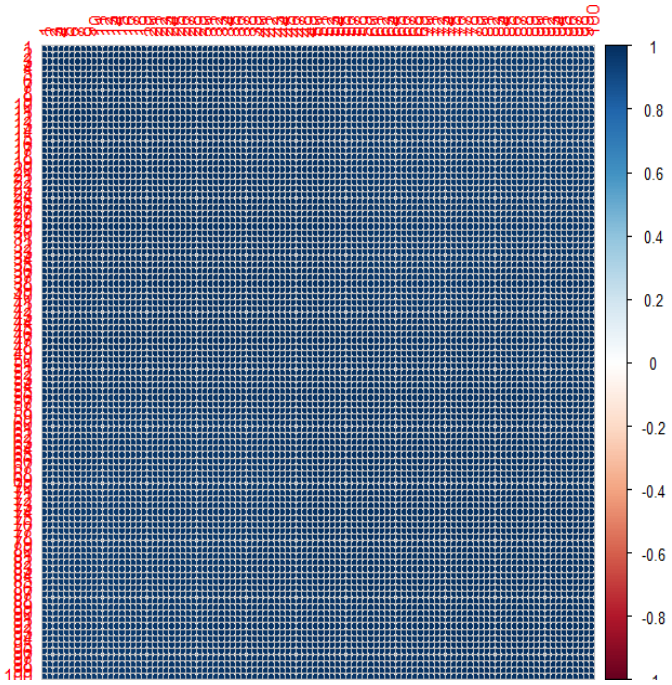
Homework 3
Applied predictive modelling

Supriya Bachal

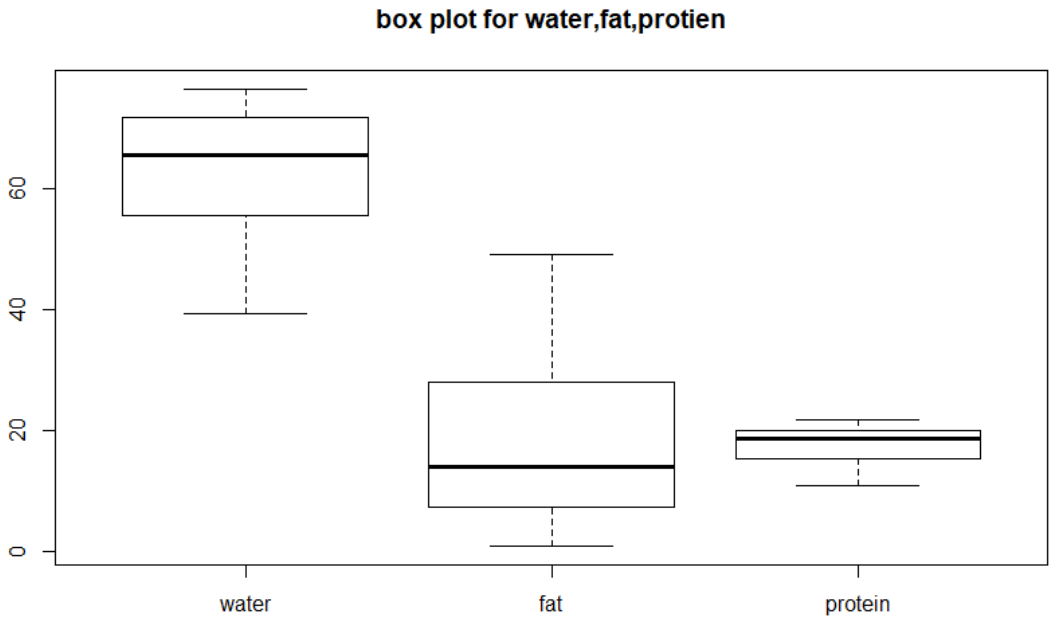
6.1. Infrared (IR) spectroscopy technology is used to determine the chemical Makeup of a substance. The theory of IR spectroscopy holds that unique Molecular structures absorb IR frequencies differently. In practice a spectrometer Fires a series of IR frequencies into a sample material, and the device Measures the absorbance of the sample at each individual frequency. This Series of measurements creates a spectrum profile which can then be used to Determine the chemical makeup of the sample material.
A Tecator Infratec Food and Feed Analyzer instrument was used to analyze 215 samples of meat across 100 frequencies. A sample of these frequency profiles is displayed in Fig. 6.20. In addition to an IR profile, analytical chemistry Determined the percent content of water, fat, and protein for each sample.
If we can establish a predictive relationship between IR spectrum and fat Content, then food scientists could predict a sample's fat content with IR Instead of using analytical chemistry. This would provide costs savings, since Analytical chemistry is a more expensive, time-consuming process:

(a) Start R and use these commands to load the data:
> library(caret)
> data(tecator)
> # use ?tecator to see more details
The matrix absorp contains the 100 absorbance values for the 215 samples,
While matrix endpoints contains the percent of moisture, fat, and protein
In columns 1–3, respectively

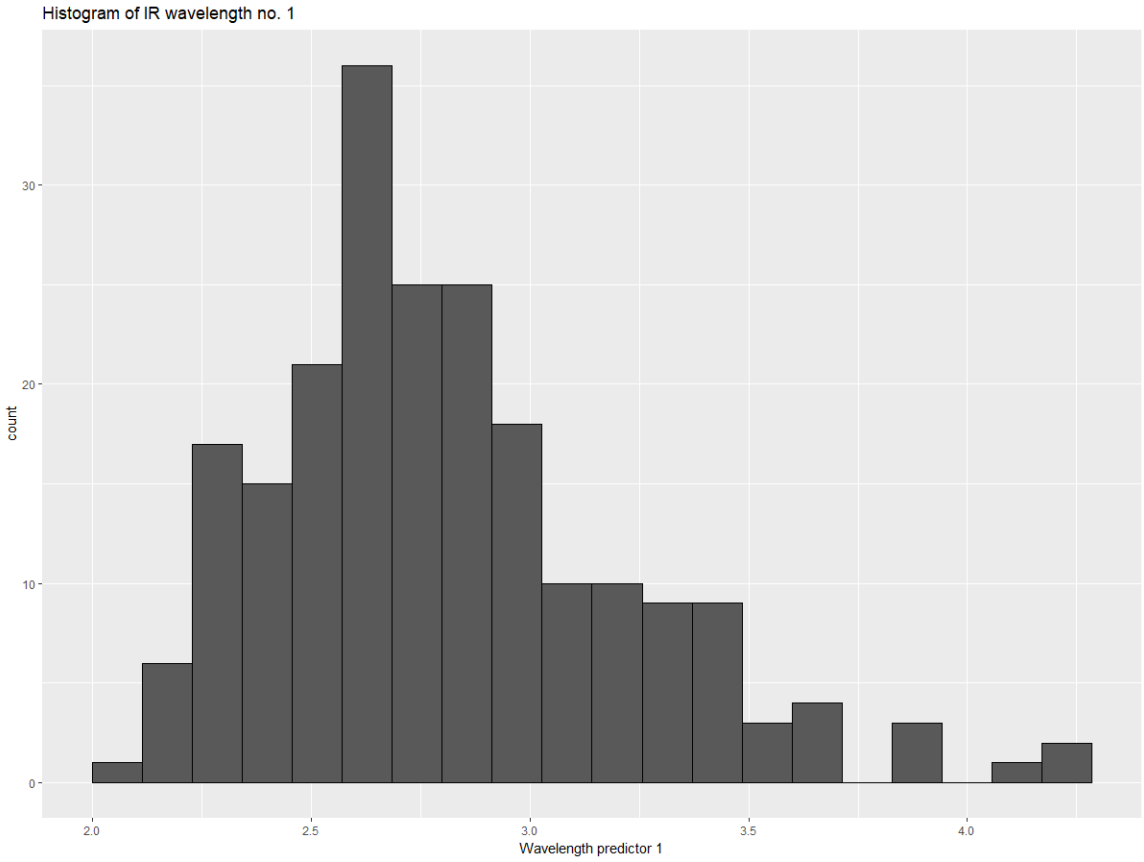
As indicated by the theory of IR, the remarkable structure of atomic retains the distinctive range. Thus, since the fat assimilates particular range given the hypothesis, one can anticipate how much fat is on the nourishment in light of the prescient model, which will spare the time and cost.



The initial Exploration with the dataset confirmed that the all the predictor values in the absorp were very highly correlated with each other. Most Absorption values lie between 2 and 5.



The Box plots for the endpoints leads us to believe that most of the meats have fat contents in the range of 5 to 30. With the median being at around 10. The minimum value for fat is 0.9 and the maximum value of fat is 76.6.We can also interpret the absence of outliers, which means we do not need to use any outlier preprocessing for this data set.

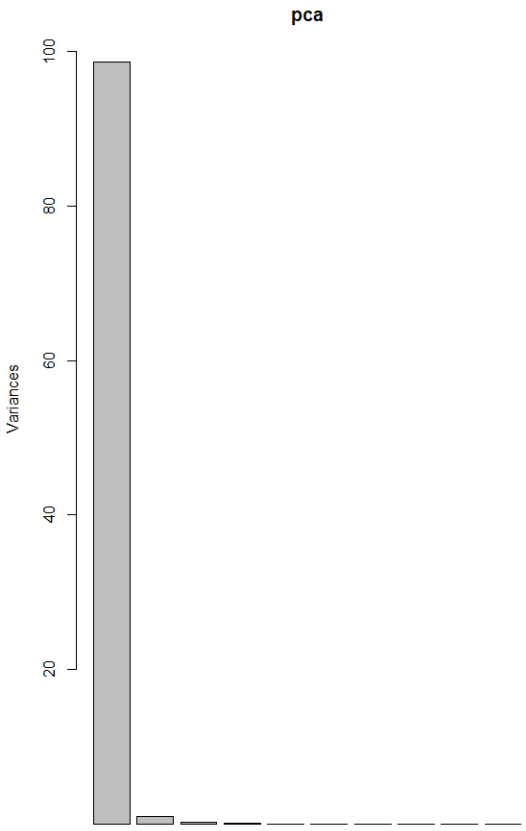


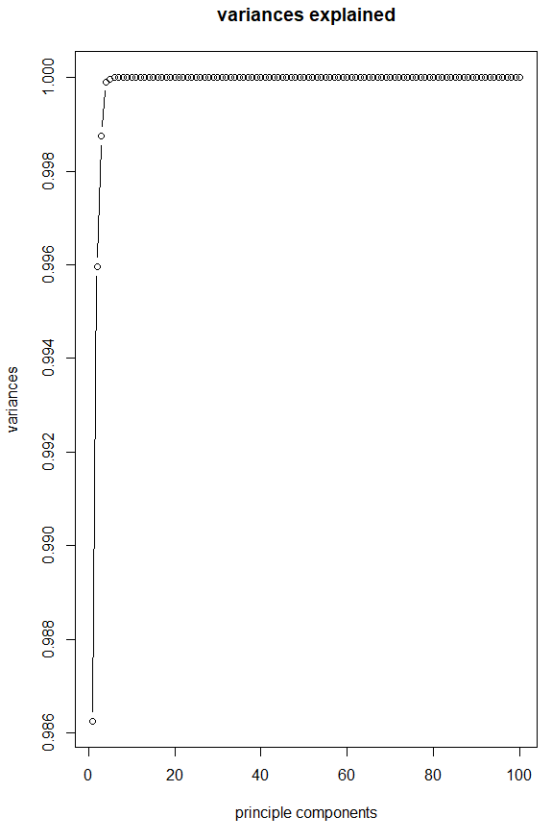
The histogram of the wavelength shows us the there is a skewed pattern in the predictors. The predictors are positively skewed however since our models will not be heavily affected by this we do not transform these predictors.

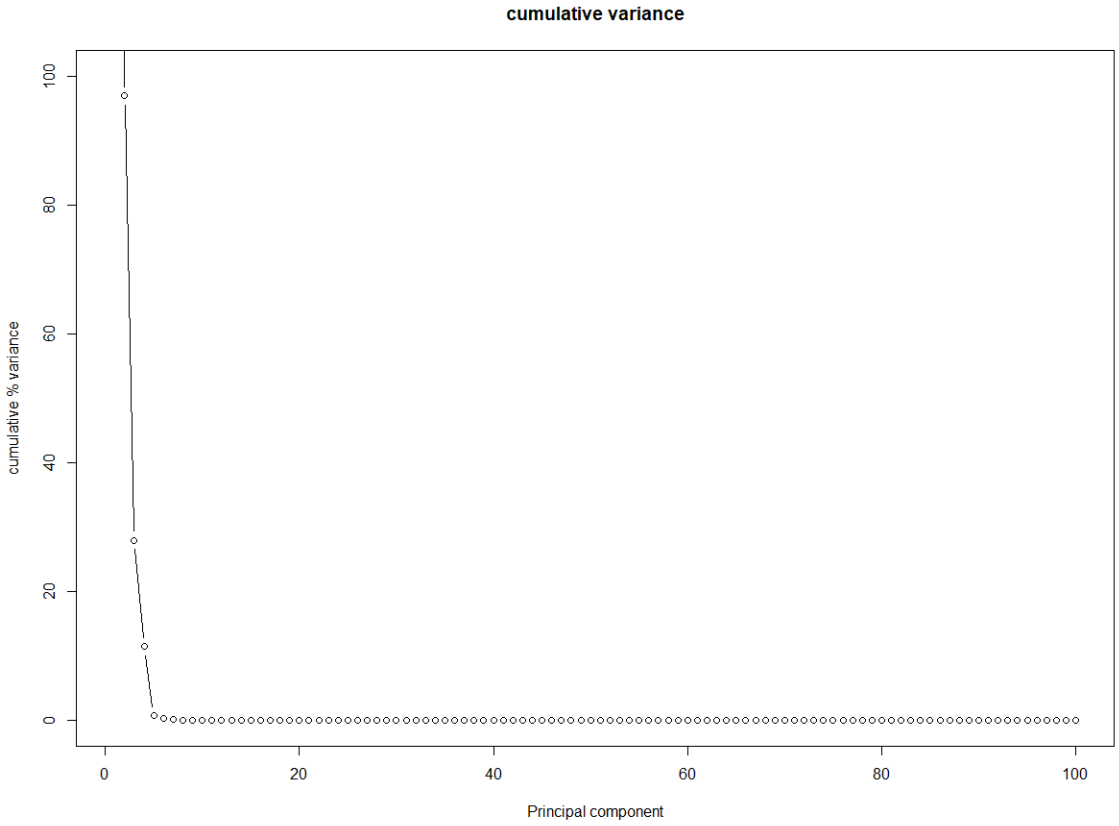
(b) In this example the predictors are the measurements at the individual frequencies.
Because the frequencies lie in a systematic order (850–1,050 nm), the predictors have a high degree of correlation. Hence, the data lie in a smaller dimension than the total number of predictors (100). Use PCA to determine the effective dimension of these data. What is the effective dimension?

To carry out the PCA we use the prcomp function with the caret package and center and scale the data.

pca <- prcomp(absorp,center = TRUE,scale = TRUE)





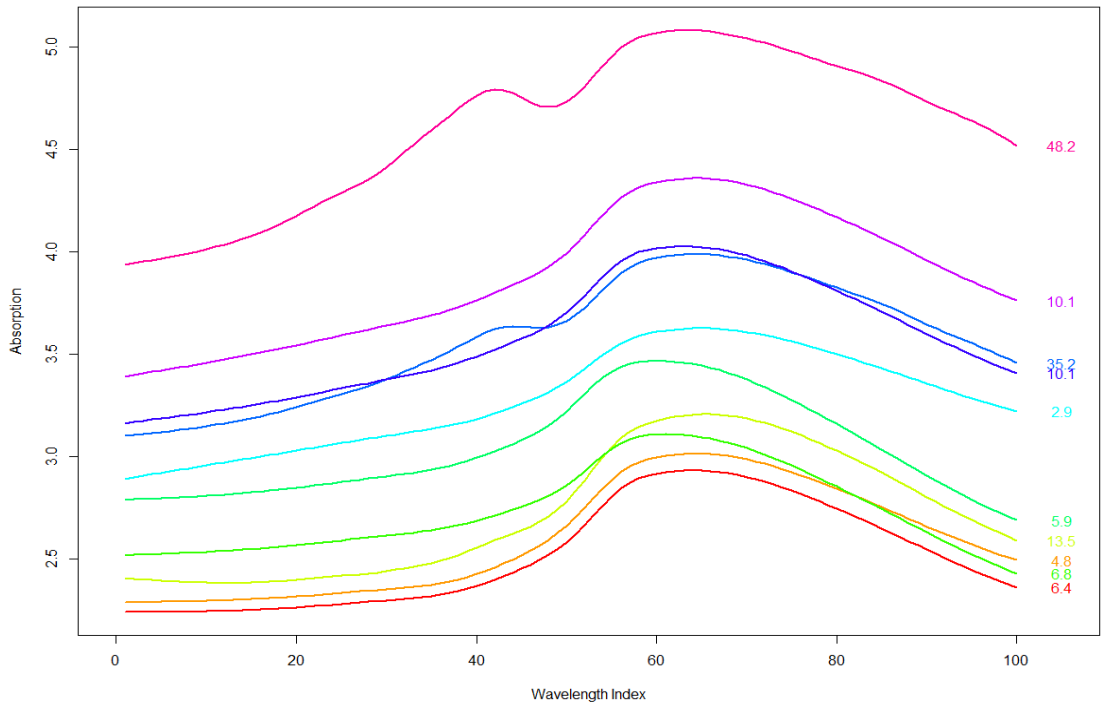


Given the plots, the first principal components explains most of the variance in the data. In addition, after 5 components, the variance explained reached 100%. It means there is high cor linearity among variables. The first plot for visualization was the one with the variance explained however, I could not clearly see the no of components as they were clustered together so I made the second visualization to add clarity. The histogram confirms that after 5 components the variance reaches 100 percent. The summary for the first 5 components is as follows:

	PC1	PC2	PC3	PC4	PC5
Standard deviation	9.931072	0.9847361	0.5285114	0.3382748	0.08037979
Proportion of Variance	0.986260	0.0097000	0.0027900	0.0011400	0.00006000
Cumulative Proportion	0.986260	0.9959600	0.9987500	0.9999000	0.99996000

As seen from the summary 98.62% of the variance is explained by the first component itself and hence it is the major component for the analysis with reference to the linear combination of predictors. Additionally we could also consider nonlinear summarizations of data.

To understand the relationship between the absorption values vs the wavelength, a subset of 10 random values was made and the corresponding, fat values were taken. Every color represents an absorption value and a corresponding fat content between 48.2 and 2.9.



To further investigate the data before splitting it I wanted to see how the fat percentages are distributed within the samples. Here is the table showing the number of instances for the different fat percentages

numbers	Freq	
1	0-10	77
2	10-20	61
3	20-30	36
4	30-60	41

c) Split the data into a training and a test set, pre-process the data, and build each variety of models described in this chapter. For those models with tuning parameters, what are the optimal values of the tuning parameter(s)?

The data is not suitable for cross validation since it has only 215 samples. So we decided to use "LGOCV" with 5 repeats. To split the data we use the train control from the caret library.

ctrl <-trainControl(method = "LGOCV", number = 5,p=.75)

We use 75% as training set and 25% as testing set.

1. The first model built was a Linear Regression model.

   To build this model a threshold value of 90% correlation was chosen to select the predictors. We observe from the summary of the model that only one independent predictor was used to build the model. The performance of the model can be evaluated from its RMSE value, which is 11.1.

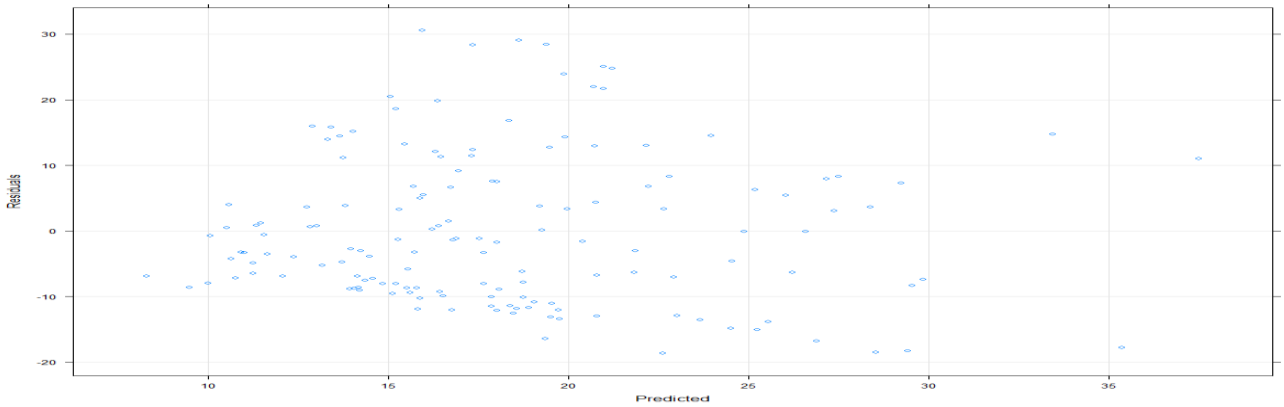
Linear Regression

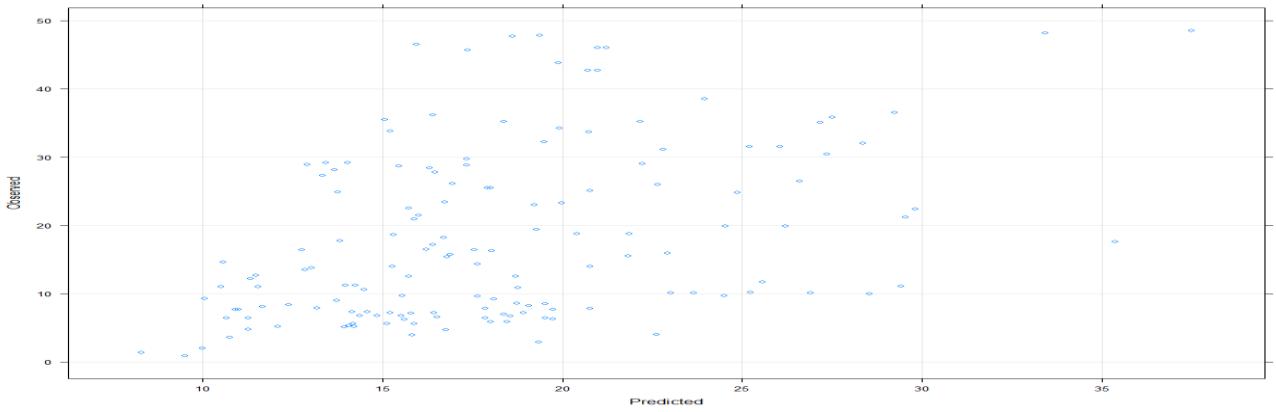
163 samples
1 predictor

No pre-processing
Resampling: Repeated Train/Test Splits Estimated (5 reps, 75%)
Summary of sample sizes: 124, 124, 124, 124, 124
Resampling results:

RMSE	Rsquared	MAE
11.11397	0.2304765	9.14265

These are the two plots to evaluate the performance of the model. One of the plots is the observed value vs the predicted value the other one is that of the residuals. As we can see there is no pattern in the observed vs predicted values which means that therew is no or very little correlation between them. Which indicates that the model did not perform very well. As for the residuals, there are more residuals in the bottom portion and higher valued residuals in the top portion however; the residuals are evenly distributed.





2. RLM model is usually used for dealing with outliers and homoscedastic data ,even though our dataset has none of these properties since the question demanded to use all models we implement RLM as well.
   For rlm we cannot have a singular predictor covariance matrix thus we preprocess with PCA:

set.seed(1)

rlm_model=train(Train[,101],Train[,101],method="rlm",preProcess=c("pca"),trControl=ctrl)

Summary of the RLM model is as follows:

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were intercept = TRUE and psi = psi.hampel.
Robust Linear Model

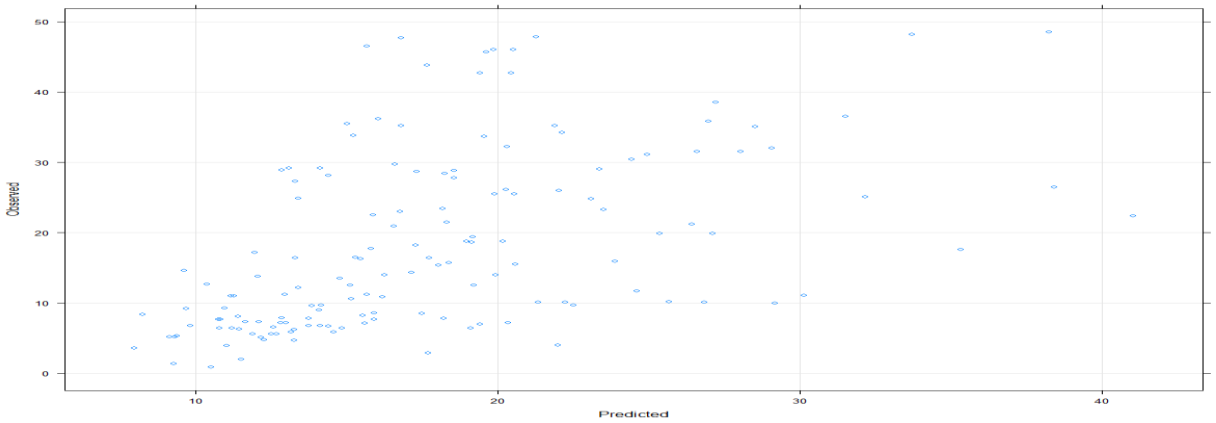
163 samples
100 predictors

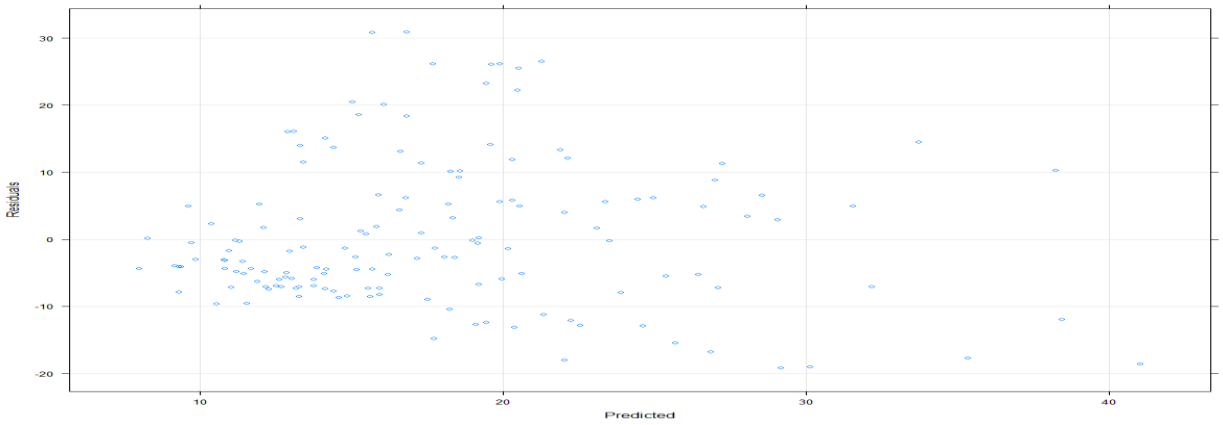
Pre-processing: principal component signal extraction (100), centered (100), scaled (100)
Resampling: Repeated Train/Test Splits Estimated (5 reps, 75%)
Summary of sample sizes: 124, 124, 124, 124, 124
Resampling results across tuning parameters:

intercept	psi	RMSE	Rsquared	MAE
FALSE	psi.huber	21.30030	0.3254849	18.620285
FALSE	psi.hampel	21.31960	0.3178190	18.646044
FALSE	psi.bisquare	21.29260	0.3249492	18.573792
TRUE	psi.huber	10.61796	0.3248411	8.141147
TRUE	psi.hampel	10.55503	0.3200146	8.326236
TRUE	psi.bisquare	10.64477	0.3246944	8.129551

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were intercept = TRUE and psi = psi.hampel.

Which indeed means that the RMSE is 10.5 for this model. It establishes that it performs a little better than the linear model.





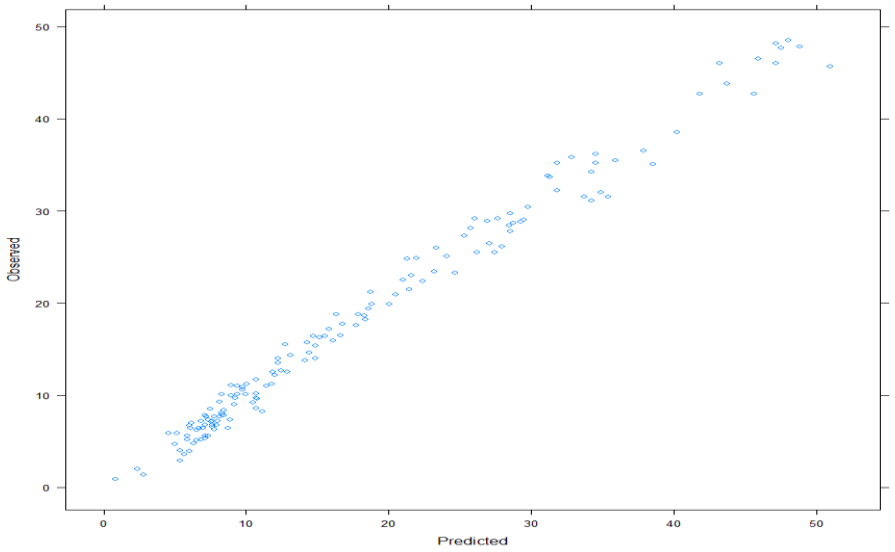
3. The PCR model was implemented on this data using the following code:

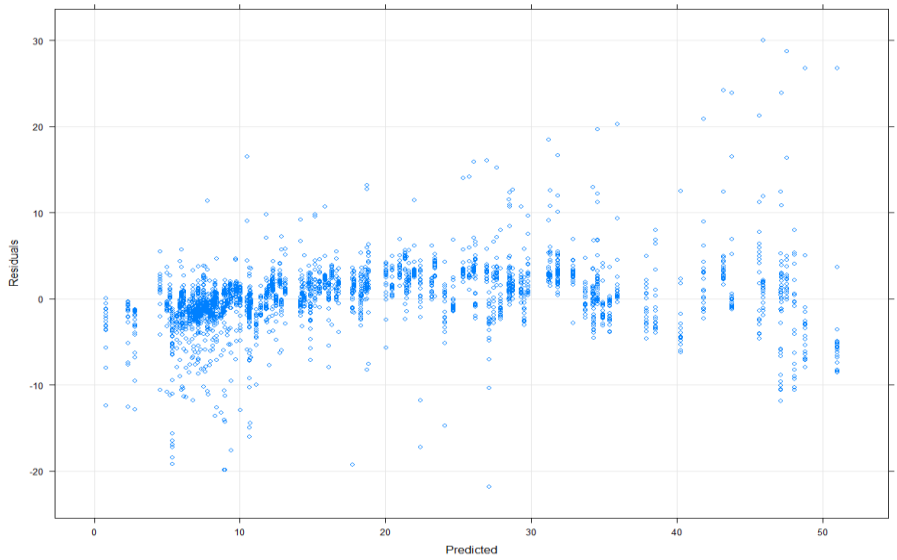
PCR <- train(y =Train[,101] , x = Train[,-101], method = "pcr",trControl = ctrl,tuneLength=25)

The tune length was chosen to be 25 .The best RMSE was 2.58 for 16 predictors. T he plots shows that there is a strong pattern in the observed vs predicted values, th us we can say that the model did a good job at predicting the values. The residual plot also shows appropriate behavior.

ncomp	RMSE	Rsquared	MAE
1	10.627729	0.2993532	8.562489
2	10.640628	0.2992878	8.519212
3	8.113044	0.5892546	6.096007
4	4.343164	0.8884487	3.368914
5	3.498749	0.9292436	2.695119
6	3.186437	0.9412674	2.367312
7	3.136030	0.9432869	2.319718
8	3.122363	0.9447836	2.256299
9	3.077401	0.9462370	2.269609

10	3.074187	0.9461419	2.301295
11	2.834915	0.9525326	2.199592
12	2.913620	0.9488626	2.309188
13	2.925539	0.9487389	2.304019
14	2.893150	0.9503614	2.276475
15	2.833587	0.9534902	2.200531
16	2.583810	0.9617547	1.997992
17	2.752988	0.9580129	2.055830
18	2.727945	0.9581410	2.041521
19	2.960649	0.9517838	2.107726
20	2.954541	0.9519300	2.108689
21	2.934897	0.9523863	2.095458
22	2.816743	0.9542456	2.040074
23	2.910588	0.9492736	1.978043
24	2.894319	0.9500870	1.999269
25	2.779262	0.9544376	1.954891

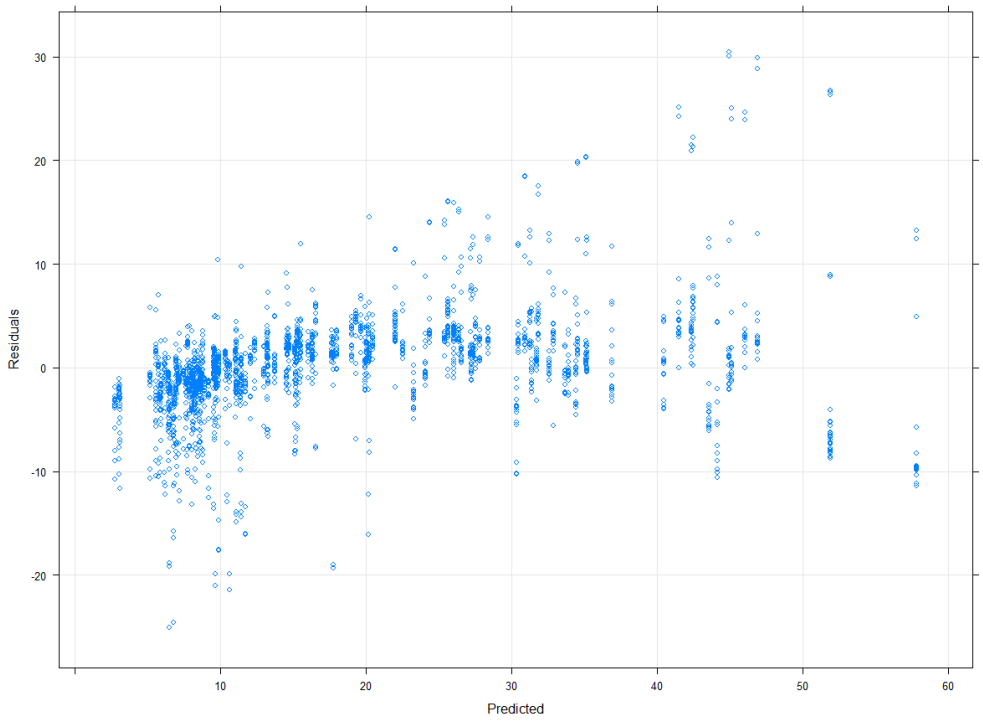


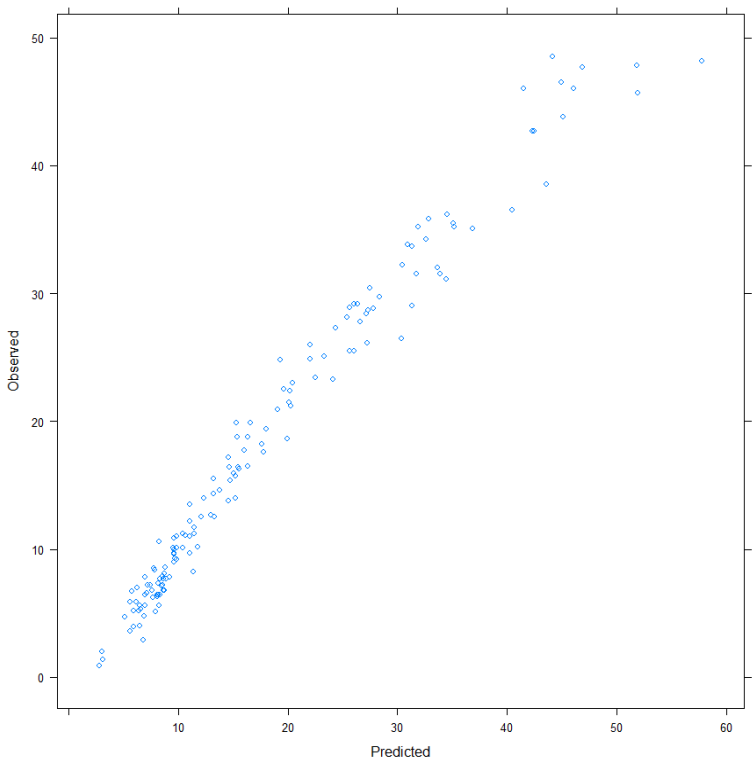


4. PLS Model was implemented using the following code:
   PLS <- train(y =Train[,101] , x = Train[,-101], method = "pls",preProcess=c("center","scale"),trControl = ctrl,tuneLength=25)

ncomp	RMSE	Rsquared	MAE
1	10.643722	0.2969975	8.584599
2	6.773286	0.7101878	5.012824
3	5.572418	0.8064884	4.115805
4	4.309526	0.8904063	3.351301
5	3.198210	0.9414655	2.339831
6	3.190998	0.9412995	2.353225
7	3.064207	0.9470725	2.206677
8	3.087945	0.9457460	2.261480
9	2.952283	0.9492857	2.288731
10	2.825851	0.9516280	2.246056
11	2.870393	0.9499581	2.290753
12	2.988820	0.9486453	2.243210
13	2.708144	0.9594643	2.022789
14	2.715399	0.9585977	2.029196
15	2.757921	0.9574485	1.993409
16	2.928037	0.9520755	2.015474
17	2.663415	0.9588269	1.902652
18	2.471738	0.9654171	1.810884
19	2.461300	0.9649078	1.780286
20	2.463416	0.9658904	1.760537
21	2.685104	0.9607576	1.878862

22	2.743685	0.9587711	1.850678
23	2.777816	0.9576040	1.873661
24	2.735002	0.9577879	1.868724
25	2.937455	0.9563290	1.954361

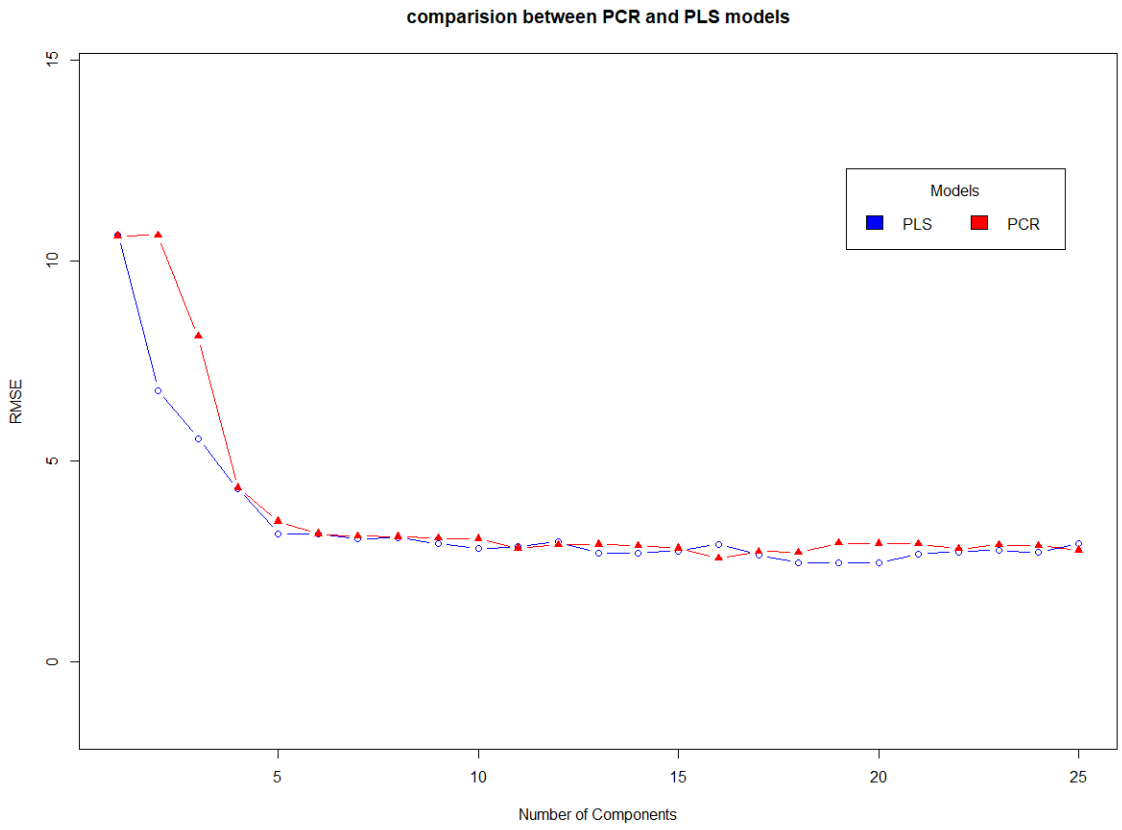

Residuals
Predicted
30
20
10
0
-10
-20
10
20
30
40
50
60


Observed
Predicted
50
40
30
20
10
0
10
20
30
40
50
60

The PLS model was performing similar to the PCR model, as indicated by the plots.

The best value for the RMSE with the PLS model was 2.46 for 19 components.

Next we compare the PCR and PLS model for better performance.



The Graph above shows that the model PCR and PLS perform very similar in terms of RMSE values as 2.58 & 2.46 however the PCR gives a less complex model with 16 predictors as oppose to the 19 predictors use by PLS.

5. Ridge Regression - penalise square of coefficient
   Code to implement this model is :
   RM<- train(y =Train[,101],x = Train[,-101],method = "ridge",trControl = ctrl,tuneGrid = ridgeGrid,preProcess=c("center","scale"))
   The ridge grid was set to :
   ridgeGrid <- data.frame(.lambda = c(0,0.0001, 0.001, 0.01,0.1))

The Ridge regression uses lambda values between 0 to 0.1

Ridge Regression

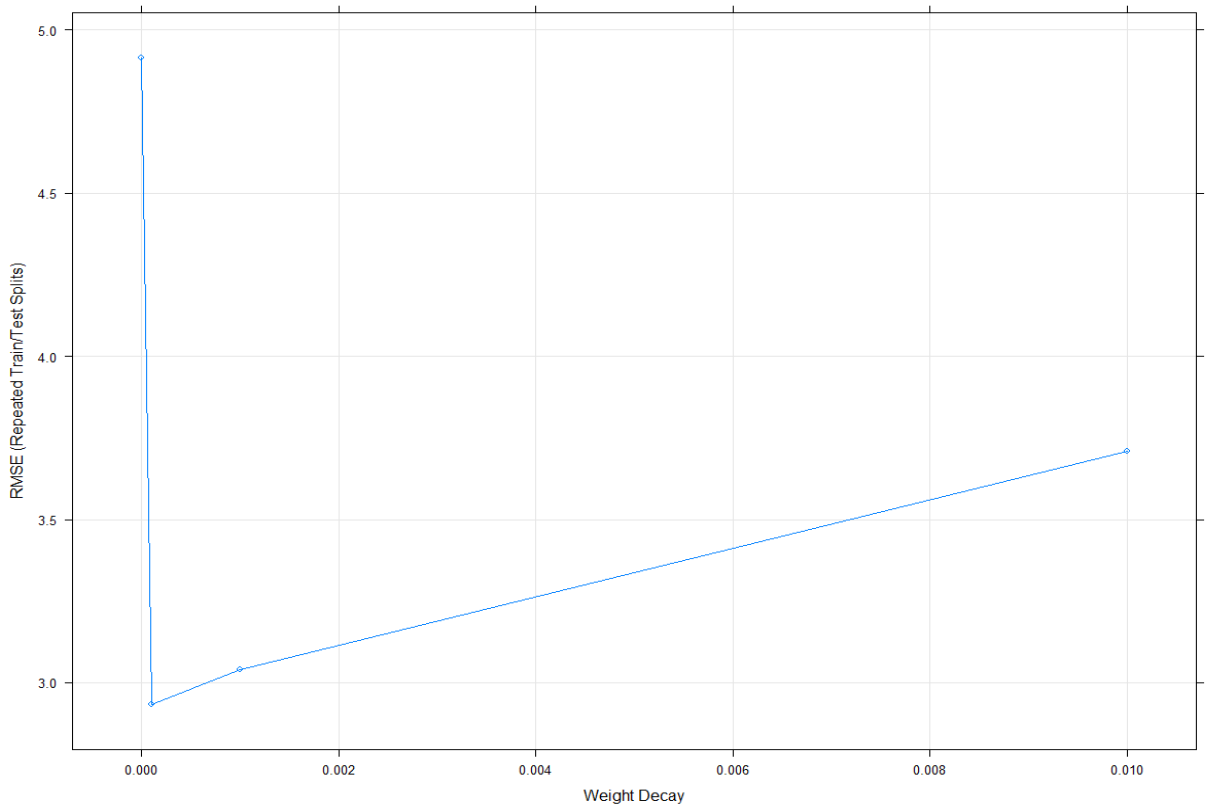
163 samples
100 predictors

Pre-processing: centered (100), scaled (100)
Resampling: Repeated Train/Test Splits Estimated (5 reps, 75%)
Summary of sample sizes: 124, 124, 124, 124, 124
Resampling results across tuning parameters:

lambda	RMSE	Rsquared	MAE
0e+00	4.914019	0.8882616	2.902447
1e-04	2.934167	0.9558180	2.267345
1e-03	3.040376	0.9537380	2.384255
1e-02	3.710175	0.9246771	3.123035
1e-01	4.832154	0.8569034	3.834798

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was lambda = 1e-04.

An interesting observation while implementing the model was that as the tuning parameter is set finer the value of the best lambda moves toward 0. To verify this I implemented a model with Lambda grid extending to 0.00001 and the best lambda value was 0.00001 in this case. There is no static value for lambda it moves closer to 0, as we put in finer parameters.



6. Lasso Model :
   The following code was used to implement Lasso model:

```
set.seed(1)

lasso_grid <- lasso_grid <- expand.grid(.fraction=seq(0,0.1, length=20))

Lasso <- train(y = Train[,101],x = Train[,-101], method = "lasso", trControl = ctrl,

               tuneGrid = lasso_grid)
```

The grid parameters were selected in the form of fraction parameters between 0 to 0.1.

The lasso

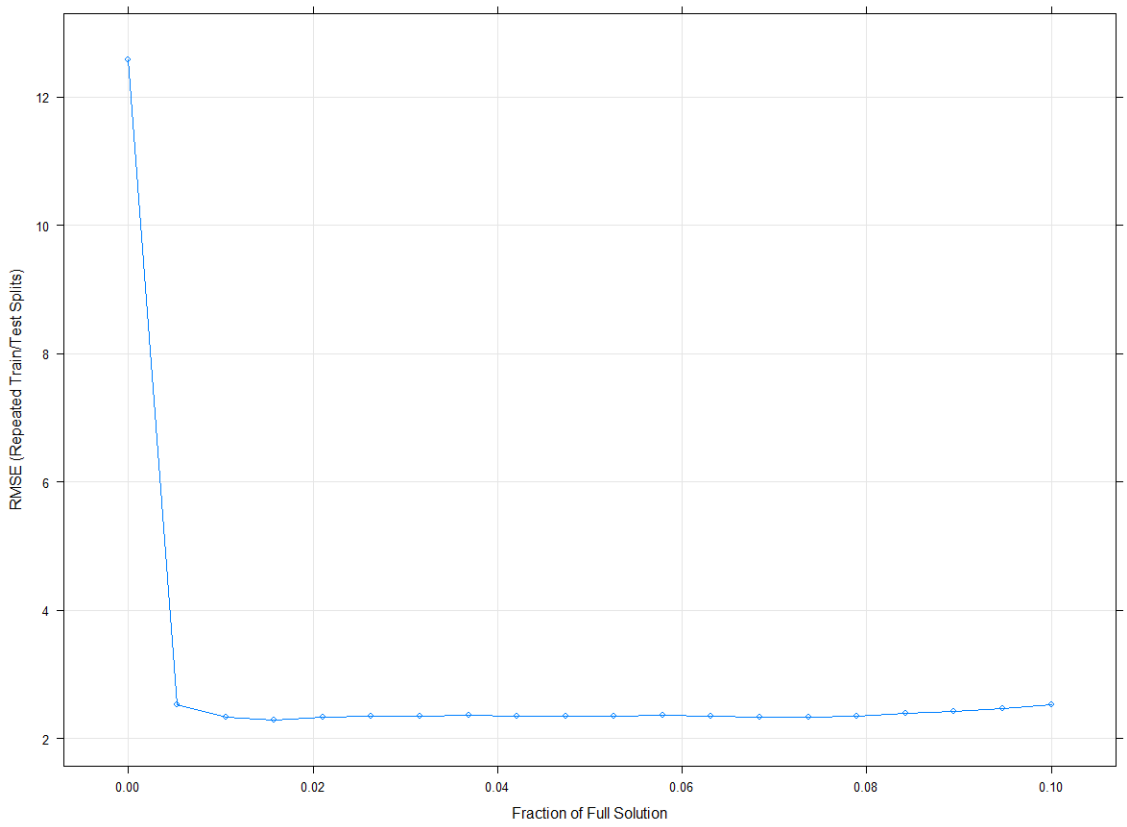
163 samples
100 predictors

No pre-processing
Resampling: Repeated Train/Test Splits Estimated (5 reps, 75%)

Summary of sample sizes: 124, 124, 124, 124, 124
Resampling results across tuning parameters:

fraction	RMSE	Rsquared	MAE
0.000000000	12.574961	NaN	10.696237
0.005263158	2.526463	0.9622234	1.959691
0.010526316	2.335365	0.9682374	1.794543
0.015789474	2.294162	0.9698386	1.740583
0.021052632	2.329030	0.9691821	1.720583
0.026315789	2.347864	0.9686924	1.691038
0.031578947	2.354168	0.9684607	1.681249
0.036842105	2.358874	0.9684357	1.687682
0.042105263	2.355805	0.9687439	1.694593
0.047368421	2.348086	0.9690578	1.696465
0.052631579	2.347767	0.9691707	1.696021
0.057894737	2.361642	0.9688745	1.698470
0.063157895	2.347330	0.9695324	1.694744
0.068421053	2.331335	0.9701243	1.686742
0.073684211	2.332103	0.9702683	1.681896
0.078947368	2.353445	0.9699895	1.696900
0.084210526	2.391985	0.9692924	1.717262
0.089473684	2.428490	0.9684846	1.731387
0.094736842	2.472501	0.9675176	1.744057
0.100000000	2.523552	0.9662329	1.757509

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was fraction = 0.01578947.



7. Elastinet model (combination of ridge and Lasso):
   The code used for implementing the Elastinet model is as follows:
   enet_grid <- expand.grid(.fraction=seq(0,0.1, length=20), .lambda = c(0, 0.0001, 0.001, 0.01))
   Enet <- train(y = Train[,101],x = Train[,-101], method = "enet", trControl = ctrl, tuneGrid = enet_grid)

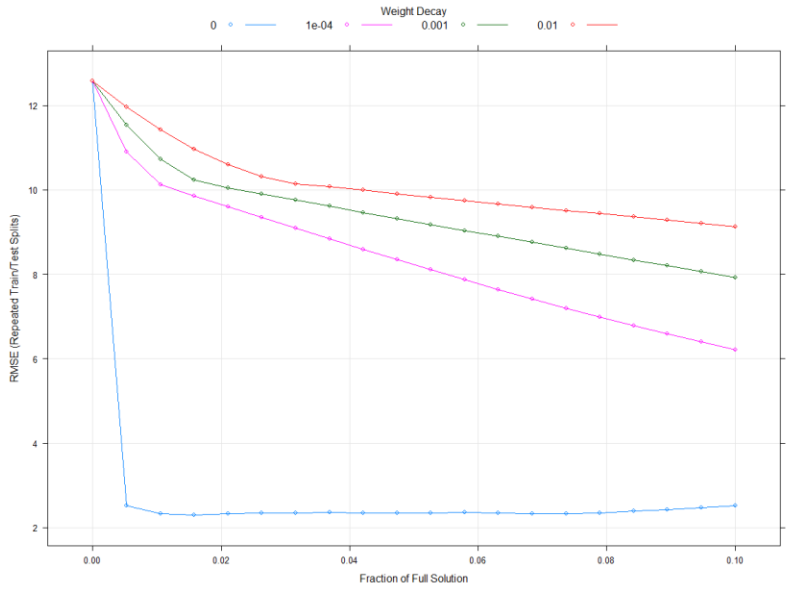
A grid function with fraction between 0.001 to 0.1 and lambda values between 0 to 0.0001 are used.

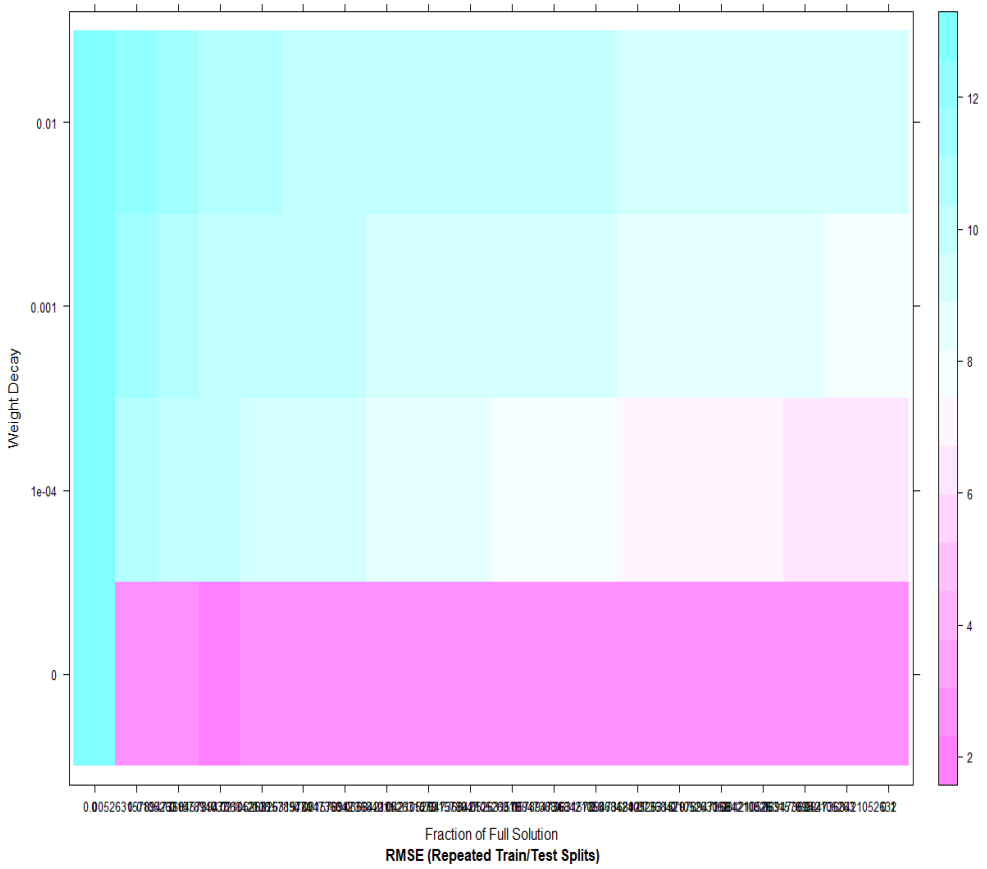
The model generates a grid for variety of combinations of these values(see appendix) but the best value selected are :

lambda	fraction	RMSE	Rsquared	MAE
0e+00	0.000000000	12.574961	NaN	10.696237
0e+00	0.005263158	2.526463	0.9622234	1.959691
0e+00	0.010526316	2.335365	0.9682374	1.794543
0e+00	0.015789474	2.294162	0.9698386	1.740583

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were fraction = 0.01578947 and lambda = 0.

Since the Lambda value for this model is 0 we could consider this model to be a las so model in practice.





The above representation shows the enet model performance for various parameters of lambda and fraction.

Best tuning parameters for each of the models are represented below:

	Ridge	Lasso	Enet
lambda	0.00001	NA	0.00000000
fraction	NA	0.01578947	0.01578947

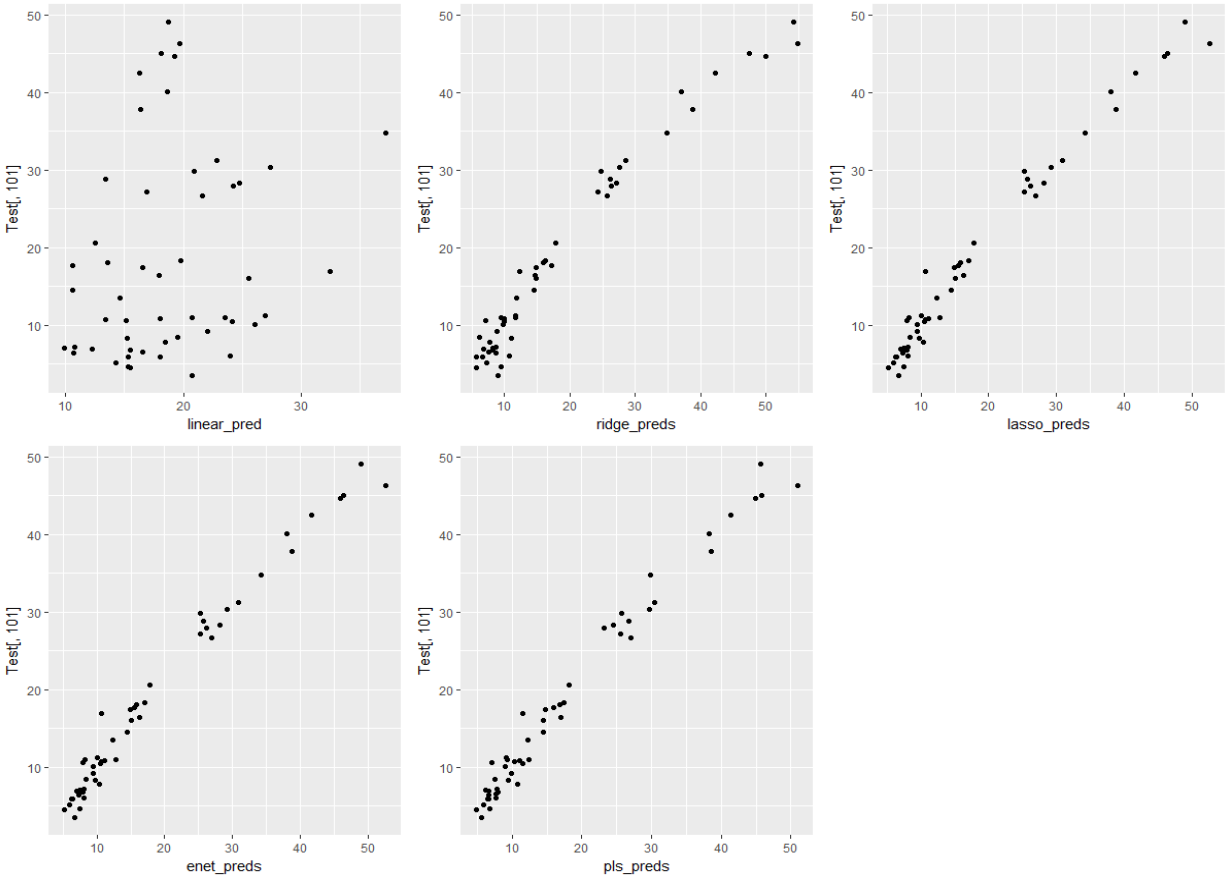
The table and graph for Enet agree with each other.

d)Which model has the best predictive ability? Is any model significantly better or worse than the others?

As the plot shows, the various models do a good job at predicting, except linear regression model.

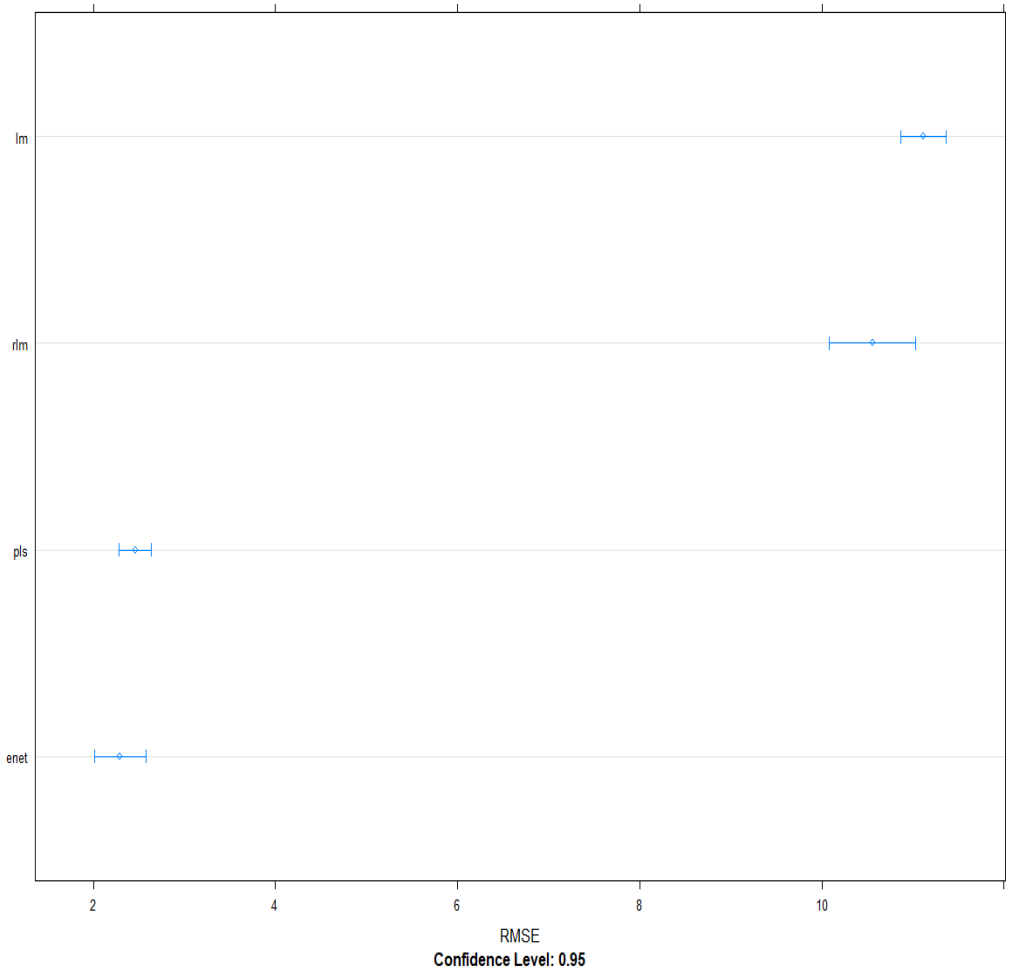
Linear	PLS	Ridge	Lasso	Enet
0.2154802	0.9868296	0.9782466	0.9860839	0.9860839

The above table shows the values for the correlation between the predicted and observed values for each of the models. The Enet as well as the PLS model seems to be the best performing model amongst all the models. The PLS model seems to be the best model; this could be due to the nature of collinearity of the dataset. The collinearity may also be the reason for the worst performing model name i.e. linear regression.



(e) Explain which model you would use for predicting the fat content of a Sample

The plot shows the confidence intervals for the various models that we have implemented. To make the call between Enet and PLS we can refer to the R squared values for Elastic net it is 0.9698386 and for PLS it is 0.9649078.



Question 2

Developing a model to predict permeability (see Sect. 1.4) could save significant resources for a pharmaceutical company, while at the same time more

rapidly identifying molecules that have a sufficient permeability to become a drug

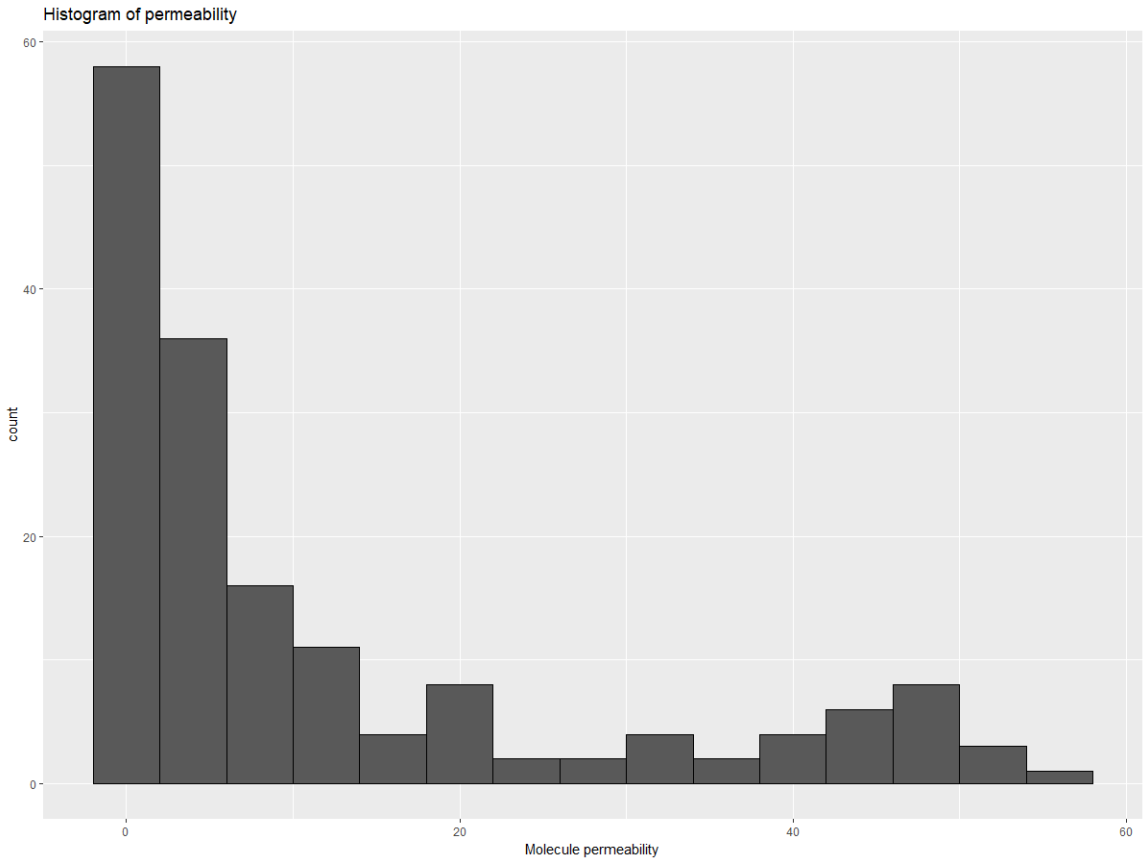
(a) Start R and use these commands to load the data:

The matrix fingerprints contains the 1,107 binary molecular predictors for the 165 compounds, while permeability contains permeability response

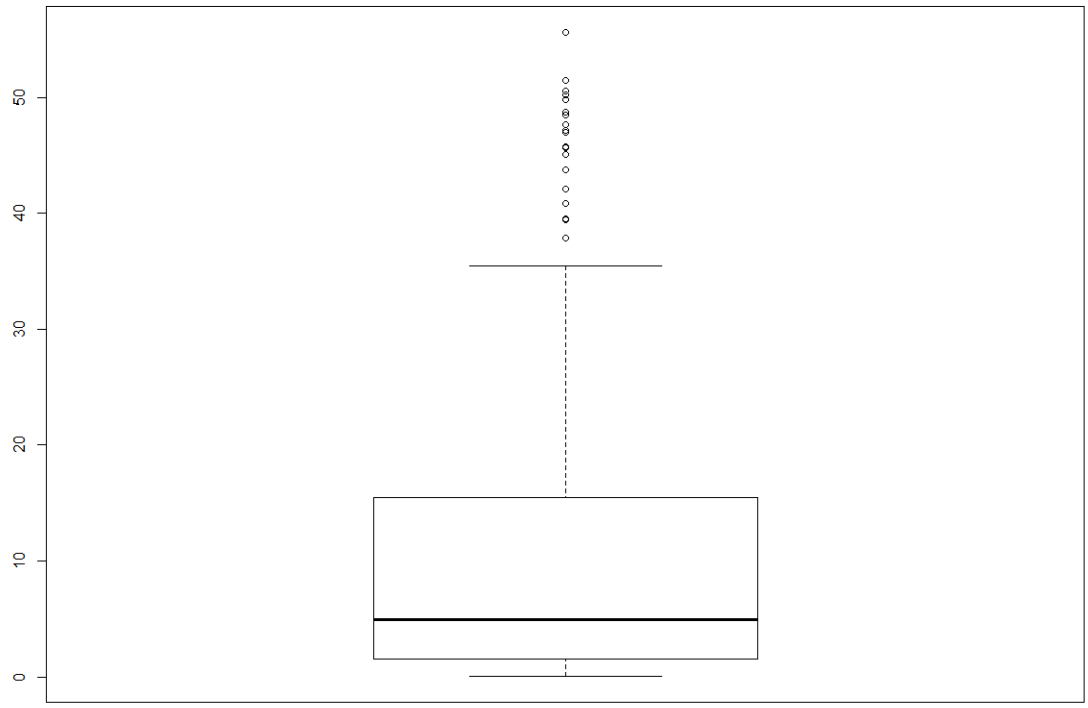
```
> library(AppliedPredictiveModeling)
> data(permeability)
```

Distribution of the permeability is as follows.

The histogram looks very right skewed.



The boxplot shows that there are many outliers in permeability.The median is at 5.



b)The fingerprint predictors indicate the presence or absence of substructures of a molecule and are often sparse meaning that relatively few of the molecules contain each substructure. Filter out the predictors that have low frequencies using the nearZeroVar *function from the* caret *package. How many predictors are left for modeling?*

The dimensions for fingerprint are :

Rows:165 columns:1107 there are no missing values in the dataset.

The predictors of the dataset are all in the form of 1 or zero meaning they are binary predictors.

Finding the near zero variance code:

```
ZC <- nearZeroVar(fingerprints, saveMetrics = T); ZC
```

```
freqRatio      percentUnique zeroVar   nzv
X1  3.342105       1.212121   FALSE    FALSE
```

```
X2   3.459459      1.212121   FALSE    FALSE
X3   3.714286      1.212121   FALSE    FALSE
X4   3.714286      1.212121   FALSE    FALSE
X5   3.714286      1.212121   FALSE    FALSE
X6   1.229730      1.212121   FALSE    FALSE
```

Next to find the values having near zero variance we use the following code:

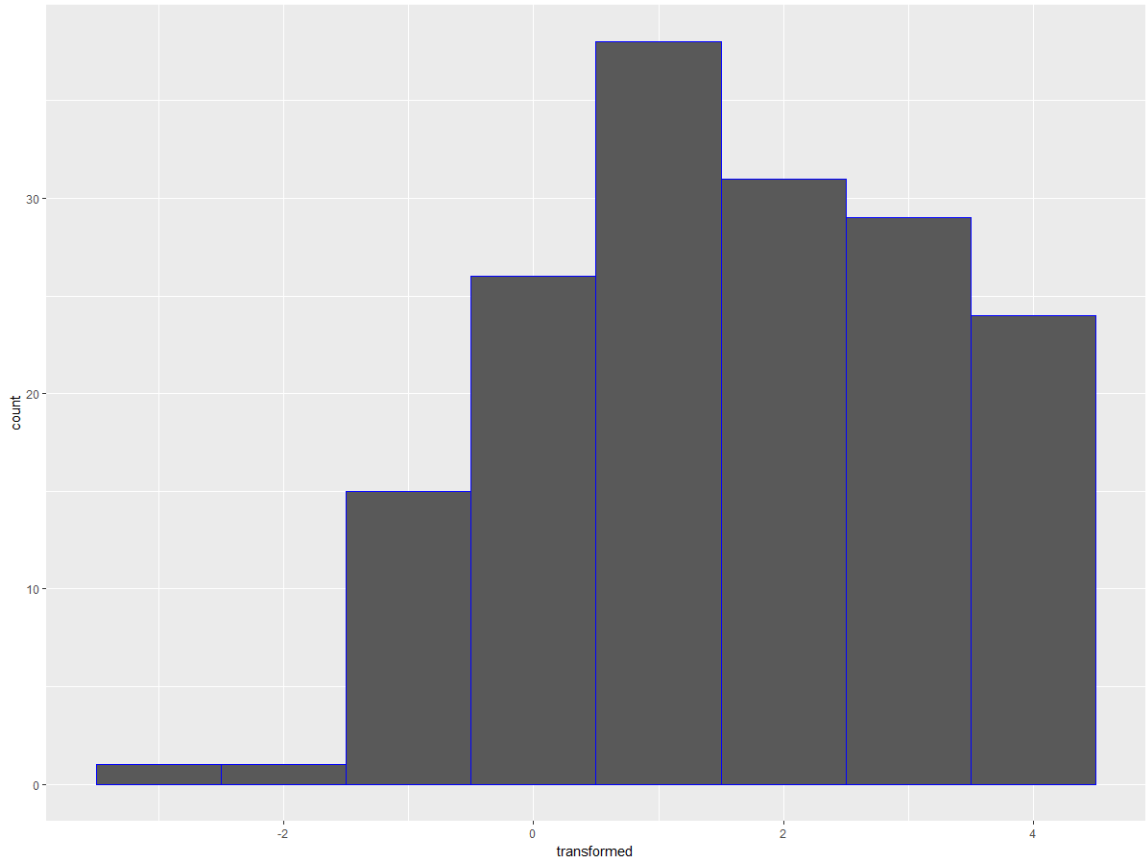
ZC[ZC[, "nzv"] == TRUE,]

There are 719 predictors having near zero variance.

If we eliminate all the predictors having zero or near zero variance we are left with 388 predictors.

c)Split the data into a training and a test set, pre-process the data, and tune a PLS model. How many latent variables are optimal and what is the corresponding resampled estimate of R2?

To transform the variables to get rid of the skew the BOX_COX transformation is used.



The represented histogram is of permeability after transformation. The skewness is mitigated as shown.

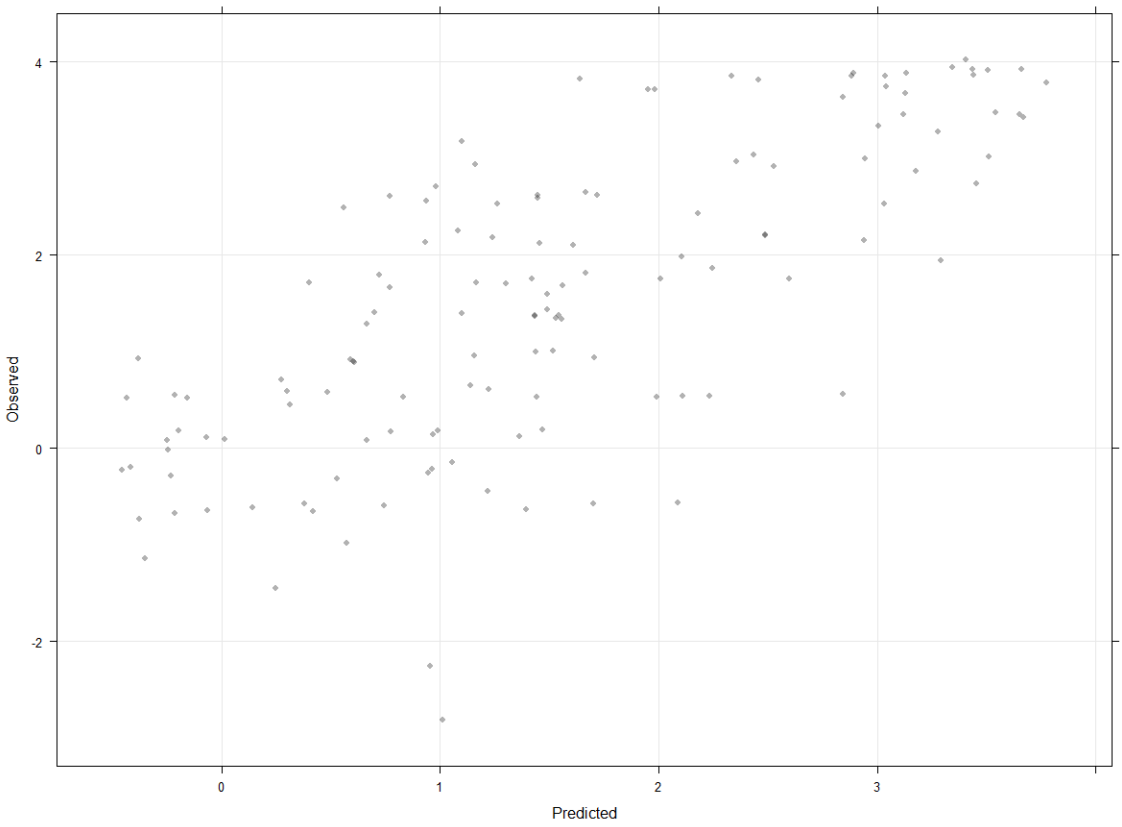
The code:

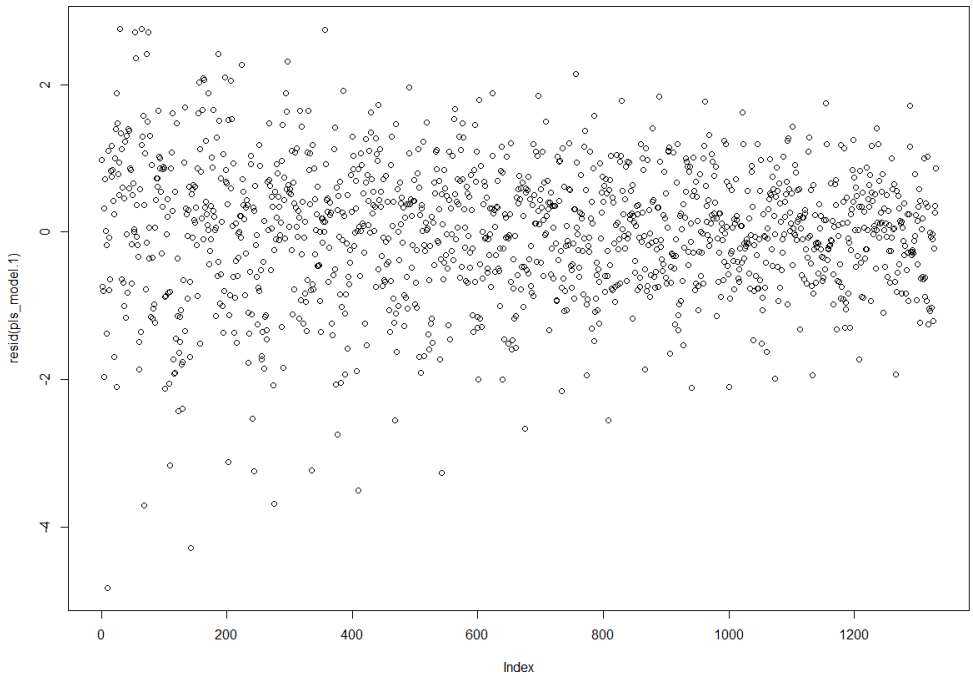
pls_model.1 <- train(y = Y_train,x = training,method = "pls",preprocess=c("center,scale"),metric = "Rsquared,trControl = ctrl)

10 components were selected for the PLS model.

The variances explained for each of the predictors is as follows:

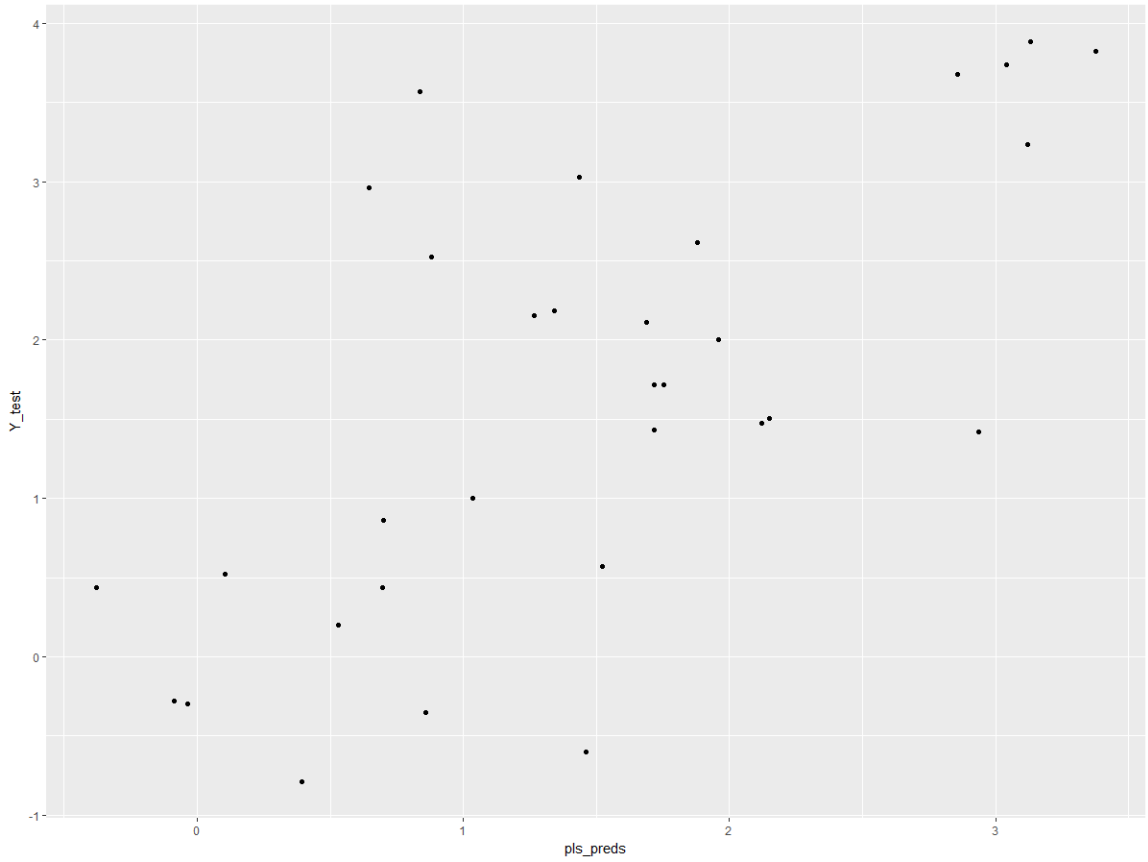
```
Partial Least Squares

133 samples
388 predictors

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 5 times)
Summary of sample sizes: 107, 105, 108, 106, 106, 105,
Resampling results across tuning parameters:

  ncomp  RMSE      Rsquared   MAE
   1     13.87956  0.2553456  10.393159
   2     12.28987  0.4140507   8.610179
   3     12.04756  0.4382266   9.018351
   4     12.19326  0.4382745   9.244138
   5     12.01914  0.4563073   9.061951
   6     11.81938  0.4765397   8.787029
   7     11.86513  0.4787456   8.760796
   8     11.73009  0.4950284   8.750518
   9     11.88617  0.4930332   8.859533
  10     11.92773  0.4970881   8.879051
  11     12.01408  0.4955018   8.854646
  12     12.05105  0.4965128   8.913338
  13     12.13350  0.4935610   9.020930
  14     12.16737  0.4910860   9.102319
  15     12.20607  0.4906662   9.134394
  16     12.45004  0.4751036   9.270980
  17     12.62121  0.4674697   9.383462
  18     12.77392  0.4620536   9.501380
  19     13.03887  0.4502849   9.690201
  20     13.32019  0.4375460   9.883096

Rsquared was used to select the optimal model using  the largest value.
The final value used for the model was ncomp = 10.The R-squared value
is 0.497881
```


Observed
Predicted
4
2
0
-2
0
1
2
3


resid(pls_model.1)
Index
2
0
-2
-4
0
200
400
600
800
1000
1200

The two plots for the model show that the prediction is pretty close to the actual value. The residual plot shows that the residuals are all clustered in the positive end and thus the model maybe overfitting

d) Predict the response for the test set. What is the test set estimate of R^2?



This is a plot for the observed vs predicted value for the test set, the model doesn't do very well on predicting the test set.

```
The estimated R square is calculated by squaring the correlation
between the predicted and observed values.
```

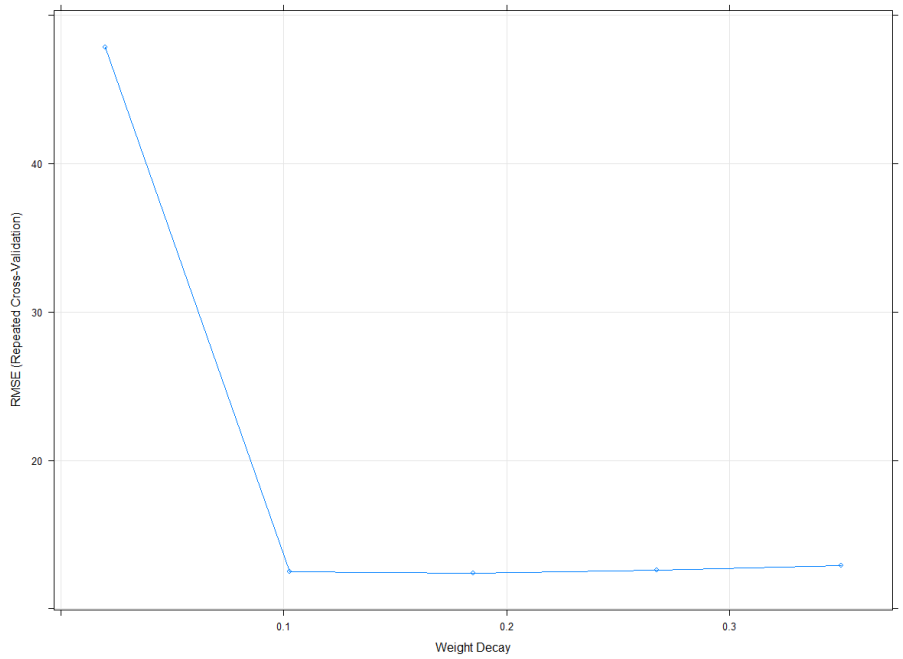
```
cor(pls_preds,Y_test)^2
```

[1] 0.3699912

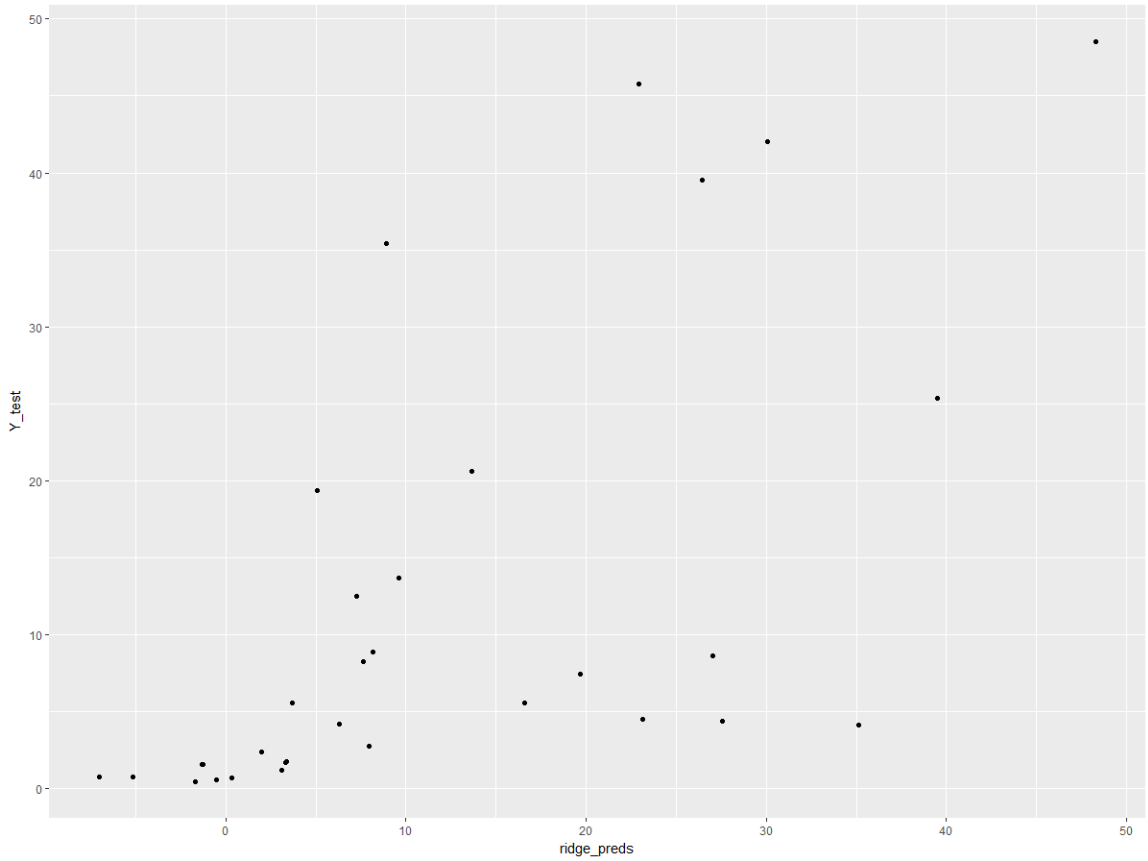
(e) Try building other models discussed in this chapter. Do any have better predictive performance?

Ridge_Model

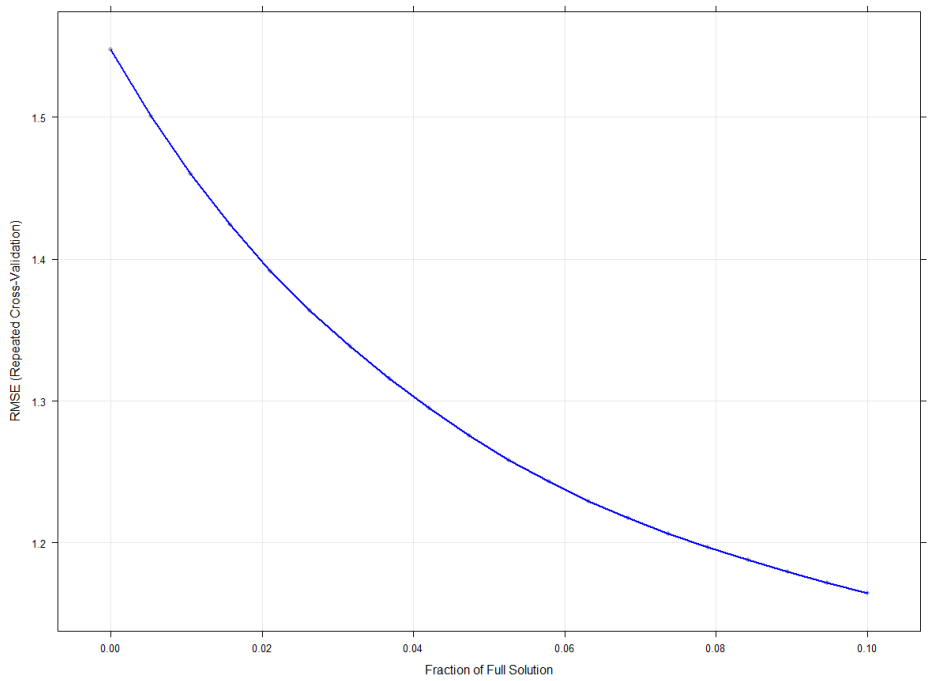
For the Ridge model the pre-processing used was center and scale.



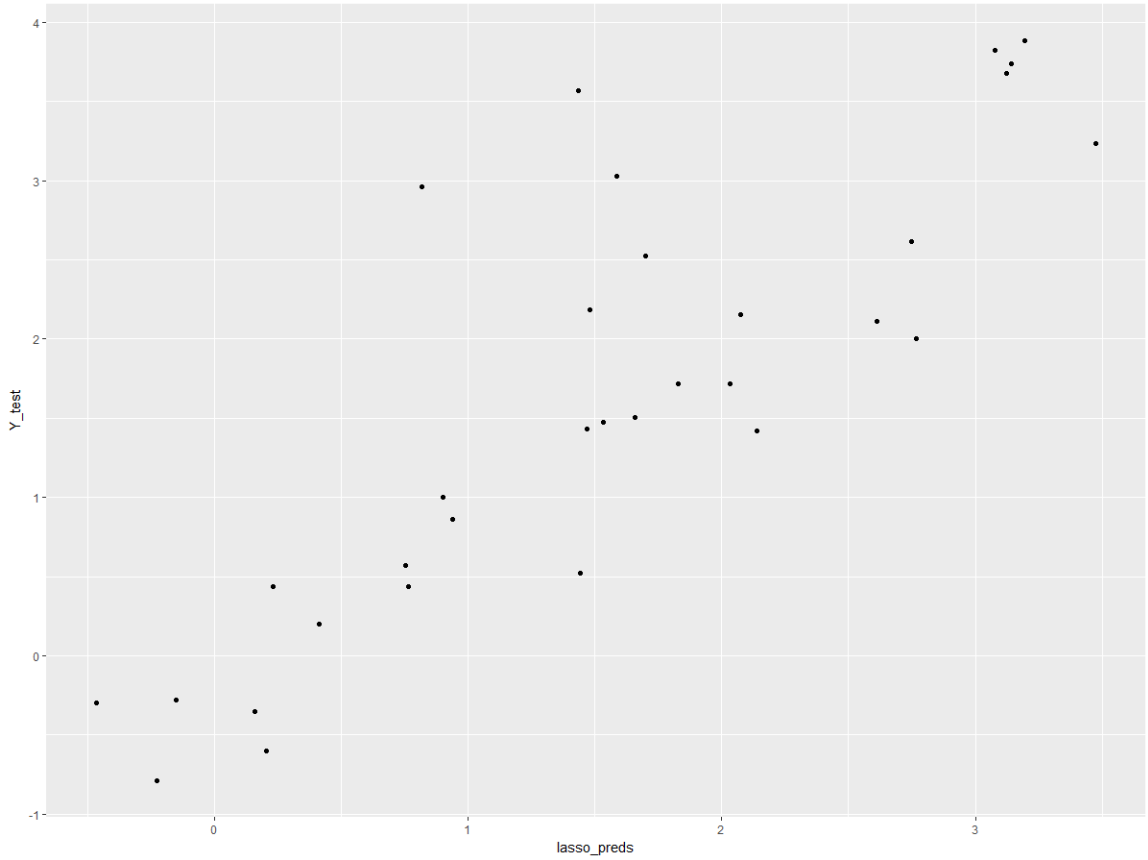

```
Ridge Regression

133 samples
388 predictors

Pre-processing: centered (388), scaled (388)
Resampling: Cross-Validated (5 fold, repeated 5 times)
Summary of sample sizes: 107, 105, 108, 106, 106, 105, ...
Resampling results across tuning parameters:

  lambda  RMSE      Rsquared   MAE
  0.0200  47.81407  0.3566056  30.643369
  0.1025  12.52186  0.4846626   9.218888
  0.1850  12.40759  0.4997030   9.211737
  0.2675  12.60715  0.5044045   9.422211
  0.3500  12.92568  0.5053998   9.708446

RMSE was used to select the optimal model using  the smallest value.
The final value used for the model was lambda = 0.185.

The following graph shows the relationship between the predicted and ob
served values for the test set, we can see that there is no definite re
lationship between the two.

The rsquared value for the test set is :
cor(ridge_preds,Y_test)^2
[1] 0.4042836
```



Lasso Model:
RMSE was used to select the optimal model using the smallest value. The final value used for the model was fraction = 0.1.The RMSE value was 1.164653.



The above graph describes the RMSE values of the lasso model.



The R-squared value of the lasso model on test set is 0.7186724

```
cor(lasso_preds, Y_test)^2
```

[1] 0.7186724

So the R squared value of Lasso is much better than the other two model
s.
RMSE_lasso <- sqrt(sum((Y_test - lasso_preds)^2)/32);

RMSE value for the Lasso on the test set is 0.7 which is an improvement over the training set.

Enet model:
Elasticnet

133 samples
388 predictors

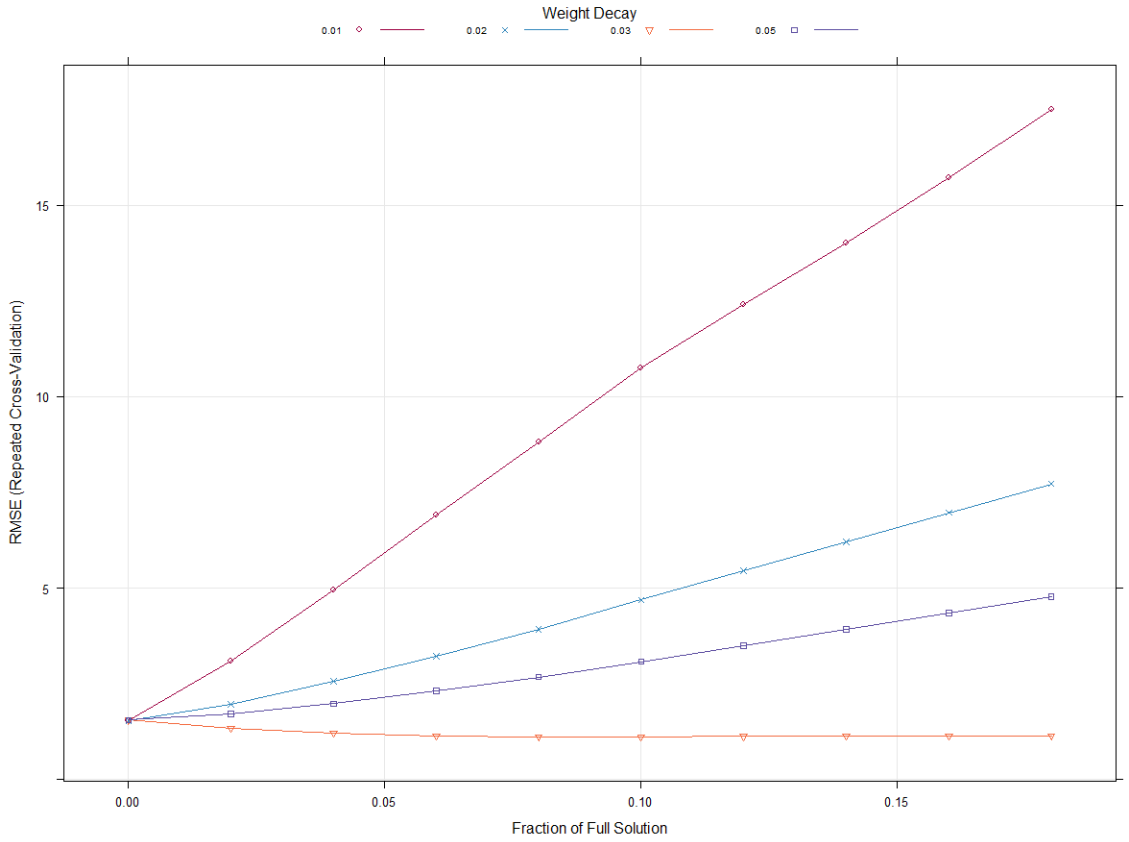
No pre-processing
Resampling: Cross-Validated (5 fold, repeated 5 times)
Summary of sample sizes: 107, 105, 108, 106, 106, 105, ...
Resampling results across tuning parameters:

lambda	fraction	RMSE	Rsquared	MAE
0.01	0.00001	1.542410	0.3530632	1.3050897
0.01	0.02001	3.101000	0.4164072	2.2002854
0.01	0.04001	4.962470	0.4235094	3.3571484
0.01	0.06001	6.903734	0.4239248	4.5760964
0.01	0.08001	8.825316	0.4127790	5.7604534
0.01	0.10001	10.744545	0.4077516	6.9379692
0.01	0.12001	12.405371	0.4053407	8.1023333
0.01	0.14001	14.027044	0.4045607	9.2909423
0.01	0.16001	15.719607	0.4034463	10.5665744
0.01	0.18001	17.517129	0.4014706	11.9162669
0.02	0.00001	1.546745	0.3548353	1.3149559
0.02	0.02001	1.959705	0.4225840	1.5253401
0.02	0.04001	2.562475	0.4523982	1.9058218
0.02	0.06001	3.219543	0.4674654	2.3391058
0.02	0.08001	3.930838	0.4642539	2.8259303
0.02	0.10001	4.692823	0.4526365	3.3632241
0.02	0.12001	5.457709	0.4466608	3.9001568
0.02	0.14001	6.215600	0.4448989	4.4305824
0.02	0.16001	6.970504	0.4448113	4.9567924
0.02	0.18001	7.725448	0.4453116	5.4831217
0.03	0.00001	1.558156	0.3505657	1.3258477
0.03	0.02001	1.340602	0.4421100	1.1041423
0.03	0.04001	1.213948	0.4768397	0.9747237
0.03	0.06001	1.143585	0.4976460	0.9016048
0.03	0.08001	1.110699	0.5050029	0.8666260
0.03	0.10001	1.115091	0.4968946	0.8735941
0.03	0.12001	1.126841	0.4884948	0.8839305
0.03	0.14001	1.135488	0.4830397	0.8906044
0.03	0.16001	1.139760	0.4820473	0.8924005
0.03	0.18001	1.143503	0.4823311	0.8931365
0.05	0.00001	1.554605	0.3505657	1.3212450
0.05	0.02001	1.708934	0.4303993	1.3697399
0.05	0.04001	1.982484	0.4539437	1.5056770
0.05	0.06001	2.311668	0.4740497	1.6905185
0.05	0.08001	2.678763	0.4819323	1.9204758

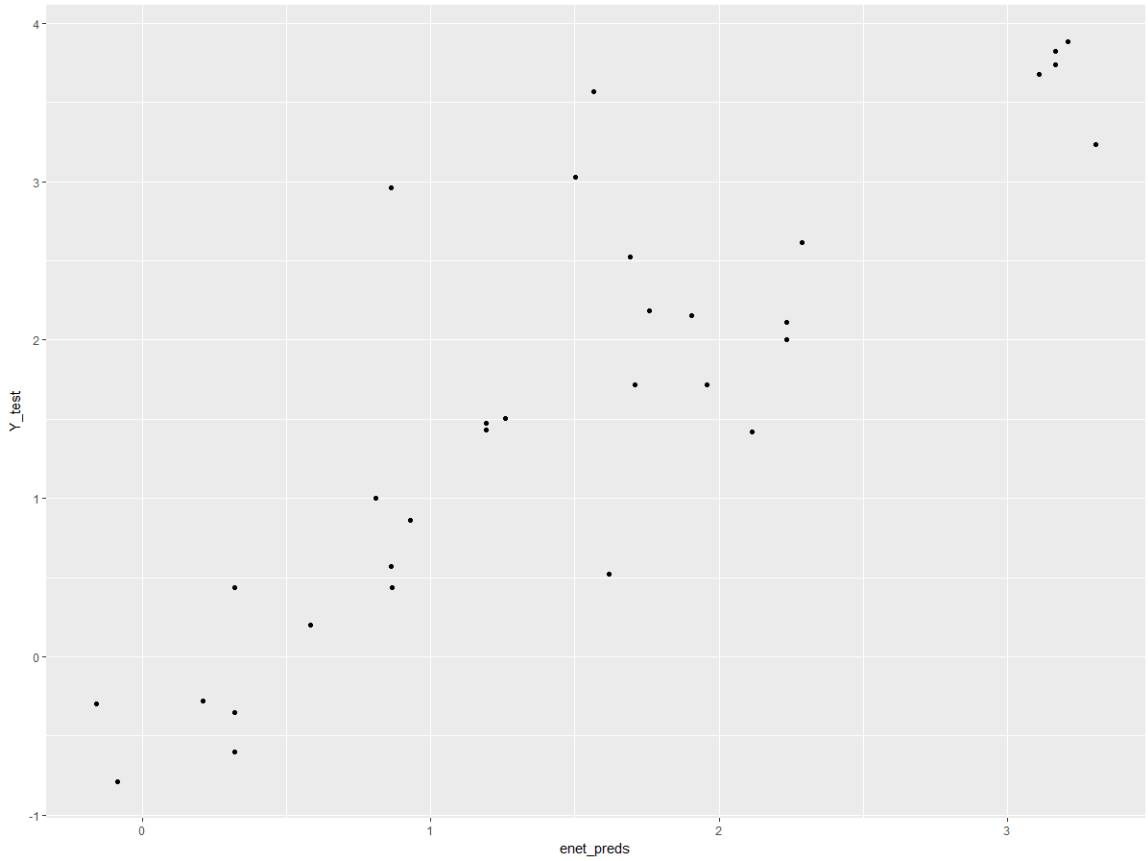
0.05	0.10001	3.080209	0.4790967	2.1893589
0.05	0.12001	3.502949	0.4689049	2.4799058
0.05	0.14001	3.927102	0.4612689	2.7688356
0.05	0.16001	4.349513	0.4564818	3.0575382
0.05	0.18001	4.771012	0.4557356	3.3388552

RMSE was used to select the optimal model using the smallest value. The final values used for the model were fraction = 0.08001 and lambda = 0.03.

RMSE was used to select the optimal model using the smallest value. The final values used for the model were fraction = 0.08001 and lambda = 0.03.The RMSE was 1.110699



The following graph gives the observed vs predicted values of the variables.



Model summary:

For Test Set

ENET: R square: 0.7440922 RMSE: 0.7467911

Lasso Rsquare: 0.7186724 RMSE: 0.7502953

PLS: R_squared 0.3699912 RMSE 1.039602

Clearly from the comparisons using R squared the most justified model would be the Elastic net model The RMSE of this model is also the lowest among all the other models.

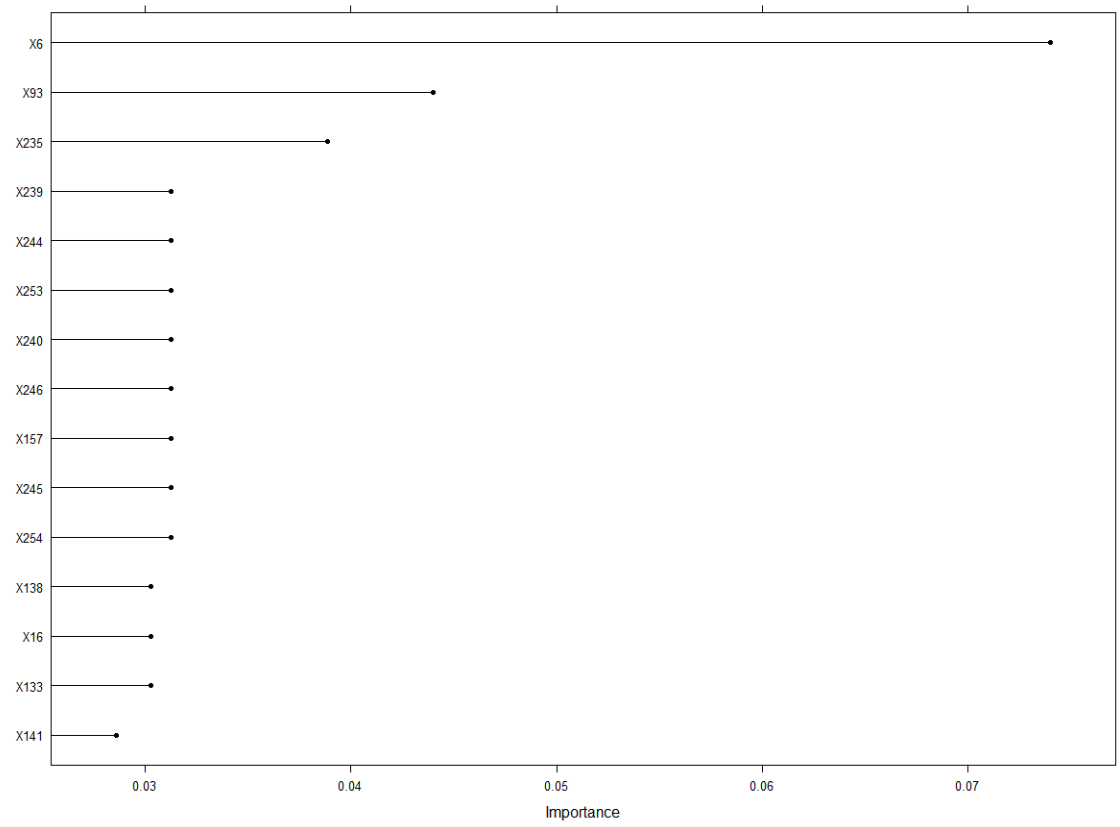
f) Would you recommend any of your models to replace the permeability Laboratory experiment?

None of the methods were very accurate to predict the outcome. Most methods have R squared value in the 0.7 to 0.8 range which means about 30 percent of the times the model will not predict properly additionally the Residuals of the models indicate overfitting. I would not recommend any of these models to replace the

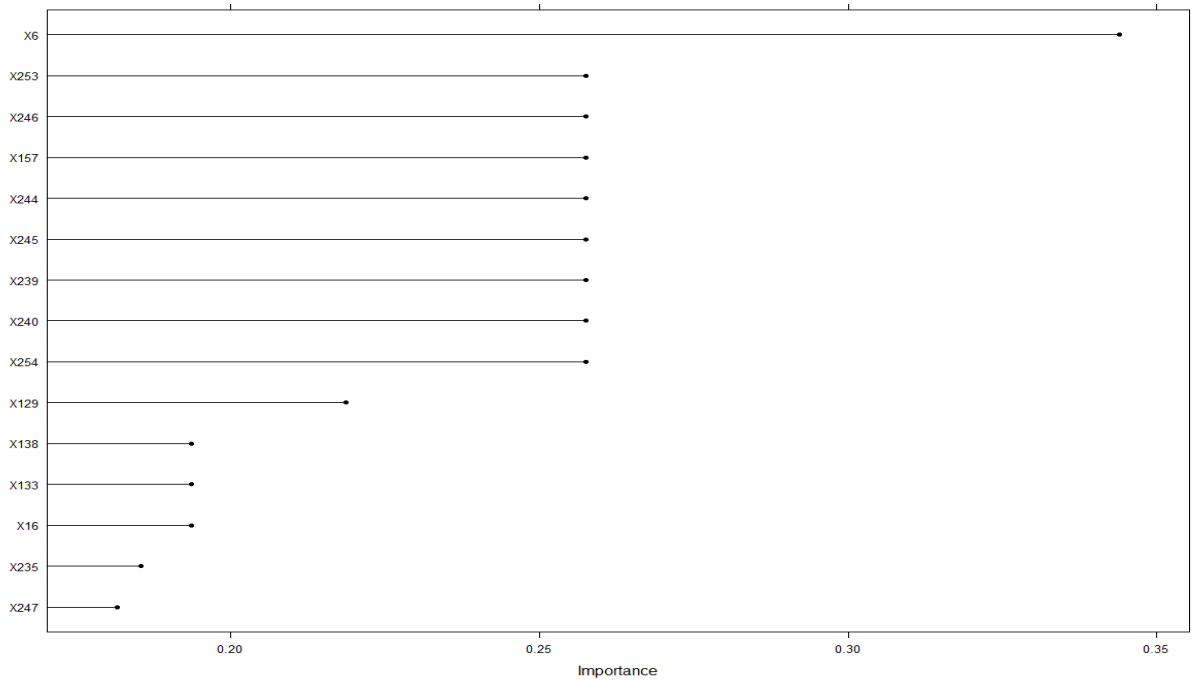
Laboratory methods.

6.2 (g) *Which predictors are most important in the model you have trained? Do any predictors dominate the list?*

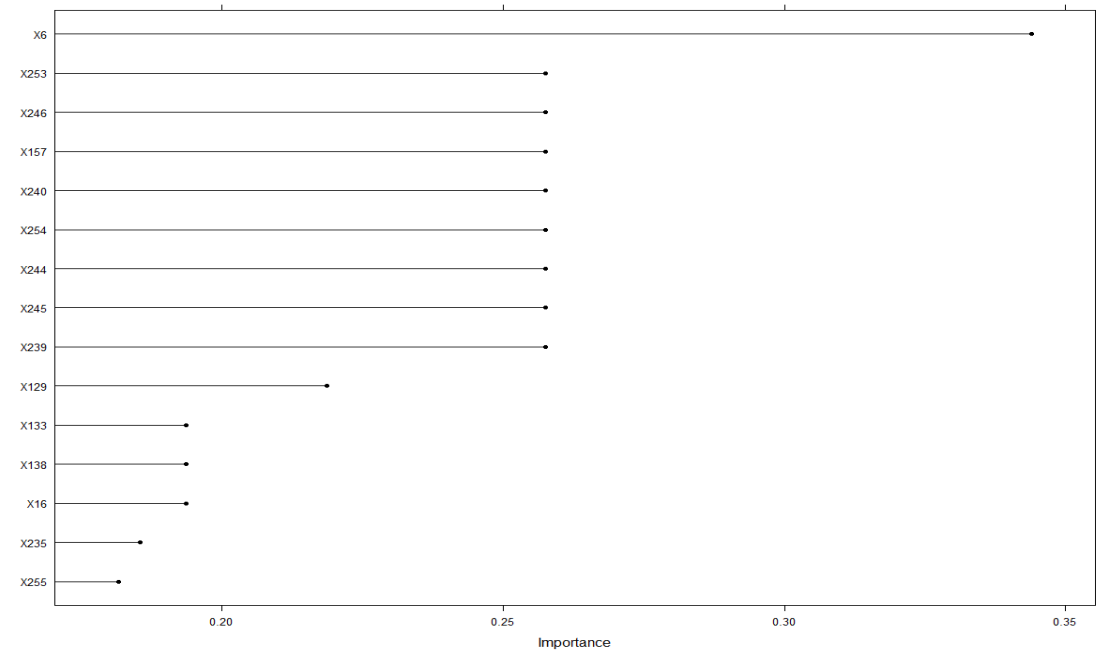
PLS model



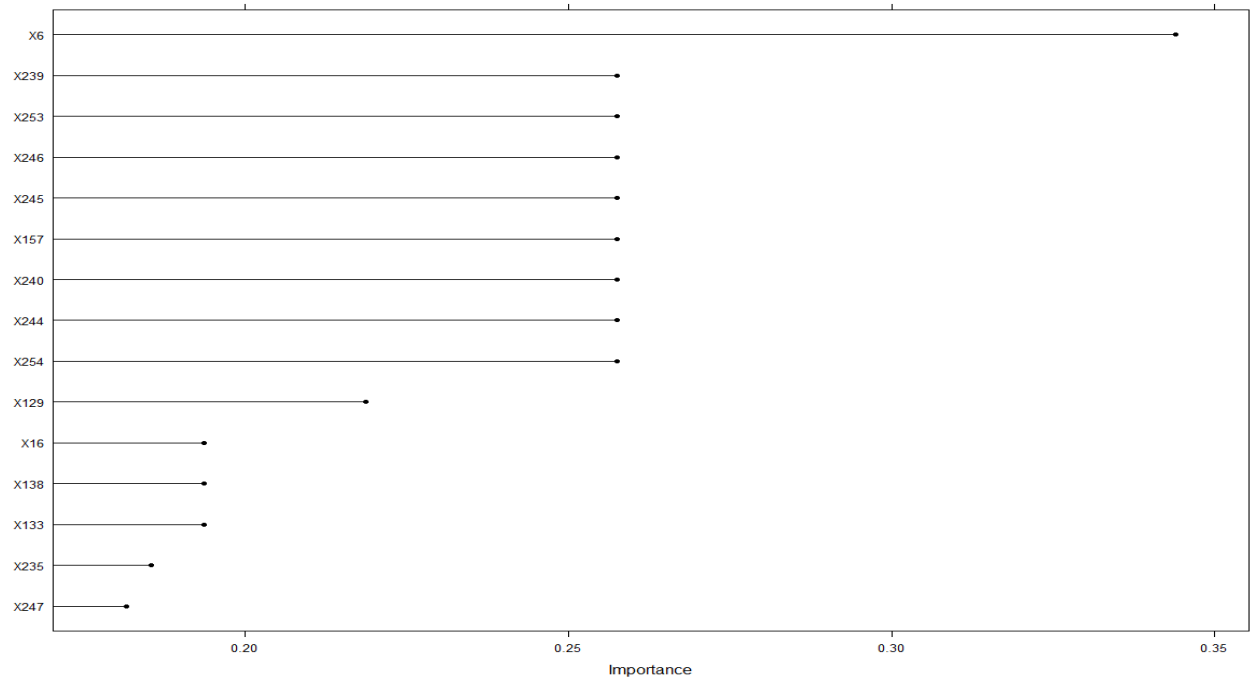
Ridge model



Enet model



Lasso model



PLS		Enet		lasso_model		Ridge	
X6	0.07402	X6	0.3440	X6	0.3440	X6	0.3440
X93	0.04398	X239	0.2576	X239	0.2576	X157	0.2576
X235	0.03885	X246	0.2576	X253	0.2576	X246	0.2576
X244	0.03126	X244	0.2576	X240	0.2576	X254	0.2576
X157	0.03126	X245	0.2576	X157	0.2576	X240	0.2576
X246	0.03126	X157	0.2576	X244	0.2576	X239	0.2576
X245	0.03126	X240	0.2576	X254	0.2576	X253	0.2576
X240	0.03126	X253	0.2576	X246	0.2576	X245	0.2576
X253	0.03126	X254	0.2576	X245	0.2576	X244	0.2576
X254	0.03126	X129	0.2186	X129	0.2186	X129	0.2186
X239	0.03126	X16	0.1936	X133	0.1936	X133	0.1936
X138	0.03026	X133	0.1936	X138	0.1936	X138	0.1936
X133	0.03026	X138	0.1936	X16	0.1936	X16	0.1936
X16	0.03026	X235	0.1854	X235	0.1854	X235	0.1854
X141	0.02859	X255	0.1815	X247	0.1815	X247	0.1815
X247	0.02796	X247	0.1815	X255	0.1815	X255	0.1815
X255	0.02796	X260	0.1764	X265	0.1764	X265	0.1764
X266	0.02786	X265	0.1764	X260	0.1764	X260	0.1764
X262	0.02786	X125	0.1764	X130	0.1764	X125	0.1764
X129	0.02755	X130	0.1764	X125	0.1764	X130	0.1764

X6 is one of the most important predictors in all the models so X6 surely dominates the list. Apart from that X239, 246, X244, X138etc.

```
#(a) Start R and use these commands to load the data:
library(AppliedPredictiveModeling)
data(permeability)
#The fingerprint predictors indicate the presence or absence of substructures
#of a molecule and are often sparse meaning that relatively few of the
#molecules contain each substructure. Filter out the predictors that have
#low frequencies using the nearZeroVar function from the caret package.
#How many predictors are left for modeling

library(caret)

ZC <- nearZeroVar(fingerprints, saveMetrics = T,freqCut = 95/5);
NZV<-ZC[ZC[, "nzv"] == TRUE, ]
ZV<-ZC[ZC[, "zeroVar"] == TRUE, ]
drop_cols<-row.names(NZV)
ZV<-row.names(ZV)
a<-fingerprints[ , !(colnames(fingerprints) %in% drop_cols)]
a<-a[ , !(colnames(a) %in% ZV)]



# positively skewed
ggplot() + geom_histogram(aes(x = permeability), binwidth = 4, col = 1) +
  labs(title = "Histogram of permeability", x = "Molecule permeability") +
  theme_bw()
```

```
Trans <- BoxCoxTrans(permeability)

trans <- preProcess(permeability, method = c("BoxCox"))

transformed <- predict(trans, permeability)

ggplot() + geom_histogram(aes(x = transformed), binwidth = 1, col = 12)


set.seed(1)

Partition <- createDataPartition(transformed, times = 1, p = 0.80,list=FALSE)



set.seed(1)

training <- a[Partition, ]

test <- a[-Partition, ]

Y_train <- transformed[Partition]

Y_test <- transformed[-Partition]


ctrl = trainControl("repeatedcv", number = 5, repeats = 5)

set.seed(1)

pls_model.1 <- train(y = Y_train,x = training,method = "pls",preprocess=c("center","scale"),metric =
"Rsquared",trControl = ctrl)

pls_model.1

plot(pls_model.1)

pls_preds <- predict(pls_model.1, test)

cor(pls_preds,Y_test)

e<-ggplot() + geom_point(aes(x = pls_preds, y = Y_test))

e

pls_model.1


set.seed(1)
```

```
ridgeGrid <- data.frame(lambda = seq(0.02, .35, length = 5))
ridgeTune <- train(x = training, y = Y_train,
                   method = "ridge",
                   tuneGrid = ridgeGrid,
                   trControl = ctrl,preProc = c("center", "scale"))
ridge_preds <- predict(ridgeTune, test)
ridgeTune
b<-ggplot() + geom_point(aes(x = ridge_preds, y = Y_test))
b
cor(ridge_preds,Y_test)^2

# Lasso
set.seed(1)
lasso_grid <-expand.grid(.fraction=seq(0.1,0.5,length=20))
lasso_model <- train(y = Y_train,
                     x = training,
                     method = "lasso",
                     tuneGrid = lasso_grid,
                     metric = "RMSE",
                     trControl = ctrl,
                     preProcess=c("center", "scale","pca"),importance=TRUE)
lasso_model
lasso_preds <- predict(lasso_model, test)
c<-ggplot() + geom_point(aes(x = lasso_preds, y = Y_test))

RMSE_lasso <- sqrt(sum((Y_test - lasso_preds)^2)/n); RMSE_lasso
cor(lasso_preds, Y_test)

enet_preds <- predict(enet_model, test)
```

```
d<-ggplot() + geom_point(aes(x = enet_preds, y = Y_test))


RMSE_enet <- sqrt(sum((Y_test - enet_preds)^2)/n); RMSE_enet

cor(enet_preds, Y_test)


set.seed(1)

enet_grid <- expand.grid(.fraction = seq(0.001, 0.1, 0.01), .lambda = c(0, 0.0001, 0.001, 0.01))

Enet <- train(y = Y_train,

        x = training, , method = "enet", trControl = ctrl,

        tuneGrid = enet_grid)

Enet

plot(Enet)




pca <- prcomp(training,center = TRUE,scale = TRUE,importance=TRUE)

s <- summary(pca)

x.var <- (s$sdev ^ 2)

p.x.pvar<-x.var*100


plot(pca,ylim=c(10,100))

plot(p.x.pvar,xlab="Principal component", ylab="% variance", ylim=c(0,100), type='b',main="variance in
percentage")

plot(1:133, s$importance[3,], type = "b",main="variances explained",xlab = "principle
components",ylab="variances")

print("proportions of variance:")

print(p.x.pvar)


par(mfrow=c(2,2))
```

```
plsImp1 <- varImp(pls_model.1, scale = FALSE)
plot(plsImp1, top=15, scales = list(y = list(cex = 0.8)))
plsImp2 <- varImp(enet_model, scale = FALSE)
plot(plsImp2, top=15, scales = list(y = list(cex = 0.8)))
plsImp3 <- varImp(ridge_model, scale = FALSE)
plot(plsImp3, top=15, scales = list(y = list(cex = 0.8)))
plsImp4 <- varImp(lasso_model, scale = FALSE)
plot(plsImp4, top=15, scales = list(y = list(cex = 0.8)))



#Exploring data and correlations
library(caret)
data(tecator)
Matrix<-cor(absorp)
library(corrplot)
corrplot(Matrix)
Matrix[1:5, 1:5]

#predictor distribution

ggplot(data = data.frame(absorp)) +
  geom_histogram(aes(x = X1), bins = 20, col = 1) +
  labs(title = "Histogram of IR wavelength no. 1",
       x = "Wavelength predictor 1")

#outlier analysis
boxplot(endpoints,main="box plot for water,fat and protien")
```

```
#b)In this example the predictors are the measurements at the individual frequencies. Because the frequencies

#lie in a systematic order (850-1,050nm), the predictors have a high degree of

#correlation. Hence, the data lie in a smaller dimension than the total number of predictors (215).

#Use PCA to determine the e???ective dimension of these data. What is the e???ective dimension?

#PCA components:

pca <- prcomp(absorp,center = TRUE,scale = TRUE)

s <- summary(pca)

x.var <- (s$sdev ^ 2)

p.x.pvar<-x.var*100

par(mfrow=c(1,2))

plot(pca,ylim=c(10,100))

plot(p.x.pvar,xlab="Principal component", ylab="% variance", ylim=c(0,100), type='b',main="variance in percentage")

plot(1:100, s$importance[3,], type = "b",main="variances explained",xlab = "principle components",ylab="variances")

print("proportions of variance:")

print(p.x.pvar)


#c)Split the data into a training and a test set, pre-process the data, and build each variety of

#models described in this chapter.

#For those models with tuning parameters, what are the optimal values of the tuning parameter(s)?

library("caret")

set.seed(1)

data <- data.frame(cbind(absorp,endpoints[,2]))

colnames(data) <- c(names(data[,1:100]), "Y")

iTrain <- createDataPartition(y = data$Y, p = 0.75,list = FALSE)

Train <- data[iTrain, ]

Test <- data[-iTrain, ]
```

```
set.seed(1)

ctrl <-trainControl(method = "LGOCV", number = 5,p=.75)


# Linear regression


mc <- findCorrelation(training, cutoff = 0.90)

Linear_train <- data.frame(training[ ,-mc])

# colnames(training_linear) <- "X1"

set.seed(1)

LM <- train(y =Train[,101] , x = Linear_train, method = "lm",trControl = ctrl)

LM

LM$results

xyplot(Train[,101] ~ predict(LM), type = c("p", "g"), xlab = "Predicted", ylab = "Observed")

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


#PLS AND PCR

set.seed(1)

PLS <- train(y =Train[,101] , x = Train[,-101], method = "pls",preProcess=c("center","scale"),trControl =
ctrl,tuneLength=25)

PLS

summary(PLS)

xyplot(Train[,101] ~ predict(PLS), xlab = "Predicted", ylab = "Observed",line=TRUE,asp=1)

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




set.seed(1)

PCR <- train(y =Train[,101] , x = Train[,-101], method = "pcr",trControl = ctrl,tuneLength=25)

PCR
```

```
summary(PCR)

xyplot(Train[,101] ~ predict(PCR), xlab = "Predicted", ylab = "Observed",line=TRUE,asp=1)

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



#comparision

dev.off()

plot(PLS$results[,2],type='b',col="blue",ylab="RMSE",xlab="Number of
Components",xlim=c(1,25),asp=TRUE,ylim=c(1,12),main="comparision between PCR and PLS models")

points(PCR$results, col=2,type='b',pch=17)

legend("topright", inset=.05, title="Models",c("PLS","PCR"), fill=c("blue","red"), horiz=TRUE)



set.seed(1)

# Ridge Regression - penalise square of coefficient

ridgeGrid <- data.frame(.lambda = c(0, 0.0001, 0.001, 0.01,0.1))

RM<- train(y =Train[,101],x = Train[,-101],method = "ridge",trControl = ctrl,tuneGrid =
ridgeGrid,preProcess=c("center","scale"))

RM

plot(RM)



#Lasso

set.seed(1)

lasso_grid <- lasso_grid <- expand.grid(.fraction=seq(0,0.1, length=20))

Lasso <- train(y = Train[,101],x = Train[,-101], method = "lasso", trControl = ctrl,

                tuneGrid = lasso_grid)

Lasso

plot(Lasso)
```

```
#Elastinet
set.seed(1)
enet_grid <- expand.grid(.fraction = seq(0.001, 0.1, 0.01), .lambda = c(0, 0.0001, 0.001, 0.01))
Enet <- train(y = Train[,101],x = Train[,-101], method = "enet", trControl = ctrl,
          tuneGrid = enet_grid)
Enet
plot(Enet)

# prediction on Test set
par(mfrow=c(3,2))
# Linear regression
test_linear <- data.frame(Test[ ,-mc])
colnames(test_linear) <- colnames(Linear_train)
linear_pred <- predict(LM, test_linear)
a<-ggplot() + geom_point(aes(x = linear_pred, y = Test[,101]))

n <- length(Test[,101])
RMSE_lm <- sqrt(sum((Test[,101] - linear_pred)^2)/n); RMSE_lm
cor(linear_pred, Test[,101])

# Ridge regression
ridge_preds <- predict(RM, test)
b<-ggplot() + geom_point(aes(x = ridge_preds, y = Test[,101]))

RMSE_ridge <- sqrt(sum((Test[,101] - ridge_preds)^2)/n); RMSE_ridge
cor(ridge_preds, Test[,101])
```

```
# Lasso
lasso_preds <- predict(Lasso, test)
c<-ggplot() + geom_point(aes(x = lasso_preds, y = Test[,101]))

RMSE_lasso <- sqrt(sum((Test[,101] - lasso_preds)^2)/n); RMSE_lasso
cor(lasso_preds, Test[,101])

# Elastic net
enet_preds <- predict(Enet, test)
d<-ggplot() + geom_point(aes(x = enet_preds, y = Test[,101]))

RMSE_enet <- sqrt(sum((Test[,101] - enet_preds)^2)/n); RMSE_enet
cor(enet_preds, Test[,101])

# PLS
pls_preds <- predict(PLS, Test[,-101])
e<-ggplot() + geom_point(aes(x = pls_preds, y = Test[,101]))

RMSE_pls <- sqrt(sum((Test[,101] - pls_preds)^2)/n); RMSE_pls
list<-c(cor(linear_pred, Test[,101]),
cor(pls_preds, Test[,101]),
cor(ridge_preds, Test[,101]),
cor(lasso_preds, Test[,101]),
cor(enet_preds, Test[,101]))
list<-c(cor(linear_pred, Test[,101]),
         cor(pls_preds, Test[,101]),
         cor(ridge_preds, Test[,101]),
        cor(lasso_preds, Test[,101]),
 cor(enet_preds, Test[,101]))
```

```
names(list)<-c("Linear","PLS","Ridge","Lasso","Enet")

Correlation<-as.table(list)

Correlation
```