# Computer Lab 2

*Thomas Zhang*

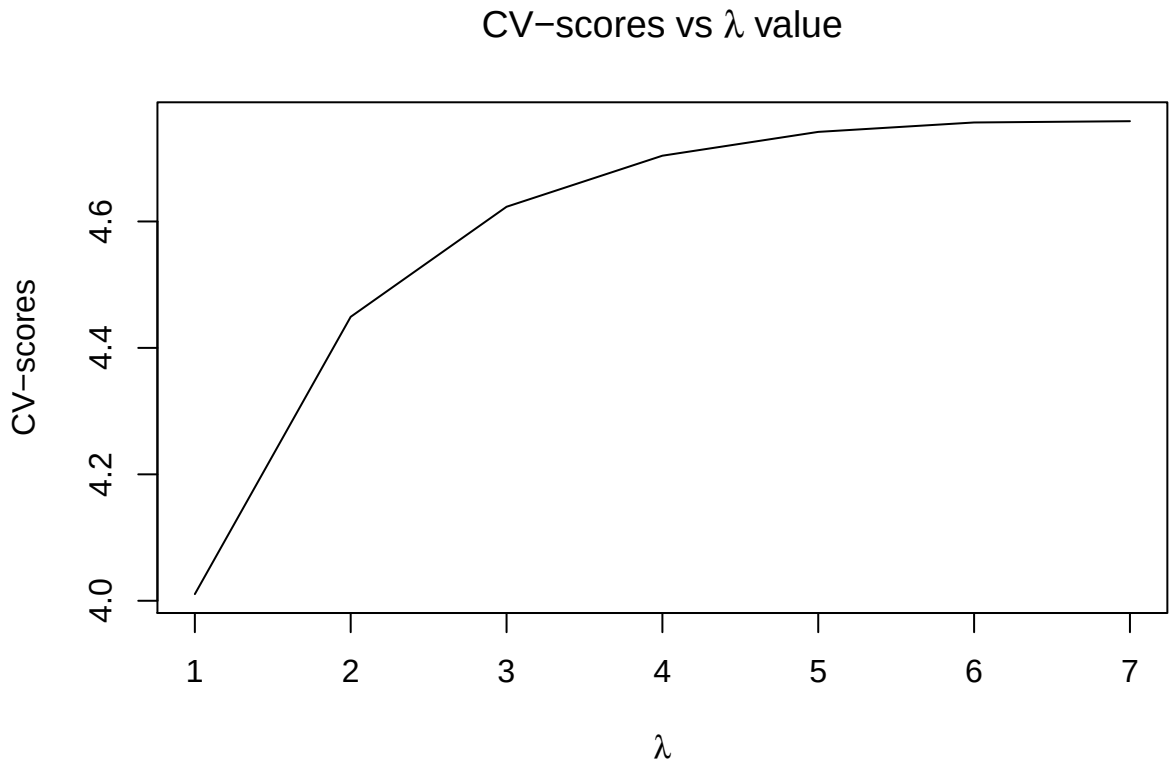
*2015-11-04*

## Assignment 1

The first assignment is a ridge regression on the data set `longley` in the **MASS** library. Ridge regression works by minimizing the expression

$$\min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{b}\|^2 + \|\lambda\mathbf{x}\|^2$$

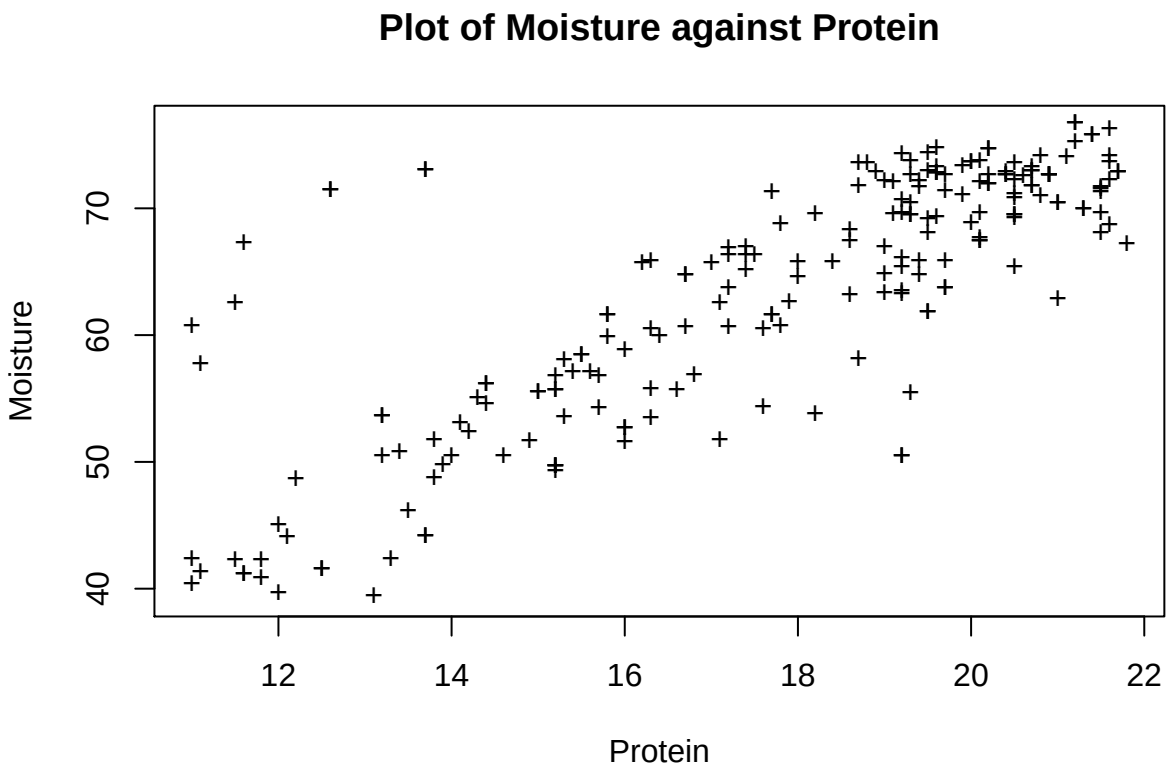
The data set, according to its documentation, contains highly collinear macroceonomic data (can be seen using `pairs`). We perform ridge regression with response variable set to number of people Employed and features set to all other columns in the data set. The hyperparameter $\lambda$ used to penalize the square of the absolute value of the coefficient vector we try in turn to set to 1,2...7 and then we make a plot of the cross-validation scores obtained in 10-fold cross validation, where the loss function is defined to be the sum of the squared residuals. Lower CV-score is better performance by measure of this loss function.




We see that the lowest CV-score is actually obtained at $\lambda = 1$, and then the score gets larger with each subsequent integer $\lambda$ up to seven. This is quite surprising, especially in the face of the fact that the globally optimal $\lambda$ is located somewhere between 300 and 2000, depending on the way one partitions the folds of the cross-validation.

## Assignment 2

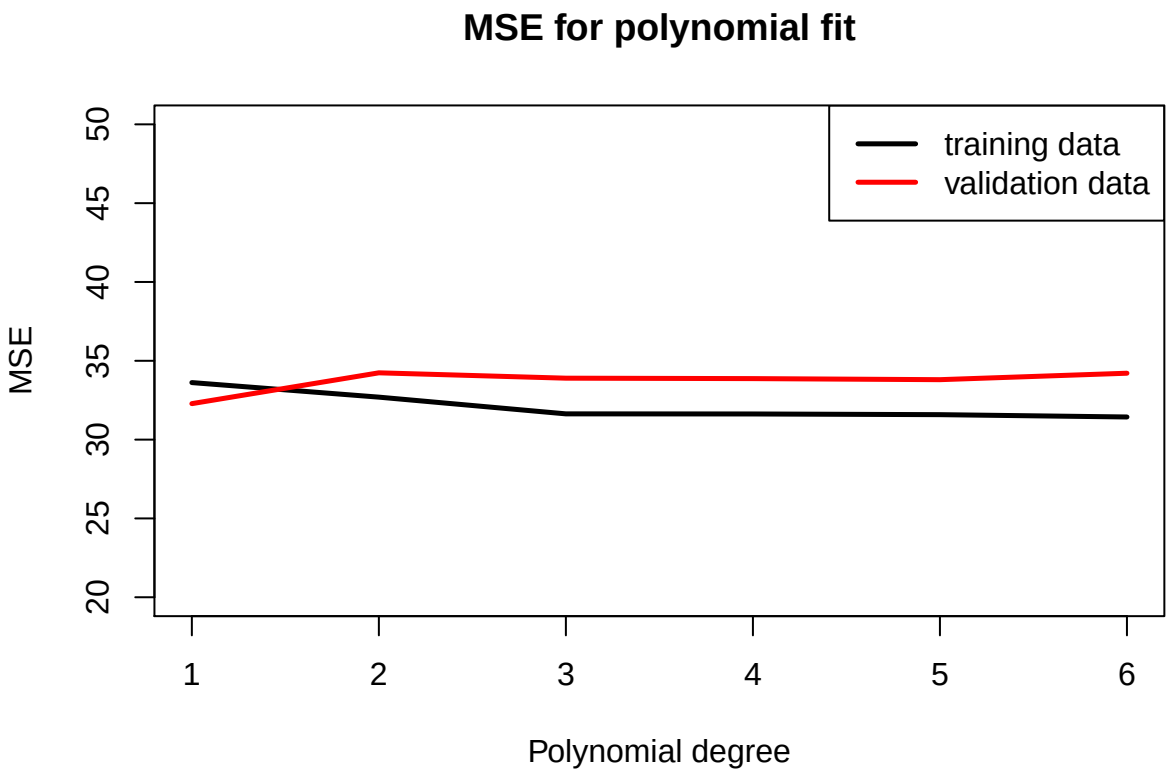
Now we import data from the excel file `tecator.xlsx` and plot the Moisture data against the Protein data.




We see that there are some outliers in the plot but overall the data could be described by a probabilistic model consisting of a polynomial function of degree $i$ with normally distributed error term, such as

$$y_j = \sum_{i=0}^{n} w_i x_j^i + \varepsilon$$

where $y_j$ is the jth Moisture observation, $x_j$ is the jth Protein observation and $\varepsilon$ is $N(0, \sigma^2)$ distributed. We proceed to fit this model for $n = 1 \dots 6$ by the method of least squares using training data consisting of half the observations and make predictions for both training dataset and validation dataset where validation dataset is the complement to the training dataset. Then we compare the Mean Squared Error (actually the mean residual sum of squares) of both the training data set and the validation data set in a plot. Due to the numerical instability at higher polynomial degrees using least squares, we use the *Singular Value Decomposition algorithm* when solving the least squares problem.




We see that the MSE for the training data is monotonically decreasing, as it should, while the MSE for the validation data actually increases a little as we increase the degree of the polynomial fit above linear. This would suggest that the linear model is the best one. We see here an example of the Bias-Variance tradeoff as the polynomial degree increases. Bias decreases, as the higher order polynomials track the data points more closely and hence training data MSE decreases, while variance increases as the polynomial fit becomes more 'wiggly' and the validation data MSE increases a little.

Now we calculate the AIC values for the different polynomial models, defined as

$$AIC = -2 loglikelihood(D) + 2d(model)$$

where $loglikelihood(D)$ is the loglikelihood of the outcome of the entire dataset $D$ given parameters found using least squares and $d(model)$ is the number of parameters in the model. The parameter $\sigma^2$ is estimated as the MSE for the entire data set when fitted with the linear model.

```
## [1] "AIC at polynomial degree 1 is : 1367.54608761313"
## [1] "AIC at polynomial degree 2 is : 1372.95246776673"
## [1] "AIC at polynomial degree 3 is : 1370.38993610719"
## [1] "AIC at polynomial degree 4 is : 1372.28365601597"
## [1] "AIC at polynomial degree 5 is : 1373.92026628577"
## [1] "AIC at polynomial degree 6 is : 1376.78221860386"
```

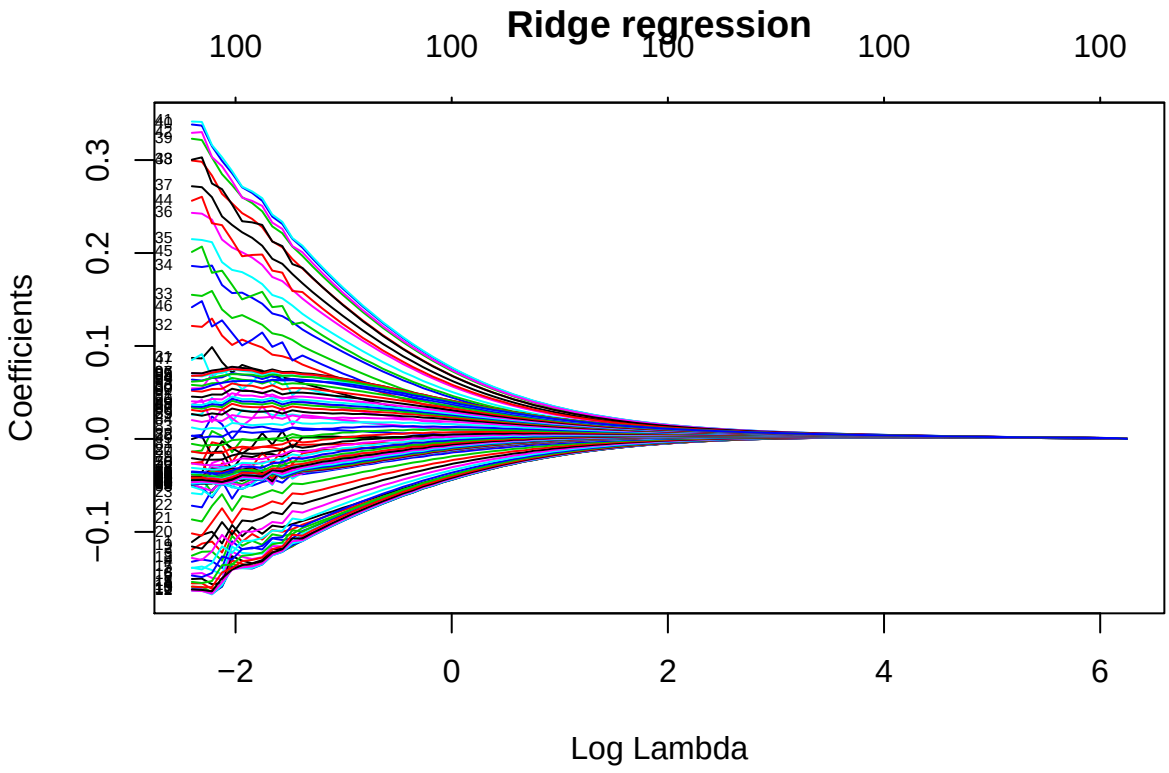
We see that the best model in terms of AIC is the linear model followed by the cubic model.

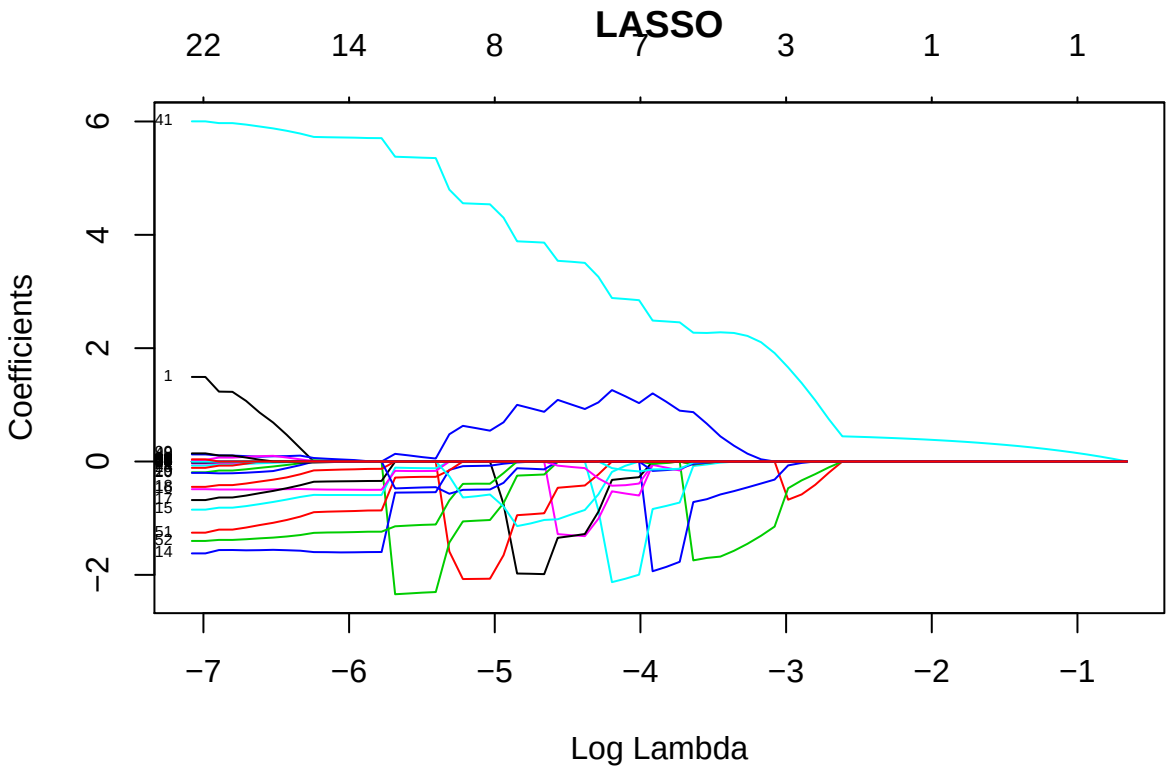
Next we perform variable selection using `stepAIC` for a linear model in which Fat is the response variable and the 100 channels in excel file `tecator.xlsx` are the features. `stepAIC` ensures that the AIC decreases in every step of the iteration.

```
[1] "During stepAIC 37 channels were removed, and 63 channels remain"
```

We are left a model with 63 channels, which is rather better than 100 channels. Probably some channels were collinear.

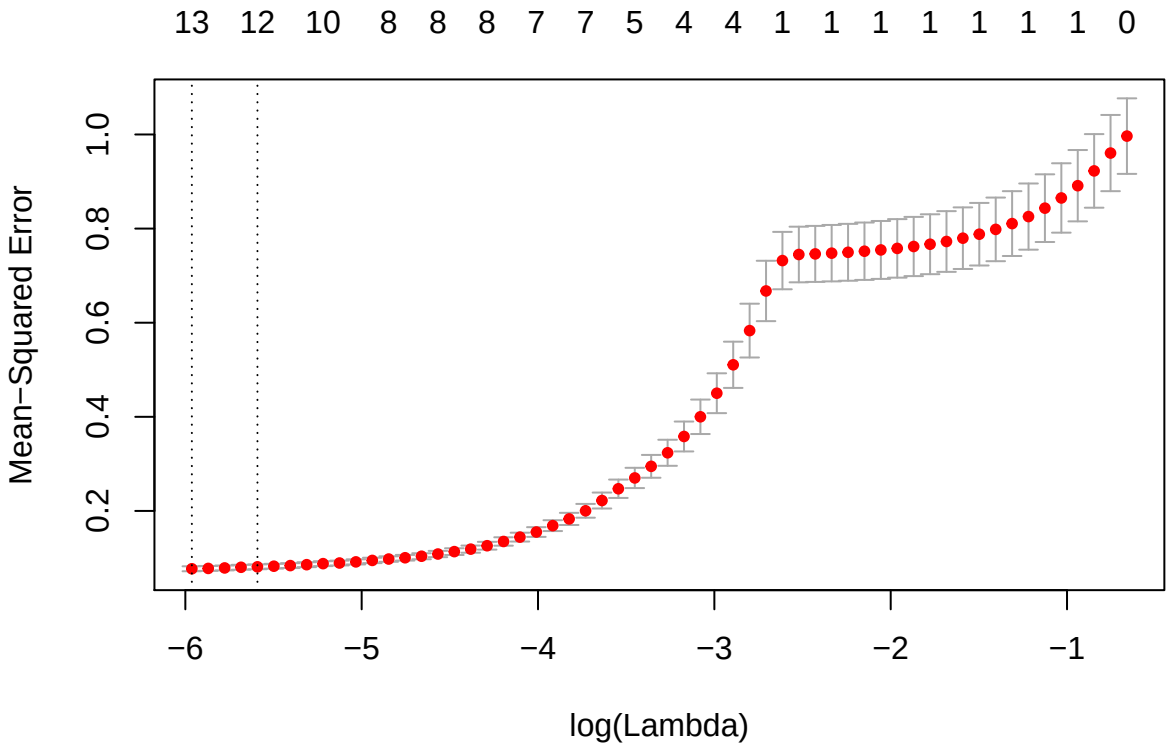
Now we fit a ridge regression model and a LASSO model (with the same features and response variables as used during the above variable selection with `stepAIC`) and plot how model coefficients depend on the logarithm of parameter Lambda.






We see that all coefficients are non-zero coefficients in the ridge regression model at all values of Log Lambda, while the LASSO has considerably fewer non-zero coefficients. We also see a funny behaviour in LASSO where it repeatedly makes one non-zero coefficient zero and simultaneously makes another zero coefficient non-zero as Log Lambda increases. It could be said that if you want a short expression for the model, pick LASSO over ridge regression.

Lastly, we will try, using cross-validation, to find the optimal LASSO model in terms of CV-score.




```
## [1] "Optimal model Lambda:  0.00257"
```

```
## [1] "At best Lambda, 14 parameters are non-zero, including intercept"
```

It seems obvious that the MSE ( proportional to CV-score) will asymptotically go to zero for even more negative Log Lambda than pictured, and the function simply picked the most negative Log Lambda out of the ones it evaluated.The optimal Log Lambda could actually be as high as -4 in my opinion, since no appreciable increase in MSE happens before then. If that is the case, then comparing with the plots above, optimal number of features cound be as low as seven.

## Appendix

### Code

```
library(MASS)

data <- longley

# I do not like picking Nfolds = 10 when nrow(X) = 16

Ridgeregression <- function(X,Y,Lambda,Nfolds){

  n <- Nfolds
```

```
  rows <- 1:nrow(X)

  partsize <- floor( nrow(X) / n)

  test_index <-list()

  set.seed(-56789)
  for(i in 1:n){
    test_index[[i]] <- sample(rows,partsize)

    rows <- setdiff(rows,test_index[[i]])
  }

  for(k in 1:length(rows)){
    test_index[[k]] <- c(test_index[[k]], rows[k])
  }

  cvscore <- 0

  for(j in 1:n){
    Xjth <- as.matrix(X[-test_index[[j]],])
    Xjthcenter <- colMeans(Xjth)
    Xjth <- Xjth - matrix(rep(Xjthcenter,nrow(Xjth)),ncol = ncol(Xjth), byrow=TRUE)

    Yjth <- as.matrix(Y[-test_index[[j]]])
    Yjthcenter <- colMeans(Yjth)
    Yjth <- Yjth - Yjthcenter
    w_ridge <- solve(t(Xjth) %*% Xjth + Lambda * diag(ncol(Xjth))) %*% t(Xjth) %*% Yjth

    jth_pred <- (as.matrix(X[test_index[[j]],]) -
                   matrix(rep(Xjthcenter,nrow(X[test_index[[j]],])),
                   ncol = ncol(Xjth), byrow=TRUE))  %*% w_ridge

    loss <- (Y[test_index[[j]]] - Yjthcenter - jth_pred)^2

    loss_sum <- sum(loss)

    cvscore <- cvscore + loss_sum
  }

  return(cvscore)
}



scores <- c()
for(lambda in seq(from=1, to=7, by=1)){
  scores <- c(scores,Ridgeregression(data[,1:6],data$Employed,lambda,10))

}
which.min(scores)
plot(scores,type="l",ylab="CV-scores",xlab=expression(lambda),
     main=c("CV-score vs",expression(lambda), "value"))
```

```r
library(XLConnect)
library(glmnet)
library(MASS)

#FROM THIS FILE LOCATION, EXCEL FILES SHOULD BE FOUND IN A SUBFOLDER IN THIS FILE LOCATION CALLED DATA
wb = loadWorkbook("data/tecator.xlsx")
data2 = readWorksheet(wb, sheet = "data", header = TRUE)

plot(data2$Protein,data2$Moisture,pch="+",main="Plot of Moisture against Protein",
     xlab = "Protein", ylab = "Moisture")

# $y_{j} = \sum_{i=0}^{n} w_{i}x_{j}^{i} + e$ , $e$ is $N(0,\sigma^2)$ distributed.

n=dim(data2)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=data2[id,]
valid=data2[-id,]

powerlist <-list()
validlist <- list()
wvalues <- list()
for(i in 1:6){
  powerlist[[i]] <- train$Protein^i
  validlist[[i]] <- valid$Protein^i
}

MSEtrain <-c()
MSEvalid <-c()


for(j in 1:6){
  jpt <- 1
  X <- rep(1,dim(train)[1])
  Xvalid <- rep(1,dim(valid)[1])
  while(jpt <= j){
  X <- cbind(X,powerlist[[jpt]])
  Xvalid <- cbind(Xvalid,validlist[[jpt]])
  jpt <- jpt + 1
  }

  #Using SVD decomp to find least squares as the conditionnumber is too high

  svdobj <- svd(X)
  #d <- diag(svdobj$d)
  z <- t(svdobj$u) %*% (train$Moisture)
  z <- z / svdobj$d
  wvalues[[j]] <- svdobj$v %*% z


  #wvalues <- solve(t(X) %*% X) %*% t(X) %*% (train$Moisture)

  trainpredicted <- X %*% wvalues[[j]]
```

```
  validpredicted <- Xvalid %*% wvalues[[j]]

  MSEtrain <- c(MSEtrain,sum(((train$Moisture) - trainpredicted)^2) / dim(train)[1])
  MSEvalid <- c(MSEvalid,sum(((valid$Moisture) - validpredicted)^2) / dim(valid)[1])
}

plot(MSEtrain,type="l",main="MSE for polynomial fit",xlab="Polynomial degree",
     ylab="MSE",ylim=c(20,50),lwd=2.5)
lines(MSEvalid,col="red",lwd=2.5)
legend("topright",c("training data","validation data"),
       lty=c(1,1),
       lwd=c(2.5,2.5),col=c("black","red"))

#AIC = -2loglik(D) + 2d(model) where d(model) is the no. of parameters. sigma2 counts as a parameter
# We make assumption that the outcomes are normally distributed where for observation j mean is
# \sum_{i=0}^{n} w_{i}x_{j}^{i} and variance is \sigma^2

#We estimate \sigma^2 by the MSE of linear model of total data set.

Xlinearforentire <- cbind(rep(1,n),data2$Protein)
predlinearforentire <- Xlinearforentire %*% wvalues[[1]]
sigma2 <- sum(((data2$Moisture) - predlinearforentire)^2) / n

AIC_calc <- function(data2,wvalues,sigma2){
  AICs <- c()

  for(j in 1:6){
    jpt <- 1
    Xentire <- rep(1,n)
    while(jpt <= j){
      Xentire <- cbind(Xentire,data2$Protein^jpt)
      jpt <- jpt + 1
    }
    entirepredicted <- Xentire %*% wvalues[[j]]
    loglik <- n * (log(2 * pi) + log(sigma2)) +
      1 / sigma2 * sum((data2$Moisture - entirepredicted)^2)
    AIC <- loglik + 2 * (j + 2)
    AICs <- c(AICs , AIC)
  }
  return(AICs)
}

AICvec <- AIC_calc(data2,wvalues,sigma2)
for(j in 1:6){
  print(paste("AIC at polynomial degree",j,"is :",AICvec[j]))
}

fat.lm <- lm(Fat ~.,data = data2[,2:102])
fat.lm2 <- stepAIC(fat.lm,trace = FALSE)
fat.lm2$anova

covariates <- scale(data2[,2:101])
response <- scale(data2[,102])
```

```r
ridgemod <- glmnet(as.matrix(covariates), response,
                   family = "gaussian", alpha = 0)
plot(ridgemod, xvar = "lambda" , label = TRUE)

lassomod <- glmnet(as.matrix(covariates), response,
                   family = "gaussian", alpha = 1)
plot(lassomod, xvar = "lambda" , label = TRUE)

#lambda search for smaller lambda is not very fruitful since the sd of cvm
#is larger than the improvement.
lassocv <- cv.glmnet(as.matrix(covariates), response, alpha =1)
plot(lassocv)
lassocv$lambda.min
coef(lassocv, s= "lambda.min")
# 14 parameters are non-zero, including inercept
```