Verkefni 5

Árni Víðir Jóhannesson & Ottó Hólm Reynisson

11.október 2016

The model

In the description of this project, we are asked to make a model to evaluate houseprices(nuvirdi) in Reykjavik. The first step is usually to model the price as a function of all other explanatory variables and work from there.That is

$$y_{nuvirdi} = \beta_1 x_{kaupdagur} + \dots$$

We will be doing our diagnostics with respect to this model.

Residuals

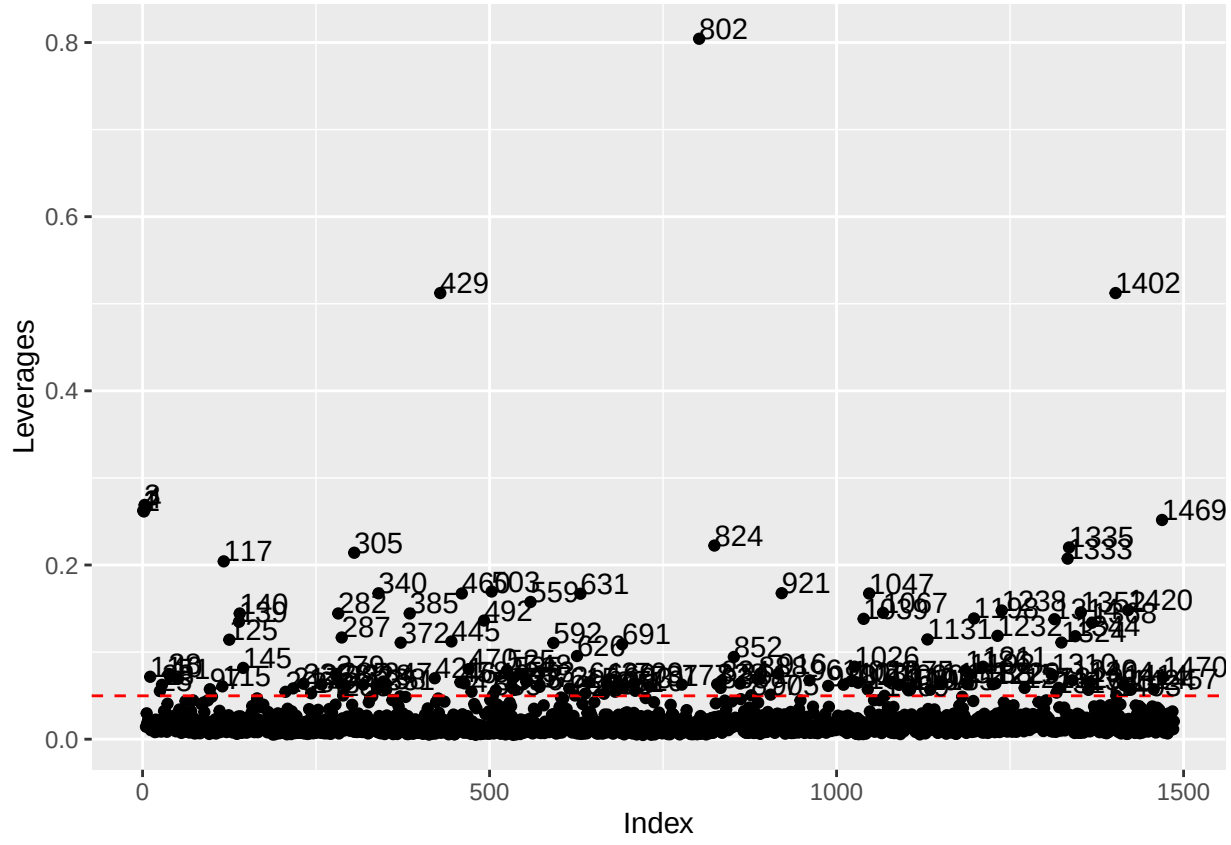
Begin by looking at the residuals from this model

Mynd 1: Indexplot for the residuals.

Here the blue and green line represent 2 and 3 standard deviations from the mean. We identify those points that are two standard deviations away from the mean. We clearly see that there are some possible outliers that need further diagnostics.

Leverages

The next thing to do is looking at the leverages, that is the measure of how far independent variable values of an observation are from those of the other observation. Figure two marks those points that are more than $\frac{2p}{n} = \frac{2 \cdot 37}{1486} \approx 0.05$

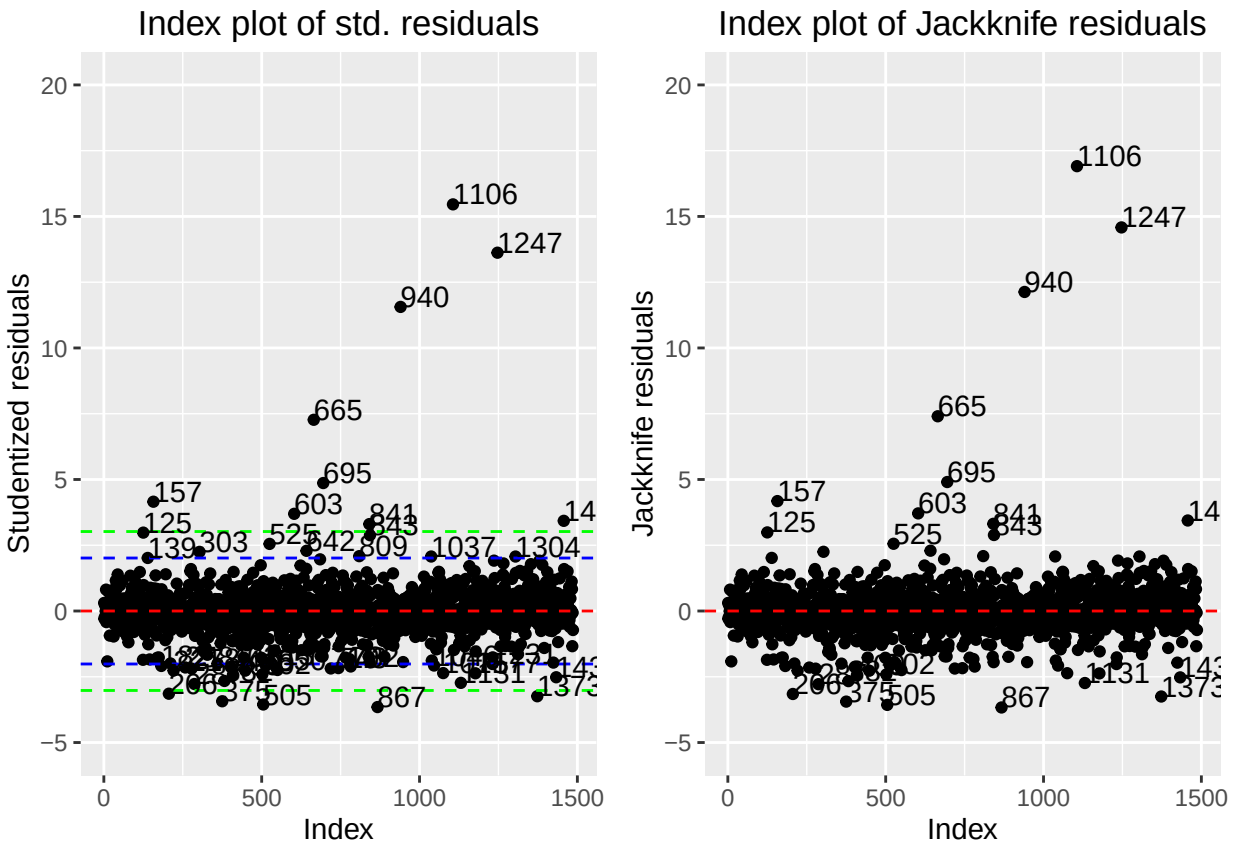


Mynd 2: Indexplot of leverages

Studentized residuals

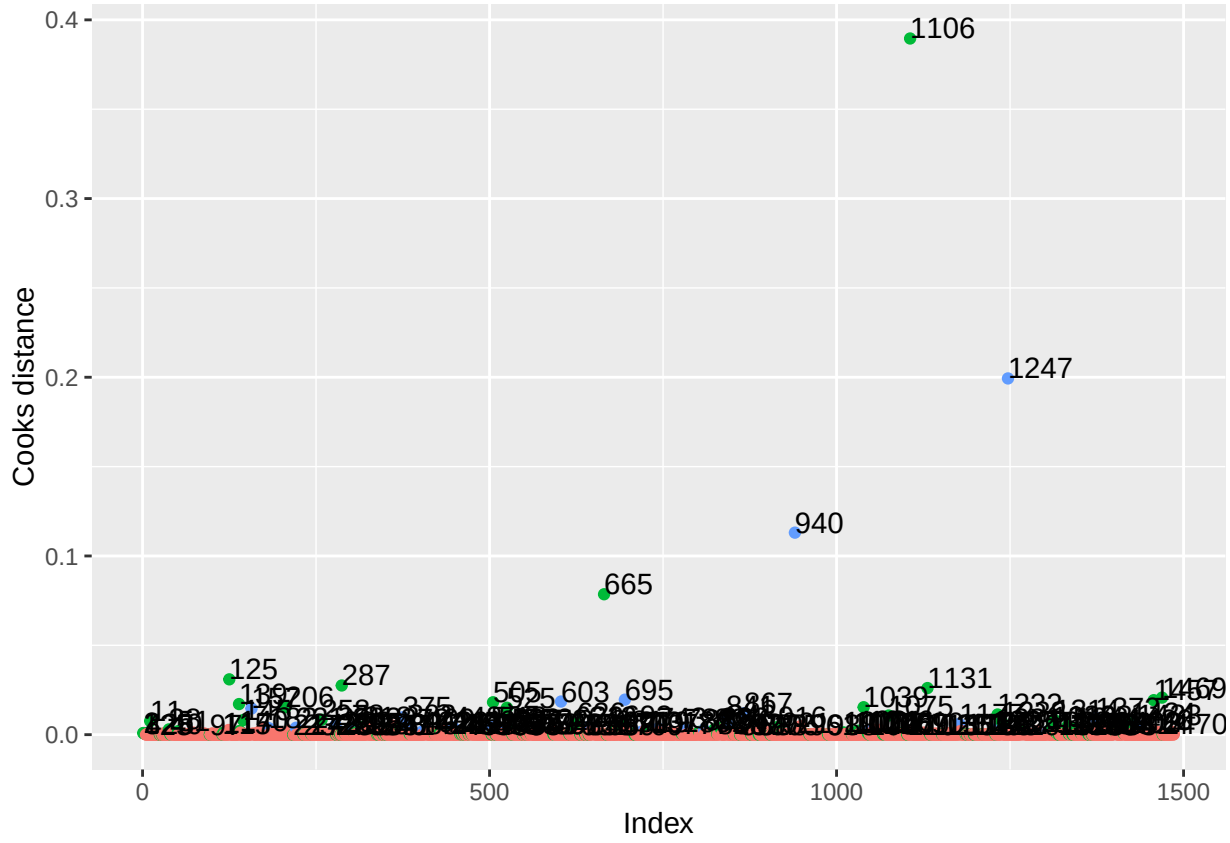
Studentized residuals are sometimes preferred in residual plots as they have been standardized to have equal variance. They are also a big part in the Jackkiife residulas that follows

Blue and green line represent as before 2 and 3 sd from the mean.



Cook's distance

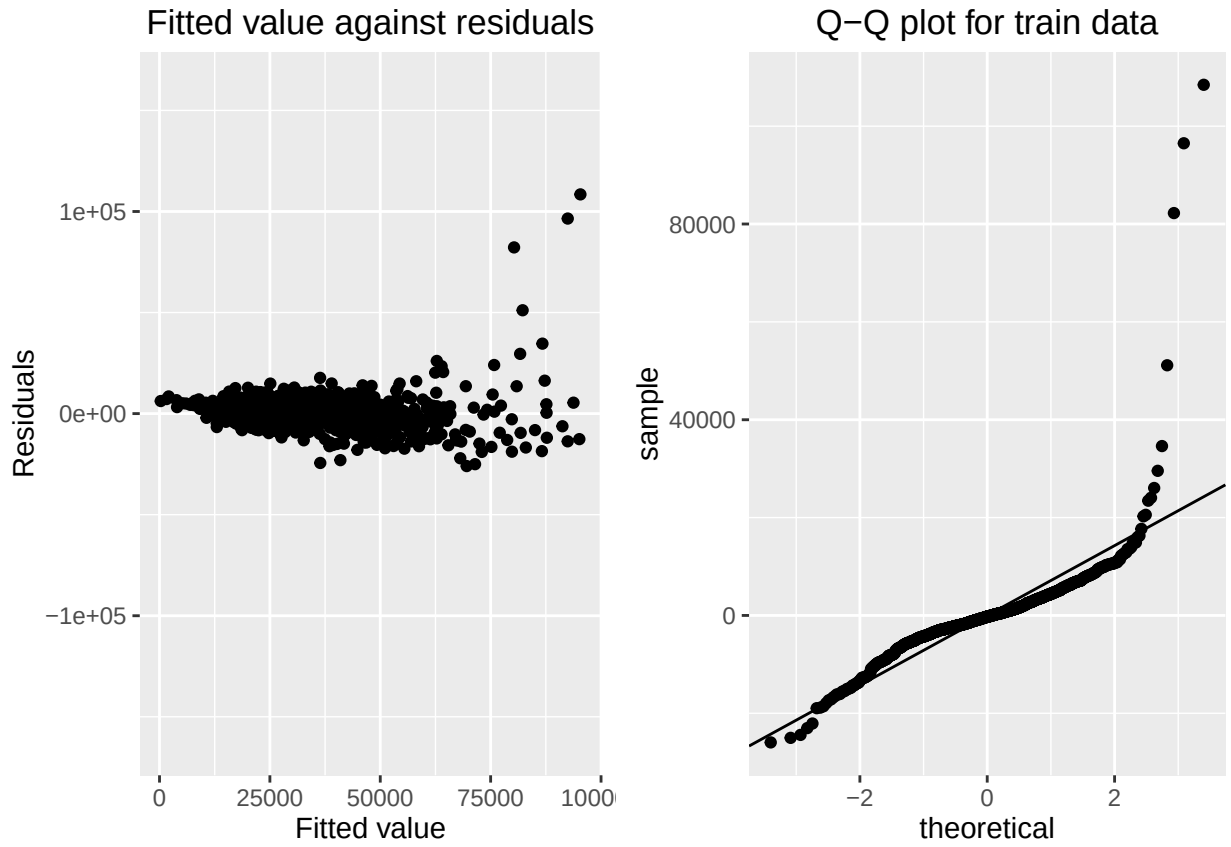
Cook's distance (calculated w.r.t Jackknife and Std.Residuals) is a good way to diagnose influential points in the model. Points with high Cooks distance are affecting the model more than the others. The green points have high Cooks distance, but the blue points have high Cooks distance but also high leverage.



Q-Q plot and fitted plot

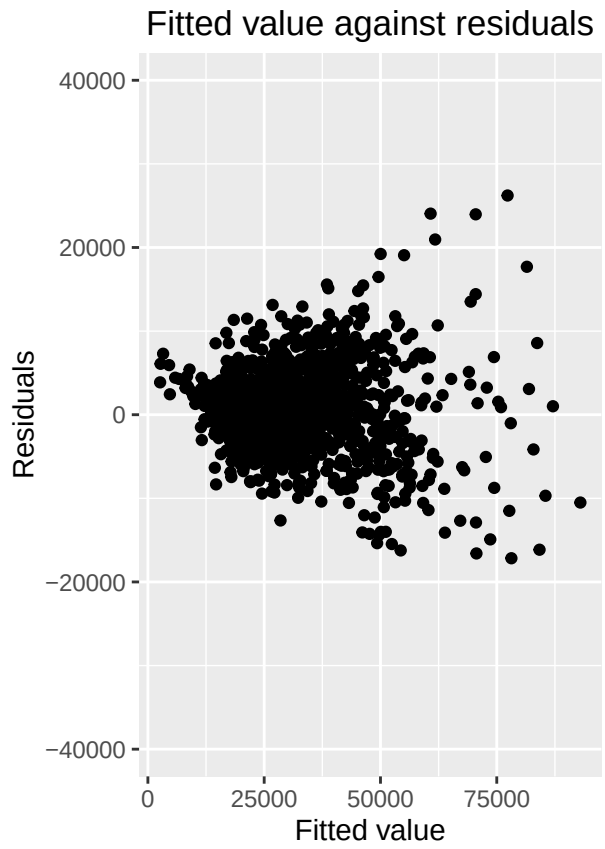
To see how well the model fits the data, we plot the fitted value against residuals. This should be scatterplot with no specified form.

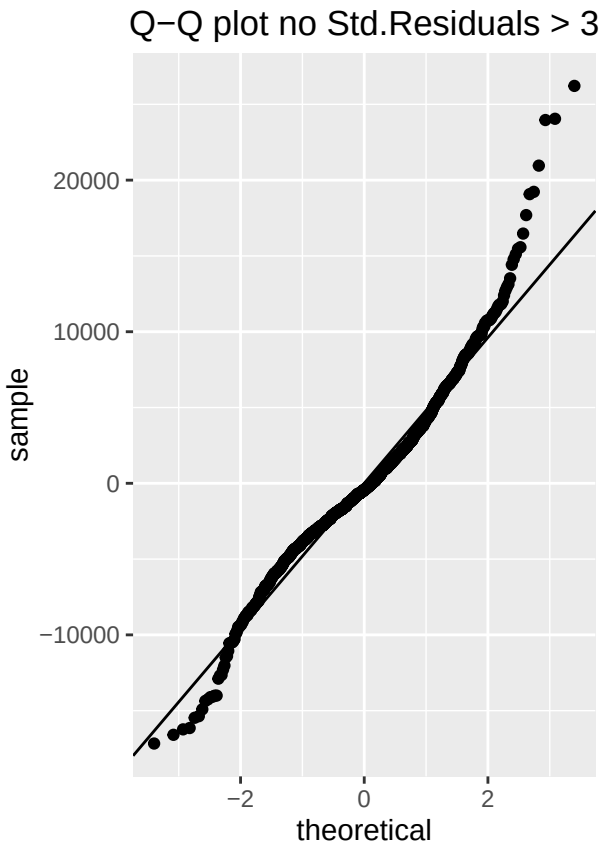
We clearly see this is not what we expected to see. Also the QQ plot This means we have to do some transformation and remove the biggest outliers.



First we calculate the R-adjusted for the first model and the whole train data and get $R - adjusted$= 0.8520812. The R-squared for this data set and model is, $R - squared$=0.8018309 Now by removing the residuals that have std. residuals > 3 we have new model.

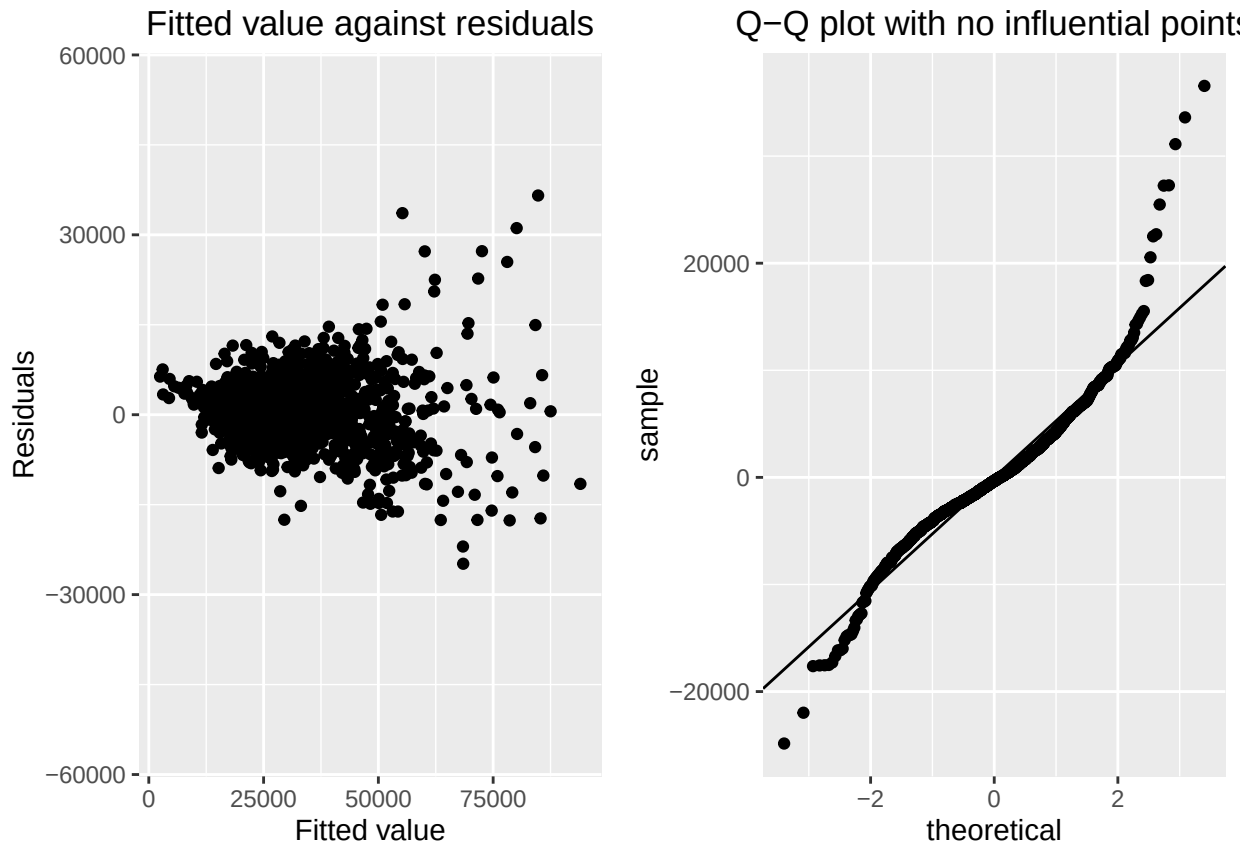
The plots below show that the model in instatnly better for our train data, just by removing some outliers.





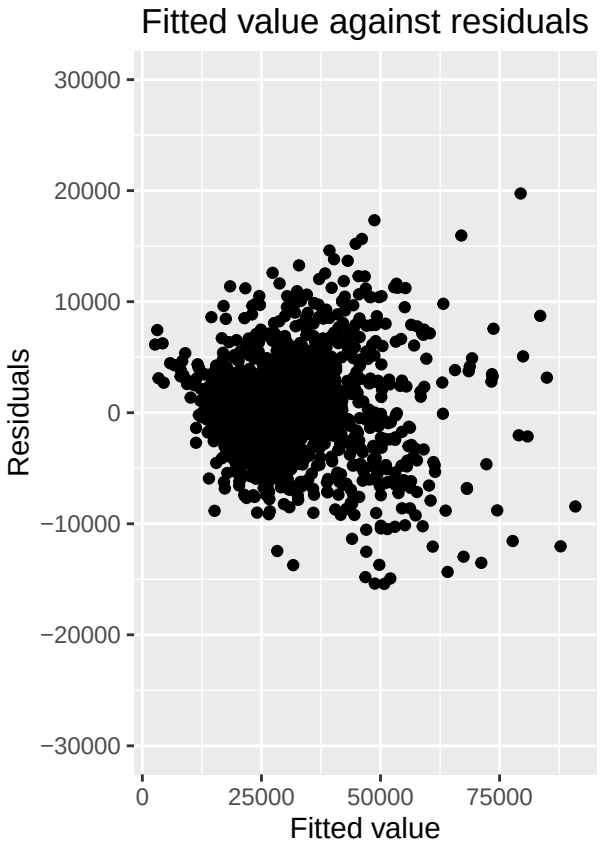
The R-adjusted for fitting the model with new train data is $R - adjustedNooutliers$=0.8572009 while R-squared for the train data gets better, $R - squared$=0.875189

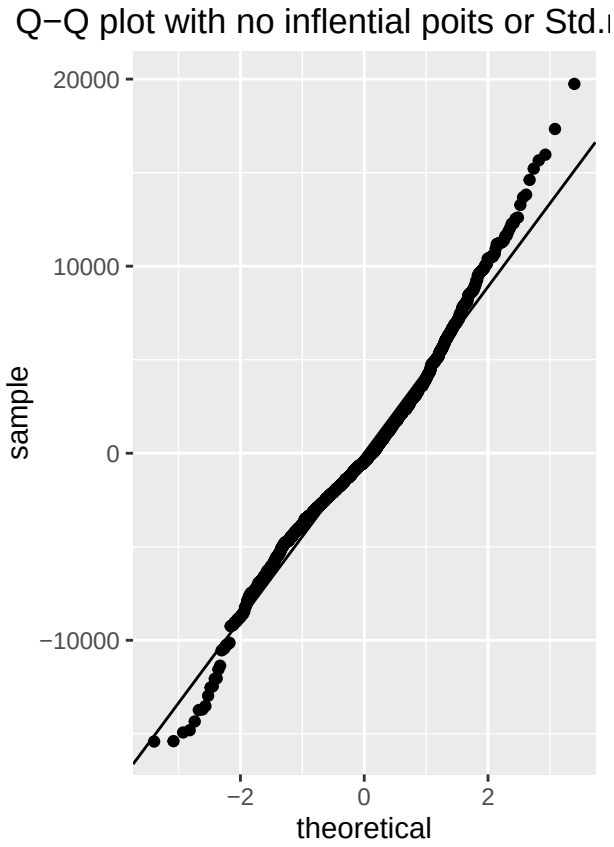
With the Cooks distance we can find the most influential points affecting our model. We want to remove all influential points with Cooks distance > 0.0017953 and see how to model fits to that data.



Now our R-adjusted is still worse than for the whole train data, $R - adjusted$= 0.8574566 while R-squared keeps getting higher $R - squared$=0.8624224

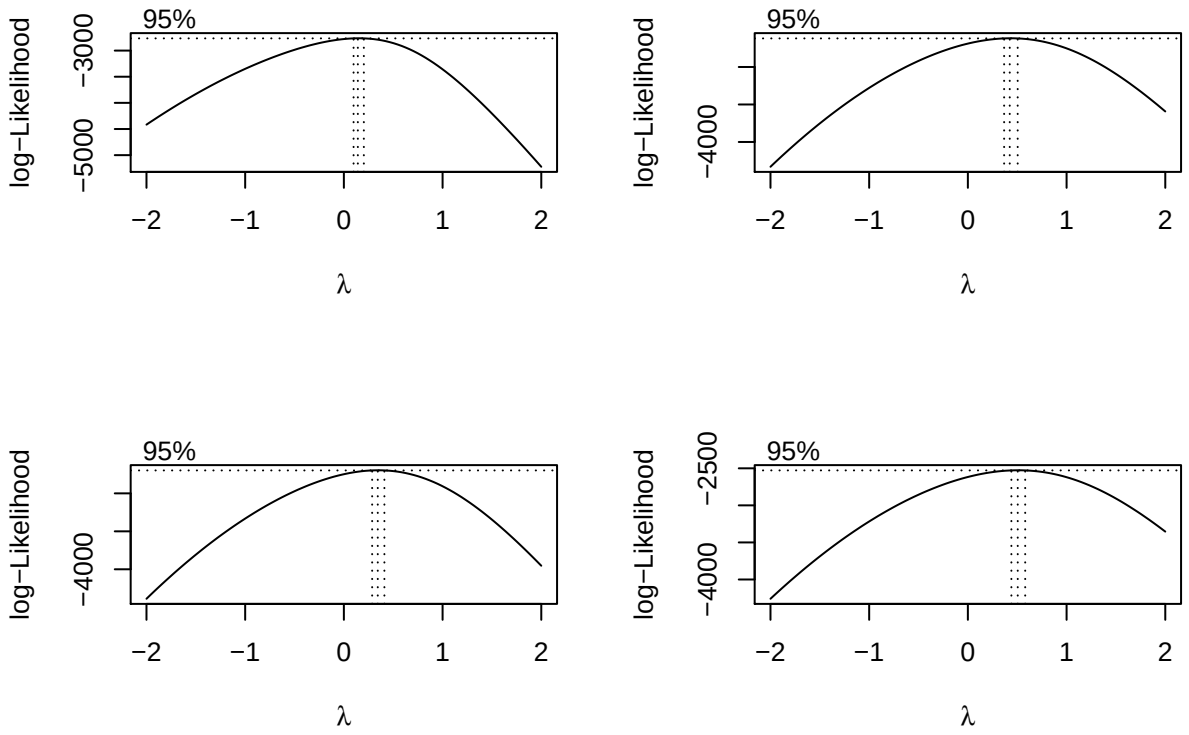
Last data set we make is with no influential points and no outliers. The previous model should fit this data set very well but on the other half R-adjuested might be getting lower.


Fitted value against residuals
Residuals
Fitted value
30000
20000
10000
0
−10000
−20000
−30000
0
25000
50000
75000


Q−Q plot with no inflential poits or Std.
sample
theoretical
20000
10000
0
−10000
−2
0
2

Transformation

We know that nuvirdi has an unusually heavy tale so we'll start by transforming our response variable using boxcox.



Mynd 3: Boxcox plot for the four models. Top right: Model with all the training data, top left: Model with no outliers, bottom right: Model with no influential points and bottom left: Model with no outliers and no influential points.

```
Radj.ALLBC <- BCTranformResponseRadj(lm.all, train, test)
Radj.NOBC <- BCTranformResponseRadj(lm.allNoOutlier, trainNO, test)
Radj.NIBC <- BCTranformResponseRadj(lm.allNoInfluential, trainNoInflu, test)
Radj.NONIBC <- BCTranformResponseRadj(lm.allNoInflueNoOutlier, trainNONI, test)
```

Here below we can see the R_{adj} for the four models after transforming the response variable. R_{adj} is calculated using the test set.

	No changes	No outl.	No infl.	No outl. and no infl.
R_{adj}	0.8335258	0.8520399	0.8446827	0.8527298

From the ggpairs image we can see that *ibm*2 has a heavy right tail as well so lets try log-transforming that variable to see if we get better results.

```
Radj.AllBCAndIBM2 <- TransformBCandIBM2(lm.all, train, test)
Radj.NOBCAndIBM2 <- TransformBCandIBM2(lm.allNoOutlier, train, test)
Radj.NIBCAndIBM2 <- TransformBCandIBM2(lm.allNoInfluential, trainNoInflu, test)
Radj.NONIBCAndIBM2 <- TransformBCandIBM2(lm.allNoInflueNoOutlier, trainNONI, test)
```

Here below we can see the R_{adj} for the four models after transforming the response variable and $ibm2$. R_{adj} is calculated using the test set. Now we get much better results for R_{adj}.

	No changes	No outl.	No infl.	No outl. and no infl.
R_{adj}	0.8871675	0.8816331	0.8813979	0.8693338

Variable selection

We saw from the transformation chapter that we got the best models by transforming both the response variable and $ibm2$. So we'll be using those models for variable selection.

```
# Feching models and datasets
ALL <- GetBCandIBM2ModelAndDt(lm.all,train, test)
NO <- GetBCandIBM2ModelAndDt(lm.allNoOutlier, trainNO, test)
NI <- GetBCandIBM2ModelAndDt(lm.allNoInfluential, trainNoInflu, test)
NONI <- GetBCandIBM2ModelAndDt(lm.allNoInflueNoOutlier, trainNONI, test)
```

Lets try to use BIC and AIC criteria to select our variables.

```
# BIC tests
ALLBIC <- findBestBICModel(lm(nuvirdi ~ 1, data = ALL$train),
                           ALL$model, ALL$train, ALL$test, ALL$lambda)
NOBIC <- findBestBICModel(lm(nuvirdi ~ 1, data = NO$train),
                          NO$model, NO$train, NO$test, NO$lambda)
NIBIC <- findBestBICModel(lm(nuvirdi ~ 1, data = NI$train),
                          NI$model, NI$train, NI$test, NI$lambda)
NONIBIC <- findBestBICModel(lm(nuvirdi ~ 1, data = NONI$train),
                            NONI$model, NONI$train, NONI$test, NONI$lambda)

# AIC tests
ALLAIC <- findBestAICModel(lm(nuvirdi ~ 1, data = ALL$train),
                           ALL$model, ALL$train, ALL$test, ALL$lambda)
NOAIC <- findBestAICModel(lm(nuvirdi ~ 1, data = NO$train),
                          NO$model, NO$train, NO$test, NO$lambda)
NIAIC <- findBestAICModel(lm(nuvirdi ~ 1, data = NI$train),
                          NI$model, NI$train, NI$test, NI$lambda)
NONIAIC <- findBestAICModel(lm(nuvirdi ~ 1, data = NONI$train),
                            NONI$model, NONI$train, NONI$test, NONI$lambda)
```

We can see that we get the best R_{adj} when using the AIC crite

	No changes	No outl.	No infl.	No outl. and no infl.
$R_{adj(BIC)}$	0.8866053	0.8770973	0.8804325	0.8712037
$R_{adj(AIC)}$	0.8876416	0.8776122	0.8823058	0.8715814

Lets now try something different. Lets use the transformed data without any changes and use the add1 function to add explanatory variables.

```
add1(lm(nuvirdi~1, data = ALL$train),~ ibm2 + kdagur + matssvaedi + teg_eign + undirmatssvaedi +
       haednr + fjhaed+fjstof+byggar+fjsturt+stig10+fjbilast+fjbkar+ibteg+k.ar+byggar+lyfta+
       fjklos+fjeld+fjherb ,data=ALL$train, test = "F")
```

Lets start by adding ibm2.

```
add1(lm(nuvirdi~ibm2, data = ALL$train),~ ibm2 + kdagur + matssvaedi + teg_eign + undirmatssvaedi +
       haednr + fjhaed+fjstof+byggar+fjsturt+stig10+fjbilast+fjbkar+ibteg+k.ar+byggar+lyfta+
       fjklos+fjeld+fjherb ,data=ALL$train, test = "F")
```

Lets now add matssvaedi. First lets see what model to use.

```
drop1(lm(nuvirdi~ibm2*matssvaedi, data = ALL$train), test = "F")
```

```
## Single term deletions
##
## Model:
## nuvirdi ~ ibm2 * matssvaedi
##                 Df Sum of Sq    RSS    AIC F value    Pr(>F)
## <none>                       4151.3 1546.6
## ibm2:matssvaedi  4     181.8 4333.1 1602.3   16.16 5.849e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can see from the drop1 function that the best model seems to have different slope and different intercept when just using ibm2 and matssvaedi. Lets continue adding variables.

```
lm.temp <- lm(nuvirdi~ibm2*matssvaedi, data = ALL$train)
add1(lm.temp,~ ibm2*matssvaedi + kdagur + teg_eign + undirmatssvaedi + haednr+   fjhaed+fjstof+byggar+f
fjklos+fjeld+fjherb ,data=ALL$train, test = "F")
Radj.temp <- CalculateRadjLambda(lm.temp, ALL$test, ALL$lambda)
# Adding kdagur
lm.temp <- lm(nuvirdi~ibm2*matssvaedi+kdagur, data = ALL$train)
add1(lm.temp,~ ibm2*matssvaedi + kdagur + teg_eign + undirmatssvaedi + haednr+   fjhaed+fjstof+byggar+f
lyfta+fjklos+fjeld+fjherb ,data=ALL$train, test = "F")
Radj.temp <- CalculateRadjLambda(lm.temp, ALL$test, ALL$lambda)
# Adding teg_eign
lm.temp <- lm(nuvirdi~ibm2*matssvaedi+kdagur+teg_eign, data = ALL$train)
add1(lm.temp,~ ibm2*matssvaedi + kdagur + teg_eign + undirmatssvaedi + haednr+ fjhaed+fjstof+byggar+fjs
       lyfta+fjklos+fjeld+fjherb ,data=ALL$train, test = "F")
Radj.temp <- CalculateRadjLambda(lm.temp, ALL$test, ALL$lambda)
# Adding byggar
lm.temp <- lm(nuvirdi~ibm2*matssvaedi+kdagur+teg_eign+byggar, data = ALL$train)
add1(lm.temp,~ ibm2*matssvaedi + kdagur + teg_eign + undirmatssvaedi + haednr+ fjhaed+fjstof+byggar+fjs
lyfta+fjklos+fjeld+fjherb ,data=ALL$train, test = "F")
Radj.temp <- CalculateRadjLambda(lm.temp, ALL$test, ALL$lambda)
# Adding undirmatssvaedi
lm.temp <- lm(nuvirdi~ibm2*matssvaedi+kdagur+teg_eign+byggar+
                undirmatssvaedi, data = ALL$train)
add1(lm.temp,~ ibm2*matssvaedi + kdagur + teg_eign + undirmatssvaedi + haednr+ fjhaed+fjstof+byggar+fjs
       lyfta+fjklos+fjeld+fjherb ,data=ALL$train, test = "F")
Radj.temp <- CalculateRadjLambda(lm.temp, ALL$test, ALL$lambda)
# Adding haednr
lm.temp <- lm(nuvirdi~ibm2*matssvaedi+kdagur+teg_eign+byggar+
                undirmatssvaedi+haednr, data = ALL$train)
add1(lm.temp,~ ibm2*matssvaedi + kdagur + teg_eign + undirmatssvaedi + haednr+ fjhaed+fjstof+byggar+fjs
       lyfta+fjklos+fjeld+fjherb ,data=ALL$train, test = "F")
Radj.temp <- CalculateRadjLambda(lm.temp, ALL$test, ALL$lambda)
# Adding fjhaed
lm.temp <- lm(nuvirdi~ibm2*matssvaedi+kdagur+teg_eign+byggar+
                undirmatssvaedi+haednr+fjhaed, data = ALL$train)
add1(lm.temp,~ ibm2*matssvaedi + kdagur + teg_eign + undirmatssvaedi + haednr+ fjhaed+fjstof+byggar+fjs
lyfta+fjklos+fjeld+fjherb ,data=ALL$train, test = "F")
Radj.temp <- CalculateRadjLambda(lm.temp, ALL$test, ALL$lambda)
# Adding fjbilast
```

```
lm.temp <- lm(nuvirdi~ibm2*matssvaedi+kdagur+teg_eign+byggar+
                undirmatssvaedi+haednr+fjhaed+fjbilast, data = ALL$train)
add1(lm.temp,~ ibm2*matssvaedi + kdagur + teg_eign + undirmatssvaedi + haednr+ fjhaed+fjstof+byggar+fjs
lyfta+fjklos+fjeld+fjherb ,data=ALL$train, test = "F")
Radj.temp <- CalculateRadjLambda(lm.temp, ALL$test, ALL$lambda)
# Adding fjstof
lm.temp <- lm(nuvirdi~ibm2*matssvaedi+kdagur+teg_eign+byggar+
                undirmatssvaedi+haednr+fjhaed+fjbilast+
                fjstof, data = ALL$train)
add1(lm.temp,~ ibm2*matssvaedi + kdagur + teg_eign + undirmatssvaedi + haednr+ fjhaed+fjstof+byggar+fjs
lyfta+fjklos+fjeld+fjherb ,data=ALL$train, test = "F")
Radj.temp <- CalculateRadjLambda(lm.temp, ALL$test, ALL$lambda)
# adding lyfta
lm.temp <- lm(nuvirdi~ibm2*matssvaedi+kdagur+teg_eign+byggar+
                undirmatssvaedi+haednr+fjhaed+fjbilast+fjstof+
                lyfta, data = ALL$train)
add1(lm.temp,~ ibm2*matssvaedi + kdagur + teg_eign + undirmatssvaedi + haednr + fjhaed+fjstof+byggar+fjs
lyfta+fjklos+fjeld+fjherb ,data=ALL$train, test = "F")
Radj.lyfta <- CalculateRadjLambda(lm.temp, ALL$test, ALL$lambda)
Radj.add1lyftaTrain <- summary(lm.temp)$adj.r.squared
# adding fjsturt
lm.temp <- lm(nuvirdi~ibm2*matssvaedi+kdagur+teg_eign+byggar+
                undirmatssvaedi+haednr+fjhaed+fjbilast+fjstof+
                lyfta+fjsturt, data = ALL$train)
add1(lm.temp,~ ibm2*matssvaedi + kdagur + teg_eign + undirmatssvaedi + haednr + fjhaed+fjstof+byggar+fjs
lyfta+fjklos+fjeld+fjherb ,data=ALL$train, test = "F")
Radj.fjsturt <- CalculateRadjLambda(lm.temp, ALL$test, ALL$lambda)
Radj.add1fjsturtTrain <- summary(lm.temp)$adj.r.squared
# adding stig10
lm.temp <- lm(nuvirdi~ibm2*matssvaedi+kdagur+teg_eign+byggar+
                undirmatssvaedi+haednr+fjhaed+fjbilast+fjstof+
                lyfta+fjsturt+stig10, data = ALL$train)
add1(lm.temp,~ ibm2*matssvaedi + kdagur + teg_eign + undirmatssvaedi + haednr + fjhaed+fjstof+byggar+fjs
lyfta+fjklos+fjeld+fjherb ,data=ALL$train, test = "F")
Radj.add1Final <- CalculateRadjLambda(lm.temp, ALL$test, ALL$lambda)
Radj.add1FinalTrain <- summary(lm.temp)$adj.r.squared
```

The table below shows R_{adj} for the last three steps when using the add1 function.

	Add lyfta	Add fjsturt	Add stig10
$R_{adj(add1)Test}$	0.8961315	0.8975622	0.8952104
$R_{adj(add1)}Train$	0.9002563	0.90061	0.9010742

After using the add1 function until there was no significant explanatory variable left we got $R_{adj} = 0.8952104$. Lets try using drop1 instead with different intercept and slope for matsvaedi.

```
lm.temp <- lm(nuvirdi ~ ibm2*matssvaedi + kdagur + teg_eign + undirmatssvaedi +
                haednr + fjhaed+fjstof+byggar+fjsturt+stig10+fjbilast+fjbkar+
                ibteg+k.ar+byggar+lyfta+fjklos+fjeld+fjherb, data = ALL$train)
drop1(lm.temp, test = "F")
Radj.temp <- CalculateRadjLambda(lm.temp, ALL$test, ALL$lambda)
# Dropping k.ar
lm.temp <- lm(nuvirdi ~ ibm2*matssvaedi + kdagur + teg_eign + undirmatssvaedi +
```

```
                haednr + fjhaed+fjstof+byggar+fjsturt+stig10+fjbilast+fjbkar+
                ibteg+byggar+lyfta+fjklos+fjeld+fjherb, data = ALL$train)
drop1(lm.temp, test = "F")
Radj.temp <- CalculateRadjLambda(lm.temp, ALL$test, ALL$lambda)
# Dropping fjklos
lm.temp <- lm(nuvirdi ~ ibm2*matssvaedi + kdagur + teg_eign + undirmatssvaedi +
                haednr + fjhaed+fjstof+byggar+fjsturt+stig10+fjbilast+fjbkar+
                ibteg+byggar+lyfta+fjeld+fjherb, data = ALL$train)
drop1(lm.temp, test = "F")
Radj.temp <- CalculateRadjLambda(lm.temp, ALL$test, ALL$lambda)
# Dropping fjeld
lm.temp <- lm(nuvirdi ~ ibm2*matssvaedi + kdagur + teg_eign + undirmatssvaedi +
                haednr + fjhaed+fjstof+byggar+fjsturt+stig10+fjbilast+fjbkar+
                ibteg+byggar+lyfta+fjherb, data = ALL$train)
drop1(lm.temp, test = "F")
Radj.temp <- CalculateRadjLambda(lm.temp, ALL$test, ALL$lambda)
# Dropping ibteg
lm.temp <- lm(nuvirdi ~ ibm2*matssvaedi + kdagur + teg_eign + undirmatssvaedi +
                haednr + fjhaed+fjstof+byggar+fjsturt+stig10+fjbilast+fjbkar+
                byggar+lyfta+fjherb, data = ALL$train)
drop1(lm.temp, test = "F")
Radj.drIbteg <- CalculateRadjLambda(lm.temp, ALL$test, ALL$lambda)
Radj.drIbtegTrain <- summary(lm.temp)$adj.r.squared
# Dropping fjherb
lm.temp <- lm(nuvirdi ~ ibm2*matssvaedi + kdagur + teg_eign + undirmatssvaedi +
                haednr + fjhaed+fjstof+byggar+fjsturt+stig10+fjbilast+fjbkar+
                byggar+lyfta, data = ALL$train)
drop1(lm.temp, test = "F")
Radj.drfjherb <- CalculateRadjLambda(lm.temp, ALL$test, ALL$lambda)
Radj.drfjherbTrain <- summary(lm.temp)$adj.r.squared
# Dropping fjbkar
lm.temp <- lm(nuvirdi ~ ibm2*matssvaedi + kdagur + teg_eign + undirmatssvaedi +
                haednr + fjhaed+fjstof+byggar+fjsturt+stig10+fjbilast+byggar+
                lyfta, data = ALL$train)
drop1(lm.temp, test = "F")
Radj.drfjbkar<- CalculateRadjLambda(lm.temp, ALL$test, ALL$lambda)
Radj.drfjbkarTrain <- summary(lm.temp)$adj.r.squared
```

Table below shows R_{adj} for the last three steps when using the drop1 function.

	Drop ibteg	Drop fjherb	Drop fjbkar
$R_{adj(drop1)Test}$	0.8933716	0.8951836	0.8952104
$R_{adj(drop1)Train}$	0.9014552	0.9012159	0.9010742

Lets now try to use BIC and AIC starting with the model (nuvirdi ~ ibm2*matssvaedi).

```
null <- lm(nuvirdi~ibm2*matssvaedi, data = ALL$train)
full <- lm(nuvirdi~ibm2*matssvaedi + kdagur + teg_eign + undirmatssvaedi +
             haednr + fjhaed+fjstof+byggar+fjsturt+stig10+fjbilast+fjbkar+
             ibteg+k.ar+byggar+lyfta+fjklos+fjeld+fjherb, data = ALL$train)
# BIC tests
ALLBIC <- findBestBICModel(null, full, ALL$train, ALL$test, ALL$lambda)
RadjBICTrain <- summary(ALLBIC$model)$adj.r.squared
```

```
# AIC tests
ALLAIC <- findBestAICModel(null, full, ALL$train, ALL$test, ALL$lambda)
RadjAICTrain <- summary(ALLAIC$model)$adj.r.squared
```

The table below shows the best R_{adj} for each test when we start with different intercepts for matssvaedi.

	add1	drop1	AIC	BIC
$R_{adjTest}$	0.8975622	0.8952104	0.8933716	0.8931438
$R_{adjTrain}$	0.90061	0.9010742	0.9014552	0.9002197

Appendix

```
library(MASS)
library(ggplot2)
library(plyr)
library(faraway)
library(car)
library(gridExtra)
library(MASS)
library(data.table)
library(GGally)

verdmat <-read.table("gagnasafn_endurmat2017_litid.csv", sep=',', header=T)
RE_data <- data.frame(verdmat)
dt.re <- data.table(verdmat)
#Matsv:
#  Vesturb. 11
#  miðbær s þ 31
#  hlíðar 80
#  grafarvogur 120
#  seljahverfi 150

dt.verdmat <- data.table(verdmat)
dt.RE_data <- dt.verdmat[matssvaedi %in% c(11,31,80,120,150)]
# Bæta við dálki kaupár
dt.RE_data <- dt.RE_data[,k.ar:=as.numeric(substring(kdagur,1,4))]
dt.RE_data <- dt.RE_data[,matssvaedi:=as.factor(matssvaedi)]
dt.RE_data <- dt.RE_data[,undirmatssvaedi:=as.factor(undirmatssvaedi)]
dt.RE_data <- dt.RE_data[,teg_eign:=as.factor(teg_eign)]


# Byrjum á að taka út þær breytur sem við teljum ekki
dt.RE_data <- subset(dt.RE_data, select=c("kdagur","nuvirdi","teg_eign","byggar","haednr","lyfta","ibm2
                                          "fjbilast","fjbkar","fjsturt","fjklos","fjeld","fjherb",
                                          "fjstof","fjgeym","stig10","matssvaedi","undirmatssvaedi","ib

dt.RE_data <- dt.RE_data[,kdagur:=as.Date(kdagur)]

#breyta lyftu í boolean og breyta ibteg í factor, 11=serbýli, 12=fjölbýli

for(i in 1:nrow(dt.RE_data)){
  if(dt.RE_data$lyfta[i]!=0){dt.RE_data$lyfta[i]=1}
  else{}
}

for(i in 1:nrow(dt.RE_data)){
  if(dt.RE_data$ibteg[i]==11){dt.RE_data$ibteg="Serbyli"}
  else{dt.RE_data$ibteg[i]="Fjolbyli"}
}

dt.RE_data <- dt.RE_data[,ibteg:=as.factor(ibteg)]

n<-dim(dt.RE_data)[1]
```

```
# Training and testing data sets
set.seed(1)
rows<-sample(1:n,n/3)
train <- data.table(dt.RE_data[-rows,])
test <- data.table(dt.RE_data[rows,])


multiplot <- function(..., plotlist=NULL, file, cols=1, layout=NULL) {
  library(grid)

  # Make a list from the ... arguments and plotlist
  plots <- c(list(...), plotlist)

  numPlots = length(plots)

  # If layout is NULL, then use 'cols' to determine layout
  if (is.null(layout)) {
    # Make the panel
    # ncol: Number of columns of plots
    # nrow: Number of rows needed, calculated from # of cols
    layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),
                     ncol = cols, nrow = ceiling(numPlots/cols))
  }

  if (numPlots==1) {
    print(plots[[1]])

  } else {
    # Set up the page
    grid.newpage()
    pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout))))

    # Make each plot, in the correct location
    for (i in 1:numPlots) {
      # Get the i,j matrix positions of the regions that contain this subplot
      matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE))

      print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row,
                                      layout.pos.col = matchidx$col))
    }
  }
}
```

```
if(!exists("dt.RE_data")){ source("setUp.R")}

indexPlotLeverage <- function(model){
  diag <- data.table(fortify(model))
  diag$.index = c(1:length(diag$.resid))
  p<-length(coef(model))
  n<-length(fitted(model))
  lev_index<-ggplot(diag, aes(x=seq(1:length(.hat)),y=.hat))+geom_point()
  lev_index<-lev_index+geom_hline(yintercept=2*p/n, col="red", linetype="dashed")
  lev_index<-lev_index+xlab("Index")+ylab("Leverages")
```

```
  lev_index<-lev_index+geom_text(aes(label=ifelse(.hat>2*p/n,.index,"")),hjust=0, vjust=0)
  lev_index
}
#model<-lm.all
fittedVsresiduals <- function(model){
  diag <- data.table(fortify(model))
  diag$.index = c(1:length(diag$.resid))
  p<-length(coef(model))
  n<-length(fitted(model))
  fitVsres <-ggplot(diag, aes(x=.fitted,y=.resid))+geom_point()+ylim(-1.5*max(diag$.resid),1.5*max(diag$
  fitVsres <- fitVsres+labs(title="Fitted value against residuals")+ylab("Residuals")+xlab("Fitted valu
}


# blue dots have high residuals and green have high leverage
indexPlotCookdistance <- function(model){
  diag <- data.table(fortify(model))
  diag$.index = c(1:length(diag$.resid))
  p <- length(coef(model))
  n <- length(fitted(model))
  cook_index <- ggplot(diag, aes(x=seq(1:length(.cooksd)),y=.cooksd))
  cook_index <- cook_index + geom_point( aes(color = ifelse(abs(.hat)>2*p/n,"blue",ifelse(abs(.stdresid
  cook_index <- cook_index + xlab("Index")+ylab("Cooks distance") + theme(legend.position="none")
  cook_index <- cook_index + geom_text(aes(label=ifelse(abs(.cooksd)>0.1 | abs(.hat)>2*p/n | abs(.stdres
  cook_index
}

indexPlotResiduals <- function(model){
  diag <- data.table(fortify(model))
  diag$.index = c(1:length(diag$.resid))
  res_sd <- sd(diag$.resid)
  res_index <- ggplot(diag, aes(x=seq(1:length(.resid)),y=.resid)) + geom_point()
  res_index <- res_index + geom_hline(yintercept=0, col="red", linetype="dashed")
  res_index <- res_index + geom_hline(yintercept=2*res_sd, col="blue", linetype="dashed")
  res_index <- res_index + geom_hline(yintercept=-2*res_sd, col="blue", linetype="dashed")
  res_index <- res_index + geom_hline(yintercept=3*res_sd, col="green", linetype="dashed")
  res_index <- res_index + geom_hline(yintercept=-3*res_sd, col="green", linetype="dashed")
  res_index <- res_index + xlab("Index")+ylab("Residuals")+labs(title = "Index plot of residuals")
  res_index <- res_index + geom_text(aes(label=ifelse(abs(.resid)>2*res_sd,.index,"")),hjust=-.5, vjust
  res_index
}

indexPlotStResiduals <- function(model){
  diag <- data.table(fortify(model))
  diag$.index = c(1:length(diag$.resid))
  Stres_sd <- sd(diag$.stdresid)
  Stres_index<-ggplot(diag, aes(x=seq(1:length(.stdresid)),y=.stdresid))+geom_point()
  Stres_index<-Stres_index+geom_hline(yintercept=0, col="red", linetype="dashed")
  Stres_index <- Stres_index + geom_hline(yintercept=2*Stres_sd, col="blue", linetype="dashed")
  Stres_index <- Stres_index + geom_hline(yintercept=-2*Stres_sd, col="blue", linetype="dashed")
  Stres_index <- Stres_index + geom_hline(yintercept=3*Stres_sd, col="green", linetype="dashed")
  Stres_index <- Stres_index + geom_hline(yintercept=-3*Stres_sd, col="green", linetype="dashed")
  Stres_index <- Stres_index+xlab("Index")+ylab("Studentized residuals")+labs(title = "Index plot of st
```

```
  Stres_index <- Stres_index+geom_text(aes(label=ifelse(abs(.stdresid)>2*Stres_sd,.index,"")),hjust=0, v
  Stres_index
}

indexPlotJackResiduals <- function(model){
  diag <- data.table(fortify(model))
  diag$.index <- c(1:length(diag$.resid))
  diag$.jack<-rstudent(model)
  Jack_index<-ggplot(diag, aes(x=seq(1:length(.jack)),y=.jack))+geom_point()
  Jack_index<-Jack_index+geom_hline(yintercept=0, col="red", linetype="dashed")
  Jack_index<-Jack_index+xlab("Index")+ylab("Jackknife residuals")
  Jack_index<-Jack_index+geom_text(aes(label=ifelse(abs(.jack)>2.4,.index,"")),hjust=0, vjust=0)
  Jack_index <- Jack_index + labs(title="Index plot of Jackknife residuals")
}


removeOutliersWStdResMoreThanThree <- function(dt){
  model <- lm(nuvirdi ~ .,data=dt)
  diag <- data.table(fortify(model))
  rowsWithNoOutliers <- diag[abs(diag$.stdresid) < 3,]
  rowsWithNoOutliers <- rowsWithNoOutliers[ ,c(".hat",".sigma",".cooksd",".fitted",".resid",".stdresid"]
  return(
    rowsWithNoOutliers
  )
}

#Breytig
# Þurfum líklega að laga þetta fall
removeInfluential <- function(dt, maxCookDistance){
  model <- lm(nuvirdi ~ .,data=dt)
  diag <- data.table(fortify(model))
  rowsWithNoInfluentials <- diag[diag$.cooksd < maxCookDistance,]
  rowsWithNoInfluentials <- rowsWithNoInfluentials[ ,c(".hat",".sigma",".cooksd",".fitted",".resid",".s
  #kdagur <- rowsWithNoInfluentials$kdagur
  #nuvirdi <- rowsWithNoInfluentials$nuvirdi
  #teg_eign <- as.factor(rowsWithNoInfluentials$teg_eign)
  #byggar <- rowsWithNoInfluentials$byggar
  #haednr <- rowsWithNoInfluentials$haednr
  #lyfta <- rowsWithNoInfluentials$lyfta
  #ibm2  <- rowsWithNoInfluentials$ibm2
  #fjhaed <- rowsWithNoInfluentials$fjhaed
  #fjbilast  <- rowsWithNoInfluentials$fjbilast
  #fjbkar  <- rowsWithNoInfluentials$fjbkar
  #fjsturt <- rowsWithNoInfluentials$fjsturt
  #fjklos  <- rowsWithNoInfluentials$fjklos
  #fjeld  <- rowsWithNoInfluentials$fjeld
  #fjherb  <- rowsWithNoInfluentials$fjherb
  #fjstof  <- rowsWithNoInfluentials$fjstof
  #fjgeym  <- rowsWithNoInfluentials$fjgeym
  #stig10  <- rowsWithNoInfluentials$stig10
  #ibteg <- as.factor(rowsWithNoInfluentials$ibteg)
  #k.ar <- rowsWithNoInfluentials$k.ar
  return(
```

```
    rowsWithNoInfluentials
  )
}


QQplotResiduals <- function(model){
  diag <- data.table(fortify(model))
  res_sd <- sd(diag$.resid)
  QQResP<-ggplot(diag, aes(sample = .resid)) + stat_qq()
  QQResP <- QQResP + geom_abline(slope=res_sd)
 # QQResP <- QQResP + ylab("Residuals")
  QQResP <- QQResP + labs(title="QQ plot", ylab="Theoretical",xlab="Residuals")
}

# Þetta fall virkar ekki
RemoveNOutliers <- function(dt, n){
  lm.temp <- lm(nuvirdi ~ ., data=dt)
  diag <- fortify(lm.temp)
  n <- length(diag$.hat)
  diag$.index <- c(1:n)
  temp <- data.table(diag)
  index <- numeric()
  for (i in c(1:n)) {
    append(index, temp$.index[temp$.resid == max(temp$.resid)])
    temp <- temp[temp$.resid != max(temp$.resid), ]
  }
  return(index)
}
```

```
CalculateRadjusted <- function(model, dt){
  y <- dt[,2,with=FALSE]
  X <- dt[,-2, with=FALSE]
  n <- length(y$nuvirdi)
  p <- length(coef(model))
  residuals <- predict.lm(model,X)-y
  y_mean <- mean(y$nuvirdi)
  SStot <- sum((y$nuvirdi-y_mean)^2)
  SSres <- (sum(residuals$nuvirdi^2))
  Rsquared <- 1-SSres/SStot
  Radjusted <- 1-((1-Rsquared)*(n-1))/(n-p-1)
  return(Radjusted)
}

CalculateRadjLambda <- function(model, dt, lambda){
  y <- dt[,2,with=FALSE]
  X <- dt[,-2, with=FALSE]
  n <- length(y$nuvirdi)
  p <- length(coef(model))
  pred <- (predict.lm(model,X)*lambda + 1)^(1/lambda)
  y <- (y$nuvirdi*lambda + 1)^(1/lambda)
  residuals <- pred-y
  y_mean <- mean(y)
  SStot <- sum((y-y_mean)^2)
```

```
  SSres <- (sum(residuals^2))
  Rsquared <- 1-SSres/SStot
  Radjusted <- 1-((1-Rsquared)*(n-1))/(n-p-1)
  return(Radjusted)
}

bcTransF <- function(dt, lambda){
  dt$nuvirdi <- yjPower(dt$nuvirdi, lambda)
  return(dt)
}

BCTranformResponseRadj <- function(model, dTr, dTe){
  bc <- boxcox(model, plotit = FALSE)
  lambda <- with(bc, x[which.max(y)])
  trainBC <- bcTransF(dTr, lambda)
  testBC <- bcTransF(dTe, lambda)
  lm.temp <- lm(nuvirdi ~ ., data = trainBC)
  return(CalculateRadjLambda(lm.temp, testBC, lambda))
}

TransformBCandIBM2 <- function(model, dTr, dTe){
  lambda <- with(boxcox(model, plotit = FALSE), x[which.max(y)])
  trainBC <- bcTransF(dTr, lambda )
  testBC <- bcTransF(dTe, lambda )
  trainBC$ibm2 <- log(trainBC$ibm2)
  testBC$ibm2 <- log(testBC$ibm2)
  lm.temp <- lm(nuvirdi ~ ., data = trainBC)
  return(CalculateRadjLambda(lm.temp, testBC, lambda))
}

GetBCandIBM2ModelAndDt <- function(model, dTr, dTe){
  lambda <- with(boxcox(model, plotit = FALSE), x[which.max(y)])
  trainBC <- bcTransF(dTr, lambda )
  testBC <- bcTransF(dTe, lambda )
  trainBC$ibm2 <- log(trainBC$ibm2)
  testBC$ibm2 <- log(testBC$ibm2)
  lm.temp <- lm(nuvirdi ~ ., data = trainBC)
  return(list(model=lm.temp, train=trainBC, test=testBC, lambda=lambda))
}
```

```
findBestBICModel <- function(null, full, dTr, dTe, lambda){
  lm.BIC <- step(null, scope = list(lower=null,upper=full), direction="both", k=log(dim(dTr)[1]))
  Radj.BICStep <- CalculateRadjLambda(lm.BIC, dTe, lambda) # 0.8889675 rétt: 0.8352896
  return(list(model=lm.BIC, Radj=Radj.BICStep))
}

findBestAICModel <- function(null, full, dTr, dTe, lambda){
  lm.AIC <- step(null, scope = list(lower=null,upper=full), direction="both", k=2)
  Radj.AICStep <- CalculateRadjLambda(lm.AIC, dTe, lambda) # 0.889269 # rétt: 0.8360234
  return(list(model=lm.AIC, Radj=Radj.AICStep))
}
```