STATS216 Homework 2

Darragh Hanley

Thursday, February 05, 2015

---

1) Logistic regression can give poor results when the two classes can be perfectly separated by a linear decision boundary. Consider just a logistic regression with a single predictor, X,....

(a) Show that the likelihood function $L(\beta_0; \beta_1)$ is always strictly less than 1.

First we will examine p(x) under different values of $\beta_0 + \beta_1 X$

As $\beta_0 + \beta_1 X$ goes to ∞, p(x) goes to e^{∞} / (1 + e^{∞}). e^{∞} is infinity, so p(x) goes to 1, but never reaches 1 as the numerator is always less than the denomiator.

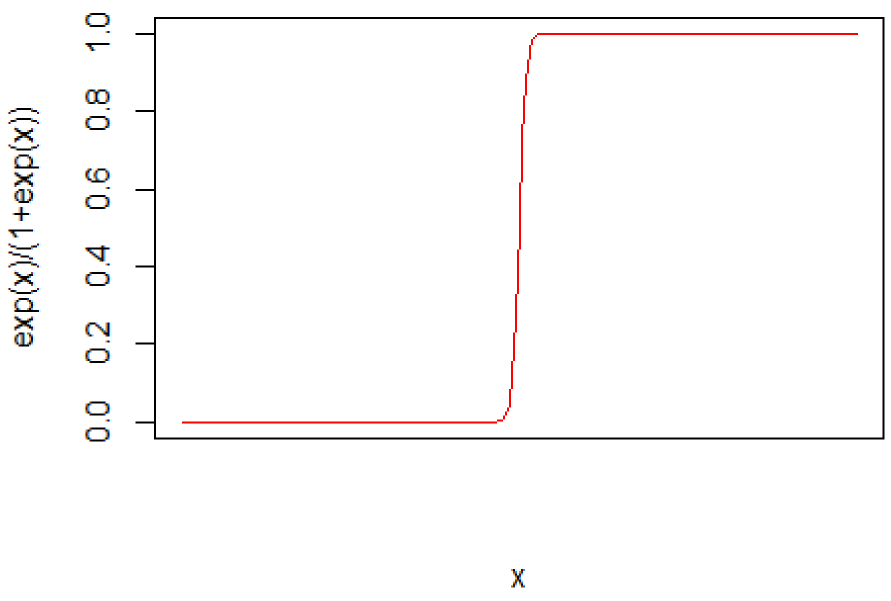
As $\beta_0 + \beta_1 X$ goes to $-\infty$, p(x) goes to $e^{-\infty}$ / (1 + $e^{-\infty}$). e^{∞} is 0, so p(x) goes to 0, but never reaches 1 as the numerator never reaches 0, and the denominator goes to but never reaches 1.

Between both of the above we have the case where $\beta_0 + \beta_1 X$ = 0. Here, P(x) = e^0 / (1 - e^0) = 1 / (1+1) = .5

We can plot these values of P(x), as seen below.

The likelihood function is made of two seperate probabilities. Each of these probabilities follows the model p(X) = $e^{\beta_0+\beta_1 X}$ / (1 + $e^{\beta_0+\beta_1 X}$). Therefore the likelihood function is always between 0 and 1 but never reaches 0 and 1. So, the likelihood function L(β_0; β_1) is always strictly less than 1.

```r
x <- -100:100
p <- exp(x)/(1+exp(x))
plot(x, p, type="l",xlab="x", col="red", xaxt="n", ylab="exp(x)/(1+exp(x))")
```



(b) Suppose that all of the x_i corresponding to y_i = 0 are negative, and all of the other x_i are positive. In that case, show that we can get L(β_0; β_1) arbitrarily close to 1. That is, show that for any value a < 1, no matter how close to 1, you can always find values β_0 and β_1 for which L(β_0; β_1) > a. Explain why this means that $\hat{\beta_0}$ and $\hat{\beta_1}$ are undefined.

From the book, we know when the classes are well-separated, the parameter estimates for the logistic regression model are surprisingly unstable. This question explains such a case.

We have a single predictor, x_i, and our response, y_i, has two possible classes. Let us call the classes of y_i, 0 and 1.

For all of the y_i = 0. We know that x_i < 0.

For all of the other x_i, eg. where y_i = 1, x_i => 0.

Given the above we know the outcome of Y for any value of X. Once we know whether X is negative or not, we know the class of Y. So, Pr(Y = 0 | X < 0) = 1 and Pr(Y = 1 | X => 0) = 1.

However the likelihood function used to estimate Y, holds that p(X) = $e^{\beta_0+\beta_1 X}$ / (1 + $e^{\beta_0+\beta_1 X}$). And as seen in part (a), p(x) in the likelihood function never reaches 1. These are the two terms of the likelihood function.

So the likelihood function will take the p(x) as close to 1 as it can without ever reaching 1. However, no matter how close it can get to 1, for example, value a < 1; it always has room (1-a), to make p(x) closer to 1. Therefore for any value a < 1, no matter how close to 1, you can always find values β_0 and β_1 for which L(β_0; β_1) > a.

In this case, the estimates $\hat{\beta_0}$ and $\hat{\beta_1}$ used in the likelihood function is essentially a moving target. No matter what value we assign to $\hat{\beta_0}$ and $\hat{\beta_1}$ to get p(x) near to the true probability of 1, we can always change them to make them more accurate. This means $\hat{\beta_0}$ and $\hat{\beta_1}$ are undefined.

(c) Show that $\hat{\beta_0}$ and $\hat{\beta_1}$ are similarly undefined if there is any value c for which every x_i corresponding to y_i = 0 is less than c and every other x_i is larger than c. In fact, the same is true for multivariate logistic regression. Whenever there is a linear decision boundary that perfectly separates the two classes, the maximum likelihood logistic regression coefficients are undefined (but you don't have to prove this last fact).

Again, we have a single predictor, x_i, and our response, y_i, has two possible classes. Let us call the classes of y_i, 0 and 1.

For all of the y_i = 0. We know that x_i < c.

For all of the other x_i, eg. where y_i = 1, x_i => c.

Given the above we know the outcome of Y for any value of X. Once we know whether X is less than c or not, we know the class of Y. So, Pr(Y = 0 | X < c) = 1 and Pr(Y = 1 | X => c) = 1. These are the two terms of the likelihood function.

However the likelihood function used to estimate Y, holds that p(X) = $e^{\beta_0+\beta_1 X}$ / (1 + $e^{\beta_0+\beta_1 X}$). And as seen in part (a), p(x) in the likelihood function never reaches 1.

Again the likelihood function will take the p(x) as close to 1 as it can without ever reaching 1. However, no matter how close it can get to 1, for example, value a < 1; it always has room (1-a), to make p(x) closer to 1. Therefore for any value a < 1, no matter how close to 1, you can always find values β_0 and β_1 for which L(β_0; β_1) > a.

In this case, the estimates $\hat{\beta_0}$ and $\hat{\beta_1}$ used in the likelihood function is essentially a moving target. No matter what value we assign to $\hat{\beta_0}$ and $\hat{\beta_1}$ to get p(x) near to the true probability of 1, we can always change them to make them more accurate. This means $\hat{\beta_0}$ and $\hat{\beta_1}$ are similarly undefined as per question part (b).

(d) Come up with your own data set of the form in (c) and fit a logistic regression to it in R. Plot your data, as well as the logistic regression fit ^p(x). You will probably get warning messages that the fit didn't converge, and that you have numerically 0 or 1 fitted probabilities. The first message usually signals that you have fit a logistic regression to perfectly separable classes.

First we set the seed and generate 1000 random values of mean 2 and standard deviation 5. this is the predictor.

```
set.seed(124)
norm <- rnorm(100, 2, 5)
x <- norm[1:1000]
```

Now we create the response of the same length. We use c=3 as the separator of classes. For all x<3 the response is 0. And for all other the response is 1.

```
y <- rep(NA, 1000)
y[x>=3] <- 1
y[x<3] <- 0
dat=as.data.frame(cbind(x,y))
```

Now we fit the logistic regression and check the warning message.

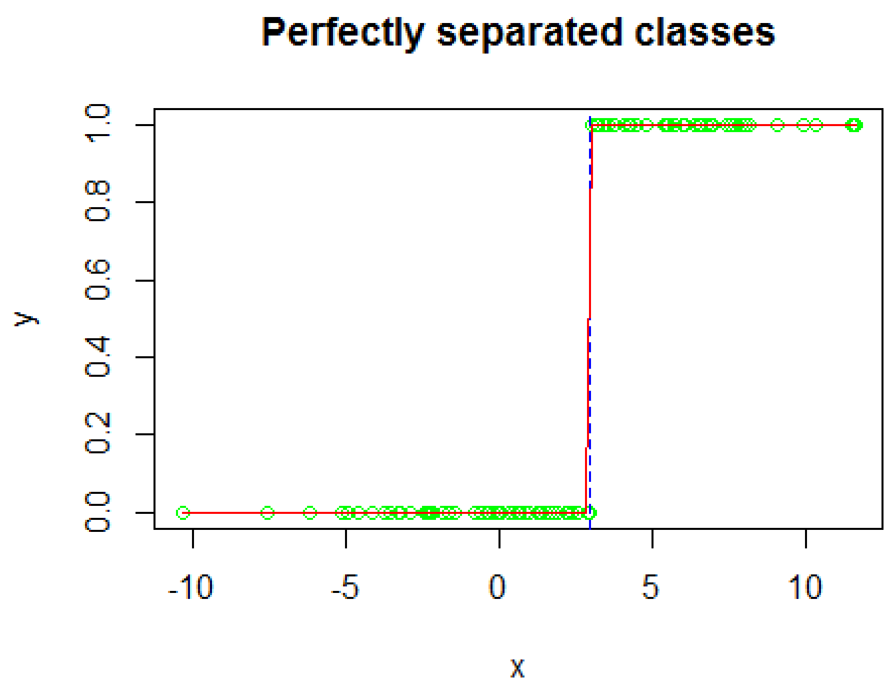
```
log.mod <- glm(y~x,family=binomial,dat)

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

Create the plot with a red line for the logistic fit, green points for the actual points and add a blue dashed line to indicate where the separating boundary is.

```
plot(x,y, col="green", main = "Perfectly separated classes")
abline(v=3,col="blue",lty=2)
curve(predict(log.mod,data.frame(x=x),type="resp"), col="red",add=TRUE)
```



2) We will now derive the probability that a given observation is part of a bootstrap sample. Suppose that we obtain a bootstrap sample from a set of n observations.

(a) What is the probability that the first bootstrap observation is not the jth observation from the original sample? Justify your answer.

The probability that the first bootstrap observation is not the jth observation is (n-1)/n.

This can be proved as follows the case because the,

1. Probability that the *first bootstrap obervation is observation j* = 1/n
2. Probability that the *first bootstrap obervation is observation j* or *first bootstrap obervation is not observation j*, is = 1
3. Therefore, probability that the *first bootstrap obervation is not observation j* = 1-(1/n) = (n-1)/n

(b) What is the probability that the second bootstrap observation is not the jth observation from the original sample?

The probability that the second observation is not the jth observation from the original sample is (n-1)/n. The same as the probability for the first observation.

(c) Argue that the probability that the jth observation is not in the bootstrap sample is (1-(1/n))^n.

In a bootstrap sample, of n observations, we choose out a single observation, n times.

As seen in part (a), above, for each single observation the probability we will not choose the jth observation is : 1-(1/n).

If two events, A and B are independent then the joint probability is P(A and B) = P(A)P(B).

In the bootstrap case, each event is independent. Therefore the we get the product of all the events.

We have n events, each of probability (1-(1/n)). Therefore we get the probability of that event n times. This is (1-(1/n))^n.

Lets test this. If there are two observations(n=2), and j=2. The probability j=2 is not drawn on the first sample is .5, the same on the second sample. The probability it is not chosen in either is the .25. Plugging this into the formula, to check our result is really .25 :

```
    n = 2
    (1-(1/n))^n == .25

## [1] TRUE
```

(d) When n = 5, what is the probability that the jth observation is in the bootstrap sample?

The formula to use for the jth observation **not in the bootstrap sample** is (1-(1/n))^n, as seen in previous questions.

The formula to use for the jth observation **in the bootstrap** sample is 1-(1-(1/n))^n. Since both events are mutually exclusive and the probability of getting one of the two events is 1.

```
    n = 5
    print(paste( "For n=5, the probability that the jth observation is in the bootstra
p sample ", round(1-(1-(1/n))^n, digits=3), sep=""))

## [1] "For n=5, the probability that the jth observation is in the bootstrap sample 0
.672"
```

(e) When n = 100, what is the probability that the jth observation is in the bootstrap sample?

```
    n = 100
    print(paste( "For n=100, the probability that the jth observation is in the bootst
rap sample", round(1-(1-(1/n))^n, digits=3)))

## [1] "For n=100, the probability that the jth observation is in the bootstrap sample
0.634"
```

(f) When n = 10,000, what is the probability that the jth observation is in the bootstrap sample?

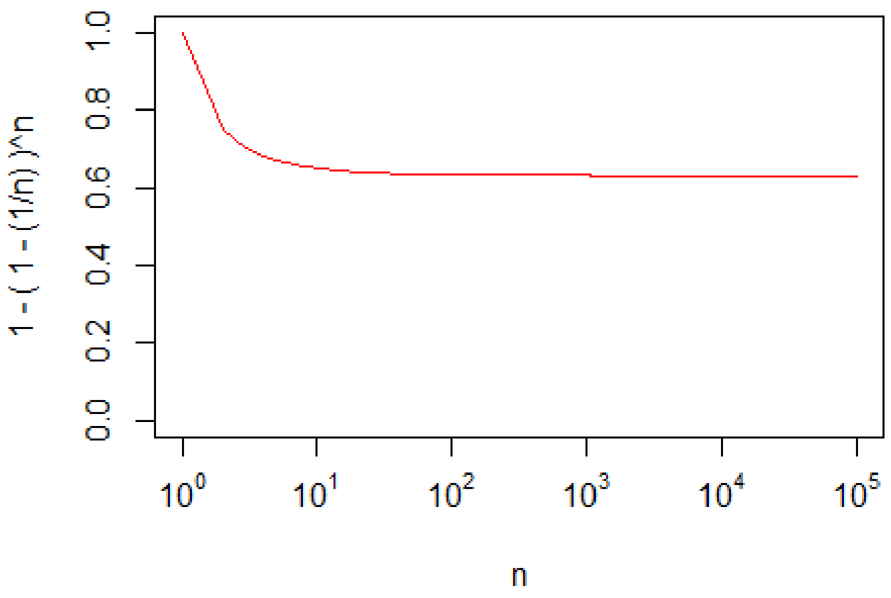
```
    n = 10000
    print(paste( "For n=10,000, the probability that the jth observation is in the boo
tstrap sample", round(1-(1-(1/n))^n, digits=3)))

## [1] "For n=10,000, the probability that the jth observation is in the bootstrap sam
ple 0.632"
```

(g) Create a plot that displays, for each integer value of n from 1 to 100000, the probability that the jth observation is in the bootstrap sample. Comment on what you observe.

It can be seen below that as the population grows larger, the probability that the jth observation will not be picked converges to one value, circa 0.632.

```
x <- 1:100000
y <- 1-(1-(1/x))^x
plot(x, y, log="x", type="l", xlab="n", col="red", xaxt="n", ylab="1 - ( 1 - (1/n) )^n
", ylim=c(0, 1))
ticks <- seq(0, 5, by=1)
labels <- sapply(ticks, function(i) as.expression(bquote(10^ .(i))))
axis(1, at=c(1, 10, 100, 1000, 10000,100000), labels=labels)
```



(h) We will now investigate numerically the probability that a bootstrap sample of size n = 100 contains the jth observation. Here j = 4. We repeatedly create bootstrap samples, and each time we record whether or not the fourth observation is contained in the bootstrap sample.

> store = rep (NA , 10000)

> for (i in 1:10000) { store [i]= sum (sample (1:100 , rep= TRUE) ==4) > 0 }

> mean (store)

Comment on the results obtained.

The below code put the theory of part (g) to experiment, by actually creating 10000 different boot straps from the same population, of sample size = 100. It then checks from these 10000 samples, what proportion of the time, a particular value was taken into the sample. The answer, .6408, is very close to the probability, seen in part (g) of the question, which large values of n converge to.

```
set.seed(1)
store = rep (NA , 10000)
for (i in 1:10000) {
    store [i]= sum ( sample (1:100 , rep= TRUE ) ==4) > 0
    }
mean ( store )

## [1] 0.6408
```

3) Suppose we estimate the regression coefficients in a linear regression model by minimizing... ridge regression formula... for a particular value of lambda. For parts (a) through (e), indicate which of i. through v. is correct. Justify your answer.

(a) As we increase λ from 0, the training RSS will: (i. Increase initially, and then eventually start decreasing in an inverted U shape. ii. Decrease initially, and then eventually start increasing in a U shape. iii. Steadily increase. iv. Steadily decrease. v. Remain constant.)

iii. Steadily increase.

with 0 as λ, the linear regression model will **minimise** the MSE of the training data. As we reduce the coefficients of the linear model by increasing λ, the training MSE will increase steadily.

Or alternatively as seen in the online quiz, increasing λ will force us to fit simpler models. This means that training RSS will steadily increase because we are less able to fit the training data exactly.

(b) Repeat (a) for test RSS.

ii. Decrease initially, and then eventually start increasing in a U shape.

Initially, as we make λ larger from 0, the bias is pretty much unchanged (slightly increasing), but the variance drops. So a ridge regression, by shrinking the coefficient toward 0, controls the variance. It doesn't allow the coefficient to be too big, and it gets rewarded because the mean square error (sum of bias & variance) goes down. However at one point the bias starts increasing more quickly and we reach a point where the test MSE is minimised. At this point Bias is rising faster than variance is falling. This causes the test RSS to rise again.

Or alternatively as seen in the online quiz, At first, we expect test RSS to improve because we are not overfitting our training data anymore. Eventually, we will start fitting models that are too simple to capture the true effects and test RSS will go up.

(c) Repeat (a) for variance.

iii. Steadily decrease.

This is rooted in the bias-variance trade-off. An increase in λ, from 0, causes a decrease in the flexibility of the ridge regression fit, leading to decreased variance and increased bias.

Or alternatively as seen in the online quiz, increasing λ will cause us to fit simpler models, which reduces the variance of the fits.

(d) Repeat (a) for (squared) bias.

iii. Steadily increase.

This is rooted in the bias-variance trade-off. An increase in λ causes a decrease in the flexibility of the ridge regression fit, leading to decreased variance and increased bias.

Or as seen online, increasing λ will cause us to fit simpler models, which have larger squared bias.

(e) Repeat (a) for the irreducible error.

v. Remain constant.

Irreducible error is the noise term in the true relationship that cannot fundamentally be reduced by any model. The reducible error can be reduced by model improvements. Given this the irreducible error does not change with any model change.

Or as seen online, increasing λ will have no effect on irreducible error. By definition, irreducible error is an aspect of the problem and has nothing to do with a particular model being fit.

(4) Teamed up with Sevvandi Kandanaarachchi, Tony Wu and Andrew Beckerman for this challenge.

(a) Fit the logistic regression model above to the data and examine the rankings. What happened to make both the team saint-mary-saint-mary and the team st.-thomas-(tx)-celts look so good? Can you explain it in terms of your answers to the first question in this problem set?

First we shall load the NACC data set and pull out all teams to a variable sorted by name.

```r
games <- read.csv("http://www.stanford.edu/~wfithian/games.csv",as.is=TRUE)
teams <- read.csv("http://www.stanford.edu/~wfithian/teams.csv",as.is=TRUE)
all.teams <- sort(unique(c(teams$team,games$home,games$away)))
```

Next we assign a variable for each game to indicate whether the home team won or not. (Learning here was that basketball does not allow tied games like soccer). Initialize a data frame of the games(rows) and teams (columns), indicating for each game which teams were home or away. Using 1 for home games and -1 as an indicator for away games.

```r
### Assign win or loss indicator (1, 0) to vector z. This will be used as a response i
n the model.
z <- with(games, ifelse(homeScore>awayScore,1,0))
X0 <- as.data.frame(matrix(0,nrow(games),length(all.teams)))
names(X0) <- all.teams
# Assign to the data frame for each game which team was home or away
for(tm in all.teams) {
  X0[[tm]] <- 1*(games$home==tm) - 1*(games$away==tm)
}
```

Remove Stanford's column to make it the baseline team against which other teams will be compared. Create the index of regular season games only and assign a home advantage coefficient which is common to all games.

```r
X <- X0[,names(X0) != "stanford-cardinal"]
reg.season.games <- which(games$gameType=="REG")
homeAdv <- 1 - games$neutralLocation
```

Perform logistic regression on the regular season games with the home advantage coefficient. Extract from this the top 25 teams. From this we can see clearly that the team saint-mary-saint-mary and the team st.-thomas-(tx)-celts look very good in comparison to other teams.

```
logrega.mod <- glm(z ~ 0 + homeAdv + ., data=X, family=binomial, subset=reg.season.gam
es)
margin.top25 <- order(coef(summary(logrega.mod))[,1],decreasing=TRUE)[1:25]
coef(summary(logrega.mod))[margin.top25,1]
```

```
##   `saint-mary-saint-mary`   `st.-thomas-(tx)-celts`
##                 14.126965                 13.270185
##        `gonzaga-bulldogs`     `louisville-cardinals`
##                  2.770782                  2.412485
##         `kansas-jayhawks`         `indiana-hoosiers`
##                  2.265964                  2.255148
##        `new-mexico-lobos`      `ohio-state-buckeyes`
##                  2.239499                  2.137208
##        `duke-blue-devils`         `georgetown-hoyas`
##                  2.106862                  2.031684
## `michigan-state-spartans`      `michigan-wolverines`
##                  1.976361                  1.960319
##   `miami-(fl)-hurricanes`   `kansas-state-wildcats`
##                  1.761367                  1.749419
##         `syracuse-orange`           `memphis-tigers`
##                  1.604678                  1.532102
##   `saint-louis-billikens`  `marquette-golden-eagles`
##                  1.521991                  1.508487
##         `butler-bulldogs`         `wisconsin-badgers`
##                  1.459961                  1.450762
##          `florida-gators`   `oklahoma-state-cowboys`
##                  1.316371                  1.303918
##              `unlv-rebels`         `arizona-wildcats`
##                  1.293854                  1.292302
##     `pittsburgh-panthers`
##                  1.266337
```

Now we will explore these two teams. We draw up a contingency table of the two top teams, how many games they were in as home and away. It can be seen that both team only played 1 away game. The 5539 indicates the games they did not play in.

```
table(X[,c("saint-mary-saint-mary","st.-thomas-(tx)-celts")])
```

```
##                        st.-thomas-(tx)-celts
## saint-mary-saint-mary   -1    0
##                    -1    0    1
##                     0    1 5539
```

In addition, it can be seen below that both teams won their game which they played.

```
bool.condition.1=games$home=="saint-mary-saint-mary"|games$away=="saint-mary-saint-mar
y"
games[bool.condition.1,]
```

```
##            date                     home                  away homeScore
## 5300 2012-11-11 nebraska-omaha-mavericks saint-mary-saint-mary        86
##      awayScore neutralLocation gameType
## 5300        96               0      REG
```

```
bool.condition.2=games$home=="st.-thomas-(tx)-celts"|games$away=="st.-thomas-(tx)-celt
s"
games[bool.condition.2,]
```

```
##             date      home                     away homeScore awayScore
## 2899 2012-11-10 rice-owls st.-thomas-(tx)-celts        59        72
##      neutralLocation gameType
## 2899               0      REG
```

We saw in question 1b, when the classes are well-separated, the parameter estimates for the logistic regression model are surprisingly unstable. And I believe here we have such a case, of well separated classes, where teams only played an away game and had no losses. Therefore, it was assigned a very high β coefficient, because with logistic regression modelling in this case the β is undefined, so difficult for the model to assign a value. Later in section (c) we have a plot which shows a similar case.

(b) Get rid of teams that played less than five games and refit the model. Make a rank table like the ones we made in class, where you compare the logistic regression rankings to the linear regression rankings, the AP Rankings, and the USA Today rankings. Which model seems to correspond better to the voters' decisions, the linear regression or logistic regression?

In order to exclude teams other than Stanford (our baseline for linear regression modelling), we have two options. Exclude the teams (columns), but leave their games (like was done for Stanford). Or exclude the games they played in. I have chosen the latter for two reasons. 1) Stanford was our baseline by removing the team columns; this baseline will no longer be valid if many other team columns are excluded. 2) If low game teams are excluded, however their games are left in - the opposition's result (win or lose) will still be available as a predictor; however the strength of the low playing team they played will be lost. For logistic regression, using either approach, there are minor differences seen in the ranking result, however for the above reasons we will go with excluding the games of these low playing teams (ie. excluding rows).

We start by adding to table "games" a column of the minimum number of games played by either team. We will concentrate on regular season games only, and create a subset vector which can be used to identify the games of interest.

```
# Collect all team games to one variable
teamplays <- table(c(games$home, games$away))
# for each regular season game, add a new variable to games representing the lowest nu
mber of games played by either team in each game
games$mingames[reg.season.games] <- vapply(reg.season.games, function(i) {
  min(
    teamplays[names(teamplays)==games$away[i]],
    teamplays[names(teamplays)==games$home[i]]
  )
}, 1)
### Create a subset index on both regular season games and games played by non-low pla
y frequency teams
subset.vector <- which(games$gameType=="REG" & games$mingames >= 5)
```

Now we perform both the logistic regression and linear model with the same index. For each we pull out the coefficients which they assigned to each team. These are then added to a table of rankings, where we compare each model to the AP Rankings, and the USA Today rankings.

```
### Perform the logistic regression excluding teams with 5 games or less and non regul
ar season games
logregb.mod <- glm(z ~ 0 + homeAdv + ., data=X, family=binomial, subset=subset.vector)
logregb.coef <- coef(logregb.mod)[paste("`",teams$team,"`",sep="")]
names(logregb.coef) <- teams$team

### Calculate the Linear Model from class, change the variable names
y <- with(games, homeScore-awayScore)
homeAdvlm <- 1 - games$neutralLocation
lmb.mod <- lm(y ~ 0 + homeAdvlm + ., data=X, subset=subset.vector)
lmb.coef <- coef(lmb.mod)[paste("`",teams$team,"`",sep="")]
```

```
names(lmb.coef) <- teams$team

### Create the ranking table. To fit the table rows on one line, we exclude the coeffi
cient scoring.
rank.table <- cbind(
#                     "lm Score" = lmb.coef,
                     "lm Rank"   = rank(-lmb.coef,ties="min"),
#                     "logreg Score" = logregb.coef,
                     "logreg Rank"  = rank(-logregb.coef,ties="min"),
                     "AP Rank"      = teams$apRank,
                     "USAT Rank"    = teams$usaTodayRank)
rank.table[order(logregb.coef,decreasing=TRUE)[1:25],]
```

```
##                            lm Rank logreg Rank AP Rank USAT Rank
## gonzaga-bulldogs                 4           1       1         1
## louisville-cardinals             3           2       2         2
## kansas-jayhawks                  6           3       3         3
## indiana-hoosiers                 1           4       4         5
## new-mexico-lobos                23           5      11        10
## ohio-state-buckeyes              7           6       7         6
## duke-blue-devils                 5           7       6         7
## georgetown-hoyas                18           8       8         8
## michigan-state-spartans         12           9       9         9
## michigan-wolverines              8          10      10        11
## miami-(fl)-hurricanes           14          11       5         4
## kansas-state-wildcats           36          12      12        14
## syracuse-orange                 10          13      16        18
## memphis-tigers                  43          14      19        15
## saint-louis-billikens           26          15      13        13
## marquette-golden-eagles         30          16      15        16
## butler-bulldogs                 50          17      NA        NA
## wisconsin-badgers               11          18      18        17
## florida-gators                   2          19      14        12
## oklahoma-state-cowboys          20          20      17        19
## unlv-rebels                     32          21      NA        NA
## arizona-wildcats                16          22      21        20
## pittsburgh-panthers              9          23      20        22
## notre-dame-fighting-irish       34          24      23        NA
## colorado-state-rams             22          25      NA        NA
```

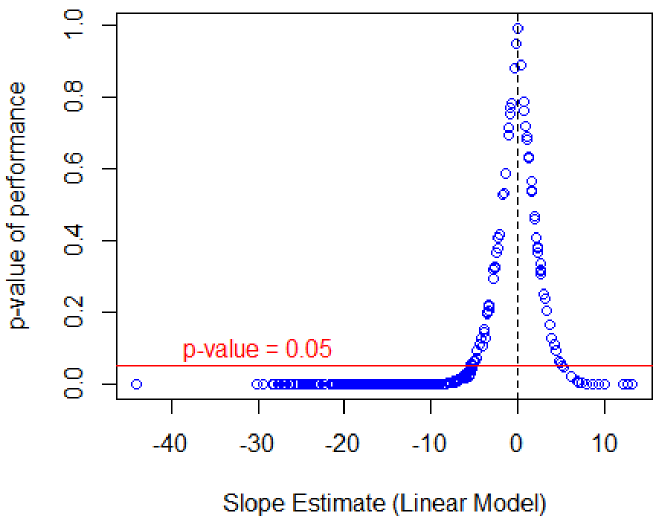
It can be seen that the logistic regression fits very well compared to the AP and USA today rankings. The linear model does not seem perform as well as logistic regression, however given the number of games and teams involved, this also performs well.

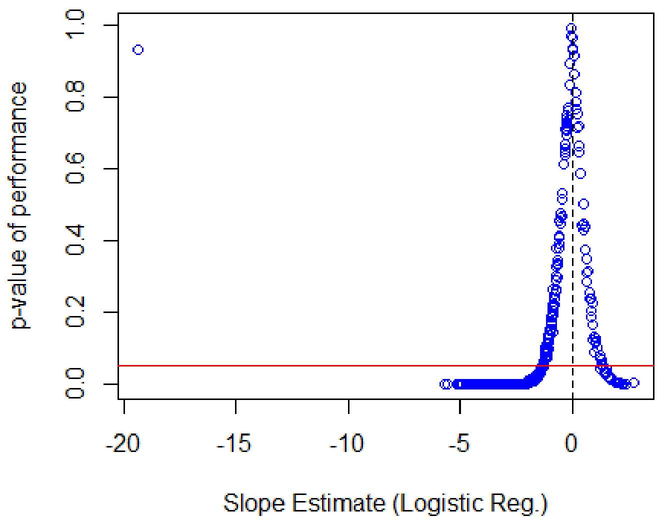
(c) When we ignore the actual value of yi and instead only use whether yi > 0, we are discarding information, so we might expect our model standard errors to be larger relative to the effect sizes. If we use the linear regression model, for what fraction of teams are we confident (p < 0.05) that the team is better (or worse) than Stanford? For what fraction are we confident if we instead use the logistic regression model?

Stanford is our one "special" baseline team j and require where $\beta_j = 0$. All other team coefficients are measured with respect to a Stanford coefficient of 0. As can be seen in the plot below, as the Coefficient/Slope Estimate reaches closer to the baseline of 0 for Stanford, the p-value of the team playing Stanford increases (ie. the probability of a certain outcome against Stanford decreases).

```
par(mfrow=c(1,2))
plot(coef(summary(lmb.mod))[-1,1], coef(summary(lmb.mod))[-1,4], xlab="Slope Estimate
(Linear Model)", ylab= "p-value of performance", col="blue")
abline(h = .05, col = "red")
```

```r
abline(v = 0, lty=2)
text(-30,.1, "p-value = 0.05", col = "red")
plot(coef(summary(logregb.mod))[-1,1], coef(summary(logregb.mod))[-1,4],xlab="Slope Es
timate (Logistic Reg.)", ylab= "p-value of performance", col="blue")
abline(h = .05, col = "red")
abline(v = 0, lty=2)
```





Notice on the above logistic regression chart, we have one outlier with a high p-value although a coefficient very far from Stanford(x=0). This team is `grambling-state-tigers` which lost all of there 28 games. This leads to perfectly separated classes which we know logistic regression does not handle well.

To determine what fraction of teams we are confident (p < 0.05) that the team is better (or worse) than Stanford, we first must determine which teams we have confidence for, and which teams not.

```r
# "Confidence" for confidence, and "No Confidence" for no confidence that the team is
better (or worse) than Stanford
linmod_pvalue <- (ifelse(coef(summary(lmb.mod))[,4]<.05,"Confidence", "No Confidence")
)
logreg_pvalue <- (ifelse(coef(summary(logregb.mod))[,4]<.05,"Confidence", "No Confiden
ce"))
```

The following table shows the proportion of records for the linear model where we have confidence or not :

```r
# Output a table showing the proportion of cases where we have confidence, and those w
ith no conifdence that teams would win or lose against Stanford for the linear model.
table(linmod_pvalue)/length(linmod_pvalue)

## linmod_pvalue
##    Confidence No Confidence
##     0.7752161     0.2247839
```

The following table shows the proportion of records for the logistic regression model where we have confidence or not :

```r
# Output a table showing the proportion of cases where we have confidence, and those w
ith no conifdence that teams would win or lose against Stanford for the logistic regre
ssion model.
table(logreg_pvalue) /length(logreg_pvalue)

## logreg_pvalue
##    Confidence No Confidence
##     0.6195965     0.3804035
```

As can be seen, for individual games, there is higher confidence of wins or loses with linear modelling. For logistic regression, we do not pick up score differentials in the model, therefore we would have less information, and thus confidence in the individual game wins. Linear model uses score differentials so gives more information and therefore confidence of individual wins.

(d) use ten-fold cross-validation to estimate the test error rate for these predictions, and also try to determine whether one model is better than the other. For each game in a given test set, there are four possible outcomes: both models are right in their prediction, both are wrong, only logistic regression is right, or only linear regression is right. Make a 2*2 contingency table displaying how often each of these four outcomes occur, over all the test examples in all ten folds of cross-validation.

Note, issues were seen when calculating section 4(d) and 4(e) in R Markdown so the R code results are pasted directly below. For reproducibility, the full question 4 code can be seen in the appendix.

In order to optimize the splitting of games into folds we will use the library caret(). In the caret() function createFolds(), the random sampling is done within the levels of y (home games teams) in an attempt to balance the class distributions within the folds. We can ensure no one team is disproportionally placed in one fold. If this was not done we may have cases where a team is disproportionally bunched in one fold preventing a good prediction for that team. Ideally, we would also consider away game teams in this proportional split, however there was insufficient time to find such a method.

```
> ### Define a matrix to hold predicted coeffients per team.
> set.seed(1)
> coefficients = matrix(,nrow=nrow(games), ncol=4)
>
> ### Create an index vector to identify train & test sets within the games of interest
only
> ### ie. For consistency with previous questions, include only regular season games, a
nd teams which played over 5 games
> index <- as.numeric(rownames(X[subset.vector,]))
>
> library(caret)
Loading required package: lattice
Loading required package: ggplot2
> folds <- createFolds(y=games$home[index], k=10, list=TRUE, returnTrain=TRUE)
```

Next we loop through each of the folds and store the predictions for each of the games.

```
> ### loop through each fold to model with train and store the results for test
> for(i in 1:10){
+
+ ### For ease of use, store the test and train indices for each loop
+         indtest  = index[-folds[[i]]]
+         indtrain = index[folds[[i]]]
+
+ ### Using the train fold only, fit log_reg and lin_mod
+         CVlog.mod <- glm(z ~ 0 + homeAdv + ., data=X, family=binomial, subset=indtrai
n)
+         CVlm.mod  <- lm(y ~ 0 + homeAdvlm + ., data=X, subset=indtrain)
+
+ ### Assign the predicted coefficients per team to the test fold
+         coefficients[indtest,1] <- coef(CVlog.mod)[paste("`",games$home[indtest],"`",
sep="")]
+         coefficients[indtest,2] <- coef(CVlog.mod)[paste("`",games$away[indtest],"`",
sep="")]
+         coefficients[indtest,3] <- coef(CVlm.mod)[paste("`",games$home[indtest],"`",s
ep="")]
+         coefficients[indtest,4] <- coef(CVlm.mod)[paste("`",games$away[indtest],"`",s
ep="")]
+         }
>
```

We then create a vector, one for each of logistic regression and linear model, to hold whether each game was predicted correctly or not. **These are both added to a contingency table of results which can be seen below.**

Note there are 32 games (from our index of games with over 5 plays and regular season only) which are not contained in the contingency table. I believe these are games for which the model did not predict the team coefficient due to the fold splitting issue mentioned at the start of the 4(d) answer.

```
> ### create vectors to hold success of the logistic regression and linear model
> winnerlog <- rep(NA, nrow(games))
> winnerlm  <- rep(NA, nrow(games))
>
> winnerlog[(games$homeScore>games$awayScore) == (coefficients[,1]>coefficients[,2])] <
- "logistic right"
> winnerlog[(games$homeScore>games$awayScore) != (coefficients[,1]>coefficients[,2])] <
- "logistic wrong"
> winnerlm[(games$homeScore>games$awayScore) == (coefficients[,3]>coefficients[,4])] <-
"linear right"
> winnerlm[(games$homeScore>games$awayScore) != (coefficients[,3]>coefficients[,4])] <-
"linear wrong"
>
> ### Create contingency table of actual result, linear model, logistic regression
> table(winnerlm,winnerlog)
              winnerlog
winnerlm       logistic right logistic wrong
  linear right           3577            274
  linear wrong            230           1207
> ### Check if all the indexed games of interest (5 or more plays and regular season) a
re in the contingency table.
> length(index) - sum(table(winnerlm,winnerlog))
[1] 32
```

(e) n11 and n22 don't tell us anything about which model is better, because they correspond to games where both models agree with each other. So to compare the two models, we need to look at n12 and n21. Let D = n12 + n21 be the number of test games in which the two models disagreed. If both models are equally good and the test set games are independent, then every time the models disagree, each model is equally likely to be right. Then, conditional on D, n12 ~ Binom(D; 1=2) For large D, the above binomial distribution is approximately normal with mean D/2 and variance D/4 (hence standard deviation sqrt(D)=2). You do not have to prove any of the above statements, just take them as given.

Use the normal approximation to carry out a test of the hypothesis that both models are equally good at predicting games. What is the conclusion of your test? What you just did is called McNemar's Test, and it is the correct way of comparing the performance of two classiffiers on a test set.

We will first look at the confidence intervals of the two intervals being the same. It can be seen that the contingency table results fall within the 95% confidence interval, meaning we cannot reject the null hypothesis that both models are equally good at predicting games.

```
> hypTest <- function(contingency){
+   D = contingency[1,2] + contingency[2,1]
+
+   lower_int = D/2 - 1.96*(sqrt(D)/2)
+   upper_int = D/2 + 1.96*(sqrt(D)/2)
+   cat('The confidence interval of D :', c(lower_int, upper_int))
+
+ }
>
> hypTest(table(winnerlm,winnerlog))
The confidence interval of D : 229.9991 274.0009
```

We next perform an Exact McNemar test (given the large samples), using R package 'exact2x2', which calculates the exact McNemar's test with appropriate matching confidence intervals. A p-value of

.05534 is seen using the contingency table. With this, there is insufficient evidence to reject the zero hypothesis using 95% confidence level, that both models are equally good at predicting games.

```
> library(exact2x2)
Loading required package: exactci
Loading required package: ssanv
> mcnemar.exact(as.matrix(table(winnerlm,winnerlog)),y=NULL, conf.level=.95)

        Exact McNemar test (with central confidence intervals)

data:  as.matrix(table(winnerlm, winnerlog))
b = 274, c = 230, p-value = 0.05534
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 0.9961101 1.4257780
sample estimates:
odds ratio
  1.191304
```

Appendix

(A)

For reproducibility purposes, below can be seen the code used in question 4 :

```
### Part A ###
### Loading the Data & identify all teams and sort by name
games <- read.csv("http://www.stanford.edu/~wfithian/games.csv",as.is=TRUE)
teams <- read.csv("http://www.stanford.edu/~wfithian/teams.csv",as.is=TRUE)
all.teams <- sort(unique(c(teams$team,games$home,games$away)))

### Assign score win or loss to vector z. This will be used a response in the model.
z <- with(games, ifelse(homeScore>awayScore,1,0))

### Initialise a data frame of the games(rows) and teams (columns),
### indicating for each game which teams were home or away
X0 <- as.data.frame(matrix(0,nrow(games),length(all.teams)))
names(X0) <- all.teams

### Assign to the data frame for each game which team was home or away
for(tm in all.teams) {
  X0[[tm]] <- 1*(games$home==tm) - 1*(games$away==tm)
}

### Remove stanford's column to make it the baseline team against which other teams
will be compared
X <- X0[,names(X0) != "stanford-cardinal"]
reg.season.games <- which(games$gameType=="REG")

### Assign a homeadvantage coefficient which is common to all games
homeAdv <- 1 - games$neutralLocation

### Perform a the logistic regression
logrega.mod <- glm(z ~ 0 + homeAdv + ., data=X, family=binomial,
subset=reg.season.games)

### Idenfity the top 25 teams
margin.top25 <- order(coef(summary(logrega.mod))[,1],decreasing=TRUE)[1:25]
coef(summary(logrega.mod))[margin.top25,1]

### Draw up a table of the two top teams, how many games they were in as home and away
### It can be seen that both team only played 1 away game, where they won.
table(X[,c("saint-mary-saint-mary","st.-thomas-(tx)-celts")])

### Look at the performance in those games, both teams won (Sevvandi's code)
bool.condition.1=games$home=="saint-mary-saint-mary"|games$away=="saint-mary-saint-
mary"
games[bool.condition.1,]

bool.condition.2=games$home=="st.-thomas-(tx)-celts"|games$away=="st.-thomas-(tx)-
celts"
games[bool.condition.2,]


### Part B ###
### Add to table "games" column of the minimum number of games played by either team
### Excluding teams with less than 5 games we have two choices. Exclude the teams
(columns), but leave their games.
### Or exclude the games they played in. I have chosen the latter for two reasons. 1)
Standofrd was our baseline
```

```
### by removing the team columns; this baseline will no longer be valid if many other
team columns are excluded.
### 2) If low game teams are excluded, hoever their games are left in - the
opposition's result (win or lose)
### will still be available as a predictor; however the strength of the low playing
team they played will be lost.
### In the team ranging using logistic regression, there are minor differences seen in
either approach once
### the logistic regression ranking is made.

teamplays <- table(c(games$home, games$away))
games$mingames[reg.season.games] <- vapply(reg.season.games, function(i) {
  min(
    teamplays[names(teamplays)==games$away[i]],
    teamplays[names(teamplays)==games$home[i]]
  )
}, 1)

### Create subset index on both regular season games and minimum games
subset.vector <- which(games$gameType=="REG" & games$mingames >= 5)


### Perform the logistic regression excluding teams with 5 games or less and non
regular season games
logregb.mod <- glm(z ~ 0 + homeAdv + ., data=X, family=binomial, subset=subset.vector)
logregb.coef <- coef(logregb.mod)[paste("`",teams$team,"`",sep="")]
names(logregb.coef) <- teams$team

### Calculate the Linear Model from class, change the variable names
y <- with(games, homeScore-awayScore)
homeAdvlm <- 1 - games$neutralLocation
lmb.mod <- lm(y ~ 0 + homeAdvlm + ., data=X, subset=subset.vector)
lmb.coef <- coef(lmb.mod)[paste("`",teams$team,"`",sep="")]
names(lmb.coef) <- teams$team

### Create the ranking table
rank.table <- cbind(
#                   "LinReg Model Score" = lmb.coef,
                    "LinReg Model Rank"  = rank(-lmb.coef,ties="min"),
#                   "LogReg Model Score" = logregb.coef,
                    "LogReg Model Rank"  = rank(-logregb.coef,ties="min"),
                    "AP Rank"     = teams$apRank,
                    "USAT Rank"   = teams$usaTodayRank)
rank.table[order(logregb.coef,decreasing=TRUE)[1:25],]


### Part C ###

par(mfrow=c(1,2))
plot(coef(summary(lmb.mod))[-1,1], coef(summary(lmb.mod))[-1,4], xlab="Slope Estimate
(Linear Model)", ylab= "p-value of performance", col="blue")
abline(h = .05, col = "red")
abline(v = 0, lty=2)
text(-30,.1, "p-value = 0.05", col = "red")
plot(coef(summary(logregb.mod))[-1,1], coef(summary(logregb.mod))[-1,4],xlab="Slope
Estimate (Logistic Reg.)", ylab= "p-value of performance", col="blue")
abline(h = .05, col = "red")
abline(v = 0, lty=2)

### Find the number of p-values greater or less than .05
linmod_pvalue <- (ifelse(coef(summary(lmb.mod))[,4]<.05,"Y", "N"))
logreg_pvalue <- (ifelse(coef(summary(logregb.mod))[,4]<.05,"Y", "N"))

### For logistic regression, we do not pick up score differentials in the mode.
### Therefore, we would have less confidence in the individual game wins.
### Linear model uses score differentials so gives more confidence of wins.
table(linmod_pvalue)/length(linmod_pvalue)
table(logreg_pvalue)/length(logreg_pvalue)

### Part D
### Use ten-fold cross-validation to estimate the test error rate for these
predictions,
### Make a 2*2 contingency table of four possible outcomes: both models are right in
their prediction, both are wrong,
### only logistic regression is right, or only linear regression is right.

### Define a matrix to hold predicted coeffients per team.
set.seed(1)
coefficients = matrix(,nrow=nrow(games), ncol=4)

### Create an index vector to identify train & test sets within the games of interest
only
### ie. For consistency with previous questions, include only regular season games, and
teams which played over 5 games
index <- as.numeric(rownames(X[subset.vector,]))

### Break the games into ten folds. Use the caret library to make the split
### For numeric y, the sample is split into groups sections based on percentiles
### and sampling is done within these subgroups. For smaller samples sizes, these two
functions
### may not do stratified splitting and, at most, will split the data into quartiles.
So in our case,
### there will be a proportional split of teams into train and test (we will not have
all home games of
### one team grouped in one test fold).
library(caret)
folds <- createFolds(y=games$home[index], k=10, list=TRUE, returnTrain=TRUE)

### loop through each fold to model with train and store the results for test
for(i in 1:10){

### For ease of use, store the test and train indices for each loop
        indtest  = index[-folds[[i]]]
        indtrain = index[folds[[i]]]

### Using the train fold only, fit log_reg and lin_mod
        CVlog.mod <- glm(z ~ 0 + homeAdv + ., data=X, family=binomial, subset=indtrain)
        CVlm.mod  <- lm(y ~ 0 + homeAdvlm + ., data=X, subset=indtrain)

### Assign the predicted coefficients per team to the test fold
        coefficients[indtest,1] <-
coef(CVlog.mod)[paste("`",games$home[indtest],"`",sep="")]
        coefficients[indtest,2] <-
coef(CVlog.mod)[paste("`",games$away[indtest],"`",sep="")]
        coefficients[indtest,3] <-
coef(CVlm.mod)[paste("`",games$home[indtest],"`",sep="")]
        coefficients[indtest,4] <-
coef(CVlm.mod)[paste("`",games$away[indtest],"`",sep="")]
        }

### create vectors to hold success of the logistic regression and linear model
winnerlog <- rep(NA, nrow(games))
winnerlm  <- rep(NA, nrow(games))

winnerlog[(games$homeScore>games$awayScore) == (coefficients[,1]>coefficients[,2])] <-
"logistic right"
winnerlog[(games$homeScore>games$awayScore) != (coefficients[,1]>coefficients[,2])] <-
"logistic wrong"
winnerlm[(games$homeScore>games$awayScore) == (coefficients[,3]>coefficients[,4])] <-
"linear right"
winnerlm[(games$homeScore>games$awayScore) != (coefficients[,3]>coefficients[,4])] <-
"linear wrong"

### Create contingency table of actual result, linear model, logistic regression
```

```
table(winnerlm,winnerlog)

### Check if all the indexed games of interest (5 or more plays and regular season) are
in the contingency table.
length(index) - sum(table(winnerlm,winnerlog))

### Part E
### Use the normal approximation to carry out a test of the hypothesis that both models
are equally good at
### predicting games. What is the conclusion of your test?
### For the Hypothesis test we use a 95% confidence interval, mean ± 2 · SE, where D =
sum of test games in which
### the two models disagreed, mean = D/2 and SE = Sqrt(D)/2

### this function will take the contingency table as input and calculate the confidence
interval for the
### errors given of D value.
hypTest <- function(contingency){
  D = contingency[1,2] + contingency[2,1]

  lower_int = D/2 - 1.96*(sqrt(D)/2)
  upper_int = D/2 + 1.96*(sqrt(D)/2)
  cat('The confidence interval of D :', c(lower_int, upper_int))

}

hypTest(table(winnerlm,winnerlog))

### We next perform an Exact McNemar test (given the large samples), using R package
'exact2x2', which calculates
### the exact McNemar's test with appropriate matching confidence intervals. A p-value
of .05534 given the contingency table.
### With this, there is insufficient evidence to reject the zero hypothesis, that both
models are qually good
### at predicting games.

library(exact2x2)
mcnemar.exact(as.matrix(table(winnerlm,winnerlog)),y=NULL, conf.level=.95)
```