

Prediction with Linear Modelling

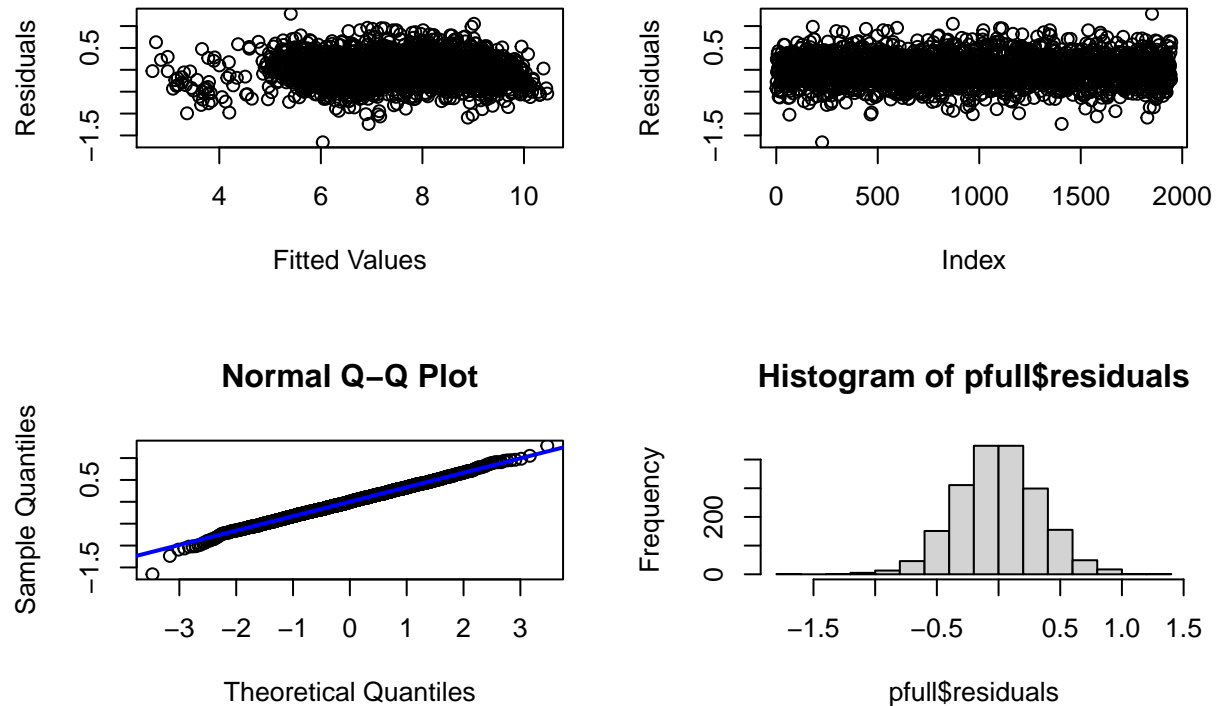
Kevin Li

12/11/2020

Summary

The objective of my analysis is to construct a fitted model based on the 1946 observations of computer-generated structures for the Covid-19 protein, and use that fitted model to predict the values of the response variables of another data set with a further 1946 observations. I decided to start off by constructing the full model with all the predictors based on the multiple linear regression model. I then decided to remove multicollinearity from the 685 predictors in protein-train.csv. I assessed regression model assumptions, and addressed outliers, then tested different model selection methods on a single train/validation set. I considered forwards selection, backwards selection, and forwards/backwards selection, with both Akaike information criterion (AIC) and Bayesian information criterion (BIC). I also considered Iterative conditional minimization (ICM) as a selection method, with AIC and BIC, and a harsher penalty than BIC. I then chose the best candidates and cross validated them, and found the best model selection method, which I then applied on the full data set in order to get the final model. I then assessed multiple linear regression assumptions for my final model. Finally, I used my final model to predict the values of the response variables of protein-test.csv. I found in my main results that forward selection with AIC as criteria had the lowest average root mean squared error (RMSPE), after cross validation, and so the model constructed with forward selection with AIC criteria would be my final model.

Exploratory analysis of dataset



In the data, there are 1946 observations and 685 predictors with 1 response. I observed that in the data, there are no categorical predictors, and that there are cases of perfect multicollinearity, indicating I should remove those predictors. In order to take a closer look at the data, I decided to construct a full multiple linear regression model based on all the observations and predictors given, and plot the residuals. In the residuals vs fitted values, we see that the data is randomly scattered around $y=0$, indicating the independence assumption is satisfied. We see that since the data was generated in temporal order, we also look at the residuals vs index plot, which also satisfies model assumptions. From the histogram and QQ plot of the residuals, we see that the normality assumption is also satisfied. The R squared value of the model is 0.9482963, which is very large and either indicates the full model is very accurate at predicting or overfitting. So we have an idea on reducing predictors to get a better model for prediction.

Methods

In order to remove multicollinearity from the data, I used the `vif` function from the “faraway” package in order to determine the variance inflation factor (VIF) of each predictor and I then used the method of removing the predictor with the largest VIF in a while loop until the largest VIF was less than 10.

In order to determine whether or not I should use a model transformation, I examined residual plots (residuals vs fitted values, residuals vs index, as well as the QQ plot and histogram of residuals). I did not look at residuals vs predictors since there are too many. Since the residual plots did not reveal any problems with assumptions, I decided not to use any transformations with my multiple linear regression model.

I also considered the effects of individual observations on my fitted multiple linear regression model. I recorded the studentized residuals, leverages and Cook’s distances for my model (after removing multicollinearity). I then plotted studentized residuals vs fitted values, and noted the observations that lay outside of 3 standard deviations. I also plotted the leverages and noted the leverages greater than twice the average. Then, to put it together, I looked at a plot of the Cook’s distances. Cook’s distances of > 0.5 are generally considered outliers. Thus, I found no outliers in the data. However, I would like to note that I used 0.5 as a cutoff point since it is the one used in the lectures, but some other general rule of thumb cutoff points are $4/(\text{number of observations})$, or $4/(\text{number of observations} - \text{number of predictors} - 1)$. When testing the latter cutoff point, I got around 200 outliers. Due to time constraints, I decided not to test its impact on prediction in a train/validation split, but I would have if time was not an issue.

Now that I removed multicollinearity and considered the effects of individual observations, I began to consider different model selection strategies.

I noted that since there were more than 500 predictors even after removing multicollinearity, it was too difficult to do an all possible regressions search. Thus I started off with a single train/validation split (80/20) and tested out each of forward/backward/forward-backward selections with both AIC and BIC as criteria. I then also tested ICM with BIC and AIC as criteria. Additionally, I tested out ICM with a harsher penalty of $2 * \text{BIC's}$. I narrowed down to a few model selection methods by comparing the root mean squared error for train and validation from each model selection method. I noticed that backward selection took much longer than forward selection or forward/backward selection and ICM was the fastest. AIC was also generally slower than BIC.

I cross validated the final candidates with $K = 5$ folds. I compared the average of each model’s root mean square error (RMSE) to find the lowest in order to determine the best procedure/selection method. To get my final model, I applied this method on the full dataset. Using this final model, I predicted the response values of `protein-test.csv`.

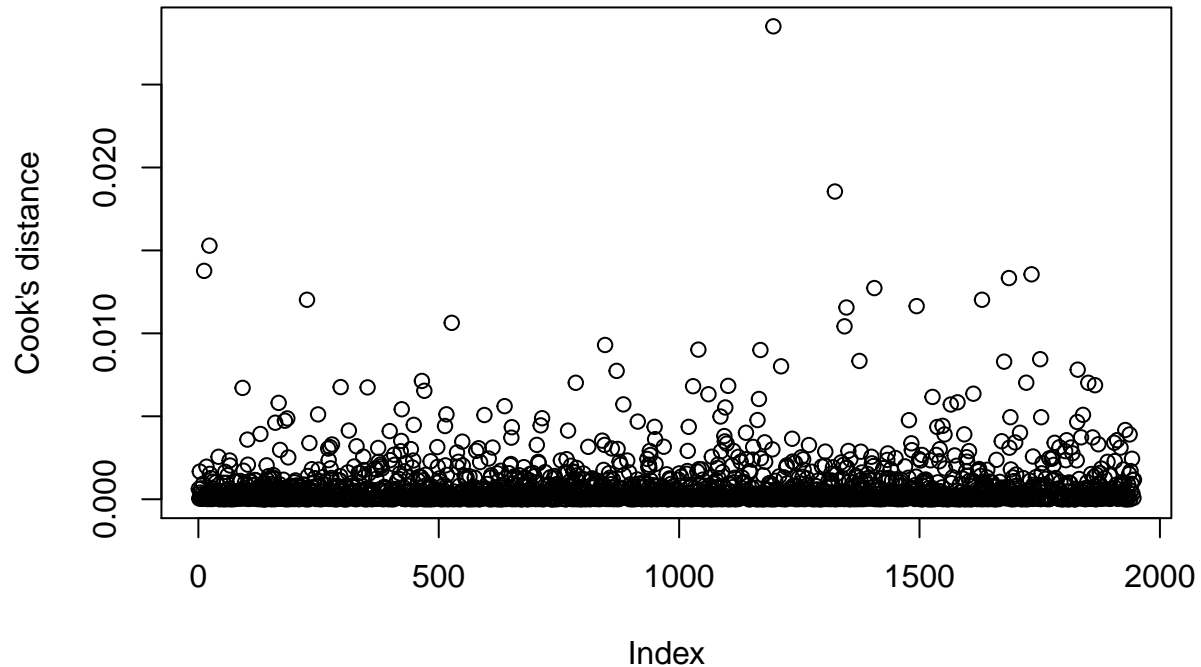
Results and discussion

Results of removing multicollinearity

After removing multicollinearity, I ended up with 551 predictors, so a total of 134 predictors were removed to get rid of multicollinearity.

Results of assessing the effects of individual observations

Using a Cook's distance cutoff point of 0.5, I observed 0 outliers in the data.



As seen in the plot, there are no points higher than $y = 0.5$.

Results of model selection

During my single train/validation split, I tested a total of 9 selection methods. Forward/backward/forward-backward, each with both AIC and BIC criteria, and ICM with AIC, BIC and a $2 \times \text{BIC}$ penalty. I recorded the train RMSE and validation RMSE for each model constructed with each of the 9 methods.

Selection method	Train RSME	Validation RMSE	AIC/BIC
Forward with AIC	0.4101569	0.6022125	2151.3292283
Backward with AIC	0.3987215	0.6339674	2109.2762061
Forward-backward with AIC	0.4115107	0.6168919	2123.5912774
Forward with BIC	0.5195751	0.6122045	2982.4444807
Backward with BIC	0.4872484	0.6298803	2988.2240144
Forward-backward BIC	0.5158826	0.6404971	2967.5852432
ICM with AIC	0.4030746	0.4512895	2103.0893615
ICM with BIC	0.5132434	0.5708434	2929.5618959
ICM with 2*BIC penalty	0.5943295	0.5708434	
Full model	0.380513	0.6227402	

Firstly, I see that both BIC and AIC values are lowest when I used ICM, with AIC/BIC when compared to the stepwise methods.

Since train RMSE can be made arbitrarily small by adding linearly independent predictors, we mainly care about the lowest validation RMSE, as long as they are approximately equal.

Since this is only a single train/validation split, I use this data to help me narrow down a few candidates for K fold cross validation.

I will note here that I tested out the ICM methods multiple times, and I got varying RMSE's for both train and validation. Sometimes they were better than the stepwise methods and sometimes they were worse, indicating a level of variability that I tested further with cross validation.

We can see that the validation RMSE is lower in forward with AIC(253 predictors) than the full model, which means the full model(551 predictors) is overfitting, so my exploratory analysis was correct in predicting we should get rid of some predictors. The stepwise methods with BIC as criteria yielded models with around 80 predictors, and had higher RMSE than forward with AIC, so I concluded they were underfitting.

With these results, I conclude that I should cross validate forward with AIC, forward with BIC, ICM with AIC and ICM with BIC to further evaluate selection strategies.

I decided on cross validation with 5 folds. If I had more time, I would have liked to use more folds.

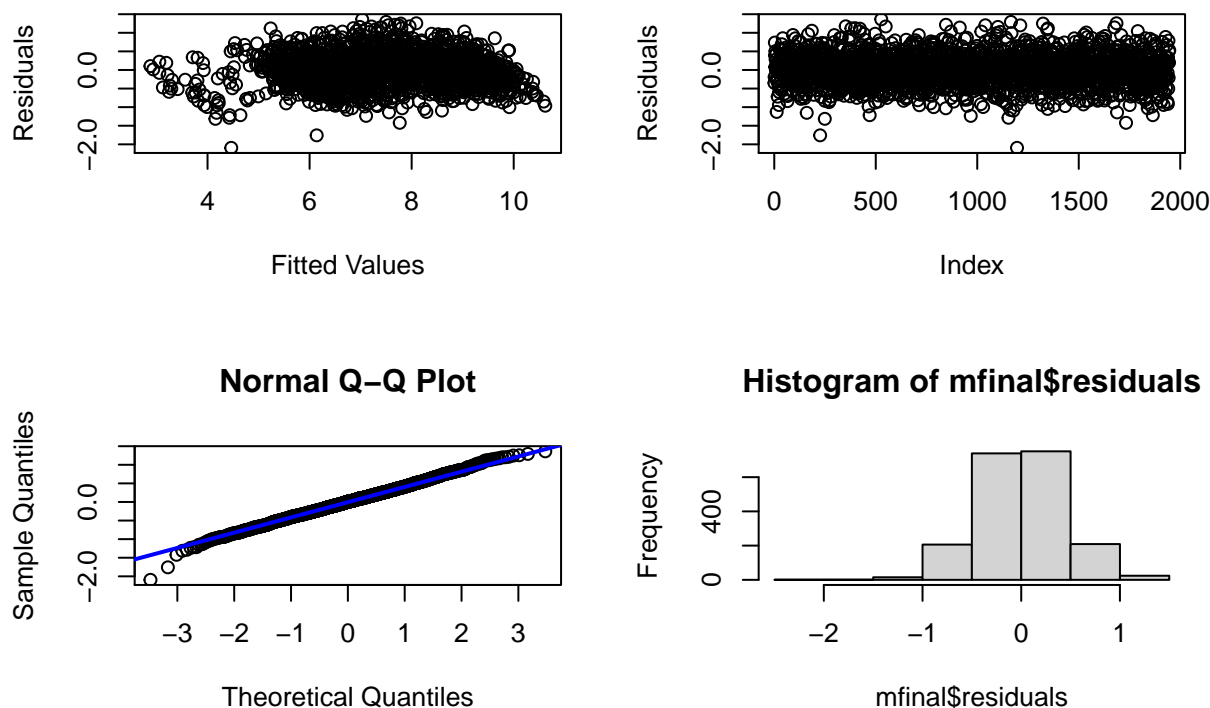
The results of cross validation are

Selection method	Average RMSE (RMSPE)
Forward AIC	0.6054771
ICM AIC	0.6093815
Forward BIC	0.6409394
ICM BIC	0.6251201

Thus I can conclude that the model constructed with forward with AIC has the lowest

RMSPE and so I can conclude that it is the best model selection method for prediction. I then apply this selection method to the entire data set to get my final model.

My final model selection method evaluated 112158 models before finishing. The R squared value for my final model is 0.9159083, and my final model has 267 predictors and 1 response. A full list of these predictors can be found in the appendix. The regression coefficient for the first 3 predictors are 0.033416, -0.097572, and -0.052900. They are all statistically significant since the corresponding p-values are much less than 0.005. ($< 1.51e-05$)



Overall my final model fits the data very well. All the regression assumptions are reasonably satisfied by my final model. In the residuals vs fitted values and residuals vs index plot, the points are randomly scattered around $y=0$. In the QQplot and histogram, normality seems to be satisfied.

I would predict the MSPE to be 0.3666026, if I applied the fitted model on new data. I am fairly confident my model should generalize well to doing prediction on new data since my model selection strategy yielded the lowest MSPE with cross validation, but since I only cross validated with 5 folds, I know I'm not as confident as I could be. If I had more time I would try 10 or even more folds.

Some interesting findings are that I noticed ICM with AIC sometimes yielded lower RMSE's than forward selection with AIC, but there was more variability, so the average RMSE was still lower. In addition, my final model did not include angles, a predictor that represents a score based on the configuration of angles in the computer generated structure of each

Covid-19 spike protein.

Appendix

```
# Detect and remove multicollinearity / commented since I saved after running
```

```
library("faraway")
```

```
## Warning: package 'faraway' was built under R version 4.0.3
```

```
ptrain <- read.csv ("protein-train.csv")
```

```
i <- 0
```

```
pred <- c()
```

```
# full model
```

```
pfull <- lm(accuracy ~ .,data = ptrain)
```

```
# vector of VIF's of predictors
```

```
# npred <- vif(pfull)
```

```
# vifpred <- unname(npred)
```

```
# copy of full model
```

```
ctrain <- ptrain[]
```

```
load("ptrain")
```

```
# # while the largest VIF >= 10
```

```
# while(max(vifpred) >= 10)
```

```
# {
```

```
#
```

```
# # find the index of the max VIF
```

```
# j <- which.max(vifpred) + 1
```

```
# ptrain <- ptrain[,-j]
```

```
# # vifpred only includes the predictors (685) while ptrain has the
```

```
# # response as well (686), so we add 1 to j
```

```
#
```

```
#
```

```
# premov <- lm(accuracy ~ .,data = ptrain) # fit new model without predictor
```

```
# # with largest VIF
```

```
#
```

```
# npred <- vif(premov)
```

```
# vifpred <- unname(npred) # find new vector of VIF's and loop until
```

```
# # every VIF is less than 10
```

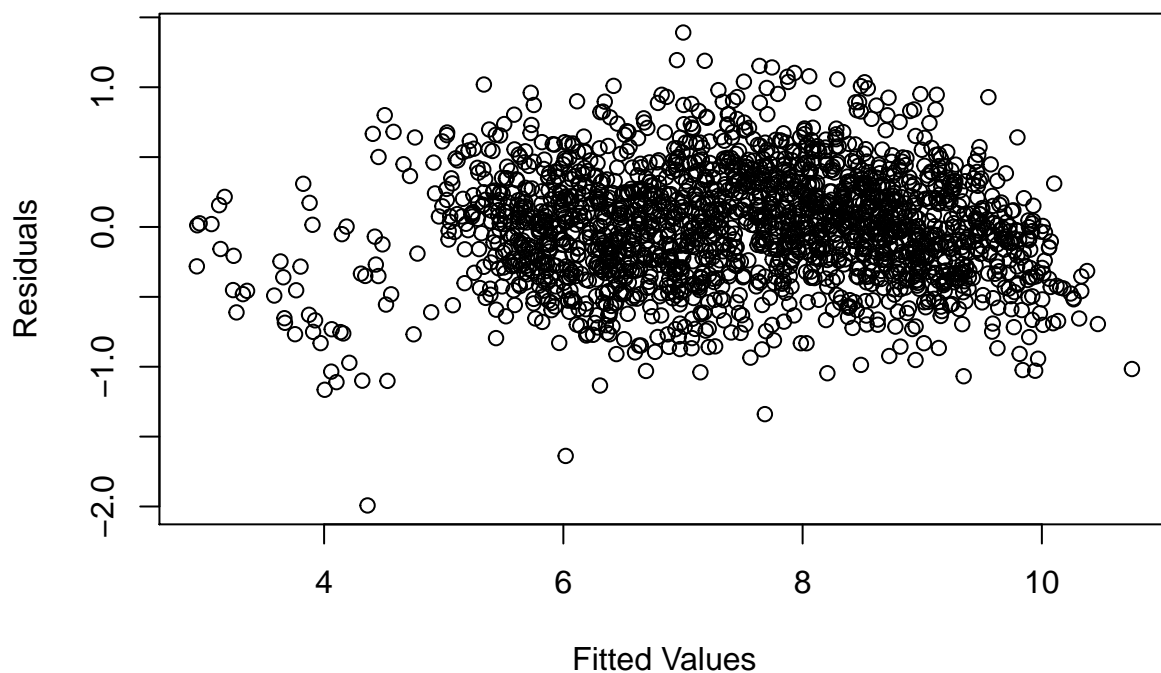
```
#
```



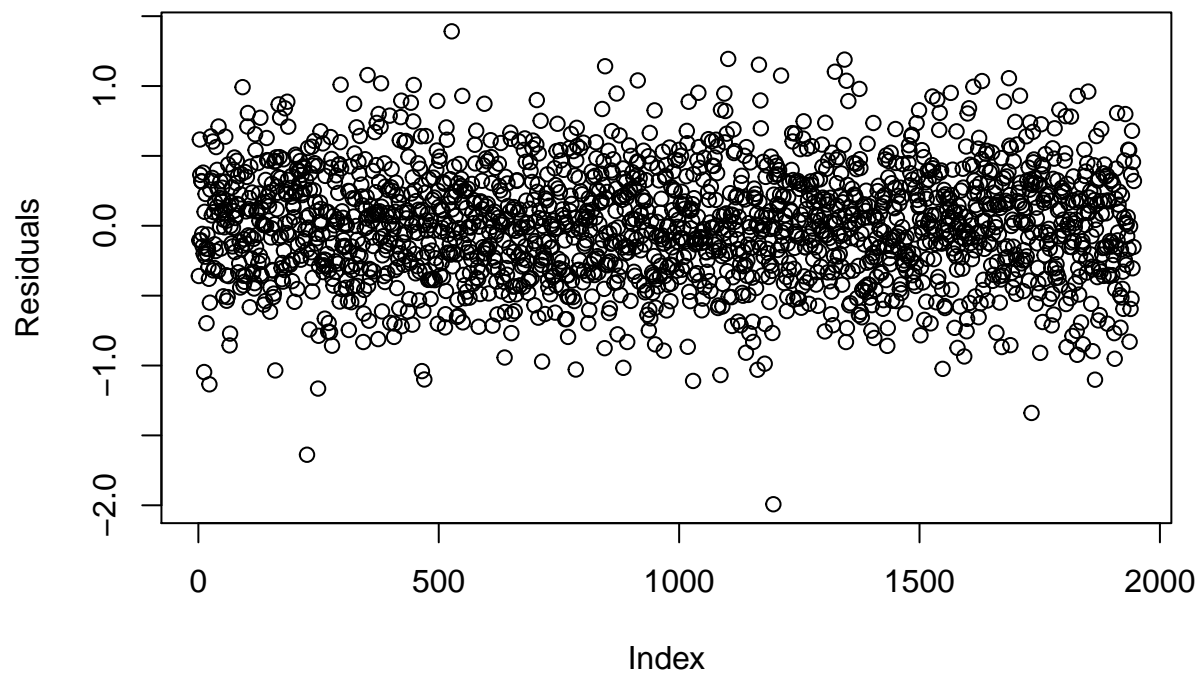
```
#
#   i <- i + 1 # count of how many predictors removed
#
# }
load("premov")
```

```
# after removing multicollinearity, we assess model linear regression
# assumptions
```

```
#summary(premov)
plot(premov$fitted.values, premov$residuals, xlab = "Fitted Values",
      ylab = "Residuals")
```

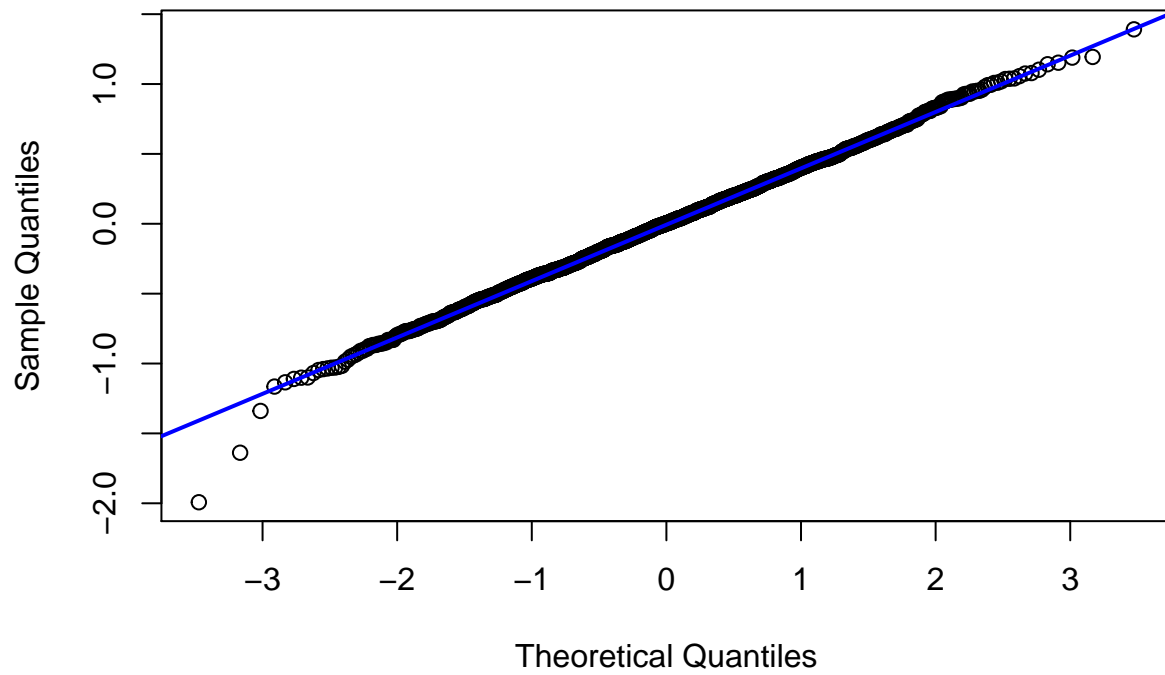


```
plot (1: nrow (ptrain), premov$residuals , xlab = " Index ", ylab =
      " Residuals ")
```



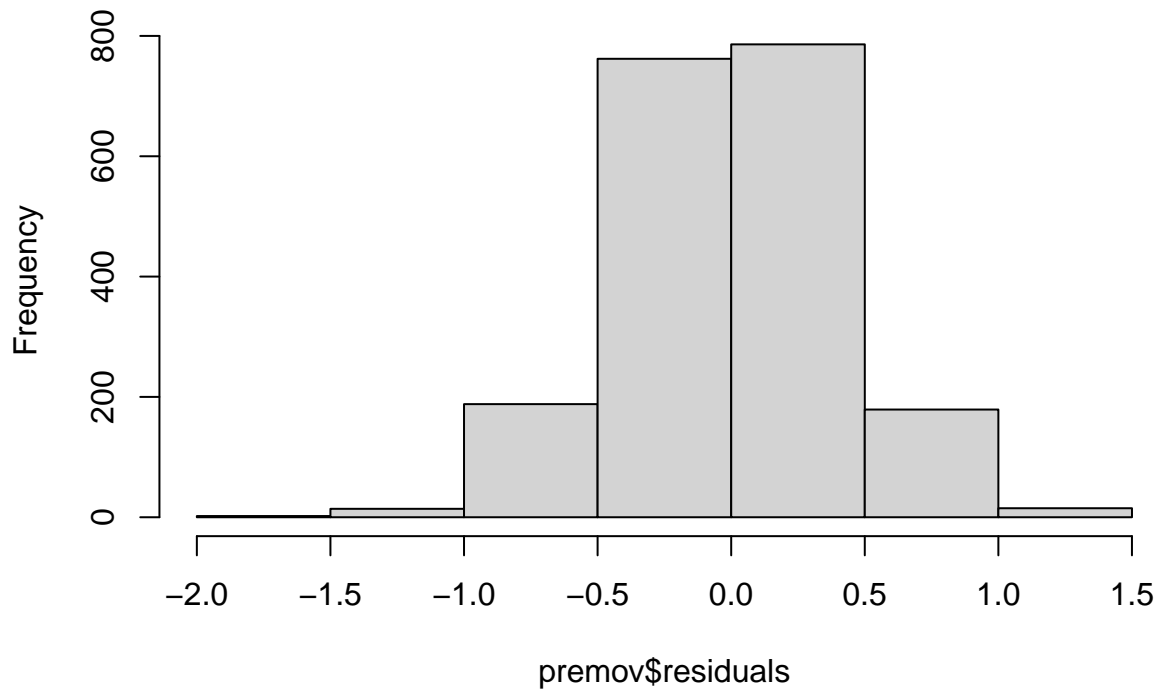
```
qqnorm(premov$residuals)
qqline(premov$residuals, col="blue", lwd = 2)
```

Normal Q-Q Plot



```
hist(premov$residuals)
```

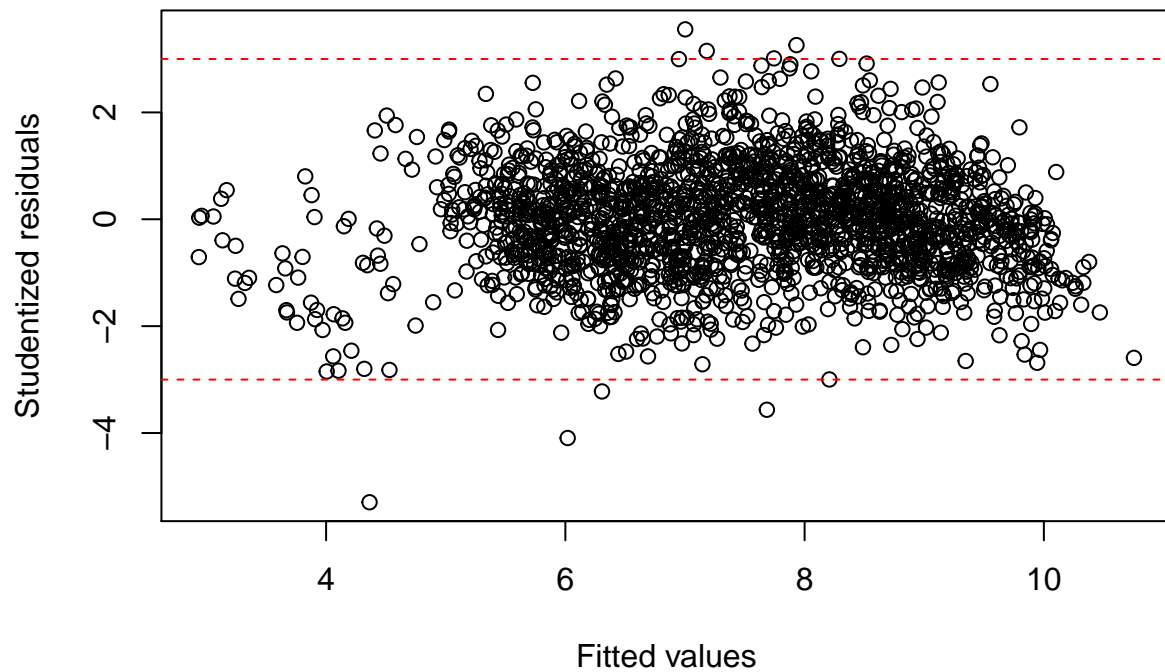
Histogram of premov\$residuals



```
library(MASS)

# quantities for individual observations
sres <- studres(premov) # studentized residuals
lev <- hatvalues(premov) # leverage
cd <- cooks.distance(premov) # Cook's distance

# plot of studentized residuals vs fitted values
plot(premov$fitted.values, sres, xlab = "Fitted values",
     ylab = "Studentized residuals")
# plots look the same
abline(h = c(3,-3), col = "red", lty=2) # lie within 3 standard deviations
```

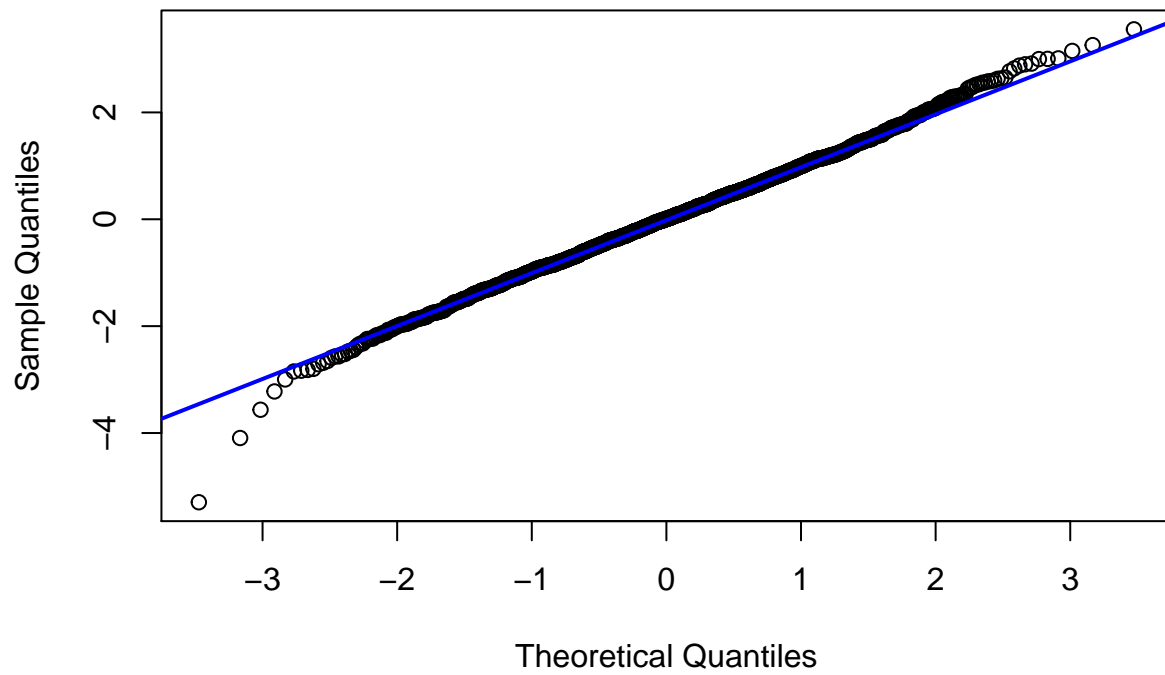


```
# of 0
which(abs(sres) > 3) # find the observations that don't lie within 3
```

```
## 23 226 527 846 1196 1324 1344 1686 1733
## 23 226 527 846 1196 1324 1344 1686 1733
```

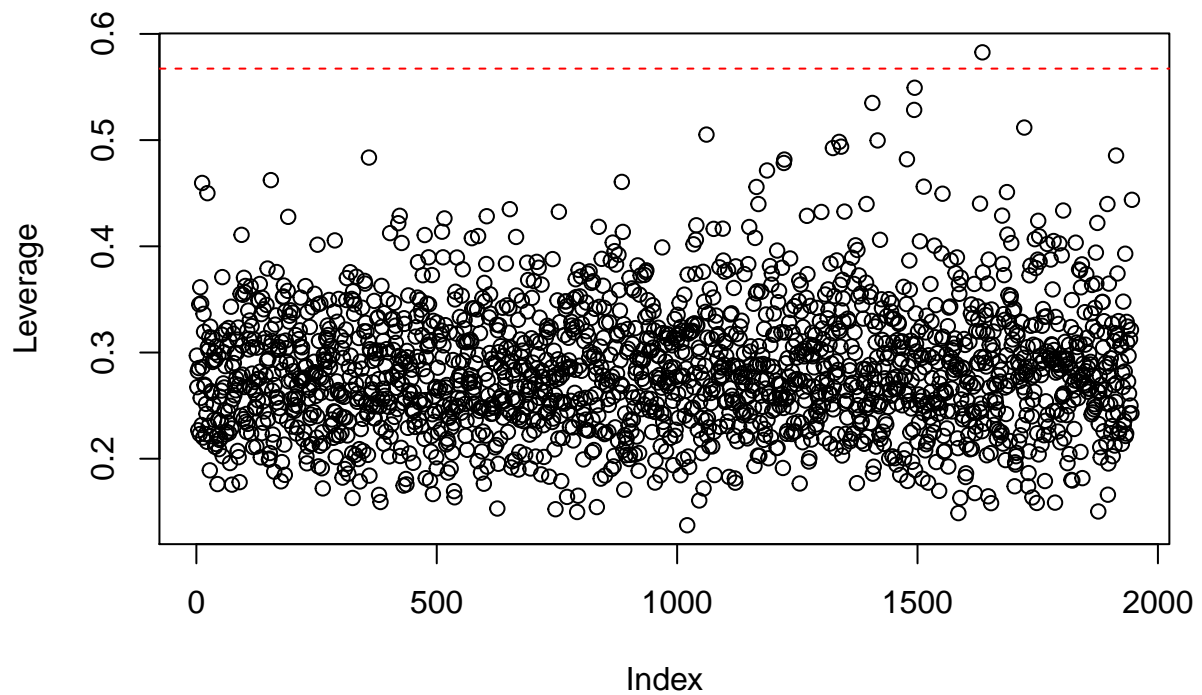
```
# standard deviations
# 23 226 527 846 1196 1324 1344 1686 1733
qqnorm(sres)
qqline(sres, col = "blue", lwd = 2)
```

Normal Q-Q Plot



```
#leverage
```

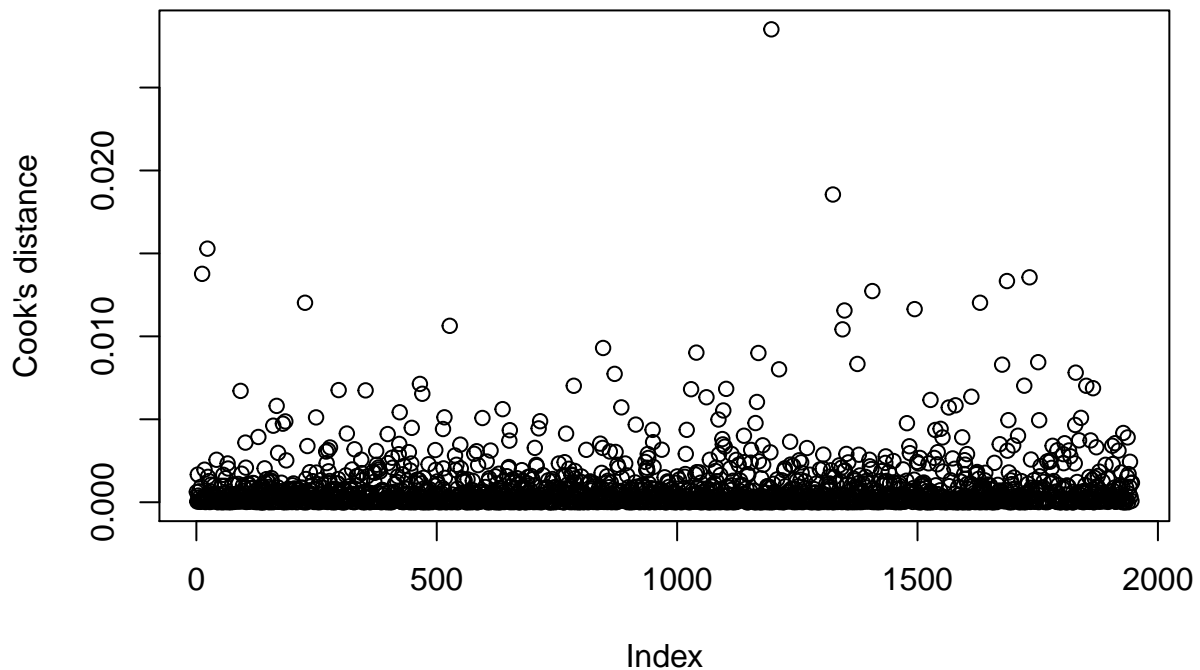
```
plot(lev, ylab = "Leverage")  
abline(h = 2 * mean(lev), col = "red", lty = 2) # leverages twice than average
```



```
# leverage are considered high  
which(lev > 2 * mean(lev)) # 1635
```

```
## 1635  
## 1635
```

```
# Cook's distance  
  
plot(cd, ylab = "Cook's distance")  
abline(h = 0.5, col = "red", lty = 2)
```



```
h <- unname(which(cd > 0.5))
# Cook's distance greater than 0.5 is generally considered large
# Thus, there are no outliers
# Other rules of thumb are  $4/n$ , and  $4/n-k-1$ 

otrain <- ptrain[-h,]
new <- lm(accuracy ~ ., data = ptrain)
# summary(new)

# Check model regression assumptions after outliers removed

# plot(new$fitted.values, new$residuals, xlab = "Fitted Values",
#       ylab = "Residuals")
# qqnorm(new$residuals)
# qqline(new$residuals, col="blue", lwd = 2)
# hist(new$residuals)

# First we test out different model selection methods with a single
# train/validation split.

library(MASS)
```



```

N <- nrow(ptrain)
set.seed(20688307)

trainInd <- sample(1: N ,round(N*0.8) ,replace = F)
# 80/20 train/validation split
trainSet <- ptrain[trainInd ,]
validSet <- ptrain[-trainInd ,]

full <- lm(accuracy ~ ., data = trainSet)
empty <- lm(accuracy ~ 1, data = trainSet)

# stepwise forward with aic

load("m_aicf")

#m_aicf <- stepAIC(object = empty, scope = list(upper = full, lower = empty), direction

# stepwise backward with aic

load("m_aicb")

#m_aicb <- stepAIC(object = full, scope = list(upper = full, lower = empty), direction

# stepwise forward/backward with aic

load("m_aich")

#m_aich <- stepAIC(object = empty, scope = list(upper = full, lower = empty), direction
#= "both")

# stepwise forward with bic

load("m_bicf")

#m_bicf <- stepAIC(object = empty, scope = list(upper = full, lower = empty), direction

```

```

# stepwise backward with bic

load("m_bicb")

#m_bicb <- stepAIC(object = full, scope = list(upper = full, lower = empty), direction

#stepwise forward-backward with bic

load("m_bich")

#m_bich <- stepAIC(object = empty, scope = list(upper = full, lower = empty), direction

# We calculate the RMSE on train and validation for each model

# Full model (after removing multicollinearity)

predfull <- predict(full, newdata = validSet)
fullv <- sqrt(mean((validSet$accuracy - predfull)^2)) #RMSE for validation
fullt <- sqrt(mean(full$residuals^2)) # RMSE for train

# AIC-forward selection

predaf <- predict(m_aicf, newdata = validSet)
aicfv <- sqrt(mean((validSet$accuracy - predaf)^2))
aicft <- sqrt(mean(m_aicf$residuals ^2))

faic <- AIC(m_aicf)

# AIC-backward selection

predab <- predict(m_aicb, newdata = validSet)
aicbv <- sqrt(mean((validSet$accuracy - predab)^2))
aicbt <- sqrt(mean(m_aicb$residuals^2))

baic <- AIC(m_aicb)

# AIC-forward/backward selection

predah <- predict(m_aich, newdata = validSet)

```

```

aichv <- sqrt(mean((validSet$accuracy - predah )^2))
aicht <- sqrt(mean(m_aich$residuals ^2))

haic <- AIC(m_aich)

# BIC-forward selection

predbf <- predict (m_bicf , newdata = validSet)
bicfv <- sqrt(mean((validSet$accuracy - predbf )^2))
bicft <- sqrt(mean(m_bicf$residuals^2))

fbic <- BIC(m_bicf)

# BIC-backward selection

predbb <- predict(m_bicb, newdata = validSet)
bicbv <- sqrt(mean((validSet$accuracy - predbb)^2))
bicbt <- sqrt(mean(m_bicb$residuals^2))

bbic <- BIC(m_bicb)

# BIC-forward/backward selection

predbh <- predict(m_bich, newdata = validSet)
bichv <- sqrt(mean((validSet$accuracy - predbh)^2))
bicht <- sqrt(mean(m_bich$residuals ^2))

hbic <- BIC(m_bich)

#ICM with no penalty (AIC)
library(MASS)

N <- nrow(ptrain)

load("m_bestA")

#set.seed(20688307)

# trainInd <- sample(1: N ,round(N*0.8) ,replace = F)
# # 80/20 train/validation split
# trainSet <- ptrain[trainInd ,]
# validSet <- ptrain[-trainInd ,]
#

```

```

# full <- lm(accuracy ~ ., data = trainSet)
# empty <- lm(accuracy ~ 1, data = trainSet)
#
# varlist = c()
# varnames = names(trainSet)
# n = nrow(trainSet)
# varoder <- sample(1:ncol(trainSet)) #random order of variables
# minCrit = Inf
# noChange = F
#
# while(!noChange) {
#   noChange = T
#   for (i in varoder) {
#     if (i == 1)
#       next
#
#     if (i %in% varlist & length(varlist) > 1) {
#       index = c(1, varlist[varlist != i])
#       trainVars = trainSet[, index]
#
#       fit = lm(accuracy ~ ., data = trainVars)
#
#       if (AIC(fit) < minCrit) {
#         minCrit = AIC (fit)
#         varlist = varlist[varlist != i]
#         #print(paste0(" Criterion : ", round(minCrit, 1) , ", variables : ",
#         #           paste0 (varnames[varlist], collapse = " "))
#         bestA_model = fit
#         noChange = F
#       }
#
#     } else if (!i %in% varlist) {
#       index = c(1, varlist, i)
#       trainVars = trainSet[, index]
#
#       fit = lm(accuracy ~ ., data = trainVars)
#
#       if (AIC(fit) < minCrit) {
#         minCrit = AIC(fit)
#         varlist = c(varlist, i)
#         # print(paste0(" Criterion : ", round(minCrit, 1), ", variables : ",
#         #           # paste0(varnames[varlist], collapse = " "))
#
#         bestA_model = fit

```

```

#           noChange = F
#       }
#   }
# }

# ICM with BIC penalty
library(MASS)

N <- nrow(ptrain)
#set.seed(20688307) different AIC each time without seed

load("m_bestB")

# trainInd <- sample(1: N ,round(N*0.8) ,replace = F)
# # 80/20 train/validation split
# trainSet <- ptrain[trainInd ,]
# validSet <- ptrain[-trainInd ,]
#
# full <- lm(accuracy ~ ., data = trainSet)
# empty <- lm(accuracy ~ 1, data = trainSet)
#
# pen <- log (nrow (trainSet))
# varlist = c()
# varnames = names(trainSet)
# n = nrow(trainSet)
# varoder <- sample(1:ncol(trainSet)) #random order of variables
# minCrit = Inf
# noChange = F
#
# while(!noChange) {
#   noChange = T
#   for (i in varoder) {
#     if (i == 1)
#       next
#
#     if (i %in% varlist & length(varlist) > 1) {
#       index = c(1, varlist[varlist != i])
#       trainVars = trainSet[, index]
#
#       fit = lm(accuracy ~ ., data = trainVars)
#
#       if (AIC(fit, k=pen) < minCrit) {
#         minCrit = AIC (fit, k = pen)

```

```

#         varlist = varlist[varlist!= i]
#         # print(paste0(" Criterion : ", round(minCrit, 1) , ", variables : ",
#         #         paste0 (varnames[varlist], collapse = " "))
#         bestB_model = fit
#         noChange = F
#     }
#
# } else if (!i %in% varlist) {
#     index = c(1, varlist, i)
#     trainVars = trainSet[, index]
#
#     fit = lm(accuracy ~ ., data = trainVars)
#
#     if (AIC(fit, k = pen) < minCrit) {
#         minCrit = AIC(fit, k = pen)
#         varlist = c(varlist, i)
#         # print(paste0(" Criterion : ", round(minCrit, 1), ", variables : ",
#         #         #         paste0(varnames[varlist], collapse = " "))
#
#         bestB_model = fit
#         noChange = F
#     }
# }
# }

```

```

# ICM with 2*BIC penalty
library(MASS)

N <- nrow(ptrain)
#set.seed(20688307) different AIC each time without seed

load("m_bestC")

# trainInd <- sample(1: N ,round(N*0.8) ,replace = F)
# # 80/20 train/validation split
# trainSet <- ptrain[trainInd ,]
# validSet <- ptrain[-trainInd ,]
#
# full <- lm(accuracy ~ ., data = trainSet)
# empty <- lm(accuracy ~ 1, data = trainSet)
#
# pen <- 2 * log (nrow (trainSet))
# varlist = c()

```

```

# varnames = names(trainSet)
# n = nrow(trainSet)
# varoder <- sample(1:ncol(trainSet)) #random order of variables
# minCrit = Inf
# noChange = F
#
# while(!noChange) {
#   noChange = T
#   for (i in varoder) {
#     if (i == 1)
#       next
#
#     if (i %in% varlist & length(varlist) > 1) {
#       index = c(1, varlist[varlist != i])
#       trainVars = trainSet[, index]
#
#       fit = lm(accuracy ~ ., data = trainVars)
#
#       if (AIC(fit, k=pen) < minCrit) {
#         minCrit = AIC (fit, k = pen)
#         varlist = varlist[varlist!= i]
#         # print(paste0(" Criterion : ", round(minCrit, 1) , ", variables : ",
#         #           paste0 (varnames[varlist], collapse = " ")))
#         bestC_model = fit
#         noChange = F
#       }
#
#     } else if (!i %in% varlist) {
#       index = c(1, varlist, i)
#       trainVars = trainSet[, index]
#
#       fit = lm(accuracy ~ ., data = trainVars)
#
#       if (AIC(fit, k = pen) < minCrit) {
#         minCrit = AIC(fit, k = pen)
#         varlist = c(varlist, i)
#         # print(paste0(" Criterion : ", round(minCrit, 1), ", variables : ",
#         #           paste0(varnames[varlist], collapse = " ")))
#
#         bestC_model = fit
#         noChange = F
#       }
#     }
#   }
# }

```

```

# }

# ICM RMSE - we get varying RMSE's since different orderings for ICM
# Sometimes we get a higher RMSE, sometimes lower. So we can cross validate to
# check further.

# ICM - no penalty

bestA <- predict(bestA_model, newdata = validSet)
bestAv <- sqrt(mean((validSet$accuracy - bestA)^2))
bestAt <- sqrt(mean(bestA_model$residuals ^2))

bestaic <- AIC(bestA_model)

# ICM -BIC penalty

bestB <- predict(bestB_model, newdata = validSet)
bestBv <- sqrt(mean((validSet$accuracy - bestB)^2))
bestBt <- sqrt(mean(bestB_model$residuals ^2))

bestbic <- BIC(bestB_model)

# ICM -2 * BIC penalty

bestC <- predict(bestC_model, newdata = validSet)
bestCv <- sqrt(mean((validSet$accuracy - bestC)^2))
bestCt <- sqrt(mean(bestC_model$residuals ^2))

load("RMSE1")
load("RMSE2")
load("RMSE3")
load("RMSE4")

RMSPE1 <- mean(RMSE1)
RMSPE2 <- mean(RMSE2)
RMSPE3 <- mean(RMSE3)
RMSPE4 <- mean(RMSE4)

# Cross Validation to determine the best selection method
# library (MASS)
# N <- nrow(ptrain)
# set.seed(20688307)

```



```

#
# K<-5
#
# validSetSplits <- sample((1:N)%K + 1)
# RMSE1 <- c()
# RMSE2 <- c()
# RMSE3 <- c()
# RMSE4 <- c()
#
# for (p in 1:K) {
#   validSet <- ptrain[validSetSplits == p ,]
#   trainSet <- ptrain[validSetSplits != p ,]
#
#   full <- lm (accuracy ~ . , data = trainSet)
#   empty <- lm (accuracy ~ 1 , data = trainSet)
#
#
# m1 <- stepAIC(object = empty, scope = list(upper = full, lower = empty), direction =
#
# pred1 <- predict( m1, newdata = validSet)
# RMSE1[p] <- sqrt(mean((validSet$accuracy - pred1)^2))
#
#
#
# varlist = c()
# varnames = names(trainSet)
# n = nrow(trainSet)
# varorder <- sample(1:ncol(trainSet)) #random order of variables
# minCrit = Inf
# noChange = F
#
# while(!noChange) {
#   noChange = T
#   for (i in varorder) {
#     if (i == 1)
#       next
#
#     if (i %in% varlist & length(varlist) > 1) {
#       index = c(1, varlist[varlist != i])
#       trainVars = trainSet[, index]
#
#       fit = lm(accuracy ~ ., data = trainVars)
#
#       if (AIC(fit) < minCrit) {

```

```

#         minCrit = AIC (fit)
#         varlist = varlist[varlist != i]
#         #print(paste0(" Criterion : ", round(minCrit, 1) , ", variables : ",
#         #paste0 (varnames[varlist], collapse = " "))
#         m2 = fit
#         noChange = F
#     }
#
# } else if (!i %in% varlist) {
#     index = c(1, varlist, i)
#     trainVars = trainSet[, index]
#
#     fit = lm(accuracy ~ ., data = trainVars)
#
#     if (AIC(fit) < minCrit) {
#         minCrit = AIC(fit)
#         varlist = c(varlist, i)
#         # print(paste0(" Criterion : ", round(minCrit, 1), ", variables : ",
#         #         paste0(varnames[varlist], collapse = " "))
#
#         m2 = fit
#         noChange = F
#     }
# }
# }
#
# pred2 <- predict(m2 ,newdata = validSet)
# RMSE2[p] <- sqrt(mean((validSet$accuracy - pred2)^2))
#
#
#
#
# m3 <- stepAIC(object = empty, scope = list(upper = full, lower = empty), direction
# pred3 <- predict(m3 , newdata = validSet )
# RMSE3[p] <- sqrt(mean(( validSet$accuracy - pred3)^2))
#
#
#
#
# pen <- log (nrow (trainSet))
# varlist = c()
# varnames = names(trainSet)
# n = nrow(trainSet)

```

```

# varoder <- sample(1:ncol(trainSet)) #random order of variables
# minCrit = Inf
# noChange = F
#
# while(!noChange) {
#   noChange = T
#   for (z in varoder) {
#     if (z == 1)
#       next
#
#     if (z %in% varlist & length(varlist) > 1) {
#       index = c(1, varlist[varlist != z])
#       trainVars = trainSet[, index]
#
#       fit = lm(accuracy ~ ., data = trainVars)
#
#       if (AIC(fit, k=pen) < minCrit) {
#         minCrit = AIC (fit, k = pen)
#         varlist = varlist[varlist!= z]
#         #print(paste0(" Criterion : ", round(minCrit, 1) , ", variables : ",
#           #paste0 (varnames[varlist], collapse = " "))
#
#         m4 = fit
#         noChange = F
#       }
#
#     } else if (!z %in% varlist) {
#       index = c(1, varlist, z)
#       trainVars = trainSet[, index]
#
#       fit = lm(accuracy ~ ., data = trainVars)
#
#       if (AIC(fit, k = pen) < minCrit) {
#         minCrit = AIC(fit, k = pen)
#         varlist = c(varlist, z)
#         # print(paste0(" Criterion : ", round(minCrit, 1), ", variables : ",
#           #paste0(varnames[varlist], collapse = " "))
#
#         m4 = fit
#         noChange = F
#       }
#     }
#   }
# }
# }
#

```

```
# pred4 <- predict(m4, newdata = validSet)
# RMSE4[p] <- sqrt(mean((validSet$accuracy - pred4)^2))
#
#
# }
#
#
```

```
# We apply selected procedure to the full dataset
```

```
load("mfinal")
```

```
# full <- lm(accuracy ~ ., data = ptrain)
# empty <- lm(accuracy ~ 1, data = ptrain)
#
# mfinal <- stepAIC(object = empty, scope = list(upper = full, lower = empty), directi
```

```
mfinal$coefficients
```

```
##          (Intercept)          aliph1HC_aliph2HC_long
##          3.085736324          0.033415822
##          scLysN_bbC_vlong          aliph2HC_bbN_medshort
##          -0.097572193          -0.052900348
## aliph1HC_aromaticC_medshort aromaticC_hydroxyl0_medlong
##          0.110722520          -0.039630202
##          carbonylC_aromaticC_short          aliph1HC_aromaticC_vlong
##          0.110159934          0.080347635
##          bbC_bb0_short          aliph1HC_aliph1HC_vlong
##          -0.027907824          0.149503560
##          carboxylC_scLysN_vlong          sulfur_bbC_vlong
##          0.041247792          -0.017002155
##          aliph1HC_aromaticC_long          aromaticC_hydroxyl0_long
##          0.082598536          -0.030726393
##          aliph2HC_scLysN_vlong          aliph3HC_bbC_vlong
##          0.037928555          0.010450667
##          aliph1HC_bbN_vlong          bbN_bb0_long
##          -0.013684296          -0.011956478
##          aliph2HC_bbN_vlong          sulfur_bbC_medlong
##          0.001059020          -0.063075774
##          carboxyl0_bbC_vlong          aromaticC_sulfur_long
##          0.024634615          0.037697347
##          aliph1HC_aromaticC_medlong          aromaticC_scAGN_short
##          0.070662023          0.012264308
##          aliph1HC_bb0_long          aliph1HC_bbC_medshort
```

##	-0.080248036	0.076676022
##	aliph2HC_aromaticC_vlong	scLysN_bbN_vlong
##	0.012469819	-0.074743867
##	bbN_bbC_medshort	bbO_bbO_short
##	-0.023752349	-0.022324954
##	aliph1HC_sulfur_short	aliph3HC_bbN_short
##	0.198692224	0.100834275
##	aliph1HC_bbO_medlong	aliph2HC_carboxylO_vlong
##	-0.126807275	-0.009171958
##	aliph3HC_aliph3HC_short	scAGN_bbN_long
##	0.038660284	0.034621603
##	aliph3HC_aromaticC_long	aliph2HC_scArgN_vlong
##	-0.016014025	0.113686020
##	aliph2HC_aliph3HC_short	aliph1HC_aliph3HC_short
##	0.035877308	0.145722316
##	scArgN_bbO_medlong	aliph1HC_bbCA_vlong
##	0.465020986	0.006617271
##	carboxylC_bbN_long	carboxylO_carboxylO_vlong
##	0.074742456	-0.010173256
##	scAGN_bbN_medlong	aliph1HC_bbProN_medlong
##	0.014973172	0.283258729
##	bbCA_bbO_vshort	aliph1HC_scArgN_long
##	-0.362888385	0.362998718
##	sulfur_bbCA_short	scLysN_carboxylO_long
##	-0.126617592	0.132046893
##	carbonylC_hydroxylO_vlong	aliph2HC_aliph3HC_vlong
##	-0.055209272	0.013393611
##	aliph2HC_hydroxylO_long	carbonylC_bbProN_medlong
##	0.011610725	-0.154285898
##	bbN_bbN_medshort	aliph3HC_scArgN_vlong
##	-0.051056941	0.268086387
##	aliph1HC_bbProN_vlong	aromaticC_bbO_vlong
##	0.048928470	0.013720180
##	aliph2HC_aliph2HC_short	scLysN_bbN_medlong
##	0.044249959	-0.026692625
##	aliph1HC_aliph3HC_medlong	aliph1HC_bbN_medshort
##	-0.044883584	-0.099658805
##	aliph1HC_aliph1HC_long	carboxylC_aromaticC_long
##	0.139026676	0.009206965
##	carbonylC_bbProN_long	sulfur_bbO_vlong
##	-0.097289640	-0.031887005
##	bbN_bbC_medlong	aliph2HC_aromaticC_long
##	-0.016180792	0.004385553
##	scAGN_carboxylO_medshort	bbProN_carboxylO_vlong
##	-0.080649720	0.091184664

##	bb0_bb0_vshort	bbProN_bb0_long
##	0.146228095	0.035606456
##	bbCA_bb0_medshort	aliph1HC_hydroxyl0_vlong
##	-0.028972857	0.077876330
##	aliph3HC_aromaticC_vlong	carboxylC_carboxylC_vlong
##	-0.018148494	-0.230068159
##	carboxylC_hydroxyl0_short	carboxyl0_bbCA_medlong
##	-0.247938453	0.059548280
##	aliph2HC_hydroxyl0_medlong	aliph3HC_aromaticC_medlong
##	0.010365249	-0.014508451
##	aromaticC_bb0_medlong	aromaticC_bb0_vshort
##	0.016919610	-0.189646813
##	aromaticC_bbC_short	aromaticC_aromaticC_vlong
##	0.098443505	-0.008677942
##	aromaticC_bbProN_vlong	aliph3HC_hydroxyl0_short
##	-0.043302601	-0.080619545
##	bbC_bbC_long	aliph1HC_bb0_medshort
##	-0.001633968	-0.083598662
##	aliph3HC_hydroxyl0_long	carbonylC_bbProN_vlong
##	0.002522520	-0.084920281
##	carbonylC_aliph3HC_medlong	carbonylC_sulfur_vlong
##	0.016342590	-0.051448147
##	aromaticC_scLysN_vlong	carboxyl0_bbCA_long
##	-0.059700876	0.037532441
##	aromaticC_carbonyl0_medlong	carbonylC_sulfur_short
##	0.015775515	-0.160344227
##	aliph2HC_scLysN_medshort	scAGN_hydroxyl0_long
##	-0.112110634	-0.044511253
##	hydroxyl0_sulfur_short	hydroxyl0_sulfur_medshort
##	0.169724232	0.141965152
##	hydroxyl0_sulfur_long	hydroxyl0_sulfur_vlong
##	0.055146213	0.037033233
##	sulfur_bbCA_medshort	carbonylC_scLysN_vlong
##	-0.033038837	0.017447277
##	carbonylC_bbN_vlong	aliph3HC_hydroxyl0_medlong
##	-0.026666517	0.001390584
##	aliph2HC_aliph3HC_vshort	carbonylC_bbC_medlong
##	0.093825630	-0.031946936
##	scAGN_bb0_medlong	aliph1HC_bb0_short
##	0.034180334	-0.104519186
##	aliph2HC_scLysN_long	sulfur_bb0_long
##	0.012691939	-0.028302323
##	bbProN_bbCA_medshort	scAGN_bb0_short
##	0.053310119	0.077763462
##	aromaticC_sulfur_medlong	aromaticC_hydroxyl0_vshort

##	0.036376737	-0.139132948
##	aliph1HC_hydroxyl0_long	aliph3HC_bbN_medlong
##	0.094946454	0.010331282
##	aliph3HC_bbN_long	bbN_bb0_medlong
##	0.009687819	-0.006754636
##	bbProN_bbN_long	hydroxyl0_bbC_medlong
##	0.059957710	0.019037832
##	aromaticC_bbProN_medlong	bbN_bb0_medshort
##	-0.062357352	-0.015189454
##	carbonylC_hydroxyl0_medshort	aliph2HC_aromaticC_medshort
##	-0.030684302	0.020226009
##	aliph1HC_hydroxyl0_medshort	aliph1HC_hydroxyl0_medlong
##	0.133336205	0.143327723
##	aliph3HC_hydroxyl0_medshort	carboxyl0_bbC_medshort
##	-0.052614362	-0.003362942
##	aliph2HC_bbC_vlong	aliph3HC_bb0_medshort
##	0.005501942	0.024198115
##	aliph2HC_bb0_vshort	aliph1HC_aliph3HC_vlong
##	0.078716145	-0.045279213
##	hydroxyl0_sulfur_medlong	sulfur_bb0_short
##	0.057980490	-0.045000808
##	aliph2HC_bbProN_medlong	scLysN_bbCA_long
##	-0.036046483	-0.061679233
##	carboxylC_aliph2HC_long	scLysN_hydroxyl0_long
##	-0.033814766	0.034865570
##	scLysN_bbC_medlong	aliph3HC_aliph3HC_vlong
##	-0.038809919	0.042269473
##	aliph3HC_bbProN_medlong	aromaticC_scAGN_long
##	-0.046140394	0.016416593
##	aliph1HC_bbN_medlong	bbProN_bbN_medlong
##	-0.033305850	0.049402040
##	scAGN_carboxyl0_vshort	scAGN_carboxyl0_vshort
##	-0.103702815	-0.181275377
##	bbN_bbC_vshort	aliph2HC_aliph3HC_medshort
##	-0.034485953	0.013959357
##	scAGN_bb0_long	carbonylC_aliph3HC_short
##	0.020451777	-0.054674429
##	aliph1HC_carboxyl0_medshort	aliph3HC_scArgN_long
##	0.049188422	0.147521127
##	scArgN_bb0_long	aliph3HC_scAGN_medshort
##	-0.167596507	0.037133114
##	scAGN_bbC_vlong	aliph3HC_bbCA_short
##	-0.011720693	0.032853035
##	aromaticC_sulfur_vlong	carbonylC_bb0_short
##	0.006866154	-0.053511772

##	aliph1HC_bbCA_medshort	sulfur_sulfur_vlong
##	0.089850472	0.101705497
##	carbonylC_scLysN_long	carbonylC_carbonyl0_vlong
##	-0.063068273	0.053671218
##	aromaticC_bbN_vlong	aromaticC_hydroxyl0_medshort
##	0.005685735	-0.034600715
##	hydroxyl0_carboxyl0_long	hydroxyl0_carboxyl0_vshort
##	-0.040825251	0.128573027
##	aliph3HC_sulfur_medshort	aliph1HC_sulfur_medshort
##	0.091243632	-0.065549264
##	aliph3HC_sulfur_medlong	aliph3HC_carboxyl0_vlong
##	0.070642595	-0.039253563
##	carbonylC_carbonyl0_short	aromaticC_sulfur_short
##	0.125956082	-0.049771200
##	aliph3HC_sulfur_vlong	carbonyl0_bbN_medlong
##	0.035178195	-0.022447784
##	hydroxyl0_bbN_medlong	carboxyl0_sulfur_medlong
##	0.018727370	0.064175413
##	carboxyl0_sulfur_vlong	aliph2HC_bbC_medshort
##	0.049402755	0.015149426
##	bbCA_bb0_short	bbProN_sulfur_vlong
##	-0.037120901	-0.077863378
##	carboxyl0_sulfur_medshort	aliph3HC_sulfur_long
##	0.055188545	0.035887993
##	carbonylC_sulfur_medlong	scLysN_carbonyl0_vlong
##	-0.080799206	-0.045165791
##	aliph1HC_scAGN_short	carboxylC_aliph2HC_medlong
##	-0.035500366	-0.023592740
##	carboxylC_bbN_vlong	bbProN_bbN_medshort
##	0.033677823	0.036127352
##	carbonyl0_carboxyl0_vlong	aliph3HC_carbonyl0_medshort
##	-0.050357059	0.057780707
##	scAGN_bb0_medshort	scAGN_carbonyl0_short
##	0.033740243	-0.092842141
##	carbonylC_carbonyl0_medshort	aliph2HC_carbonyl0_long
##	0.070085778	0.012806071
##	aliph3HC_bbProN_vlong	scArgN_hydroxyl0_short
##	0.034847864	-0.104232851
##	bbProN_bb0_vlong	aromaticC_scAGN_medshort
##	0.028829040	-0.020626043
##	aliph1HC_bbN_short	carbonylC_aliph2HC_short
##	-0.066846762	0.015064798
##	hydroxyl0_carboxyl0_short	carbonylC_sulfur_long
##	0.052280040	-0.067967872
##	aliph2HC_carbonyl0_vlong	carbonylC_carbonylC_vlong

##	0.013665337	-0.053412695
##	scAGN_bbProN_medshort	scAGN_bbO_vshort
##	-0.110165410	0.058690082
##	carboxylC_bbC_vlong	aromaticC_scLysN_medlong
##	-0.023729055	-0.074723138
##	carboxylO_bbC_medlong	aliph3HC_sulfur_short
##	-0.046276599	0.059825467
##	aliph1HC_aliph2HC_short	aromaticC_bbCA_vlong
##	-0.099632709	0.006726620
##	aliph1HC_aliph2HC_medshort	aromaticC_bbC_vlong
##	-0.032012144	-0.007449817
##	aliph2HC_aliph3HC_long	bbCA_bbO_long
##	-0.013490532	-0.005993055
##	aliph2HC_sulfur_medshort	scLysN_bbO_medlong
##	-0.021344664	0.051987113
##	aliph2HC_carboxylO_vshort	carboxylC_aliph2HC_vlong
##	0.065010336	0.029854412
##	scArgN_carboxylO_long	aromaticC_bbN_long
##	-0.062484734	0.005016098
##	carboxylC_aliph3HC_vlong	carbonylO_bbO_short
##	-0.029941360	0.026266941
##	aliph2HC_carboxylO_medshort	aliph2HC_scLysN_medlong
##	-0.032709942	-0.042976497
##	aliph3HC_bbO_medlong	aliph1HC_carboxylO_vlong
##	0.014514297	0.019793344
##	aliph2HC_aliph2HC_vlong	carboxylC_bbC_medlong
##	0.005725962	0.037538066
##	aliph3HC_aromaticC_medshort	carboxylO_bbN_vshort
##	0.010878726	0.101576622
##	carboxylO_bbCA_medshort	carboxylC_bbO_vlong
##	0.052583382	-0.024579277
##	aliph1HC_scAGN_vlong	aliph1HC_carboxylO_vlong
##	0.019577410	-0.052434293
##	carbonylC_bbN_long	aliph3HC_aliph3HC_long
##	-0.008464008	0.041788129
##	aliph3HC_aliph3HC_medlong	hydroxylO_bbN_long
##	0.032817099	0.010621954
##	aliph3HC_carboxylO_medlong	carbonylC_aliph1HC_medshort
##	0.019437489	-0.051558885
##	hydroxylO_hydroxylO_medlong	bbProN_carboxylO_medlong
##	0.029409867	0.044446613
##	scAGN_scLysN_medlong	aliph3HC_carboxylO_short
##	-0.071215490	0.027438451
##	carboxylO_bbO_vlong	scLysN_bbN_long
##	0.009993058	-0.058036401

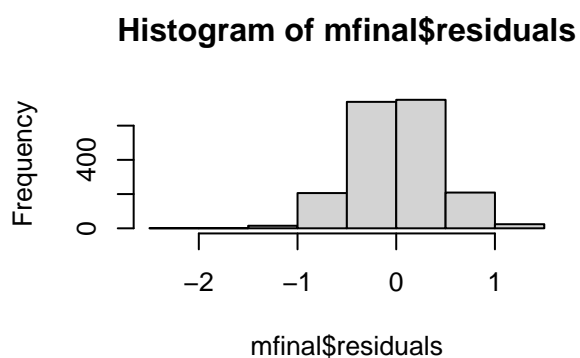
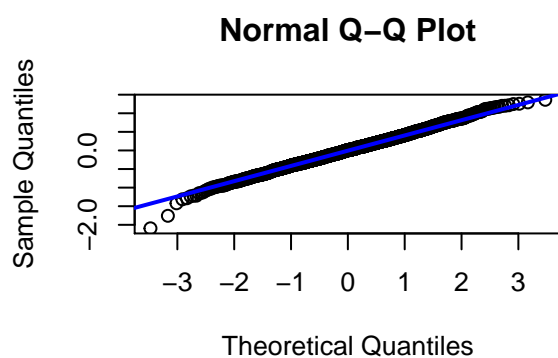
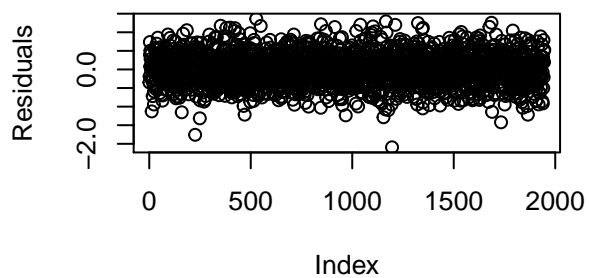
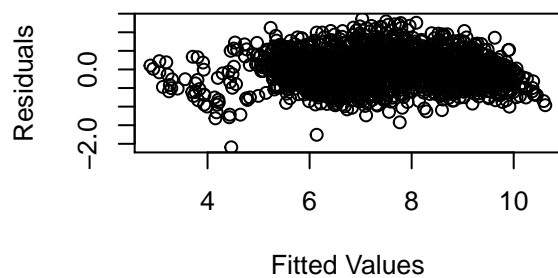
```
##          carbonyl0_bbCA_long          bbN_bbN_long
##          -0.010172362          -0.011761693
##          bbN_bbN_vlong      aliph2HC_hydroxyl0_vlong
##          -0.011307629          0.005828448
##          aliph3HC_scLysN_vlong      sulfur_bbN_long
##          0.054436693          0.015240198
##          aliph3HC_bbCA_medshort      sulfur_bbN_medshort
##          -0.016090571          0.031679472
##          sulfur_bbN_short      carboxylC_carboxyl0_vlong
##          0.053296717          0.026011662
## hydroxyl0_carboxyl0_medshort      carboxyl0_carboxyl0_long
##          -0.073726235          0.056920368
##          aromaticC_bb0_long      aromaticC_aromaticC_short
##          0.006316804          -0.009064569
##          aromaticC_bbN_medshort      carbonyl0_sulfur_medshort
##          -0.009741127          0.033785599
##          aliph2HC_bbC_long      aliph3HC_carboxyl0_medshort
##          0.004397429          0.049722809
```

We check the model for linear regression assumptions again

```
#summary(premov)
par ( mfrow = c (2 ,2) )

plot(mfinal$fitted.values, mfinal$residuals, xlab = "Fitted Values",
     ylab = "Residuals")
plot (1: nrow (ptrain), mfinal$residuals , xlab = " Index ", ylab =
     " Residuals ")

qqnorm(mfinal$residuals)
qqline(mfinal$residuals, col="blue", lwd = 2)
hist(mfinal$residuals)
```



```
# use final model to predict response for protein-test.csv

ptest <- read.csv ("protein-test.csv")

predfinal <- predict(mfinal, newdata = ptest)

writeLines(as.character(predfinal), "mypreds1.txt.txt")
```