Assignment 7

Name: Anoosha Sagar

Student ID: 605028604

Lasso Regression

Testing Code

```
> n = 50
> p = 200
> s = 10
> X = matrix(rnorm(n * p), nrow=n)
> lambda_all = (100 : 1) * 10
> beta_true = matrix(rep(0, p), nrow = p)
> beta_true[1:s] = 1 : s
> Y = X %*% beta_true + rnorm(n)
> beta_all = Lasso(X, Y, lambda_all)
> ans = glmnet(X, Y)
> plot(ans)
```

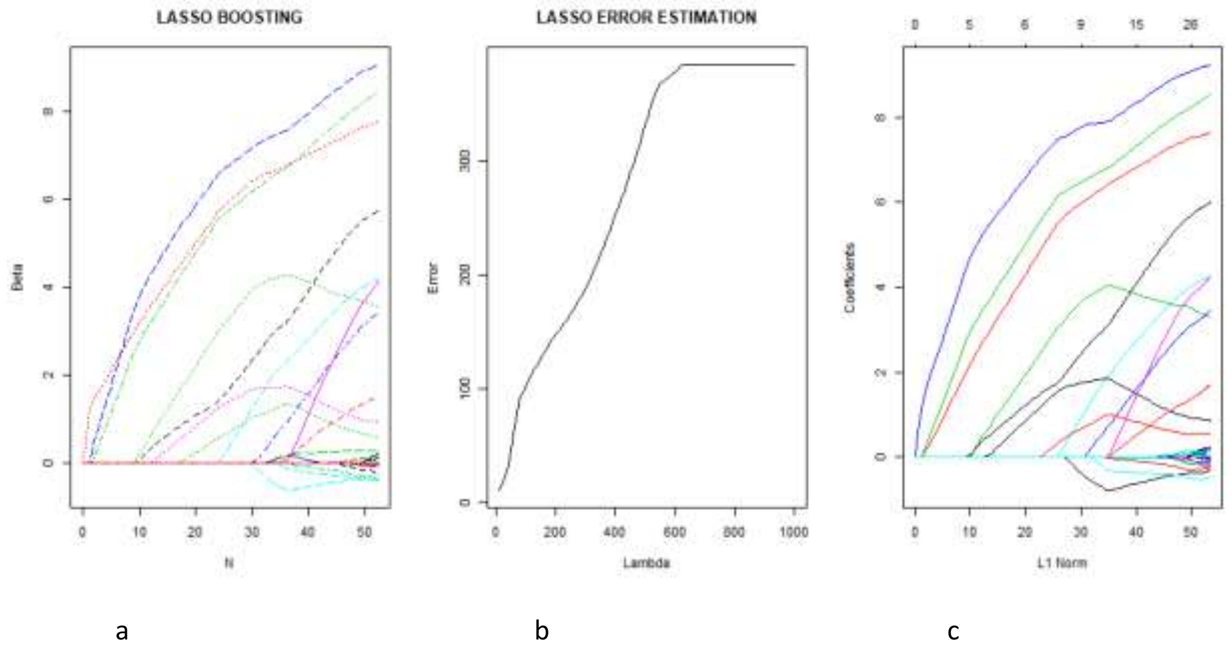


Fig 1: a)Result of Lasso Boosting; b) L asso Error Estimation; c) Lasso Boosting using glmnet

Testing with real world data sets

Linear Regression, Ridge Regression, and Lasso Boosting

Data Set Used: Swiss Data Set

Description: Standardized fertility measure and socio-economic indicators for each of 47 French-speaking provinces of Switzerland at about 1888.

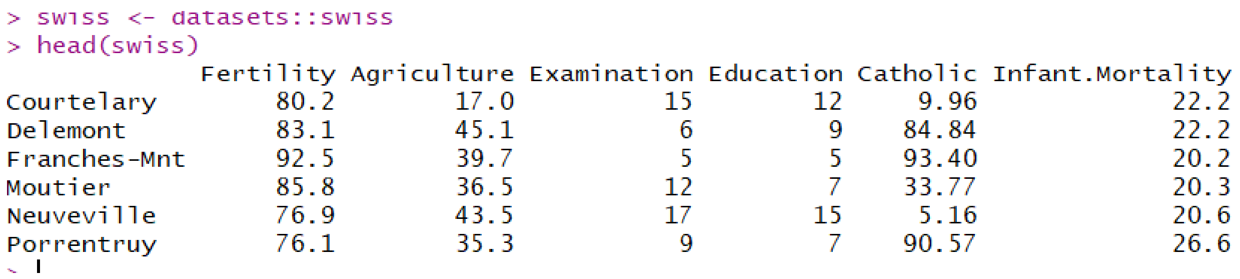
Format:

A data frame with 47 observations on 6 variables, *each* of which is in percent, i.e., in *[0, 100]*.

[,1]	Fertility	*Ig*, 'common standardized fertility measure'
[,2]	Agriculture	% of males involved in agriculture as occupation
[,3]	Examination	% draftees receiving highest mark on army examination
[,4]	Education	% education beyond primary school for draftees.
[,5]	Catholic	% 'catholic' (as opposed to 'protestant').
[,6]	Infant.Mortality	live births who live less than 1 year.

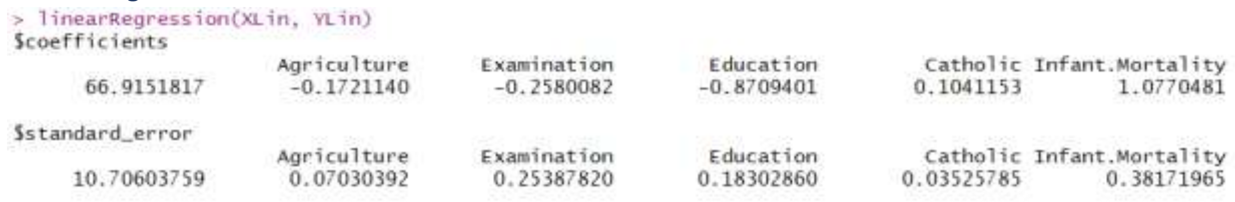
All variables but 'Fertility' give proportions of the population.

Snapshot of Data



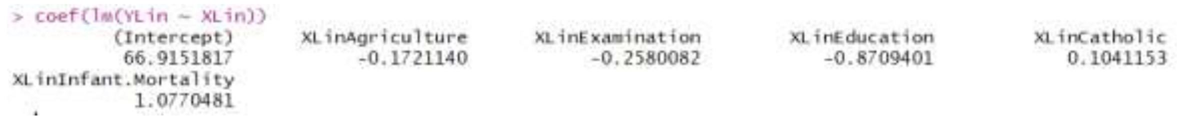
```
> swiss <- datasets::swiss
> head(swiss)
             Fertility Agriculture Examination Education Catholic Infant.Mortality
Courtelary        80.2        17.0          15        12     9.96             22.2
Delemont          83.1        45.1           6         9    84.84             22.2
Franches-Mnt      92.5        39.7           5         5    93.40             20.2
Moutier           85.8        36.5          12         7    33.77             20.3
Neuveville        76.9        43.5          17        15     5.16             20.6
Porrentruy        76.1        35.3           9         7    90.57             26.6
>
```

Linear Regression



```
> linearRegression(XLin, YLin)
$coefficients
                   Agriculture      Examination        Education         Catholic Infant.Mortality
     66.9151817     -0.1721140       -0.2580082       -0.8709401        0.1041153        1.0770481

$standard_error
                   Agriculture      Examination        Education         Catholic Infant.Mortality
    10.70603759     0.07030392       0.25387820       0.18302860       0.03525785       0.38171965
```

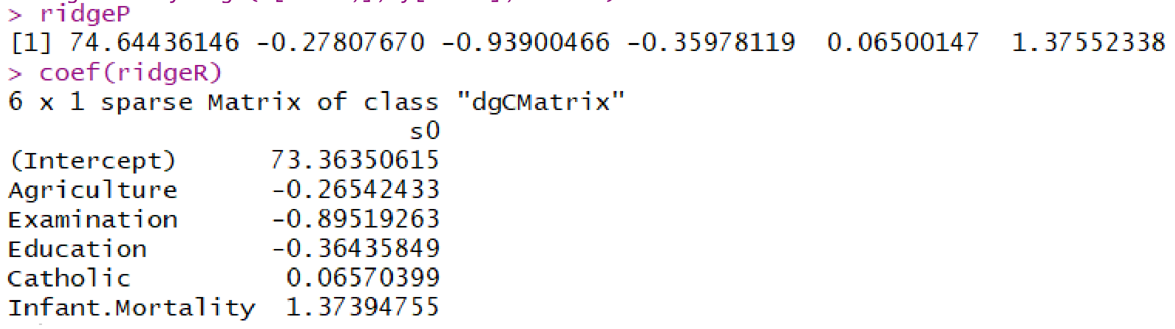


```
> coef(lm(YLin ~ XLin))
         (Intercept)      XLinAgriculture      XLinExamination        XLinEducation         XLinCatholic
          66.9151817           -0.1721140           -0.2580082           -0.8709401            0.1041153
XLinInfant.Mortality
           1.0770481
```

The values of coefficients obtained using the built in function for linear regression provided by R and the function in my package are similar.

Ridge Regression

```
> XRidge <- model.matrix(Fertility~., swiss)[,-1]
> YRidge <- swiss$Fertility
> lambda <- 10^seq(10, -2, length = 100)
> set.seed(489)
> train <- sample(1: nrow(XRidge), nrow(XRidge)/2)
> test <- (-train)
> yTest <- YRidge[test]
> lambda <- 0.1
> ridgeR <- glmnet(XRidge[train,], YRidge[train], alpha = 0, lambda = lambda)
> ridgeP <- myRidge(x[train,], y[train], lambda)
```

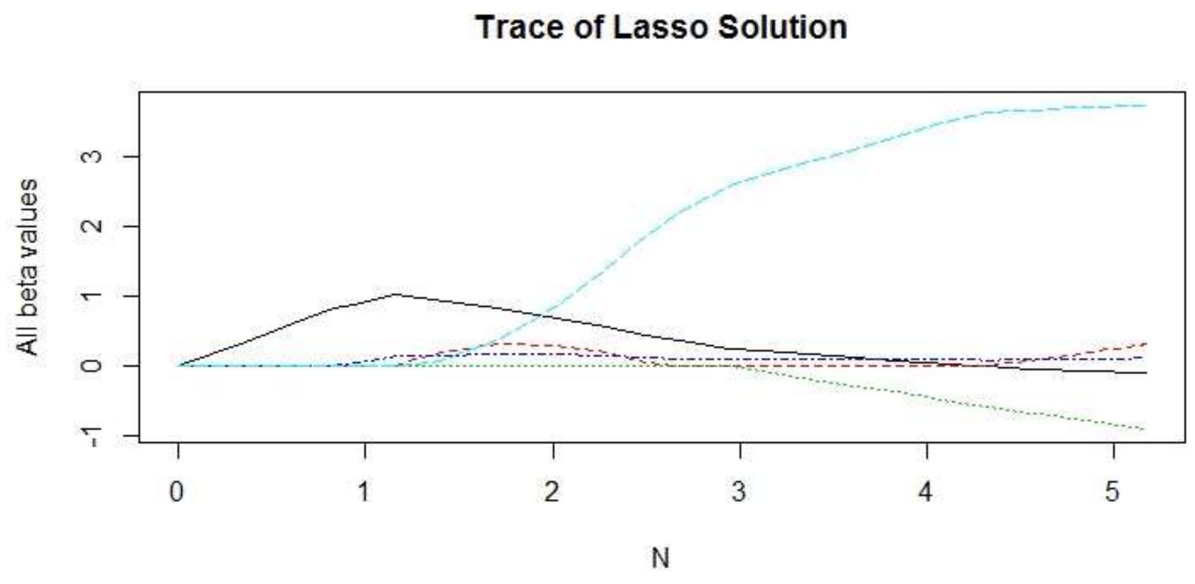


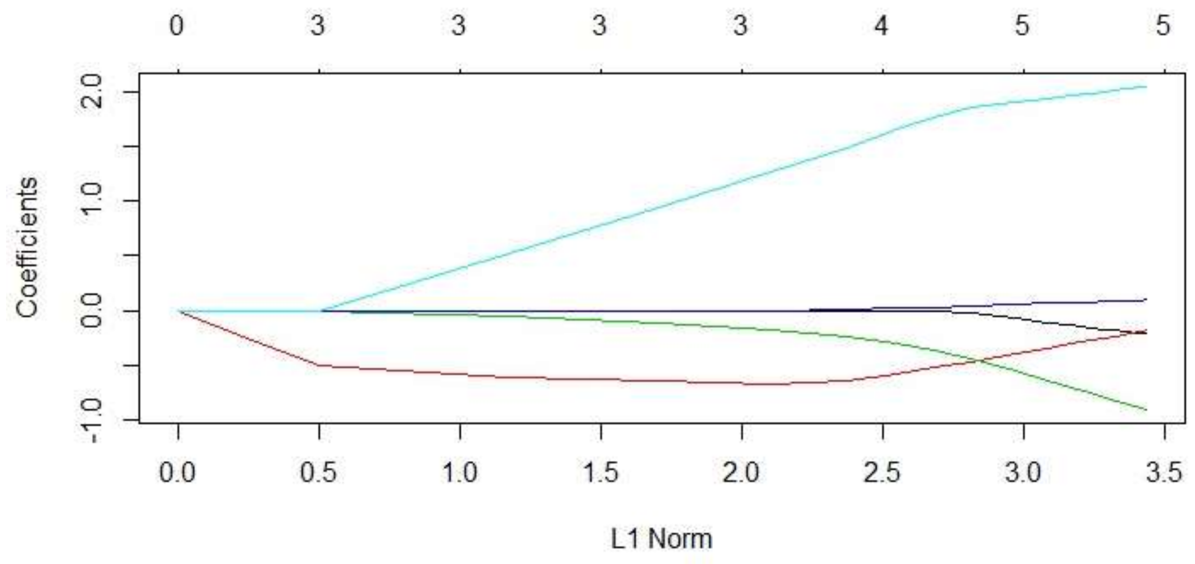
```
> ridgeP
[1] 74.64436146 -0.27807670 -0.93900466 -0.35978119  0.06500147  1.37552338
> coef(ridgeR)
6 x 1 sparse Matrix of class "dgCMatrix"
                          s0
(Intercept)      73.36350615
Agriculture      -0.26542433
Examination      -0.89519263
Education        -0.36435849
Catholic          0.06570399
Infant.Mortality  1.37394755
```

The values of coefficients obtained using the built in function for ridge regression provided by R and the function in my package are similar.

Lasso Boosting

```
> lambda <- 10^seq(10, -2, length = 100)
> library(Stats202ARegression)
> cv.out <- cv.glmnet(XRidge[train,], YRidge[train], alpha = 0)
> bestlam <- cv.out$lambda.min
> lasso.mod <- glmnet(XRidge[train,], YRidge[train], alpha = 1, lambda = lambda)
> lasso.pred <- predict(lasso.mod, s = bestlam, newx = XRidge[test,])
> mean((lasso.pred-ytest)^2)
[1] 113.7041
> lasso.coef <- predict(lasso.mod, type = 'coefficients', s = bestlam)[1:6,]
> lasso.coef
     (Intercept)      Agriculture      Examination
     57.85476722      -0.06225277      -0.50145205
       Education         Catholic Infant.Mortality
     -0.12425311       0.04456320       1.25231604
> res <- Lasso(XRidge[train,], YRidge[train], lambda)
> plot(lasso.mod)
```





The values of coefficients obtained using the built-in function for lasso boosting provided by R and the function in my package are similar.

As we can observe, the results obtained in all the three scenarios are different. This occurs as linear regression doesn't perform regularization whereas Ridge Regression performs regularization.

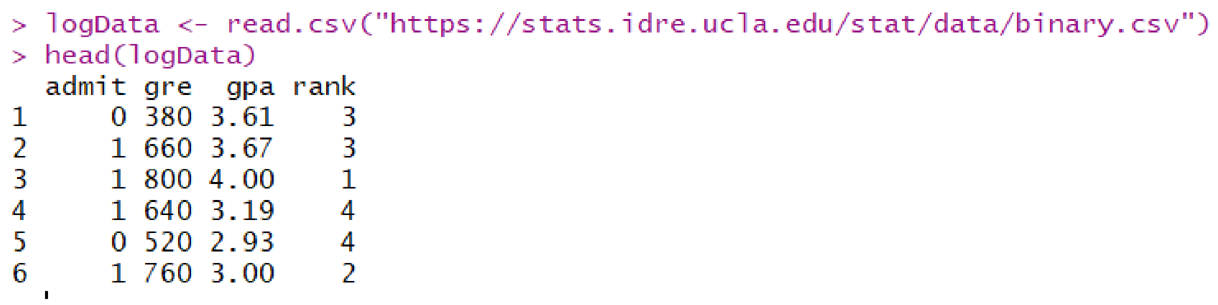
Lasso Boosting performs variable selection and regularization to enhance the prediction accuracy.

Logistic Regression

Data Set Used: IDRE's Binary Data Set (https://stats.idre.ucla.edu/stat/data/binary.csv)

Description: This dataset has a binary response (outcome, dependent) variable called admit. There are three predictor variables: gre, gpa and rank. We will treat the variables gre and gpa as continuous. The variable rank takes on the values 1 through 4. Institutions with a rank of 1 have the highest prestige, while those with a rank of 4 have the lowest.

Snapshot of Data



```
> logData <- read.csv("https://stats.idre.ucla.edu/stat/data/binary.csv")
> head(logData)
  admit gre  gpa rank
1     0 380 3.61    3
2     1 660 3.67    3
3     1 800 4.00    1
4     1 640 3.19    4
5     0 520 2.93    4
6     1 760 3.00    2
```

```
> logData <- read.csv("https://stats.idre.ucla.edu/stat/data/binary.csv")
> X <- as.matrix(logData[, 2:4])
> Y <- as.matrix(logData[, 1])
> X[, 1] <- (X[, 1] - mean(X[, 1])) / sd(X[, 1])
> X[, 2] <- (X[, 2] - mean(X[, 2])) / sd(X[, 2])
> X[, 3] <- (X[, 3] - mean(X[, 3])) / sd(X[, 3])
> logisticRegression(X, Y)
$coefficients
       gre        gpa       rank
 0.2233584  0.2510192 -0.4472078

$standard_error
      gre       gpa      rank
0.1147555 0.1140612 0.1082179

> print(glm(formula = Y ~ X + 0, family = "binomial"))

Call:  glm(formula = Y ~ X + 0, family = "binomial")

Coefficients:
   Xgre     Xgpa    Xrank
 0.2217   0.2500  -0.4453

Degrees of Freedom: 400 Total (i.e. Null);  397 Residual
Null Deviance:          554.5
Residual Deviance: 519.9     AIC: 525.9
```

The values of coefficients obtained using the built in function for logistic regression provided by R and the function in my package are similar.

PCA

Data Set Used: Iris data set

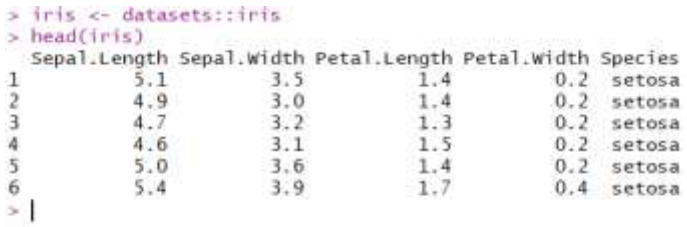
Description: This famous (Fisher's or Anderson's) iris data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are Iris setosa, versicolor, and virginica.

Format:

`iris` is a data frame with 150 cases (rows) and 5 variables (columns) named `Sepal.Length`, `Sepal.Width`, `Petal.Length`, `Petal.Width`, and `Species`.

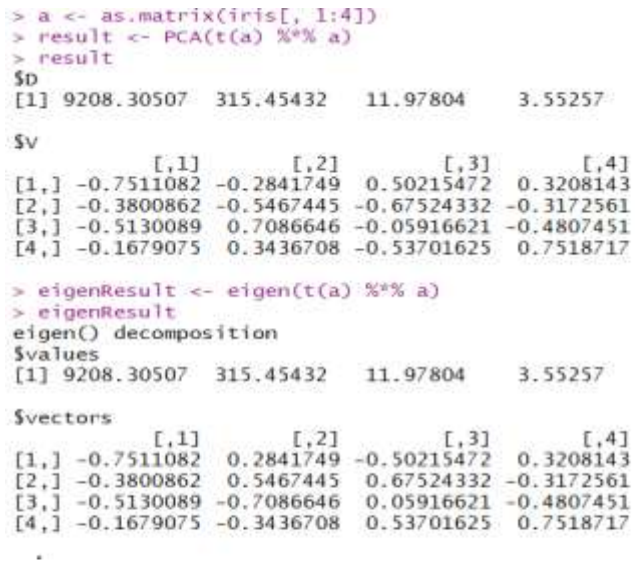
`iris3` gives the same data arranged as a 3-dimensional array of size 50 by 4 by 3, as represented by S-PLUS. The first dimension gives the case number within the species subsample, the second the measurements with names `Sepal L.`, `Sepal W.`, `Petal L.`, and `Petal W.`, and the third the species.

Snapshot of Data



```
> iris <- datasets::iris
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
> |
```

Comparison of Results Obtained from PCA function in my package and the built in eigen function



```
> a <- as.matrix(iris[, 1:4])
> result <- PCA(t(a) %*% a)
> result
$D
[1] 9208.30507  315.45432   11.97804    3.55257

$V
           [,1]       [,2]        [,3]       [,4]
[1,] -0.7511082 -0.2841749  0.50215472  0.3208143
[2,] -0.3800862 -0.5467445 -0.67524332 -0.3172561
[3,] -0.5130089  0.7086646 -0.05916621 -0.4807451
[4,] -0.1679075  0.3436708 -0.53701625  0.7518717

> eigenResult <- eigen(t(a) %*% a)
> eigenResult
eigen() decomposition
$values
[1] 9208.30507  315.45432   11.97804    3.55257

$vectors
           [,1]       [,2]        [,3]       [,4]
[1,] -0.7511082  0.2841749 -0.50215472  0.3208143
[2,] -0.3800862  0.5467445  0.67524332 -0.3172561
[3,] -0.5130089 -0.7086646  0.05916621 -0.4807451
[4,] -0.1679075 -0.3436708  0.53701625  0.7518717
```

The eigen values obtained using both the functions obtained is same.