1.Data Manipulation

I used the following code to perform the Data manipulation:

```
############Data Manipulation#############
# Remove completely missing columns.
missCol=logical(length = dim(myData)[2])
for(ii in 1:dim(myData)[2]){
    if(sum(abs(myData[,ii])>matrix(0.0001,dim(myData)[1],1), na.rm = TRUE)>0){
    missCol[ii] = T
    }else{
    missCol[ii] = F
    }
}
myData1=myData[,missCol]
# Remove rows with completely missing predictors.
missRow = logical(length = dim(myData1)[1])
for(ii in 1:dim(myData1)[1]){
    if(sum(abs(myData1[ii,2:dim(myData1)[2]])>matrix(0.0001,1,dim(myData1)[2]), na.rm = TRUE)>0){
    missRow[ii] = T
    }else{
    missRow[ii] = F
    }
}
myData2=myData1[missRow,]
# Remove columns with constant predictors.
myData3=myData2[,apply(myData2,2,function(xx){
    if(var(xx,na.rm=TRUE)<0.0001){
    return(F)
    }else{
    return(T)}
    }
)]
# Remove (second or later occurrance of) columns which are collinear (correlation 1) with another.
correlation = cor(myData3[,-1], use ="pairwise.complete.obs")
temp = matrix(data = FALSE, nrow = dim(myData3)[2]-1, ncol = dim(myData3)[2]-1)
for(ii in 1:(dim(myData3)[2]-1)){
    for(jj in ii:(dim(myData3)[2]-1)){
        if (correlation[ii,jj]>0.9999 & ii!=jj){
            temp[ii,jj] = T
        }
    }
}
corrtemp = colSums(temp)
corrtemp = (corrtemp==0)
corrtemp = append(corrtemp,1,after = 0)
myData4 = myData3[,as.logical(corrtemp)]
```

This yields the following results:

Number of missing columns removed :	**3**
Number of missing rows removed :	4
Number of columns with constant predictors:	0
Number of collinear columns removed:	16

2. Multiple Imputation

Multiple imputation is a statistical technique for analyzing incomplete data sets, that is, data sets for which some entries are missing. As a result, multiple imputation allows us to perform complete dataset analyses (which many/most statistical and machine learning procedures require), and obtain corresponding

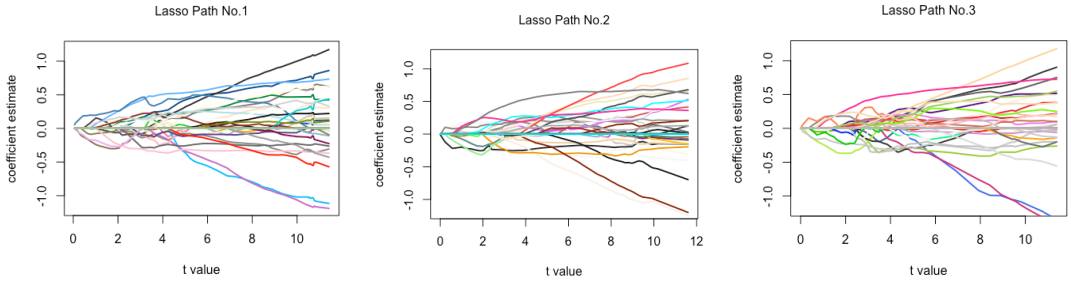
estimates of interest on each complete dataset. In this project, three imputed datasets are created through the step() function. This yields a new dataset of 150 new predictors.

3. Lasso via Modification to Least Angle Regression

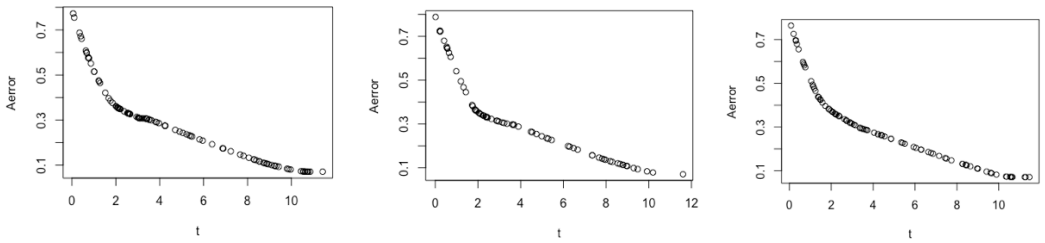
The lasso algorithm proposed by Efron is realized through R.

Dataset1	Dataset2	Dataset3

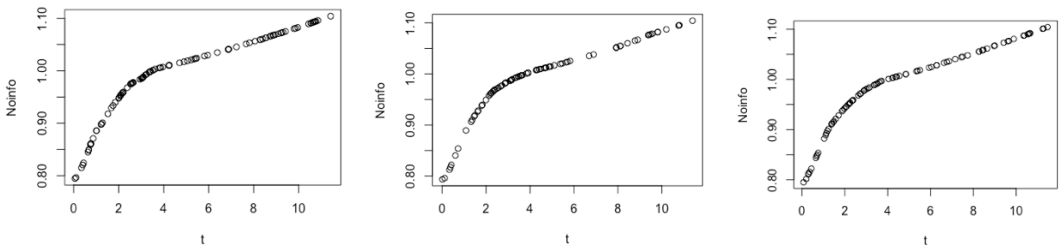
Lasso Path



apparent mean absolute predictive error rate



the no information error rate



4. ACCURACY ASSESSMENT AND MODEL SELECTION

For each multiply imputed dataset, we generate B=25 bootstrap samples. Then we use them to calculate the so-called leave-one-out bootstrap estimate predictive error for every t:

$$\widehat{Err}^{(1)} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{|C^{-i}|}\sum_{b\in C^{-i}} Q(y_i, \hat{\mu}^b(x_i))$$

$$Q\left(y_i, \hat{\mu}^b(x_i)\right) = \left|(y_i - \overline{y^b}) - ((\boldsymbol{x}_i - \overline{\boldsymbol{x}^b})/sd(\boldsymbol{x}^b)) \times \boldsymbol{\beta}^b\right|$$

Since we have different $t=\Sigma|\beta i|$ for different bootstrap sample. We need to use linear interpolation in order to get the same t values.

We compute the 0.632+ bootstrap estimate for every data set based on the relative overfitting rate, no information error rate, apparent error rate and the leave-one-out bootstrap estimate of error.

Apparent error rate:

$$\overline{\text{err}} = \frac{1}{n}\sum_{i=1}^{n} Q(y_i, \hat{\mu}(\mathbf{x}_i)).$$

No information error rate:

$$\hat{\gamma} = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} Q(y_i, \hat{\mu}(\mathbf{x}_j)).$$

Relative rate of overfitting:

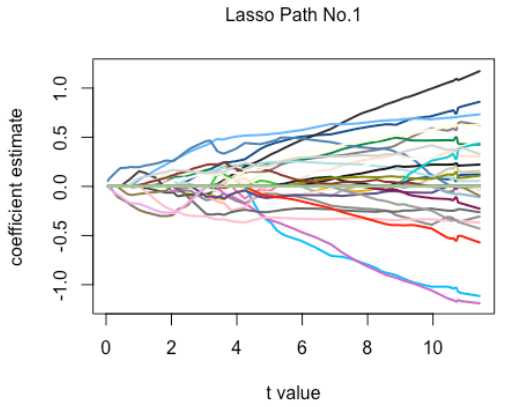
$$\hat{R} = \frac{\widehat{\text{Err}}^{(1)} - \overline{\text{err}}}{\hat{\gamma} - \overline{\text{err}}}.$$

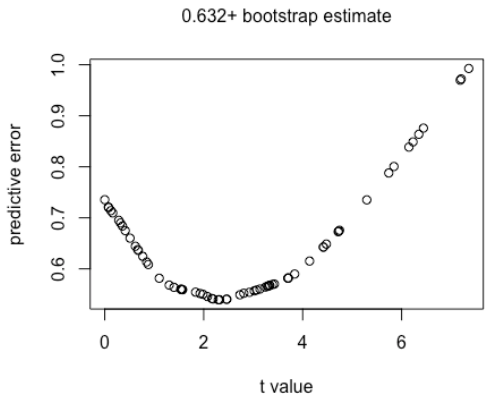
The 0.632+ bootstrap estimate of predictive error:

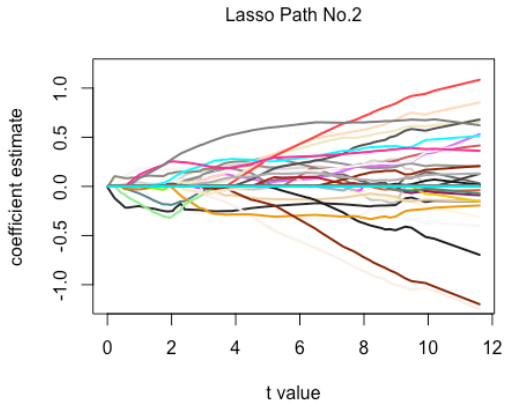
$$\widehat{\text{Err}}^{(0.632+)} = (1-\hat{w}) \times \overline{\text{err}} + \hat{w} \times \widehat{\text{Err}}^{(1)}, \quad \text{where} \quad \hat{w} = \frac{0.632}{1 - 0.368\hat{R}}.$$

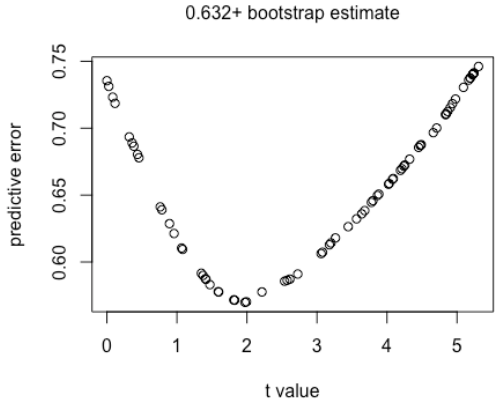
5.Visualization

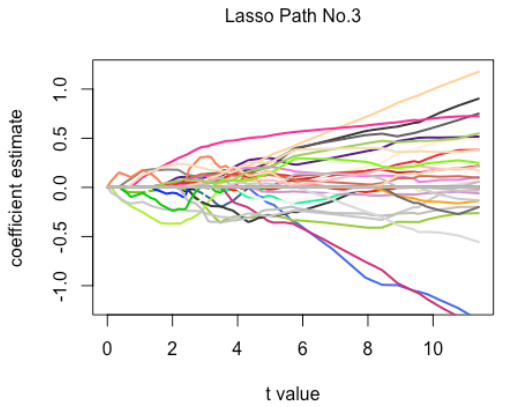
We plot a 3 row by 2 column panel plot: the first column contains the lasso paths for each of the datasets. The second column contains the 0.632+ bootstrap estimate of expected mean absolute error for the 3 data sets

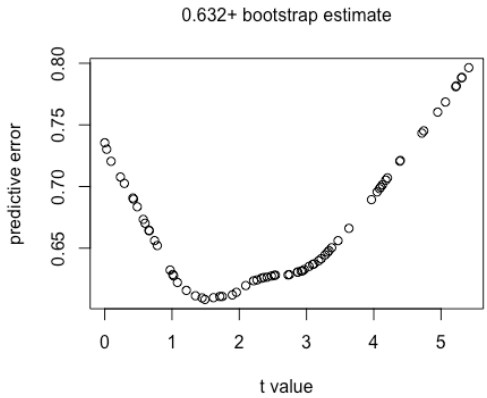




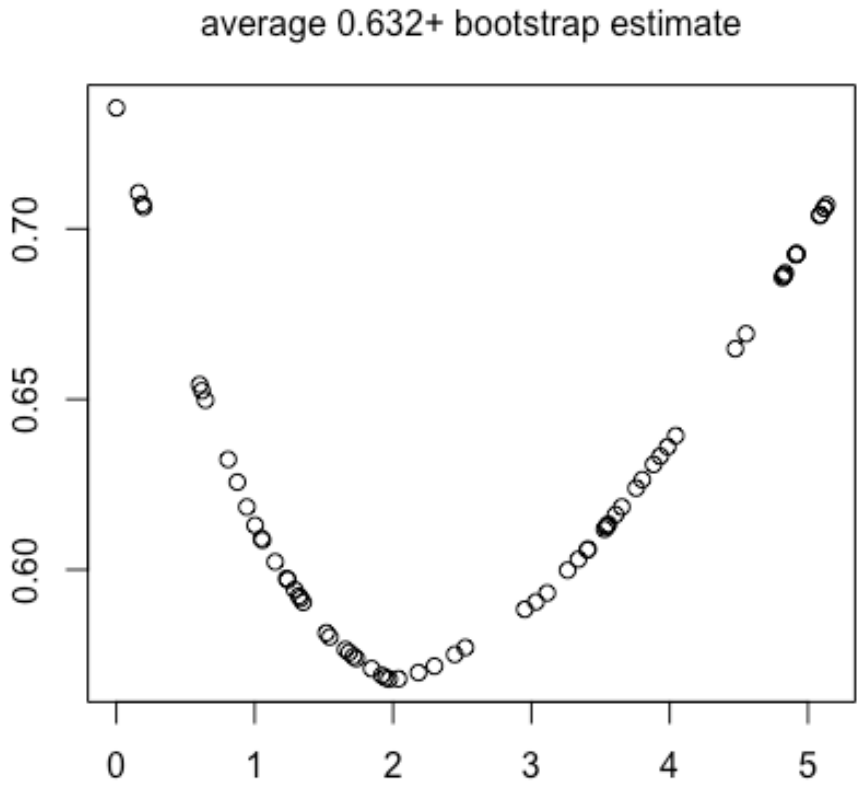








average estimate of expected mean absolute error across three multiple imputation datasets is plotted.

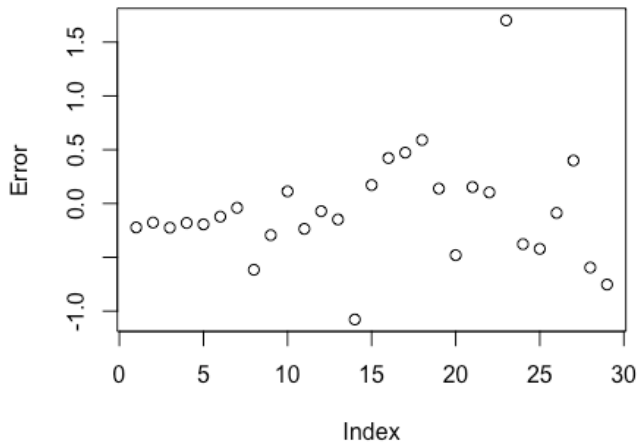


6.Interpretation

In this example, we choose t = 2. Then I averaged the Coefficients of the regression to assess the quality of the fit. I calculated the error between the actual response and the prediction.

The observations 14, 24 weren't fitted very well (largest error).

The signs of the coefficients are very important. They show which predictors predict an increase or a decrease of the PFS . We should be cautious about the interpretation of the absolute values of the coefficients. In fact, their impact depends on the range of the predictors.
The predictor " v1 High.Intensity.Large.Zone.Emphasis average " has the largest coefficient 0.281901
The predictor " v1 Texture.Strength average " has the largest absolute coefficient with value -0.287835
If the coefficient is positive, the increase of the corresponding predictor will increase the PFS, vice versa. The increase of 1 unit of the predictor(Texture Strength) will lead the PFS to decrease 0.75



R Code

```
[1] "v1 Correlation1 average"
[2] "v1 Maximum.Probability average"
[3] "v1 Information.Measure.of.Correlation1 average"
[4] "v1 Homogeneity1.or.Inverse.Difference average"
[5] "v1 Cluster.Prominence.1 average"
[6] "v1 Coarseness average"
[7] "v1 Complexity average"
[8] "v1 Texture.Strength average"
[9] "v1 Entropy.2 average"
[10] "v1 Mean.SD average"
[11] "v1 Large.Zone.Emphasis average"
[12] "v1 Zone.Percentage average"
[13] "v1 High.Intensity.Large.Zone.Emphasis average"
[14] "v1 Cluster.Shade difference"
[15] "v1 Sum.of.Squares.or.Variance difference"
[16] "v1 Sum.Variance difference"
[17] "v1 Difference.Variance difference"
[18] "v1 Information.Measure.of.Correlation1 difference"
[19] "v1 Inverse.Difference.Moment.Normalized difference"
[20] "v1 Contrast.1 difference"
[21] "v1 Cluster.Shade.1 difference"
[22] "v1 Sum.Average.1 difference"
[23] "v1 Sum.Variance.1 difference"
[24] "v1 Difference.Entropy.1 difference"
[25] "v1 Information.Measure.of.Correlation1.1 difference"
[26] "v1 Inverse.Difference.Moment.Normalized.1 difference"
[27] "v1 Long.Run.Emphasis difference"
[28] "v1 Coarseness difference"
[29] "v1 Texture.Strength difference"
[30] "v1 Entropy.2 difference"
[31] "v1 Maximum.Intensity difference"
[32] "v1 Mean.Intensity difference"
[33] "v1 High.Intensity.Zone.Emphasis difference"
```

R Code

```
# Read in labelname and response data
myXData=read.csv("texture.csv",header=TRUE)
myYData=read.csv("textureResponse.csv",header=TRUE)
# Restrict attention to particular variables

myLogical=sapply(attributes(myXData)$names,function(varname){
    if(varname %in% c("PatID","TimePoint","PFS")){
        return(TRUE)
    }else{
        return(((length(grep(pattern="v1_",varname))>0))&
              (!((length(grep(pattern="Xa",varname))>0)|
            (length(grep(pattern="DeltaX",varname))>0))))
    }
})

myXData=myXData[,myLogical]
# Rearrange data

myXtemp=data.matrix(myXData[,6:dim(myXData)[2]])

patientIDsTemp=sort(unique(c(myXData$PatID,myYData$Pat.No)))

timePointsTemp=sort(unique(as.character(myXData$TimePoint)))

PFS=matrix(NA,length(patientIDsTemp),1)

predictors=matrix(NA,length(patientIDsTemp),(dim(myXtemp)[2])*length(timePointsTemp))

counter=1

for(pt in patientIDsTemp){
    index=which(myYData$Pat.No==pt)
    if(length(index)==1){
        PFS[counter]=log(myYData$PFS[index])
}
for(jj in 1:length(timePointsTemp)){
    index=which((myXData$TimePoint==timePointsTemp[jj])&
                (myXData$PatID==pt))
    if(length(index)==1){
        predictors[counter,((jj-1)*dim(myXtemp)[2]+1):(jj*dim(myXtemp)[2])]=
        myXtemp[index,]
    }
}
counter=counter+1
}
predictorsStar=apply(predictors,2,function(xx){
```

```
    if(sum(!is.na(xx))>=2){
        return((xx-mean(xx,na.rm=TRUE))/sd(xx,na.rm=TRUE))
    }else{
        return(xx)
    }
})
myData=data.frame(PFS,predictorsStar)
myData$PFS=myData$PFS-mean(myData$PFS,na.rm=TRUE)
# Label columns
myNames=c("PFS",
    paste(attributes(myXData)$names[6:dim(myXData)[2]],"_baseline1",sep=""),
    paste(attributes(myXData)$names[6:dim(myXData)[2]],"_baseline2",sep=""),
    paste(attributes(myXData)$names[6:dim(myXData)[2]],"_followUp",sep=""))
attributes(myData)$names=myNames


############Data Manipulation##############
# Remove completely missing columns.
missCol=logical(length = dim(myData)[2])
for(ii in 1:dim(myData)[2]){
    if(sum(abs(myData[,ii])>matrix(0.0001,dim(myData)[1],1), na.rm = TRUE)>0){
    missCol[ii] = T
    }else{
    missCol[ii] = F
    }
}
myData1=myData[,missCol]
# Remove rows with completely missing predictors.
missRow = logical(length = dim(myData1)[1])
for(ii in 1:dim(myData1)[1]){
    if(sum(abs(myData1[ii,2:dim(myData1)[2]])>matrix(0.0001,1,dim(myData1)[2]), na.rm = TRUE)>0){
    missRow[ii] = T
    }else{
    missRow[ii] = F
    }
}
myData2=myData1[missRow,]
# Remove columns with constant predictors.
myData3=myData2[,apply(myData2,2,function(xx){
    if(var(xx,na.rm=TRUE)<0.0001){
    return(F)
    }else{
    return(T)}
    }
)]
# Remove (second or later occurrance of) columns which are collinear (correlation 1) with another.
```

```
correlation = cor(myData3[,-1], use ="pairwise.complete.obs")
temp = matrix(data = FALSE, nrow = dim(myData3)[2]-1, ncol = dim(myData3)[2]-1)
for(ii in 1:(dim(myData3)[2]-1)){
    for(jj in ii:(dim(myData3)[2]-1)){
        if (correlation[ii,jj]>0.9999 & ii!=jj){
            temp[ii,jj] = T
        }
    }
}
corrtemp = colSums(temp)
corrtemp = (corrtemp==0)
corrtemp = append(corrtemp,1,after = 0)
myData4 = myData3[,as.logical(corrtemp)]

############ Multiple Imputation ##############
newmyData=myData4
NA_logical =logical(length = dim(newmyData)[2])
for(i in 1:dim(newmyData)[2]){
  NA_logical[i]=any(which(is.na(newmyData[,i])))
}
  tempData = newmyData
  for (i in 1:dim(newmyData)[2]){
     if(NA_logical[i] == T){
       row_index = which((is.na(tempData[,i])) == T)
       row_No_NA = which((is.na(tempData[,i])) == F)
       tempData[row_index,i] = sample(tempData[row_No_NA,i],size=length(row_index),replace=T)
       }
  }

  for (i in 1:dim(newmyData)[2]) {
    if(NA_logical[i] == T){
        row_NA = which((is.na(newmyData[,i])) == T)
      linereg = tempData[-row_NA,i]
      mini=lm(linereg~1,data = tempData[-row_NA,-i])
        maxi=lm(linereg~.,data = tempData[-row_NA,-i])
        selectvar = step(mini,scope = list(upper=maxi,lower = mini),direction="forward",steps = 10,k = 2)
        coefficient = coef(summary(selectvar))
      currentList = names(selectvar$coefficients)
        regdata = c()
        for(j in 1:length(currentList)){
          if(currentList[j] == "(Intercept)"){
              regdata = cbind(regdata,tempData[,i])
          }else{
              regdata = cbind(regdata,tempData[,which(attributes(tempData)$name == currentList[j])])
          }
        }
```

```
      RegData = regdata %*% as.matrix(coefficient[,1])
      noisyplus = RegData[row_NA] + rnorm(length(row_NA),0,var(coefficient[,2]))
        for(j in 1:length(row_NA)){
          indexfinal = which.min(abs(RegData[-row_NA]-noisyplus[j]))
         newmyData[row_NA[j],i] = newmyData[-row_NA[j],i][indexfinal]
        }
    }
  }
############ After Imputation #############
Data=newmyData
FUlabel=grep("followUp",attributes(Data)$names)
BL1label=grep("baseline1",attributes(Data)$names)
BL2label=grep("baseline2",attributes(Data)$names)

avg_dif=Data[1]
for (i in 1:75){
    labelname=attributes(Data)$names[FUlabel[i]]
    labelname1=substr(labelname, 1,(nchar(labelname)-8))
    indexbs1=BL1label[which(grepl(labelname1,attributes(Data)$names[BL1label]))]
    indexbs2=BL2label[which(grepl(labelname1,attributes(Data)$names[BL2label]))]
    a=b=0
    if(length(indexbs1)>0){
        a=Data[,indexbs1]
    }
    if(length(indexbs2)>0){
        b=Data[,indexbs2]
    }

    avg=paste(labelname1,"_Average",sep="")
    diff=paste(labelname1,"_Diff",sep="")
    if(length(a)>1 && length(b)>1){
        avg_dif[avg]=(a+b)/2
        avg_dif[diff]=Data[,FUlabel[i]]-(a+b)/2
    }
    if(length(a)>1 && length(b)==1){
        avg_dif[avg]=a
        avg_dif[diff]=Data[,FUlabel[i]]-a
    }
    if(length(a)==1 && length(b)>1){
    avg_dif[avg]=b
    avg_dif[diff]=Data[,FUlabel[i]]-b
    }
}
write.csv(avg_dif,"myData(manipulate)-1.csv")
```

```
rm(list=ls())
myData = read.csv("myData(manipulate)-1.csv")
# Remove row names
myData = myData[,2:dim(myData)[2]]
###################### ITERATIONS ######################
myX = as.matrix(myData[,2:dim(myData)[2]])
myY = myData[,1]
dn = dim(myX)[1]
dp = dim(myX)[2]
meany =mean(myY)
Aerror=c()
Noinfo=c()
# Standardize the predictors
myX = scale(myX)
myY = scale(myY)
nbsteps = 150
k=1
XA = NULL
activeSet = rep(F, dp)
mu = rep(0,dn)
beta = rep(0,dp)
BETA = matrix(0,dp,nbsteps)
C_remove = F
logic_end = T
while(k<=nbsteps && logic_end) {
  error = myY-mu
  if(!C_remove) {
    errCor = cor(myX, error)
    errCor[activeSet] = 0
    index = which.max(abs(errCor))
    activeSet[index] = T
  } else {
    index = which(rmVar==names(as.data.frame(myX)))
    activeSet[index] = F
  }
  sign = sign(cor(myX, error))
  signId = diag(dp)
  for(i in 1:dp) {
    signId[i,i] =sign[i]
  }
  XA = (myX %*% signId)[,activeSet]
  XA = as.matrix((myX %*% signId)[,activeSet])
  oneN = rep(1,sum(activeSet))
  cHat = t(myX) %*% (error)
  cCap =max(abs(t(XA) %*% error))
  a = t(myX) %*% XA %*% solve(t(XA) %*% XA) %*% oneN
```

```
  gamma = c(((cCap-cHat)/(1-a))[!activeSet],
            ((cCap+cHat)/(1+a))[!activeSet])
  gamma = min(gamma[gamma>0.00001])
  #lasso modification
  bTilde = solve(t(XA) %*% XA) %*% oneN
  cStar = -(BETA[activeSet,k-1] / (sign[activeSet]*bTilde))
  cj=Inf
  if(length(cStar[cStar>0.00001])==0) {
    cj=Inf
    indexStar = 0
    rmVar = NULL
  } else {
    cj = min(cStar[cStar>0.00001])
    indexStar = which(cStar==cj)
    rmVar = names(as.data.frame(myX))[activeSet][indexStar]
  }
  C_remove = cj < gamma
  if(C_remove) { gamma = cj }

  uTilde = gamma * (XA %*% solve(t(XA) %*% XA) %*% oneN)
  mu = mu + uTilde
  b = gamma * (solve(t(XA) %*% XA) %*% oneN)
  beta[activeSet] = beta[activeSet] + b*sign[activeSet]
  BETA[,k] = beta
  error = myY-mu
  cHat = t(myX) %*% error
  k = k+1
  logic_end = dim(XA)[2] < dim(XA)[1]-1
  if(C_remove) {logic_end = T}
  Aerror=c(Aerror,sum(abs((myY-meany-myX %*%beta)/29)))
  summ = 0
  for(i in 1:29){
    response = matrix(myY[i],29,1)
    summ = summ + sum(abs(response-meany-mu))
  }
  Noinfo=c(Noinfo,summ/29^2)
}
k = k-1
BETA = BETA[,1:k]
t = colSums(abs(BETA))
plot(t, rep(0,k), type="l", ylim=c(-1.2, 1.2), xlab="t value", ylab="coefficient estimate")
title("Lasso Path No.1", cex.main = 1, font.main= 1)
for(i in 1:dp) { lines(t, BETA[i,], lwd=2, col=sample(colours(), 1)) }
plot(t,Aerror)
plot(t,Noinfo)
```

```
rm(list=ls())
library(Hmisc)
#myData = read.csv("myData-3.csv")
myData = read.csv("myData(manipulate)-1.csv")
# Remove row names
myData = myData[,2:dim(myData)[2]]
source("Lars Al.R")
###############Set up the bootStrap###############
B=25
myY=as.vector(myData[,1])
Mean=mean(myY)
dn=length(myData[,1])
Data=myData
Data0=apply(Data[,2:{dim(Data)[2]}],2,function(xx){
return((xx-mean(xx,na.rm=TRUE))/sd(xx,na.rm=TRUE))
})
Data[,2:dim(Data)[2]]=Data0
Data=as.matrix(Data)
boots=matrix(0,B,dn)
for(i in 1:B){
    boots[i,]=sample(dn,replace=TRUE)
}
Bootstrps=vector("list",B)
for(k in 1:B){
    Data2=Data[boots[k,],]
    Data0=apply(Data2[,2:{dim(Data2)[2]}],2,function(xx){
    return((xx-mean(xx,na.rm=TRUE))/sd(xx,na.rm=TRUE))
    })
    Data2[,2:dim(Data2)[2]]=Data0
    Data2=as.matrix(Data2)
    Bootstrps[[k]]=myLARS(Data2,"lasso",500)
}
###############Leave-One-Out Bootstrap###############
L=c()
for (b in 1:B){
    L=c(L,length(Bootstrps[[b]]$Ts))
}
index=which(L==max(L))[1]
Ts=Bootstrps[[index]]$Ts
Err=matrix(0,length(Ts),2)
for (t in 1:length(Ts)){
    err=0
    N=0
    for(i in 1:dn){
        Ci=apply(boots,1,function(xx){return(!any(xx==i))})
        if(sum(Ci)>0){
```

```
            bootstrp=Bootstrps[Ci]
            N=N+1
            er=0
            for (bi in bootstrp){
                Tsx=bi$Ts
                Betay=as.matrix(bi$Betas)
                Coef1=t(sapply(Betay, unlist))
                Coef2=matrix(0,150,1)
                for (c in 1:150){
                    Coef2[c]=approxExtrap(as.matrix(bi$Ts),as.matrix(Coef1[,c]),Ts[t])$y
#Coef2[c]=approx(as.matrix(bi$Ts),as.matrix(Coef[,c]),Ts[t],rule=2)$y
                }
                er=er+abs(myY[i]-Mean-Data[i,2:dim(Data)[2]]%*%Coef2)
            }
        }
        er=er/sum(Ci)
        err=err+er
    }
    err=err/N
    Err[t,1]=Ts[t]
    Err[t,2]=err
}
###############0.632+ bootstrap###############
Results=myLARS(Data,"lasso")
AbsErr=Results$AppErrs
NoInErr=Results$NoInfErrs
Tss1=Err[,1]
Tss2=Results$Ts
Err632plus=matrix(0,length(Tss1),2)
Err632=matrix(0,length(Tss1),2)
for (t in 1:length(Tss1)){
    errhat=Err[t,2]
    abs_err=approxExtrap(as.matrix(Tss2),as.matrix(AbsErr),Tss1[t])$y
    noinf_er=approxExtrap(as.matrix(Tss2),as.matrix(NoInErr),Tss1[t])$y
    R=(errhat-abs_err)/(noinf_er-abs_err)
    omega=0.632/(1-0.368*R)
    Err632plus[t,1]=Tss1[t]
    Err632[t,1]=Tss1[t]
    Err632plus[t,2]=(1-omega)*abs_err+omega*errhat
    Err632[t,2]=0.632*abs_err+0.368*errhat
}
plot(Err632plus, xlab="t value", ylab="predictive error")
title("0.632+ bootstrap estimate", cex.main = 1, font.main= 1)
plot(Err632, xlab="t value", ylab="predictive error")
title("0.632+ bootstrap estimate", cex.main = 1, font.main= 1)
write.csv(Err632plus,file="Error632p-3.csv")
```

```
rm(list=ls())
library(Hmisc)
###############Average 3 Dataset###############
Error1 = read.csv("Error632-1p.csv")
Error2 = read.csv("Error632-2p.csv")
Error3 = read.csv("Error632-3p.csv")
#myData = read.csv("myData(manipulate)-1.csv")
# Remove row names
Error1 = as.matrix(Error1[,2:dim(Error1)[2]])
Error2 = as.matrix(Error2[,2:dim(Error2)[2]])
Error3 = as.matrix(Error3[,2:dim(Error3)[2]])

MatErr=vector("list",3)
MatErr[[1]]=Error1
MatErr[[2]]=Error2
MatErr[[3]]=Error3
Ts=MatErr[[1]][,1]
Overall=matrix(0,length(Ts),2)
Overall[,1]=Ts
for(t in 1:length(Ts)){
err=MatErr[[1]][t,2]
for(j in 2:3){
err=err+approxExtrap(as.matrix(MatErr[[j]][,1]),as.matrix(MatErr[[j]][,2]),Ts[t])$y
}
Overall[t,2]=err/3
}
par(mfrow=c(1,1))
plot(Overall, xlab="t value", ylab="predictive error")
title("average 0.632+ bootstrap estimate", cex.main = 1, font.main= 1)
```