NUEN 647
Uncertainty Quantification for Nuclear Engineering
Homework 3

Due on Saturday, December 10, 2016

Dr. McClarren

Paul Mendoza

Contents

Problem 1

Fit the data in Table 1 to a linear model using

(a) Least squares

(b) Ridge Regression

(c) Lasso Regression

Table 1: Data to fit linear model $y = a + bx_1 + cx_2$

	x_1	x_2	y
1	0.99	0.98	6.42
2	-0.75	-0.76	0.20
3	-0.50	-0.48	0.80
4	-1.08	-1.08	-0.57
5	0.09	0.09	4.75
6	-1.28	-1.27	-1.42
7	-0.79	-0.79	1.07
8	-1.17	-1.17	0.20
9	-0.57	-0.57	1.08
10	-1.62	-1.62	-0.15
11	0.34	0.35	2.90
12	0.51	0.51	3.37
13	-0.91	-0.92	0.05
14	1.85	1.86	5.50
15	-1.12	-1.12	0.17
16	-0.70	-0.70	1.72
17	1.19	1.18	3.97
18	1.24	1.23	6.38
19	-0.52	-0.52	3.29
20	-1.41	-1.41	-1.49

Be sure to do cross-validation for each fit, and for each method present your best estimate of the model.

Did this problem in R, and appended the PDF on the following pages.

Problem 1 continued on next page. . .

Problem1

Paul Mendoza

December 9, 2016

```
require(magrittr)
require(dplyr)
require(ggplot2)
require(glmnet)
```

Fit the data in Table 1 to a linear model using:

```
Table1
```

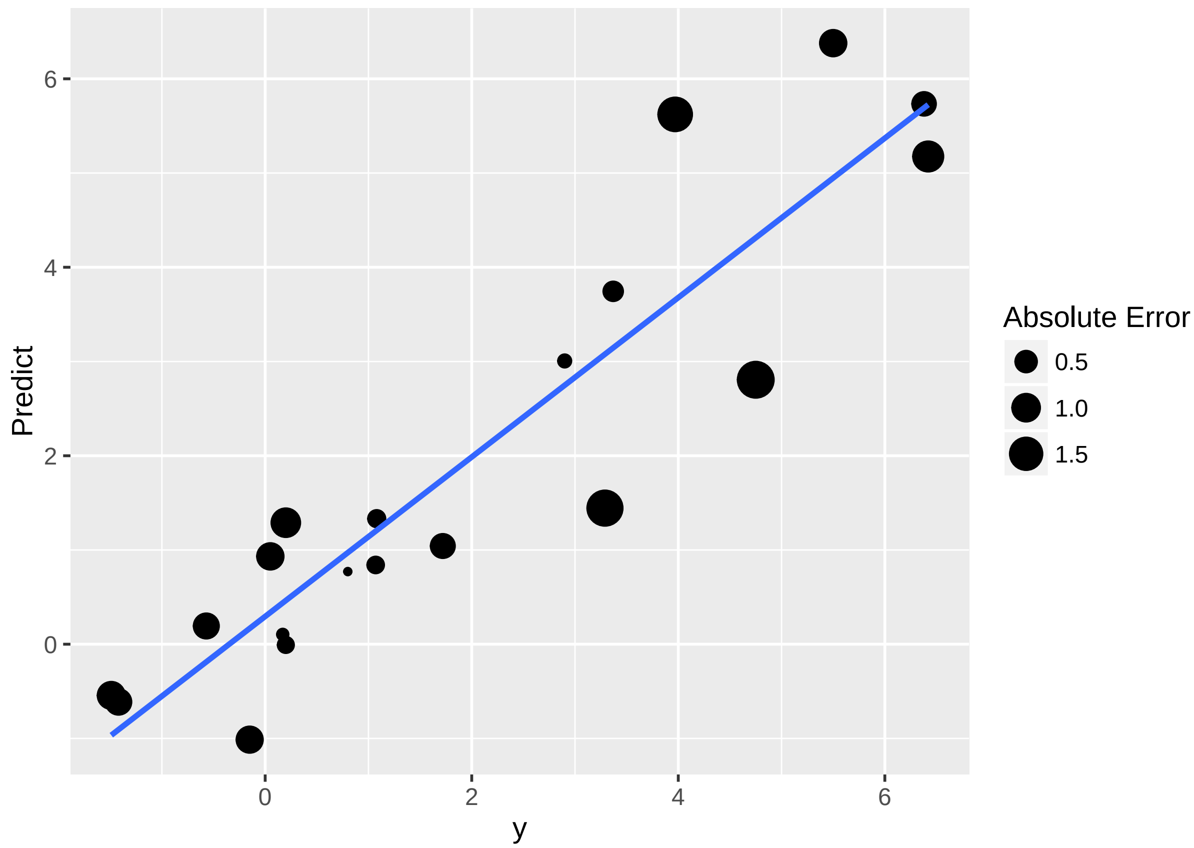
```
##       x1    x2     y
## 1   0.99  0.98  6.42
## 2  -0.75 -0.76  0.20
## 3  -0.50 -0.48  0.80
## 4  -1.08 -1.08 -0.57
## 5   0.09  0.09  4.75
## 6  -1.28 -1.27 -1.42
## 7  -0.79 -0.79  1.07
## 8  -1.17 -1.17  0.20
## 9  -0.57 -0.57  1.08
## 10 -1.62 -1.62 -0.15
## 11  0.34  0.35  2.90
## 12  0.51  0.51  3.37
## 13 -0.91 -0.92  0.05
## 14  1.85  1.86  5.50
## 15 -1.12 -1.12  0.17
## 16 -0.70 -0.70  1.72
## 17  1.19  1.18  3.97
## 18  1.24  1.23  6.38
## 19 -0.52 -0.52  3.29
## 20 -1.41 -1.41 -1.49
```

1. Least Squares

```
LeastFit<-lm(formula = y~x1+x2,data=Table1)
LeastFit
```

```
##
## Call:
## lm(formula = y ~ x1 + x2, data = Table1)
##
## Coefficients:
## (Intercept)           x1           x2
##       2.606       38.133      -35.900
```

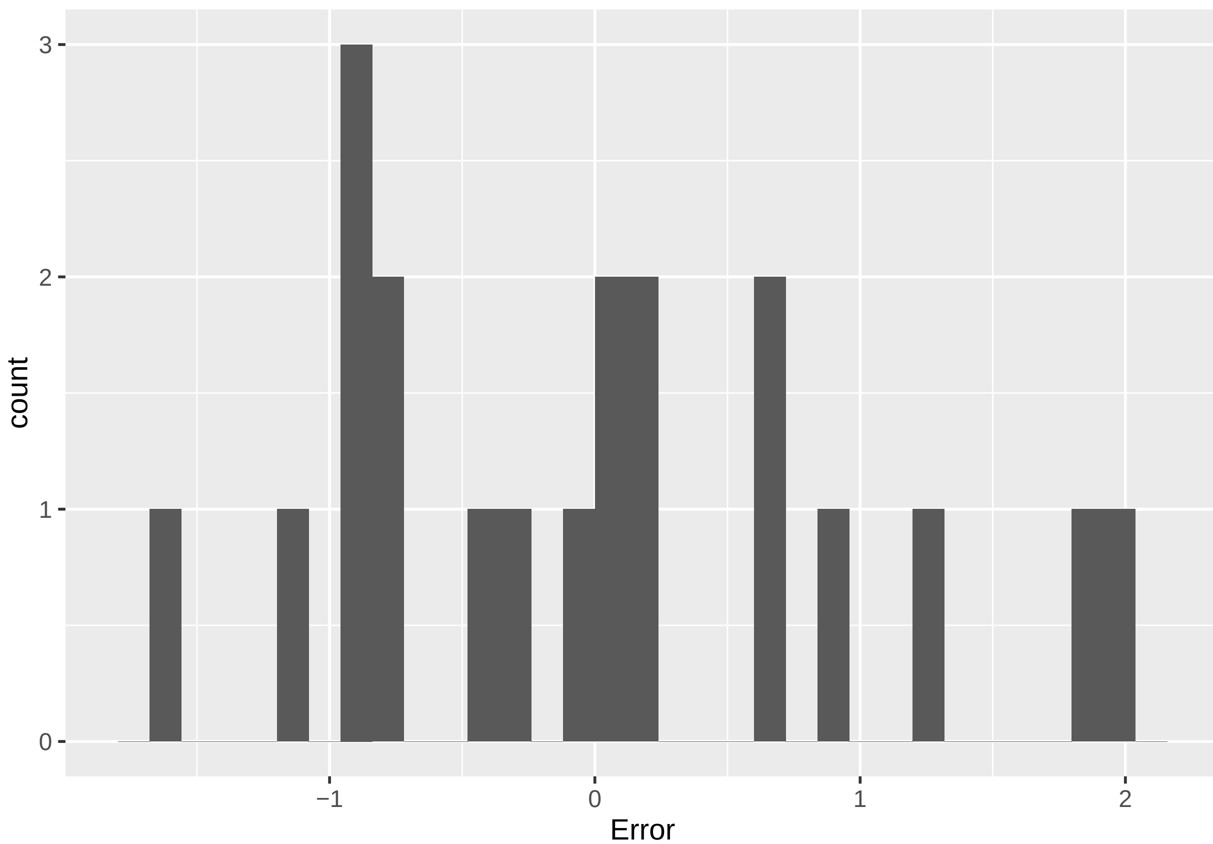
```
plotDF<-Table1
plotDF[,'Type']<-'Train'
plotDF$Predict<-predict(LeastFit,plotDF[,1:2])
plotDF$Error<-plotDF$y-plotDF$Predict
ggplot(plotDF,aes(x=y,y=Predict,size=abs(Error)))+geom_point()+
  scale_size("Absolute Error")+geom_smooth(method="lm",se=F,size=1)
```




```
sqrt(var(data.frame(plotDF %>% filter(Type=="Train") %>% select(Error))))/20
```

```
##              Error
## Error 0.04902635
```

```
ggplot(plotDF,aes(x=Error)) + geom_histogram()
```

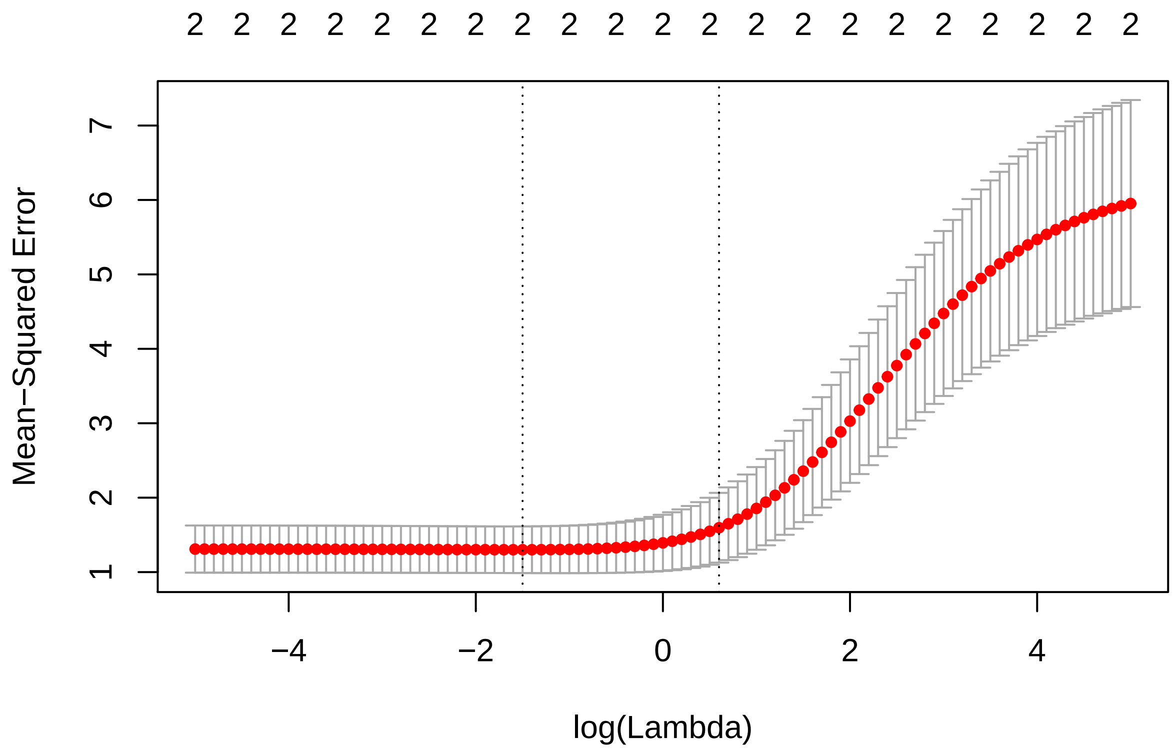



```
sensitivities <- coef(LeastFit)
sensDF <- data.frame(Method = 0, Var = 0, Value=0)
sensDF[1:length(sensitivities),'Method'] <- "Least-Squares"
sensDF[1:length(sensitivities),'Var'] <- names(sensitivities)
sensDF[1:length(sensitivities),'Value'] <- (sensitivities)
rowStart <- length(sensitivities)+1
```

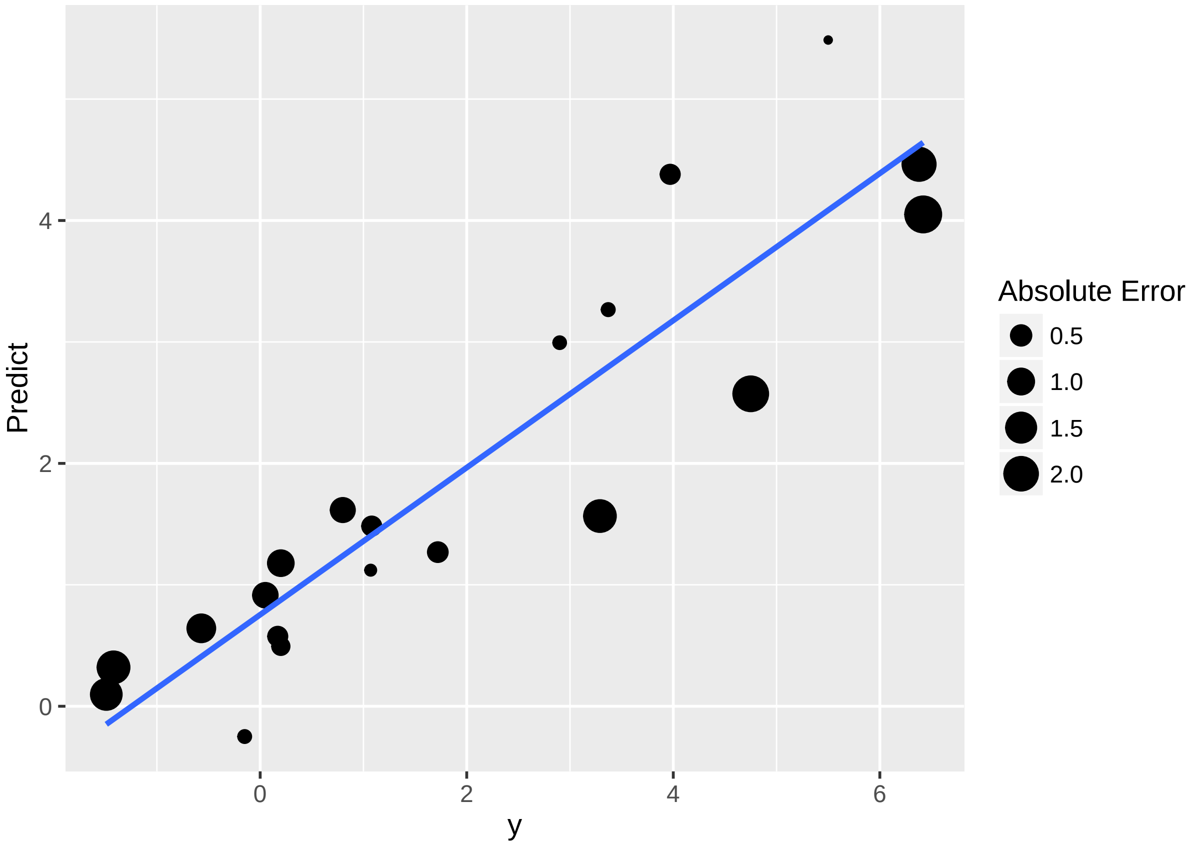
2. Ridge Regression

Ridge regression sets alpha=0, which adds damping to the coefficients

```
crossValid <- cv.glmnet(as.matrix(Table1[,1:2]),
                        as.matrix(Table1$y),alpha = 0,
                        lambda=exp(seq(-5,5,by=0.1)))

plot(crossValid)
```



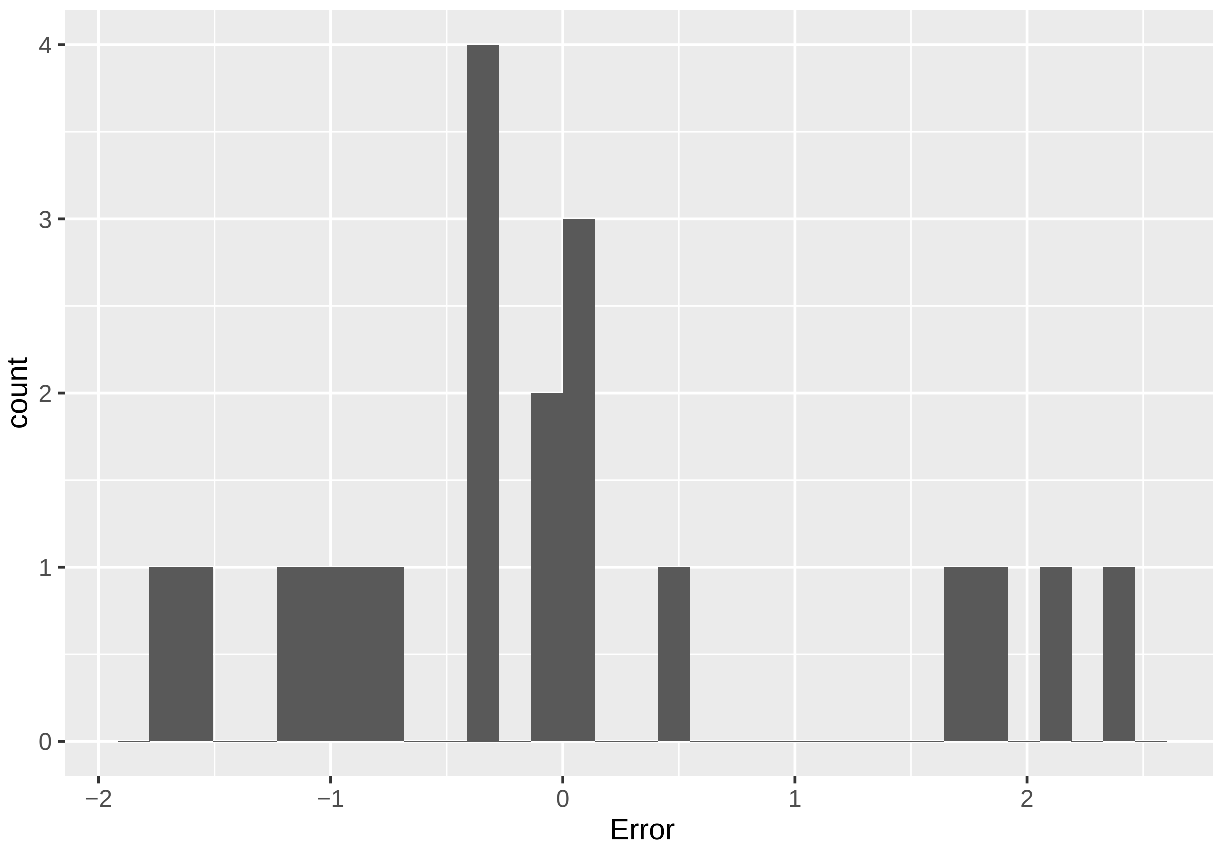

```
lambda <- crossValid$lambda.min
sensitivities <- coef(crossValid)
plotDF <- Table1
plotDF[,'Type'] <- 'Train'
plotDF[,"Predict" ]<- data.frame(Predict=predict(crossValid,as.matrix(plotDF[,1:2]),
                                                  lambda=lambda))
plotDF$Error <- plotDF$y-plotDF$Predict
ggplot(plotDF,aes(x=y,y=Predict,size=abs(Error))) + geom_point() +
scale_size("Absolute Error") + geom_smooth(method="lm",se=F,size=1)
```




```r
sqrt(var(data.frame(plotDF %>% filter(Type=="Train") %>% select(Error))))/20
```

```
##             Error
## Error 0.05982956
```

```r
ggplot(plotDF,aes(x=Error)) + geom_histogram()
```

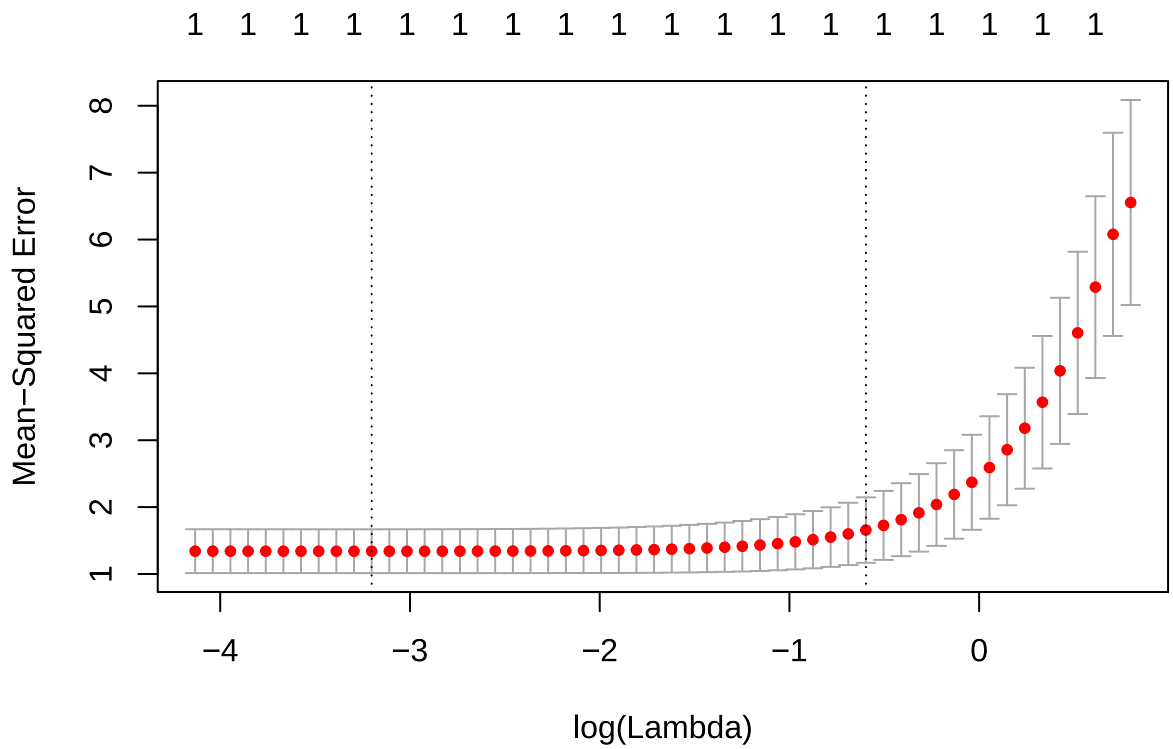



```
sensDF[rowStart:(rowStart + length(sensitivities)-1),'Method'] <- "Ridge"
sensDF[rowStart:(rowStart+length(sensitivities)-1),'Var']<-t(t(rownames(sensitivities)))
sensDF[rowStart:(rowStart +length(sensitivities)-1),'Value']<-as.numeric(sensitivities)
rowStart <- rowStart + length(sensitivities)
```

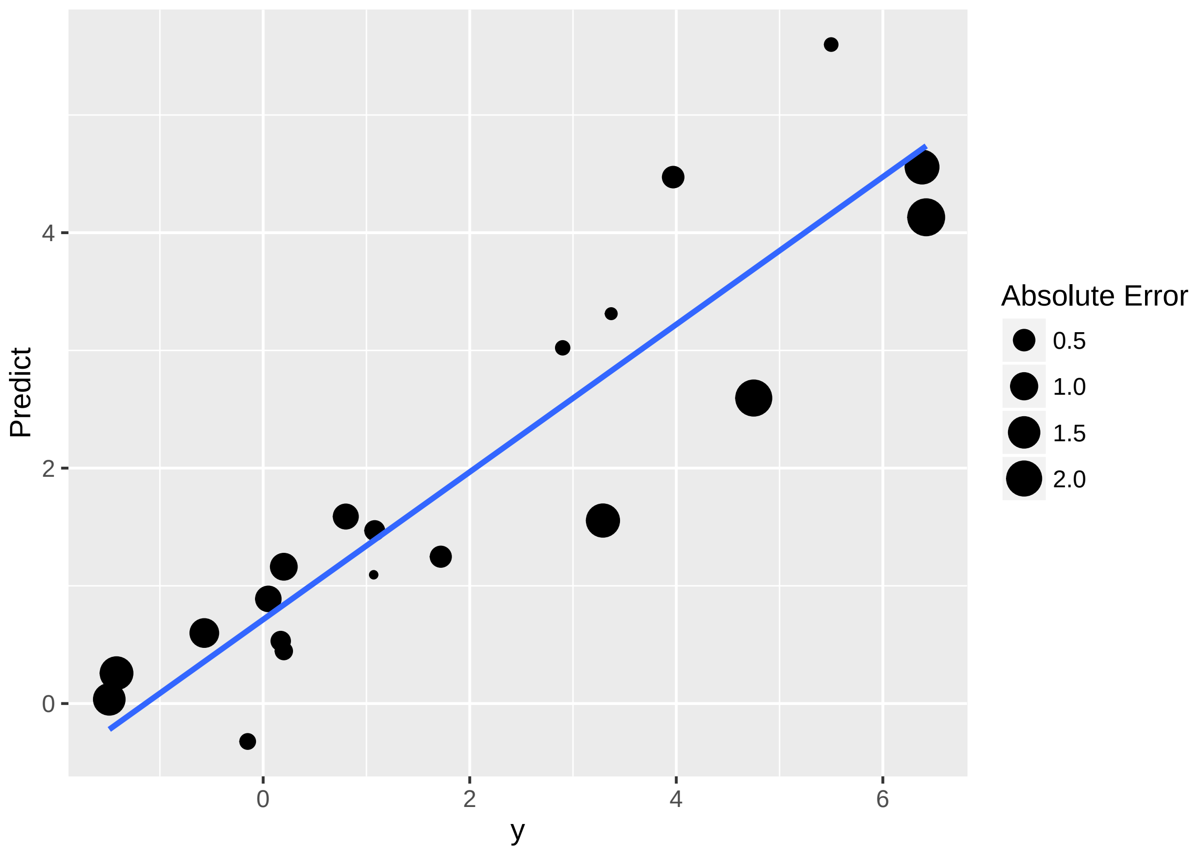
3. Lasso Regression

Lasso regression sets alpha=1

```
crossValid <- cv.glmnet(as.matrix(Table1[,1:2]),as.matrix(Table1$y),alpha = 1)
plot(crossValid)
```



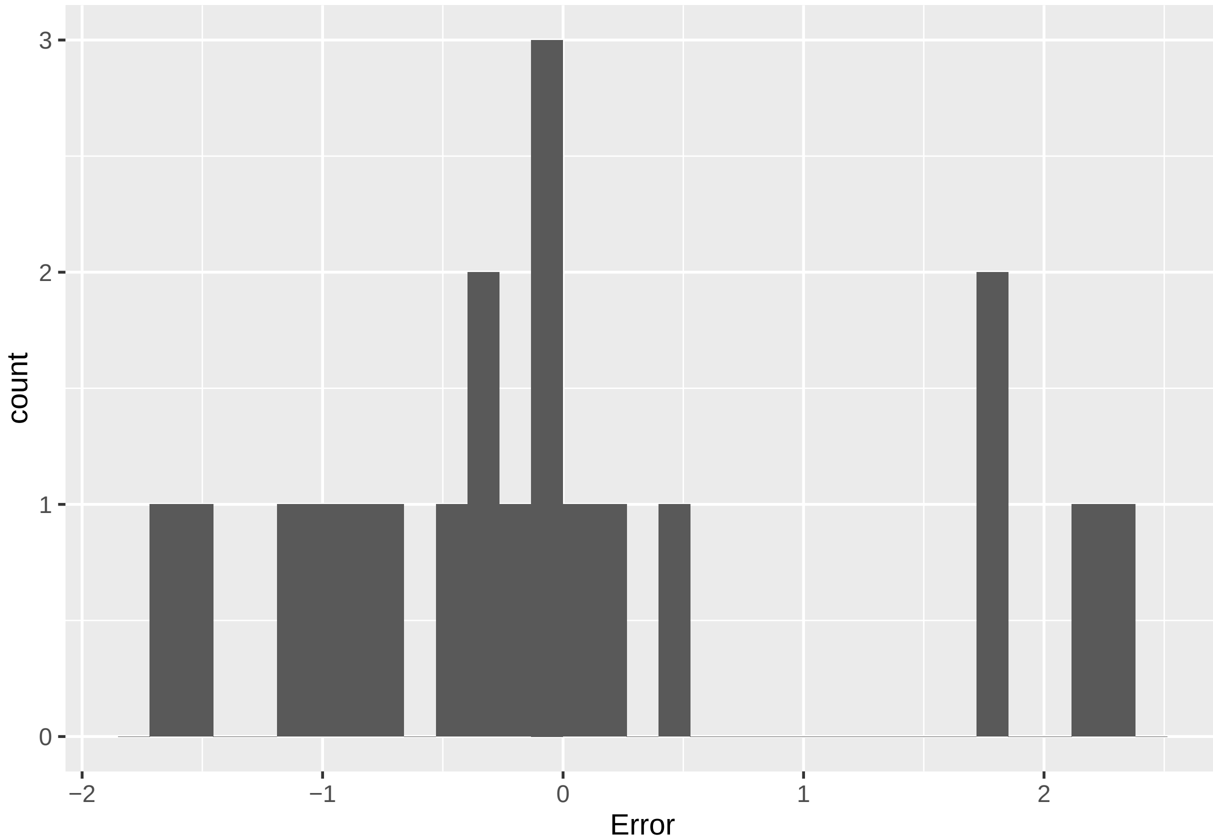

```
lambda <- crossValid$lambda.min
sensitivities <- coef(crossValid)
plotDF <- Table1
plotDF[,'Type'] <- 'Train'
plotDF[,"Predict" ]<- data.frame(Predict=predict(crossValid,as.matrix(Table1[,1:2]),
                                                 lambda=lambda))

plotDF$Error <- plotDF$y-plotDF$Predict
ggplot(plotDF,aes(x=y,y=Predict,size=abs(Error))) + geom_point() +
scale_size("Absolute Error") + geom_smooth(method="lm",se=F,size=1)
```




```
sqrt(var(data.frame(plotDF %>% filter(Type=="Train") %>% select(Error))))/20
```

```
##             Error
## Error 0.05832321
```

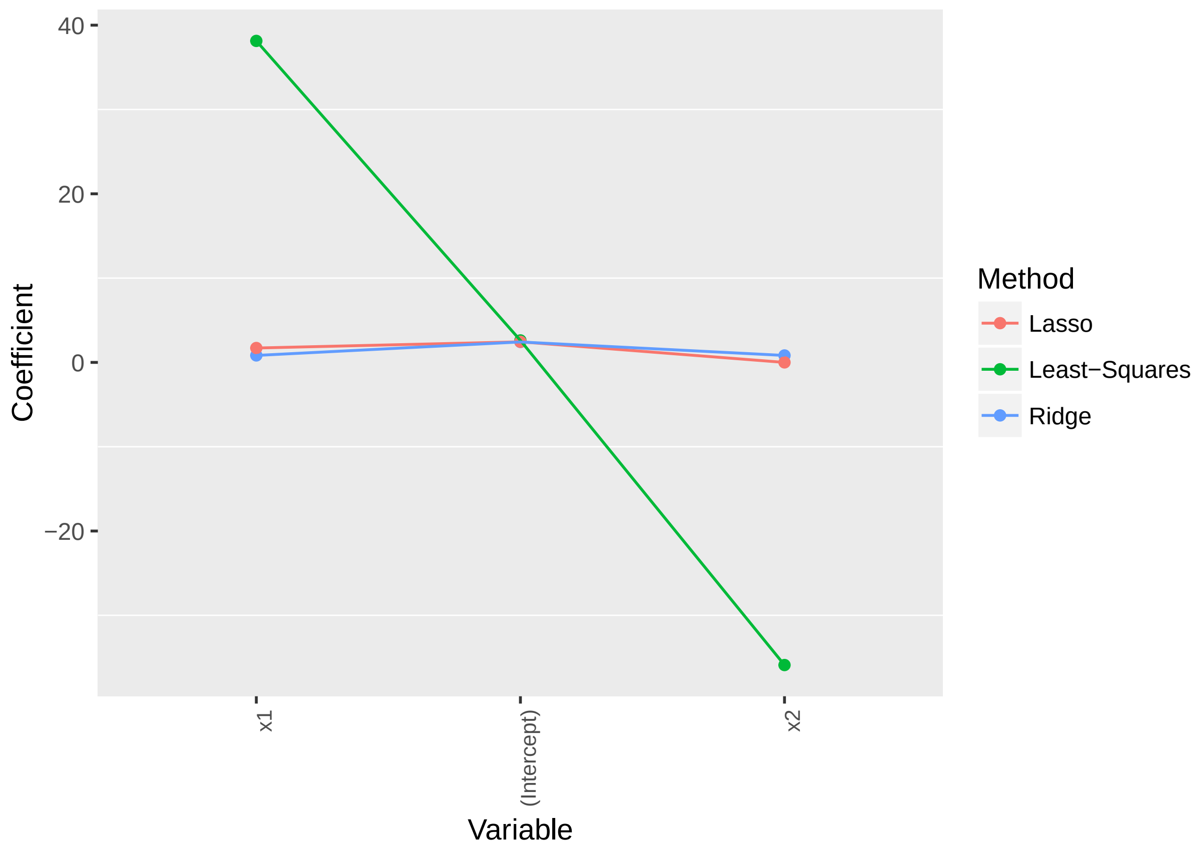
```
ggplot(plotDF,aes(x=Error)) + geom_histogram()
```




```
sensDF[rowStart:(rowStart + length(sensitivities)-1),'Method'] <- "Lasso"
sensDF[rowStart:(rowStart+length(sensitivities)-1),'Var']<-t(t(rownames(sensitivities)))
sensDF[rowStart:(rowStart +length(sensitivities)-1),'Value']<-as.numeric(sensitivities)
rowStart <- rowStart + length(sensitivities)
```

Compare Methods

```
ggplot(sensDF,aes(x=reorder(Var,-Value),y=Value,color=Method,group=Method)) +
  geom_point() + geom_line() +
  theme(panel.grid.major = element_blank(),
        axis.text.x = element_text(angle = 90, hjust = 1, size=8))+
  scale_x_discrete("Variable") + scale_y_continuous("Coefficient")
```



The Lasso and Ridge are both more bounded in their coefficents.

Problem 2

Derive the adjoint operator for the equation

$$-\bigtriangledown^2 \phi(x,y,z) + \frac{1}{L^2}\phi(x,y,z) = \frac{Q}{D}$$

$$\phi(0,y,z) = \phi(x,0,z) = \phi(x,y,0) = \phi(X,y,z) = \phi(x,Y,z) = \phi(x,y,Z) = C$$

Compute the sensitivity to the QOI:

$$QoI = \int_0^X dx \int_0^Y dy \int_0^Z dz \frac{D}{L^2}\phi(x,y,z)$$

for X,Y,Z,L,, and Q.

Derive the adjoint operator

Define the operator $\mathcal{L}$ as

$$\mathcal{L} = \bigtriangledown^2 + \frac{1}{L^2}$$

and the adjoint $\mathcal{L}^\dagger$ as

$$\mathcal{L}^\dagger = \bigtriangledown^2 + \frac{1}{L^2}$$

$$\phi^\dagger(0,y,z) = \phi^\dagger(x,0,z) = \phi^\dagger(x,y,0) = \phi^\dagger(X,y,z) = \phi^\dagger(x,Y,z) = \phi^\dagger(x,y,Z) = C$$

Setting:

$$\left|\frac{\delta\phi^\dagger}{\delta x}\right|_{x=0} = \left|\frac{\delta\phi}{\delta x}\right|_{x=0}$$

and

$$\left|\frac{\delta\phi^\dagger}{\delta x}\right|_{x=X} = \left|\frac{\delta\phi}{\delta x}\right|_{x=X}$$

and similar for the other two dimentions. Also define the inner product as:

$$(u,v) = \int_0^X dx \int_0^Y dy \int_0^Z dz\ uv$$

Proof, in order to prove that this is an adjoint operator for the above equation, it needs to be shown that $(\mathcal{L}\phi, \phi^\dagger) = (\phi, \mathcal{L}^\dagger\phi^\dagger)$.

Equivalent to:

$$\int_0^X dx \int_0^Y dy \int_0^Z dz \left(\phi^\dagger \bigtriangledown^2 \phi + \phi^\dagger \frac{\phi}{L^2}\right) = \int_0^X dx \int_0^Y dy \int_0^Z dz \left(\phi \bigtriangledown^2 \phi^\dagger + \phi\frac{\phi^\dagger}{L^2}\right) \tag{1}$$

The terms

$$\int_0^X dx \int_0^Y dy \int_0^Z dz \left(\phi^\dagger \frac{\phi}{L^2}\right) = \int_0^X dx \int_0^Y dy \int_0^Z dz \left(\phi\frac{\phi^\dagger}{L^2}\right)$$

are equal. For the other term with, $\bigtriangledown^2$, we can expand to:

$$\int_0^X dx \int_0^Y dy \int_0^Z dz \left(\phi^\dagger \left[\frac{\delta^2\phi}{\delta x^2} + \frac{\delta^2\phi}{\delta y^2} + \frac{\delta^2\phi}{\delta z^2}\right]\right)$$

Focusing on the x terms, noting that y and z will have the same derivation. Integration by parts, with $u = \phi^\dagger$, $du = \frac{\delta \phi^\dagger}{\delta x} dx$, and $v = \frac{\delta \phi}{dx}$, $dv = \frac{\delta^2 \phi}{\delta x^2} dx$ yields.

$$\int_0^Y dy \int_0^Z dz \left(\int_0^X dx \; \phi^\dagger \frac{\delta^2 \phi}{\delta x^2} \right) = \int_0^Y dy \int_0^Z dz \left(\left| \phi^\dagger \frac{\delta \phi}{\delta x} \right|_{x=0}^{x=X} - \int_0^X dx \; \frac{\delta \phi}{\delta x} \frac{\delta \phi^\dagger}{\delta x} dx \right)$$

Performing another integration by parts, with $u = \frac{\delta \phi^\dagger}{\delta x}$, $du = \frac{\delta^2 \phi^d ag}{\delta x} dx$ and, $v = \phi$, $dv = \frac{\delta \phi}{\delta x} dx$.

$$= \int_0^Y dy \int_0^Z dz \left(\left| \phi^\dagger \frac{\delta \phi}{\delta x} \right|_{x=0}^{x=X} - \left| \phi \frac{\delta \phi^\dagger}{\delta x} \right|_{x=0}^{x=X} + \int_0^X dx \; \phi \frac{\delta^2 \phi^\dagger}{\delta x^2} dx \right)$$

At the boundaries, both ϕ and $\phi^\dagger$ are a constant, and the derivatives of both at the boundaries are equal, and therefore those terms cancel, leaving

$$\int_0^Y dy \int_0^Z dz \left(\int_0^X dx \; \phi \frac{\delta^2 \phi^\dagger}{\delta x^2} dx \right)$$

which is equal to the x component of the $\bigtriangledown^2$ term of the RHS of equation 1 above.

Compute the sensitivity to the QoI:

I don't know if this is the right way to go about this, but if you modify the first equation listed in this problem to:

$$\phi = L^2 \left[\frac{Q}{D} + \bigtriangledown^2 \phi \right]$$

and use it as a substitution, then we get something like this,

$$\begin{aligned} \text{QoI} &= \int_0^X dx \int_0^Y dy \int_0^Z dz \; \frac{D}{L^2} \phi \\ &= \int_0^X dx \int_0^Y dy \int_0^Z dz \; \frac{D}{L^2} \left(L^2 \left[\frac{Q}{D} + \bigtriangledown^2 \phi \right] \right) \\ &= \int_0^X dx \int_0^Y dy \int_0^Z dz \; \left[Q + D \bigtriangledown^2 \phi \right] \end{aligned}$$

If Q and D both are independent of space, then

$$\text{QoI} = QXYZ + D \left[\int_0^Y dy \int_0^Z dz \; \left| \frac{\delta \phi}{\delta x} \right|_{x=0}^{x=X} + \int_0^X dx \int_0^Z dz \; \left| \frac{\delta \phi}{\delta y} \right|_{y=0}^{y=Y} + \int_0^X dx \int_0^Z dz \; \left| \frac{\delta \phi}{\delta z} \right|_{z=0}^{z=Z} \right]$$

I'm not sure how the adjoint helps me though.

Problem 3

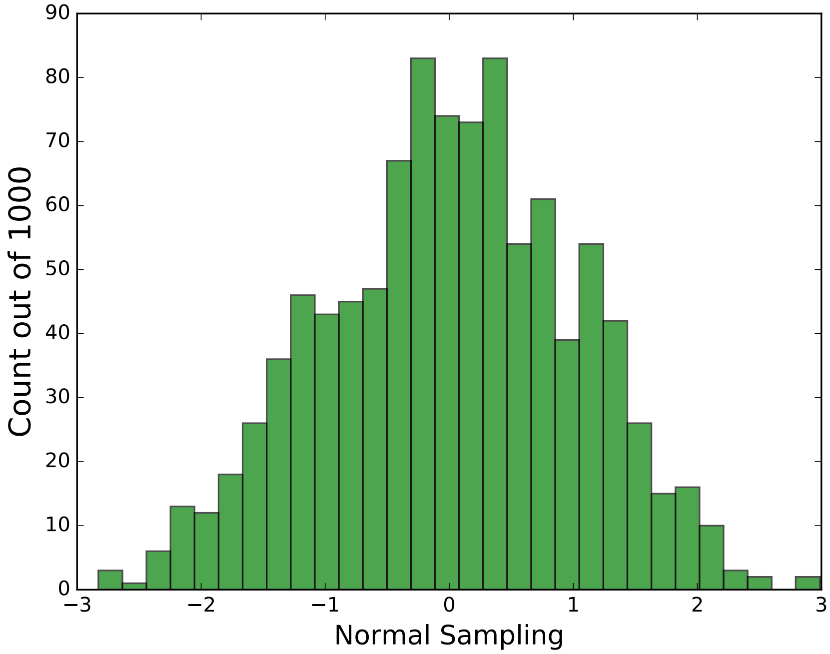
For the random variable X ∼ N(0,1) draw fifty samples and generate histograms using the following sampling techniques

(a) Simple random sampling

(b) Stratifid sampling

Listing 1: Script for Problem

```
#!/usr/bin/env python3

###############################################################
#################### Import packages ########################
###############################################################

import numpy as np
import matplotlib.pyplot as plt
import time
start_time = time.time()
from scipy.stats import norm

###############################################################
##################### Calculations ##########################
###############################################################

N=1000
Nbins=30
RandomNumbers=np.random.uniform(low=0,high=1,size=N)

#Sample the inverse of the CDF of the standard normal
#distribution
Samples=norm.ppf(RandomNumbers)

#Generate histogram
fig=plt.figure()
ax=fig.add_subplot(111)
ax.set_xlabel('Normal Sampling',fontsize=16)
ax.set_ylabel('Count out of '+str(N),fontsize=18)
ax.hist(Samples,Nbins,color='green',alpha=0.7,edgecolor='black')
#ax.set_xlim(-500,500)
plt.savefig("NNorm.pdf")


#Strat sampling...



##################### Time To execute ################

print("--- %s seconds ---" % (time.time() - start_time))
```


Count out of 1000
Normal Sampling
0
10
20
30
40
50
60
70
80
90
−3
−2
−1
0
1
2
3