# ISYE6414 OAN HW5 self review solutions

*Mar 31, 2019*

In this exercise you will analyze the "fat" data in the Faraway package. Your task is to predict the percentage of body fat based on different body measurements. Age, weight, height, and 10 body circumference measurements are recorded for 252 men. Each man's percentage of body fat was accurately estimated by an underwater weighing technique.

Because of the fairly large number of variables to be included in the model, the model can be difficult to interpret. Moreover, it may be of interest to identify few important variables that could predict body fat better than others. Thus you will apply variable selection approaches to reduce the model to a smaller set of predicting variables while keeping its explanatory power at a similar level as the full model.

Attribute Information:

A data frame with 252 observations on the following 18 variables.

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/cc)

age Age (yrs)

weight Weight (lbs)

height Height (inches)

adipos Adiposity index = Weight/Heightˆ2 (kg/mˆ2)

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

Load the data using the following command. Remove the "brozek" and "density" columns. Set a seed (set.seed(123)) and randomly split your data into 80% training and 20% testing sets. Double check the dimensions of the split datasets. Do all analyses on the training data:

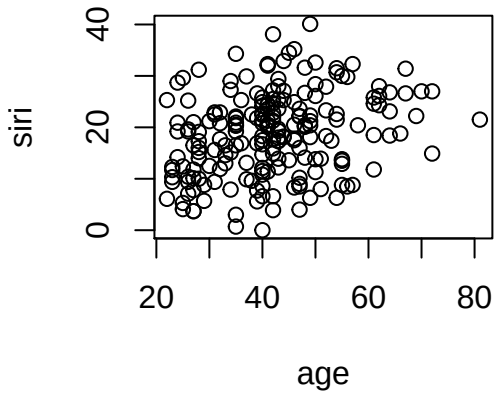
data=faraway::fat

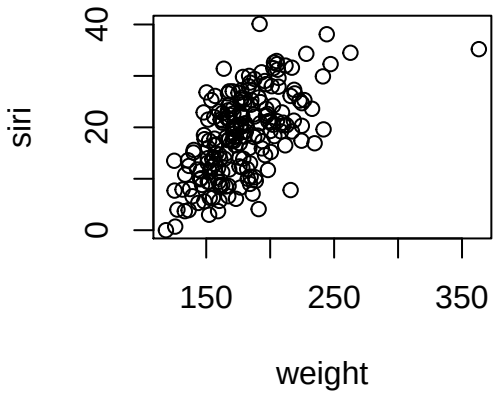
**Question 1: Exploratory Data Analysis**

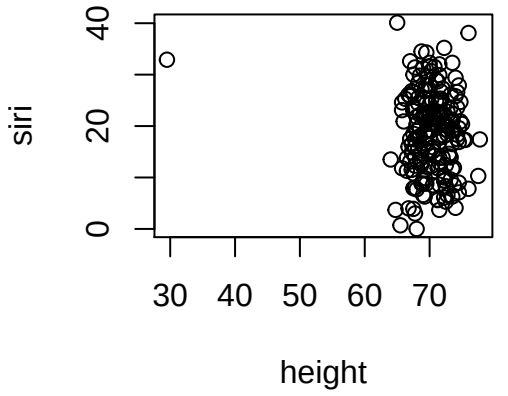
a) Begin by doing some exploratory analysis. Plot scatterplots to examine the relationships between all the explanatory variables and the response siri:

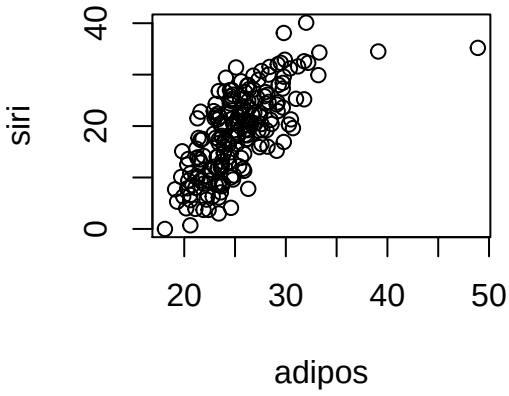
Do you observe any relationship of siri with any of the predictors? Do you visually observe any outlier or high leverage point in any of the plots? Briefly note down your observations

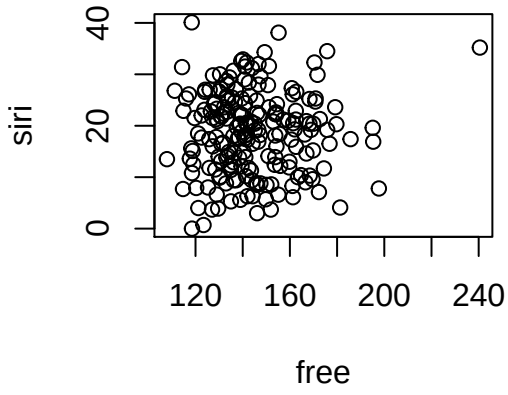
```
data=faraway::fat
data=data[,-c(1,3)]

# Splitting data
set.seed(123)
train=data[sample(0.8*nrow(data)),]
test=data[!(rownames(data) %in% rownames(train)),]

#Plotting siri vs all predictors
plot(siri~.,train)
```

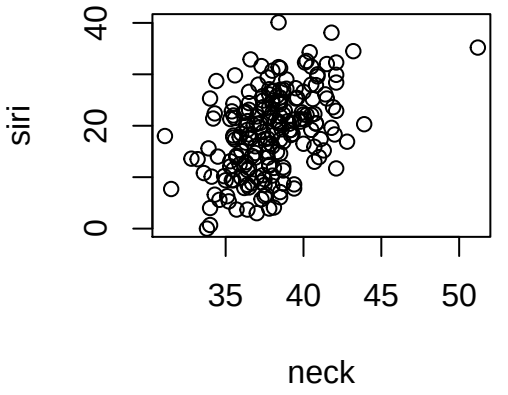


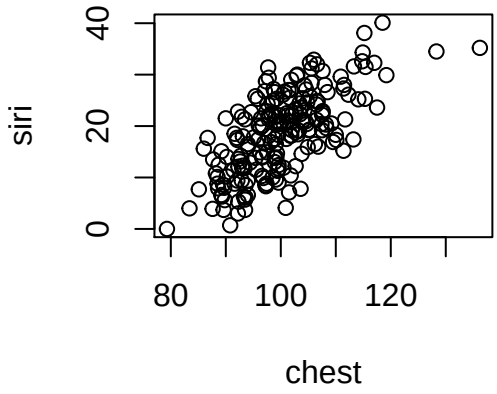



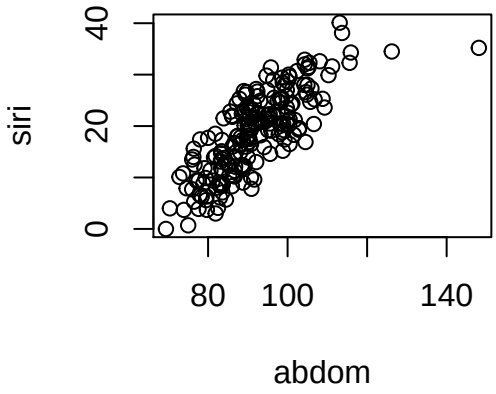

siri
0
20
40
30
40
50
60
70
height

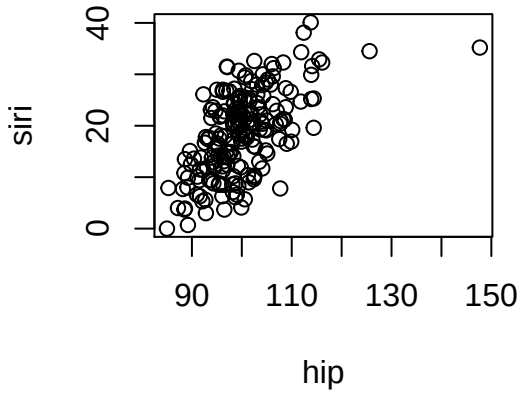

siri
0
20
40
20
30
40
50
adipos

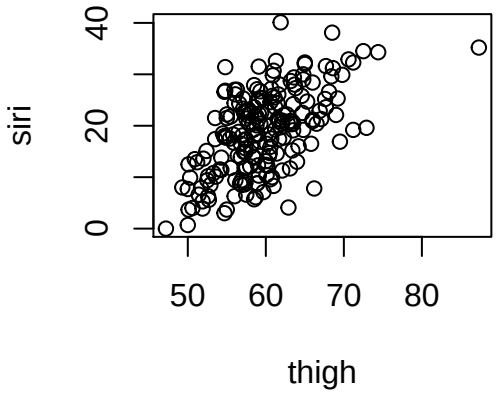

siri
0
20
40
120
160
200
240
free

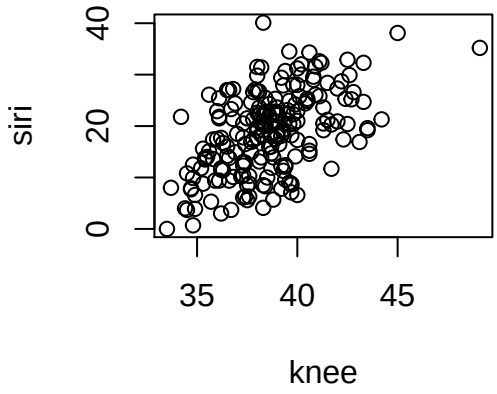

siri
0
20
40
35
40
45
50
neck

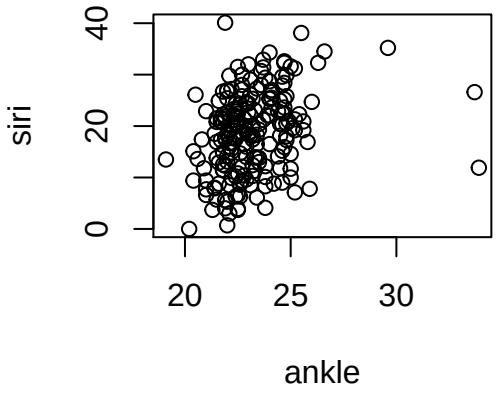

siri
0
20
40
80
100
120
chest

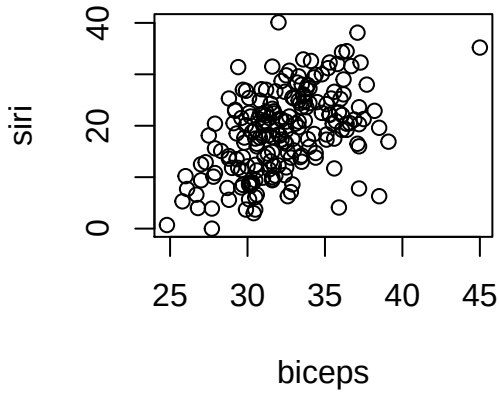

siri
0
20
40
80
100
140
abdom

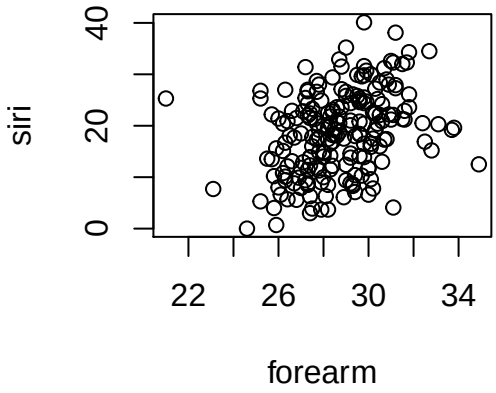

siri
0
20
40
90
110
130
150
hip

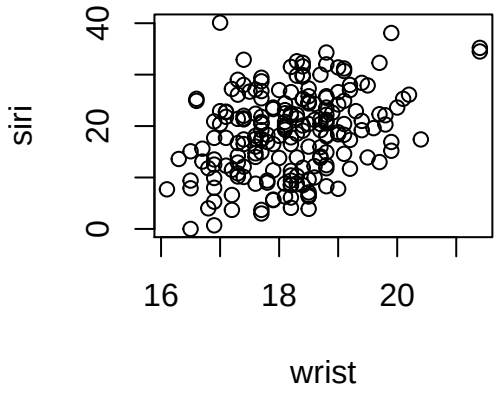

siri
0
20
40
50
60
70
80
thigh


siri
0
20
40
35
40
45
knee


siri
0
20
40
20
25
30
ankle


siri
0
20
40
25
30
35
40
45
biceps


siri
0
20
40
22
26
30
34
forearm




**Obervations**

From the plots, there is a clear linear relationship of body fat with many of the predictors especially abdom, thigh, chest and adipos.

There are a few high leverage points in some plots. For example there are a couple of points with abdom values greater than 120. Similar points can be observed in other plots

b) Plot a correlation matrix of all variables vs all other variables:

library(corrplot)

corrplot(cor(train),method="ellipse")

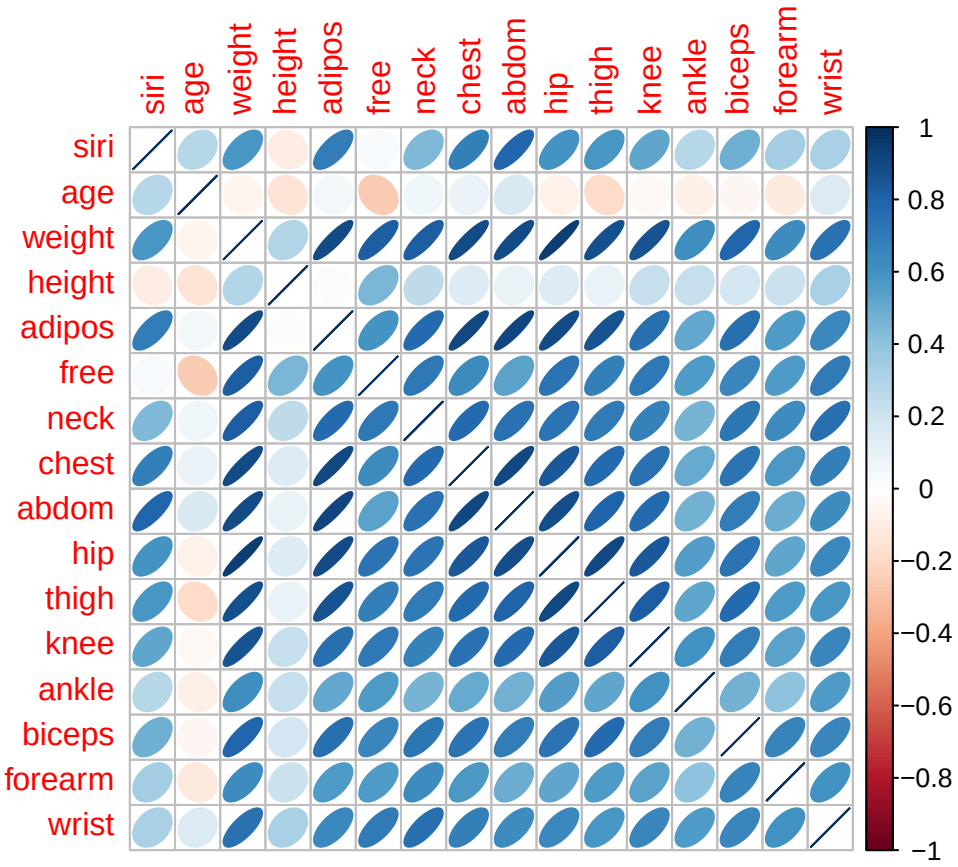
The above code plots a inter-correlation matrix of all predictors. The darker and thinner ellipses mean stronger correlation while lighter and thicker ellipses mean weak correlations. Blue stands for a positive and red for a negative relationship.

Which are the variables that are most strongly correlated with siri? Did you expect these results based on your insights from the previous scatterplots?

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
corrplot(cor(train),method="ellipse")
```




**Obervations**

Weight, adipos, chest, abdom, hip, thigh are few variables having a strong positive correlation with siri. This matches with the observation from the scatterplots

c) Collinearity leads to imprecise estimates of $\beta$. The signs of the coefficients can be the opposite of what intuition about the effect of the predictor might suggest. The standard errors are inflated so that t-tests may fail to reveal significant factors. The fit becomes very sensitive to measurement errors where small changes in y can lead to large changes in $\hat{\beta}$.

From the plot in 1b, do you find any multicollinearity among the predictors? Which set of predictors are correlated with each other?

**Obervations**

There is multicollinearity among some of the predictors. Weight, neck, adipos, chest, abdom are few of the predictors that are strongly inter-correlated.

d) Another way to detect multicollinearity in a model is by using Variance Inflation Factor (VIF). The equation below shows that the variance of a parameter is proportional to its VIF:

$$var(\hat{\beta}_j) \propto \frac{1}{1 - R_j^2}$$

where $R_j^2$ is the R-squared of the model with the jth variable as the response and all other variables as predictors. A high VIF ($\frac{1}{1-R_j^2}$) leads to higher variance in parameter estimates. If a variable is correlated with other variables it will have a high $R_j^2$ which instead will lead to a high VIF (more than 10 means multicollinearity)

Fit a linear regression model on your training data with siri as response vs all other variables as predictors. Use the vif() function in R to calculate VIFs of the predictors. Which variables have VIFs more than 10? Do you detect multicollinearity among the predictors?

Note: Although vif() function requires a model to be fit, it is more of an exploratory analysis since it doesn't involve the response

```
lmod=lm(siri~.,train)
vif(lmod)
```

```
##       age    weight    height    adipos      free      neck     chest
##  2.024062 48.092563  2.131428 15.920219  8.638952  4.435106  9.835945
##     abdom       hip     thigh      knee     ankle    biceps   forearm
## 17.669396 16.722325  9.501493  5.054944  1.874423  3.637442  2.251956
##     wrist
##  3.635517
```

**Obervations**

There is multicollinearity among the predictors. Weight, abdom, hip and adipos are predictors having VIFs more than 10

e) A third way to detect multicollinearity is to calculate the condition numbers of the predictor covariance matrix. Examine the eigenvalues of $X^TX$, $\lambda_1 \geq \lambda_2 ..... \geq \lambda_p$, where X is the model matrix of the fitted model, $\lambda_1$ is the largest eigenvalue and $\lambda_p$ the smallest. Zero eigenvalues denote exact collinearity while the presence of some small eigenvalues indicates multicollinearity.

The condition number $\kappa_j$ measures the relative size of the $j^{th}$ eigenvalue and is defined as:

$$\kappa_j = \sqrt{\frac{\lambda_1}{\lambda_j}}$$

A $\kappa_j$>30 is considered large and associated with multicollinearity

Use the eigen() function in R to calculate the eigenvalues of the covariance matrix. Then calculate the condition numbers associated with all eigenvalues relative to the largest eigenvalue. How many condition numbers are greater than 30? Do you detect multicollinearity?

```
x =model.matrix(lmod)[,-1]
e = eigen(t(x) %*% x)
sqrt(e$val[1]/e$val)
```

```
##  [1]   1.00000  19.50439  25.05902  38.20581  85.32331  97.27234 123.36528
##  [8] 159.24933 208.92010 214.64213 233.08810 257.69025 301.21943 384.52923
## [15] 648.70796
```

**Obervations**

There is significant multicollinearity among some of the predictors. 12 of 15 condition numbers are more than 30. This is a bad condition and would lead to imprecise estimates ofthe parameters

**Question 2: Fitting Linear Regression Model**

a) Fit a linear regression model on your training data with siri as response vs all other variables as predictors. Which predictors are significant at the 99% confidence level?

```
lmod=lm(siri~.,train)
summary(lmod)
```

```
##
## Call:
## lm(formula = siri ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7159 -0.6063  0.1915  0.9233  3.0378
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -10.594560   6.724660  -1.575 0.116854
## age           0.019788   0.012616   1.568 0.118477
## weight        0.385665   0.024169  15.957  < 2e-16 ***
## height        0.040285   0.038650   1.042 0.298627
## adipos       -0.555997   0.114766  -4.845 2.67e-06 ***
## free         -0.573169   0.016145 -35.502  < 2e-16 ***
## neck         -0.050896   0.088461  -0.575 0.565757
## chest         0.163546   0.039993   4.089 6.45e-05 ***
## abdom         0.092616   0.041098   2.254 0.025397 *
## hip          -0.004559   0.058004  -0.079 0.937444
## thigh         0.185935   0.058567   3.175 0.001757 **
## knee          0.167939   0.099010   1.696 0.091533 .
## ankle         0.109082   0.079849   1.366 0.173563
## biceps        0.113841   0.065605   1.735 0.084363 .
## forearm       0.284899   0.078281   3.639 0.000354 ***
## wrist        -0.054045   0.212586  -0.254 0.799603
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.46 on 185 degrees of freedom
## Multiple R-squared:  0.9693, Adjusted R-squared:  0.9668
## F-statistic: 389.9 on 15 and 185 DF,  p-value: < 2.2e-16
```

b) Build a new model on the training data with only the predictors that are statistically significant at the 99% confidence level. Perform an ANOVA test to compare this new model with the full model. Which one would you prefer? Explain

```
newmod=lm(siri~weight+free+adipos+chest+thigh+forearm,train)
anova(newmod,lmod)
```

```
## Analysis of Variance Table
##
## Model 1: siri ~ weight + free + adipos + chest + thigh + forearm
## Model 2: siri ~ age + weight + height + adipos + free + neck + chest +
##     abdom + hip + thigh + knee + ankle + biceps + forearm + wrist
##   Res.Df    RSS Df Sum of Sq      F   Pr(>F)
## 1    194 442.98
## 2    185 394.53  9    48.456 2.5247 0.009431 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

### Obervations

A p-value of 0.009 suggests that a simplification of considering only the significant predictors is not justified. Hence the full model is preferred between the two

c) Use the full model in 2a and predict() function in R to predict the response on the testing data. Calculate and report the RMSE (root mean squared error) of the response obtained on both the training and testing data.Why do you think there is a difference in the errors between the 2 datasets?

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y})^2}{n}}$$

```
rmse= function(x,y)
{
  return(sqrt(mean((x-y)^2)))
}

# Train RMSE
rmse(fitted(lmod),train$siri)
```

```
## [1] 1.401006
```

```
# Test RMSE
pred=predict(lmod,test)
rmse(pred,test$siri)
```

```
## [1] 1.78331
```

### Obervations

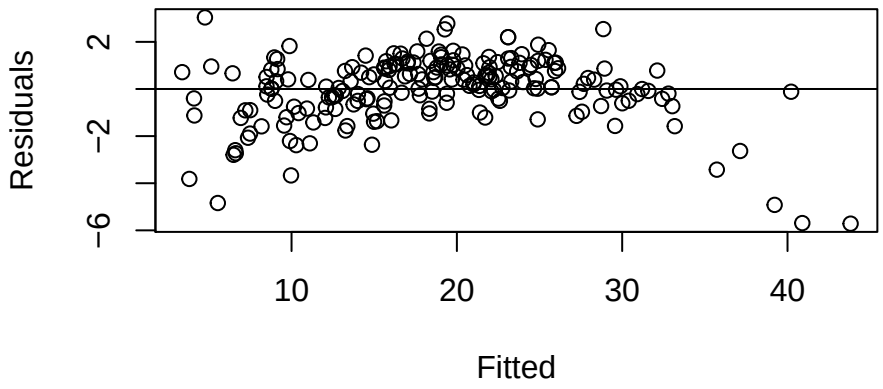
The test error is slightly more than the training error since we have trained our model on the training dataset and the model fits to the random effects of the training data slightly more than it generalizes on unknown data. This will be true in most cases (in expectation)

d) Perform a residual analysis to check for non-constant variance.State your observation. If you find any anomaly, perform a boxcox transformation on the response to remove any heteroscedasticity. What is your optimal choice of lambda? Fit a different model transforming the response by the optimal lambda and check for non-constant variance again. What do you observe?

Note: If your response has 0, add a small positive number to it during boxcox transformation:

For example: boxcox(lm(I(y+1)~x1+x2+...,train))

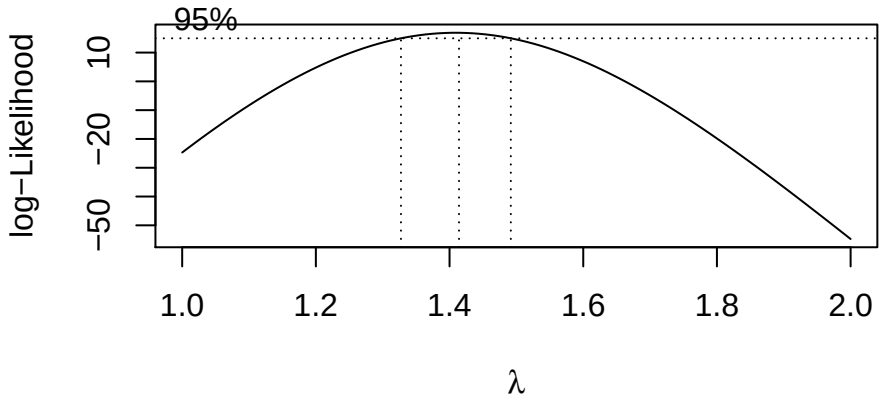
```
# Residual analysis
plot(fitted(lmod),residuals(lmod),ylab="Residuals",xlab="Fitted")
abline(0,0)
```



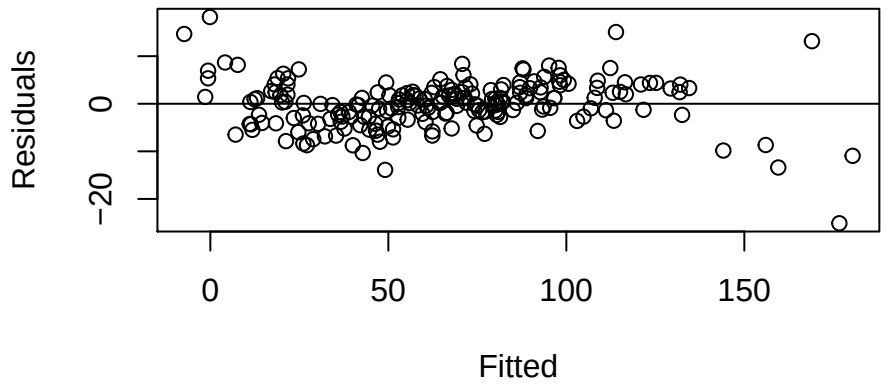

**Obervations**

There is non constant variance among the residuals. Some quadratic structure is still unexplained.

```
# Boxcox transformation
boxcox(lm(I(siri+1)~.,train),lambda = seq(1,2,0.1))
```



```
# Transformed model
lambda=1.41
lmodt=lm(I(siri^lambda)~.,train)
plot(fitted(lmodt),residuals(lmodt),ylab="Residuals",xlab="Fitted")
abline(0,0)
```




The optimal choice of lambda is 1.41. After refitting the model with this transformation, the non-constant variance is significantly reduced

e) Use the transformed model in 2d. Calculate and report the RMSE of the response obtained on both the training and testing data. Did the difference in training and testing errors increase or decrease as compared to the results of question 2c? Explain the change. Which results do you prefer?

Note: Please remember to back transform your predicted value to original scale before calculating. Also, if any predicted value is negative, convert them to 0 and then proceed

```
# Train RMSE
fit=ifelse(fitted(lmodt)<0,0,fitted(lmodt))
rmse(fit^(1/lambda),train$siri)
```

```
## [1] 1.287345
```

```
# Test RMSE
pred=predict(lmodt,test)
pred=ifelse(pred<0,0,pred)
rmse(pred^(1/lambda),test$siri)
```

```
## [1] 2.132381
```

**Obervations**

The training error reduced while the testing error increased. This happened since the model now overfits the training data even more on account of using a complicated transformation. I prefer the previous results more due to low generalization error

**Question 3: Variable selection using Stepwise Regression**

a) Use the leaps function in the leaps package and perform an all subset regression on the training data by "minimizing" Mallow's Cp statistics (method="Cp"). Report the variables of the best model, its training and testing error

```
library(leaps)
leapmod=leaps(train[,c(2:16)],train$siri,method="Cp")
bestmod=leapmod$which[which(leapmod$Cp==min(leapmod$Cp)),]
cbind(bestmod,colnames(train[,-1]))
```

```
##   bestmod
## 1 "FALSE" "age"
## 2 "TRUE"  "weight"
## 3 "FALSE" "height"
## 4 "TRUE"  "adipos"
## 5 "TRUE"  "free"
## 6 "FALSE" "neck"
## 7 "TRUE"  "chest"
## 8 "TRUE"  "abdom"
## 9 "FALSE" "hip"
## A "TRUE"  "thigh"
## B "TRUE"  "knee"
## C "TRUE"  "ankle"
## D "TRUE"  "biceps"
## E "TRUE"  "forearm"
## F "FALSE" "wrist"
```

```
# Final leaps model
lmodleap=lm(siri~.-age-height-neck-hip-wrist,train)
summary(lmodleap)
```

```
##
## Call:
## lm(formula = siri ~ . - age - height - neck - hip - wrist, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.8984 -0.5431  0.2203  0.9331  3.2127
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -8.86775    4.14674  -2.138 0.033754 *
## weight       0.38777    0.02125  18.251  < 2e-16 ***
## adipos      -0.61239    0.09646  -6.348 1.56e-09 ***
## free        -0.57664    0.01500 -38.452  < 2e-16 ***
## chest        0.16189    0.03895   4.157 4.88e-05 ***
## abdom        0.11189    0.03812   2.935 0.003749 **
## thigh        0.14121    0.04912   2.875 0.004500 **
## knee         0.19432    0.09576   2.029 0.043832 *
## ankle        0.11074    0.07716   1.435 0.152877
## biceps       0.13151    0.06363   2.067 0.040108 *
## forearm      0.26941    0.07341   3.670 0.000315 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.454 on 190 degrees of freedom
## Multiple R-squared:  0.9688, Adjusted R-squared:  0.9671
## F-statistic: 589.3 on 10 and 190 DF,  p-value: < 2.2e-16
```

```
# Train RMSE
rmse(fitted(lmodleap),train$siri)
```

```
## [1] 1.414026
```

```
# Test RMSE
pred=predict(lmodleap,test)
rmse(pred,test$siri)
```

```
## [1] 1.744894
```

**Obervations**

The all subset regression discarded the variables age, height, neck, hip and wrist. A lower test error is achieved than in 2c

a) Use the step() function on the original model in 2a to perform a backward stepwise regression by minimizing AIC. What was the change in AIC from the original model? Report the variables of the final model, its training and testing error. (Keep trace=FALSE)

```
stepmod=step(lmod,trace=F)
summary(stepmod)
```

```
##
## Call:
## lm(formula = siri ~ age + weight + adipos + free + chest + abdom +
##     thigh + knee + ankle + biceps + forearm, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7915 -0.5948  0.1545  0.9259  3.1107
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -9.40167    4.15278  -2.264 0.024714 *
## age          0.01621    0.01142   1.419 0.157410
## weight       0.39056    0.02128  18.353  < 2e-16 ***
## adipos      -0.62461    0.09659  -6.467 8.34e-10 ***
## free        -0.57513    0.01499 -38.357  < 2e-16 ***
## chest        0.16417    0.03888   4.223 3.75e-05 ***
## abdom        0.09484    0.03987   2.378 0.018382 *
## thigh        0.17667    0.05498   3.213 0.001544 **
## knee         0.16853    0.09722   1.733 0.084639 .
## ankle        0.11193    0.07696   1.454 0.147504
```

```
## biceps       0.11568    0.06443   1.795 0.074205 .
## forearm      0.27452    0.07330   3.745 0.000239 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.451 on 189 degrees of freedom
## Multiple R-squared:  0.9691, Adjusted R-squared:  0.9673
## F-statistic: 538.7 on 11 and 189 DF,  p-value: < 2.2e-16
```

```
# AIC change
AIC(lmod)-AIC(stepmod)
```

```
## [1] 6.412704
```

```
# Train RMSE
rmse(fitted(stepmod),train$siri)
```

```
## [1] 1.406549
```

```
# Test RMSE
pred=predict(stepmod,test)
rmse(pred,test$siri)
```

```
## [1] 1.76955
```

**Obervations**

Stepwise regression discarded the variables height, neck, hip and wrist from the original model. A lower test error is achieved than in 2c

**Question 4: Variable selection using Ridge, Lasso and Elasticnet regression**

a) Ridge Regression uses L2 regularization and causes coefficients to shrink for a large regularization parameter $\lambda$. The OLS loss function can be modified as:

$$L(\beta) = (Y - X\beta)^T(Y - X\beta) + \lambda\beta^T\beta$$

which gives the parameter estimates:

$$\hat{\beta} = (X^TX + \lambda I)^{-1}X^Ty$$

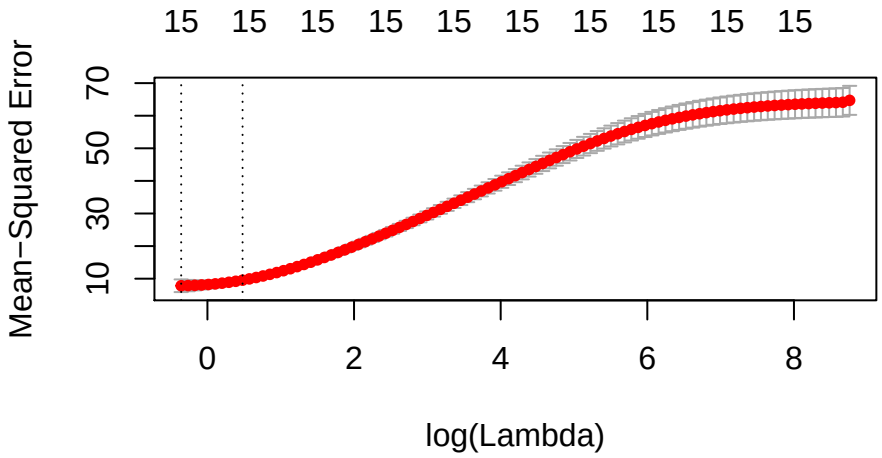
Use the glmnet() function in the library glmnet to build a Ridge Regression model by using the full model matrix of 2a as the training dataset.Perform a 10 fold cross validation with the training data and report the optimal $\lambda$ (lambda.min). Use this $\lambda$ to build the final model and report its training and testing error

Note: Remove the intercept column from the model matrix of the full model

```
X_train=model.matrix(lmod)[,-1]
y_train=train$siri

X_test=model.matrix(lm(siri~.,test))[,-1]
y_test=test$siri

# CV
library(glmnet)
```

```
## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-16
```

```
cv=cv.glmnet(X_train,y_train,alpha=0,nfolds = 10)
plot(cv)
```




```
#Optimal lambda
cv$lambda.min
```

```
## [1] 0.6991271
```

```
# Final ridge model
ridgemod=glmnet(X_train,y_train,alpha=0,lambda = cv$lambda.min)

# Train RMSE
fit=predict(ridgemod,X_train)
rmse(fit,y_train)
```

```
## [1] 2.300987
```

```
# Test RMSE
pred=predict(ridgemod,X_test)
rmse(pred,y_test)
```

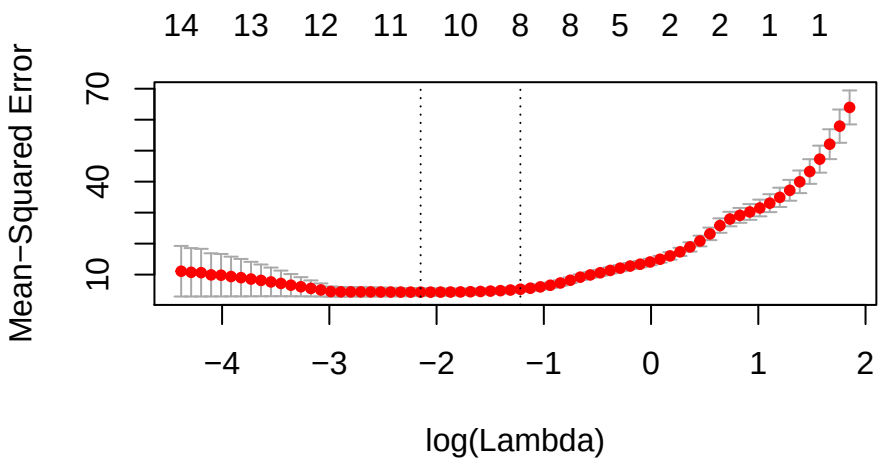
```
## [1] 2.724141
```

b) Lasso Regression uses L1 regularization and further reduces some coefficients to 0. The OLS loss function can be modified as:

$$L(\beta) = (Y - X\beta)^T(Y - X\beta) + \lambda||\beta||_1$$

Use the glmnet() function to build a lasso Regression model by using the full model matrix of 2a as the training dataset.Perform a 10 fold cross validation with the training data and report the optimal $\lambda$ (lambda.min). Use this $\lambda$ to build the final model. Report the final variables obtained (non 0 coefficients), the model training and testing error

```
# CV
library(glmnet)
cv=cv.glmnet(X_train,y_train,alpha=1,nfolds = 10)
plot(cv)
```




```
# Optimal lambda
cv$lambda.min
```

```
## [1] 0.1166214
```

```
# Final ridge model
lassomod=glmnet(X_train,y_train,alpha=1,lambda = cv$lambda.min)

#Coefficients
coef(lassomod)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                         s0
## (Intercept) -1.464213e+01
## age          6.618696e-04
## weight       2.836956e-01
## height       6.104115e-02
## adipos       .
## free        -4.893842e-01
## neck         .
## chest        8.066871e-02
## abdom        1.821601e-01
## hip          .
## thigh        5.670327e-02
## knee         2.779210e-01
## ankle        2.900518e-02
## biceps       7.961946e-02
## forearm      2.367560e-01
## wrist        .
```

```
# Train RMSE
fit=predict(lassomod,X_train)
rmse(fit,y_train)
```

```
## [1] 1.616942
```

```
# Test RMSE
pred=predict(lassomod,X_test)
rmse(pred,y_test)
```

```
## [1] 1.647812
```

**Obervations**

Predictors adipos, neck, hip and wrist were removed

c) Among all the variable selection models you built, which model has the lowest testing error? Which one is a low variance model? Which variable selection model would you prefer for predictive purposes?

Lasso has least testing RMSE and also the lowest difference in training and testing error and thus a low variance(less overfitting). I would prefer this model due to low generalization error and low overfitting as well. L1 Regularization helped in reducing variance and improving the prediction results

**Question 5: Principal Component Regression**

Although most variable selection methods penalizes the model for selecting more variables, they do not inherently account for removing multicollinearity. Principal Component Analysis (PCA) is a popular method for finding low-dimensional linear structure in higher dimensional data. It finds mutually orthogonal Principal Components (PCs) that are linear combinations of the other predictors. After obtaining the required PCs, we fit a linear model with the response vs the PCs. This is called Principal Component Regression. It is not a variable selection method, but it is a good way to remove multicollinearity in a model and obtain a good fit

In this question you will directly fit a PCR model on the training data and perform cross validation by minimizing RMSE to obtain the desired number of PCs. Use the following code to plot the Cross Validation curve:

library(pls)

set.seed(123)

pcrmod = pcr(siri ~ ., data=train, validation="CV", ncomp=15)

pcrCV = RMSEP(pcrmod, estimate="CV")

plot(pcrCV,main="")

What should be the best choice of # of PCs? (which.min(pcrCV$val))

Report the training and testing error and compare with the best variable selection model you found in 4c in terms of overfitting (high variance) and generalization error. Which one between these two would you use?

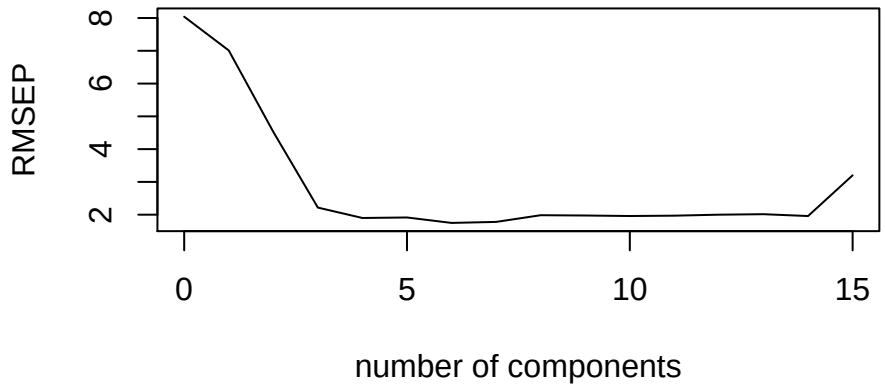
For predicting use:

predict(pcrmod,train,ncomp = best # of PCs)

```
library(pls)
```

```
##
## Attaching package: 'pls'

## The following object is masked from 'package:corrplot':
##
##     corrplot

## The following object is masked from 'package:stats':
##
##     loadings
```

```
set.seed(123)
pcrmod = pcr(siri ~ ., data=train, validation="CV", ncomp=15)
pcrCV = RMSEP(pcrmod, estimate="CV")
plot(pcrCV,main="")
```



```r
# Best # of PCs
which.min(pcrCV$val)
```

```
## [1] 7
```

```r
# Training error
rmse(predict(pcrmod,train,ncomp = 7),train$siri)
```

```
## [1] 1.563399
```

```r
# Testing error
pred = predict(pcrmod, test, ncomp=7)
rmse(pred, test$siri)
```

```
## [1] 1.633589
```

**Obervations**

Top 7 components have the lowest Cross Validation RMSE and is the best choice. The training and testing error is higher than Lasso but the difference between them is lowest among all models. Thus PCR is the lowest variance model (less overfitting). If using more variables is costlier, I will prefer Lasso over PCR due to lowest test error, else PCR is an equally good choice