Homework Assignment for Chapter 6

6.1. Infrared spectroscopy technology

a) Load the dataset

```
> head(absorp)
       [,1]    [,2]    [,3]    [,4]    [,5]    [,6]    [,7]    [,8]    [,9]   [,10]
[1,] 2.61776 2.61814 2.61859 2.61912 2.61981 2.62071 2.62186 2.62334 2.62511 2.62722
[2,] 2.83454 2.83871 2.84283 2.84705 2.85138 2.85587 2.86060 2.86566 2.87093 2.87661
[3,] 2.58284 2.58458 2.58629 2.58808 2.58996 2.59192 2.59401 2.59627 2.59873 2.60131
[4,] 2.82286 2.82460 2.82630 2.82814 2.83001 2.83192 2.83392 2.83606 2.83842 2.84097
[5,] 2.78813 2.78989 2.79167 2.79350 2.79538 2.79746 2.79984 2.80254 2.80553 2.80890
[6,] 3.00993 3.01540 3.02086 3.02634 3.03190 3.03756 3.04341 3.04955 3.05599 3.06274
       [,11]   [,12]   [,13]   [,14]   [,15]   [,16]   [,17]   [,18]   [,19]   [,20]
[1,] 2.62964 2.63245 2.63565 2.63933 2.64353 2.64825 2.65350 2.65937 2.66585 2.67281
[2,] 2.88264 2.88898 2.89577 2.90308 2.91097 2.91953 2.92873 2.93863 2.94929 2.96072
[3,] 2.60414 2.60714 2.61029 2.61361 2.61714 2.62089 2.62486 2.62909 2.63361 2.63835
[4,] 2.84374 2.84664 2.84975 2.85307 2.85661 2.86038 2.86437 2.86860 2.87308 2.87789
[5,] 2.81272 2.81704 2.82184 2.82710 2.83294 2.83945 2.84664 2.85458 2.86331 2.87280
[6,] 3.06982 3.07724 3.08511 3.09343 3.10231 3.11185 3.12205 3.13294 3.14457 3.15703
       [,21]   [,22]   [,23]   [,24]   [,25]   [,26]   [,27]   [,28]   [,29]   [,30]
[1,] 2.68008 2.68733 2.69427 2.70073 2.70684 2.71281 2.71914 2.72628 2.73462 2.74416
[2,] 2.97272 2.98493 2.99690 3.00833 3.01920 3.02990 3.04101 3.05345 3.06777 3.08416
[3,] 2.64330 2.64838 2.65354 2.65870 2.66375 2.66880 2.67383 2.67892 2.68411 2.68937
[4,] 2.88301 2.88832 2.89374 2.89917 2.90457 2.90991 2.91521 2.92043 2.92565 2.93082
[5,] 2.88291 2.89335 2.90374 2.91371 2.92305 2.93187 2.94060 2.94986 2.96035 2.97241
[6,] 3.17038 3.18429 3.19840 3.21225 3.22552 3.23827 3.25084 3.26393 3.27851 3.29514
```

```
> head(endpoints)
     [,1] [,2] [,3]
[1,] 60.5 22.5 16.7
[2,] 46.0 40.1 13.5
[3,] 71.0  8.4 20.5
[4,] 72.8  5.9 20.7
[5,] 58.3 25.5 15.5
[6,] 44.0 42.7 13.7
```

b) What is the effective dimension?

```
> head(percentV)
[1] 98.626192582  0.969705229  0.279324276  0.114429868  0.006460911  0.002624591
```

When applying PCA, the output shows that the first component (PC1) has a value of 98.6262, implying that the first component explains 98.6% of variance.

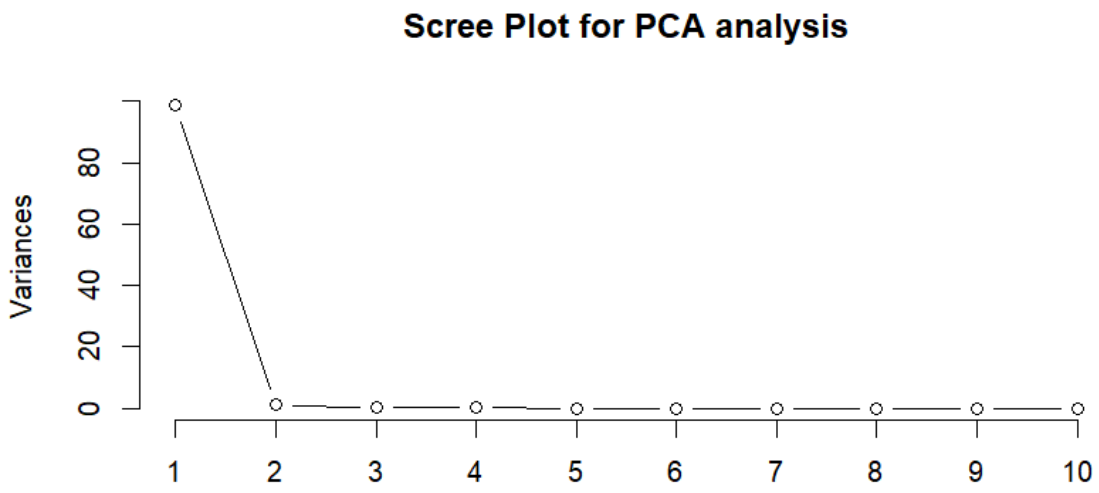

Fig 1: Scree plot for the Principal Components

Looking at the scree plot(Fig 1), the elbow comes at the second component hence it would be best to retain the first component. Therefore, 1 PC is the effective dimension of the data.

c) Split data. Finding Average RMSE

From the dataset, we have three endpoints (water, protein, and fat). However, the required predictive relationship is between IR spectrum and fat to give an insight on cost saving.

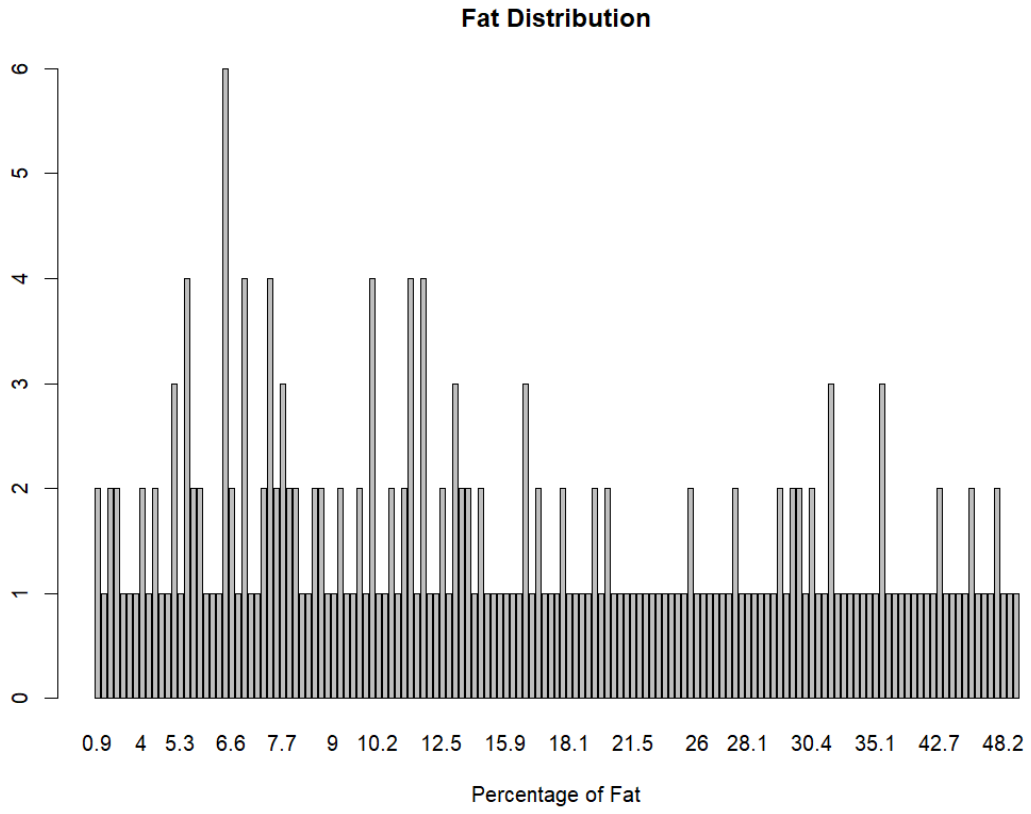

Fig 2: Fat Distribution(Response predictor distribution)

Given the sample size, the data was split into 80/20(training/testing). I used 3-fold cross validation to tune the model.

```
Principal Component Analysis

174 samples
100 predictors

No pre-processing
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 116, 117, 115
Resampling results across tuning parameters:

  ncomp  RMSE       Rsquared   MAE
   1     10.981883  0.2506470  8.710124
   2     10.674797  0.2935905  8.235222
   3      7.544293  0.6489096  5.722118
   4      4.199755  0.8935570  3.381306
   5      3.471809  0.9268505  2.748016
   6      3.112264  0.9399246  2.476476
   7      3.116500  0.9396248  2.485477
   8      3.122610  0.9393366  2.473963
   9      2.963186  0.9451893  2.322251
  10      3.070592  0.9412835  2.348413
  11      2.975077  0.9456320  2.272556
  12      2.950664  0.9461570  2.259002
  13      3.026526  0.9432326  2.314966
  14      3.042392  0.9427781  2.313484
  15      3.001441  0.9437882  2.284206
  16      3.031198  0.9439979  2.146183
  17      3.110435  0.9414000  2.139050
  18      3.063905  0.9435952  2.134869
  19      3.121507  0.9419063  2.094736
  20      3.074808  0.9433026  2.008535

RMSE was used to select the optimal model using
 the smallest value.
The final value used for the model was ncomp = 12.
```

12 components were the best option in building the model according to the outcome. The model performance based on the final parameter was recorded.

ncomp	RMSE	Rsquared	MAE
12	2.950664	0.9461570	2.259002

Average RMSE:

```
> av
[1] 4.132782
```

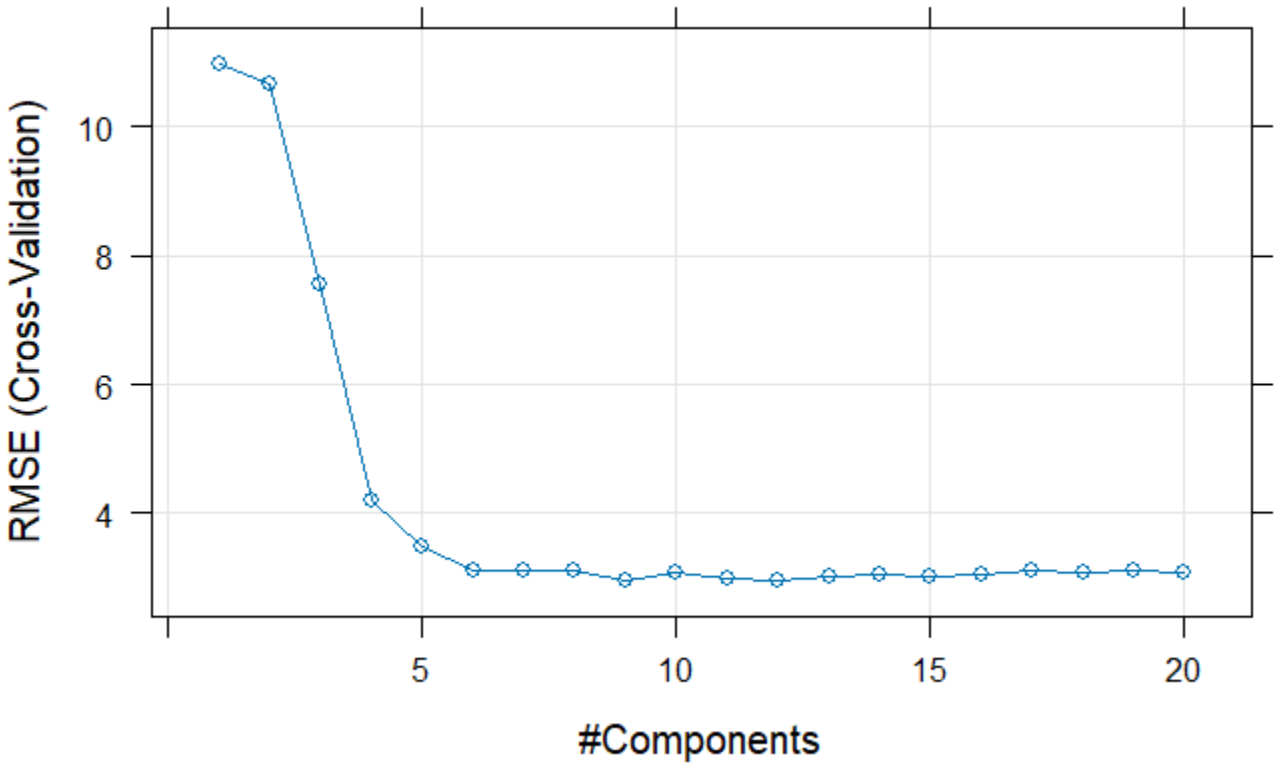

Fig 3: Plot of RMSE vs Components.

The average RMSE was 4.132782. Looking at **Fig 3**, the RMSE is lower at the 12th component before rising. Therefore, 12 is the optimal number of components.

d)Prediction. What is the RMSE?

```
> test_pred <- predict(pcrm, newdata =as.data.frame(testAbs))
> postResample(testF, test_pred)
     RMSE  Rsquared       MAE
2.4646632 0.9728986 1.7212337
```

The test data results had an RMSE=2.4646632 and Rsquared was 97.29%

6.2 Developing a model.

a) Start R

```
> head(permeability)
  permeability
1       12.520
2        1.120
3       19.405
4        1.730
5        1.680
6        0.510
```

b)Predictors left for modeling.

```
> dim(fingerprints)
[1]  165 1107
>
> fingerprints_F<-fingerprints[,-nearZeroVar(fingerprints)]
> dim(fingerprints_F)
[1] 165 388
```

There were 1107 predictors in the dataset. However, after removing near Zero Variance predictors, only 388 predictors are left for training and testing.

c) Pre-process the data.

The data was centered and scaled, and I split the data 80% training and 20% modelling.

```
Partial Least Squares

133 samples
388 predictors

Pre-processing: centered (388), scaled (388)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 119, 121, 119, 120, 120, 120, ...
Resampling results across tuning parameters:

  ncomp  RMSE      Rsquared   MAE
   1     12.89902  0.3366671   9.723079
   2     11.57073  0.4528188   8.008647
   3     11.74004  0.4540994   8.638716
   4     11.93784  0.4435949   8.785540
   5     11.72550  0.4550101   8.696446
   6     11.50509  0.4693658   8.546633
   7     11.39297  0.4871664   8.653850
   8     11.13762  0.5051883   8.558643
   9     11.18859  0.5013221   8.642897
  10     11.29239  0.4985107   8.727017
  11     11.47258  0.4836285   8.824869
  12     11.42934  0.4880340   8.975986
  13     11.74486  0.4702584   9.189660
  14     12.08391  0.4502299   9.380611
  15     12.27229  0.4497430   9.560355
  16     12.53866  0.4429663   9.685105
  17     12.57991  0.4380633   9.684354
  18     12.60036  0.4371346   9.779651
  19     12.65043  0.4323267   9.815696
  20     12.94446  0.4207868  10.028232

Rsquared was used to select the optimal
 model using the largest value.
The final value used for the model was
 ncomp = 8.
```

The RMSE was 11.13762. Using R^2 as the decision metric, the optimal number of components was 8 with the maximum R^2 being 0.5051883.

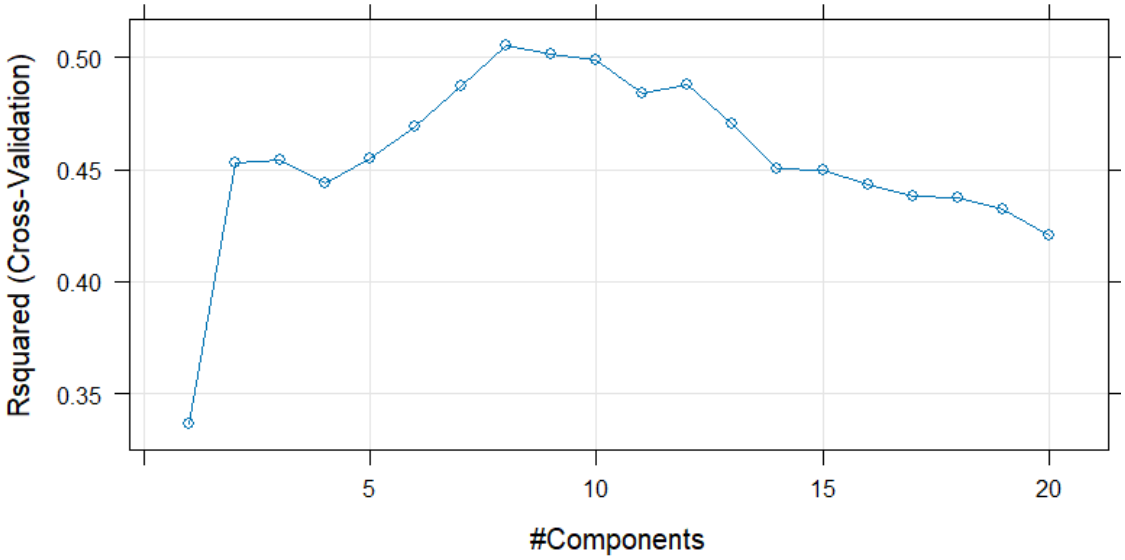

Fig 4: Plot of Rsquared vs Components

From Fig 4, the Rsquared is highest at the 8th component.

d) Predict the response for the test set.

```
> pls_pred<-predict(plsT, newdata=testF_X)
> postResample(pls_pred,testP_y)
      RMSE   Rsquared        MAE
11.0223004  0.5357105  7.9250542
```

The set estimate of R^2 is 0.5357105

6.3 Chemical manufacturing process dataset.

a) Start R

```
> str(ChemicalManufacturingProcess)
'data.frame':   176 obs. of  58 variables:
 $ Yield                 : num  38 42.4 42 41.4 42.5 ...
 $ BiologicalMaterial01  : num  6.25 8.01 8.01 8.01 7.47 6.12 7.48 6.94
6.94 6.94 ...
 $ BiologicalMaterial02  : num  49.6 61 61 61 63.3 ...
 $ BiologicalMaterial03  : num  57 67.5 67.5 67.5 72.2 ...
 $ BiologicalMaterial04  : num  12.7 14.6 14.6 14.6 14 ...
 $ BiologicalMaterial05  : num  19.5 19.4 19.4 19.4 17.9 ...
 $ BiologicalMaterial06  : num  43.7 53.1 53.1 53.1 54.7 ...
 $ BiologicalMaterial07  : num  100 100 100 100 100 100 100 100 100 100
...
 $ BiologicalMaterial08  : num  16.7 19 19 19 18.2 ...
 $ BiologicalMaterial09  : num  11.4 12.6 12.6 12.6 12.8 ...
 $ BiologicalMaterial10  : num  3.46 3.46 3.46 3.46 3.05 3.78 3.04 3.85
3.85 3.85 ...
 $ BiologicalMaterial11  : num  138 154 154 154 148 ...
 $ BiologicalMaterial12  : num  18.8 21.1 21.1 21.1 21.1 ...
 $ ManufacturingProcess01: num  NA 0 0 0 10.7 12 11.5 12 12 12 ...
 $ ManufacturingProcess02: num  NA 0 0 0 0 0 0 0 0 0 ...
 $ ManufacturingProcess03: num  NA NA NA NA NA NA 1.56 1.55 1.56 1.55 .
 $ ManufacturingProcess04: num  NA 917 912 911 918 924 933 929 928 938
...
```

b) Use imputation.

```
> sum(is.na(ChemicalManufacturingProcess))
[1] 106
> missing <- preProcess(ChemicalManufacturingProcess, method =c("knnImpute"))
> df.chem<- predict(missing, ChemicalManufacturingProcess)
> sum(is.na(df.chem))
[1] 0
```

There were about 106 missing values in the original dataset. We used knn imputation, filling in the missing values through estimation of the missing values from their k-nearest neighbors with observed value, to impute the data.

c) Preprocess the data.

Table 1: Summary of models

Model	Best Tuning Parameter	Training		Testing	
		RMSE	R^2	RMSE	R^2
Linear Model		1.630829	0.247266	4.04149157	0.06609376
Ridge	Lambda=0.1	1.789900	0.3451704	0.7022892	0.6910460
LASSO	Fraction=0.05357895, lambda=0	0.6807791	0.5250750	0.7500739	0.6506603
E-Net	Fraction=0.3, lambda=0.1	0.6473696	0.5720017	0.6830295	0.7051587

Ridge Regression

```
Ridge Regression

144 samples
 57 predictor

Pre-processing: centered (57), scaled (57)
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 96, 96, 96
Resampling results across tuning parameters:

  lambda       RMSE      Rsquared   MAE
  0.000000000  5.801474  0.1612990  1.7157518
  0.005263158  2.406718  0.3043383  0.9784249
  0.010526316  2.114447  0.3180564  0.8826469
  0.015789474  2.021686  0.3247268  0.8484350
  0.021052632  1.982119  0.3287515  0.8301052
  0.026315789  1.960486  0.3315107  0.8175486
  0.031578947  1.945439  0.3335775  0.8078926
  0.036842105  1.932799  0.3352260  0.7996058
  0.042105263  1.920933  0.3366013  0.7924054
  0.047368421  1.909203  0.3377863  0.7899299
  0.052631579  1.897388  0.3388318  0.7892693
  0.057894737  1.885438  0.3397705  0.7880704
  0.063157895  1.873377  0.3406247  0.7864812
  0.068421053  1.861255  0.3414102  0.7846105
  0.073684211  1.849124  0.3421385  0.7825390
  0.078947368  1.837037  0.3428184  0.7803270
  0.084210526  1.825038  0.3434566  0.7780203
  0.089473684  1.813163  0.3440585  0.7756538
  0.094736842  1.801442  0.3446286  0.7732542
  0.100000000  1.789900  0.3451704  0.7709577

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was lambda = 0.1.
>
> predictions <- predict(ridgeTune, newdata = x.test)
>
> postResample(predictions,y.test)
     RMSE  Rsquared       MAE
0.6485367 0.6725848 0.5178750
```

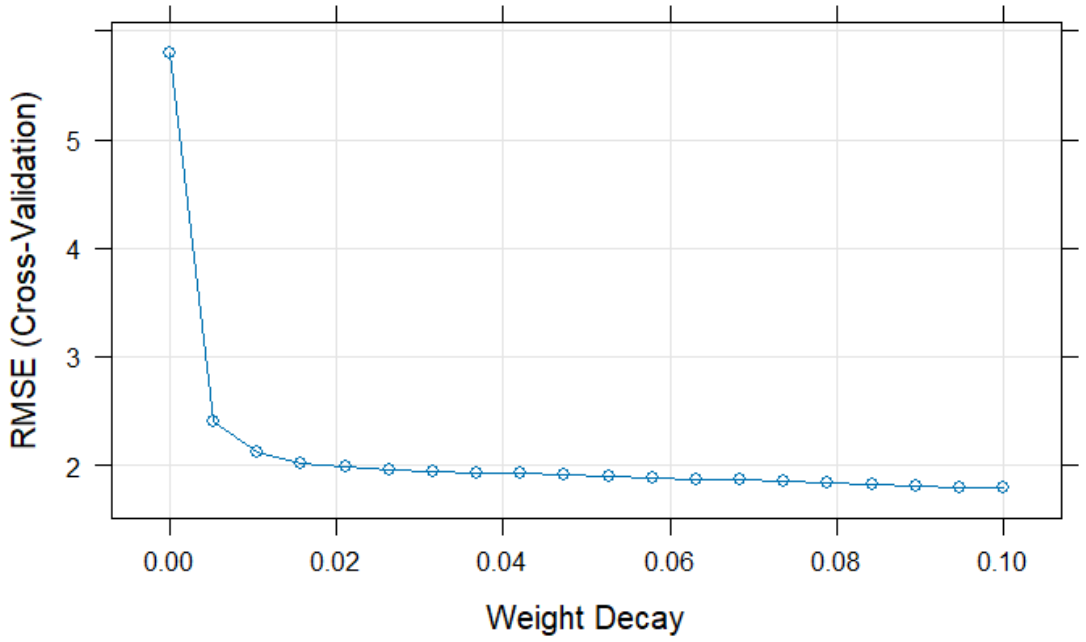

Fig 5(Ridge regression): Plot of RMSE vs Weight Decay

Elastic net regression

```
Elasticnet

144 samples
 57 predictor

Pre-processing: centered (57), scaled (57)
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 96, 96, 96
Resampling results across tuning parameters:

  lambda  fraction  RMSE       Rsquared   MAE
  0.00    0.05      0.6550001  0.5552066  0.5272496
  0.00    0.10      0.6858266  0.5158991  0.5337051
  0.00    0.15      0.8527491  0.4137763  0.5903827
  0.00    0.20      1.0732940  0.3750885  0.6499410
  0.00    0.25      1.1970200  0.3871221  0.6927066
  0.00    0.30      1.3424879  0.3651347  0.7347707
  0.00    0.35      1.5522053  0.2984128  0.7961139
  0.00    0.40      2.2204057  0.1497813  0.9235511
  0.00    0.45      2.4336431  0.1370653  0.9800084
  0.00    0.50      2.9277200  0.1276432  1.0701826
  0.00    0.55      3.5432835  0.1259823  1.1728828
  0.00    0.60      4.0178080  0.1287329  1.2531552
  0.00    0.65      4.6363548  0.1326446  1.3551335
  0.00    0.70      5.6099914  0.1405593  1.5065075
  0.00    0.75      6.5900352  0.1525768  1.6592029
  0.00    0.80      7.4832416  0.1681905  1.7980887
  0.00    0.85      8.3198719  0.1816350  1.9225324
  0.00    0.90      9.1681089  0.1802042  2.0486018
  0.00    0.95      9.5687587  0.1597062  2.1199093
  0.00    1.00      9.7966937  0.1338916  2.1551359
  0.01    0.05      0.8038932  0.4893171  0.6596207
  0.01    0.10      0.6837837  0.5711442  0.5654375
  0.01    0.15      0.6556885  0.5731002  0.5307640
  0.01    0.20      0.6715284  0.5464063  0.5260737
  0.01    0.25      0.7010075  0.5193454  0.5313463
  0.01    0.30      0.7386383  0.4929133  0.5445757
  0.01    0.35      0.7687504  0.4707460  0.5591002
```

```
  0.10    0.25      0.6508584  0.5787468  0.5320246
  0.10    0.30      0.6473696  0.5720017  0.5242300
  0.10    0.35      0.6565031  0.5569265  0.5232792
  0.10    0.40      0.6658324  0.5471502  0.5229320
  0.10    0.45      0.6803073  0.5344306  0.5265959
  0.10    0.50      0.6995555  0.5207917  0.5318436
  0.10    0.55      0.7222560  0.5028822  0.5450033
  0.10    0.60      0.7427175  0.4873943  0.5571681
  0.10    0.65      0.7812109  0.4766820  0.5670633
  0.10    0.70      0.8534451  0.4549681  0.5830104
  0.10    0.75      0.9127256  0.4419263  0.5961727
  0.10    0.80      0.9858524  0.4266883  0.6182073
  0.10    0.85      1.0497528  0.4106259  0.6377264
  0.10    0.90      1.0765449  0.3989210  0.6500582
  0.10    0.95      1.1095611  0.3808866  0.6656470
  0.10    1.00      1.1543994  0.3570700  0.6832747

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were fraction = 0.3 and lambda = 0.1.
> plot(enetTune, metric='RMSE')
>
>
> prediction4 <- predict(enetTune, newdata = x.test)
>
> postResample(prediction4,y.test)
     RMSE  Rsquared       MAE
0.6830295 0.7051587 0.5456038
>
```

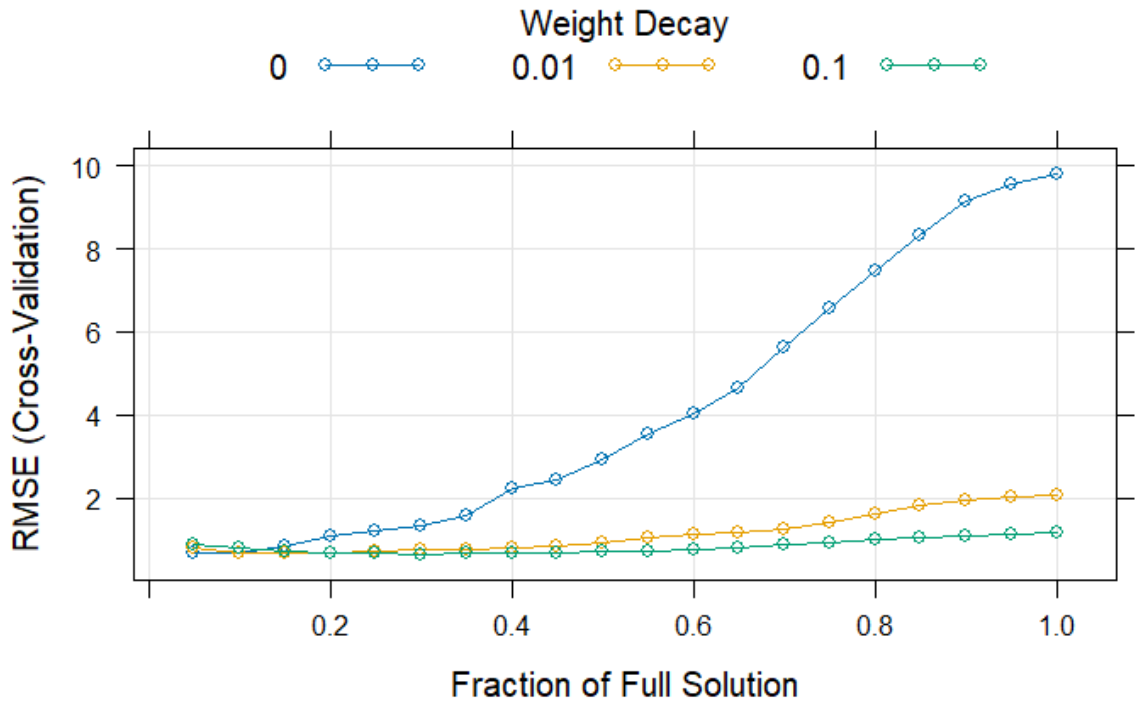

Fig 6(Elastic net): Plot of RMSE vs Fractions

Lasso Regression

```
Elasticnet

144 samples
 57 predictor

Pre-processing: centered (57), scaled (57)
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 96, 96, 96
Resampling results across tuning parameters:

  fraction    RMSE       Rsquared   MAE
  0.00100000  0.9489249  0.3578846  0.7763634
  0.05357895  0.6807791  0.5250750  0.5287749
  0.10615789  0.7602200  0.4759067  0.5492581
  0.15873684  0.7933802  0.4507810  0.5870030
  0.21131579  1.0631323  0.3314055  0.6562056
  0.26389474  1.3456723  0.2798757  0.7139566
  0.31647368  1.2114803  0.3329889  0.6969071
  0.36905263  0.9762385  0.3817955  0.6652785
  0.42163158  0.7984148  0.4614330  0.6271220
  0.47421053  0.8954646  0.4112180  0.6710709
  0.52678947  1.2889091  0.3396911  0.7545495
  0.57936842  1.8994365  0.3131844  0.8779116
  0.63194737  2.3789519  0.2874964  0.9816004
  0.68452632  2.8923695  0.2498663  1.0875711
  0.73710526  3.5252843  0.2200818  1.2091865
  0.78968421  4.3784198  0.1996690  1.3639698
  0.84226316  5.2321376  0.1863726  1.5177916
  0.89484211  6.0822639  0.1772516  1.6707088
  0.94742105  6.9352385  0.1699441  1.8239693
  1.00000000  7.7853219  0.1639586  1.9772799

Tuning parameter 'lambda' was held constant at a value of 0
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were fraction = 0.05357895 and lambda = 0.
```

```
> prediction6 <- predict(lassoTune, newdata = x.test)
>
> postResample(prediction6,y.test)
     RMSE  Rsquared       MAE
0.7022892 0.6910460 0.5574308
```

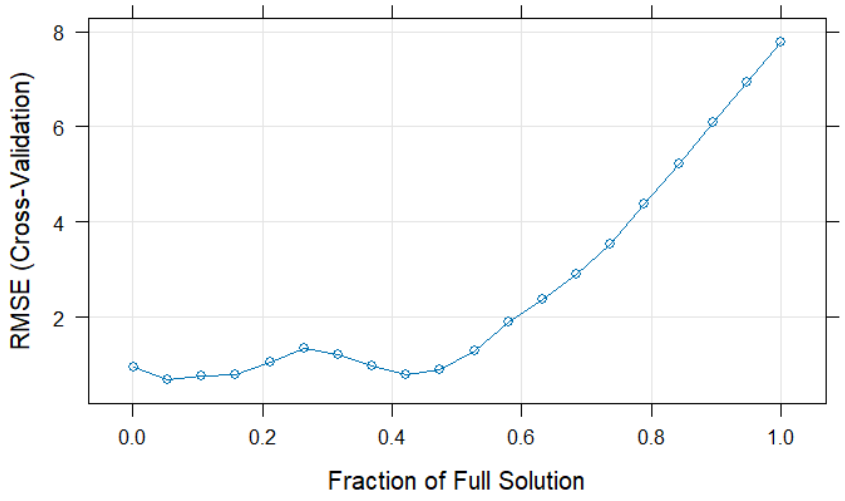

Fig 7(Lasso): Plot of RMSE vs Fractions

Linear Regression

```
Linear Regression

144 samples
 40 predictor

Pre-processing: centered (40), scaled (40)
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 96, 96, 96
Resampling results:

  RMSE      Rsquared  MAE
  1.630829  0.247266  0.7973174

Tuning parameter 'intercept' was held constant at a value of TRUE
>
>
> prediction7 <- predict(lm_model, newdata = X.test)
>
> postResample(prediction7,y.test)
      RMSE   Rsquared        MAE
4.04149157 0.06609376 1.18062499
```

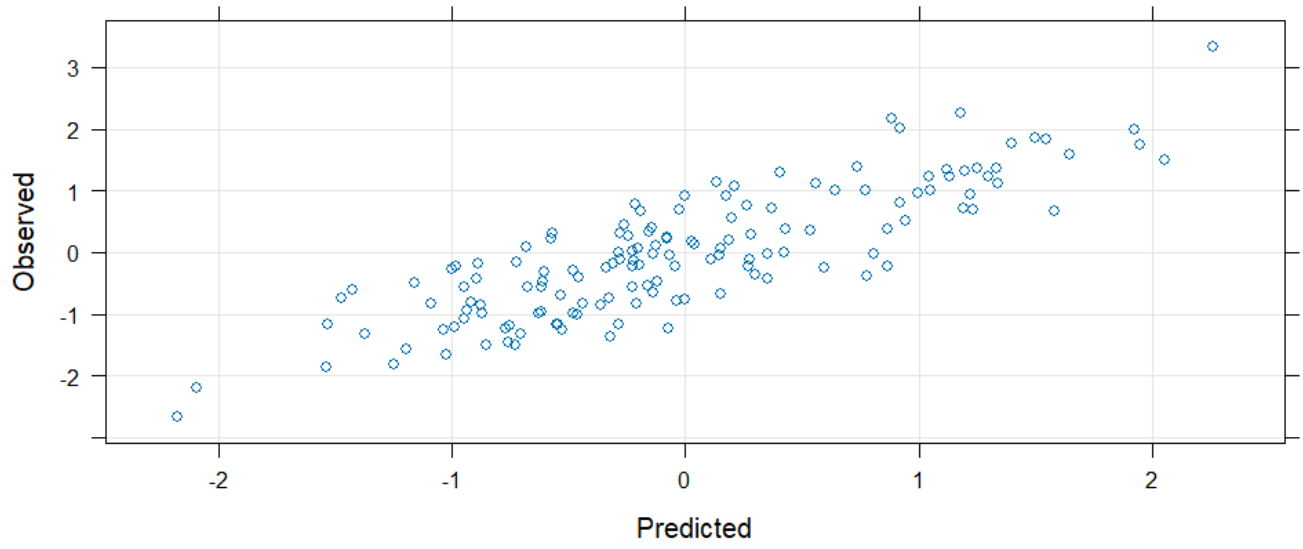

Fig 8(Linear Model): Plot of observed(y) vs predicted(yhat)

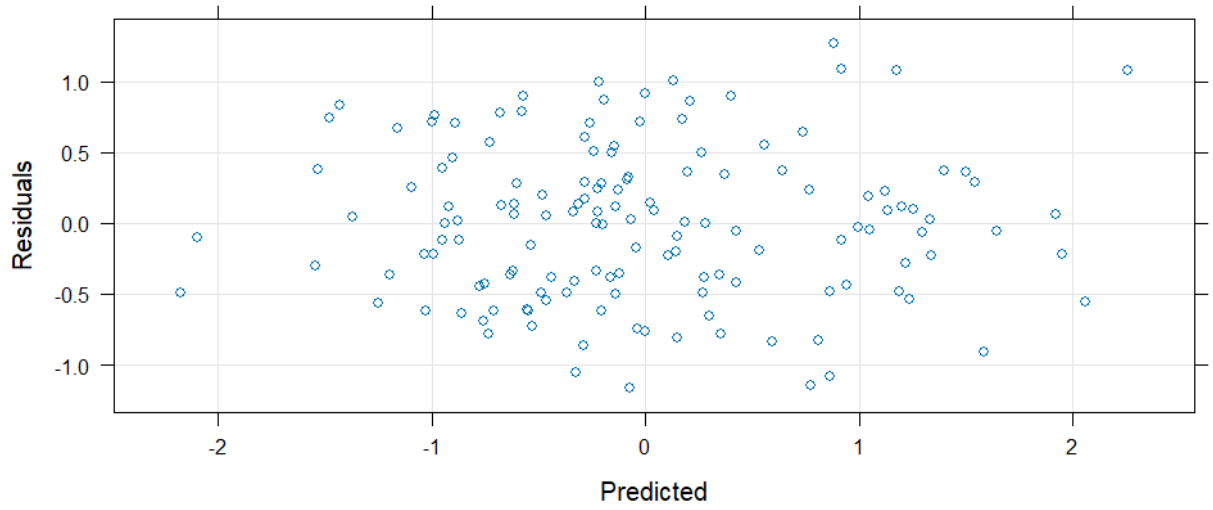

Fig 9(Linear Model): Residuals vs predicted(yhat)

d) Predict. What is the RMSE?

The four different models were trained on the dataset and predicted on the testing set. The performance of every model was calculated by predicting it on new data. The Elastic Net model had the best performance with a lower RMSE of 0.6830.

e) Which model has the best predictive ability?

Based on the performance of the models, Elastic net regression model has the best predictive ability as it produced better statistical results. It has a higher R squared of 70.52% and a lower RMSE of 0.6830 compared to the other models.

f) Which predictors are the most important?

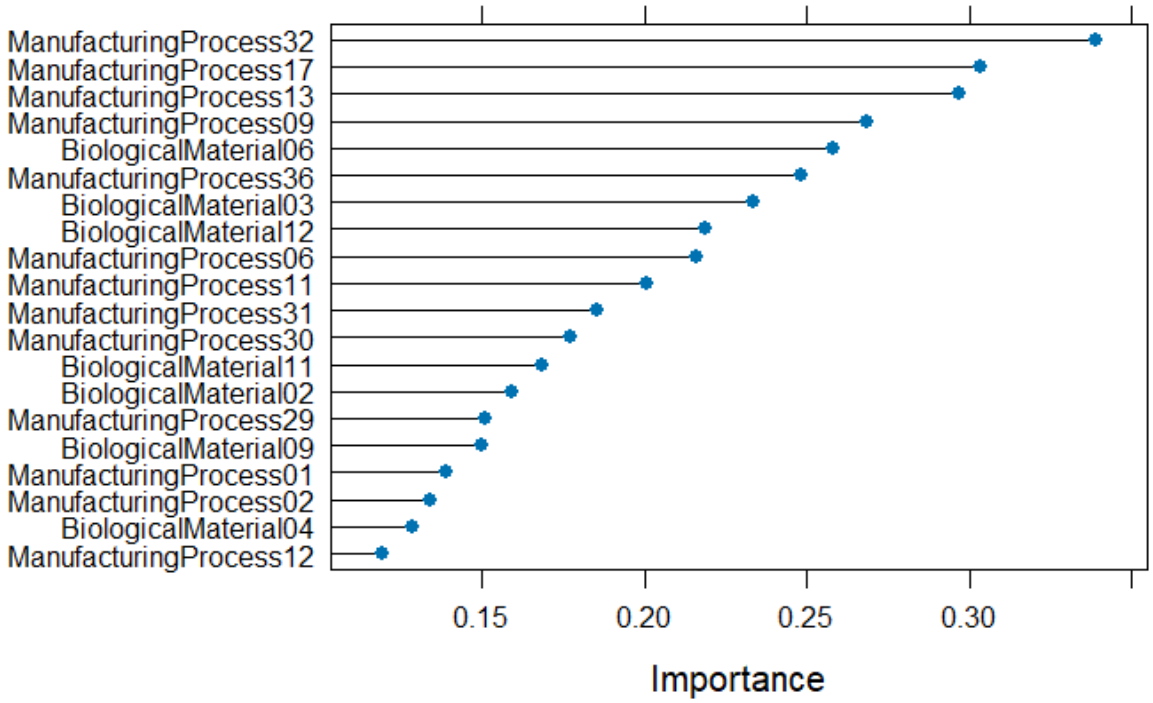

Fig 10: Graph showing predictors from the most important in a decreasing order.

Fig 10 shows a graph of predictors from the Elastic net models in terms of their performance in the model. The most important predictors are ranked with ManufacturingProcess 32 being the top in terms of importance. ManufacturingProcess 17 and 13 make the top three most important predictors.

Appendix

```
#Question 6.1
#Part a
library(caret)
data(tecator)
head(absorp)
head(endpoints)
#part b
#scale and center data prior to PCA
pcaO<- prcomp(absorp, center=TRUE, scale=TRUE)

# cumulative percentage of variance for each component
percentV<-pcaO$sdev^2/sum(pcaO$sdev^2)*100
head(percentV)

screeplot(pcaO, npcs=10, type='lines', main='Scree Plot for PCA analysis')

#part c
c<-table(endpoints[,2])
barplot(c, main='Fat Distribution',xlab='Percentage of Fat')



set.seed(100)
trainR<-createDataPartition(endpoints[,2], p=0.8, list=FALSE)
trainAbs<-absorp[trainR,]
testAbs<-absorp[-trainR,]
trainF<-endpoints[trainR,2]
testF<-endpoints[-trainR,2]
```

```
str(trainAbs)
trainAbs<-as.data.frame(trainAbs)

ctrl<-trainControl(method='cv', number=3)

pcrm<-train(x=trainAbs, y=trainF, method='pcr',
        trControl=ctrl, tuneLength=20, preProc=c('center','scale'))
print(pcrm)
plot(pcrm)

#Average RMSE
#rmsev<-pcrm$results$RMSE
# av<-mean(rmsev)
# av

#part d
#prediction<-predict(pcrm, testAbs)
# part d
# Predict the response for the test set
test_pred <- predict(pcrm, newdata =as.data.frame(testAbs))
test_pred
rmse <- sqrt(mean((testF - test_pred)^2))
rmse

postResample(testF, test_pred)
```

```
#Question 6.2
#part a
library(AppliedPredictiveModeling)
data(permeability)
head(permeability)
summary(permeability)
str(permeability)

#part b
dim(fingerprints)

fingerprints_F<-fingerprints[,-nearZeroVar(fingerprints)]
dim(fingerprints_F)

#part c
set.seed(100)
#index for training
trainFp<-createDataPartition(permeability, p=0.8, list=FALSE)

#train
trainP_y<-permeability[trainFp,]
trainF_X<-fingerprints_F[trainFp,]

#test
testP_y<-permeability[-trainFp,]
testF_X<-fingerprints_F[-trainFp,]

trainF_X<-as.data.frame(trainF_X)
```

```
#cv
ctrl<-trainControl(method='cv', number=10)
plsT <- train(x=trainF_X,y=trainP_y, method = "pls", metric = "Rsquared",
              tuneLength = 20, trControl = ctrl,
            preProc = c("center", "scale"))

#plsR<-plsT$results
plsT

plot(plsT)

# set.seed(1)
# ctrl<-trainControl(method='cv',number=10)
# plsT<-train(x=trainF, y=trainP, method='pls', tuneLength =20,
#             trControl =ctrl, preProc=c('center', 'scale') )
# print(plsT)
# plot(plsT, metric='Rsquared', main='PLS tuning parameter for permeability Data')



#part d

#predict
pls_pred<-predict(plsT, newdata=testF_X)
postResample(pls_pred,testP_y)
```

```
#part 6.3
#part a
library(AppliedPredictiveModeling)
data("ChemicalManufacturingProcess")

library(caret)

missing <- preProcess(ChemicalManufacturingProcess, method =c("knnImpute"))
df.chem<- predict(missing, ChemicalManufacturingProcess)
any(is.na(df.chem))

colYield <-which(colnames(ChemicalManufacturingProcess) == "Yield")

X<-as.data.frame(df.chem[,-colYield])
Y<-df.chem[,colYield]
dim(X)
# checking neg
neg_cols_c = 0
for (col in 1:ncol(X)) {
  neg_cols_c = neg_cols_c + length(which(X[,col] < 0))
}
neg_cols_c

pos_c = 0
for (col in 1:ncol(X)) {
  pos_c = pos_c + length(which(X[,col] >= 0))
}

pos_c
```

```
pos_c + neg_cols_c
nrow(X)*ncol(X)

# stores indexes of max value
max = which(X == max(X), arr.ind = TRUE)
print(paste("Maximum value: ", X[max]))
print(max)

# stores indexes of min value
min = which(X == min(X), arr.ind = TRUE)
print(paste("Maximum value: ",X[min]))
print(min)

X[22,30]

head(X[,1])
head(X[,1]+10)

pos_val = 15
for (col in 1:ncol(X)) {
  X[,col] =X[,col] + pos_val
}
tail(X[,1])

X
dim(X)



ctrl <- trainControl(method = "cv", number = 3)
```

```
trainR <- createDataPartition(ChemicalManufacturingProcess$Yield, p=0.8, list=FALSE)

x.train <- X[trainR,]
y.train <- Y[trainR]

x.test <- X[-trainR,]
y.test <- Y[-trainR]
dim(x.train)

#ridge

ridgeGrid <- data.frame(.lambda = seq(0, 0.1, length = 20))

ridgeTune <- train(x = x.train,
                   y = y.train,
                   method = "ridge",
                   tuneGrid = ridgeGrid,
                   tuneLength = 20,
                   trControl = ctrl,
                   preProc = c("center", "scale"))

ridgeTune

predictions <- predict(ridgeTune, newdata = x.test)

postResample(predictions,y.test)
```

```
plot(ridgeTune, metric='RMSE')



imp<-varImp(enetTune, scale=FALSE)
plot(imp,top=20)
#enet

enetGrid <- expand.grid(.lambda = c(0,0.1, by=.01),
                        .fraction = seq(.05, 1, length = 20))
set.seed(100)
enetTune <- train(x = x.train,
                  y = y.train,
                  method = "enet",
                  tuneGrid = enetGrid,
                  trControl = ctrl,
                  preProc = c("center", "scale"))
enetTune
plot(enetTune, metric='RMSE')



prediction4 <- predict(enetTune, newdata = x.test)

postResample(prediction4,y.test)

plot(enetTune, metric='RMSE')



dev.off()
```

```
# lasso

library(elasticnet)
set.seed(1)
LassoGrid <- expand.grid(.lambda = 0, .fraction = seq(0.001, 1, length = 20))
set.seed(1)
lassoTune <- train(x.train, y.train, method = "enet",
                   tuneGrid = LassoGrid, trControl = ctrl,
                   preProc = c("center", "scale"))
lassoTune



prediction6 <- predict(lassoTune, newdata = x.test)

postResample(prediction6,y.test)

plot(lassoTune, metric='RMSE')



#lm

nzvpp <- nearZeroVar(X)
nzvpp
df <- X[-nzvpp]
dim(df)

highCorrpp <- findCorrelation(cor(X), cutoff=0.80)
df<- X[, -highCorrpp]
```

```
# preP <- preProcess(misX, method = c("BoxCox"))
# df <- predict(preP,misX)


library(caret)

set.seed(100)
trainR <- createDataPartition(Y, p=0.8, list=FALSE)
X.train <- df[trainR, ]
y.train <- Y[trainR]
X.test <- df[-trainR, ]
y.test <- Y[-trainR]



ctrl <- trainControl(method = "cv", number = 3)

#linear model

set.seed(100)
lm_model <- train(X.train, y.train, method = "lm",
                  trControl = ctrl, preProcess = c('center','scale'))
lm_model


prediction7 <- predict(lm_model, newdata = X.test)

postResample(prediction7,y.test)
```

```
xyplot(y.train ~ predict(lm_model),
      ## plot the points (type = 'p') and a background grid ('g')
      type = c("p", "g"),
      xlab = "Predicted", ylab = "Observed")
```

```
xyplot(resid(lm_model) ~ predict(lm_model),
      type = c("p", "g"),
      xlab = "Predicted", ylab = "Residuals")
```