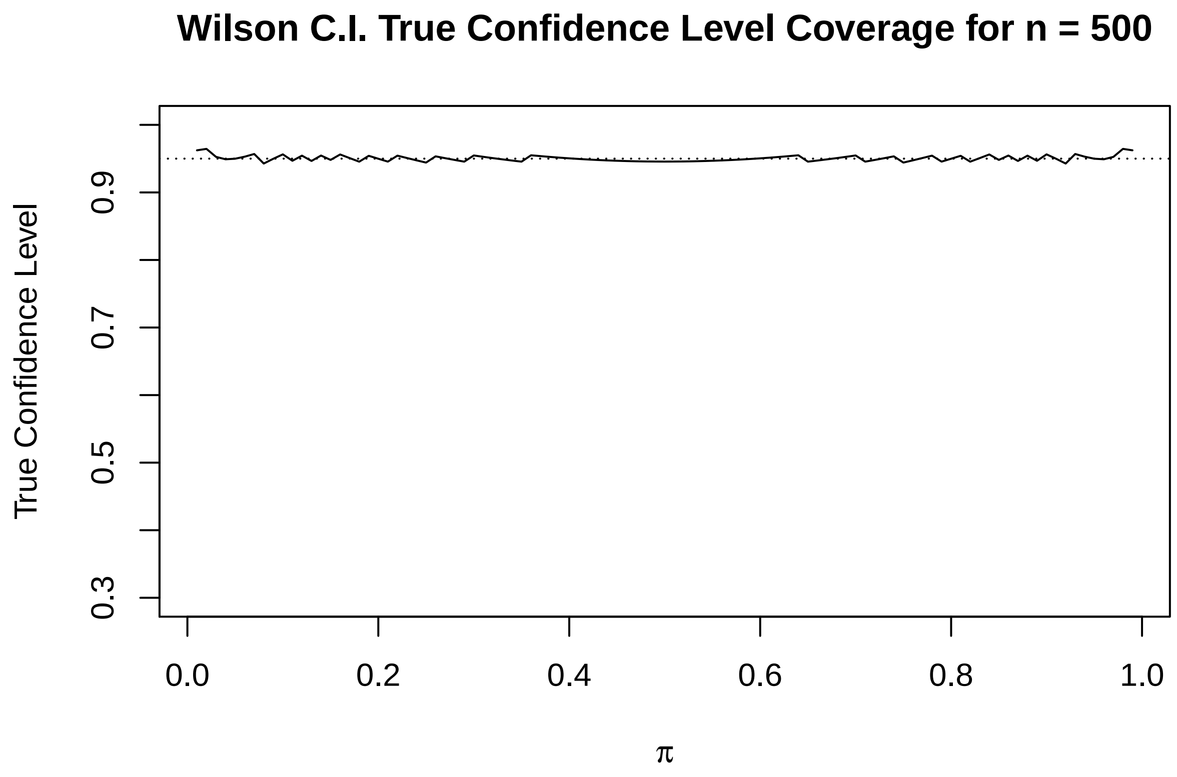
Question 1.2: (1) Modify the code for the Wilson Interval. (2) Do the exercise for `n=10, n=50, n=100, n=500`. (3) Plot the graphs. (4) Describe what you have observed from the results and compare the Wald and Wilson intervals based on your results. Use the same `pi.seq` as in the live session code.

```
# Code is modified for the Wilson confidence interval

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

    # There is a finite number of values w, the number of
    # successes, can take This ranges from 0 up to the number of
    # trials
    w = 0:n

    # calculate the probability of observing each w under the
    # true pi
    pmf = dbinom(x = w, size = n, prob = pi)

    # The critical value for stated alpha
    Z0 = qnorm(p = 1 - alpha/2)

    # This is the adjusted estimate pi used in Wilson CIs
    pi.adj.hat = (w + Z0^2/2)/(n + Z0^2)

    # This is the estimated pi.hat
    pi.hat = w/n

    # Wilson confidence interval calculated for each w
    Wilson.CI_lower.bound = pi.adj.hat - Z0 * sqrt(n)/(n + Z0^2) *
        sqrt(pi.hat * (1 - pi.hat) + Z0^2/(4 * n))
    Wilson.CI_upper.bound = pi.adj.hat + Z0 * sqrt(n)/(n + Z0^2) *
        sqrt(pi.hat * (1 - pi.hat) + Z0^2/(4 * n))

    # determine whether the CIs above contain the true pi
    covered.pi = ifelse(test = pi > Wilson.CI_lower.bound, yes = ifelse(test = pi <
        Wilson.CI_upper.bound, yes = 1, no = 0), no = 0)

    # the true confidence level is the sum of the probabilities
    # of the CIs which do contain the true pi
    Wilson.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(Wilson.CI.true.coverage)
}

Wilson.CI.true.coverage.pi.range <- function(n, pi.seq = seq(0.01,
    0.99, by = 0.01)) {
    # Computes the Wilson CI true coverage for a range of pi
    # values

    # Setup matrix for storing
    Wilson.CI.true.matrix = matrix(data = NA, nrow = length(pi.seq),
        ncol = 2)

    # calculate the wald CI true coverage for each true pi value
    # and store in matrix
    counter = 1
    for (pi in pi.seq) {
        true.coverage = Wilson.CI.true.coverage(pi = pi, alpha = 0.05,
            n = n)
        Wilson.CI.true.matrix[counter, ] = c(pi, true.coverage)
        # Make sure to use sum of wald.df = data.frame(w, pi.hat,
        # round(data.frame(pmf,
        # wald.CI_lower.bound,wald.CI_upper.bound),4), covered.pi)

        counter = counter + 1
    }
    return(Wilson.CI.true.matrix)
}

# Generate the plots for desired n values Same plotting
# function as for Wald still works
for (n in c(10, 50, 100, 500)) {
    n.Wilson.CI.true.matrix <- Wilson.CI.true.coverage.pi.range(n)
    plot.CI.true.coverage(n.Wilson.CI.true.matrix, main = paste0("Wilson C.I. ",
        "True Confidence Level Coverage for n = ", n), ylim = c(0.3,
        1))  #ylim set to same as Wald for better comparison
}
```




Wilson C.I. True Confidence Level Coverage for n = 50
True Confidence Level
0.3
0.5
0.7
0.9
0.0
0.2
0.4
0.6
0.8
1.0
π


Wilson C.I. True Confidence Level Coverage for n = 100
True Confidence Level
0.3
0.5
0.7
0.9
0.0
0.2
0.4
0.6
0.8
1.0
π



The Wilson CI performs much better than the Wald especially for extreme values of π. Even for a sample size of 10, the Wilson hovers around 0.95, and appears better on average than Wald at n=100. For each sample size, 50 and 100, Wilson performs slightly better than Wald on average for the entire range of values (even close to $\pi = 0.5$, Wilson zigzags above and below 0.95, while Wald is almost entirely below 0.95). At n=500, Wilson is better at extreme values of π but performs similarly to Wald at values closer to 0.5. The Wilson interval has a similar effect as parameter smoothing since the adjusted estimate for $\hat{\pi}$ adds a function of the critical value to both the number of successes and total n. This prevents the extreme behavior observed in the Wald CI.

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*.

Exercise 8

Continuing Exercise 7, use the Distance, Weather, Wind15, Temperature, Grass, Pressure, and Ice explanatory variables as linear terms in a new logistic regression model to predict Good.

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

```r
# Read in the data
placekick <- read.csv("data/placekick.BW.csv")
summary(placekick)
```

```
##       GameNum              Kicker     Good        Distance
##  2003-P201:  11   Feely     :  74   N: 438   Min.   :18.00
##  2002-1312:   9   Vinatieri :  74   Y:1565   1st Qu.:28.00
##  2003-0216:   9   Wilkins   :  73            Median :37.00
##  2003-0504:   9   Vanderjagt:  72            Mean   :36.35
##  2003-0507:   9   Akers     :  71            3rd Qu.:44.00
##  2002-0212:   8   GramaticaM:  71            Max.   :62.00
##  (Other)  :1948   (Other)   :1568
##      Weather         Wind15        Temperature     Grass        Pressure
##  Clouds  :717   Min.   :0.0000   Cold: 259   Min.   :0.000   N:1864
##  Inside  :385   1st Qu.:0.0000   Hot : 198   1st Qu.:0.000   Y: 139
##  SnowRain:171   Median :0.0000   Nice:1546   Median :1.000
##  Sun     :730   Mean   :0.1318               Mean   :0.653
##                 3rd Qu.:0.0000               3rd Qu.:1.000
##                 Max.   :1.0000               Max.   :1.000
##
##       Ice
##  Min.   :0.00000
##  1st Qu.:0.00000
##  Median :0.00000
##  Mean   :0.01897
##  3rd Qu.:0.00000
##  Max.   :1.00000
##
```

The variable Wind, Grass, and Ice variables are currently encoded as 0 and 1, rather than as factors Y and N. To be consistent with Good and Pressure, we will convert these to factors with levels Y and N to indicate that they are categorical variables rather than numeric. While this isn't necessary from a fitting perspective, it makes it clear when making predictions and constructing odds ratios which level of effect we are dealing with.

```r
placekick$Wind15 <- factor(ifelse(placekick$Wind15 == 1, "Y",
    "N"))
placekick$Grass <- factor(ifelse(placekick$Grass == 1, "Y", "N"))
placekick$Ice <- factor(ifelse(placekick$Ice == 1, "Y", "N"))
```

The explanatory variables are below:

Outcome variable: Good is represented as a binary indicator where GoodY = 1 is success and GoodY = 0 is failure in making the field goal.

Indicator variables:

1. Wind15Y with value 1 indicates that the wind was >= 15 miles per hour and the placekick is outdoors, 0 otherwise
2. GrassY with value 1 indicates that kick was attempted on a grass field, and 0 otherwise
3. PressureY is represented as 1 if the attempt was in the last 3 minutes of a game and a successful field goal causes a lead change, and 0 otherwise.
4. IceY is 1 if Pressure is also 1 and a time-out is called prior to the attempted kick; 0 otherwise

Categorical variables:

1. Weather has 4 levels, "Clouds", "Inside", "SnowRain", "Sun". The base level is Clouds, which happens when all other indicator levels are 0. WeatherInside = 1 indicates the game was played indoors. WeatherSnowRain = 1 indicates snow/rain, and WeatherSun = 1 indicates the weather was sunny.
2. Temperature has 3 levels: Base level "Cold" means less than 40 degrees F and outside of a dome, which happens when all indicator levels are 0; TemperatureNice = 1 means temperature was between 40 and 80 degrees F or inside of a dome; TemperatureHot means greater than 80 degrees F and outside of a dome.

The model we want to estimate is below.

$$\begin{aligned}\log \frac{\pi_{GoodY}}{1-\pi_{GoodY}} &= \beta_0 + \beta_1 * \text{Distance} + \beta_2 * \text{WeatherInside} \\ &+ \beta_3 * \text{WeatherSnowRain} + \beta_4 * \text{WeatherSun} \\ &+ \beta_5 * \text{Wind15Y} + \beta_6 * \text{TemperatureHot} \\ &+ \beta_7 * \text{TemperatureNice} + \beta_8 * \text{GrassY} \\ &+ \beta_9 * \text{PressureY} + \beta_{10} * \text{IceY}\end{aligned}$$

We estimate the model:

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

```
##
## Call:
```

```
## glm(formula = Good ~ Distance + Weather + Wind15 + Temperature +
##     Grass + Pressure + Ice, family = binomial(link = "logit"),
##     data = placekick)
##
## 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 .
## Wind15Y         -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
## GrassY          -0.328435   0.160050  -2.052   0.0402 *
## PressureY        0.270174   0.262809   1.028   0.3039
## IceY            -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
```

The model we've fit is:

$$\begin{aligned}\log \frac{\pi_{GoodY}}{1-\pi_{GoodY}} &= 5.740 - 0.110 * \text{Distance} - 0.083 * \text{WeatherInside} \\ &- 0.444 * \text{WeatherSnowRain} - 0.248 * \text{WeatherSun} \\ &- 0.244 * \text{Wind15Y} + 0.250 * \text{TemperatureHot} \\ &+ 0.235 * \text{TemperatureNice} - 0.328 * \text{GrassY} \\ &+ 0.271 * \text{PressureY} - 0.876 * \text{IceY}\end{aligned}$$

(b)

The authors use "Sun" as the base level category for Weather, which is not the default level R uses. Describe how "Sun" can be specified as the base level in R. Re-estimate the model

This is done using the relevel() function within R. This function takes as input a factor and a reference level, and changes the base level to the specified reference level. We can then store this into the dataframe as a re-ordered factor. The equation of the model now becomes:

$$\begin{aligned}\log \frac{\pi_{GoodY}}{1-\pi_{GoodY}} &= \beta_0 + \beta_1 * \text{Distance} + \beta_2 * \text{WeatherClouds} \\ &+ \beta_3 * \text{WeatherInside} + \beta_4 * \text{WeatherSnowRain} \\ &+ \beta_5 * \text{Wind15Y} + \beta_6 * \text{TemperatureHot} \\ &+ \beta_7 * \text{TemperatureNice} + \beta_8 * \text{GrassY} \\ &+ \beta_9 * \text{PressureY} + \beta_{10} * \text{IceY}\end{aligned}$$

```r
# re-level the Weather variable to have 'Sun' as the base
placekick.sun.base <- placekick
placekick.sun.base$Weather <- relevel(placekick.sun.base$Weather,
    "Sun")

# re-fit the model
mod.fit.sun_base <- glm(Good ~ Distance + Weather + Wind15 +
    Temperature + Grass + Pressure + Ice, family = binomial(link = "logit"),
    data = placekick.sun.base)
summary(mod.fit.sun_base)
```

```
##
## Call:
## glm(formula = Good ~ Distance + Weather + Wind15 + Temperature +
##     Grass + Pressure + Ice, family = binomial(link = "logit"),
##     data = placekick.sun.base)
##
## 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
## Wind15Y         -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
## GrassY          -0.328435   0.160050  -2.052   0.0402 *
## PressureY        0.270174   0.262809   1.028   0.3039
## IceY            -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
```

The coefficients for only the Weather levels have changed. This is as expected because the variables used for both models are exactly the same. The Weather variables have changed because all levels are now evaluated against Sun (notice that the Weather_sun_baseClouds is the negative of the previous coefficient on WeatherSun).

The fit model is:

$$\begin{aligned}\log\frac{\pi_{GoodY}}{1-\pi_{GoodY}} &= 5.493 - 0.110 * \text{Distance} + 0.248 * \text{WeatherClouds}\\ &+ 0.165 * \text{WeatherInside} - 0.197 * \text{WeatherSnowRain}\\ &- 0.244 * \text{Wind15Y} + 0.250 * \text{TemperatureHot}\\ &+ 0.235 * \text{TemperatureNice} - 0.328 * \text{GrassY}\\ &+ 0.271 * \text{PressureY} - 0.876 * \text{IceY}\end{aligned}$$

(c)

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

We will perform Type II LRTs for each variable using the Anova function from the car package. This function calculates a ChiSq statistic based on the ratio of maximum likelihoods of the model without the variable of interest, each time comparing to the full model. For all of the variables, we will conduct a hypothesis test at a significance level of $\alpha = 0.05$. **The null hypothesis is that the coefficient(s) corresponding to the variable is 0**. For example, **for Weather**, the null and alternative hypotheses is as follows:

$$\begin{aligned}H_0 :&\beta_2 = \beta_3 = \beta_4 = 0\\ H_a :&\text{not all of } \beta_2, \beta_3, \beta_4 \text{ are } 0\end{aligned}$$

π here represents probability that the placekick is Good = 1 (success). We perform the test below:

```r
library(car)
```

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

```r
Anova(mod.fit)
```

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

Based on the p-values of the LRTs, the variable is Distance is highly statistically significant, with p-value $<< 0.01$. Therefore, we reject the null hypothesis that the variable is not important in the model, and conclude that there is strong evidence Distance is important given that all of the other stated variables are included in the model.

The variable Grass is statistically significant with p-value = 0.038. We reject the null hypothesis and conclude that there is evidence Grass is important given that all of the other stated variables are included in the model.

The variable Ice is marginally important, with p-value = 0.054. Since we previously stated the significance level at 0.05, we fail to reject our null hypothesis that the variable is statistically insignificant for the model, given that the other variables are included in the model.

All other variables have p-values > 0.1. We fail to reject our null hypothesis for all of these other variables, and conclude each of these variables are statistically insignificant given that the rest of variables are specified in the model.

While we could re-estimate the model with only Grass and Distance, this could produce a worse model than what the LRT tests suggest. This is because the test for each individual variable is in the context of all other variables being in the model. For example, Pressure might only be insignificant gievn that Temperature is in the model, in which case we would not want to remove both variables. We could test for joint significances by re-defining models without these parameters. For example, if we wanted to test if Temperature and Pressure are jointly significant, we would specify a model without both, and perform LRT using (lowercase) anova. However, this is outside of the scope of this problem. We will not be performing such tests, and so we will keep the current model moving forward.

(d)

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

As demonstrated in the lectures for the field kick example, an odds ratio for a distance of 10 yards makes sense in the context of the game (1-yard for example is too small). Also, since the coefficient on distance is negative, that means that the odds increases as distance decreases. An intepretation in terms of increasing odds makes the most intuitive sense. Therefore, we choose c = -10 and calculate the OR as follows (where x1 is distance):

$$OR = \frac{Odds_{x_1-10}}{Odds_{x_1}} = e^{-10*\beta_1}$$

```r
exp(-10 * mod.fit$coefficients[2])
```

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

Based on the OR calculation, the estimated odds of a successful field goal changes (increases) by about 2.99 times for every 10 yard decrease in distance, holding all other variables constant.

To get a confidence interval for the OR, we first calculate a confidence interval for the coefficient β_1, then exponentiate. We have two choices for the type of CI we can calculate: Wald or profile LR. The Wald CI doesn't always work well (it often has a confidence level lower than the true confidence level) especially for small sample sizes. Since we have 2003 samples, this is likely large enough for the Wald CI. While profile LR is typically better, we will calculate both and compare, and if they are similar use the numbers from the LR.

```r
# Wald CI for beta0
beta0.wald <- confint.default(mod.fit, parm = "Distance", level = 0.95)

# Wald CI for OR
beta0.wald.CI <- as.numeric(rev(exp(-10 * beta0.wald)))

# LR CI for beta0
beta0.LR <- confint(mod.fit, parm = "Distance", level = 0.95)
```

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

```r
# CI for OR
beta0.LR.CI <- as.numeric(rev(exp(-10 * beta0.LR)))

# We will use stargazer to display nice tables
library(stargazer)
```

```
##
## Please cite as:

##  Hlavac, Marek (2018). stargazer: Well-Formatted Regression and Summary Statistics Tables.

##  R package version 5.2.2. https://CRAN.R-project.org/package=stargazer
```

```r
stargazer(data.frame(lower.CI = c(beta0.wald.CI[1], beta0.LR.CI[1]),
    upper.CI = c(beta0.wald.CI[2], beta0.LR.CI[2]), row.names = c("Wald",
        "Profile LR")), summary = FALSE, type = "text")
```

```
##
## ============================
##            lower.CI upper.CI
## ----------------------------
## Wald        2.599    3.445
## Profile LR  2.605    3.454
## ----------------------------
```

We can see that the two intervals are very similar, up to the second decimal point, and both lead to the same conclusions. Under the Profile LR confidence interval, with 95% confidence, the estimated odds of a successful placekick change by an amount between 2.61 and 3.45 times for every 10-yard decrease in distance. This interval does not contain 1, meaning there is sufficient evidence that a 10-yard decrease in distance increases the odds of success. This is consistent with the hypothesis test for the Distance variable stated earlier.

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

```
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:car':
##
##     recode

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
admissions <- read.table("data/admissions.csv", header = TRUE,
    sep = ",", row.names = "X")
glimpse(admissions)
```

```
## Observations: 400
## Variables: 4
## $ admit <int> 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,...
## $ gre   <int> 380, 660, 800, 640, 520, 760, 560, 400, 540, 700, 800, 4...
## $ gpa   <dbl> 3.61, 3.67, 4.00, 3.19, 2.93, 3.00, 2.98, 3.08, 3.39, 3....
## $ rank  <int> 3, 3, 1, 4, 4, 2, 1, 2, 3, 2, 4, 1, 1, 2, 1, 3, 4, 3, 2,...
```

```
Hmisc::describe(admissions)
```

```
## admissions
##
##  4  Variables      400  Observations
## ---------------------------------------------------------------------------
## 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
## ---------------------------------------------------------------------------
```

First of all, we can see that the dataset is fairly clean with no missing values for any of the variables. However, all values are numeric. The response "admit" should be factor, because it is a binary 1 (admitted/success) or 0 (rejected/failure).

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

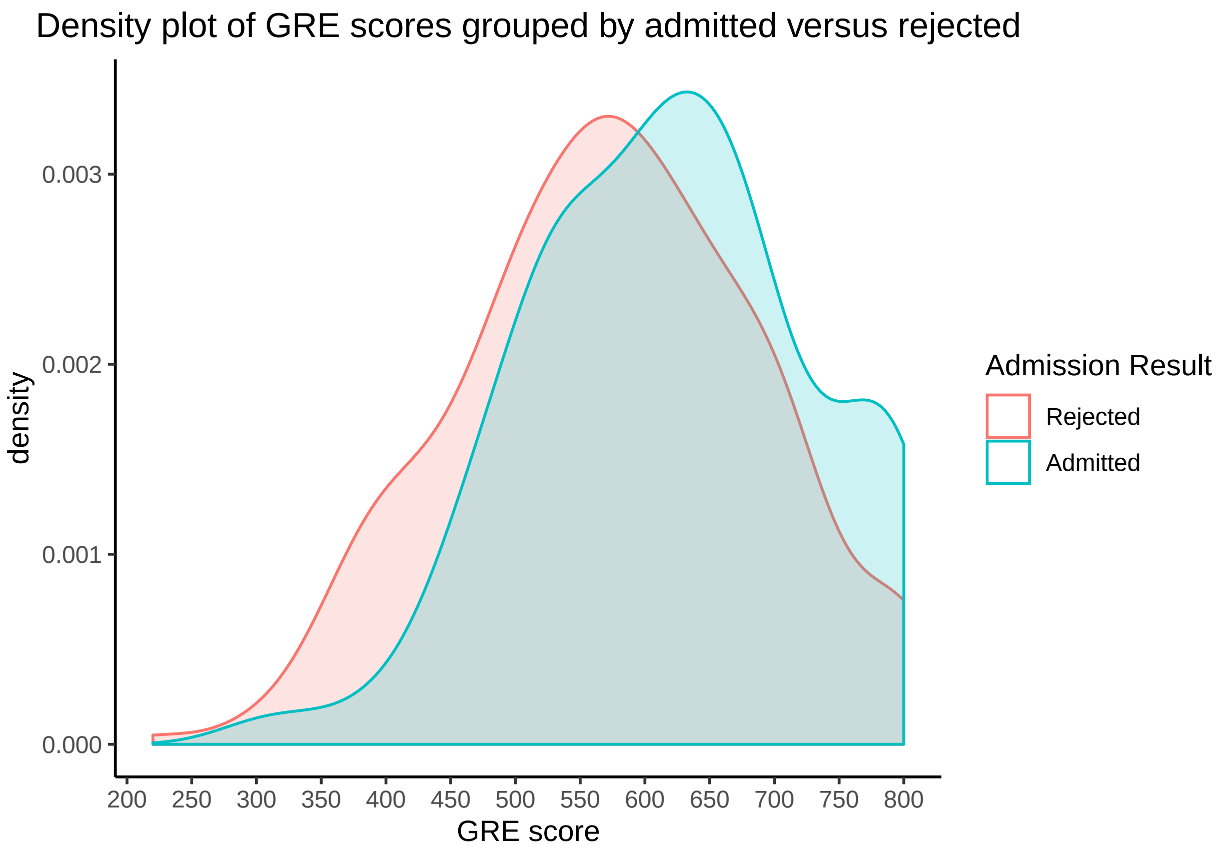
For the dependent variable admit, we have 127 cases of students being admitted (sum of the 1 values), and by implication 373 students who were not admitted. While there are almost 3 times as many students rejected as admitted, this is still a relatively balanced dataset in terms of the two categories.

We will now perform further EDA on each of the explanatory variables.

```
# EDA for the GRE variable

rbind(admitted = summary(subset(admissions, admit == 1)$gre),
    rejected = summary(subset(admissions, admit == 0)$gre))
```

```
##          Min. 1st Qu. Median     Mean 3rd Qu. Max.
## admitted  300     540    620 618.8976     680  800
## rejected  220     500     580 573.1868     660  800
```

```
ggplot(admissions, aes(x = gre, color = factor(admit), fill = factor(admit))) +
    geom_density(alpha = 0.2) + labs(x = "GRE score", color = "Admission Result",
    y = "density", title = "Density plot of GRE scores grouped by admitted versus rejected") +
    scale_color_hue(labels = c("Rejected", "Admitted")) + guides(fill = FALSE) +
    theme_classic() + theme(plot.title = element_text(hjust = 0.5)) +
    scale_x_continuous(breaks = seq(200, 800, by = 50))
```

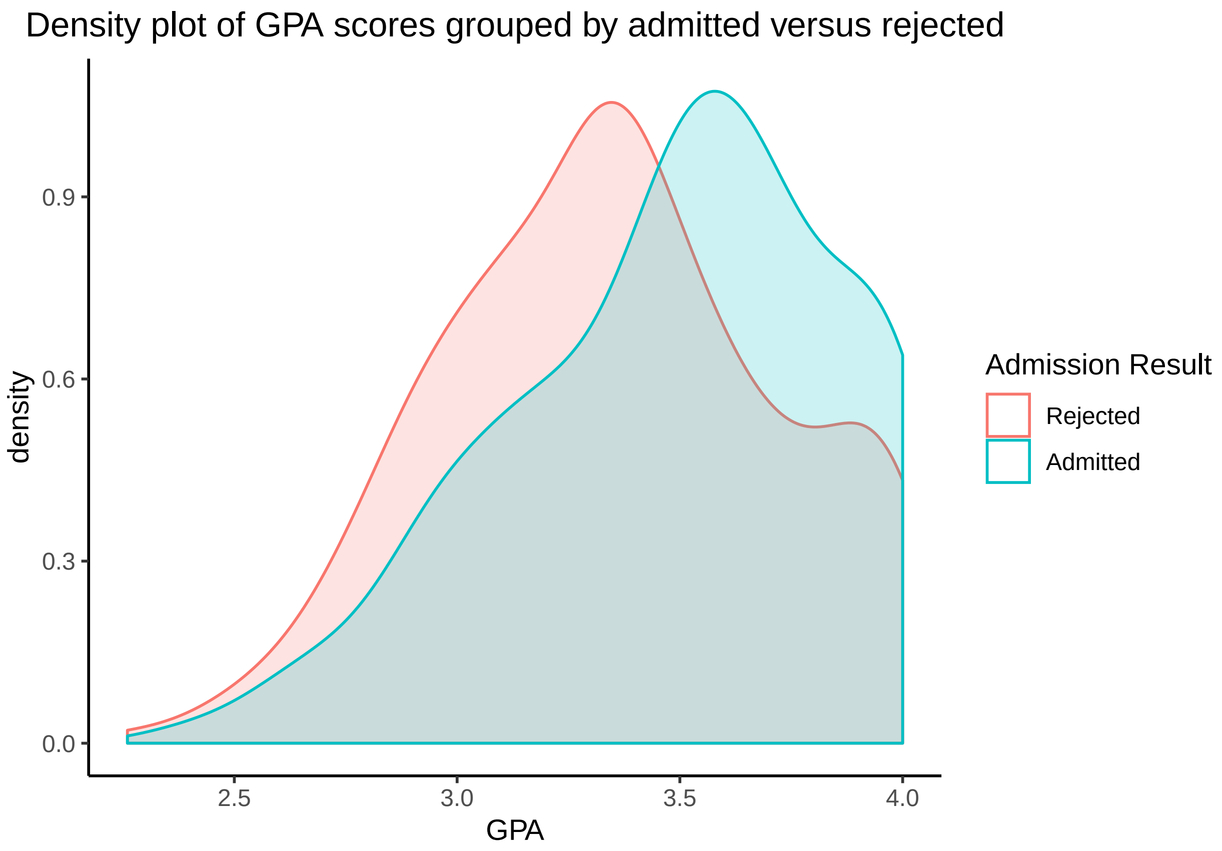


The GRE variables ranges between 200 and 800, which makes sense since this is the range of possible GRE scores. Based on the summary above, we see that for both the pool of applicants who were admitted and rejected, the range for the GRE scores are both very large (spanning ~500+ points). The median and mean are both higher for admitted applicants. Based on the density plot, while separation between the two pools exists, there is also significant overlap between the two distributions. This analysis suggests that GRE alone will not have enough predictive power, but in the context of other variables may be useful.

```
# EDA for the GPA variable

rbind(admitted = summary(subset(admissions, admit == 1)$gpa),
    rejected = summary(subset(admissions, admit == 0)$gpa))
```

```
##          Min. 1st Qu. Median     Mean 3rd Qu. Max.
## admitted 2.42    3.22   3.54 3.489213   3.755    4
## rejected 2.26    3.08   3.34 3.343700   3.610    4
```

```r
ggplot(admissions, aes(x = gpa, color = factor(admit), fill = factor(admit))) +
    geom_density(alpha = 0.2) + labs(x = "GPA", color = "Admission Result",
    y = "density", title = "Density plot of GPA scores grouped by admitted versus rejected") +
    scale_color_hue(labels = c("Rejected", "Admitted")) + guides(fill = FALSE) +
    theme_classic() + theme(plot.title = element_text(hjust = 0.5)) +
    scale_x_continuous(breaks = seq(2, 4, by = 0.5))
```

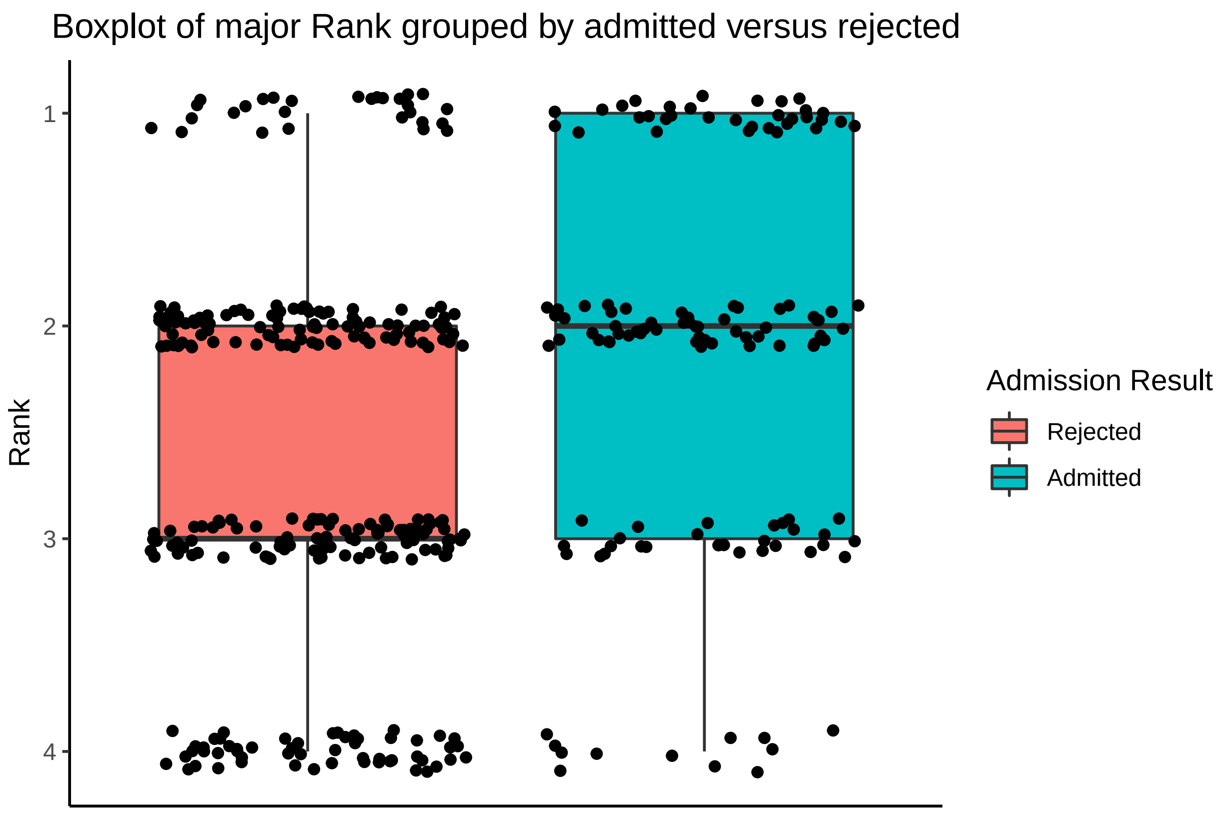


The mean and median are both higher for admitted student than rejected students. While the overlap in the density distributions are again significant, there appears to be a shift in the density peak to higher GPA for admitted students. Namely, between GPA points 3.5 and 4, the density for admitted students is much higher than for rejected. This suggests that GPA even alone might have signal especially for students with very high GPAs, and will likely be useful in conjunction with other variables.

```r
# EDA for the rank variable

rbind(admitted = summary(subset(admissions, admit == 1)$rank),
    rejected = summary(subset(admissions, admit == 0)$rank))
```

```
##          Min. 1st Qu. Median     Mean 3rd Qu. Max.
## admitted    1       1      2 2.149606       3    4
## rejected    1       2      3 2.641026       3    4
```

```r
ggplot(admissions, aes(x = admit, y = rank)) + geom_boxplot(aes(fill = factor(admit))) +
    geom_jitter(height = 0.1) + labs(x = "", fill = "Admission Result",
    y = "Rank", title = "Boxplot of major Rank grouped by admitted versus rejected") +
    scale_fill_discrete(labels = c("Rejected", "Admitted")) +
    scale_y_reverse() + theme_classic() + theme(plot.title = element_text(hjust = 0.5),
    axis.text.x = element_blank(), axis.ticks.x = element_blank())
```



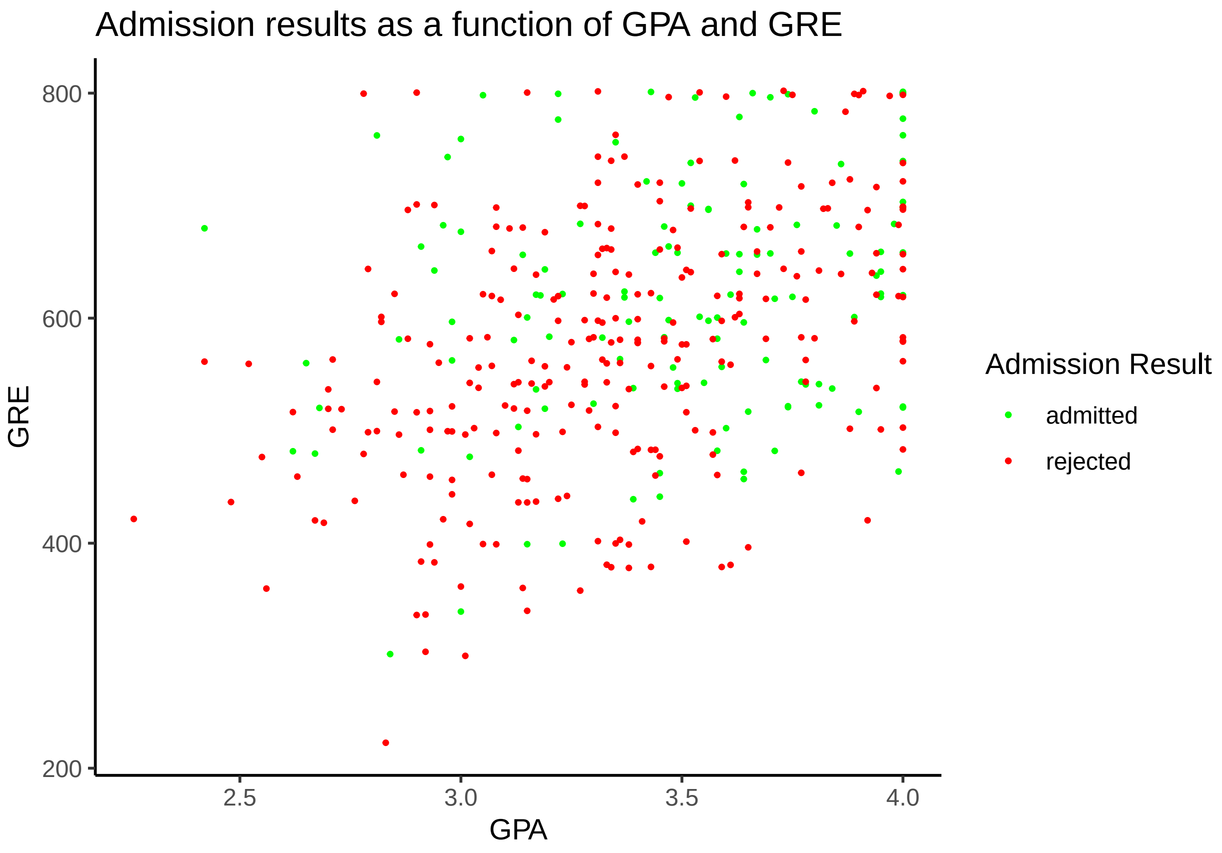
Based on the summary above, rank can only take on 4 values, from 1 to 4 for both groups (admitted versus rejected). This suggests that the measurement is based on quartile of all students in the major rather than absolute rank (unless only students in the top 4 of their class was sampled, which would be highly unlikely for a random sample). For example, 1 means top 25%, and 4 bottom 25%. Rank 1 is considered to be better than 4, and admitted students have a higher median rank (2) than rejected students. From the boxplot (note the reverse order of the y-axis), we can see that the second quartile for admitted students is the same as the upper quartile for rejected students. Also, the ratio of number of points at rank 4 going up to rank 1 for admitted:reject increases consistently, indicating that a higher proportion of rank 1 students were admitted than rank 4. There appears to be a clear difference in rank between admitted and rejected students, where the rank appears to be higher for admitted students.

Note that a bit of jitter was added to the rank in both height and width directions in order to distinguish each student.

Let's finally look at a tri-variate analysis of GRE and GPA. For this, we will plot GRE and GPA on separate axis, and for each example, look at where they fall on the grid. Note that we add a small amount of jitter for GRE in order to distinguish between overlapping points.

```r
# separate admitted versus rejected into different dataframes
admissions.admitted <- subset(admissions, admit == 1)
admissions.rejected <- subset(admissions, admit == 0)
```

```
ggplot() + geom_point(data = admissions.admitted, aes(x = gpa,
    y = jitter(gre), color = "admitted"), size = 0.5) + geom_point(data = admissions.rejected,
    aes(x = gpa, y = jitter(gre), color = "rejected"), size = 0.5) +
    labs(x = "GPA", y = "GRE", title = "Admission results as a function of GPA and GRE",
        color = "Admission Result") + theme(plot.title = element_text(hjust = 0.5)) +
    theme_classic() + scale_color_manual(values = c("green",
    "red"))
```



The trend looks fairly random, with the exception of students with both low GPA and GRE. There is a higher density of rejections for GPA < 3 and GRE < 600, and it is not entirely evident whether there is higher acceptance rates for students with high GPA and high GRE. Based on this plot, we cannot conclude anything sustantial about the joint importance of GRE and GPA.

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

Since our dataset is clean, we can directly fit the model below. Note that we treat rank as a categorical variable rather than numeric variable as explained in the EDA. We will use rank 1 as the base level, three categorical variables representing the other three levels. The Admit variable is binary, with Admit1 $=1$ when the student was admitted, and 0 otherwise.

The model we want to estimate is:

$$\log\frac{\pi_{admit1}}{1-\pi_{admit1}} = \beta_0+\beta_1*\text{gre}+\beta_2*\text{gpa}+\beta_3*\text{rank2}+\beta_4*\text{rank3}+\beta_5*\text{rank4}+\beta_6*\text{gre}^2+\beta_7*\text{gpa}^2+\beta_8*\text{gre} * \text{gpa}$$

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

```
##
## Call:
## glm(formula = admit ~ gre + gpa + factor(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
## factor(rank)2 -7.130e-01  3.202e-01  -2.227 0.025958 *
## factor(rank)3 -1.341e+00  3.474e-01  -3.861 0.000113 ***
## factor(rank)4 -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
```

The model we have fit is:

$$\log\frac{\pi_{admit1}}{1-\pi_{admit1}} = -7.33 + 0.0186 * \text{gre} - 0.178 * \text{gpa} - 0.713 * \text{rank2} - 1.34 * \text{rank3} - 1.56 * \text{rank4}$$
$$- 3.07 * 10^{-6} * \text{gre}^2 + 0.670 * \text{gpa}^2 - 0.0059 * \text{gre} * \text{gpa}$$

Based on the summary from the Wald test, rank is highly statistically significant (since all levels compared to the base are statistically significant) given that all other variables are in the model.

The other variables and interaction terms, if removed individually given that all other terms remain in the model, are not statistically significant at a significant level of 0.05. The coefficient for GRE^2 is also close to 0.

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

We can perform Type II LRTs for each variable using the (lowercase) anova function from the car package. This function calculates a ChiSq statistic based on the ratio of maximum likelihoods of the model without the variable of interest, and the full model. In this case, we are not for example testing whether GRE is significant given its interaction with GPA (which is a Type III test). We are interested in whether GRE has an effect at all, meaning we want to see if all coefficients involving GRE is statistically not significant. As a result, we need to first specify an alternative model without GRE, and test against the full model. We test at a significance level of 0.05, and the stated hypothesis test is as follows:

$$H_0 : \beta_1 = \beta_6 = \beta_8 = 0$$
$$H_a : \text{not all of } \beta_1, \beta_6, \beta_8 \text{ are } 0$$

```r
admit.no.gre <- glm(admit ~ gpa + factor(rank) + I(gpa^2), family = binomial(link = "logit"),
    data = admissions)
anova(admit.no.gre, admit.fit, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: admit ~ gpa + factor(rank) + I(gpa^2)
## Model 2: admit ~ gre + gpa + factor(rank) + I(gre^2) + I(gpa^2) + gre:gpa
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       394     462.74
## 2       391     454.90  3   7.8421  0.04939 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Based on the LRT, since the p-value < 0.05, the GRE variable, its second term, and interaction term with GPA are jointly significant (though only marginally so since the value is close to 0.05). We reject the hypothesis that model does not contain the GRE terms and conclude that we have evidence GRE has a statistically significant effect on admissions. Given that the residual deviance of model 2 is lower than model 1, this is one way to see that model 2, with the inclusion of all GRE terms, is better at predicting admissions outcome. We can also look at the AIC:

```r
data.frame(Ha.model = AIC(admit.fit), H0.model = AIC(admit.no.gre),
    row.names = "AIC")
```

```
##     Ha.model H0.model
## AIC 472.8955 474.7376
```

The conclusion with the AIC is the same, since the Ha model has lower AIC than H0, although not by much. As a result, we conclude the model under Ha is statistically different than the model under H0, and that it is a better model based on AIC and residual deviance by including the GRE terms.

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

We can estimate the effect of college GPA on admissions by looking at the odds ratio for a sensible change in GPA. Since GPA ranges from a scale of 0 to 4.3333 (let's say A+ counts since we're at Berkeley, which awards A+), a 1 unit increase is too large of a change. We will consider a 0.3 unit increase in GPA (for GPA less than 4.0), which roughly corresponds to an average increase of a letter grade level (for example, from B+ to A-). The odds ratio is for these parameters is:

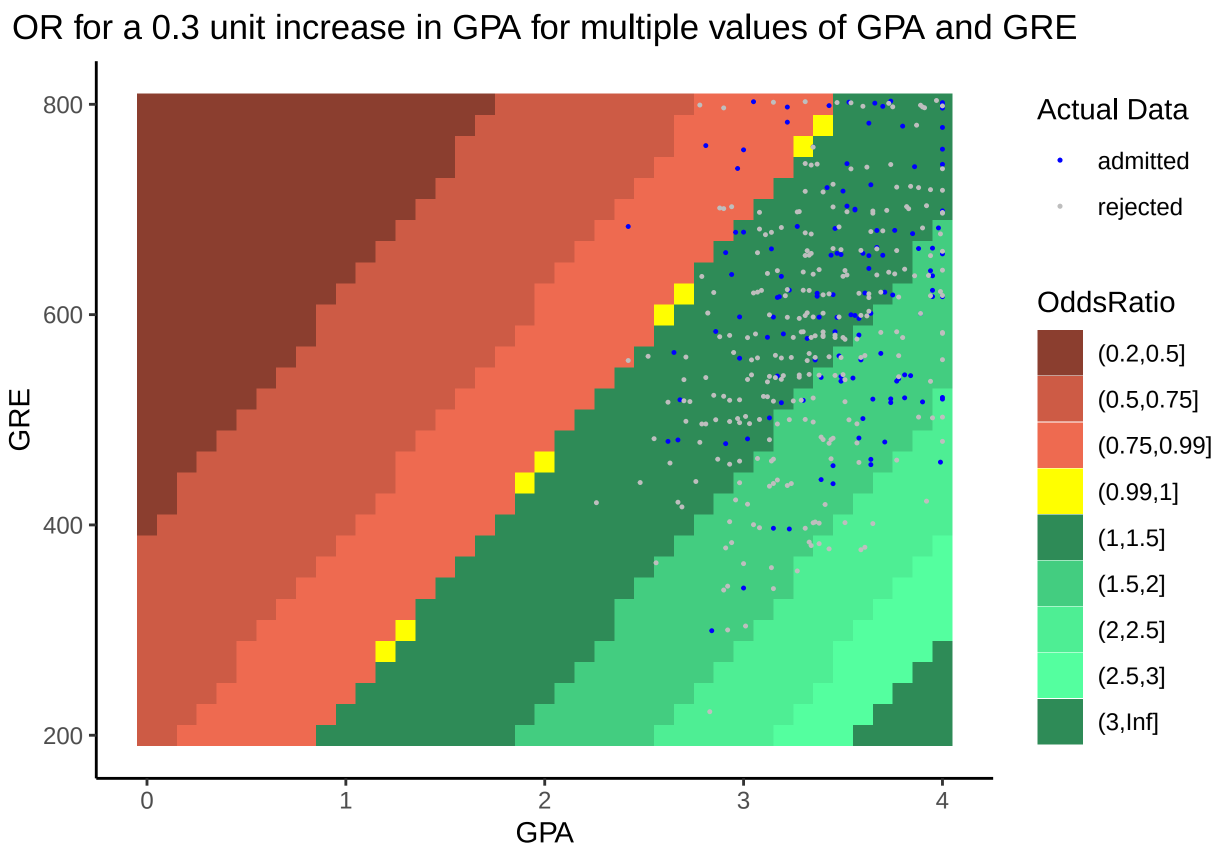
β_2 is coefficient in front of gpa. β_7 is coefficient on gpa^2. β_8 is coefficient on gre:gpa.

$$\begin{aligned} OddsRatio &= \frac{Odds_{gpa+0.3}}{Odds_{gpa}} \\ &= e^{\beta_2*(0.3)+\beta_7*(2*0.3*gpa+0.3^2)+\beta_8*gre*(0.3)} \end{aligned}$$

As a result, the OR is a function of both GPA and GRE. We will plot a heatmap of this, with the colors representing OR (binned). We will also plot our data on this plot to see which region of the GPA/GRE landscape our data populates.

```r
or.gpa <- function(gpa, gre) {
    lp <- admit.fit$coefficients["gpa"] * (0.3) + admit.fit$coefficients["I(gpa^2)"] *
        (2 * 0.3 * gpa + 0.3^2) + admit.fit$coefficients["gre:gpa"] *
        gre * (0.3)
    return(exp(lp))
}

# We will use GPA as values of 0.1
gpa_range <- seq(0, 4, 0.1)

# We will use GRE at increments of 20
gre_range <- seq(200, 800, 20)

grid_points <- expand.grid(gpa = gpa_range, gre = gre_range)
grid_points$or <- or.gpa(grid_points$gpa, grid_points$gre)

# Change the OR to a factor to clearly display different
# colors for groups of OR
grid_points$OddsRatio <- cut(grid_points$or, breaks = c(0.2,
    0.5, 0.75, 0.99, 1, 1.5, 2, 2.5, 3, Inf))

# For plotting the OR data, I will use a heat map. On top of
# this heatmap, I will draw a point for every example that
# exists in our dataset (with y-scale jitter to differentiate
# some of the points). To differentiate admitted versus not
# admitted, I will use different color dots

# separate admitted versus rejected into different dataframes
admissions.admitted <- subset(admissions, admit == 1)
admissions.rejected <- subset(admissions, admit == 0)
```

```r
ggplot() + geom_tile(data = grid_points, aes(x = gpa, y = gre,
    fill = OddsRatio)) + scale_fill_manual(values = c("coral4",
    "coral3", "coral2", "yellow", "seagreen4", "seagreen3", "seagreen2",
    "seagreen1", "seagreen"), aesthetics = "fill") + labs(x = "GPA",
    y = "GRE", title = "OR for a 0.3 unit increase in GPA for multiple values of GPA and GRE",
    color = "Actual Data") + theme_classic() + geom_point(data = admissions.admitted,
    aes(x = gpa, y = jitter(gre), color = "admitted"), size = 0.2) +
    geom_point(data = admissions.rejected, aes(x = gpa, y = jitter(gre),
        color = "rejected"), size = 0.2) + theme(plot.title = element_text(hjust = 0.5)) +
    scale_color_manual(values = c("blue", "gray"))
```



We see that a 0.3 increase in GPA has mostly a predicted positive effect on admissions odds for all candidates in the dataset (represented by dots). Overall, the lower the person's GPA, the smaller the OR, and the higher the GRE, the smaller the OR. The second part makes sense: a unit change in GPA will more strongly boost the odds of someone with a weak GRE profile compared to someone with a strong GRE profile. We can generate a smaller table of 5 example GPA values for 3 different example GRE scores:

```r
example_gpa <- c(2.5, 3.2, 3.6, 3.9, 4)
example_gre <- c(650, 700, 750)

grid <- expand.grid(gpa = example_gpa, gre = example_gre)
grid$OddsRatio <- or.gpa(grid$gpa, grid$gre)
```

```
stargazer(grid, type = "text", summary = FALSE, rownames = FALSE)
```

```
##
## ===================
## gpa   gre OddsRatio
## -------------------
## 2.500 650   0.873
## 3.200 650   1.156
## 3.600 650   1.358
## 3.900 650   1.532
## 4     650   1.594
## 2.500 700   0.799
## 3.200 700   1.058
## 3.600 700   1.243
## 3.900 700   1.402
## 4     700   1.460
## 2.500 750   0.731
## 3.200 750   0.969
## 3.600 750   1.138
## 3.900 750   1.284
## 4     750   1.336
## -------------------
```

For example, consider someone with a 3.6 GPA and a 650 GRE. The estimated odds of admissions changes by 1.358 times for every 0.3 increase in that individual's GPA. Consider someone with a 2.5 GPA and 750 GRE score. Based on the model, the estimated odds of admissions increases by $\frac{1}{0.731} = 1.370$ times for a 0.3 decrease in that individual's GPA. This is probably not true in reality, but an artifact of that fact that we have very few training points with GPA that low, and GRE that high. In fact, the entire red region is unlikely in reality. One rarely sees GPA below 2.5, and for those that do have such a low GPA, scoring above 600 on GRE (red region) is also less likely.

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

To form a confidence interval for admission probability, we will follow the procedure suggested in the lectures:

1. Construct the Wald CI
2. Construct the profile LR
3. If the two are similar, then use the LR

We will do all of this within the mcprofile package. The probability of success is given by:

$$\pi = \frac{e^{\beta_0+\beta_1 * \text{gre}+\beta_2 * \text{gpa}+\beta_3 * \text{rank2}+\beta_4 * \text{rank3}+\beta_5 * \text{rank4}+\beta_6 * \text{gre}^2+\beta_7 * \text{gpa}^2+\beta_8 * \text{gre} * \text{gpa}}}{1+e^{\beta_0+\beta_1 * \text{gre}+\beta_2 * \text{gpa}+\beta_3 * \text{rank2}+\beta_4 * \text{rank3}+\beta_5 * \text{rank4}+\beta_6 * \text{gre}^2+\beta_7 * \text{gpa}^2+\beta_8 * \text{gre} * \text{gpa}}}$$

Therefore, we need the full linear combination of parameters.

```
library(mcprofile)
gpa <- 3.3
gre <- 720
```

```
rank <- c(0, 0, 0)  #since this individual has rank 1

# This matrix represents the values multiplied to each
# coefficient For example, beta_0 is the intercept (so
# multiplied by 1 as the first term), And beta_5 is
# multiplied by the value of gpa^2

K <- matrix(data = c(1, gre, gpa, rank, gre^2, gpa^2, gre * gpa),
    nrow = 1)
linear.combo <- mcprofile(admit.fit, CM = K)

# Wald confidence interval
save.wald <- wald(linear.combo)
ci.wald.logit <- confint(save.wald, level = 0.95, adjust = "none")
wald.ci <- round(exp(ci.wald.logit$confint)/(1 + exp(ci.wald.logit$confint)),
    3)

# Profile LR confidence interval
ci.lr.logit <- confint(linear.combo, level = 0.95, adjust = "none")
lr.ci <- round(exp(ci.lr.logit$confint)/(1 + exp(ci.lr.logit$confint)),
    3)

rbind(wald_ci = wald.ci, profile_lr_ci = lr.ci)
```

```
##                lower upper
## wald_ci        0.434 0.735
## profile_lr_ci  0.436 0.737
```

The CIs are almost exactly the same for both estimates, so we will use the profile LR. The interpretation is that with 95% confidence, the probability that this individual will be admitted is between 0.436 and 0.737. Since 0.5 is contained in this interval, we do not have sufficient evidence to suggest that this person is more likely to be admitted, or more likely to be rejected.

4. Binary Logistic Regression (2 points)

Linear Probability Model

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

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.

```r
Hmisc::describe(Mroz)
```

```
## Mroz
##
##  8  Variables      753  Observations
## ---------------------------------------------------------------------------
## lfp
##        n  missing distinct
##      753        0        2
##
## Value         no   yes
## Frequency    325   428
## Proportion 0.432 0.568
## ---------------------------------------------------------------------------
## k5
##        n  missing distinct     Info     Mean      Gmd
##      753        0        4    0.475   0.2377   0.3967
##
## Value          0     1     2     3
## Frequency    606   118    26     3
## Proportion 0.805 0.157 0.035 0.004
## ---------------------------------------------------------------------------
## k618
##        n  missing distinct     Info     Mean      Gmd
##      753        0        9    0.932    1.353     1.42
##
## Value          0     1     2     3     4     5     6     7     8
## Frequency    258   185   162   103    30    12     1     1     1
## Proportion 0.343 0.246 0.215 0.137 0.040 0.016 0.001 0.001 0.001
## ---------------------------------------------------------------------------
## age
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      753        0       31    0.999    42.54    9.289     30.6     32.0
##      .25      .50      .75      .90      .95
##     36.0     43.0     49.0     54.0     56.0
##
## lowest : 30 31 32 33 34, highest: 56 57 58 59 60
## ---------------------------------------------------------------------------
## wc
##        n  missing distinct
##      753        0        2
##
## Value         no   yes
## Frequency    541   212
## Proportion 0.718 0.282
## ---------------------------------------------------------------------------
## hc
##        n  missing distinct
##      753        0        2
##
## Value         no   yes
## Frequency    458   295
## Proportion 0.608 0.392
## ---------------------------------------------------------------------------
## lwg
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      753        0      676        1    1.097   0.6151   0.2166   0.4984
##      .25      .50      .75      .90      .95
##   0.8181   1.0684   1.3997   1.7600   2.0753
##
## lowest : -2.054124 -1.822531 -1.766441 -1.543298 -1.029619
## highest:  2.905078  3.064725  3.113515  3.155581  3.218876
## ---------------------------------------------------------------------------
## inc
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      753        0      621        1    20.13    11.55    7.048    9.026
##      .25      .50      .75      .90      .95
##   13.025   17.700   24.466   32.697   40.920
##
## lowest : -0.029  1.200  1.500  2.134  2.200, highest: 77.000 79.800 88.000 91.000 96.000
## ---------------------------------------------------------------------------
```

We note that wc/hc are both factor variables, and we assign wcyes = 1 when the wife attended college and hcyes = 1 when the husband attended college. The rest of the data is clean with no missing values.

The model we want to estimate is:

$$\pi_{lfp} = \beta_0 + \beta_1 * \text{k5} + \beta_2 * \text{k618} + \beta_3 * \text{age} + \beta_4 * \text{wcyes} + \beta_5 * \text{hcyes} + \beta_6 * \text{lwg} + \beta_7 * \text{inc}$$

In order to estimate an LPM, we need to convert lfp from two categories into 0 and 1 (0 meaning no, 1 meaning yes). The sync material converted to 2 and 1 rather than 1 and 0, but 1 (success) and 0 (failure) makes more sense:

```r
Mroz$lfp_numeric <- ifelse(Mroz$lfp == "yes", yes = 1, no = 0)
mroz.lm <- lm(lfp_numeric ~ k5 + k618 + age + wc + hc + lwg +
    inc, data = Mroz)
summary(mroz.lm)
```

```
##
## Call:
## lm(formula = lfp_numeric ~ 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)  1.143548   0.127053   9.001  < 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
```

The adjusted R-squared is very low, at 0.14, meaning this model would not be good for making predictions.

The model we've fit is:

$$\pi_{lfp} = 1.14 - 0.295 * \text{k5} - 0.011 * \text{k618} - 0.013 * \text{age} + 0.164 * \text{wcyes} + 0.019 * \text{hcyes} + 0.123 * \text{lwg} - 0.007 * \text{inc}$$

Interpretation:

According to the LPM, the coefficients represent the change in probability of labor force participation for a unit change in the coefficients **holding other variables in the model constant**, except the log transformed coefficient which has a percentage interpretation:

1. For each additional child under 5, the probability decreases by 0.295, holding other variables constant
2. For each additional child between ages 6 and 18, the probability decreases by 0.011, holding other variables constant
3. For each additional year in wife's age, the probability decreases by 0.013, holding other variables constant
4. For each additional year of wife's college attendance, the probability increases by 0.164, holding other variables constant
5. For each additional year of husband's college attendance, the probability increases by 0.019, holding other variables constant

6. For a 10% increase in wage, the probability increases by approximately $0.123 * log(1.1) = 0.012$, holding other variables constant
7. For each additional thousand of family income excluding wife's, the probability decreases by -0.007, holding other variables constant

CLM assumptions and Model Diagnostics:

CLM 1 assumes that the probability is linearly dependent on the explanatory variables with the inclusion of an error term.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k + u$$

This is already flawed because the model does not bound y in any way, whereas we know that probability is bounded between 0 and 1. This means that for the model to be true, the error term must compensate (which violates CLM 4 property below).

CLM 2: Random Sampling

The model requires that the samples represent a random sample of the population of interest. Since we don't have background knowledge of data collection, this is difficult to ascertain. The Mroz paper states that the dataset comes from PSID, which is a reputable household survey organization that focuses on collecting quality data, so we can safely assume that the sample is decent as close to random as possible.

CLM 3: No perfect multi-collinearity

First, multi-collinearity is guaranteed when we have more features than samples, which is not the case here. Second, multi-collinearity can occur when one variable is a perfect linear combination of another set of variables. In that case, the one of those variables are regressed on the remaining of the group, the R^2 will be 1. R would have warned us if this were the case (by evaluating whether the covariance matrix is singular), so we have fulfilled this requirement.

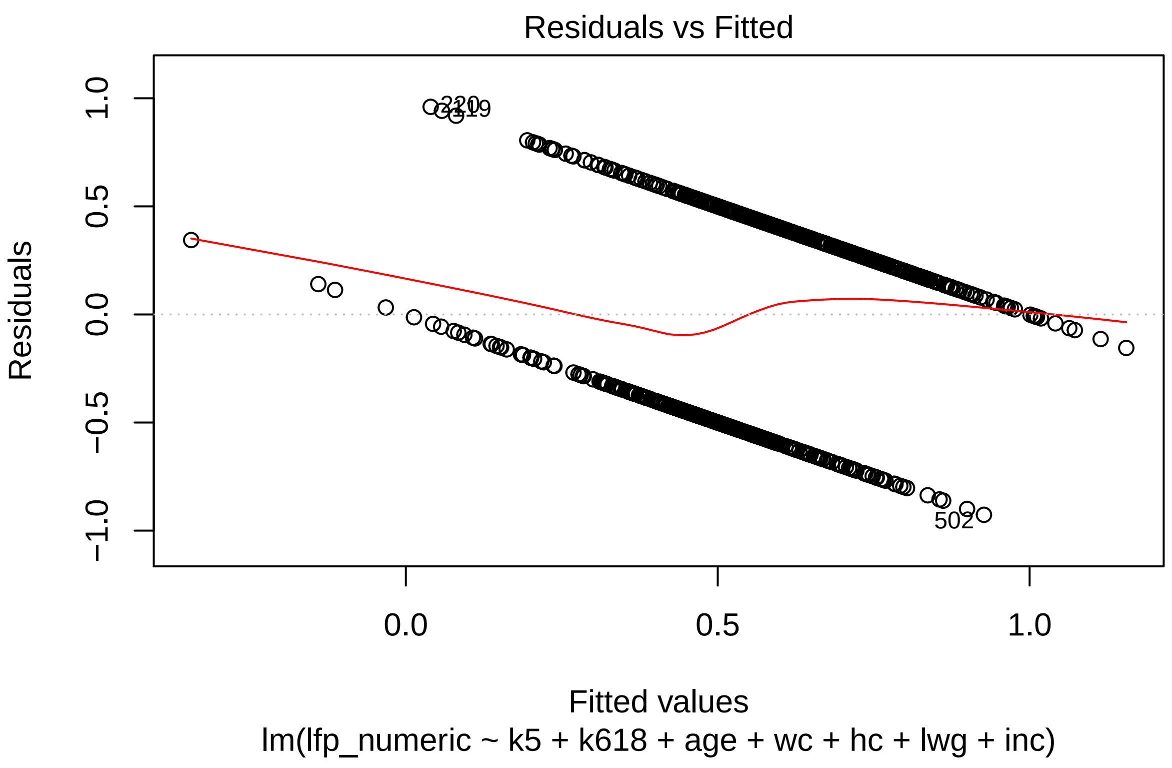
CLM 4: Zero Conditional Mean

This assumption states that the expectation of the error conditional on the explanatory variables x is 0.

$$E(u|x_1, x_2, ..., x_k) = 0$$

For the LPM to hold, this cannot be true, because we need the error term to compensate for the linearity in the parameters in order to bound y between 0 and 1. Under zero conditional mean, we expect that the residuals on the residuals versus fitted value plot to have an expected value of 0 across the board. To check this, we plot the residual agains the fitted values for our set.

```r
plot(mroz.lm, which = 1)
```

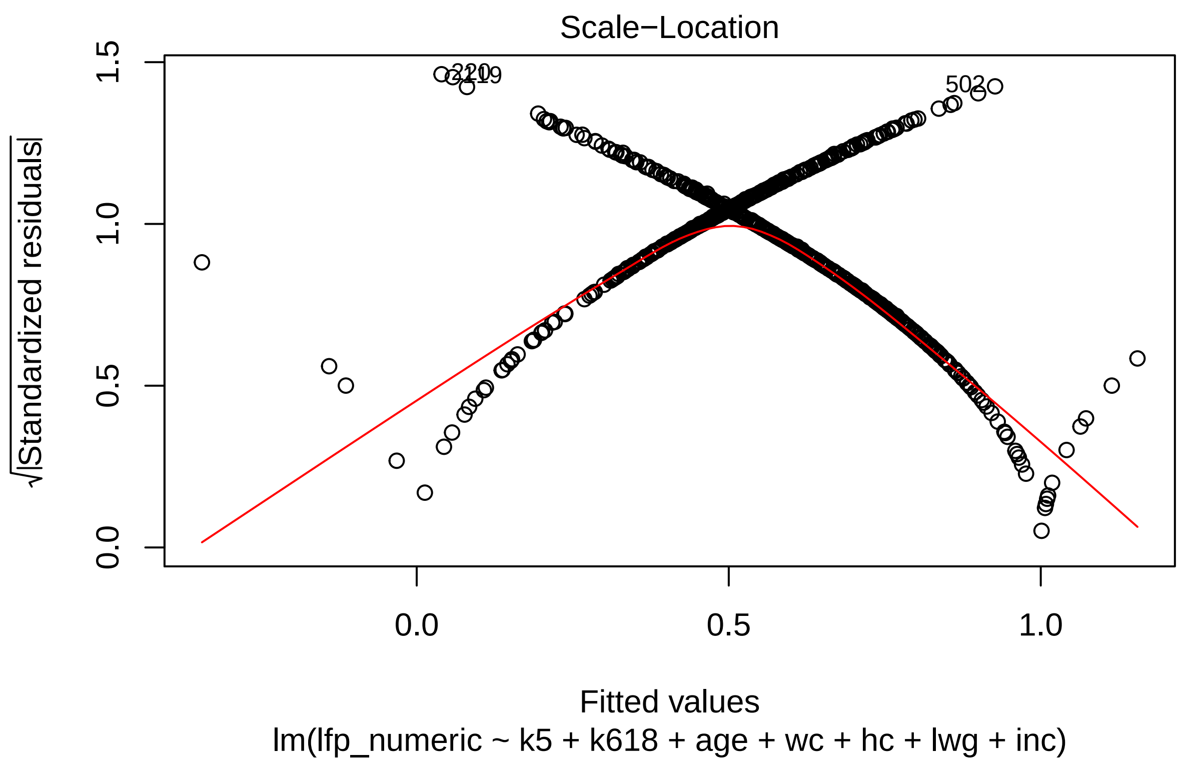


We see that there are two distinct lines, one for variables where the actual outcome of lfp=0, and one for where lfp=1. However, the mean of the residuals is not 0 for all fitted values. In fact, the mean takes the linear form between the two observed lines. Therefore, we fail to satisfy CLM 4.

CLM 5: Homoskedasticity

Homoskedasticity assumption is that the variance of the error terms are constant for any combination of x_k values. We know that homoskedasticity is violated because the outcome is binary, so the variance is $\pi * (1 - \pi)$. The easiest assessment is with the scale-location plot. If there is homoskedasticity, we would expect a roughly horizontal line of data points on this plot. This is clearly not the case below:

```r
plot(mroz.lm, which = 3)
```



One way to test for homoskedasticity is the Breusch-Pagan Test. The null hypothesis of the test states that we have homoskedasticity. We will test at a standard significance level of 0.05.

$$H_0 : \text{Homoskedasticity}$$
$$H_a : \text{Heteroskedasticity}$$

```
# lmtest was covered in 203 and heavily used for testing
# hypothesis surrounding the linear regression model.
# Therefore, I will use it here as well to test for
# homoskedasticity.
library(lmtest)
```

```
## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
bptest(mroz.lm)
```

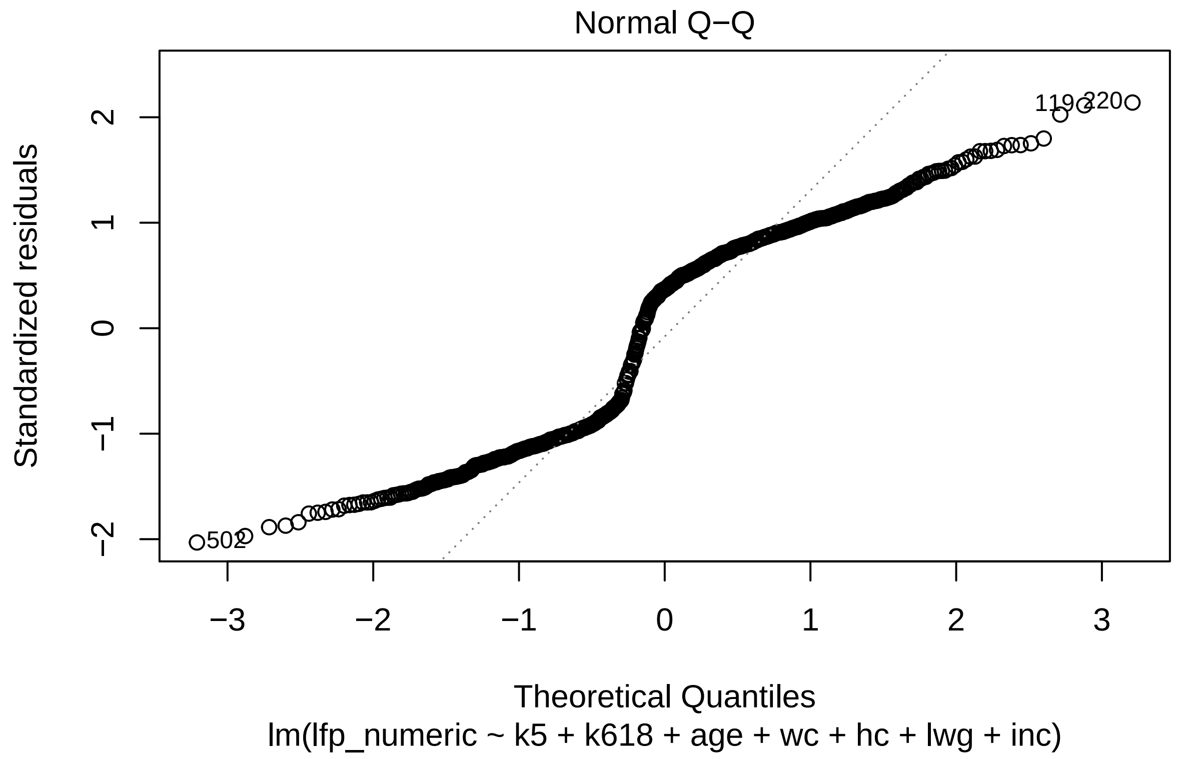
```
##
##  studentized Breusch-Pagan test
##
## data:  mroz.lm
## BP = 97.603, df = 7, p-value < 2.2e-16
```

Since the $p - value << 0.05$, we reject the null hypothesis that we have homoskedasticity and have strong evidence for heteroskedasticity.

CLM 6: Normality

CLM 6 assumes that population error is independent of the explanatory variables x_1 through x_k, and that the error term is normally distributed with mean 0 and constant variance. We can check this with the qqplot of the fitted values versus residuals plot.

```r
plot(mroz.lm, which = 2)
```



We see that most of the points lie away from where we would expect them to be if the residuals are normally distributed. To test for normality, we can perform the Shapiro-Wilk test of normality.

We will test at a standard significance level of 0.05.

$$H_0 : \text{Residuals are normal}$$
$$H_a : \text{Residuals are not normal}$$

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

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

Since p << 0.01, we reject the null hypothesis that the residuals are normal.

So in effect, CLM4,5,6 are all significantly violated, and the assumption made in CLM1 is incorrect.

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`.

Let pi_{lfp} be the probability of the female labor force participation. The model we want to estimate is:

$$\log \frac{\pi_{lfp}}{1-\pi_{lfp}} = \beta_0 + \beta_1 * \text{age} + \beta_2 * \text{inc} + \beta_3 * \text{wcyes} + \beta_4 * \text{hcyes} + \beta_5 * \text{lwg} + \beta_6 * \text{totalKids} + \beta_7 * \text{age}^2$$

We first construct the new columns totalKids and age_squared, then fit the model. Note we can specify age^2 directly into the model as well.

```
Mroz$totalKids <- Mroz$k5 + Mroz$k618
Mroz$age_squared <- Mroz$age^2
mroz.glm <- glm(lfp ~ age + inc + wc + hc + lwg + totalKids +
    age_squared, family = binomial(link = "logit"), data = Mroz)
summary(mroz.glm)
```

```
##
## Call:
## glm(formula = lfp ~ age + inc + wc + hc + lwg + totalKids + age_squared,
##     family = binomial(link = "logit"), 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
```

The model we fit is:

$$\log \frac{\pi_{lfp}}{1-\pi_{lfp}} = -5.29+0.312*\text{age}-0.035*\text{inc}+0.666*\text{wcyes}+0.098*\text{hcyes}+0.550*\text{lwg}-0.222\text{totalKids}-0.004*\text{age}^2$$

```r
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
```

Based on the LRT and the Wald tests, all parameters except for husband age is statistically significant given that the other variables are included in the model. Since we only have one insignificant term, we can safely refit the model without this parameter, and will choose to do so here.

The model we want to estimate is:

$$\log \frac{\pi_{lfp}}{1-\pi_{lfp}} = \beta_0 + \beta_1 * \text{age} + \beta_2 * \text{inc} + \beta_3 * \text{wcyes} + \beta_4 * \text{lwg} + \beta_5 * \text{totalKids} + \beta_6 * \text{age}^2$$

```r
mroz.glm <- glm(lfp ~ age + inc + wc + lwg + totalKids + age_squared,
    family = binomial(link = "logit"), data = Mroz)
mroz.glm
```

```
##
## Call:  glm(formula = lfp ~ age + inc + wc + lwg + totalKids + age_squared,
##     family = binomial(link = "logit"), data = Mroz)
##
## Coefficients:
## (Intercept)          age          inc        wcyes          lwg
##   -5.150511     0.311895    -0.033435     0.713378     0.550747
##   totalKids  age_squared
##   -0.221626    -0.004051
##
## Degrees of Freedom: 752 Total (i.e. Null);  746 Residual
## Null Deviance:       1030
## Residual Deviance: 952.3     AIC: 966.3
```

The model we fit is:

$$\log \frac{\pi_{lfp}}{1-\pi_{lfp}} = -5.15+0.312*\text{age}-0.033*\text{inc}+0.713*\text{wcyes}+0.551*\text{lwg}-0.222*\text{totalKids}-0.004*\text{age}^2$$

We verify that all parameters retained their significance:

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

```
## Analysis of Deviance Table (Type II tests)
##
## Response: lfp
##              LR Chisq Df Pr(>Chisq)
## age            8.4001  1  0.0037521 **
## inc           21.5602  1  3.429e-06 ***
## wc            13.6420  1  0.0002212 ***
## lwg           15.0819  1  0.0001029 ***
## totalKids     12.3448  1  0.0004422 ***
## age_squared   10.5371  1  0.0011700 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Question 4.3: Is the age effect statistically significant?

To test whether age is statistically significant, we generate a second model without any of the age terms (age and age_squared), and perform the LRT on the two models. This is equivalent to the Type II (same as problem 3.3), which is testing whether any terms containing age is significant. Note that based on the Anova test above, we already know that age is statistically significant given that age_squared is also in the model. We test the age variable at a significant level of 0.05 with the following hypotheses:

$$H_0 : logit(lfp) = \beta_0 + \beta_1 * inc + \beta_2 * wcyes + \beta_3 * lwg + \beta_4 * totalKids$$
$$H_a : logit(lfp) = \beta_0 + \beta_1 * age + \beta_2 * inc + \beta_3 * wcyes + \beta_4 * lwg + \beta_5 * totalKids + \beta_6 * age\_squared$$

```r
mroz.glm.no_age <- glm(lfp ~ inc + wc + lwg + totalKids, family = binomial(link = "logit"),
    data = Mroz)

anova(mroz.glm.no_age, mroz.glm, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: lfp ~ inc + wc + lwg + totalKids
## Model 2: lfp ~ age + inc + wc + lwg + totalKids + age_squared
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1       748     972.45
## 2       746     952.27  2    20.18 4.149e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Since the p-value is <<0.001, the age variable is highly statistically significant. We reject H0 that the age variable is not present in the model as we have strong evidence that age and its quadratic term is important given that the other variables are in the model.

```r
data.frame(Ha.model = AIC(mroz.glm), H0.model = AIC(mroz.glm.no_age),
    row.names = "AIC")
```

```
##     Ha.model H0.model
## AIC 966.2662 982.4462
```

In addition, both AIC and residual deviance suggests that the model including age has more explanatory power than the model without age. So we conclude that not only is age statistically significant, but also that including age plus its quadratic term improves model performance.

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.

First, we calculate what the OR is:

$$\begin{aligned}
\frac{lfp_{age=45}}{lfp_{age=45-5}} &= \frac{e^{\beta_0+\beta_1*(age-5)+\beta_2*inc+\beta_3*wcyes+\beta_4*lwg+\beta_5*totalKids+\beta_6*(age-5)^2}}{e^{\beta_0+\beta_1*age+\beta_2*inc+\beta_3*wcyes+\beta_4*lwg+\beta_5*totalKids+\beta_6*age^2}} \\
&= \frac{e^{\beta_1*(age-5)+\beta_6*(age^2-10*age+25)}}{e^{\beta_1*age+\beta_6*age^2}} \\
&= e^{\beta_1*(-5)+\beta_6*(-10*age+25)}
\end{aligned}$$

```r
OR.age45 <- exp(-5 * mroz.glm$coefficients["age"] + mroz.glm$coefficients["age_squared"] *
    (-10 * 45 + 25)) %>% unname
print(OR.age45)
```

```
## [1] 1.176272
```

The interpretation is that the estimated odds of a 45 year old female participating in the labor force increase by 1.18 times for a 5 year decrease in age. This makes sense intuitively that younger individuals have a higher odds of participating in the labor force.

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.

For this particular problem, we would like to estimate the profile likelihood CI of probability of labor force participation for:

1. age = 40
2. inc = 20
3. wcyes = 0
4. lwg = 1
5. totalKids = 0

The probability of labor force participation can be calculated from mcprofile. The order of coefficients need to be specified for the package:

```r
# order of coefficients: (Intercept), age, inc, wcyes, lwg,
# totalKids, age_squared
K <- matrix(c(1, 40, 20, 0, 1, 0, 1600), nrow = 1, ncol = 7)
```

```r
# Generate the mcprofile object
linear.combo <- mcprofile(object = mroz.glm, CM = K)
ci.logit.profile <- confint(linear.combo, level = 0.95)

# Generate the OR for probability
data.frame(pi.hat = ci.logit.profile$estimate, round(exp(ci.logit.profile$confint)/(1 +
    exp(ci.logit.profile$confint)), 3), row.names = "Probability CI")
```

```
##                 Estimate lower upper
## Probability CI 0.7251771 0.595 0.745
```

This suggests that with 95% confidence, the estimated 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 is between 0.595 and 0.745. Since this interval does not contain 0.5, it is statistically more likely that these individuals participate in the workforce than they do not.

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.

```r
wheat <- read.csv("data/wheat.csv")
wheat.density.only <- wheat %>% select(density, type)
head(wheat.density.only, n = 3)
```

```
##    density    type
## 1 1.349253 Healthy
## 2 1.287440 Healthy
## 3 1.233985 Healthy
```

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).

First, I will explain how the likelihood is formed in the context of multinomial regression.

The dataset consists of 3 response categories (Y = healthy, sprout, scab), 1 explanatory variables (density) and 275 examples. In the context of multinomial regression, we allow each example to have its own set of probabilities for each category (parameters), but model the parameters for sprout and scab as log odd ratios to the base level (healthy). This allows us to write the probability of each response category as the follow (pg 153, but with beta_j1 representing density):

$$\pi_{healthy} = \frac{1}{1+\sum_{j=2}^{3} e^{\beta_{j0}+\beta_{j1}x_1}}$$

$$\pi_{scab} = \frac{e^{\beta_{20}+\beta_{21}*x_1}}{1+\sum_{j=2}^{3} e^{\beta_{j0}+\beta_{j1}x_1}}$$

$$\pi_{sprout} = \frac{e^{\beta_{30}+\beta_{31}*x_1}}{1+\sum_{j=2}^{3} e^{\beta_{j0}+\beta_{j1}x_1}}$$

Each example then is treated as drawn from a multinomial distribution with the probability of each category given above. However, since we allow the probability parameter to vary with density, the marginal distribution simplifies for each sample. Suppose that samples with density x has probabilities π_k, π_g, π_h, and a sample with density x is observed to belong to category $k = 1$ (letting $g = 2$,$h = 3$ represent the other two categories that the sample does not belong to). Then, the probability of observing this sample is:

$$\begin{aligned} P(N_1 = 1, N_2 = 0, N_3 = 0) &= \frac{n!}{\prod_{j=1}^{3} n_j!}\prod_{j=1}^{3}\pi_j^{n_j} \\ &= \frac{1!}{0! * 0! * 1!}\pi_k^1\pi_g^0\pi_g^0 \\ &= \pi_k \end{aligned}$$

The likelihood of a parameter set given data is simply the probability of joint distribution of observing the data. Since the samples are assumed to be iid, the likelihood function is simply a product of the marginal distributions of each sample, which as shown above is π_k, **where k represents the category where each sample belongs**. Note even if multiple samples have the exact same explanatory variable of density x, the likelihood is still the product of the individual π_k, where k is the category where the samples belong, since the samples are assumed to be independent. Suppose that there are N examples with outcomes denoted by y_i. With each example having 3 π values representing each category k (where k = Healthy, Scab, Sprout), the likelihood is then:

$$L(\beta|y_1, ..., y_n) = \prod_{i=1}^{N} \pi_{y_i=k}$$

The log likelihood is a sum of these log terms over all i examples:

$$\log[L(\beta|y_1, ..., y_n)] = \sum_{i=1}^{N} \log \pi_{y_i=k}$$

```r
# Note: See 5.2 for vectorized implementation for speed. This
# version is slow but spells out every step carefully in
# order to explain the MLE process.

logLik_multinomial <- function(beta, X, Y) {

    # The logLik_multinomial will take a table of coefficients
    # (beta), an array for the density variable, and an array of
    # responses (Y), and returns the log likelihood of observing
    # such samples assuming the data follows a multinomial
    # distribution. This function specifically operates on the
    # wheat dataset.

    # Guarantee that beta has the correct dimensions. Optim will
    # flatten values into a single vector
    beta <- matrix(beta, nrow = 2)

    # These are nested functions for calculating each of the pi
    # values. Each sample will call one of these functions
    # Depending on which category it belows:

    pi_healthy <- function(x) {
        denom <- 1 + exp(beta[1, 1] + beta[1, 2] * x) + exp(beta[2,
            1] + beta[2, 2] * x)
        return(1/(denom))
    }

    pi_scab <- function(x) {
        return(exp(beta[1, 1] + beta[1, 2] * x) * pi_healthy(x))
    }
    pi_sprout <- function(x) {
        return(exp(beta[2, 1] + beta[2, 2] * x) * pi_healthy(x))
    }

    # To make calculations clear, we will iterate through each
    # example
    logLik <- 0
    for (i in 1:length(X)) {

        row <- X[i]
        response <- Y[i]

        if (response == "Healthy") {
            prob_i <- pi_healthy(row)
        } else if (response == "Scab") {
            prob_i <- pi_scab(row)
        } else if (response == "Sprout") {
            prob_i <- pi_sprout(row)
        } else {
            stop("Category not recognized")
        }
        logLik <- logLik + log(prob_i)
    }
    return(logLik)
}

library(nnet)
mod.mul <- 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
```

```r
# NOTE: I've split density and outcome into 2 separate
# variables. The function logLik_multinomial accepts the two
# variables separately
beta <- coefficients(mod.mul)
Y <- wheat$type
X <- wheat$density

# Gather the timing
My_function <- logLik_multinomial(beta, X, Y)

R_function <- logLik(mod.mul)[1]
```

```
data.frame(My_function = My_function, R_function = R_function,
    row.names = "LogLikelihood")
```

```
##               My_function R_function
## LogLikelihood   -229.7123  -229.7123
```

We can see that my function and the R function produces the same estimates.

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 $log\left(\frac{\pi_2}{\pi_1}\right)$).

First, we get the starting estimates:

```
# I first vectorize the calculations in order to speed it up.
# Optim is taking way too long looping through each example.
# In addition, this function can generalize to ANY number of
# features in X.

logLik_multinomial_FAST <- function(beta, X, Y) {

    # Guarantee that beta has the correct dimensions. Optim will
    # flatten values into a single vector
    beta <- matrix(beta, nrow = 2, byrow = TRUE)

    # calculate matrices of probability of each category (added
    # column of 1 to X for beta_0) Even though we only need one
    # category probability for each example, the matrix operation
    # is very fast
    pi_healthy <- 1/(1 + rowSums(exp(as.matrix(cbind(1, X)) %*%
        beta)))
    pi_scab <- rowSums(exp(as.matrix(cbind(1, X)) %*% beta[,
        1])) * pi_healthy
    pi_sprout <- rowSums(exp(as.matrix(cbind(1, X)) %*% beta[,
        2])) * pi_healthy

    # log transform, then extract only the value corresponding to
    # correct type outcome The log likelihood is found by summing
    # the probabilities of each example corresponding to its
    # outcome
    pi_healthy <- sum(log(pi_healthy)[Y == "Healthy"])
    pi_scab <- sum(log(pi_scab)[Y == "Scab"])
    pi_sprout <- sum(log(pi_sprout)[Y == "Sprout"])

    # final log likelihood is the sum of all examples
    return(pi_healthy + pi_scab + pi_sprout)
}
```

```r
My_function_FAST <- logLik_multinomial_FAST(beta, X, Y)

LL_compare <- cbind(My_function = My_function, My_function_FAST = My_function_FAST,
    R_function = R_function)
rownames(LL_compare) <- "LogLikelihood"
print(LL_compare)
```

```
##               My_function My_function_FAST R_function
## LogLikelihood   -229.7123        -229.7123  -229.7123
```

We see that the functions all produce the same results. for the estimated covariance matrix:

```r
# Fitting Scab to Healthy:
scab_healthy <- subset(wheat, type %in% c("Healthy", "Scab"))

# Set the Healthy as 'failure'; otherwise, R will default to
# using Healthy as 'success' due to alphabetical ordering and
# the sign on the type coefficient will be negative Also,
# note that explicitly seting srw as 1 and hrw as 0 is not
# strictly necessary. R will set srw as 2, and hrw as 1, by
# default for factors, and this will not influence our
# estimates very much.
scab_healthy$type_adj <- ifelse(scab_healthy$type == "Healthy",
    yes = 0, no = 1)

# Fit the model
mod.scab_healthy <- glm(type_adj ~ density, data = scab_healthy,
    family = binomial(link = "logit"))

# Fitting Sprout to Healthy:
sprout_healthy <- subset(wheat, type %in% c("Healthy", "Sprout"))

# Set the Healthy as 'failure' like above
sprout_healthy$type_adj <- ifelse(sprout_healthy$type == "Healthy",
    yes = 0, no = 1)

mod.sprout_healthy <- glm(type_adj ~ density, data = sprout_healthy,
    family = binomial(link = "logit"))

# gather coefficients into one table
coefs <- rbind(Scab = coefficients(mod.scab_healthy), Sprout = coefficients(mod.sprout_healthy

# Use optim to optimize; note that the data for X was defined
# above Here, I crank maxit to 1k and epsilon to 1e-10 to try
# to get to as close as possible to multinom In reality, the
# default values (maxit = 50, epsilon=1e-8) will already
# produce very similar values
mod.fit.optim <- optim(par = coefs, fn = logLik_multinomial_FAST,
    X = X, Y = Y, method = "BFGS", hessian = TRUE, control = list(fnscale = -1,
        trace = TRUE))
```

```
## initial  value 278.050806
## iter  10 value 229.717163
## final  value 229.712285
## converged
```

```
print("optim")
```

```
## [1] "optim"
```

```
print(mod.fit.optim$par)
```

```
##        (Intercept)   density
## Scab      29.39074 -24.57235
## Sprout    19.13215 -15.48479
```

```
print("multinom")
```

```
## [1] "multinom"
```

```
print(coefficients(mod.mul))
```

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

We see that optim converged to a likelihood ratio that is very similar to that converged by multinom. In addition, the coefficient estimates are the same up to the second decimal point.

```
# covariance, which we get from inverting the Hessian
cov.mat <- -solve(mod.fit.optim$hessian)

# Note that since beta was given as output from the logistic
# regression models, the columns represent the variables and
# the rows represent the labels. Optim will return the
# hessian in a column first format, meaning it generate the
# labels as [1,1]Scab:(Intercept), [2,1]Sprout:(Intercept),
# [1,2]Scab:density, [2,2]Sprout:density.  We could either
# supply beta as beta transpose, or switch the rows and
# columns after returning the Hessian.  For now, we will do
# the latter.

# We will swap rows 2 and 3 and columns 2 and 3 in order to
# have the same labels as returned from vcov
cov.mat[c(2, 3), ] = cov.mat[c(3, 2), ]
cov.mat[, c(2, 3)] = cov.mat[, c(3, 2)]

# label the rows and columns
rownames(cov.mat) <- c("Scab:(Intercept)", "Scab:density", "Sprout:(Intercept)",
    "Sprout:density")
colnames(cov.mat) <- c("Scab:(Intercept)", "Scab:density", "Sprout:(Intercept)",
    "Sprout:density")


print("covariance matrix from optim")
```

```
## [1] "covariance matrix from optim"
```

```
print(cov.mat)
```

```
##                    Scab:(Intercept) Scab:density Sprout:(Intercept)
## Scab:(Intercept)          13.528395   -11.084130          10.332695
## Scab:density             -11.084130     9.113213          -8.334216
## Sprout:(Intercept)        10.332695    -8.334216          11.143625
## Sprout:density            -8.278662     6.686854          -8.976673
##                    Sprout:density
## Scab:(Intercept)        -8.278662
## Scab:density             6.686854
## Sprout:(Intercept)      -8.976673
## Sprout:density           7.248587
```

```
print("covariance matrix from vcov")
```

```
## [1] "covariance matrix from vcov"
```

```
print(vcov(mod.mul))
```

```
##                    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
```

From the matrix above, all values are highly similar to each other up to the second decimal point. Therefore, we've also reproduced the co-variance matrix.

*This work was done as part of the W271 - Statistical Methods for Discrete Response, Time Series, and Panel Data course under the U.C. Berkeley Master of Information and Data Science program.