

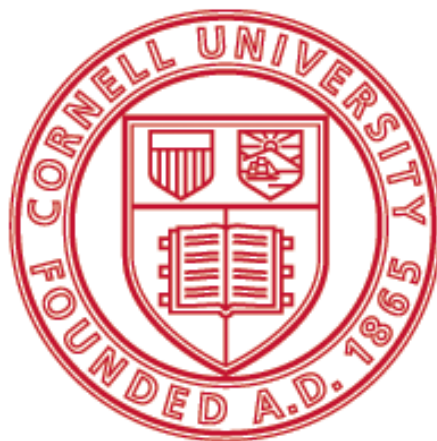
# STSCI 4740

# Final Report

By

Rui Chen (rc687)  
Mengdi Fan (mf696)  
Qianyan Yao (qy62)

Instructor: Professor Yang Ning



Cornell University

## **Abstract**

We use all data from the original dataset. We use Subset Selection (Best Subset Selection, Forward Stepwise Selection, and Backward Stepwise Selection), Shrinkage Methods (Ridge Regression and Lasso), Dimension Reduction Methods (Principal Components Regression and Partial Least Squares), and Tree-Based Methods (Bagging and Random Forest). Our best model is a linear model with 5 predictors, Akt2, Rik, Sphk2, Pik3r1 and Rac1, which we obtained by using the Forward Stepwise Selection.

## **1. Overview of the Dataset**

### **1.1 Dataset Description and General Check**

The dataset Kidney includes 40 observations (expression) and 24 variables (gene names). We did the check as follows.

#### **1.1.1 Check Extreme Values**

For each variable, we display the boxplot to check whether there is any extreme value. We find there is no extreme value.

#### **1.1.2 Check Correlation**

We check the correlation between every possible pairs of variables by looking at the correlation matrix and the correlation plot. We find the correlation between any two variables is not strong. The correlation is actually between -0.5657 and 0.6668.

#### **1.1.3 Check N/A**

We check that there is no N/A in the dataset.

#### **1.1.4 Check Normality**

We use QQ plots to check whether variables are normal. The plots show that all variables are normal, so no transformation is needed.

### **1.2 Data Splitting**

We partition the data into two groups: training set and test set, using the function `sample(1:40, 30)`. Training set includes 30 observations (`n_training = 30`), and test set includes 10 observations (`n_test = 10`). While fitting the model, we want to make sure `n_training > p`, which is the reason why we split the data this way instead of splitting in halves. We use `set.seed(1)` to set the seed.

To give a more objective result and avoid randomness, we tests all seeds from 1 to 50 to perform Ridge and Lasso. We obtain test results for all 50 seeds and display them in a boxplot.

## 2. Model Exploration

### 2.1 Subset Selection Methods

In this section we consider best subset and stepwise model selection procedures (forward and backward).

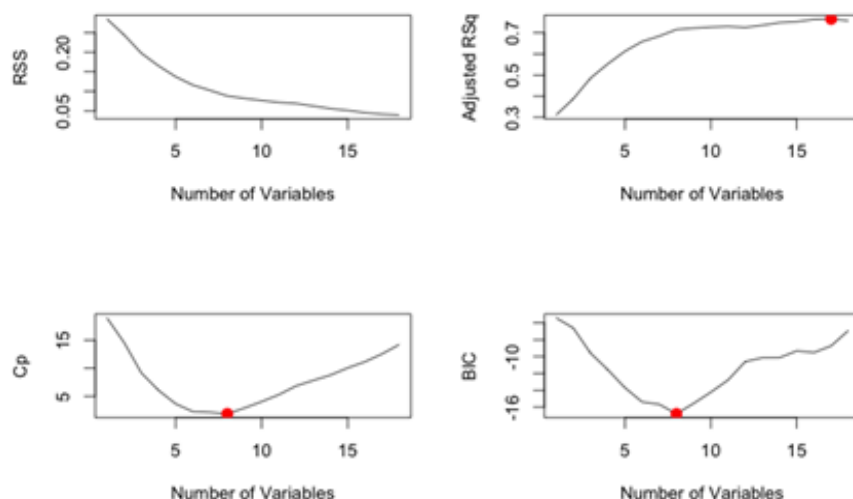
For all the methods, we use training data to fit the model. We use the function `predict()` to get the prediction and then calculate the MSE.

For this project, we did not use the full data to do fit the model but only use the training data. This means we choose not to use the build-in function `cv. Err`. An advantage of doing this is that we will be able to compare MSE values we obtain from subset selection methods to the other MSE values we get by using all the other methods (to fit the model using the training data consistently).

#### 2.1.1 Best Subset Selection

The best model using adjusted R-squared is with 17 variables, including Cdc42, Pla2g6, Akt2, Plcg2, Rac2, Rik, Pik3cd, Map2k1, Pik3r3, Nras, Nos3, Pik3r1, Ppp3cb, Nfatc4, Mapk13, Rac1 and Nfat5.

The best model using Cp and BIC as criteria is with 8 variables, including Akt2, Plcg2, Rac2, Sphk2, Nos3, Ppp3cb, Mapk13, Rac1 and Mapk1.



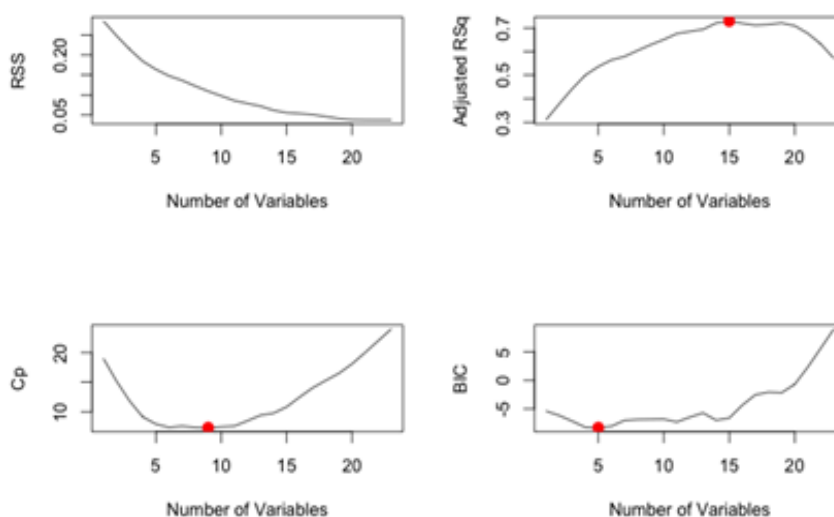
**Figure 1** Best subset selection

#### 2.1.2 Forward Stepwise Selection

The best model using adjusted R-squared is with 15 variables, including Cdc42, Pla2g6, Akt2, Plcg2, Rac2, Rik, Pik3cd, Sphk2, Nras, Nos3, Pik3r1, Ppp3cb, Map2k2, Mapk13 and Rac1.

The best model using Cp as criterion is with 9 variables, including Akt2, Plcg2, Rac2, Rik, Pik3cd, Sphk2, Nos3, Pik3r1 and Rac1.

The best model using BIC as criterion is with 5 variables, including Akt2, Rik, Sphk2, Pik3r1 and Rac1.

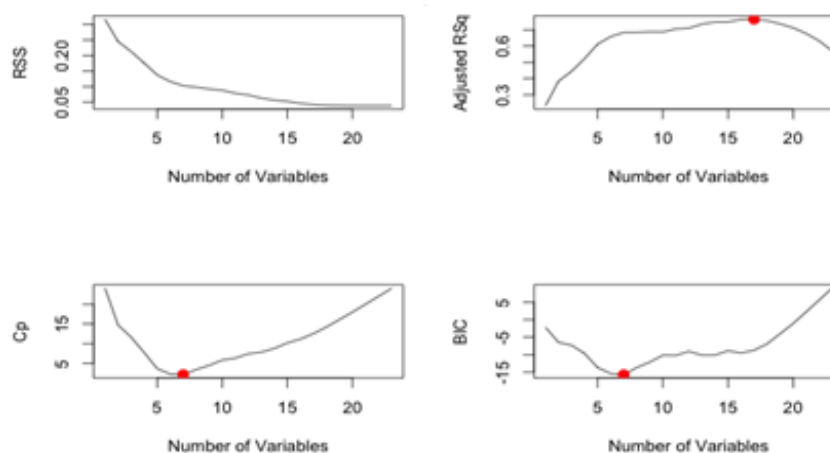


**Figure 2** Forward Stepwise Selection

### 2.1.3 Backward Stepwise Selection

The best model using adjusted R-squared is with 17 variables, including Cdc42, Pla2g6, Akt2, Plcg2, Rac2, Rik, Pik3cd, Map2k1, Pik3r3, Nras, Nos3, Pik3r1, Ppp3cb, Nfatc4, Mapk13, Rac1, and Nfat5.

The best model using Cp and BIC as criterion is with 7 variables, including Akt2, Plcg2, Rac2, Nos3, Ppp3cb, Mapk13 and Rac1.



**Figure 3** Backward Stepwise Selection

## 2.2 Shrinkage Methods

We want to try if we can fit a model involving all  $p$  predictors so we try shrinkage methods, which shrink the coefficients towards zero. Shrinking the coefficient estimates can significantly reduce their variance. In this section, we try to use Ridge and Lasso methods. We use 5-fold cross validation to choose a best lambda which produces the lowest MSE on the training data. Then we build models of Ridge and Lasso using the given lambdas. Next, we compute the test MSE on the

testing dataset. Besides running only seed(1), we also write a for loop to run seed(1) to seed(50) to generate different training dataset and test dataset.

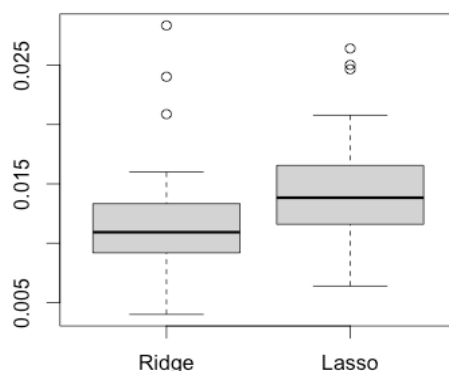
### 2.2.1 Ridge

For seed (1), we get a best lambda 0.344 using k-fold cross validation. We fit the model using this lambda and get the test MSE=0.01113, which is relatively small.

### 2.2.2 Lasso

For seed (1), we get a best lambda 0.433 using k-fold cross validation. We fit the model using this lambda and get the test MSE = 0.01576, which is relatively small. From the R output we can see after Lasso regression, lots of coefficients are vanished. There are only three coefficients (except for the intercept coefficient): Rik 0.0756, Pik3r3 0.0913, and Ppp3cb 0.0434. Thus, Lasso performs variable selection and yields a sparse model.

We want to test which shrinkage method is better in this case. We run a for loop from set.seed(1) to set.seed(50) to generate different training datasets and test datasets. The test MSE under different dataset are shown in the boxplot, as below. Generally, Ridge regression yields smaller test MSE than Lasso regression. So for this dataset Ridge regression is slightly better than lasso, even though both MSE values are relatively small. Thus, among the two shrinkage methods, we prefer Ridge rather than Lasso.



**Figure 4** MSE Boxplot vs. Ridge & Lasso

## 2.3 Dimension Reduction Methods

In this dataset, there exists high dimensional problems since the number of features ( $p=23$ ) is close to the sample size ( $n=40$ ). Thus, we also derive low-dimensional sets of features to explain the variability and make prediction, using dimension reduction methods .

### 2.3.1 Principal Component Regression (PCR)

In PCR, each of the dimensions found is a linear combination of the  $p$  features. When we run cross validation, it shows that the model with 4 principle components has the smallest cross validation

error. We refit the model with  $M=4$  using training dataset and make prediction using test dataset. It turns out that MSE (0.02654) is relatively high compared to other methods such as Ridge and Lasso. Also, only 55% of the variance in response variable can be explained by variance in the first four principal components, which does not show advantages over the linear models. In addition, PCR does not perform feature selection, thus it is relatively hard to interpret compared to linear models.

### **2.3.2 Partial Least Squares (PLS)**

The supervised dimension reduction of PLS can reduce bias, and it attempts to find directions that help explain both the response and the predictors. Using cross validation, it shows one-component model performs best in terms of the lowest CV error. However, only 59% of the variance in response variable can be explained by variance in the first component, which is also not very high. This confirms that PLS often performs no better than Ridge or PCR in practice, with its relatively high test MSE (0.02843) using this dataset.

## **2.4 Tree-Based Methods**

Last, we test tree-based methods including bagging and random forest. They both involve producing multiple trees which are then combined to yield a single consensus prediction.

### **2.4.1 Bagging**

We use  $mtry=23$  to consider all 23 predictors for each split of the tree. The model shows a relatively low test MSE (0.01457). However, it is not possible to represent the result using a single tree. This is because it involves creating multiple training dataset using bootstrap, fitting a decision tree to each dataset, and then combining all the trees to create a single predictive model. That means it decreases the MSE at the expense of interpretability.

### **2.4.2 Random Forest**















In addition, we use  $mtry=8$  in random forest to consider 8 predictors for each split of the tree. All the other procedures are very similar to bagging. As the results shown, random forest only explains 14.23% of the variation and the test MSE is not small enough, either. It is a relatively complex model and is hard to interpret as well.

## **3. Model selection and Interpretation**

### **3.1 Comparison of Models**

We fit multiple linear and nonlinear models, including linear regression, Ridge, Lasso, PCA, PLS, bagging, and random forest, and use the test MSE as the main criterion for model selection. In terms of interpretability, we also take the number of parameters into consideration. The results are listed in the table below.

**Table 1** Model Comparison

Model	Test MSE		# Parameters or Components
Linear, full	0.09013		23
Linear, best subset selection (Adj R2)	0.08723		17
Linear, best subset selection (Cp, AIC, BIC)	0.01827		8
Linear, forward subset selection (Adj R2)	0.04448		15
Linear, forward subset selection (Cp, AIC)	0.01306		9
Linear, forward subset selection (BIC)	0.00799		5
Linear, backward subset selection (Adj R2)	0.08723		17
Linear, backward subset selection (Cp, AIC, BIC)	0.02200		7
Ridge regression	0.01113		23
Lasso regression	0.01576		3
PCR	0.02654		4
PLS	0.02843		1
Bagging	0.01457		-
Random Forest	0.01283		-

### 3.2 Final Model Recommendation

We conclude that the linear model with five parameters would be the best model, which is obtained by using the Forward Stepwise Selection. It is selected by BIC and has the smallest test MSE (0.00799). In addition, this model is parsimonious with only five variables, which is easier to interpret compare to nonlinear models and most of the other linear models. Based on the R output, the model can be written in a form below.

$$\text{Mapk1} = -0.49644 - 0.41529 \cdot \text{Akt2} + 0.34023 \cdot \text{Rik} - 0.16635 \cdot \text{Sphk2} - 0.24678 \cdot \text{Pik3r1} + 0.39354 \cdot \text{Rac1} + \epsilon$$

```
Call:
lm(formula = Mapk1 ~ Akt2 + Rik + Sphk2 + Pik3r1 + Rac1, data = kidney[train,
])
```

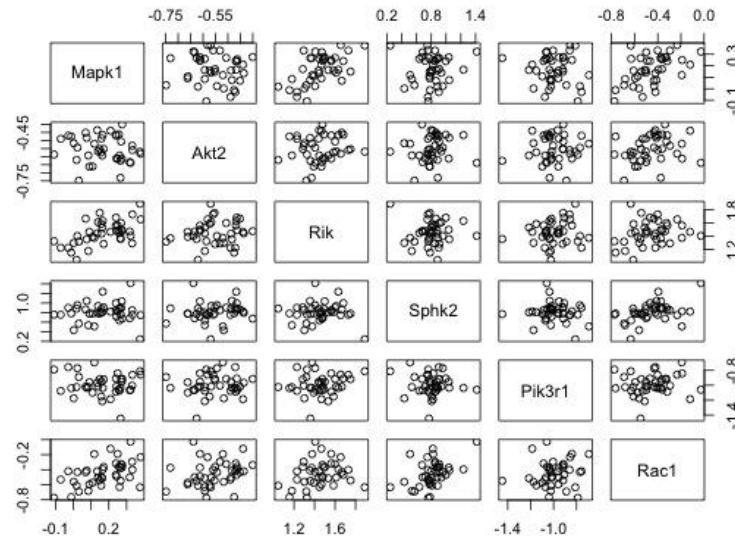
```
Residuals:
    Min       1Q   Median       3Q      Max
-0.127412 -0.063122 -0.000875  0.066505  0.175906
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.49644    0.23358  -2.125 0.044045 *
Akt2        -0.41529    0.24018  -1.729 0.096645 .
Rik          0.34023    0.08878   3.832 0.000805 ***
Sphk2       -0.16635    0.09674  -1.720 0.098380 .
Pik3r1      -0.24678    0.10215  -2.416 0.023668 *
Rac1         0.39354    0.11004   3.576 0.001525 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 0.08269 on 24 degrees of freedom
Multiple R-squared:  0.6161,    Adjusted R-squared:  0.5362
F-statistic: 7.705 on 5 and 24 DF,  p-value: 0.0001927
```

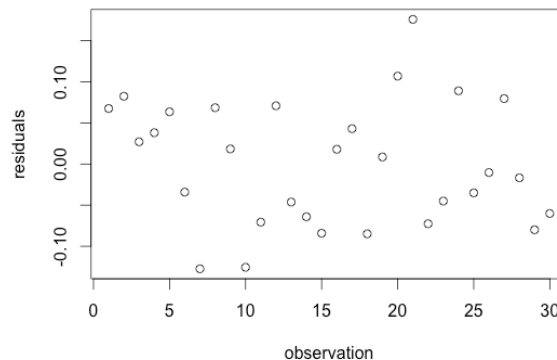
### 3.3 Model Diagnostics and Interpretation

For our final model, we check the correlation between each of the five predictors and Mapk1. As we can see from the pairwise plot below. There is no explicit correlation between each pairs.



**Figure 5** Pairwise Correlation Plot

We also check the homogeneity of variance to make sure the error term is not heteroscedastic. The residual is randomly distributed around 0, and there is no explicitly specific trend, indicating no violation detected.



**Figure 6** Residual Distribution

The multiple R-squared and adjusted R-squared are 0.6161 and 0.5362, which are not very high. That means about 50%-60% variation in Mapk1 can be interpreted by the five variables listed as above. When we fit the dataset into the full regression model, which contains all variables, the multiple R-squared is 0.7792 and the adjusted R-squared is 0.4619, which are not high, either. So, we believe that the adjusted R-squared should not be the primary criteria for selection, and we use test MSE instead. The test MSE of this model is the lowest among all the models that we build, which also supports it to be the most preferable one.



### 3.4 Model Advantages Over Others

3.4.1 Our recommended model is easy to interpret, since it is a linear regression with a sparse structure. For example, for every unit increase in Akt2, Mapk1 decreases 0.41529 unit; for every unit increase in Rik, Mapk1 increases 0.34023 unit; for every unit increase in Sphk2, Mapk1 decreases 0.16635 unit.

3.4.2 It also has the lowest test MSE among all the models that we test, indicating its relatively high accuracy.

3.4.3 Comparing to other nonlinear models, this model has less variance because it is less flexible. In terms of the bias-variance trade-off, this model might have a higher bias than some nonlinear models. However, this model has smaller test MSE than nonlinear models and we believe that the influence of the lower variance is more important in this case.

3.4.4 This model costs less computational time and is less expensive to implement than models with more variables.

### 3.5 Originality of Approach

3.5.1 For the subset selection, unlike the approaches taught in class, we use the training dataset to fit the model instead of using the full dataset. We want to make our dataset consistent, so we will be able to compare the results with other methods (Shrinkage Methods, Dimension Reduction Methods, and Tree-based Methods).

3.5.2 We carefully consider the way we split the data. Since our sample size is relatively small, splitting the data in two halves will make the  $n_{\text{training}}$  less than the number of predictors. Therefore, we decide to choose  $n_{\text{training}} = 30$ , which is especially helpful when we fit the model using the stepwise selections. In fact, we have also attempted to split the data into two halves (except backward stepwise selection due to  $n < p$ ), and compare the results to the one with  $n_{\text{training}} = 30$ . We find there is no significant difference between the two methods.

3.5.3 We check the general assumptions before fitting the models and after obtaining the best model, such as outliers, correlation, normality, and heteroscedasticity.

## Appendix: R Code & Output

```
setwd("D:/Courses/4740 Data Mining and Machine Learning/Final Project")
data = read.csv('Kidney_2.csv')
attach(data)
colnames(data)
```

```
## [1] "Gene.Name" "Expressionb" "Expression" "Expression.1"
## [5] "Expression.2" "Expression.3" "Expression.4" "Expression.5"
## [9] "Expression.6" "Expression.7" "Expression.8" "Expression.9"
## [13] "Expression.10" "Expression.11" "Expression.12" "Expression.13"
## [17] "Expression.14" "Expression.15" "Expression.16" "Expression.17"
## [21] "Expression.18" "Expression.19" "Expression.20" "Expression.21"
## [25] "Expression.22" "Expression.23" "Expression.24" "Expression.25"
## [29] "Expression.26" "Expression.27" "Expression.28" "Expression.29"
## [33] "Expression.30" "Expression.31" "Expression.32" "Expression.33"
## [37] "Expression.34" "Expression.35" "Expression.36" "Expression.37"
## [41] "Expression.38"
```

```
summary(data)
```

```
## Gene.Name Expressionb Expression Expression.1
## Akt2 :1 Min. :-1.75854 Min. :-1.81959 Min. :-1.76608
## Cdc42 :1 1st Qu.: -0.44080 1st Qu.: -0.43828 1st Qu.: -0.43240
## Map2k1 :1 Median : 0.05070 Median : 0.06437 Median :-0.07228
## Map2k2 :1 Mean : 0.05825 Mean : 0.09826 Mean : 0.09216
## Mapk1 :1 3rd Qu.: 0.65903 3rd Qu.: 0.74603 3rd Qu.: 0.75239
## Mapk13 :1 Max. : 1.72994 Max. : 1.93055 Max. : 1.67885
## (Other):18
## Expression.2 Expression.3 Expression.4
## Min. :-1.774458 Min. :-1.85556 Min. :-2.14822
## 1st Qu.: -0.594854 1st Qu.: -0.53336 1st Qu.: -0.49130
## Median : -0.031662 Median : 0.12064 Median : 0.08698
## Mean :-0.007052 Mean : 0.06908 Mean : 0.02971
## 3rd Qu.: 0.611862 3rd Qu.: 0.78581 3rd Qu.: 0.75630
## Max. : 1.526860 Max. : 1.57064 Max. : 1.61456
##
## Expression.5 Expression.6 Expression.7
## Min. :-1.64665 Min. :-1.86184 Min. :-2.09376
## 1st Qu.: -0.63367 1st Qu.: -0.40069 1st Qu.: -0.41019
## Median : -0.02051 Median : 0.16191 Median : 0.03641
## Mean : 0.04455 Mean : 0.09563 Mean : 0.06090
## 3rd Qu.: 0.61224 3rd Qu.: 0.82367 3rd Qu.: 0.73355
## Max. : 2.03056 Max. : 1.70699 Max. : 1.73055
##
## Expression.8 Expression.9 Expression.10
## Min. :-1.64085 Min. :-2.13583 Min. :-1.97821
## 1st Qu.: -0.61198 1st Qu.: -0.41031 1st Qu.: -0.51951
## Median : 0.02591 Median : 0.08112 Median : -0.01609
## Mean : 0.06665 Mean : 0.06481 Mean : 0.04752
```

```

## 3rd Qu.: 0.76093 3rd Qu.: 0.71799 3rd Qu.: 0.77070
## Max. : 1.61626 Max. : 1.59670 Max. : 1.61165
##
## Expression.11 Expression.12 Expression.13
## Min. :-2.06658 Min. :-1.73588 Min. :-1.80871
## 1st Qu.: -0.42503 1st Qu.: -0.46548 1st Qu.: -0.58527
## Median : 0.02725 Median : 0.16161 Median : 0.01807
## Mean : 0.05490 Mean : 0.09172 Mean : 0.03918
## 3rd Qu.: 0.76802 3rd Qu.: 0.73735 3rd Qu.: 0.61591
## Max. : 1.50906 Max. : 1.79096 Max. : 1.88972
##
## Expression.14 Expression.15 Expression.16
## Min. :-1.82506 Min. :-1.74927 Min. :-1.74579
## 1st Qu.: -0.31888 1st Qu.: -0.60218 1st Qu.: -0.52167
## Median : -0.02758 Median : -0.02110 Median : 0.03132
## Mean : 0.04788 Mean : 0.04223 Mean : 0.06121
## 3rd Qu.: 0.57019 3rd Qu.: 0.65008 3rd Qu.: 0.70528
## Max. : 1.59178 Max. : 1.57107 Max. : 1.71218
##
## Expression.17 Expression.18 Expression.19
## Min. :-2.152497 Min. :-2.426034 Min. :-2.0825
## 1st Qu.: -0.353941 1st Qu.: -0.493647 1st Qu.: -0.4130
## Median : -0.006289 Median : -0.068097 Median : 0.1225
## Mean : 0.042417 Mean : 0.004264 Mean : 0.1081
## 3rd Qu.: 0.809804 3rd Qu.: 0.692937 3rd Qu.: 0.8836
## Max. : 1.620685 Max. : 1.633695 Max. : 1.6324
##
## Expression.20 Expression.21 Expression.22
## Min. :-1.61170 Min. :-1.90479 Min. :-2.0719
## 1st Qu.: -0.56782 1st Qu.: -0.70545 1st Qu.: -0.3774
## Median : 0.08991 Median : -0.08574 Median : 0.1868
## Mean : 0.04861 Mean : -0.05435 Mean : 0.1048
## 3rd Qu.: 0.69095 3rd Qu.: 0.58751 3rd Qu.: 0.6082
## Max. : 1.61409 Max. : 1.52689 Max. : 1.7133
##
## Expression.23 Expression.24 Expression.25
## Min. :-1.93373 Min. :-1.24253 Min. :-2.08250
## 1st Qu.: -0.35329 1st Qu.: -0.52789 1st Qu.: -0.37656
## Median : 0.04056 Median : -0.03750 Median : 0.04898
## Mean : 0.07619 Mean : 0.04566 Mean : 0.05284
## 3rd Qu.: 0.68734 3rd Qu.: 0.50023 3rd Qu.: 0.72395
## Max. : 1.81733 Max. : 1.75214 Max. : 1.47236
##
## Expression.26 Expression.27 Expression.28
## Min. :-2.13259 Min. :-1.96593 Min. :-1.77209
## 1st Qu.: -0.42532 1st Qu.: -0.42934 1st Qu.: -0.48495
## Median : 0.08351 Median : 0.08936 Median : 0.10776
## Mean : 0.07427 Mean : 0.07659 Mean : 0.01758
## 3rd Qu.: 0.78653 3rd Qu.: 0.69069 3rd Qu.: 0.42749

```

```

## Max. : 1.88694 Max. : 1.63443 Max. : 1.72834
##
## Expression.29 Expression.30 Expression.31
## Min. :-1.99438 Min. :-1.72659 Min. :-2.13421
## 1st Qu.: -0.37198 1st Qu.: -0.58238 1st Qu.: -0.49991
## Median : 0.06800 Median : -0.05432 Median : 0.07327
## Mean : 0.07737 Mean : 0.07126 Mean : 0.01977
## 3rd Qu.: 0.68615 3rd Qu.: 0.78303 3rd Qu.: 0.65887
## Max. : 1.69495 Max. : 1.72019 Max. : 1.57690
##
## Expression.32 Expression.33 Expression.34
## Min. :-1.77100 Min. :-1.77847 Min. :-1.9872
## 1st Qu.: -0.61497 1st Qu.: -0.32331 1st Qu.: -0.3868
## Median : 0.02450 Median : 0.03393 Median : 0.1578
## Mean : 0.01529 Mean : 0.09440 Mean : 0.0920
## 3rd Qu.: 0.58419 3rd Qu.: 0.76002 3rd Qu.: 0.6726
## Max. : 1.65656 Max. : 1.67891 Max. : 1.6763
##
## Expression.35 Expression.36 Expression.37
## Min. :-1.7557226 Min. :-1.807596 Min. :-1.46292
## 1st Qu.: -0.5335911 1st Qu.: -0.547514 1st Qu.: -0.61455
## Median : -0.0002499 Median : -0.043991 Median : -0.05907
## Mean : 0.0683202 Mean : -0.006718 Mean : 0.03174
## 3rd Qu.: 0.6968382 3rd Qu.: 0.698290 3rd Qu.: 0.75701
## Max. : 1.5951527 Max. : 1.479099 Max. : 1.57256
##
## Expression.38
## Min. :-2.08181
## 1st Qu.: -0.54550
## Median : 0.08216
## Mean : 0.05354
## 3rd Qu.: 0.68442
## Max. : 1.78271
##

name=as.character(data[,1])
Yname=as.character(name[5])
Xname=as.character(name[-5])
names=c(Yname,Xname); names

## [1] "Mapk1" "Cdc42" "Pla2g6" "Akt2" "Plcg2" "Rac2"
## [7] "Rik" "Mapkapk2" "Pik3cd" "Pla2g5" "Sphk2" "Map2k1"
## [13] "Pik3r3" "Ptk2" "Nras" "Nos3" "Pik3r1" "Pik3ca"
## [19] "Ppp3cb" "Map2k2" "Nfatc4" "Mapk13" "Rac1" "Nfat5"

sum(is.na(data))

## [1] 0

X=data[-5,-1]
Y=data[5,-1]

```

```

X=t(X)
Y=t(Y)
data.new=as.data.frame(cbind(Y,X))
colnames(data.new)=names
attach(data.new)
dim(data.new)

## [1] 40 24

cor(data.new)

##          Mapk1   Cdc42   Pla2g6   Akt2   Plcg2
## Mapk1  1.00000000 0.32382541 0.12637487 -0.10043508 0.360000896
## Cdc42  0.32382541 1.00000000 0.30774699 0.25975868 0.255018310
## Pla2g6 0.12637487 0.30774699 1.00000000 0.26176096 0.317954713
## Akt2   -0.10043508 0.25975868 0.26176096 1.00000000 0.144828383
## Plcg2  0.36000090 0.25501831 0.31795471 0.14482838 1.000000000
## Rac2   -0.35006134 0.05735213 0.09128247 0.10883997 0.188068168
## Rik    0.57496897 0.38685532 -0.03453407 0.12507725 0.219222255
## Mapkapk2 -0.28125510 -0.28846823 0.14110570 0.10318078 -0.182079974
## Pik3cd -0.49566490 -0.18732334 -0.29268969 0.12649725 -0.292129995
## Pla2g5 -0.14499454 0.22438630 -0.22002496 -0.05670540 -0.316604952
## Sphk2   0.05458890 0.29264263 0.24988699 0.11023945 0.415586087
## Map2k1  0.50139836 0.43738203 0.19572012 0.06214825 0.231474707
## Pik3r3  0.61388814 0.39797767 0.04119677 0.06425531 0.169985928
## Ptk2    0.29004364 -0.10519658 0.36448698 0.17082901 0.363091177
## Nras    0.06482386 0.35093725 0.20530952 0.37774351 -0.004287570
## Nos3    -0.13624150 -0.20147206 -0.31735883 -0.27245180 -0.565689537
## Pik3r1 -0.08934163 -0.01359486 0.13728390 0.07341933 -0.118117742
## Pik3ca  0.15797316 0.17898354 -0.07182237 0.26809508 0.081046573
## Ppp3cb  0.29229464 -0.02622784 -0.27685686 -0.06100969 -0.045965149
## Map2k2  0.16456399 0.28779249 0.32727972 0.36942637 0.158578378
## Nfatc4  0.20491215 0.25404012 -0.12515883 0.01675147 0.009669447
## Mapk13 -0.27255613 -0.36512189 -0.15870464 -0.16730469 -0.275809745
## Rac1    0.51042472 0.27591724 0.60442420 0.24376503 0.414888624
## Nfat5   0.55318150 0.28207880 -0.08897631 0.18420057 0.257766764
##          Rac2    Rik  Mapkapk2  Pik3cd  Pla2g5
## Mapk1  -0.35006134 0.57496897 -0.28125510 -0.49566490 -0.14499454
## Cdc42   0.05735213 0.38685532 -0.28846823 -0.18732334 0.22438630
## Pla2g6  0.09128247 -0.03453407 0.14110570 -0.29268969 -0.22002496
## Akt2    0.10883997 0.12507725 0.10318078 0.12649725 -0.05670540
## Plcg2   0.18806817 0.21922226 -0.18207997 -0.29212999 -0.31660495
## Rac2    1.00000000 -0.31161692 0.02354855 0.15651959 -0.07622294
## Rik     -0.31161692 1.00000000 -0.38078005 -0.30142694 -0.07251939
## Mapkapk2 0.02354855 -0.38078005 1.00000000 -0.04363767 -0.27495137
## Pik3cd  0.15651959 -0.30142694 -0.04363767 1.00000000 0.18833514
## Pla2g5 -0.07622294 -0.07251939 -0.27495137 0.18833514 1.00000000
## Sphk2   0.06616536 -0.01989209 0.03182342 -0.22813618 0.05721474
## Map2k1  -0.24664354 0.50920904 -0.16898459 -0.46110806 0.13791399
## Pik3r3  -0.42938104 0.66678170 -0.27912921 -0.46690851 0.11272089

```

```

## Ptk2    0.01663222 0.03132119 0.09635901 -0.42289897 -0.24260325
## Nras    -0.20641486 -0.01663286 0.14717324 0.11251555 0.19324609
## Nos3    -0.04010797 -0.20352203 0.06570328 0.32340924 0.34994807
## Pik3r1  -0.22595933 0.15170294 0.12279064 -0.18366427 -0.25914269
## Pik3ca  -0.18604398 0.40202165 -0.30042267 -0.14705000 -0.06436600
## Ppp3cb  0.03401020 0.41758443 -0.18039900 -0.14619717 -0.17567158
## Map2k2  -0.37695501 0.20736165 0.04840621 -0.24501755 0.08252341
## Nfatc4   0.04329162 0.20123839 -0.22877758 -0.08608306 0.30970940
## Mapk13   0.08458620 -0.17560755 0.20450053 0.33719746 -0.16062567
## Rac1     -0.08317212 0.21763299 -0.01132525 -0.52462196 -0.18801920
## Nfat5    -0.47420036 0.60331359 -0.28875219 -0.24134913 0.05126095
##          Sphk2   Map2k1   Pik3r3    Ptk2     Nras
## Mapk1     0.05458890 0.50139836 0.61388814 0.29004364 0.06482386
## Cdc42     0.29264263 0.43738203 0.39797767 -0.10519658 0.35093725
## Pla2g6    0.24988699 0.19572012 0.04119677 0.36448698 0.20530952
## Akt2      0.11023945 0.06214825 0.06425531 0.17082901 0.37774351
## Plcg2     0.41558609 0.23147471 0.16998593 0.36309118 -0.00428757
## Rac2      0.06616536 -0.24664354 -0.42938104 0.01663222 -0.20641486
## Rik       -0.01989209 0.50920904 0.66678170 0.03132119 -0.01663286
## Mapkapk2  0.03182342 -0.16898459 -0.27912921 0.09635901 0.14717324
## Pik3cd   -0.22813618 -0.46110806 -0.46690851 -0.42289897 0.11251555
## Pla2g5    0.05721474 0.13791399 0.11272089 -0.24260325 0.19324609
## Sphk2     1.00000000 0.06758939 0.02447562 0.15746475 0.09680373
## Map2k1    0.06758939 1.00000000 0.65252029 0.18010725 0.19437736
## Pik3r3    0.02447562 0.65252029 1.00000000 0.08929290 0.20387992
## Ptk2      0.15746475 0.18010725 0.08929290 1.00000000 -0.04094120
## Nras      0.09680373 0.19437736 0.20387992 -0.04094120 1.00000000
## Nos3      -0.29281377 -0.32579983 0.01821741 -0.44646160 0.06771539
## Pik3r1    -0.10534110 -0.03614175 0.03395026 0.05038497 -0.10364926
## Pik3ca    -0.03494826 0.05012565 0.24464224 -0.01089647 -0.08248218
## Ppp3cb    -0.15182392 0.15812750 0.29331554 -0.17206838 -0.13189692
## Map2k2    0.07101735 0.22900059 0.28963483 0.20165726 0.48720374
## Nfatc4    -0.05968273 0.20217551 0.29158552 -0.08742012 0.26728094
## Mapk13    -0.17695075 -0.46366949 -0.31957621 -0.39939295 -0.20169950
## Rac1      0.44757901 0.29678604 0.27138084 0.54096671 0.10744659
## Nfat5     0.19373954 0.48120337 0.63494403 0.21983325 0.16555547
##          Nos3    Pik3r1   Pik3ca   Ppp3cb   Map2k2
## Mapk1     -0.13624150 -0.089341629 0.15797316 0.29229464 0.16456399
## Cdc42     -0.20147206 -0.013594861 0.17898354 -0.02622784 0.28779249
## Pla2g6    -0.31735883 0.137283905 -0.07182237 -0.27685686 0.32727972
## Akt2      -0.27245180 0.073419330 0.26809508 -0.06100969 0.36942637
## Plcg2     -0.56568954 -0.118117742 0.08104657 -0.04596515 0.15857838
## Rac2      -0.04010797 -0.225959329 -0.18604398 0.03401020 -0.37695501
## Rik       -0.20352203 0.151702938 0.40202165 0.41758443 0.20736165
## Mapkapk2  0.06570328 0.122790642 -0.30042267 -0.18039900 0.04840621
## Pik3cd    0.32340924 -0.183664267 -0.14705000 -0.14619717 -0.24501755
## Pla2g5    0.34994807 -0.259142694 -0.06436600 -0.17567158 0.08252341
## Sphk2     -0.29281377 -0.105341104 -0.03494826 -0.15182392 0.07101735
## Map2k1    -0.32579983 -0.036141753 0.05012565 0.15812750 0.22900059

```

```

## Pik3r3  0.01821741 0.033950263 0.24464224 0.29331554 0.28963483
## Ptk2    -0.44646160 0.050384970 -0.01089647 -0.17206838 0.20165726
## Nras    0.06771539 -0.103649259 -0.08248218 -0.13189692 0.48720374
## Nos3    1.00000000 -0.384576580 -0.31008440 0.28434093 -0.32596800
## Pik3r1  -0.38457658 1.000000000 0.33393371 -0.33143394 0.20751129
## Pik3ca  -0.31008440 0.333933709 1.00000000 -0.20368174 0.17396555
## Ppp3cb  0.28434093 -0.331433940 -0.20368174 1.00000000 -0.17154639
## Map2k2  -0.32596800 0.207511291 0.17396555 -0.17154639 1.00000000
## Nfatc4  -0.01734103 -0.048258896 0.30889410 -0.16087633 -0.11724755
## Mapk13  0.48565091 -0.159993860 -0.38568380 0.34935612 -0.16733644
## Rac1    -0.44222084 0.112985653 0.06653154 -0.07794746 0.35516361
## Nfat5    -0.20491044 0.009375022 0.23729055 0.22434873 0.27021927
##          Nfatc4  Mapk13  Rac1    Nfat5
## Mapk1    0.204912154 -0.2725561 0.51042472 0.553181495
## Cdc42    0.254040124 -0.3651219 0.27591724 0.282078800
## Pla2g6   -0.125158826 -0.1587046 0.60442420 -0.088976307
## Akt2     0.016751468 -0.1673047 0.24376503 0.184200568
## Plcg2    0.009669447 -0.2758097 0.41488862 0.257766764
## Rac2     0.043291619 0.0845862 -0.08317212 -0.474200361
## Rik      0.201238395 -0.1756075 0.21763299 0.603313591
## Mapkapk2 -0.228777578 0.2045005 -0.01132525 -0.288752195
## Pik3cd   -0.086083058 0.3371975 -0.52462196 -0.241349128
## Pla2g5    0.309709397 -0.1606257 -0.18801920 0.051260950
## Sphk2    -0.059682730 -0.1769508 0.44757901 0.193739541
## Map2k1    0.202175507 -0.4636695 0.29678604 0.481203372
## Pik3r3    0.291585515 -0.3195762 0.27138084 0.634944033
## Ptk2     -0.087420115 -0.3993930 0.54096671 0.219833248
## Nras     0.267280943 -0.2016995 0.10744659 0.165555474
## Nos3     -0.017341032 0.4856509 -0.44222084 -0.204910441
## Pik3r1   -0.048258896 -0.1599939 0.11298565 0.009375022
## Pik3ca    0.308894101 -0.3856838 0.06653154 0.237290545
## Ppp3cb   -0.160876335 0.3493561 -0.07794746 0.224348730
## Map2k2   -0.117247545 -0.1673364 0.35516361 0.270219268
## Nfatc4    1.000000000 -0.4193829 0.05610068 0.119398980
## Mapk13   -0.419382927 1.0000000 -0.32191767 -0.326862901
## Rac1     0.056100679 -0.3219177 1.00000000 0.350064545
## Nfat5     0.119398980 -0.3268629 0.35006454 1.000000000

```

```

#for (i in 1:24) {qqnorm(data.new[i,])}
#pairs(data.new)

```

```

#Data splitting

```

```

set.seed(1)
train=sample(1:40,30)
test=(-train)
x.train=model.matrix(Mapk1~.,data=data.new[train,]),[-1]
x.Train=data.new[train,]
y.train=data.new[train,]$Mapk1
x.test=model.matrix(Mapk1~.,data=data.new[test,]),[-1]

```

```

x.Test=data.new[test,]
y.test=data.new[test,]$Mapk1

#Linear regression
lm.fit=lm(Mapk1~.,data=data.new[train,])
summary(lm.fit)

##
## Call:
## lm(formula = Mapk1 ~ ., data = data.new[train, ])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.05437 -0.03059 -0.01172  0.02696  0.07002
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.193420   1.306921   1.678  0.1443
## Cdc42         0.585876   0.394385   1.486  0.1879
## Pla2g6       -0.082394   0.131337  -0.627  0.5535
## Akt2        -0.741362   0.495726  -1.496  0.1854
## Plcg2        1.150232   0.359618   3.198  0.0186 *
## Rac2        -0.376555   0.209549  -1.797  0.1225
## Rik         -0.492680   0.304637  -1.617  0.1569
## Mapkapk2     0.030011   0.145225   0.207  0.8431
## Pik3cd       0.197100   0.212270   0.929  0.3890
## Pla2g5       0.023063   0.246544   0.094  0.9285
## Sphk2        0.003737   0.185222   0.020  0.9846
## Map2k1       0.339068   0.350323   0.968  0.3705
## Pik3r3      -0.442619   0.308156  -1.436  0.2009
## Ptk2        0.204924   0.448383   0.457  0.6637
## Nras       -0.810464   0.464847  -1.744  0.1319
## Nos3        0.828659   0.340659   2.433  0.0510 .
## Pik3r1       0.578247   0.311648   1.855  0.1129
## Pik3ca      -0.033148   0.444767  -0.075  0.9430
## Ppp3cb       1.200872   0.547025   2.195  0.0706 .
## Map2k2       0.082450   0.367899   0.224  0.8301
## Nfatc4       0.328215   0.351076   0.935  0.3859
## Mapk13      -0.307627   0.171044  -1.799  0.1222
## Rac1        0.401733   0.208347   1.928  0.1021
## Nfat5       0.206987   0.211397   0.979  0.3653
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07955 on 6 degrees of freedom
## Multiple R-squared:  0.9112, Adjusted R-squared:  0.5708
## F-statistic: 2.677 on 23 and 6 DF, p-value: 0.1118

lm.predict=predict(lm.fit,type="response",newdata=x.Test)
mean((y.test-lm.predict)^2)

```



```
## [1] 0.09013329

#all subset selection
library(leaps)

## Warning: package 'leaps' was built under R version 3.2.5

library(fmsb)

## Warning: package 'fmsb' was built under R version 3.2.5

regfit.full=regsubsets(Mapk1~.,data=data.new[train,],nvmax=23)
reg.summary=summary(regfit.full)
reg.summary

## Subset selection object
## Call: regsubsets.formula(Mapk1 ~ ., data = data.new[train, ], nvmax = 23)
## 23 Variables (and intercept)
##      Forced in Forced out
## Cdc42      FALSE  FALSE
## Pla2g6      FALSE  FALSE
## Akt2        FALSE  FALSE
## Plcg2        FALSE  FALSE
## Rac2        FALSE  FALSE
## Rik         FALSE  FALSE
## Mapkapk2    FALSE  FALSE
## Pik3cd      FALSE  FALSE
## Pla2g5      FALSE  FALSE
## Sphk2       FALSE  FALSE
## Map2k1      FALSE  FALSE
## Pik3r3      FALSE  FALSE
## Ptk2        FALSE  FALSE
## Nras        FALSE  FALSE
## Nos3        FALSE  FALSE
## Pik3r1      FALSE  FALSE
## Pik3ca      FALSE  FALSE
## Ppp3cb      FALSE  FALSE
## Map2k2      FALSE  FALSE
## Nfatc4      FALSE  FALSE
## Mapk13      FALSE  FALSE
## Rac1        FALSE  FALSE
## Nfat5       FALSE  FALSE
## 1 subsets of each size up to 23
## Selection Algorithm: exhaustive
##      Cdc42 Pla2g6 Akt2 Plcg2 Rac2 Rik Mapkapk2 Pik3cd Pla2g5 Sphk2
## 1 ( 1 ) " " " " " " " " " " " " " " " " " " " " " " " " " " " "
## 2 ( 1 ) " " " " " " " " " " " " " " " " " " " " " " " " " " " "
## 3 ( 1 ) " " " " " " " " " " " " " " " " " " " " " " " " " " " "
## 4 ( 1 ) " " " " "*" " " " " " " " " " " " " " " " " " " " " " "
## 5 ( 1 ) " " " " "*" " " " " " " " " " " " " " " " " " " " " " "
## 6 ( 1 ) " " " " "*" "*" " " " " " " " " " " " " " " " " " " " " "
```

```

## 7 (1)  " " " " " * " " * " " * " " " " " " " " " " " " " " " "
## 8 (1)  " " " " " * " " * " " * " " " " " " " " " " " * " "
## 9 (1)  " " " " * " " * " " * " " * " " " " " " " " " " " * " "
## 10 (1) " * " " * " " * " " * " " * " " " " " " " " " " " * " "
## 11 (1) " * " " * " " * " " * " " * " " " " " " " " " " " * " "
## 12 (1) " * " " " " " * " " * " " * " " " " " * " " " " " " " * " "
## 13 (1) " * " " " " " * " " * " " * " " * " " " " " " " " " " " "
## 14 (1) " * " " " " " * " " * " " * " " * " " " " " " " " " " " "
## 15 (1) " * " " * " " * " " * " " * " " * " " " " " " " " " " " "
## 16 (1) " * " " " " " * " " * " " * " " * " " " " " " " * " " " " " "
## 17 (1) " * " " * " " " * " " * " " " * " " * " " " " " " " " " " " "
## 18 (1) " * " " * " " " * " " * " " " * " " * " " " " " " " " " " " "
## 19 (1) " * " " * " " " * " " * " " " * " " * " " " " " " " " " " " "
## 20 (1) " * " " * " " " * " " * " " " * " " * " " " " " " " " " " " "
## 21 (1) " * " " * " " " * " " * " " " * " " * " " " " " " " " " " " "
## 22 (1) " * " " * " " " * " " * " " " * " " * " " " " " " " " " " " "
## 23 (1) " * " " * " " " * " " * " " " * " " * " " " " " " " " " " " "

```

```
## 8 ( 1) "*" "*" " "
## 9 ( 1) "*" "*" " "
## 10 ( 1) " " "*" " "
## 11 ( 1) "*" "*" " "
## 12 ( 1) "*" "*" " "
## 13 ( 1) "*" "*" "*"
## 14 ( 1) "*" "*" "*"
## 15 ( 1) "*" "*" "*"
## 16 ( 1) "*" "*" "*"
## 17 ( 1) "*" "*" "*"
## 18 ( 1) "*" "*" "*"
## 19 ( 1) "*" "*" "*"
## 20 ( 1) "*" "*" "*"
## 21 ( 1) "*" "*" "*"
## 22 ( 1) "*" "*" "*"
## 23 ( 1) "*" "*" "*"

```

```
par(mfrow=c(2,2))
plot(reg.summary$rss,xlab="Number of Variables",ylab="RSS",type="l")
plot(reg.summary$adjr2,xlab="Number of Variables",ylab="Adjusted RSq",type="l")
which.max(reg.summary$adjr2)

```

```
## [1] 17

```

```
points(17,reg.summary$adjr2[17],col="red",cex=2,pch=20)
plot(reg.summary$cp,xlab="Number of Variables",ylab="Cp",type="l")
which.min(reg.summary$cp)

```

```
## [1] 8

```

```
points(8,reg.summary$cp[8],col="red",cex=2,pch=20)
which.min(reg.summary$bic)

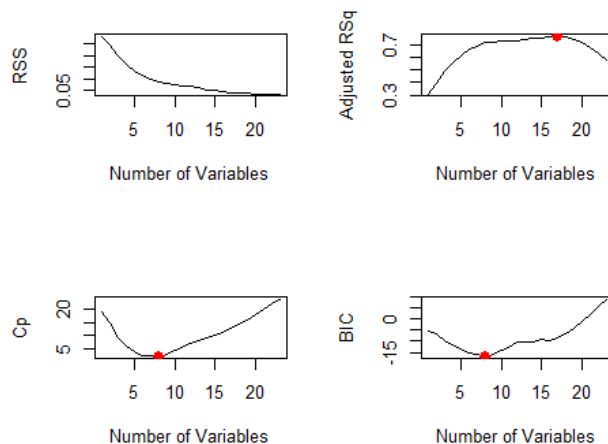
```

```
## [1] 8

```

```
plot(reg.summary$bic,xlab="Number of Variables",ylab="BIC",type="l")
points(8,reg.summary$bic[8],col="red",cex=2,pch=20)

```



reg.summary\$which

```
## (Intercept) Cdc42 Pla2g6 Akt2 Plcg2 Rac2 Rik Mapkapk2 Pik3cd Pla2g5
## 1    TRUE FALSE FALSE FALSE FALSE FALSE TRUE  FALSE FALSE FALSE
## 2    TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 3    TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 4    TRUE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE
## 5    TRUE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE
## 6    TRUE FALSE FALSE TRUE TRUE FALSE FALSE FALSE FALSE FALSE
## 7    TRUE FALSE FALSE TRUE TRUE TRUE FALSE FALSE FALSE FALSE
## 8    TRUE FALSE FALSE TRUE TRUE TRUE FALSE FALSE FALSE FALSE
## 9    TRUE FALSE TRUE TRUE TRUE TRUE FALSE FALSE FALSE FALSE
## 10   TRUE TRUE TRUE TRUE TRUE TRUE FALSE FALSE FALSE FALSE
## 11   TRUE TRUE TRUE TRUE TRUE TRUE FALSE FALSE FALSE FALSE
## 12   TRUE TRUE FALSE TRUE TRUE TRUE FALSE TRUE FALSE FALSE
## 13   TRUE TRUE FALSE TRUE TRUE TRUE TRUE FALSE FALSE FALSE
## 14   TRUE TRUE FALSE TRUE TRUE TRUE TRUE FALSE FALSE FALSE
## 15   TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE FALSE FALSE
## 16   TRUE TRUE FALSE TRUE TRUE TRUE TRUE FALSE TRUE FALSE
## 17   TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE FALSE
## 18   TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE FALSE
## 19   TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE
## 20   TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE
## 21   TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## 22   TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## 23   TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## Sphk2 Map2k1 Pik3r3 Ptk2 Nras Nos3 Pik3r1 Pik3ca Ppp3cb Map2k2
## 1 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 2 FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE TRUE FALSE
## 3 FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE TRUE FALSE
## 4 FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE TRUE FALSE
## 5 FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE TRUE FALSE
## 6 FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE TRUE FALSE
## 7 FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE TRUE FALSE
## 8 TRUE FALSE FALSE FALSE FALSE TRUE FALSE FALSE TRUE FALSE
## 9 TRUE FALSE FALSE FALSE FALSE TRUE FALSE FALSE TRUE FALSE
## 10 TRUE FALSE FALSE FALSE FALSE TRUE FALSE FALSE TRUE TRUE
## 11 TRUE FALSE FALSE FALSE FALSE TRUE FALSE FALSE TRUE TRUE
## 12 TRUE FALSE TRUE FALSE TRUE TRUE FALSE FALSE TRUE FALSE
## 13 FALSE FALSE TRUE FALSE TRUE TRUE TRUE FALSE TRUE FALSE
## 14 FALSE TRUE TRUE FALSE TRUE TRUE TRUE FALSE TRUE FALSE
## 15 FALSE TRUE TRUE FALSE TRUE TRUE TRUE FALSE TRUE FALSE
## 16 FALSE TRUE TRUE FALSE TRUE TRUE TRUE FALSE TRUE FALSE
## 17 FALSE TRUE TRUE FALSE TRUE TRUE TRUE FALSE TRUE FALSE
## 18 FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE FALSE
## 19 FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE FALSE
## 20 FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE
## 21 FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE
## 22 FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## 23 TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

```

## Nfatc4 Mapk13 Rac1 Nfat5
## 1 FALSE FALSE FALSE FALSE
## 2 FALSE FALSE FALSE FALSE
## 3 FALSE FALSE TRUE FALSE
## 4 FALSE FALSE TRUE FALSE
## 5 FALSE TRUE TRUE FALSE
## 6 FALSE TRUE TRUE FALSE
## 7 FALSE TRUE TRUE FALSE
## 8 FALSE TRUE TRUE FALSE
## 9 FALSE TRUE TRUE FALSE
## 10 FALSE FALSE TRUE FALSE
## 11 FALSE TRUE TRUE FALSE
## 12 FALSE TRUE TRUE FALSE
## 13 FALSE TRUE TRUE TRUE
## 14 FALSE TRUE TRUE TRUE
## 15 FALSE TRUE TRUE TRUE
## 16 TRUE TRUE TRUE TRUE
## 17 TRUE TRUE TRUE TRUE
## 18 TRUE TRUE TRUE TRUE
## 19 TRUE TRUE TRUE TRUE
## 20 TRUE TRUE TRUE TRUE
## 21 TRUE TRUE TRUE TRUE
## 22 TRUE TRUE TRUE TRUE
## 23 TRUE TRUE TRUE TRUE

subvar=names(which(reg.summary$which[17,]==T))[-1]
subvar.wi.y=c("Mapk1",subvar)
lm.best.17=lm(Mapk1~.,data=data.new[train,],[,subvar.wi.y])
summary(lm.best.17)

##
## Call:
## lm(formula = Mapk1 ~ ., data = data.new[train, ], subvar.wi.y)
##
## Residuals:
##    Min     1Q   Median     3Q    Max
## -0.05368 -0.02812 -0.01002  0.03599  0.06385
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.12574   0.69260   3.069 0.009731 **
## Cdc42        0.51239   0.21232   2.413 0.032714 *
## Pla2g6       -0.08976   0.08593  -1.045 0.316776
## Akt2         -0.69366   0.24604  -2.819 0.015481 *
## Plcg2        1.06958   0.22168   4.825 0.000416 ***
## Rac2         -0.35882   0.13433  -2.671 0.020366 *
## Rik          -0.49817   0.18203  -2.737 0.018037 *
## Pik3cd       0.16085   0.11245   1.430 0.178119
## Map2k1       0.30713   0.17451   1.760 0.103866
## Pik3r3      -0.37539   0.18059  -2.079 0.059766 .

```

```

## Nras      -0.72369  0.22068 -3.279 0.006587 **
## Nos3       0.77983  0.16205  4.812 0.000425 ***
## Pik3r1     0.52547  0.15422  3.407 0.005199 **
## Ppp3cb     1.10859  0.27891  3.975 0.001844 **
## Nfatc4     0.23592  0.18725  1.260 0.231652
## Mapk13    -0.30188  0.07817 -3.862 0.002261 **
## Rac1       0.45820  0.12746  3.595 0.003679 **
## Nfat5      0.20813  0.14377  1.448 0.173331
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05883 on 12 degrees of freedom
## Multiple R-squared:  0.9029, Adjusted R-squared:  0.7653
## F-statistic: 6.562 on 17 and 12 DF,  p-value: 0.001004

lm.predict=predict(lm.best.17,type="response",newdata=x.Test)
mean((y.test-lm.predict)^2)

## [1] 0.08723346

subvar=names(which(reg.summary$which[8,]==T))[-1]
subvar.wi.y=c("Mapk1",subvar)
lm.best.8=lm(Mapk1~.,data=data.new[train,][,subvar.wi.y])
summary(lm.best.8)

##
## Call:
## lm(formula = Mapk1 ~ ., data = data.new[train, ][, subvar.wi.y])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.11298 -0.04218  0.01143  0.03786  0.08400
##
## Coefficients:
##      Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.60791    0.36742   1.655  0.11288
## Akt2        -0.57736    0.20788  -2.777  0.01129 *
## Plcg2        0.48823    0.15379   3.175  0.00456 **
## Rac2        -0.16681    0.08668  -1.924  0.06796 .
## Sphk2       -0.15673    0.08478  -1.849  0.07862 .
## Nos3        0.23564    0.08561   2.752  0.01193 *
## Ppp3cb      0.56811    0.16059   3.538  0.00195 **
## Mapk13     -0.10691    0.05653  -1.891  0.07247 .
## Rac1        0.35340    0.09856   3.586  0.00174 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06476 on 21 degrees of freedom
## Multiple R-squared:  0.794, Adjusted R-squared:  0.7155
## F-statistic: 10.12 on 8 and 21 DF,  p-value: 1.082e-05

```

```
lm.predict=predict(lm.best.8,type="response",newdata=x.Test)
mean((y.test-lm.predict)^2)
```

```
## [1] 0.01826749
```

### *# Forward Stepwise Selection*

```
regfit.fwd=regsubsets(Mapk1~.,data=data.new[train,],nvmax=23,method="forward")
regfit.fwd.summary=summary(regfit.fwd)
summary(regfit.fwd)
```

```
## Subset selection object
## Call: regsubsets.formula(Mapk1 ~ ., data = data.new[train, ], nvmax = 23,
##   method = "forward")
## 23 Variables (and intercept)
##      Forced in Forced out
## Cdc42      FALSE      FALSE
## Pla2g6      FALSE      FALSE
## Akt2        FALSE      FALSE
## Plcg2        FALSE      FALSE
## Rac2         FALSE      FALSE
## Rik          FALSE      FALSE
## Mapkapk2     FALSE      FALSE
## Pik3cd       FALSE      FALSE
## Pla2g5       FALSE      FALSE
## Sphk2        FALSE      FALSE
## Map2k1       FALSE      FALSE
## Pik3r3       FALSE      FALSE
## Ptk2         FALSE      FALSE
## Nras         FALSE      FALSE
## Nos3         FALSE      FALSE
## Pik3r1       FALSE      FALSE
## Pik3ca       FALSE      FALSE
## Ppp3cb       FALSE      FALSE
## Map2k2       FALSE      FALSE
## Nfatc4       FALSE      FALSE
## Mapk13       FALSE      FALSE
## Rac1         FALSE      FALSE
## Nfat5        FALSE      FALSE
## 1 subsets of each size up to 23
## Selection Algorithm: forward
##      Cdc42 Pla2g6 Akt2 Plcg2 Rac2 Rik Mapkapk2 Pik3cd Pla2g5 Sphk2
## 1 ( 1 ) " " " " " " " " " " " " " " " " " " " " " " " " " " " "
## 2 ( 1 ) " " " " " " " " " " " " " " " " " " " " " " " " " " " "
## 3 ( 1 ) " " " " " * " " " " " " " * " " " " " " " " " " " " " "
## 4 ( 1 ) " " " " " * " " " " " " " * " " " " " " " " " " " " " "
## 5 ( 1 ) " " " " " * " " " " " " " * " " " " " " " " " " " * " "
## 6 ( 1 ) " " " " " * " " " " " " " * " " " " " " " * " " " * " "
## 7 ( 1 ) " " " " " * " " * " " " " * " " " " " " " * " " " * " "
## 8 ( 1 ) " " " " " * " " * " " * " " " " " " " * " " " " " * " "
## 9 ( 1 ) " " " " " * " " * " " * " " " " " " " * " " " " " * " "
```

```

## 10 (1) " " " " "*" "*" "*" "*" " " "*" " " "*"
## 11 (1) " " " " "*" "*" "*" "*" " " "*" " " "*"
## 12 (1) " " "*" "*" "*" "*" "*" "*" " " "*" " " "*"
## 13 (1) "*" "*" "*" "*" "*" "*" " " "*" " " "*"
## 14 (1) "*" "*" "*" "*" "*" "*" " " "*" " " "*"
## 15 (1) "*" "*" "*" "*" "*" "*" " " "*" " " "*"
## 16 (1) "*" "*" "*" "*" "*" "*" " " "*" " " "*"
## 17 (1) "*" "*" "*" "*" "*" "*" " " "*" " " "*"
## 18 (1) "*" "*" "*" "*" "*" "*" " " "*" " " "*"
## 19 (1) "*" "*" "*" "*" "*" "*" " " "*" " " "*"
## 20 (1) "*" "*" "*" "*" "*" "*" " " "*" " " "*"
## 21 (1) "*" "*" "*" "*" "*" "*" "*" "*" "*" " " " " "*"
## 22 (1) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" " " "*"
## 23 (1) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" " " "*"

```

## Map2k1 Pik3r3 Ptk2 Nras Nos3 Pik3r1 Pik3ca Ppp3cb Map2k2 Nfatc4

```

## 1 (1) " " " " " " " " " " " " " " " "
## 2 (1) " " " " " " " " " " " " " " " "
## 3 (1) " " " " " " " " " " " " " " " "
## 4 (1) " " " " " " " " "*" " " " " " " "
## 5 (1) " " " " " " " " "*" " " " " " " "
## 6 (1) " " " " " " " " "*" " " " " " " "
## 7 (1) " " " " " " " " "*" " " " " " " "
## 8 (1) " " " " " " " " "*" " " " " " " "
## 9 (1) " " " " " " " " "*" "*" " " " " " " "
## 10 (1) " " " " " " " " "*" "*" " " " " "*" " " "
## 11 (1) " " " " " " " " "*" "*" " " " " "*" " " "
## 12 (1) " " " " " " " " "*" "*" " " " " "*" " " "
## 13 (1) " " " " " " " " "*" "*" " " " " "*" " " "
## 14 (1) " " " " " " "*" "*" "*" " " " "*" " " "
## 15 (1) " " " " " " "*" "*" "*" " " " "*" "*" " "
## 16 (1) " " "*" " " " "*" "*" "*" " " " "*" "*" " "
## 17 (1) " " "*" " " " "*" "*" "*" " " " "*" "*" " "
## 18 (1) "*" "*" " " " "*" "*" "*" " " " "*" "*" " "
## 19 (1) "*" "*" " " " "*" "*" "*" " " " "*" "*" "*"
## 20 (1) "*" "*" "*" "*" "*" "*" " " " "*" "*" "*"
## 21 (1) "*" "*" "*" "*" "*" "*" " " " "*" "*" "*"
## 22 (1) "*" "*" "*" "*" "*" "*" " " " "*" "*" "*"
## 23 (1) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"

```

## Mapk13 Rac1 Nfat5

```

## 1 (1) " " " " " "
## 2 (1) " " "*" " "
## 3 (1) " " "*" " "
## 4 (1) " " "*" " "
## 5 (1) " " "*" " "
## 6 (1) " " "*" " "
## 7 (1) " " "*" " "
## 8 (1) " " "*" " "
## 9 (1) " " "*" " "
## 10 (1) " " "*" " "

```



```
## 11 ( 1) "*"  "*"  " "
## 12 ( 1) "*"  "*"  " "
## 13 ( 1) "*"  "*"  " "
## 14 ( 1) "*"  "*"  " "
## 15 ( 1) "*"  "*"  " "
## 16 ( 1) "*"  "*"  " "
## 17 ( 1) "*"  "*"  "*"
## 18 ( 1) "*"  "*"  "*"
## 19 ( 1) "*"  "*"  "*"
## 20 ( 1) "*"  "*"  "*"
## 21 ( 1) "*"  "*"  "*"
## 22 ( 1) "*"  "*"  "*"
## 23 ( 1) "*"  "*"  "*"

```

```
par(mfrow=c(2,2))
plot(regfit.fwd.summary$rss,xlab="Number of Variables",ylab="RSS",type="l")
plot(regfit.fwd.summary$adjr2,xlab="Number of Variables",ylab="Adjusted RSq",type="l")
which.max(regfit.fwd.summary$adjr2)

```

```
## [1] 15

```

```
points(15,regfit.fwd.summary$adjr2[15], col="red",cex=2,pch=20)
plot(regfit.fwd.summary$cp,xlab="Number of Variables",ylab="Cp",type="l")
which.min(regfit.fwd.summary$cp)

```

```
## [1] 9

```

```
points(9,regfit.fwd.summary$cp[9],col="red",cex=2,pch=20)
which.min(regfit.fwd.summary$bic)

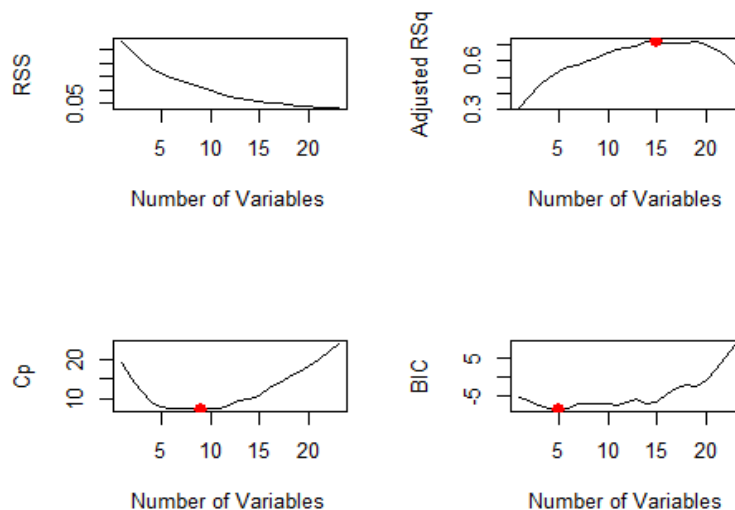
```

```
## [1] 5

```

```
plot(regfit.fwd.summary$bic,xlab="Number of Variables",ylab="BIC",type="l")
points(5,regfit.fwd.summary$bic[5],col="red",cex=2,pch=20)

```



```

subvar=names(which(regfit.fwd.summary$which[15,]==T))[-1]
subvar.wi.y=c("Mapk1",subvar)
lm.fwd.15=glm(Mapk1~.,data=data.new[train,][,subvar.wi.y])
summary(lm.fwd.15)

##
## Call:
## glm(formula = Mapk1 ~ ., data = data.new[train, ][, subvar.wi.y])
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -0.096528 -0.031380 -0.009457  0.031266  0.101899
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.28378   0.69316   1.852  0.08522 .
## Cdc42        0.45933   0.21988   2.089  0.05545 .
## Pla2g6       -0.13749   0.08640  -1.591  0.13387
## Akt2         -0.62105   0.25699  -2.417  0.02990 *
## Plcg2         0.80231   0.20179   3.976  0.00138 **
## Rac2         -0.34285   0.12936  -2.650  0.01902 *
## Rik          -0.30437   0.16797  -1.812  0.09149 .
## Pik3cd        0.09102   0.11969   0.760  0.45960
## Sphk2        -0.10091   0.10071  -1.002  0.33334
## Nras         -0.30378   0.22934  -1.325  0.20652
## Nos3          0.41982   0.14878   2.822  0.01359 *
## Pik3r1        0.24775   0.15263   1.623  0.12684
## Ppp3cb        0.73266   0.30014   2.441  0.02853 *
## Map2k2       -0.17842   0.14607  -1.221  0.24208
## Mapk13       -0.17493   0.09472  -1.847  0.08603 .
## Rac1         0.49281   0.13302   3.705  0.00236 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.003991317)
##
##    Null deviance: 0.427557 on 29  degrees of freedom
## Residual deviance: 0.055878 on 14  degrees of freedom
## AIC: -69.437
##
## Number of Fisher Scoring iterations: 2

lm.predict=predict(lm.fwd.15,type="response",newdata=x.Test)
mean((y.test-lm.predict)^2)

## [1] 0.044479

subvar=names(which(regfit.fwd.summary$which[9,]==T))[-1]
subvar.wi.y=c("Mapk1",subvar)
lm.fwd.9=lm(Mapk1~.,data=data.new[train,][,subvar.wi.y])
summary(lm.fwd.9)

```

```
##
## Call:
## lm(formula = Mapk1 ~ ., data = data.new[train, ], subvar.wi.y))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.110811 -0.042194 -0.004338  0.015799  0.146232
##
## Coefficients:
##      Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.37937   0.36574  -1.037  0.3120
## Akt2        -0.42623   0.22329  -1.909  0.0707 .
## Plcg2        0.50193   0.20238   2.480  0.0222 *
## Rac2        -0.16523   0.09897  -1.670  0.1106
## Rik         0.18512   0.09379   1.974  0.0624 .
## Pik3cd      -0.11429   0.09409  -1.215  0.2386
## Sphk2       -0.19854   0.09077  -2.187  0.0408 *
## Nos3        0.16165   0.10484   1.542  0.1388
## Pik3r1      -0.10266   0.12244  -0.838  0.4117
## Rac1        0.24628   0.11502   2.141  0.0448 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07399 on 20 degrees of freedom
## Multiple R-squared:  0.7439, Adjusted R-squared:  0.6286
## F-statistic: 6.455 on 9 and 20 DF, p-value: 0.0002636

lm.predict=predict(lm.fwd.9,type="response",newdata=x.Test)
mean((y.test-lm.predict)^2)

## [1] 0.01306337

subvar=names(which(regfit.fwd.summary$which[5,]==T))[-1]
subvar.wi.y=c("Mapk1",subvar)
lm.fwd.5=lm(Mapk1~.,data=data.new[train,],subvar.wi.y))
summary(lm.fwd.5)

##
## Call:
## lm(formula = Mapk1 ~ ., data = data.new[train, ], subvar.wi.y))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.127412 -0.063122 -0.000875  0.066505  0.175906
##
## Coefficients:
##      Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.49644   0.23358  -2.125 0.044045 *
## Akt2        -0.41529   0.24018  -1.729 0.096645 .
## Rik         0.34023   0.08878   3.832 0.000805 ***
## Sphk2       -0.16635   0.09674  -1.720 0.098380 .
```

```

## Pik3r1    -0.24678  0.10215 -2.416 0.023668 *
## Rac1      0.39354  0.11004  3.576 0.001525 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08269 on 24 degrees of freedom
## Multiple R-squared:  0.6161, Adjusted R-squared:  0.5362
## F-statistic: 7.705 on 5 and 24 DF,  p-value: 0.0001927

lm.predict=predict(lm.fwd.5,type="response",newdata=x.Test)
mean((y.test-lm.predict)^2)

## [1] 0.007990794

# Backward Stepwise Selection
regfit.bwd=regsubsets(Mapk1~.,data=data.new[train,],nvmax=23,method="backward")
regfit.bwd.summary=summary(regfit.bwd)
summary(regfit.bwd)

## Subset selection object
## Call: regsubsets.formula(Mapk1 ~ ., data = data.new[train, ], nvmax = 23,
##   method = "backward")
## 23 Variables (and intercept)
##      Forced in Forced out
## Cdc42      FALSE      FALSE
## Pla2g6      FALSE      FALSE
## Akt2        FALSE      FALSE
## Plcg2        FALSE      FALSE
## Rac2         FALSE      FALSE
## Rik          FALSE      FALSE
## Mapkapk2     FALSE      FALSE
## Pik3cd       FALSE      FALSE
## Pla2g5       FALSE      FALSE
## Sphk2        FALSE      FALSE
## Map2k1       FALSE      FALSE
## Pik3r3       FALSE      FALSE
## Ptk2         FALSE      FALSE
## Nras         FALSE      FALSE
## Nos3         FALSE      FALSE
## Pik3r1       FALSE      FALSE
## Pik3ca       FALSE      FALSE
## Ppp3cb       FALSE      FALSE
## Map2k2       FALSE      FALSE
## Nfatc4       FALSE      FALSE
## Mapk13       FALSE      FALSE
## Rac1         FALSE      FALSE
## Nfat5        FALSE      FALSE
## 1 subsets of each size up to 23
## Selection Algorithm: backward
##      Cdc42 Pla2g6 Akt2 Plcg2 Rac2 Rik Mapkapk2 Pik3cd Pla2g5 Sphk2
## 1 ( 1 ) " " " " " " " " " " " " " " " " " " " " " " " " " "

```

```

## 2 (1) " " " " " " " " " " " " " " " "
## 3 (1) " " " " "*" " " " " " " " " " " " "
## 4 (1) " " " " "*" " " " " " " " " " " " "
## 5 (1) " " " " "*" " " " " " " " " " " " "
## 6 (1) " " " " "*" "*" " " " " " " " " " "
## 7 (1) " " " " "*" "*" "*" " " " " " " " "
## 8 (1) " " " " "*" "*" "*" " " " " " " " "
## 9 (1) "*" " " " "*" "*" "*" " " " " " " " "
## 10 (1) "*" " " " "*" "*" "*" " " " " " " " "
## 11 (1) "*" " " " "*" "*" "*" " " " " " " " "
## 12 (1) "*" " " " "*" "*" "*" "*" " " " " " " "
## 13 (1) "*" " " " "*" "*" "*" "*" " " " " " " "
## 14 (1) "*" " " " "*" "*" "*" "*" " " " " " " "
## 15 (1) "*" " " " "*" "*" "*" "*" "*" " " " " " "
## 16 (1) "*" " " " "*" "*" "*" "*" "*" " " " " " "
## 17 (1) "*" "*" " "*" "*" "*" "*" " " " " " " "
## 18 (1) "*" "*" " "*" "*" "*" "*" " " " " " " "
## 19 (1) "*" "*" " "*" "*" "*" "*" " " " " " " "
## 20 (1) "*" "*" " "*" "*" "*" "*" "*" " " " " " "
## 21 (1) "*" "*" " "*" "*" "*" "*" "*" " " " " " "
## 22 (1) "*" "*" " "*" "*" "*" "*" "*" " " " " " "
## 23 (1) "*" "*" " "*" "*" "*" "*" "*" " " " " "*"

```

## Map2k1 Pik3r3 Ptk2 Nras Nos3 Pik3r1 Pik3ca Ppp3cb Map2k2 Nfatc4

```

## 1 (1) " " " " " " " " " " " " "*" " " " "
## 2 (1) " " " " " " " " " " " " "*" " " " "
## 3 (1) " " " " " " " " " " " " "*" " " " "
## 4 (1) " " " " " " " " " " " " "*" " " " "
## 5 (1) " " " " " " " " "*" " " " " " "*" " " " "
## 6 (1) " " " " " " " " "*" " " " " " "*" " " " "
## 7 (1) " " " " " " " " "*" " " " " " "*" " " " "
## 8 (1) " " " " " " "*" " "*" " " " " " "*" " " " "
## 9 (1) " " " " " " "*" "*" " " " " " "*" " " " "
## 10 (1) " " "*" " " " "*" "*" "*" " " " " "*" " " " "
## 11 (1) " " "*" " " " "*" "*" "*" " " " " "*" " " " "
## 12 (1) " " "*" " " " "*" "*" "*" " " " " "*" " " " "
## 13 (1) " " "*" " " " "*" "*" "*" " " " " "*" " " " "
## 14 (1) "*" "*" " " " "*" "*" "*" " " " " "*" " " " "
## 15 (1) "*" "*" " " " "*" "*" "*" " " " " "*" " " " "
## 16 (1) "*" "*" " " " "*" "*" "*" " " " " "*" " " "*"
## 17 (1) "*" "*" " " " "*" "*" "*" " " " " "*" " " "*"
## 18 (1) "*" "*" " "*" "*" "*" "*" " " " " "*" " " "*"
## 19 (1) "*" "*" " "*" "*" "*" "*" " " " " "*" " "*" "*"
## 20 (1) "*" "*" " "*" "*" "*" "*" " " " " "*" " "*" "*"
## 21 (1) "*" "*" " "*" "*" "*" "*" " " " " "*" " "*" "*"
## 22 (1) "*" "*" " "*" "*" "*" "*" "*" " " " " "*" "*"
## 23 (1) "*" "*" " "*" "*" "*" "*" "*" " " " " "*" "*"

```

## Mapk13 Rac1 Nfat5

```

## 1 (1) " " " " " "
## 2 (1) " " "*" " "

```

```
## 3 ( 1) " " "*" " "
## 4 ( 1) "*" "*" " "
## 5 ( 1) "*" "*" " "
## 6 ( 1) "*" "*" " "
## 7 ( 1) "*" "*" " "
## 8 ( 1) "*" "*" " "
## 9 ( 1) "*" "*" " "
## 10 ( 1) "*" "*" " "
## 11 ( 1) "*" "*" " "
## 12 ( 1) "*" "*" " "
## 13 ( 1) "*" "*" "*"
## 14 ( 1) "*" "*" "*"
## 15 ( 1) "*" "*" "*"
## 16 ( 1) "*" "*" "*"
## 17 ( 1) "*" "*" "*"
## 18 ( 1) "*" "*" "*"
## 19 ( 1) "*" "*" "*"
## 20 ( 1) "*" "*" "*"
## 21 ( 1) "*" "*" "*"
## 22 ( 1) "*" "*" "*"
## 23 ( 1) "*" "*" "*"

```

```
par(mfrow=c(2,2))
plot(regfit.bwd.summary$rss,xlab="Number of Variables",ylab="RSS",type="l")
plot(regfit.bwd.summary$adjr2,xlab="Number of Variables",ylab="Adjusted RSq",type="l")
which.max(regfit.bwd.summary$adjr2)

```

```
## [1] 17

```

```
points(17,regfit.bwd.summary$adjr2[17], col="red",cex=2,pch=20)
plot(regfit.bwd.summary$cp,xlab="Number of Variables",ylab="Cp",type="l")
which.min(regfit.bwd.summary$cp)

```

```
## [1] 7

```

```
points(7,regfit.bwd.summary$cp[7],col="red",cex=2,pch=20)
which.min(regfit.bwd.summary$bic)

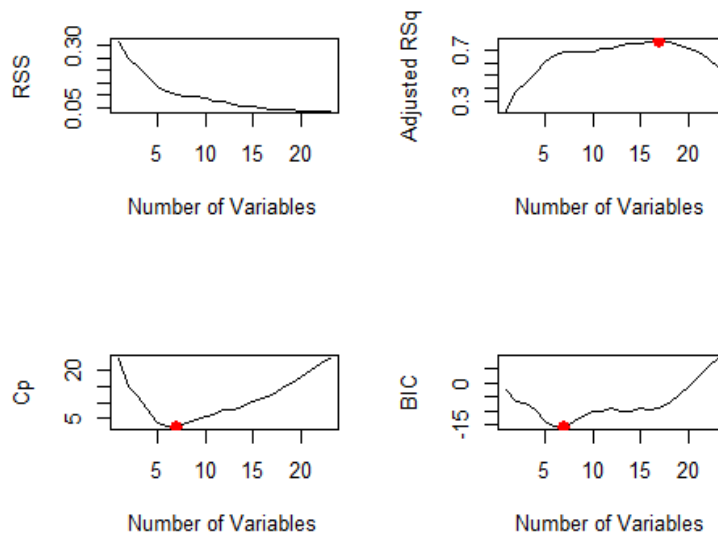
```

```
## [1] 7

```

```
plot(regfit.bwd.summary$bic,xlab="Number of Variables",ylab="BIC",type="l")
points(7,regfit.bwd.summary$bic[7],col="red",cex=2,pch=20)

```



```
subvar=names(which(regfit.bwd.summary$which[17,]==T))[-1]
subvar.wi.y=c("Mapk1",subvar)
lm.bwd.17=lm(Mapk1~.,data=data.new[train,],[subvar.wi.y])
summary(lm.bwd.17)
```

```
##
## Call:
## lm(formula = Mapk1 ~ ., data = data.new[train, ], subvar.wi.y)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.05368 -0.02812 -0.01002  0.03599  0.06385
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.12574    0.69260   3.069 0.009731 **
## Cdc42         0.51239    0.21232   2.413 0.032714 *
## Pla2g6       -0.08976    0.08593  -1.045 0.316776
## Akt2        -0.69366    0.24604  -2.819 0.015481 *
## Plcg2        1.06958    0.22168   4.825 0.000416 ***
## Rac2        -0.35882    0.13433  -2.671 0.020366 *
## Rik         -0.49817    0.18203  -2.737 0.018037 *
## Pik3cd       0.16085    0.11245   1.430 0.178119
## Map2k1       0.30713    0.17451   1.760 0.103866
## Pik3r3      -0.37539    0.18059  -2.079 0.059766 .
## Nras        -0.72369    0.22068  -3.279 0.006587 **
## Nos3        0.77983    0.16205   4.812 0.000425 ***
## Pik3r1       0.52547    0.15422   3.407 0.005199 **
## Ppp3cb       1.10859    0.27891   3.975 0.001844 **
## Nfatc4       0.23592    0.18725   1.260 0.231652
## Mapk13      -0.30188    0.07817  -3.862 0.002261 **
## Rac1        0.45820    0.12746   3.595 0.003679 **
## Nfat5       0.20813    0.14377   1.448 0.173331
```

```

## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05883 on 12 degrees of freedom
## Multiple R-squared:  0.9029, Adjusted R-squared:  0.7653
## F-statistic: 6.562 on 17 and 12 DF,  p-value: 0.001004

lm.predict=predict(lm.bwd.17,type="response",newdata=x.Test)
mean((y.test-lm.predict)^2)

## [1] 0.08723346

subvar=names(which(regfit.bwd.summary$which[7,]==T))[-1]
subvar.wi.y=c("Mapk1",subvar)
lm.bwd.7=lm(Mapk1~.,data=data.new[train,][,subvar.wi.y])
summary(lm.bwd.7)

##
## Call:
## lm(formula = Mapk1 ~ ., data = data.new[train, ][, subvar.wi.y])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.12165 -0.03929 -0.01180  0.04514  0.13765
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.63525   0.38677   1.642  0.11471
## Akt2        -0.69710   0.20811  -3.350  0.00290 **
## Plcg2         0.40520   0.15496   2.615  0.01582 *
## Rac2        -0.15455   0.09105  -1.697  0.10373
## Nos3         0.27077   0.08794   3.079  0.00549 **
## Ppp3cb       0.60007   0.16820   3.568  0.00172 **
## Mapk13      -0.15030   0.05418  -2.774  0.01107 *
## Rac1         0.35147   0.10383   3.385  0.00266 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06823 on 22 degrees of freedom
## Multiple R-squared:  0.7605, Adjusted R-squared:  0.6843
## F-statistic: 9.979 on 7 and 22 DF,  p-value: 1.395e-05

lm.predict=predict(lm.bwd.7,type="response",newdata=x.Test)
mean((y.test-lm.predict)^2)

## [1] 0.02199643

#Ridge Regression
set.seed(1)
library(glmnet)

## Warning: package 'glmnet' was built under R version 3.2.5

```



```

## Loading required package: Matrix
## Loading required package: foreach

## Warning: package 'foreach' was built under R version 3.2.5

## Loaded glmnet 2.0-5

grid=10^seq(10,-2,length=100)
ridge.mod=glmnet(x.train,y.train,alpha=0,lambda=grid)
dim(coef(ridge.mod))

## [1] 24 100

set.seed(1)
cv.out=cv.glmnet(x.train,y.train,alpha=0,nfolds=5)
bestlam=cv.out$lambda.min
bestlam #1.014296

## [1] 0.3435062

pred.ridge=predict(ridge.mod,s=bestlam,newx=x.test)
mean((y.test-pred.ridge)^2)

## [1] 0.01112933

#LASSO Regression
set.seed(1)
cv.out=cv.glmnet(x.train,y.train,alpha=1,nfold=5)
bestlam=cv.out$lambda.min
bestlam #0.004707944

## [1] 0.04334557

lasso.mod=glmnet(x.train,y.train,alpha=1,lambda=bestlam)
lasso.coef=coef(lasso.mod)[,1]
lasso.coef[lasso.coef!=0]

## (Intercept)    Rik    Pik3r3    Ppp3cb
## 0.001082049 0.075613447 0.091374216 0.043440758

pred.lasso=predict(lasso.mod,s=bestlam,newx=x.test)
mean((y.test-pred.lasso)^2) #0.0005978611

## [1] 0.01575946

#Principal Components Regression
library(pls)

## Warning: package 'pls' was built under R version 3.2.5

##
## Attaching package: 'pls'
##
## The following object is masked from 'package:stats':

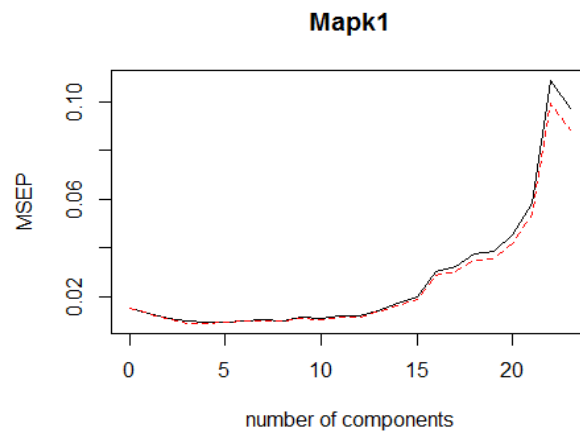
```

```
##
## loadings

set.seed(1)
pcr.fit=pcr(Mapk1~.,data=data.new[train,],scale=TRUE,validation="CV")
summary(pcr.fit) # M=4 has smallest CV error

## Data: X dimension: 30 23
## Y dimension: 30 1
## Fit method: svdpc
## Number of components considered: 23
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
## (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps
## CV 0.1235 0.1143 0.1055 0.09907 0.09628 0.09775 0.0999
## adjCV 0.1235 0.1138 0.1046 0.09541 0.09454 0.09684 0.0986
## 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps
## CV 0.1012 0.1008 0.1066 0.1049 0.1097 0.1092 0.1188
## adjCV 0.1001 0.1001 0.1055 0.1030 0.1074 0.1066 0.1160
## 14 comps 15 comps 16 comps 17 comps 18 comps 19 comps
## CV 0.1314 0.1415 0.1744 0.1797 0.1937 0.1960
## adjCV 0.1280 0.1375 0.1690 0.1734 0.1870 0.1883
## 20 comps 21 comps 22 comps 23 comps
## CV 0.2126 0.2409 0.3296 0.3117
## adjCV 0.2037 0.2309 0.3152 0.2968
##
## TRAINING: % variance explained
## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps
## X 20.00 36.11 47.10 57.36 65.58 71.30 76.16
## Mapk1 26.96 38.53 54.83 55.16 56.63 59.15 59.54
## 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps 14 comps
## X 80.51 84.10 87.27 90.01 92.02 93.84 95.41
## Mapk1 60.82 62.77 67.58 68.25 69.15 69.16 69.49
## 15 comps 16 comps 17 comps 18 comps 19 comps 20 comps
## X 96.63 97.56 98.29 98.85 99.26 99.56
## Mapk1 69.99 70.26 74.90 76.27 80.40 82.69
## 21 comps 22 comps 23 comps
## X 99.79 99.94 100.00
## Mapk1 82.83 83.03 91.12

validationplot(pcr.fit,val.type="MSEP")
```



```
pcr.fit=pcr(Mapk1~.,data=data.new[train,],scale=TRUE,ncomp=4) #refit the model with M=4
summary(pcr.fit)
```

```
## Data:  X dimension: 30 23
## Y dimension: 30 1
## Fit method: svdpc
## Number of components considered: 4
## TRAINING: % variance explained
##      1 comps 2 comps 3 comps 4 comps
## X      20.00 36.11 47.10 57.36
## Mapk1  26.96 38.53 54.83 55.16
```

```
coef(pcr.fit)
```

```
## , , 4 comps
##
##      Mapk1
## Cdc42  0.0016846292
## Pla2g6 -0.0039259610
## Akt2   -0.0041321187
## Plcg2  0.0113518535
## Rac2   -0.0054099879
## Rik    0.0207740831
## Mapkapk2 -0.0101260089
## Pik3cd -0.0136267800
## Pla2g5 -0.0076242245
## Sphk2  0.0015185439
## Map2k1  0.0121214549
## Pik3r3  0.0172366624
## Ptk2    0.0037903856
## Nras    -0.0098624860
## Nos3    -0.0003698385
## Pik3r1 -0.0037810549
## Pik3ca  0.0028095643
## Ppp3cb  0.0207388866
## Map2k2  -0.0041637594
## Nfatc4  -0.0001233080
```

```

## Mapk13 0.0035046908
## Rac1 0.0065333399
## Nfat5 0.0155394602

pcr.pred=predict(pcr.fit,data=data.new[test,],ncomp=4)
mean((pcr.pred-y.test)^2)

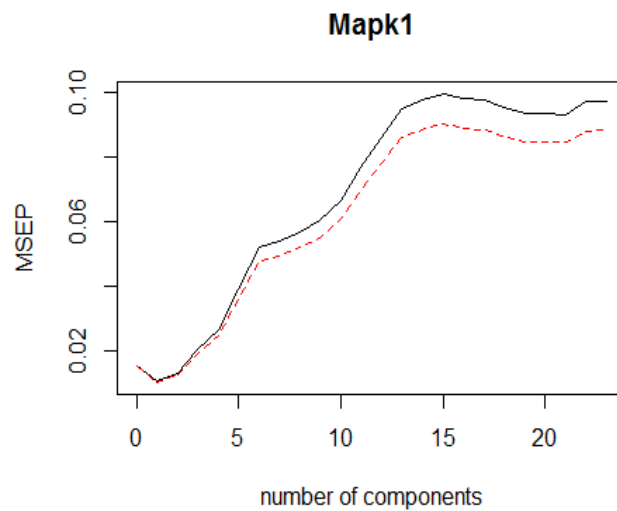
## [1] 0.02653677

#Partial Least Squares
set.seed(1)
pls.fit=plsr(Mapk1~.,data=data.new[train,],scale=TRUE,validation="CV")
summary(pls.fit)

## Data: X dimension: 30 23
## Y dimension: 30 1
## Fit method: kernelppls
## Number of components considered: 23
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
## (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps
## CV 0.1235 0.1026 0.1133 0.1429 0.1625 0.1976 0.2281
## adjCV 0.1235 0.1013 0.1110 0.1386 0.1570 0.1897 0.2182
## 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps
## CV 0.2321 0.2385 0.2454 0.2578 0.2769 0.2931 0.3080
## adjCV 0.2220 0.2280 0.2346 0.2463 0.2642 0.2794 0.2933
## 14 comps 15 comps 16 comps 17 comps 18 comps 19 comps
## CV 0.3121 0.3154 0.3134 0.3123 0.3086 0.3058
## adjCV 0.2971 0.3002 0.2983 0.2973 0.2939 0.2912
## 20 comps 21 comps 22 comps 23 comps
## CV 0.3053 0.3050 0.3115 0.3117
## adjCV 0.2907 0.2904 0.2966 0.2968
##
## TRAINING: % variance explained
## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps
## X 17.88 29.88 39.05 49.58 54.24 58.94 64.44
## Mapk1 59.24 68.56 73.83 76.62 80.69 83.45 84.51
## 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps 14 comps
## X 69.17 76.10 79.67 82.06 84.75 87.12 89.76
## Mapk1 85.46 86.13 87.10 88.14 88.98 89.95 90.56
## 15 comps 16 comps 17 comps 18 comps 19 comps 20 comps
## X 91.80 94.00 95.25 96.28 96.97 97.76
## Mapk1 90.92 91.01 91.08 91.11 91.12 91.12
## 21 comps 22 comps 23 comps
## X 99.10 99.77 100.00
## Mapk1 91.12 91.12 91.12

validationplot(pls.fit,val.type="MSEP")

```



```
pls.fit=plsr(Mapk1~.,data=data.new[train,],scale=TRUE,ncomp=1)
summary(pls.fit)
```

```
## Data:  X dimension: 30 23
## Y dimension: 30 1
## Fit method: kernelpls
## Number of components considered: 1
## TRAINING: % variance explained
##      1 comps
## X      17.88
## Mapk1  59.24
```

```
coef(pls.fit)
```

```
## , , 1 comps
##
##      Mapk1
## Cdc42  0.0040948621
## Pla2g6 -0.0012538475
## Akt2   -0.0039366372
## Plcg2  0.0127253105
## Rac2   -0.0101121424
## Rik    0.0185167927
## Mapkapk2 -0.0109144571
## Pik3cd -0.0134504705
## Pla2g5 -0.0036711029
## Sphk2  -0.0039436946
## Map2k1  0.0124796483
## Pik3r3  0.0184369632
## Ptk2    0.0069211937
## Nras   -0.0023888549
## Nos3    0.0033595571
## Pik3r1 -0.0060127035
## Pik3ca  0.0017477728
## Ppp3cb  0.0163049446
```

```

## Map2k2 -0.0006560712
## Nfatc4 0.0054499763
## Mapk13 -0.0022034685
## Rac1 0.0126386870
## Nfat5 0.0165386719

pls.pred=predict(pls.fit,data=data.new[test,],ncomp=1)
mean((pls.pred-y.test)^2)

## [1] 0.02842974

#Bagging
library(randomForest)

## Warning: package 'randomForest' was built under R version 3.2.5

## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.

set.seed(1)
bag=randomForest(Mapk1~.,data=data.new[train,],mtry=23,importance=TRUE)
bag

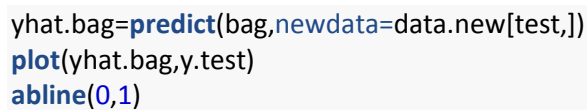
##
## Call:
## randomForest(formula = Mapk1 ~ ., data = data.new[train, ], mtry = 23, importance = TRUE)
##      Type of random forest: regression
##      Number of trees: 500
## No. of variables tried at each split: 23
##
##      Mean of squared residuals: 0.01186703
##      % Var explained: 16.73

importance(bag)

##      %IncMSE IncNodePurity
## Cdc42 4.0188765 0.018980219
## Pla2g6 2.3869442 0.004450571
## Akt2 -0.5734589 0.006523327
## Plcg2 4.5362761 0.032403609
## Rac2 -1.3901157 0.022438143
## Rik 4.1704053 0.050762947
## Mapkapk2 -0.1532739 0.004622139
## Pik3cd 0.1117296 0.020583177
## Pla2g5 1.6926099 0.003270276
## Sphk2 1.8376770 0.003811297
## Map2k1 2.1157952 0.011204442
## Pik3r3 5.0920056 0.047309892
## Ptk2 -0.7739693 0.008735872
## Nras -1.1857853 0.004624791
## Nos3 1.0153718 0.004931399
## Pik3r1 -1.9235202 0.012861336

```

```
varImpPlot(bag)
```



```
#Random Forest
set.seed(1)
```

```
rf=randomForest(Mapk1~.,data=data.new[train,],mtry=8,importance=TRUE)
rf
```

```
##
```

```
## Call:
```

```
## randomForest(formula = Mapk1 ~ ., data = data.new[train, ], mtry = 8, importance = TRUE)
```

```
##           Type of random forest: regression
```

```
##           Number of trees: 500
```

```
## No. of variables tried at each split: 8
```

```
##
```

```
##           Mean of squared residuals: 0.01080199
```

```
##           % Var explained: 24.21
```

```
importance(rf)
```

```
##           %IncMSE IncNodePurity
```

```
## Cdc42  2.8098306  0.021369978
```

```
## Pla2g6  0.5639862  0.007392889
```

```
## Akt2   1.1235487  0.008432712
```

```
## Plcg2   3.2834771  0.023425856
```

```
## Rac2    0.2086983  0.028289836
```

```
## Rik     3.8938810  0.047443010
```

```
## Mapkapk2 -0.2165913  0.004945067
```

```
## Pik3cd  1.7850925  0.017498076
```

```
## Pla2g5 -1.0061727  0.004331946
```

```
## Sphk2    0.7291081  0.009093605
```

```
## Map2k1   0.7947904  0.015056493
```

```
## Pik3r3   4.6194303  0.043732219
```

```
## Ptk2     0.8501995  0.011438922
```

```
## Nras     1.9607210  0.007098212
```

```
## Nos3    -0.4321791  0.006913462
```

```
## Pik3r1  -1.4368097  0.009430594
```

```
## Pik3ca  -0.8816951  0.005029347
```

```
## Ppp3cb   9.5253518  0.050113710
```

```
## Map2k2   0.1563517  0.003861445
```

```
## Nfatc4  -0.7617402  0.009559536
```

```
## Mapk13  -0.2345716  0.007167415
```

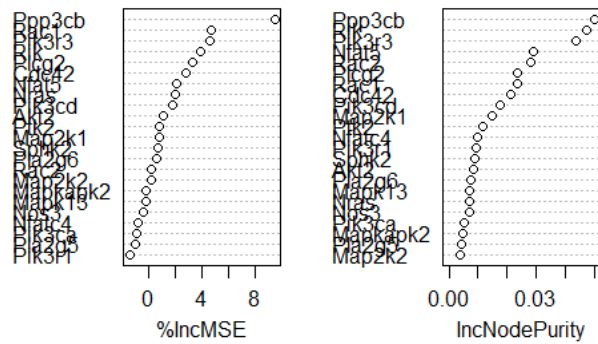
```
## Rac1     4.6800712  0.023326726
```

```
## Nfat5    2.1015002  0.028876190
```

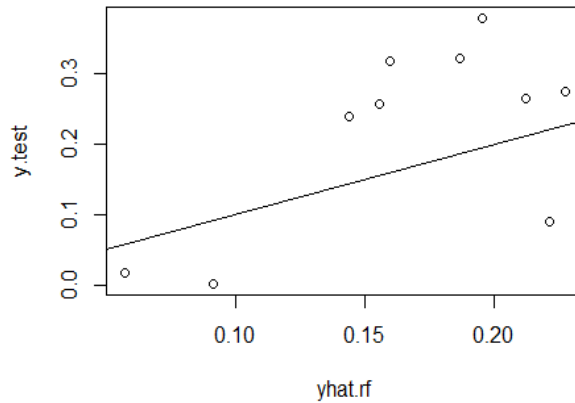
```
varImpPlot(rf)
```



rf



```
yhat.rf=predict(rf,newdata=data.new[test,])
plot(yhat.rf,y.test)
abline(0,1)
```



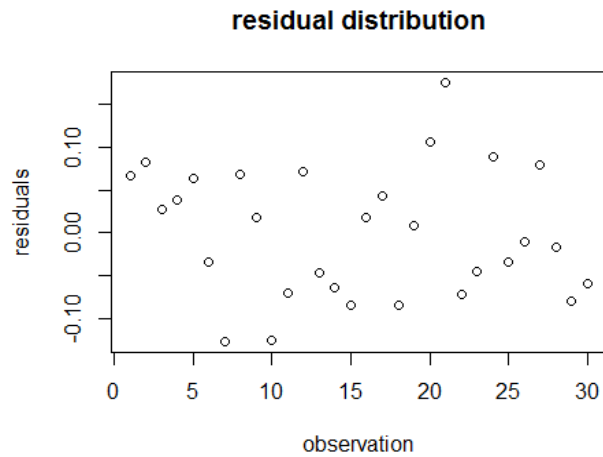
```
mean((yhat.rf-y.test)^2)
```

```
## [1] 0.01283373
```

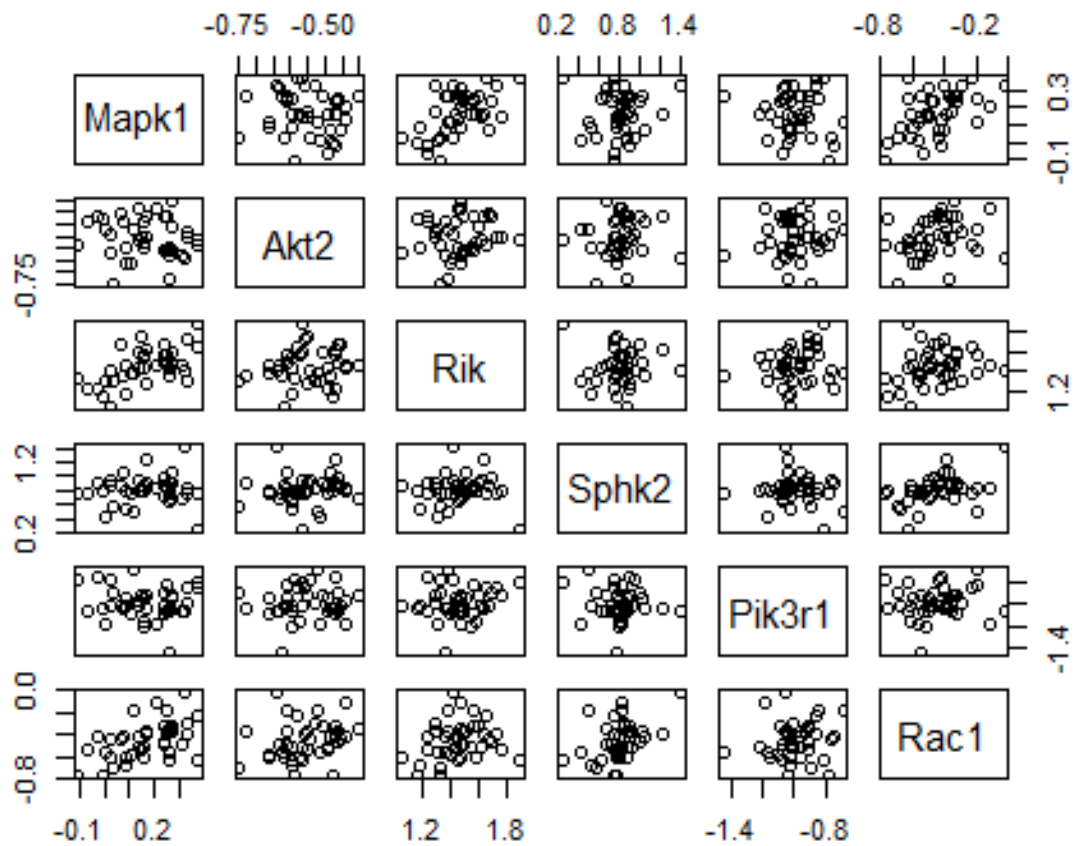
*#Check model violation and correlation btw predictors*

```
fit.best=lm(Mapk1~Akt2+Rik+Sphk2+Pik3r1+Rac1,data=data.new[train,])
```

```
plot(fit.best$residuals,ylab="residuals",xlab="observation",main="residual distribution")
```



```
pairs(data.new[,c("Mapk1", "Akt2", "Rik", "Sphk2", "Pik3r1", "Rac1")])
```



*#Compare Ridge and Lasso MSEs*

```
ridge.err=matrix(0,nrow=50,ncol=1)
```

```
lasso.err=matrix(0,nrow=50,ncol=1)
```

```
for(i in 1:50){
```

```
  set.seed(i)
```

```

train=sample(1:40,30)
test=(-train)
x=model.matrix(Mapk1~.,data.new[train,]),[-1]
y=data.new[train,]$Mapk1
xtest=model.matrix(Mapk1~.,data.new[test,]),[-1]
ytest=data.new[test,]$Mapk1

cv.out=cv.glmnet(x,y,alpha=0, nfold=5)
bestlam.ridge=cv.out$lambda.min
ridge.mod=glmnet(x,y,alpha=0,lambda=bestlam.ridge)
pred.ridge=predict(ridge.mod,s=bestlam.ridge,newx=xtest)
ridge.err[i,1]=mean((pred.ridge-ytest)^2)

cv.out=cv.glmnet(x,y,alpha=1,nfold=5)
bestlam.lasso=cv.out$lambda.min
lasso.mod=glmnet(x,y,alpha=1,lambda=bestlam.lasso)
pred.lasso=predict(lasso.mod,s=bestlam.lasso,newx=xtest)
lasso.err[i,1]=mean((pred.lasso-ytest)^2)
}
boxplot(cbind(ridge.err,lasso.err))

```

