Logistic Regression

Andrew Flint, Cole Lyman, Corinne Sexton, and Kevin Toney

October 12, 2017

Problem 1

Libraries

```
library(dummies)
library(glmnet)
```

Read in the data

```
icudata <- read.csv("data/icudata.csv")
```

Fit Logistic Model (1a)

```
model <- glm(as.factor(STA) ~ as.factor(CPR) + AGE + as.factor(RACE) + SYS + HRA + as.
factor(TYP), icudata, family = 'binomial')

# output the coefficients
model$coefficients
```

```
##      (Intercept)  as.factor(CPR)1             AGE as.factor(RACE)2
##     -3.121326454      1.387793431      0.035029828     -0.826682213
## as.factor(RACE)3              SYS              HRA  as.factor(TYP)1
##      0.264005681     -0.012837227     -0.007946038      2.303171699
```

Effect of CPR on survival (1b)

CPR is the second best indicator of survival after the type of admission. Thus, having CPR increases your chance of survival greatly.

Create LASSO Model (1c)

Need to create dummy variables because LASSO only take numeric values (no categorical/factors) Need to remove the first column so the model is identifiable.

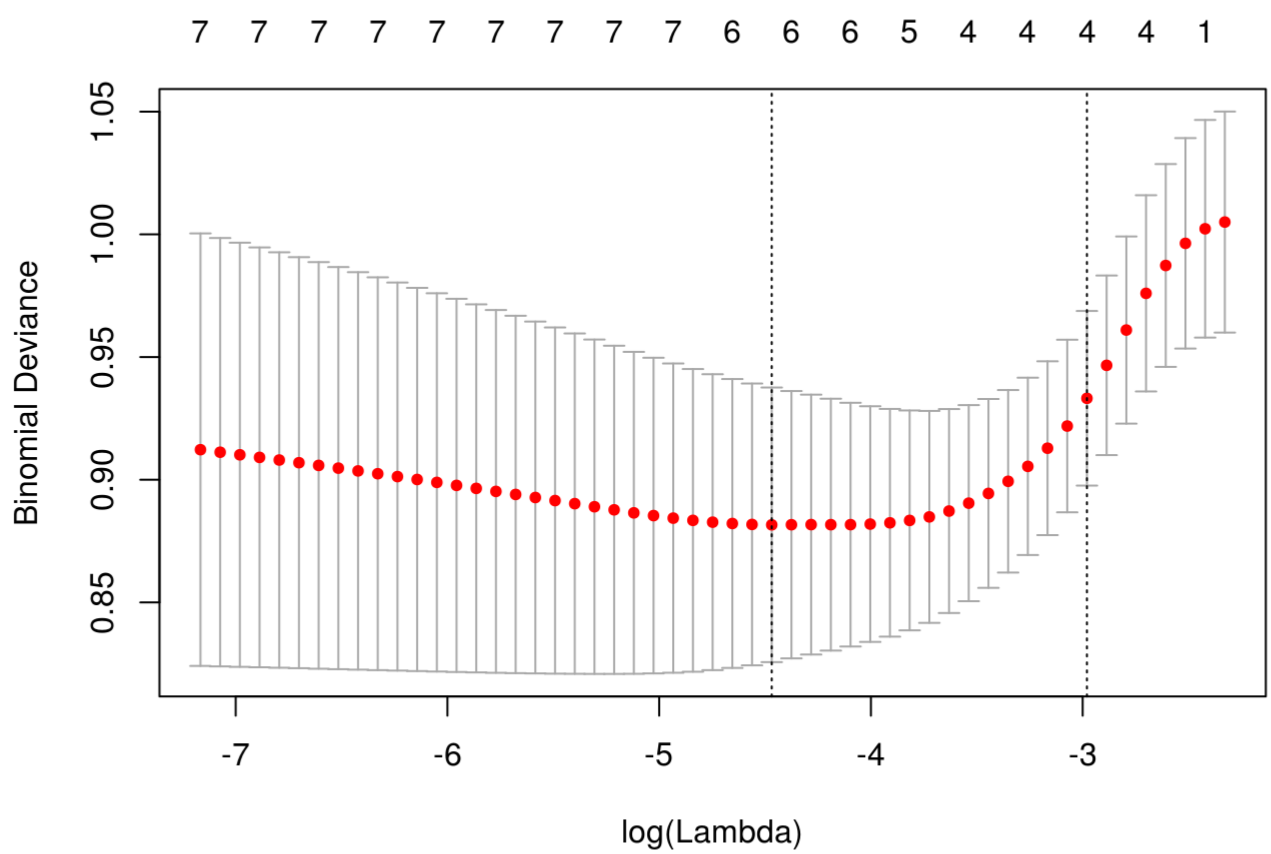
```
CPR_dummy <- dummies::dummy(icudata$CPR)
CPR_dummy <- CPR_dummy[ , -1]

RACE_dummy <- dummies::dummy(icudata$RACE)
RACE_dummy <- RACE_dummy[ , -1]

TYP_dummy <- dummies::dummy(icudata$TYP)
TYP_dummy <- TYP_dummy[ , -1]
```

Create X and y matrix/vector

```
X <- as.matrix(icudata[ , c("AGE", "SYS", "HRA")])
X <- cbind(X, CPR_dummy, RACE_dummy, TYP_dummy)
y <- icudata$STA
```

Fit LASSO model with logistic regression

```
lasso_model <- cv.glmnet(X, y, family = 'binomial')
plot(lasso_model)
```



Find optimal value of lambda and coefficients for optimal lambda

```
lasso_model$lambda.min
```

```
## [1] 0.01146592
```

Coefficients: (1d)

```
coef(lasso_model, s = "lambda.min")
```

```
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                                                                        1
## (Intercept)                                                 -3.002058860
## AGE                                                          0.027858713
## SYS                                                         -0.010109830
## HRA                                                         -0.002601166
## CPR_dummy                                                    1.140993069
## /home/cole/Code/2017/cs494/logisticRegression/problem1.Rmd2 -0.402812800
## /home/cole/Code/2017/cs494/logisticRegression/problem1.Rmd3  .
## TYP_dummy                                                    1.726744394
```

Problem 2

Libraries

```
rm(list=ls())
library(dplyr)
library(glmnet)
library(plyr)
library(stringr)
library(jsonlite)
```

Data wrangling with the tag column (2a)

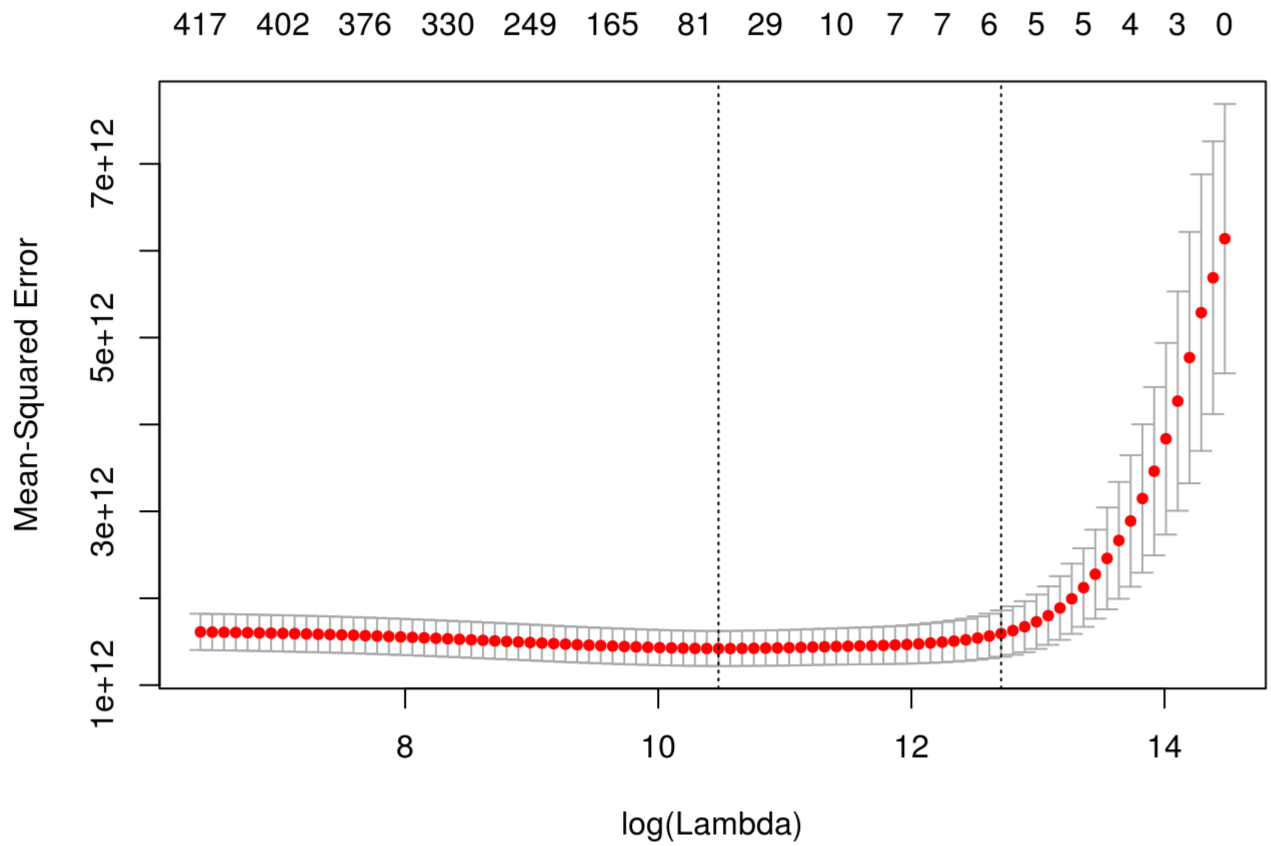
```r
ted.data <- read.csv("data/ted.csv", header = T, sep = ",")
#unique(ted.data$tags)
#each row, except 20 has a unique combination of tags.

#try splitting up the tags column by commas.
tags <- str_replace(as.character(ted.data$tags), "\\[", "")
tags <- str_replace(tags, "\\]", "")
tags <- str_replace_all(tags, "\\'", "")
tags <- str_split(tags, ", ")

diff.tags <- unique(unlist(tags))
tag.cols <- NULL
for(i in 1:length(diff.tags)) {
  tag.cols[i] <- paste("TAG_", diff.tags[i], sep = '')
}

new.data <- matrix(data=NA, nrow=nrow(ted.data), ncol=length(tag.cols))
colnames(new.data) <- tag.cols

ted.data <- cbind(ted.data, new.data)

for(i in 1:ncol(ted.data)) {
  for(j in 1:nrow(ted.data)) {
    ted.data[j,i+17] <- diff.tags[i] %in% tags[[j]]
  }
}
```

Create a new column for each rating category (2b)

```
r_names <- c("RATINGS_Funny", "RATINGS_Beautiful", "RATINGS_Ingenious", "RATINGS_Coura
geous", "RATINGS_LongWinded",
             "RATINGS_Confusing", "RATINGS_Informative", "RATINGS_Fascinating", "RATIN
GS_Unconvincing",
             "RATINGS_Persuasive","RATINGS_Jaw-Dropping",
             "RATINGS_Ok","RATINGS_Obnoxious","RATINGS_Inspiring")
rat_matrix <- matrix(NA, nrow=nrow(ted.data), ncol=length(r_names))
ted.data$ratings <- gsub("'", '"', ted.data$ratings)
ratings <- sapply(ted.data$ratings, fromJSON)
#now we got a matrix with 2,550 columns and three rows.
#this matrix goes by ratings[,col][[list#]] to get to a column first and then a row

#Standardize the rows to follow the same order.
ratingsnew <- ratings
for(i in 1:ncol(ratings)) {
  for(k in 1:3){
    if (k == 1) {
      orders = order(match(ratings[,i][[2]], ratings[,1][[2]]))
    }
    ratingsnew[,i][[k]] <- ratings[,i][[k]][orders]
  }
}

#r_names <- sort(r_names)[orders]
colnames(rat_matrix) <- r_names

#populate the new matrix of the ratings with the counts.
for(i in 1:ncol(ratings)){
  for(k in 1:ncol(rat_matrix)){
    rat_matrix[i,k] <- ratingsnew[,i][[3]][k]
  }
}

#cbind the old data frame and the ratings data frame together.
ted.data <- cbind(ted.data, rat_matrix)
```

Use LASSO to fit logistic regression (2c)

```
dummy_TAG <- dummies::dummy.data.frame(ted.data, names=grep("TAG_",names(ted.data), va
lue=T), all=F)
# eliminate extras (FALSE COLUMNS)
dummy_TAG2 <- dummy_TAG[ , grep('TRUE', colnames(dummy_TAG))]



X <- as.matrix(ted.data[ , grep('RATINGS', colnames(ted.data), value = T)])
X <- cbind(X, ted.data$comments, ted.data$duration, ted.data$num_speaker,  dummy_TAG2)
X <- as.matrix(X)
y <- ted.data$views

lasso_model <- cv.glmnet(X, y, family = 'gaussian')
plot(lasso_model)
```



Optimal lambda value (2d)

```
lasso_model$lambda.min
```

```
## [1] 35457.52
```

Top 10 and Worst 10 Tags (2e)

```
a <- coef(lasso_model, s = "lambda.min")
b <- a[,1]
```

Top 10 tags

```
# max 10
head(sort(b, decreasing=T),11)
```

```
##         TAG_magicTRUE TAG_body languageTRUE       TAG_fashionTRUE
##              958274.9              750762.1              576839.4
## TAG_relationshipsTRUE       TAG_successTRUE     TAG_potentialTRUE
##              339736.1              299042.7              250780.9
##        TAG_flightTRUE        TAG_speechTRUE    TAG_wunderkindTRUE
##              237398.3              216959.7              207806.1
##           (Intercept)   TAG_performanceTRUE
##              175241.9              162139.5
```

Worst 10 tags

```
# min ten
head(sort(b),10)
```

```
##    TAG_statisticsTRUE        TAG_TED-EdTRUE          TAG_memeTRUE
##            -400256.15            -296041.81            -269150.38
## TAG_consciousnessTRUE  TAG_presentationTRUE   TAG_advertisingTRUE
##            -235208.74            -135765.24            -103070.19
##        TAG_GoogleTRUE    TAG_simplicityTRUE TAG_global issuesTRUE
##             -99786.60             -99593.20             -91046.41
##          TAG_selfTRUE
##             -90935.56
```

Least important rating (2f)

LongWinded, Persuasive, Obnoxious, Confusing, and Unconvincing all went to 0 through the LASSO test, but besides that Jaw-Dropping had the lowest coefficient (8.842249) for the rest of the ratings.

```
b[grep("RATINGS", names(b))]
```

```
##        RATINGS_Funny    RATINGS_Beautiful    RATINGS_Ingenious
##            771.68330            165.64021            634.96058
##    RATINGS_Courageous   RATINGS_LongWinded    RATINGS_Confusing
##            537.43943            -19.10856              0.00000
##  RATINGS_Informative  RATINGS_Fascinating RATINGS_Unconvincing
##           1080.78316            589.78996            -12.09196
##   RATINGS_Persuasive RATINGS_Jaw-Dropping           RATINGS_Ok
##              0.00000             15.93627           5753.79140
##    RATINGS_Obnoxious    RATINGS_Inspiring
##              0.00000            293.44017
```