# A3

Yue Han

```
library(tidyverse)
library(ggplot2)
library(glmnet)
library(FeatureHashing)
library(stats)
```

### Question 1.1

Suppose $\hat{\beta}$ is a minimizer of (0.1) and $C = |\quad|\hat{\beta}|\quad|_1$ and $\lambda > 0$. Assume minimizer of (0.2) is unique and $\hat{\beta}$ is not a minimizer of (0.2). If so then there exists an $\alpha$ s.t. $\frac{1}{2n}|\quad|Y - X\alpha|\quad|_2^2 < \frac{1}{2n}|\quad|Y - X\hat{\beta}|\quad|_2^2$ where $|\quad|\alpha|\quad|_1 \leq |\quad|\hat{\beta}|\quad|_1$ based on (0.2) however if this is true then this makes $\alpha$ a minimizer of (0.1) and contradicts our supposition that $\hat{\beta}$ is a minimizer of (0.1). Thus $\hat{\beta}$ is a minimizer of (0.2)

### Question 1.2

let \$ =

\$ where $c > 0$and let \$ =

\$ then \$||-||^2_2 = ||

||^2_2 = ^2 = |(y-X)^2 + (c)^2| = |(y-X)^2| + |(c)^2| = ||y-X||^2_2 + ||c||^2_2 = ||y-X||^2_2 + c²||||2_2 \$ so the standard lasso problem becomes $min_{\tilde{\beta}}\{\| |y - X\beta||_2^2 + c^2||\beta||_2^2 + \tilde{\lambda}||\beta||_1\}$ and by making $c = \sqrt{\lambda c}$ and make $\tilde{\lambda} = \lambda(1-\alpha)$ we get (0.3) so the optimal value for (0.4) matches (0.3).

### Question 2

a.) $l(\beta) = \Pi_i^n p(y_i = x_i|\beta) = \Pi_i^n \frac{\beta^{x_i}}{x_i!}e^{-\lambda} = \frac{\beta^{nx_i}e^{-n\beta}}{x_i!}$

b.) Want to show \$y_i = E_{}[Y|X=x_i] \$ For poisson: $E_{\hat{\beta}}[Y|X = x_i] = \hat{\beta}(x_i)$

for logisitic: $E_{\hat{\beta}}[Y|X = x_i] = \hat{\beta}(x_i)$

so \$y_i = E_{}[Y|X=x_i] \$ thus $\sum y_i x_i = \sum E_{\hat{\beta}}[Y|X = x_i]x_i$

### Question 3.1

Let $x_1$ and $x_2$ be points on hyperplane $\beta^T x - b = 1$ and $\beta^T x - b = -1$ respectively. Then $x_1 = \frac{1+b}{\beta^T}$ and $x_2 = \frac{-1+b}{\beta^T}$. Where $1 + b$ and $-1 + b$ are constants so $x_1 - x_2 = \frac{1+b+1-b}{\beta^T} = \frac{2}{\beta^T}$ is parallel to $\beta$. Then the distance can be characterized as $||x_1 - x_2||_2 = ||2/\beta^T|| = \frac{2}{||\beta||_2}$

### Question 3.2.a)

We wish to minimize this because we want to maximize the distance between the two hyperplanes

When $y_i = 1$ we have hyperplane $\beta^T x_i - b \geq 1$ and when $y_i = -1$ we have hyperplane $\beta^T x_i - b \leq 1$ which can be put together to form $y_i(\beta^T x_i - b) \geq 1$

### Question 3.2.b.)

let d = 2 with 3 data points: $([x, x^2], 1), ([x + 1, (x + 1)^2], 1), ([x + 2, (x + 2)^2], 1)$ then

### question 3.3

Suppose 0.5 then we can implement 0.6 to minimize 0.5. By adding $C\sum\zeta_i$ we are allowing for a point to be on the wrong side of the hyperplane but since we are limiting $y_i(\beta ^T x_i -b) \geq 1- \zet _i$ we allow a small distance and it still follows the constraint. Thus it allows for a feasible solution. Thus there is a feasible solution that optimizes 0.7

### Question 4

```
load("_data/q1.RData")

head(dataTrainAll)

##                             Bid_ID                        iPinYou_ID Region
City
## 1 25c85878563f807f963c56ab260bec66 ca05a95e57fca551be0a8d0b76103aeb      1
1
## 2 a66f205776a83996a482bec445b0c59a 2d859de350c6353d4f3a744acde99213      1
1
## 3 7cd3e32d3eae34a28c733a98b05cb050 97a911a5b51b343abee414518ea99158      6
6
## 4 be77c8266500dcc23bcfb439fb504d75 490229c9c7ca3796e66f1e3f3a167ded      1
2
## 5 af6614de2b1fcc597bc986ab78ed3a13 332fdd78d9b5b8c1952e5f481f52e90b      6
6
## 6 d46fa26badbf330ef06821983cc0112b 2a902915d26797db09be1779d969c573      1
1
##   AdX                Domain                              URL Anon_URL
## 1   2 trqRTu5Jg9q9wMKYvmpENpn  2587261eb65b974b53d27120e83e624     null
## 2   3    trqRTudNXqN8ggc4JKTI 1366d94fa5a43db402d43332d4556d6d     null
## 3   3    trqRTudNXqN8ggc4JKTI 357703906681ec497e6236f6dde7a3c7     null
## 4   2    trqRTudNXqN8ggc4JKTI 357703906681ec497e6236f6dde7a3c7     null
## 5   1    trqRTudNXqN8ggc4JKTI f707ba60373c1c34a49acdf5f8beaacd     null
## 6   2       5Fa-expoBTTR1m58uG  36554aef0b031d0950a30d3f9e51b14     null
##   Ad_Width Ad_Height Ad_Vis Ad_Form Floor_Price iPinYou_Bid Comp_Bid
## 1      728       250      1       0           5           5       15
```

```
## 2     1000        250       2        1           15           15         25
## 3      200        250       0        0           30           30         60
## 4      200         90       0        0            0           35         50
## 5      300        250       1        0            0          280        285
## 6      300         90       0        0            5           20         15
##                              Key_Page Click Conv
## 1 3a7eb50444df6f61b2409f4e2f16b687     0    0
## 2 9f4e2f16b6873a7eb504df6f61b24044     0    0
## 3 3a7eb50444df6f61b2409f4e2f16b687     0    0
## 4 3a7eb50444df6f61b2409f4e2f16b687     0    0
## 5 df6f61b2409f4e2f16b6873a7eb50444     0    0
## 6 df6f61b2409f4e2f16b6873a7eb50444     0    0
```

Region 1 is 11 Region 3 is 10 Region 6 is 01

```
Region0 = as.numeric(dataTrainAll$Region==1 | dataTrainAll$Region==3)
training = data.frame(Region0)
Region1 = as.numeric(dataTrainAll$Region==1 | dataTrainAll$Region==6)
training = cbind(training, Region1)
head(training)
```

```
##   Region0 Region1
## 1       1       1
## 2       1       1
## 3       0       1
## 4       1       1
## 5       0       1
## 6       1       1
```

City 1 is 10000 City 2 is 01000 City 3 is 00100 City 4 is 00010 City 5 is 00001 City 6 is 00000

```
City0 = as.numeric(dataTrainAll$City==1)
City1 = as.numeric(dataTrainAll$City==2)
City2 = as.numeric(dataTrainAll$City==3)
City3 = as.numeric(dataTrainAll$City==4)
City4 = as.numeric(dataTrainAll$City==5)
training = cbind(training, City0,City1, City2, City3, City4)
```

AdX 1 is 10 AdX 2 is 01 AdX 3 is 00

```
AdX0 = as.numeric(dataTrainAll$AdX==1)
AdX1 = as.numeric(dataTrainAll$AdX==2)
training = cbind(training, AdX0, AdX1)
```

Domain 5Fa-expoBTTR1m58uG is 1000 Domain 5KFUl5p0Gxsvgmd4wspENpn is 0100 Domain trqRTu5Jg9q9wMKYvmpENpn is 0010 Domain trqRTudNXqN8ggc4JKTI is 0001 Domain trqRTuT-GNTYJNKbuKz is 0000

```
Domain0 = as.numeric(dataTrainAll$Domain=="5Fa-expoBTTR1m58uG")
Domain1 = as.numeric(dataTrainAll$Domain=="5KFUl5p0Gxsvgmd4wspENpn")
```

```r
Domain2 = as.numeric(dataTrainAll$Domain=="trqRTu5Jg9q9wMKYvmpENpn")
Domain3 = as.numeric(dataTrainAll$Domain=="trqRTudNXqN8ggc4JKTI")
training = cbind(training, Domain0, Domain1, Domain2, Domain3)
```

Key_Page 3a7eb50444df6f61b2409f4e2f16b687 is 10 Key_Page 9f4e2f16b6873a7eb504df6f61b24044 is 01 Key_Page df6f61b2409f4e2f16b6873a7eb50444 is 00

```r
Key_Page0 =
as.numeric(dataTrainAll$Key_Page=="3a7eb50444df6f61b2409f4e2f16b687")
Key_Page1 =
as.numeric(dataTrainAll$Key_Page=="9f4e2f16b6873a7eb504df6f61b24044")
training = cbind(training, Key_Page0, Key_Page1)
```

Ad_Vis 0 is 10 Ad_Vis 1 is 01 Ad_Vis 2 is 00

```r
Ad_Vis0 = as.numeric(dataTrainAll$Ad_Vis==0)
Ad_Vis1 = as.numeric(dataTrainAll$Ad_Vis==1)
training = cbind(training, Ad_Vis0, Ad_Vis1)
```

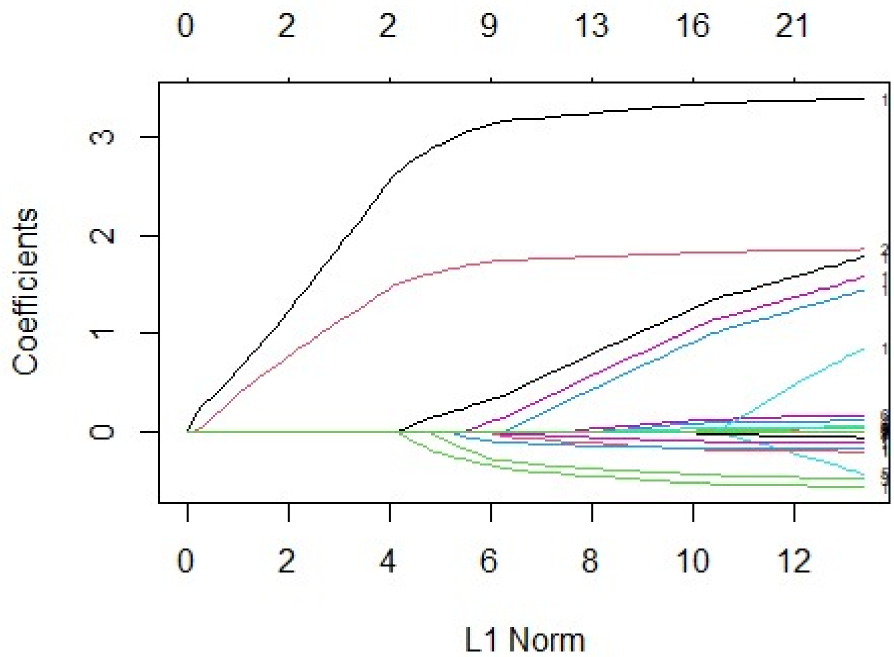
Ad_Form 0 is characterized as 0 and Ad_Form 1 is characterized as 1

```r
Ad_Form = as.numeric(dataTrainAll$Ad_Form==1)
training = cbind(training,Ad_Form)
head(training)

##   Region0 Region1 City0 City1 City2 City3 City4 AdX0 AdX1 Domain0 Domain1
## 1       1       1     1     0     0     0     0    0    1       0       0
## 2       1       1     1     0     0     0     0    0    0       0       0
## 3       0       1     0     0     0     0     0    0    0       0       0
## 4       1       1     0     1     0     0     0    0    1       0       0
## 5       0       1     0     0     0     0     0    1    0       0       0
## 6       1       1     1     0     0     0     0    0    1       1       0
##   Domain2 Domain3 Key_Page0 Key_Page1 Ad_Vis0 Ad_Vis1 Ad_Form
## 1       1       0         1         0       0       1       0
## 2       0       1         0         1       0       0       1
## 3       0       1         1         0       1       0       0
## 4       0       1         1         0       1       0       0
## 5       0       1         0         0       0       1       0
## 6       0       0         0         0       1       0       0

Ad_Width = (dataTrainAll$Ad_Width -
mean(dataTrainAll$Ad_Width))/sd(dataTrainAll$Ad_Width)
training = cbind(training, Ad_Width)

Ad_Height = (dataTrainAll$Ad_Height -
mean(dataTrainAll$Ad_Height))/sd(dataTrainAll$Ad_Height)
training = cbind(training, Ad_Height)

Floor_Price = (dataTrainAll$Floor_Price -
mean(dataTrainAll$Floor_Price))/sd(dataTrainAll$Floor_Price)
training = cbind(training, Floor_Price)
```

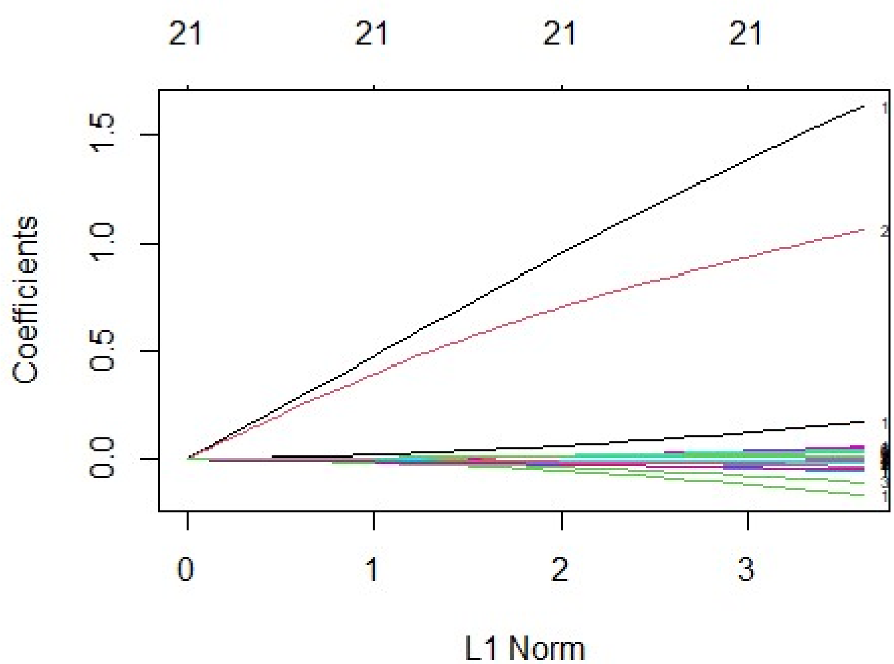
```
head(training)

##   Region0 Region1 City0 City1 City2 City3 City4 AdX0 AdX1 Domain0 Domain1
## 1       1       1     1     0     0     0     0    0    1       0       0
## 2       1       1     1     0     0     0     0    0    0       0       0
## 3       0       1     0     0     0     0     0    0    0       0       0
## 4       1       1     0     1     0     0     0    0    1       0       0
## 5       0       1     0     0     0     0     0    1    0       0       0
## 6       1       1     1     0     0     0     0    0    1       1       0
##   Domain2 Domain3 Key_Page0 Key_Page1 Ad_Vis0 Ad_Vis1 Ad_Form   Ad_Width
## 1       1       0         1         0       0       1       0  0.9635290
## 2       0       1         0         1       0       0       1  1.9635194
## 3       0       1         1         0       1       0       0 -0.9776289
## 4       0       1         1         0       1       0       0 -0.9776289
## 5       0       1         0         0       0       1       0 -0.6099853
## 6       0       0         0         0       1       0       0 -0.6099853
##    Ad_Height Floor_Price
## 1  0.3312922  -0.3132385
## 2  0.3312922   0.2422080
## 3  0.3312922   1.0753778
## 4 -1.0242751  -0.5909618
## 5  0.3312922  -0.5909618
## 6 -1.0242751  -0.3132385

Click = as.numeric(dataTrainAll$Click>0)
Click = data.frame(Click)
head(Click)

##   Click
## 1     0
## 2     0
## 3     0
## 4     0
## 5     0
## 6     0
```

Question 4.1.a.)

```
lasso = glmnet(x=data.matrix(training), y=Click$Click, family = binomial,
alpha = 1,  standardize=FALSE) #Lasso

ridge = glmnet(x=data.matrix(training), y=Click$Click, family = binomial,
alpha = 0,  standardize=FALSE) #Ridge

plot(lasso,label = TRUE)
```




```
plot(ridge, label =T)
```



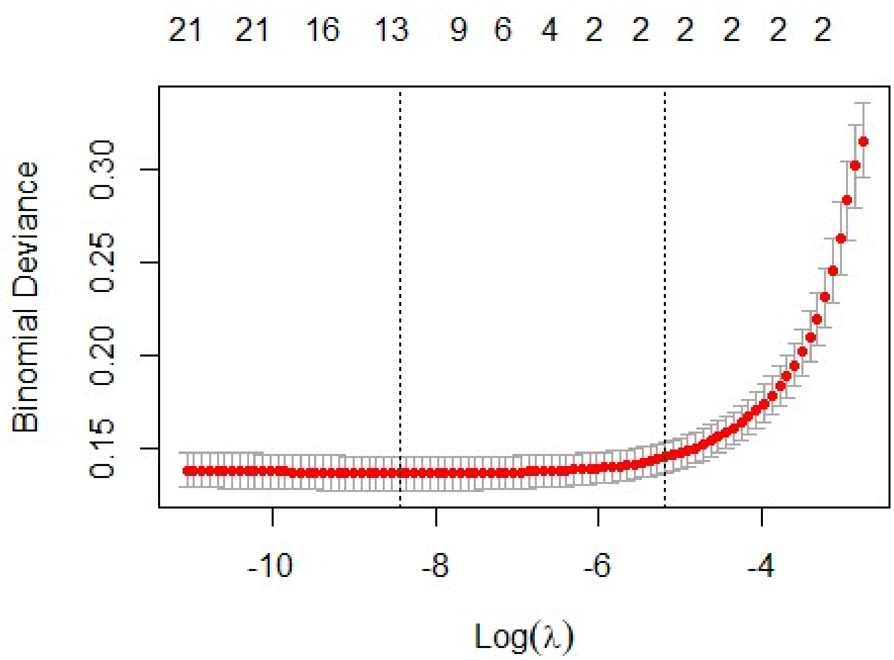

### Question
4.1.b.) Lasso plot the most important features are 19 and 20 since they the highest coefficient values throughout the change of L1 norm of lambda

Ridge plot the most important features are 19 and 20 also with the same reason as above. So from both graphs coefficients 19 and 20 are the most important. These two coefficients are Ad_Width and Ad_Height. ### Question 4.1.c.)

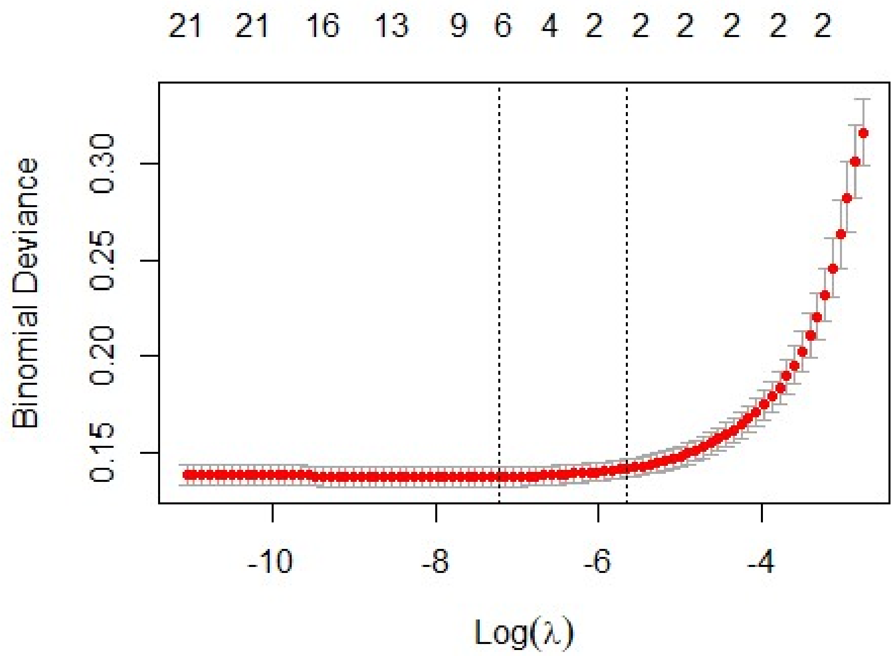
```
lasso_cv = cv.glmnet(x=data.matrix(training), y=Click$Click, nfolds = 5,
standardize=FALSE, family = "binomial" )

ridge_cv = cv.glmnet(x=data.matrix(training), y=Click$Click, nfolds = 5,
standardize=FALSE, family = "binomial")
```

Plot of lasso and ridge binomial deviance against log$\lambda$ The left most dotted line is the lambda chosen where there is the least binomial deviance.

```
plot(lasso_cv)
```



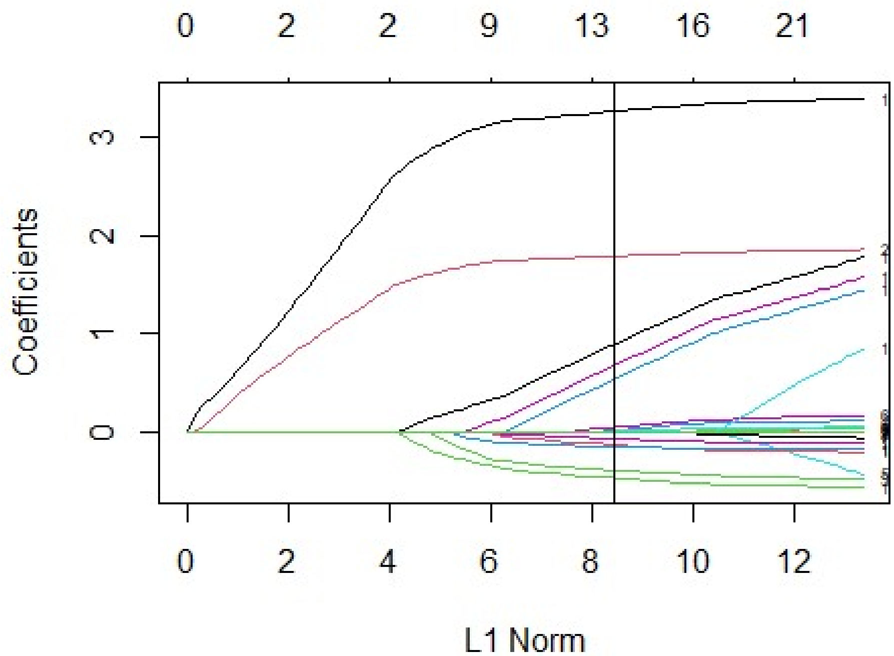

```
plot(ridge_cv)
```




```
lasso_lambda = lasso_cv$lambda.min
ridge_lambda = ridge_cv$lambda.min
```

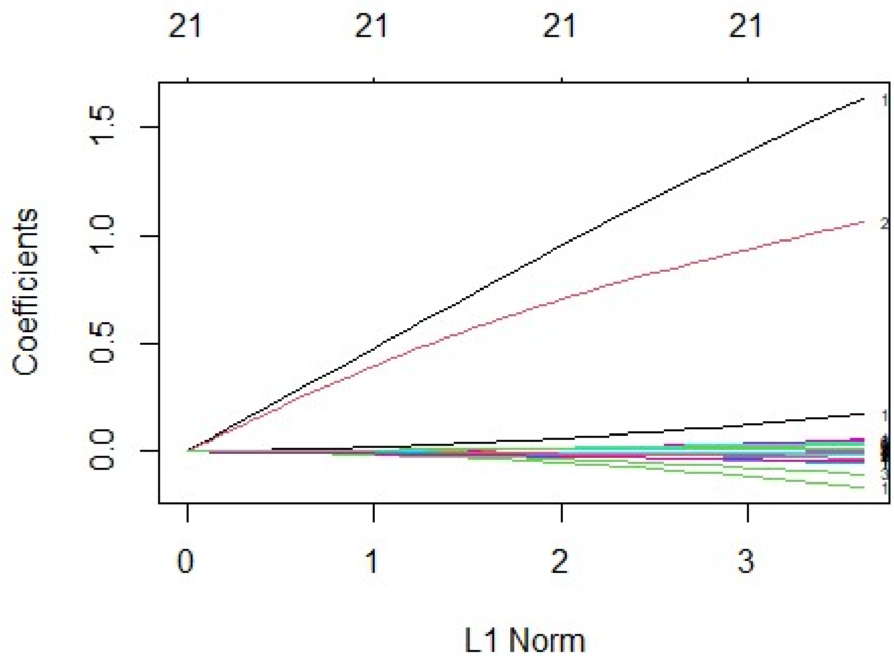
Plot of lasso with L1 norm chosen labeled we see the most important featues are 19 and 20 and there is plateau of coefficient value for these 2 features after the L1 norm chosen.

```
plot(lasso,label = TRUE)
abline(v=abs(log(lasso_lambda)))
```




L1 norm chosen is unable to be seen on the plot.

```
plot(ridge,label = TRUE)
abline(v=abs(log(ridge_lambda)))
```




As we see in both plots only a few features are important so increasing degrees of freedom does not necessarily make cross validation better

### Question 4.2

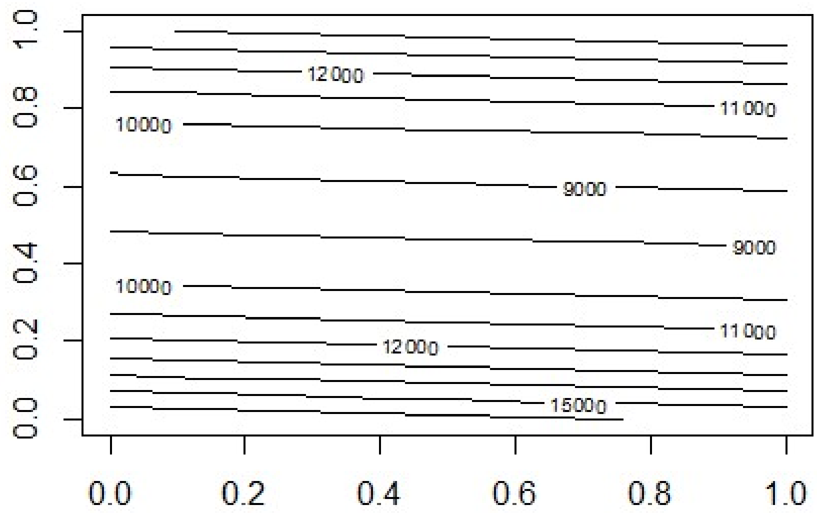
```
AdX_stand = (dataTrainAll$AdX - mean(dataTrainAll$AdX))/sd(dataTrainAll$AdX)
iPin_stand = (dataTrainAll$iPinYou_Bid -
mean(dataTrainAll$iPinYou_Bid))/sd(dataTrainAll$iPinYou_Bid)
Comp_stand = (dataTrainAll$Comp_Bid -
mean(dataTrainAll$Comp_Bid))/sd(dataTrainAll$Comp_Bid)
pred_ex = data.frame(cbind(AdX_stand, iPin_stand, Comp_stand))
head(pred_ex)

##    AdX_stand iPin_stand Comp_stand
## 1  0.0480746 -1.0502765 -1.0245268
## 2  1.3198895 -0.9118871 -0.8622491
## 3  1.3198895 -0.7043028 -0.2942775
## 4  0.0480746 -0.6351081 -0.4565551
## 5 -1.2237403  2.7554343  3.3569689
## 6  0.0480746 -0.8426923 -1.0245268

pred =lm(Comp_stand ~ AdX_stand + iPin_stand, data = pred_ex )
pred

##
## Call:
## lm(formula = Comp_stand ~ AdX_stand + iPin_stand, data = pred_ex)
##
```

```
## Coefficients:
## (Intercept)    AdX_stand   iPin_stand
##  -9.432e-16   -1.664e-01    7.722e-01
```

```
glm_pred = glmnet(x=data.matrix(pred_ex), y=pred_ex$Comp_stand, lambda =
1,family = "gaussian",standardize = FALSE)
coef(glm_pred, s= 0.5*sum(abs(coef(pred))))
```

```
## 4 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept) -1.717376e-16
## AdX_stand    0.000000e+00
## iPin_stand   .
## Comp_stand   .
```

```
beta1 = seq(-.5,1,length.out=100)
beta2 = seq(-.5,1,length.out=100)

mse <- function(b1,b2) {
  m = sum((pred_ex$Comp_stand -(b1*pred_ex$AdX_stand + b2 *
pred_ex$iPin_stand))^2)
  return(m)
}

mses = NULL
for (i in 1:length(beta1)){
  mses = c(mses , mse(beta1[i], beta2[i]))

}

contour(matrix(mses,5,20))
```




### Question 5

```
grav_data <- read.table("_data/LIGO.Hanford.Data.txt", header = F, col.names
= c("time", "strain"))
head(grav_data)

##         time       strain
## 1 0.2500000  0.026422536
## 2 0.2500610 -0.003132572
## 3 0.2501221 -0.130760718
## 4 0.2501831 -0.061991781
## 5 0.2502441  0.019565419
## 6 0.2503052  0.022075875

set.seed(10)
glm_time = glmnet(x=data.matrix(grav_data), y=grav_data$time, lambda =
1,family = "gaussian",standardize = FALSE)
cv_time = cv.glmnet(x=data.matrix(grav_data), y=grav_data$time, nfolds = 10,
family = "gaussian")
lambda_time = cv_time$lambda.min
```

$\hat{w}$ is more sparse because it is lasso which is know for sparcity.

### Question 6

```
library(png)

rand_pos = paste("_data/pngdata/pos/",as.character(sample(1:500,1)), ".png",
sep="")
```

```
rand_neg = paste("_data/pngdata/neg/",as.character(sample(1:500,1)), ".png",
sep="")

pos = readPNG(rand_pos)
neg = readPNG(rand_neg)

writePNG(pos, target = "posOrg.png")
writePNG(neg, target = "negOrg.png")
```

*Pos Image Original*

*Neg Image Original*

```
source("_data/functions.R")

pos = rgb2gray(pos)
neg = rgb2gray(neg)
```

```
writePNG(pos, target = "posGray.png")
writePNG(neg, target = "negGray.png")
```

*Pos Image rgb2gray*

*Neg Image rgb2gray*

```
neg = crop.r(neg, 160,96)

writePNG(neg, target = "negCrop.png")
```

*Neg Image Cropped*

```
gField_pos = grad(pos,128,64,F)
gField_neg = grad(neg,128,64,F)


setEPS()
postscript("gPos.eps")
g_Pos=grad(pos, 128, 64, T)
dev.off()

## png
##   2

setEPS()
postscript("gNeg.eps")
g_Neg=grad(neg, 128, 64, T)
dev.off()

## png
##   2

hog_pos = hog(gField_pos$xgrad, gField_pos$ygrad,4,4,6)
hog_neg = hog(gField_neg$xgrad, gField_neg$ygrad,4,4,6)

hog_pos #Pos image after hog

##  [1] 0.16992188 0.12304688 0.17187500 0.16601562 0.15820312 0.21093750
##  [7] 0.21679688 0.11328125 0.17773438 0.20117188 0.11718750 0.17382812
## [13] 0.20898438 0.11914062 0.17187500 0.20703125 0.14257812 0.15039062
## [19] 0.22070312 0.06445312 0.06445312 0.48437500 0.10546875 0.06054688
## [25] 0.17968750 0.18945312 0.11718750 0.16601562 0.13476562 0.21289062
## [31] 0.14453125 0.11132812 0.11718750 0.17578125 0.14843750 0.30273438
## [37] 0.10351562 0.10937500 0.23632812 0.10937500 0.11718750 0.32421875
## [43] 0.26367188 0.15234375 0.11328125 0.20898438 0.11523438 0.14648438
## [49] 0.09960938 0.16210938 0.13085938 0.15234375 0.28125000 0.17382812
## [55] 0.12304688 0.05859375 0.17187500 0.26757812 0.14257812 0.23632812
## [61] 0.14843750 0.17382812 0.19531250 0.23828125 0.07421875 0.16992188
```

```
## [67] 0.10546875 0.30664062 0.11718750 0.15234375 0.18750000 0.13085938
## [73] 0.07226562 0.16796875 0.19726562 0.11328125 0.31250000 0.13671875
## [79] 0.10156250 0.08203125 0.15234375 0.15234375 0.30273438 0.20898438
## [85] 0.20703125 0.09765625 0.22070312 0.14062500 0.04687500 0.28710938
## [91] 0.05468750 0.22851562 0.19140625 0.23632812 0.21484375 0.07421875
```

```
hog_neg #neg image after hog
```

```
##  [1] 0.177734375 0.080078125 0.150390625 0.324218750 0.103515625
0.164062500
##  [7] 0.169921875 0.115234375 0.167968750 0.259765625 0.142578125
0.128906250
## [13] 0.001953125 0.023437500 0.000000000 0.000000000 0.021484375
0.000000000
## [19] 0.128906250 0.107421875 0.181640625 0.058593750 0.085937500
0.066406250
## [25] 0.164062500 0.136718750 0.158203125 0.220703125 0.132812500
0.187500000
## [31] 0.166015625 0.113281250 0.183593750 0.207031250 0.173828125
0.156250000
## [37] 0.031250000 0.029296875 0.039062500 0.017578125 0.035156250
0.035156250
## [43] 0.076171875 0.123046875 0.126953125 0.089843750 0.078125000
0.044921875
## [49] 0.169921875 0.171875000 0.187500000 0.175781250 0.105468750
0.189453125
## [55] 0.166015625 0.179687500 0.132812500 0.162109375 0.212890625
0.146484375
## [61] 0.097656250 0.076171875 0.074218750 0.109375000 0.097656250
0.033203125
## [67] 0.109375000 0.125000000 0.074218750 0.070312500 0.115234375
0.101562500
## [73] 0.173828125 0.191406250 0.156250000 0.150390625 0.140625000
0.187500000
## [79] 0.160156250 0.208984375 0.144531250 0.130859375 0.199218750
0.156250000
## [85] 0.121093750 0.207031250 0.121093750 0.162109375 0.263671875
0.093750000
## [91] 0.080078125 0.144531250 0.072265625 0.041015625 0.169921875
0.054687500
```

### Question 6 Part I b.)

```
features = data.frame()

dir_pos = dir("_data/pngdata/pos")
dir_neg = dir("_data/pngdata/neg")
for (i in 1:length(dir_pos)) {
  rand_pos = paste("_data/pngdata/pos/",dir_pos[i], sep="")
  rand_neg = paste("_data/pngdata/neg/",dir_neg[i], sep="")
  pos = readPNG(rand_pos)
```

```
  neg = readPNG(rand_neg)
  pos = rgb2gray(pos)
  neg = rgb2gray(neg)
  neg = crop.r(neg, 160,96)
  gField_pos = grad(pos,128,64,F)
  gField_neg = grad(neg,128,64,F)
  feature = hog(gField_pos$xgrad, gField_pos$ygrad,4,4,6)
  pos_row = cbind(rbind(feature), "POS")
  feature = hog(gField_neg$xgrad, gField_neg$ygrad,4,4,6)
  neg_row = cbind(rbind(feature), "NEG")
  features  = rbind(features, pos_row, neg_row)
}

colnames(features)[97]<- "Human"
head(features)

##                   V1          V2          V3          V4          V5
## feature  0.158203125 0.154296875   0.1171875     0.15625 0.232421875
## feature1  0.09765625   0.0546875 0.150390625   0.0546875 0.072265625
## feature2 0.216796875 0.162109375  0.15234375  0.12109375 0.130859375
## feature3    0.109375 0.271484375 0.091796875 0.087890625   0.3359375
## feature4      0.1875  0.19140625 0.099609375 0.208984375 0.166015625
## feature5  0.13671875  0.12109375  0.24609375  0.14453125 0.158203125
##                   V6          V7          V8          V9         V10
## feature  0.181640625 0.142578125 0.154296875 0.216796875  0.18359375
## feature1    0.046875  0.11328125    0.140625  0.19140625 0.271484375
## feature2 0.216796875  0.08984375 0.068359375 0.162109375  0.16015625
## feature3 0.103515625  0.14453125 0.224609375  0.12890625 0.166015625
## feature4 0.146484375   0.1328125  0.20703125  0.19921875 0.201171875
## feature5 0.193359375   0.2578125 0.060546875     0.21875 0.193359375
##                  V11         V12         V13         V14         V15
## feature   0.13671875 0.166015625 0.115234375    0.203125 0.134765625
## feature1 0.146484375   0.1328125 0.123046875 0.216796875 0.064453125
## feature2  0.26171875   0.2578125   0.2109375  0.20703125  0.12109375
## feature3    0.203125   0.1328125 0.166015625 0.212890625 0.150390625
## feature4  0.16796875 0.091796875  0.18359375  0.23828125  0.15234375
## feature5  0.11328125     0.15625    0.171875 0.095703125     0.21875
##                  V16         V17         V18         V19         V20
V21
## feature  0.162109375 0.224609375  0.16015625 0.349609375 0.158203125
0.203125
## feature1    0.109375  0.34765625 0.138671875    0.109375  0.25390625
0.1953125
## feature2 0.228515625 0.083984375   0.1484375  0.29296875 0.103515625
0.109375
## feature3 0.130859375 0.212890625 0.126953125 0.099609375  0.29296875
0.09375
## feature4   0.1171875  0.16796875    0.140625  0.26171875     0.21875
0.09765625
## feature5 0.169921875 0.123046875 0.220703125 0.208984375 0.154296875
```

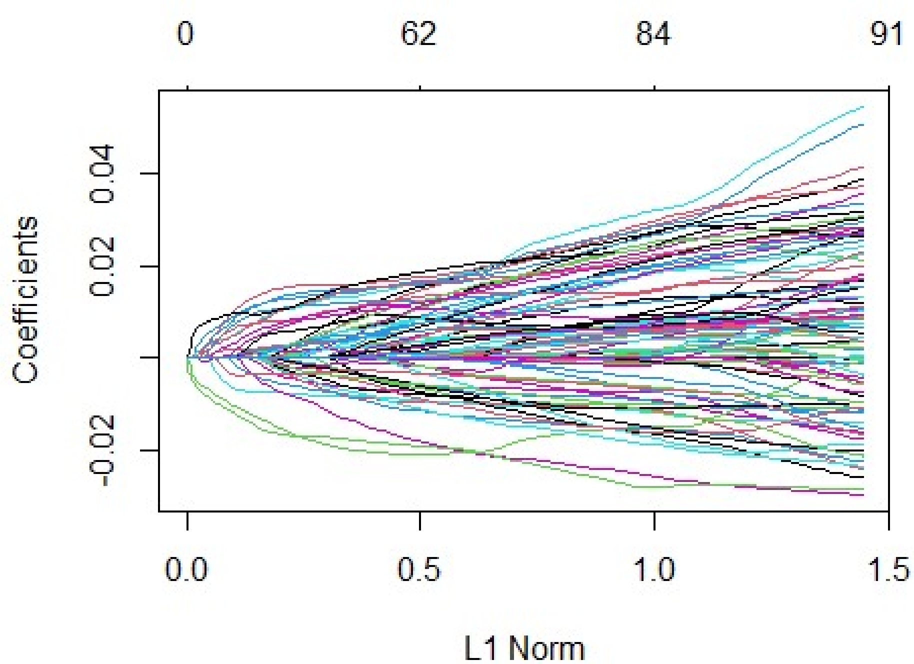
```
0.17578125
##                 V22         V23         V24         V25         V26
## feature     0.09375 0.103515625 0.091796875  0.12109375 0.169921875
## feature1 0.142578125  0.19140625 0.107421875  0.01953125 0.091796875
## feature2 0.380859375 0.064453125 0.048828125  0.21484375   0.1171875
## feature3   0.0859375  0.34765625 0.080078125 0.166015625   0.2265625
## feature4 0.095703125  0.15234375 0.173828125    0.140625 0.193359375
## feature5 0.162109375  0.14453125 0.154296875  0.19140625  0.10546875
##                 V27         V28         V29         V30         V31
## feature   0.15234375   0.1171875 0.224609375  0.21484375 0.259765625
## feature1 0.099609375   0.0703125  0.17578125 0.005859375 0.072265625
## feature2     0.15625   0.2265625  0.12890625     0.15625    0.109375
## feature3  0.09765625 0.111328125   0.2578125    0.140625 0.244140625
## feature4 0.216796875   0.1484375 0.154296875 0.146484375   0.2109375
## feature5      0.1875  0.18359375  0.10546875   0.2265625  0.24609375
##                 V32         V33         V34         V35         V36
## feature     0.15625    0.109375   0.1484375     0.09375 0.232421875
## feature1 0.123046875     0.09375    0.078125 0.236328125   0.1640625
## feature2 0.103515625 0.150390625    0.171875  0.20703125   0.2578125
## feature3    0.140625  0.14453125 0.126953125 0.142578125 0.201171875
## feature4 0.150390625 0.107421875  0.19140625   0.1796875  0.16015625
## feature5  0.07421875 0.208984375    0.203125 0.080078125      0.1875
##                 V37         V38         V39         V40         V41
## feature   0.16796875  0.11328125 0.154296875  0.16796875 0.166015625
## feature1 0.158203125 0.201171875 0.146484375  0.14453125  0.21484375
## feature2 0.201171875 0.142578125   0.1328125     0.15625  0.17578125
## feature3 0.259765625 0.111328125 0.173828125 0.146484375  0.11328125
## feature4   0.2578125 0.220703125 0.087890625  0.12109375   0.1484375
## feature5 0.205078125   0.1171875    0.171875 0.162109375  0.15234375
##                 V42         V43         V44         V45         V46
## feature   0.23046875 0.322265625   0.1640625 0.224609375  0.05859375
## feature1 0.134765625   0.1484375      0.1875  0.11328125 0.134765625
## feature2  0.19140625  0.40234375 0.064453125 0.033203125 0.314453125
## feature3   0.1953125  0.09765625 0.298828125  0.05859375 0.033203125
## feature4   0.1640625 0.240234375    0.203125 0.123046875 0.134765625
## feature5  0.19140625  0.23046875 0.197265625  0.08203125 0.044921875
##                 V47         V48         V49         V50         V51
## feature   0.08984375    0.140625 0.150390625  0.19140625    0.109375
## feature1 0.212890625 0.201171875 0.103515625  0.02734375   0.0390625
## feature2  0.12109375 0.064453125      0.1875   0.0859375 0.189453125
## feature3  0.41796875     0.09375  0.12109375 0.224609375 0.123046875
## feature4  0.14453125 0.154296875        0.25 0.146484375 0.099609375
## feature5 0.177734375 0.267578125 0.216796875  0.08984375 0.220703125
##                 V52         V53         V54         V55         V56
## feature     0.140625 0.205078125    0.203125 0.142578125   0.1171875
## feature1 0.017578125 0.083984375 0.041015625 0.138671875  0.08984375
## feature2  0.15234375 0.162109375  0.22265625    0.203125  0.13671875
## feature3 0.123046875 0.189453125     0.21875 0.158203125 0.150390625
## feature4 0.138671875     0.15625 0.208984375 0.158203125 0.130859375
## feature5   0.1953125 0.080078125 0.197265625     0.28125    0.078125
```

```
##                  V57         V58         V59         V60         V61
## feature         0.25 0.134765625      0.1875  0.16796875 0.150390625
## feature1   0.1015625 0.169921875   0.2578125 0.185546875  0.14453125
## feature2   0.1171875 0.138671875 0.181640625  0.22265625 0.208984375
## feature3  0.15234375 0.220703125 0.107421875   0.2109375 0.181640625
## feature4 0.181640625 0.130859375  0.16015625  0.23828125  0.22265625
## feature5     0.15625 0.146484375 0.107421875  0.23046875   0.2265625
##                  V62        V63         V64         V65         V66
V67
## feature   0.15234375 0.20703125 0.181640625  0.18359375       0.125
0.306640625
## feature1 0.142578125   0.140625       0.125  0.23046875 0.201171875
0.13671875
## feature2 0.146484375    0.15625 0.130859375  0.12109375 0.236328125
0.24609375
## feature3 0.099609375 0.20703125  0.16796875  0.10546875  0.23828125
0.240234375
## feature4 0.169921875 0.17578125 0.087890625 0.087890625 0.255859375
0.205078125
## feature5 0.208984375  0.1328125 0.115234375   0.1484375  0.16796875
0.19140625
##                  V68         V69         V70         V71         V72
## feature    0.2109375   0.2421875   0.0859375  0.08203125 0.072265625
## feature1 0.248046875 0.138671875 0.166015625 0.197265625    0.109375
## feature2 0.103515625  0.11328125       0.125 0.189453125  0.22265625
## feature3    0.109375  0.15234375  0.19921875  0.11328125 0.185546875
## feature4     0.15625  0.19921875  0.19921875 0.119140625  0.12109375
## feature5  0.11328125     0.21875  0.19140625   0.1328125  0.15234375
##                  V73         V74         V75         V76         V77
## feature  0.193359375  0.22265625  0.13671875  0.19921875  0.16015625
## feature1  0.07421875 0.076171875 0.103515625      0.0625      0.1875
## feature2  0.16015625 0.185546875 0.162109375 0.193359375 0.150390625
## feature3   0.1484375 0.134765625 0.123046875 0.119140625  0.22265625
## feature4 0.111328125  0.10546875   0.1953125  0.24609375   0.1953125
## feature5  0.31640625 0.033203125   0.1953125 0.244140625 0.037109375
##                  V78         V79         V80         V81         V82
## feature  0.087890625 0.119140625     0.15625 0.232421875  0.23828125
## feature1 0.154296875 0.162109375 0.126953125 0.103515625 0.130859375
## feature2   0.1484375 0.166015625 0.158203125 0.123046875 0.103515625
## feature3        0.25 0.201171875     0.15625  0.08984375 0.123046875
## feature4 0.146484375  0.08203125 0.099609375 0.396484375 0.232421875
## feature5 0.173828125 0.244140625 0.083984375 0.197265625    0.265625
##                  V83         V84         V85         V86         V87
## feature  0.169921875 0.083984375 0.119140625 0.189453125   0.1171875
## feature1 0.236328125  0.23828125   0.1953125    0.140625 0.091796875
## feature2  0.25390625   0.1953125 0.173828125 0.150390625 0.158203125
## feature3 0.224609375 0.201171875 0.181640625 0.123046875 0.119140625
## feature4 0.111328125    0.078125 0.103515625       0.125   0.2890625
## feature5 0.044921875   0.1640625  0.12890625   0.1484375   0.2109375
##                  V88         V89         V90         V91         V92
```

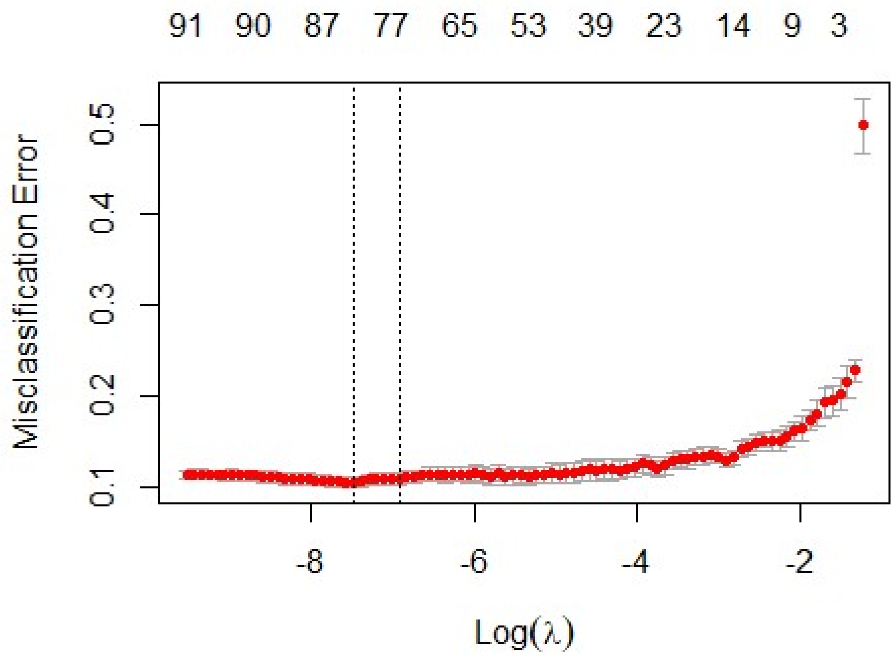
```
## feature  0.146484375 0.291015625  0.13671875 0.267578125   0.1484375
## feature1 0.134765625        0.25  0.18359375  0.15234375   0.2578125
## feature2 0.185546875 0.091796875 0.240234375   0.1328125 0.189453125
## feature3    0.203125    0.203125   0.1640625 0.197265625  0.17578125
## feature4    0.171875 0.103515625  0.20703125 0.212890625     0.15625
## feature5 0.177734375  0.17578125 0.158203125  0.22265625 0.111328125
##                  V93         V94         V95         V96 Human
## feature  0.283203125 0.115234375  0.08984375 0.095703125   POS
## feature1 0.099609375 0.080078125    0.234375 0.169921875   NEG
## feature2  0.08984375 0.150390625  0.30078125  0.13671875   POS
## feature3   0.1015625   0.1328125  0.20703125 0.185546875   NEG
## feature4  0.16796875 0.146484375 0.134765625 0.181640625   POS
## feature5 0.220703125  0.17578125 0.123046875 0.146484375   NEG

#length(dir_pos)
#change it so the rows have a good name
```

### Question 6 Part II.)

```
glm_png = glmnet(x=data.matrix(features[1:length(features)-1]),
y=features$Human, family = "binomial")

plot(glm_png)
```




```
cv_png = cv.glmnet(x=data.matrix(features[1:length(features)-1]),
y=features$Human, family = "binomial", type.measure="class")

plot(cv_png)
```




**Question 7 a.)**

```
load("_data/Amazon_SML.RData")
```

Column names are name, review and rating

```
colnames(dat)

## [1] "name"   "review" "rating"
```

there are 1312 reviews

```
nrow(dat[is.null(dat$review)])

## [1] 1312
```

There are 20 unique products

```
length(unique(dat$name))

## [1] 20

library(dplyr)
```

Vulli Sophie the Giraffe Teether has the most 5 ratings at 526 5 star ratings

```
dat %>%
  group_by(name) %>%
  filter(rating ==5) %>%
```

```
  summarise(n = n()) %>%
  filter(n == max(n))

## # A tibble: 1 × 2
##   name                                n
##   <fct>                           <int>
## 1 Vulli Sophie the Giraffe Teether   526
```

Most 1 star ratings

```
dat %>%
  group_by(name) %>%
  filter(rating ==1) %>%
  summarise(n = n()) %>%
  filter(n == max(n))

## # A tibble: 1 × 2
##   name
n
##   <fct>
<int>
## 1 Infant Optics DXR-5 2.4 GHz Digital Video Baby Monitor with Night Vision
68
```

### Question 7b.)

Amount for each rating

```
dat %>%
  group_by(rating) %>%
  summarise(n = n())

## # A tibble: 2 × 2
##   rating     n
##    <int> <int>
## 1      1   656
## 2      5   656
```

The best performance of a constant classifier is 1/2

```
source("_data/tdMat.R")

## Loading required package: NLP

##
## Attaching package: 'NLP'

## The following object is masked from 'package:ggplot2':
##
##     annotate
```

### Question 7c.)

```
source("_data/splitData.R")
```

Below are the amount of covariates with non-zero coefficients, 20 most negative words and 20 most positive words

```
set.seed(10)
lambda<-exp(seq(-20, -1, length.out = 99))
cvfit<-
cv.glmnet(train.x,train.y,family="binomial",type.measure="class",lambda=lambd
a)

lambda1se = cvfit$lambda.1se

glmfit = glmnet(x=train.x, y=train.y, lambda = lambda1se,family =
"binomial",type.measure="class")

cft = coef(glmfit, s=lambda1se)
glmfit$df # amount of covariates with non-zero coefficients

## [1] 353

cft[order(cft[,1])[1:20],0] #20 most negative words

## 20 x 0 sparse Matrix of class "dgCMatrix"
##
## swallow
## downstair
## tummi
## solv
## dissapoint
## unlink
## avoid
## philip
## bin
## wast
## useless
## click
## knock
## sad
## massiv
## scissor
## cool
## speaker
## return
## ball

cft[order(-cft[,1])[1:20],0] #20 most positive words

## 20 x 0 sparse Matrix of class "dgCMatrix"
##
## wimper
## round
## endur
```

```
## abov
## scrape
## whichev
## love
## lol
## neighborhood
## precious
## laundri
## fyi
## teeth
## poster
## grandma
## channel
## sum
## bet
## describ
## result
```

### Question 7 d.)

These two words are (from most negaive to most positive)

```
cft = cbind(coef(glmfit, s=lambda1se))
row.names(cft)[order(cft[,1])[1]]

## [1] "swallow"

row.names(cft)[order(-cft[,1])[1]]

## [1] "wimper"

head(train.tag)

##  491  368  439  344 1295  143
##  491  368  439  344 1295  143
```

missclassification rate is 0.1515

```
set.seed(10)
lambda_t<-exp(seq(-20, -1, length.out = 99))
cvfit_tst<-
cv.glmnet(test.x,test.y,family="binomial",type.measure="class",lambda=lambda_
t)

lambda1se_tst = cvfit_tst$lambda.1se

glmfit_tst = glmnet(x=test.x, y=test.y, lambda = lambda1se_tst,family =
"binomial",type.measure="class")

cvfit_tst
```

```
##
## Call:  cv.glmnet(x = test.x, y = test.y, lambda = lambda_t, type.measure =
"class",      family = "binomial")
##
## Measure: Misclassification Error
##
##       Lambda Index Measure      SE Nonzero
## min 0.00762    21  0.1288 0.02540      71
## 1se 0.04360    12  0.1515 0.02281      44
```

It is better than the misclassifiction error before.

### Question 8 a.)

```r
heart = read.csv("_data/framingham.csv")
head(heart)
```

```
##   male age education currentSmoker cigsPerDay BPMeds prevalentStroke
## 1    1  39         4             0          0      0               0
## 2    0  46         2             0          0      0               0
## 3    1  48         1             1         20      0               0
## 4    0  61         3             1         30      0               0
## 5    0  46         3             1         23      0               0
## 6    0  43         2             0          0      0               0
##   prevalentHyp diabetes totChol sysBP diaBP   BMI heartRate glucose
TenYearCHD
## 1            0        0     195 106.0    70 26.97        80      77
0
## 2            0        0     250 121.0    81 28.73        95      76
0
## 3            0        0     245 127.5    80 25.34        75      70
0
## 4            1        0     225 150.0    95 28.58        65     103
1
## 5            0        0     285 130.0    84 23.10        85      85
0
## 6            1        0     228 180.0   110 30.30        77      99
0
```

From summary the significant variables are age, cigsPerDay, sysBP and glucose

```r
heart_fit = glm(TenYearCHD ~ ., data = heart, family = "binomial")
summary(heart_fit)
```

```
##
## Call:
## glm(formula = TenYearCHD ~ ., family = "binomial", data = heart)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9582  -0.5939  -0.4264  -0.2829   2.8409
##
```

```
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)    
## (Intercept)     -8.328186   0.715449 -11.641  < 2e-16 ***
## male             0.555279   0.109033   5.093 3.53e-07 ***
## age              0.063515   0.006679   9.509  < 2e-16 ***
## education       -0.047767   0.049395  -0.967  0.33353    
## currentSmoker    0.071601   0.156752   0.457  0.64783    
## cigsPerDay       0.017914   0.006238   2.872  0.00408 ** 
## BPMeds           0.162496   0.234326   0.693  0.48802    
## prevalentStroke  0.693660   0.489569   1.417  0.15652    
## prevalentHyp     0.234208   0.138026   1.697  0.08973 .  
## diabetes         0.039167   0.315506   0.124  0.90120    
## totChol          0.002332   0.001127   2.070  0.03850 *  
## sysBP            0.015403   0.003808   4.044 5.24e-05 ***
## diaBP           -0.004159   0.006438  -0.646  0.51831    
## BMI              0.006672   0.012758   0.523  0.60097    
## heartRate       -0.003246   0.004211  -0.771  0.44082    
## glucose          0.007127   0.002234   3.190  0.00142 ** 
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 3121.2  on 3657  degrees of freedom
## Residual deviance: 2754.5  on 3642  degrees of freedom
##   (582 observations deleted due to missingness)
## AIC: 2786.5
## 
## Number of Fisher Scoring iterations: 5
```

Question 7 Part 2

```
set.seed(100)
size = nrow(heart)/5
smple = sample(nrow(heart),size = size, replace = FALSE)
heart_test = heart[smple,]
heart_train = heart[-smple,]
head(heart_test)
```

```
##      male age education currentSmoker cigsPerDay BPMeds prevalentStroke
## 3786    0  38         1             0          0      0               0
## 503     0  64         1             0          0      0               0
## 3430    0  51         1             0          0      0               0
## 3696    1  43         3             1         20      0               0
## 4090    0  64         1             0          0      0               0
## 3052    0  50         3             1          9      0               0
##      prevalentHyp diabetes totChol sysBP diaBP   BMI heartRate glucose
## 3786            0        0     214 115.0    90 25.69        80      65
## 503             0        0     251 132.0    82 28.87        70      82
## 3430            1        0     230 168.5    97 26.36        57      77
```

```
## 3696              0        0      240 147.5    88 25.60        65      113
## 4090              1        0      232 149.5    84 20.49        68       96
## 3052              0        1      210 134.0    80 18.26        64       NA
##      TenYearCHD
## 3786          0
## 503           0
## 3430          0
## 3696          0
## 4090          0
## 3052          0
```

```
head(heart_train)
```

```
##   male age education currentSmoker cigsPerDay BPMeds prevalentStroke
## 1    1  39         4             0          0      0               0
## 3    1  48         1             1         20      0               0
## 5    0  46         3             1         23      0               0
## 6    0  43         2             0          0      0               0
## 7    0  63         1             0          0      0               0
## 8    0  45         2             1         20      0               0
##   prevalentHyp diabetes totChol sysBP diaBP   BMI heartRate glucose
TenYearCHD
## 1            0        0     195 106.0    70 26.97        80      77
0
## 3            0        0     245 127.5    80 25.34        75      70
0
## 5            0        0     285 130.0    84 23.10        85      85
0
## 6            1        0     228 180.0   110 30.30        77      99
0
## 7            0        0     205 138.0    71 33.11        60      85
1
## 8            0        0     313 100.0    71 21.68        79      78
0
```

```
fit_h_train = glm(TenYearCHD ~ ., data = heart_train, family = "binomial")
probs = fit_h_train %>% predict(heart_test, type = "response")
pred = na.omit(ifelse(probs > 0.5, 1, 0))
1-mean(pred == heart_test$TenYearCHD) #misclassification error
```

```
## Warning in pred == heart_test$TenYearCHD: longer object length is not a
## multiple of shorter object length

## [1] 0.1627358
```

The curve is sort of a U curve however it seems there is plateau of the misclassification error at lambda > ~ -3.5 so we do not need to use regularization

```
heart = na.omit(heart)
heart_glm =  glmnet(x = data.matrix(heart[1: ncol(heart)-1]),y
```

```
=heart$TenYearCHD, family = binomial, lambda = 1)
heart_glm

##
## Call:  glmnet(x = data.matrix(heart[1:ncol(heart) - 1]), y =
heart$TenYearCHD,      family = binomial, lambda = 1)
##
##   Df %Dev Lambda
## 1  0    0      1

heart_cv = cv.glmnet(x=data.matrix(heart[1: ncol(heart)-1]),
y=heart$TenYearCHD, nfolds = 5, lamda = 1, type.measure="class", family =
"binomial")
plot(heart_cv)
```