Elliott Sperin

Bodyfat Regression Analysis

(i) Introduction

Objective statement: The purpose of this analysis is to better understand linear regression by using variable engineering and dimensionality reduction strategies to reduce overfitting and find variables that impact percent body fat (calculated using Brozek's equation).

Body fat percentage is a major topic when discussing health, fitness, and societal norms. It is easy to find opinions on the topic of body fat percentage; how to reduce it, it is healthy, etc. Understanding the effects body fat percentage has on a person's health should be studied to best know what the implications are for high body-fat percentage, what factors affect it, and how to prescribe the best path forward. Research has shown that a high body-fat percentage has costly implications on health. One study showed the implication of high-body-fat percentage on cardiometabolic risk in middle-aged, healthy, normal-weight adults and found that subjects with a high body-fat-percentage had a significantly higher prevalence of high blood pressure (men only), hyperglycemia, and dyslipidemia (Kim, 2012). Knowing the variables that affect body fat percentage by analyzing and making predictions from data is important for policies and innovation.

We will be conducting a regression analysis on an educational dataset from Johnson R. Journal of Statistics Education v.4, to see which significant predictors are used to predict body fat percentage (calculated using Brozek's equation). We will be using multiple types of regression in our analysis including Linear Regression, k-subset regression, stepwise regression, ridge regression, lasso regression, principal component analysis, and partial least squares regression. These regression techniques use dimension reduction, variable selection, regularization, and other techniques to produce the best model for the dataset. The metric of success in our analysis is mean squared error. The model with the lowest mean squared error will be best performing if it makes sense for our analysis.

(ii) Exploratory Data Analysis

The dataset includes 18 variables and 252 observations. The following 18 variables are included (Faraway, 1996).

Brozek	Percent body fat using Brozek's equation, 457/Density - 414.2
Siri	Percent body fat using Siri's equation, 495/Density - 450

density	Density (gm/cm^3)
Age	Age (yrs)
Weight	Weight (lbs)
Height	Height (inches)
Adipos	Adiposity index = Weight/Height^2 (kg/m^2)
Free	Fat Free Weight = (1 - fraction of body fat) * Weight, using Brozek's formula (lbs)
Neck	Neck circumference (cm)
Chest	Chest circumference (cm)
Abdom	Abdomen circumference (cm) at the umbilicus and level with the iliac crest
Hip	Hip circumference (cm)
Thigh	Thigh circumference (cm)
Knee	Knee circumference (cm)
Ankle	Ankle circumference (cm)
Biceps	Extended biceps circumference (cm)
Forearm	Forearm circumference (cm)
Wrist	Wrist circumference (cm) distal to the styloid processes

There are no NA values in the dataset.

Brozek is the dependent variables and by instruction all other variables will be used as the predictor variables.

Reading the variable descriptions, it is clear to see some variables suffer from high collinearity multicollinearity. Siri will be highly correlated with Brozek as it is another body fat calculator. Density will also be highly correlated as it is used to directly calculate Brozek. To analyze this further I created a linear regression model with Brozek as the response and all other variables as the predictor to calculate the VIF of each predictor variable. I found the following results.

Variable	siri	density	age	weight	height	adipos	free	neck
VIF	74.876	43.859	2.2917	97.611	2.285	17.87	56.181	4.5298
chest	abdom	hip	thigh	knee	ankle	biceps	forearm	wrist
11.34	19.606	15.376	8.417	4.8702	1.988	3.7698	2.3008	3.5722

Vif calculates the severity of multicollinearity by assessing the increasing of variance due to collinearity. VIF below 5 is low collinearity, VIF between 5 and 10 is moderate collinearity, and VIF above 10 is high collinearity. Age, height, neck, thigh, knee, ankle, biceps, forearm, and wrist have moderate to low levels of multicollinearity, while all other variables show high levels. It is easy to see this strong correlation in the scatterplots

below. Siri, Density, and Abdom (*Figure 1*) can easily be seen showing strong correlation with Brozek while Free, Height, and Ankle (*Figure 2*) show weak correlation with Brozek. The modelling process will remove dimensionality and engineer the best set of variables to use.

Figure 1: Strongly Correlated Variables

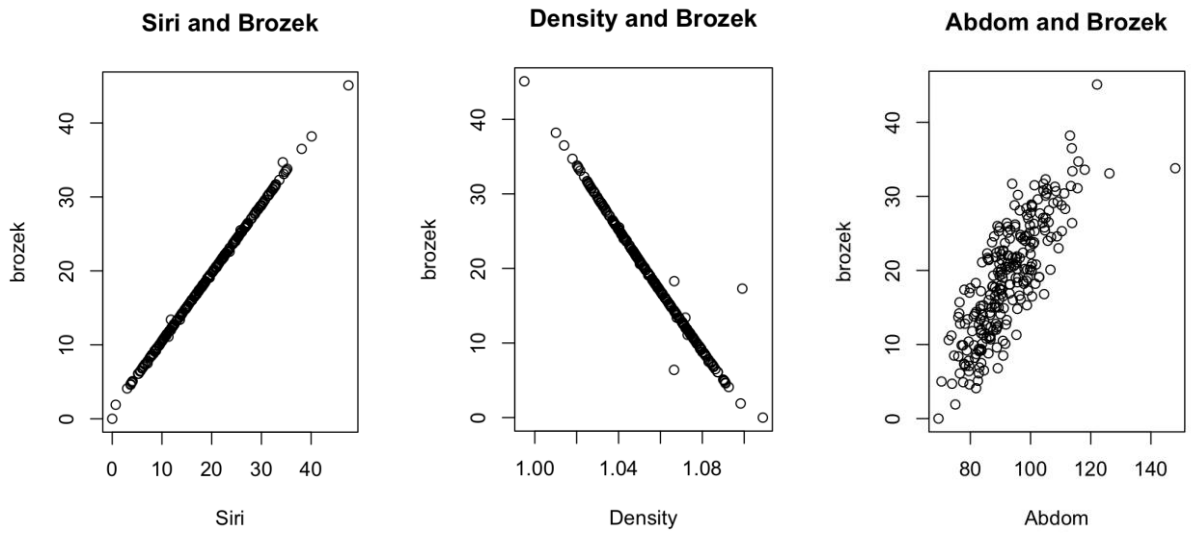
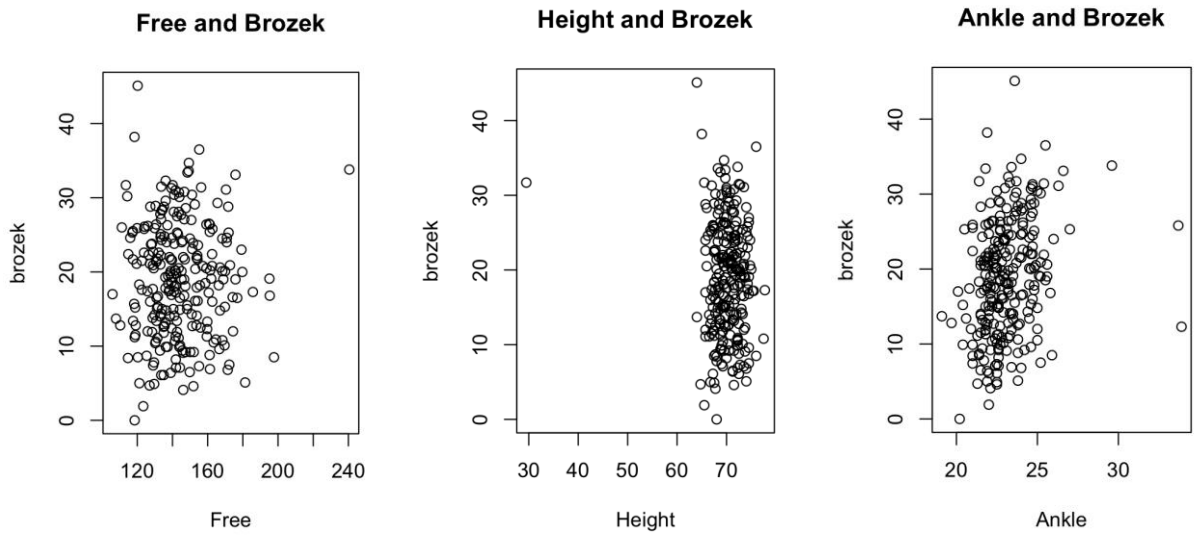
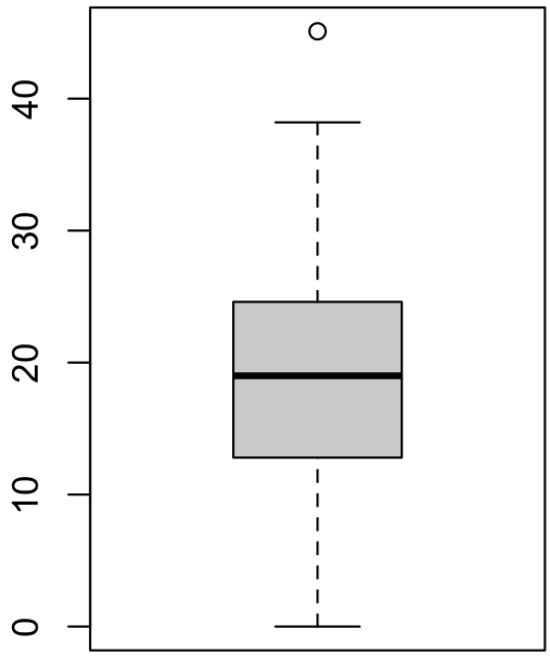


Figure 2: Weakly Correlated Variables



Outliers can impede the analysis if not properly processed. Brozek has one point that could be considered an outlier at observation 216 with value of 45.1. Please see the Box and whisker plot below (*Figure 3)*. A Grubbs test indicates that this point is an outlier with a p-value of .08146. The highest value could be considered an outlier, but since we have few observations, it is more beneficial to keep the observation included. The p-value of .08146 means the null hypothesis is rejected if the threshold is .05, but it is accepted if null is .1. For these two reasons the observation will remain.

Figure 3: Box and Whisker Plot of Brozek

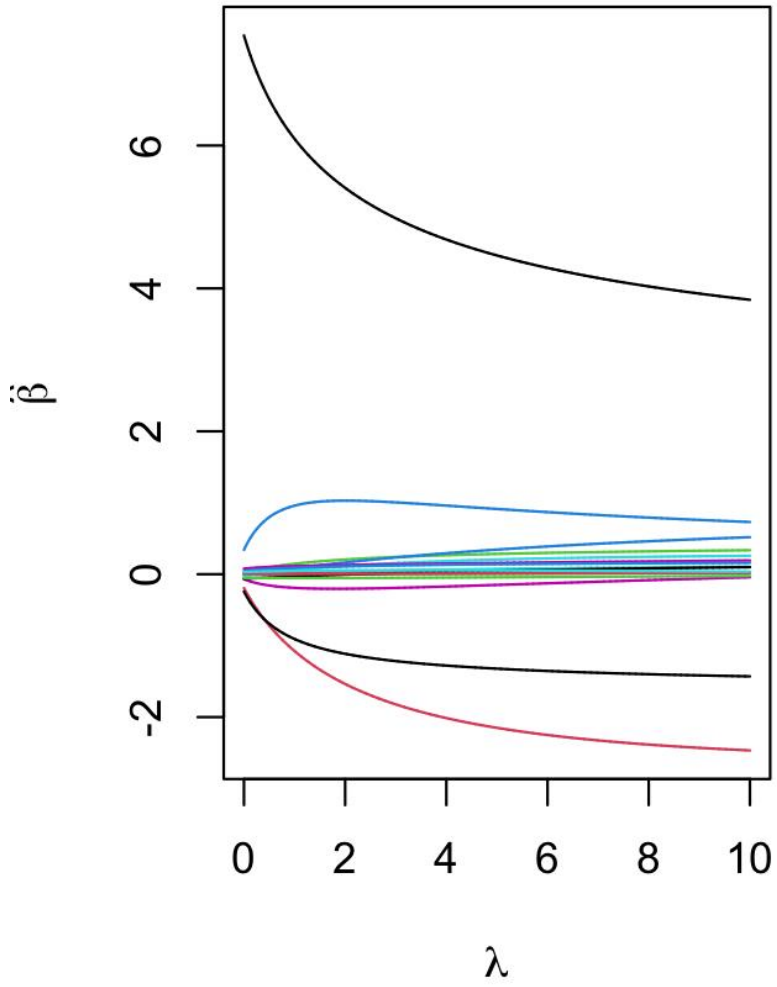


(iii) Methods

The modeling methodology is broken into two parts. 1, An initial modeling using a split test and training dataset from the original data. 2, Using Monte Carlo Cross-validation on the entire dataset.

1. An initial modeling using a split test and training dataset from the original data.
    a. Split the dataset into training and test datasets from the original data. by choosing 25 rows to be used in the test dataset and the remaining to be used for training.
    b. Model 1: Create a linear regression model from the training data using Brozek as the response variable and all other variables as the predictor variables. I then calculated the mean squared error when making a prediction on the training data and on the testing data.
    c. Model 2: Create the best subset linear regression model using the best 5 predictor variables. Use the leaps model in R to go through 100 iterations of combinations of subsets to find what is the best when k = 5. The process found siri, density, thigh, knee, and forearm to be the best subset. I then calculate the mean squared error when making a prediction on the training data and on the testing data.
    d. Model 3: Use stepwise regression to find which variables are best. I compared results for using both forward and backwards stepwise regression. Forward regression kept the following variables: siri, density forearm, biceps, thigh, knee, and wrist. Backwards regression kept the following variables: siri, density, weight, free, hip, thigh, knee, biceps, and forearm. Backwards regression produced a slightly better model with an MSE of 64.35603 vs. 64.37349. For this reason, I chose backwards regression over forward regression.
    e. Model 4: Use Ridge regression to find which variables are best. I used the MASS library to test lambda values from 0 to 100 to see which produced the best ridge regression model. Figure 4 shows the impact lambda has on different values of beta. Lambda value of 4 produces the least amount of error. I calculated the value of the coefficient when lambda = 4 and scaled the data back into the original raw data scale. The model reduces all variables to between -.1 and .1 other than siri, weight, and free. I then Calculated the mean squared error when making a prediction on the training data and on the testing data.

Figure 4: Ridge regression lambda to beta values

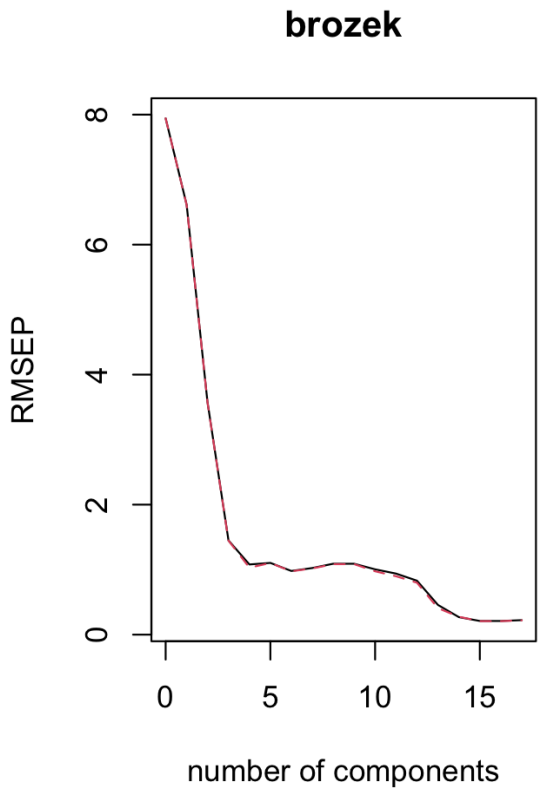


f. Model 5: Lasso Regression. Use Lars library in R to create the Lasso regression model. A lambda value of 9 reduces Mellon's Cp criterion the best, and thus this value is chosen to pick the coefficients. After converting the coefficient into the original raw data, siri and density had significant coefficients values, density, thigh, knee, biceps, forearms, wrist, and age were kept at with small coefficients, and all other variables were reduced to zero. I then calculated the mean squared error when making a prediction on the training data and on the testing data.

g. Model 6: Principal Component Analysis. Use PLS in R to create the PCA model. The analysis found that using all 17 principal components has the lowest root mean squared error. See Figure 5. 17 is the full dimension of the original data and thus the model reduces into its original form. I then

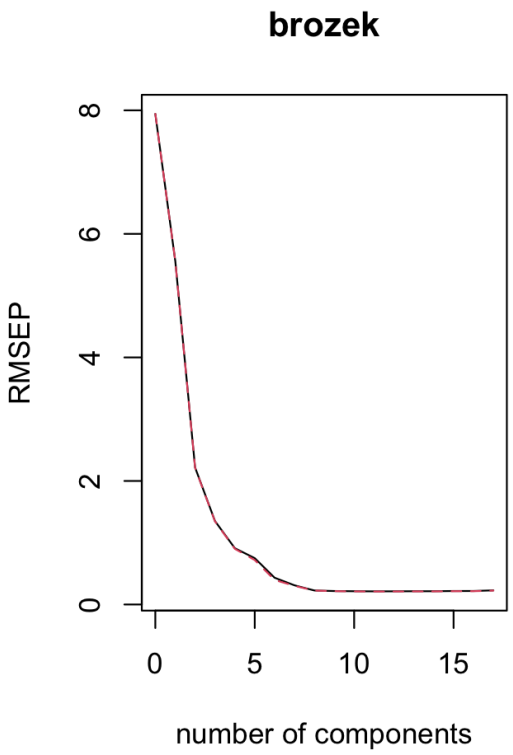
calculate the mean squared error when making a prediction on the training data and on the testing data.

Figure 5: Number of PC's and RMSEP



h. Model 7: Partial Least Square method. Use PLS library in R to calculate which number of components of partial least squares is optimal to reduce RMSEP. The optimal number of components is 17 and thus this model reduces to the original dataset. I then calculated the mean squared error when making a prediction on the training data and on the testing data.

Figure 5: Number of PC's of PLS and RMSEP



2. Using Monte Carlo Cross-validation on the entire dataset.
    a. I chose to run the cross validation for 100 iterations.
    b. I chose a random sample of 25 for the test data to be chosen for each iteration.
    c. I then used the models as constructed above to calculate the average mean squared error and variance across all 100 iterations for each model.

(iv) Results

Part 1 results: Training and Testing Datasets

Model	Linear Reg	Subset	Stepwise	Ridge	Lasso	PCA	PLS
MSE of Training data	0.029	0.031	0.029	0.029	0.031	0.029	0.029
MSE of Testing data	0.009	0.003	0.005	0.009	0.003	0.009	0.009

Part 2 results: Monte Carlo Dataset

Model	Linear Reg	Subset	Stepwise	Ridge	Lasso	PCA	PLS
MSE of CV	88.1790	88.3303	88.1760	11.0275	0.0115	0.0129	0.0130
Vars of CV	4.29E+02	4.32E+02	4.29E+02	9.05E+02	1.04E-03	1.22E-03	1.25E-03

(v) Findings

The best performing model for this analysis is the Lasso model as it produced the lowest average mean squared error of 0.0115 and variance of 1.04E-03 in the cross-validation dataset. PCA and PLS produced similar results with differences of less than .01 MSE between the three models. It's clear that the regression model, subset model, stepwise, and ridge model's do not perform as well. These models reduce error by eliminating entire variables, while LASSO, PCA, and PLS reduce error by reducing dimensionality or coefficient values. The results of this analysis show that this dataset suffers from high dimensionality and collinearity. This is supported by the findings in the EDA section that showed a strong correlation between key variables in our analysis.

The MSE results between part 1 and part 2 are drastically different. In part 1, the subset model, Lasso, stepwise models were the best performing, and all the results were within .005 differences of each other. The differences of results are found in the dataset usage. Using only testing and training dataset did not account for all the trends in the data. When using cross validation, we used every observation to best represent the data that produced results closer to the real-world data. The original dataset is also relatively small with only 252 observations, making each observation valuable for our analysis. Leaving out just 25 observations in our training dataset proved to have costly results.

Overall, this regression analysis showed the importance of proper cross validation to best account for all patterns in the data and Demensional/variable reduction techniques to eliminate highly correlated data.

Works Cited:

Kim, J. Y., Han, S.-H., & Yang, B.-M. (2013). Implication of high-body-fat percentage on cardiometabolic risk in middle-aged, healthy, normal-weight adults. Obesity, 21(8), 1571–1577. https://doi.org/10.1002/oby.20020

fat: Percentage of Body Fat and Body Measurements in faraway: Functions and Datasets for Books by Julian Faraway (1996). (n.d.). Rdrr.io. Retrieved February 4, 2024, from https://rdrr.io/cran/faraway/man/fat.html

Appendix:

```
### Read the data

library(dplyr)

library(ggplot2)

library(reshape2)

library(corrplot)

library(outliers)

library(DAAG)

### Suppose you save the data file ‘‘fat.csv" in the folder ‘‘C://Temp" of your laptop,

fat <- read.table(file = "fat.csv", sep=",", header=TRUE);

### Split the data as in Part (a)

n = dim(fat)[1]; ### total number of observations

n1 = round(n/10); ### number of observations randomly selected for testing data

#Check for NAs
```

```
summary(fat)

#no NAs

#multicorrlineary

corr_mat <- round(cor(fat),2);

corrplot(corr_mat, method="shade")



x1 = fat$free

x2 = fat$height

x3 = fat$ankle



x4 = fat$siri

x5 = fat$density

x6 = fat$abdom
```

y = fat$brozek

EDA_lm <- lm(brozek ~., data = fat)

vif(EDA_lm)

#VIF shows that age, height, neck, thigh, knee, ankle, biceps, forearm, and wrist

#show moderate to low levels of multicolinnarity, while all other variables are show high levels.

#The modelling process will remove the highly correlated variables.

#weak corralated maps

```
plot(x1, y, main="Free and Brozek", xlab="Free", ylab="brozek")

plot(x2, y, main="Height and Brozek", xlab="Height", ylab="brozek")

plot(x3, y, main="Ankle and Brozek", xlab="Ankle", ylab="brozek")




#strongest correlated

plot(x4, y, main="Siri and Brozek", xlab="Siri", ylab="brozek")

plot(x5, y, main="Density and Brozek", xlab="Density", ylab="brozek")

plot(x6, y, main="Abdom and Brozek", xlab="Abdom", ylab="brozek")



#Search for outliers

boxplot(y)

max(y)
```

grubbs.test(y)

#The highest value could be considered an outlier.

#Since we have few observations it is more beneficial to keep the observation included.

#the p-value of .08146 means the null hypothesis is rejected if the threshold is .05, but it is accepted if null is .1.

#For these two reasons the observation will remain.

To fix our ideas, let the following 25 rows of data as the testing subset:

flag = c(1, 21, 22, 57, 70, 88, 91, 94, 121, 127, 149, 151, 159, 162,

```
    164, 177, 179, 194, 206, 214, 215, 221, 240, 241, 243);

fat1train = fat[-flag,];

fat1test = fat[flag,];




###In Part (b)-(d), Please see the R code for linear regression at Canvas.

ytrue   <- fat1test$brozek

#Linear Regression Model

MSEtrain <- NULL;

MSEtest  <- NULL;



lm_model <- lm(brozek ~., data = fat1train)

summary(lm_model)
```

```
## Model 1: Training error

MSEmod1train <-   mean( (resid(lm_model) )^2);

MSEtrain <- c(MSEtrain, MSEmod1train);

# Model 1: testing error

pred1a <- predict(lm_model, fat1test[,-1]);

MSEmod1test <- mean((pred1a - ytrue)^2);

MSEmod1test

MSEtest <- c(MSEtest, MSEmod1test);
```

```
#s <- fat1train$siri+fat1train$weight+ fat1train$free+ fat1train$knee +fat1train$thigh



#Model 2: Best subset k = 5

library(leaps)

leaps <- regsubsets(brozek ~ ., data= fat1train, nbest= 100, really.big= TRUE);



## Record useful information from the output

models <- summary(leaps)$which;

models.size <- as.numeric(attr(models, "dimnames")[[1]]);

models.rss <- summary(leaps)$rss;



## 2A:  The following are to show the plots of all subset models

##   and the best subset model for each subset size k
```

```
plot(models.size, models.rss);

## find the smallest RSS values for each subset size

models.best.rss <- tapply(models.rss, models.size, min);

## Also add the results for the only intercept model

model0 <- lm( brozek ~ 1, data = fat1train);

models.best.rss <- c( sum(resid(model0)^2), models.best.rss);

## plot all RSS for all subset models and highlight the smallest values

plot( 0:8, models.best.rss, type = "b", col= "red", xlab="Subset Size k", ylab="Residual
Sum-of-Square")

points(models.size, models.rss)




# 2B: What is the best subset with k=5

op2 <- which(models.size == 5);

flag2 <- op2[which.min(models.rss[op2])];
```

```
flag2

## we can auto-find the best subset with k=5

##   this way will be useful when doing cross-validation

mod2selectedmodel <- models[flag2,];

mod2selectedmodel

mod2Xname <- paste(names(mod2selectedmodel)[mod2selectedmodel][-1],
collapse="+");

mod2Xname

mod2form <- paste ("brozek ~", mod2Xname);



## To auto-fit the best subset model with k=5 to the data

model2 <- lm( as.formula(mod2form), data= fat1train);

# Model 2: training error
```

```
MSEmod2train <- mean(resid(model2)^2);

## save this training error to the overall training error vector

MSEtrain <- c(MSEtrain, MSEmod2train);

MSEtrain

## Model 2:  testing error

pred2 <- predict(model2, fat1test[,-1]);

MSEmod2test <-   mean((pred2 - ytrue)^2);

MSEtest <- c(MSEtest, MSEmod2test);

MSEtest



#Model 3 Step wise regression

intercept_only <- lm(brozek ~ 1, data=fat1train)
```

```
#Forwards

forward <- step(intercept_only, scope = formula(lm_model), direction='forward',
trace=0)

forward$anova

forward$coefficients




MSforwardstrain <- mean(resid(forward)^2);

pred3 <- predict(forward, fat1test[,-1]);

MSforwardstest <-   mean((pred3 - ytrue)^2);

MSforwardstest

#Backwards

backward <- step(lm_model, scope = formula(lm_model), direction='backward',
trace=0)

backward$anova
```

```
backward$coefficients



MSbackwardstrain <- mean(resid(backward)^2);

MSEmod3train <- min(c(MSbackwardstrain, MSforwardstrain))

pred3 <- predict(backward, fat1test[,-1]);

MSbackwardstest <-   mean((pred3 - ytrue)^2);

MSEmod3test <- min(c(MSbackwardstest, MSforwardstest))

MSbackwardstest



MSEtrain <- c(MSEtrain, MSEmod3train);

MSEtrain

MSEtest <- c(MSEtest, MSEmod3test);

MSEtrain;
```

```
#Rdige Regression

library(MASS)

ridge <- lm.ridge( brozek ~ ., data = fat1train, lambda= seq(0,10,0.001));

plot(ridge)

matplot(ridge$lambda, t(ridge$coef), type="l", lty=1, xlab=expression(lambda),
ylab=expression(hat(beta)))

indexopt <-  which.min(ridge$GCV);

indexopt

ridge$coef[,indexopt]

## However, this coefficeints are for the the scaled/normalized data

##     instead of original raw data
```

```
## We need to transfer to the original data

## Y = X \beta + \epsilon, and find the estimated \beta value

##       for this "optimal" Ridge Regression Model

## For the estimated \beta, we need to sparate \beta_0 (intercept) with other \beta's

ridge.coeffs = ridge$coef[,indexopt]/ ridge$scales;

intercept = -sum( ridge.coeffs  * colMeans(fat1train[,-1] )  )+ mean(fat1train[,1]);

## If you want to see the coefficients estimated from the Ridge Regression

##   on the original data scale

c(intercept, ridge.coeffs);




## Model 4 (Ridge): training errors

yhat4train <- as.matrix( fat1train[,-1]) %*% as.vector(ridge.coeffs) + intercept;

MSEmod4train <- mean((yhat4train - fat1train$brozek)^2);
```

```
MSEtrain <- c(MSEtrain, MSEmod4train);

MSEtrain

## Model 4 (Ridge):  testing errors in the subset "test"

pred4test <- as.matrix( fat1test[,-1]) %*% as.vector(ridge.coeffs) + intercept

MSEmod4test <-  mean((pred4test - ytrue)^2);

MSEtest <- c(MSEtest, MSEmod4test);

MSEtest



#Model 5: Lasso

library(lars)

lars <- lars( as.matrix(fat1train[,-1]), fat1train[,1], type= "lasso", trace= TRUE);

## 5A: some useful plots for LASSO for all penalty parameters \lambda

plot(lars)
```

```
Cp1  <- summary(lars)$Cp



index1 <- which.min(Cp1)

index1

lasso.coeffs <- lars$beta[index1,]



lasso.lambda <- lars$lambda[index1]



LASSOintercept = mean(fat1train[,1]) -sum( lars$beta[index1,]  * colMeans(fat1train[,-
1] ));

c(LASSOintercept, lars$beta[index1,])s
```

```
## Model 5:  training error for lasso

##

pred5train  <- predict(lars, as.matrix(fat1train[,-1]), s=lasso.lambda, type="fit",
mode="lambda");

yhat5train <- pred5train$fit;

MSEmod5train <- mean((yhat5train - fat1train$brozek)^2);

MSEtrain <- c(MSEtrain, MSEmod5train);

MSEtrain

## Model 5:  training error for lasso

pred5test <- predict(lars, as.matrix(fat1test[,-1]), s=lasso.lambda, type="fit",
mode="lambda");
```

```
yhat5test <- pred5test$fit;

MSEmod5test <- mean( (yhat5test - fat1test$brozek)^2);

MSEtest <- c(MSEtest, MSEmod5test);

MSEtest;



#Model 6 PCA:

library(pls)

pca <- pcr(brozek~., data=fat1train, validation="CV");

validationplot(pca);

summary(pca);



ncompopt <- which.min(pca$validation$adj);

pca$validation$adj
```

```
ypred6train <- predict(pca, ncomp = ncompopt, newdata = fat1train[-1]);
```

```
## 6B(iv) Training Error with the optimal choice of PCs
```

```
MSEmod6train <- mean( (ypred6train - fat1train$brozek)^2);
```

```
MSEtrain <- c(MSEtrain, MSEmod6train);
```

```
MSEtrain;
```

```
## 6B(v) Testing Error with the optimal choice of PCs
```

```
ypred6test <- predict(pca, ncomp = ncompopt, newdata = fat1test[,-1]);
```

```
MSEmod6test <- mean( (ypred6test - fat1test$brozek)^2);
```

```
MSEtest <- c(MSEtest, MSEmod6test);
```

```
MSEtest;
```

```
MSEmod6test
```

```
### Model 7. Partial Least Squares (PLS) Regression

###

###  The idea is the same as the PCR and can be done by "pls" package

###  You need to call the fuction "plsr"  if you the code standalone

#  library(pls)

pls <- plsr(brozek ~ ., data = fat1train, validation="CV");



### 7(i) auto-select the optimal # of components of PLS

## choose the optimal # of components

mod7ncompopt <- which.min(pls$validation$adj);

## The opt # of components, it turns out to be 8 for this dataset,
```

```
##      and thus PLS also reduces to the full model!!!

validationplot(pls)

# 7(ii) Training Error with the optimal choice of "mod7ncompopt"

# note that the prediction is from "pls" with "mod7ncompopt"

ypred7train <- predict(pls, ncomp = mod7ncompopt, newdata = fat1train[,-1]);

MSEmod7train <- mean( (ypred7train - fat1train$brozek)^2);

MSEtrain <- c(MSEtrain, MSEmod7train);




## 7(iii) Testing Error with the optimal choice of "mod7ncompopt"

ypred7test <- predict(pls, ncomp = mod7ncompopt, newdata = fat1test[,-1]);

MSEmod7test <- mean( (ypred7test - fat1test$brozek)^2);

MSEtest <- c(MSEtest, MSEmod7test);
```

Check your answers

MSEtrain

Training errors of these 7 models/methods

MSEtest

Part (e): the following R code might be useful, and feel free to modify it.

save the TE values for all models in all $B=100$ loops

B= 100; ### number of loops

TEALL = NULL; ### Final TE values

set.seed(7406); ### You might want to set the seed for randomization

for (b in 1:B){

```
### randomly select 25 observations as testing data in each loop

flag <- sort(sample(1:n, n1));

fattrain <- fat[-flag,];

fattest <- fat[flag,];

### you can write your own R code here to first fit each model to "fattrain"

### then get the testing error (TE) values on the testing data "fattest"

### Suppose that you save the TE values for these five models as

### te1, te2, te3, te4, te5, te6, te7, respectively, within this loop

### Then you can save these 5 Testing Error values by using the R code

###

lm_model <- lm(brozek ~., data = fattrain)



# Model 1: testing error
```

```
pred1a <- predict(lm_model, fattest[,-1]);

te1 <- mean((pred1a - ytrue)^2);




#Model 2: Best subset k = 5

library(leaps)

leaps <- regsubsets(brozek ~ ., data= fattrain, nbest= 100, really.big= TRUE);




## Record useful information from the output

models <- summary(leaps)$which;

models.size <- as.numeric(attr(models, "dimnames")[[1]]);

models.rss <- summary(leaps)$rss;
```

```
models.best.rss <- tapply(models.rss, models.size, min);

## Also add the results for the only intercept model

model0 <- lm( brozek ~ 1, data = fattrain);

models.best.rss <- c( sum(resid(model0)^2), models.best.rss);




# 2B: What is the best subset with k=5

op2 <- which(models.size == 5);

flag2 <- op2[which.min(models.rss[op2])];

## we can auto-find the best subset with k=5




mod2selectedmodel <- models[flag2,];

mod2Xname <- paste(names(mod2selectedmodel)[mod2selectedmodel][-1],
collapse="+");

mod2form <- paste ("brozek ~", mod2Xname);
```

```
## To auto-fit the best subset model with k=5 to the data

model2 <- lm( as.formula(mod2form), data= fattrain);



## Model 2:  testing error

pred2 <- predict(model2, fattest[,-1]);

te2 <-   mean((pred2 - ytrue)^2);



#Model 3 Step wise regression

intercept_only <- lm(brozek ~ 1, data=fattrain)



#Forwards

forward <- step(intercept_only, scope = formula(lm_model), direction='forward',
trace=0)
```

```
pred3 <- predict(forward, fattest[,-1]);

MSforwardstest <- mean((pred3 - ytrue)^2);



#Backwards

backward <- step(lm_model, scope = formula(lm_model), direction='backward',
trace=0)



pred3 <- predict(backward, fattest[,-1])

MSbackwardstest <-  mean((pred3 - ytrue)^2)

te3 <- min(c(MSbackwardstest, MSforwardstest))



#Rdige Regression

library(MASS)
```

```
ridge <- lm.ridge( brozek ~ ., data = fattrain, lambda= seq(0,100,0.001));

indexopt <-  which.min(ridge$GCV);



ridge.coeffs = ridge$coef[,indexopt]/ ridge$scales;

intercept = -sum( ridge.coeffs  * colMeans(fattrain[,-1] )  )+ mean(fattrain[,1]);



## Model 4 (Ridge):  testing errors in the subset "test"

pred4test <- as.matrix( fattest[,-1]) %*% as.vector(ridge.coeffs) + intercept

MSEmod4test <-  mean((pred4test - ytrue)^2);

te4 <- c(MSEtest, MSEmod4test);



#Model 5: Lasso
```

```
library(lars)

lars <- lars( as.matrix(fattrain[,-1]), fattrain[,1], type= "lasso", trace= TRUE);

## 5A: some useful plots for LASSO for all penalty parameters \lambda




Cp1  <- summary(lars)$Cp

index1 <- which.min(Cp1)

lasso.coeffs <- lars$beta[index1,]

lasso.lambda <- lars$lambda[index1]

LASSOintercept = mean(fattrain[,1]) -sum( lars$beta[index1,]  * colMeans(fattrain[,-
1] ));

c(LASSOintercept, lars$beta[index1,])




## Model 5:  training error for lasso
```

```
 pred5test <- predict(lars, as.matrix(fattest[,-1]), s=lasso.lambda, type="fit",
mode="lambda");


 yhat5test <- pred5test$fit;


 MSEmod5test <- mean( (yhat5test - fattest$brozek)^2);


 te5 <- c(MSEtest, MSEmod5test);









 #Model 6 PCA:


 library(pls)


 pca <- pcr(brozek~., data=fattrain, validation="CV");




 ncompopt <- which.min(pca$validation$adj);


 pca$validation$adj
```

```
ypred6train <- predict(pca, ncomp = ncompopt, newdata = fattrain[-1]);
```

```
## 6B(iv) Training Error with the optimal choice of PCs
```

```
MSEmod6train <- mean( (ypred6train - fattrain$brozek)^2);
```

```
MSEtrain <- c(MSEtrain, MSEmod6train);
```

```
MSEtrain;
```

```
## 6B(v) Testing Error with the optimal choice of PCs
```

```
ypred6test <- predict(pca, ncomp = ncompopt, newdata = fattest[,-1]);
```

```
MSEmod6test <- mean( (ypred6test - fattest$brozek)^2);
```

```
te6 <- c(MSEtest, MSEmod6test);
```

```
### Model 7. Partial Least Squares (PLS) Regression
```

```
pls <- plsr(brozek ~ ., data = fattrain, validation="CV");
```

```
 mod7ncompopt <- which.min(pls$validation$adj);



 ## 7(iii) Testing Error with the optimal choice of "mod7ncompopt"

 ypred7test <- predict(pls, ncomp = mod7ncompopt, newdata = fattest[,-1]);

 MSEmod7test <- mean( (ypred7test - fattest$brozek)^2);

 te7 <- c(MSEtest, MSEmod7test);



 TEALL = rbind( TEALL, cbind(te1, te2, te3, te4, te5, te6, te7) );

}

dim(TEALL); ### This should be a Bx7 matrices

### if you want, you can change the column name of TEALL

colnames(TEALL) <- c("Regression", "k=5", "Stepwise", "Ridge", "Lasso", "PCA", "LS");
```

You can report the sample mean and sample variances for the seven models

apply(TEALL, 2, mean)

apply(TEALL, 2, var)

END