Introduction to Machine Learning - Lab 4

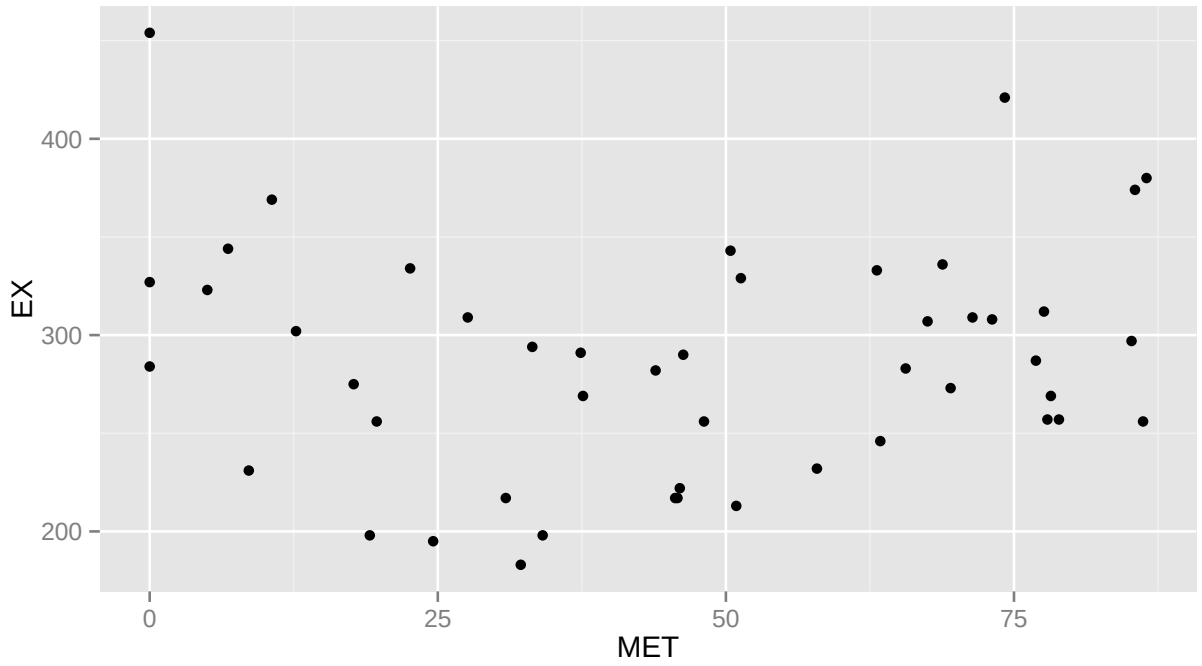
Gustav Sternelöv

Tuesday, November 17, 2015

Assignment 1

1.1

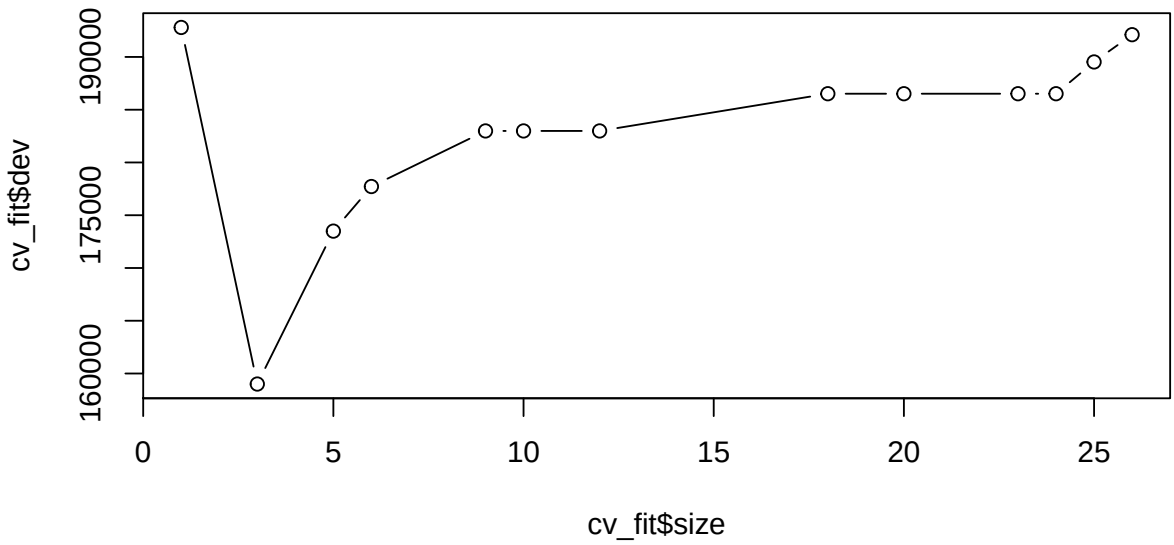
The variables analyzed in the first assignment are MET (Percentage of population living in standard metropolitan areas) and EX (Per capita state and local public expenditures ($)). The latter is the target variable and the former the input variable. A plot of MET versus EX can be seen below.



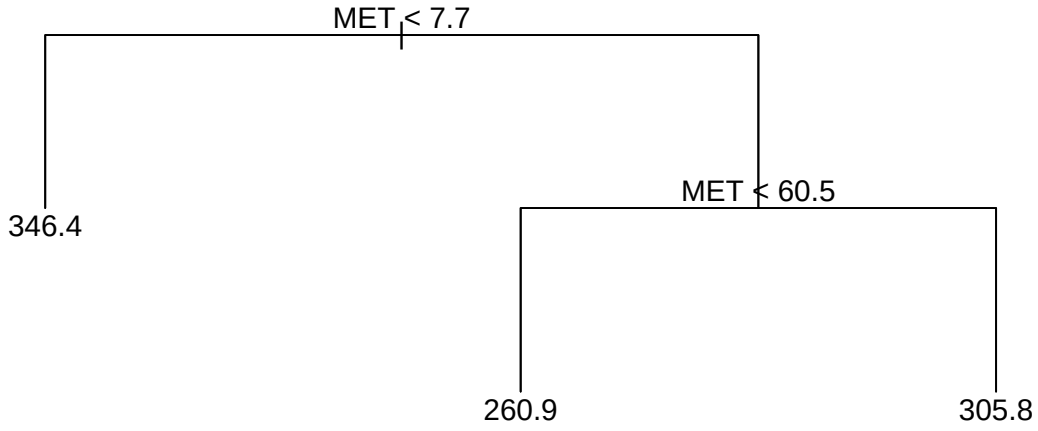
An analyze of the graph gives that for low values and for high values of MET the value of EX is high. An interpretation of this is that models who creates a linear decision boundary probably will result in bad fits. Another type of model, one that can capture a nonlinear pattern is therefore thought to be needed in this case.

1.2

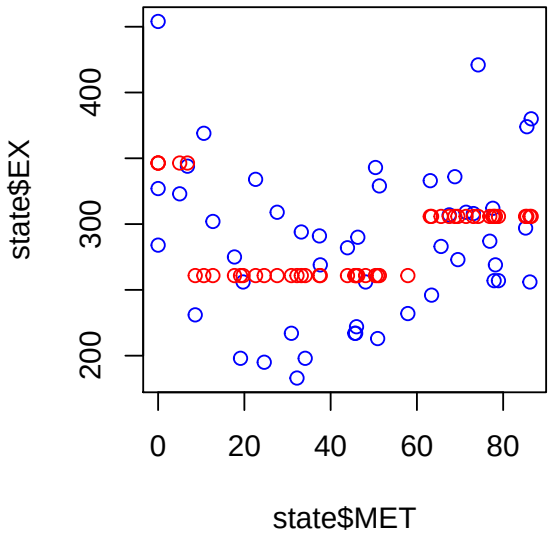
A regression tree model is fitted to the data descirbed in 1.1. The number of leaves are selected by cross-validation and the CV-score for different number of leaves is visualised by the following graph.

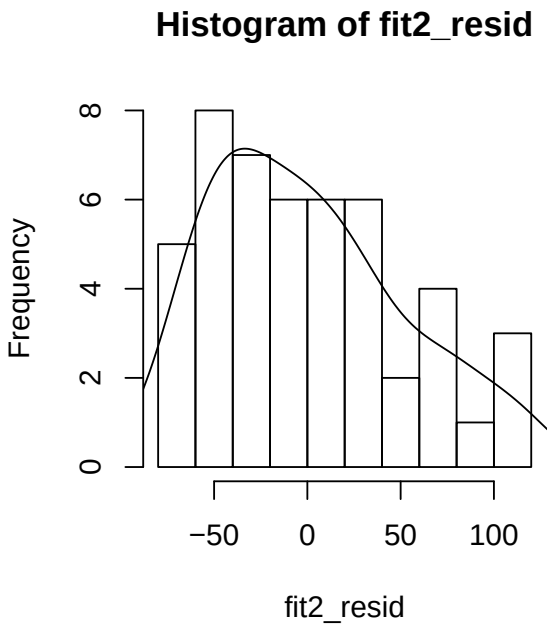


The lowest CV-score is given when three leaves are selected. The original tree is therefore pruned until it only contains three leaves. This pruning of the original tree results in the following regression tree.



As can be seen by the visualisation of the tree above, low values of MET results in a high response value. The second split is for high values where a rather high response value is given for high values of MET. For values in between, higher than 7.7 and lower than 60.5, a lower response value is given.
The original data is then plotted against the fitted values and the residuals are plotted in a histogram.



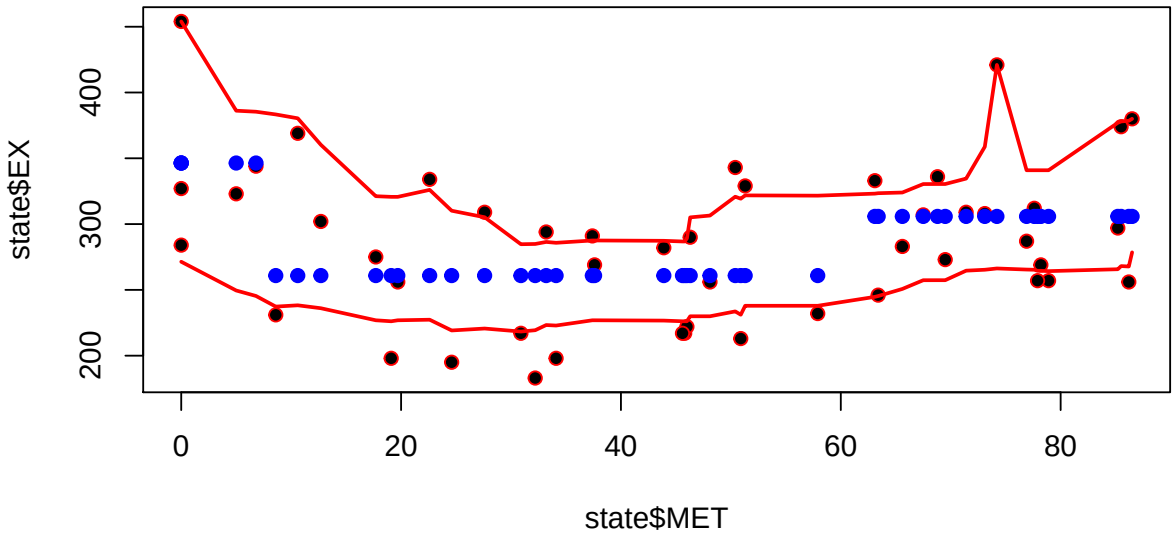


That the model seem to be quite a bad fit is clearly shown by the plot over the fitted values against the original values. Since the variance for the predicted values is low and the bias is high the model is concluded to be underfitted and quite some information seem to be lost when only returning three different values.

Since the data set has rather few observations the distribution of the residuals is a bit hard to analyse. What that can be said is that is not evident that the residuals are normally distributed.

1.3

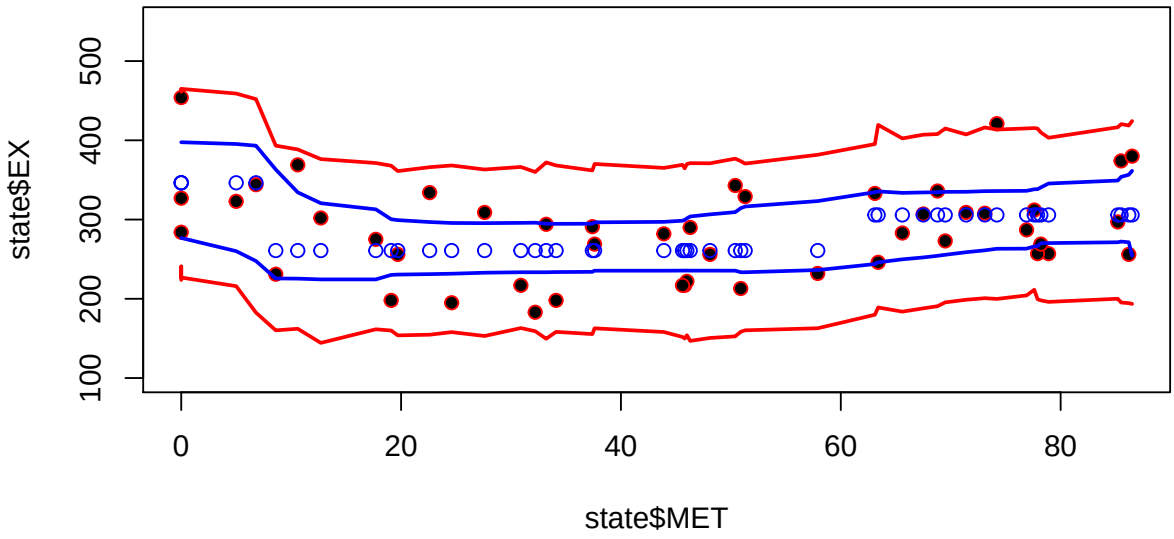
For the selected regression tree model is 95 % confidence bands computed by using a non-parametric bootstrap. The confidence bands are plotted together with both the original and fitted values.



The confidence bands are bumpy, that is because non-parametric bootstrapping do not operate under the assumption of belonging to any particluar distribution. This results in bands that takes much influence from the actual values and the effect of this are "bumpy" confidence bands.
In general the conclusion is that little knowledge about the expected response is gained by the model. That is true especially for the lowest values on the y-axis.

1.4

When using a parametric bootstrap to compute the 95 % confidence and prediction bands for the regression tree model, the following bands can be plotted:



For this data the fitted model seem to be an inappropriate choice. There is a lot of uncertainty in the data and this might be an explanation of the inappropriateness of the model. This problem is well illustrated by the prediction bands, since they capture both the expected interval for the observations and the potential effects of randomness.

Only one point are outside the prediction bands, that is approximately 5 % of data.

1.5

The estimated confidence bands with the non-parametric and the parametric bootstrap are quite similar. The former bands are more bumpy and the latter more stable. An advantage with the parametric bootstrap is that the confidence intervals are more stable and not too sensitive to extreme values as the confidence interval for the non-parametric bootstrap is.

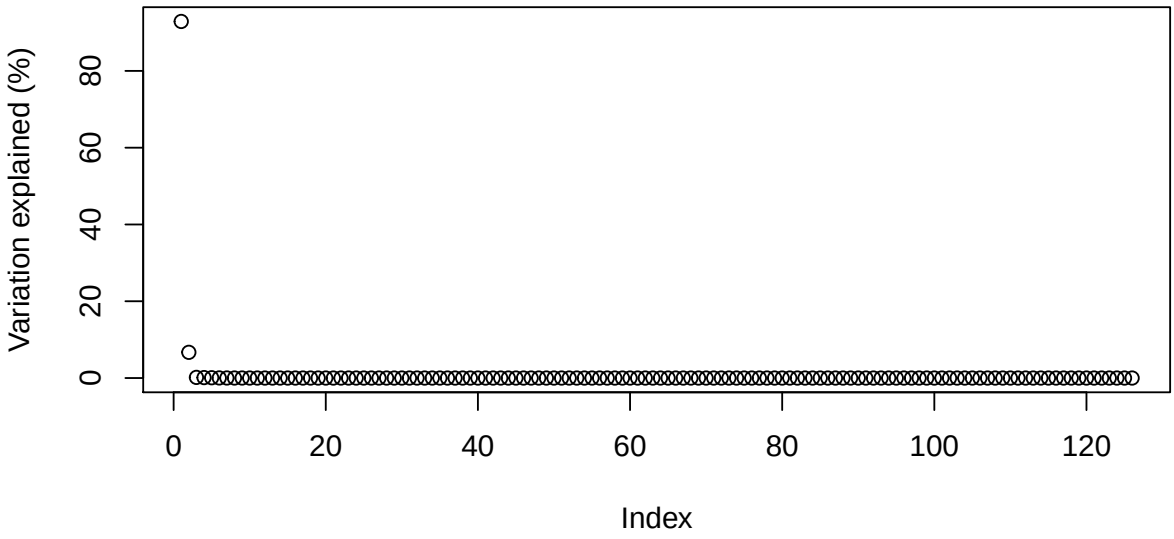
An suggestion may be that the parametric bootstrap is more apprpriate due to this advantage. Although a disadvantage may be the assumption of normality. As noted in 1.2 there might be uncertainty over making that assumption. If one of the methods has to be chosen, the parametric bootstrap is thought to be more appropriate since it gives more general and stable intervals.

Assignment 2

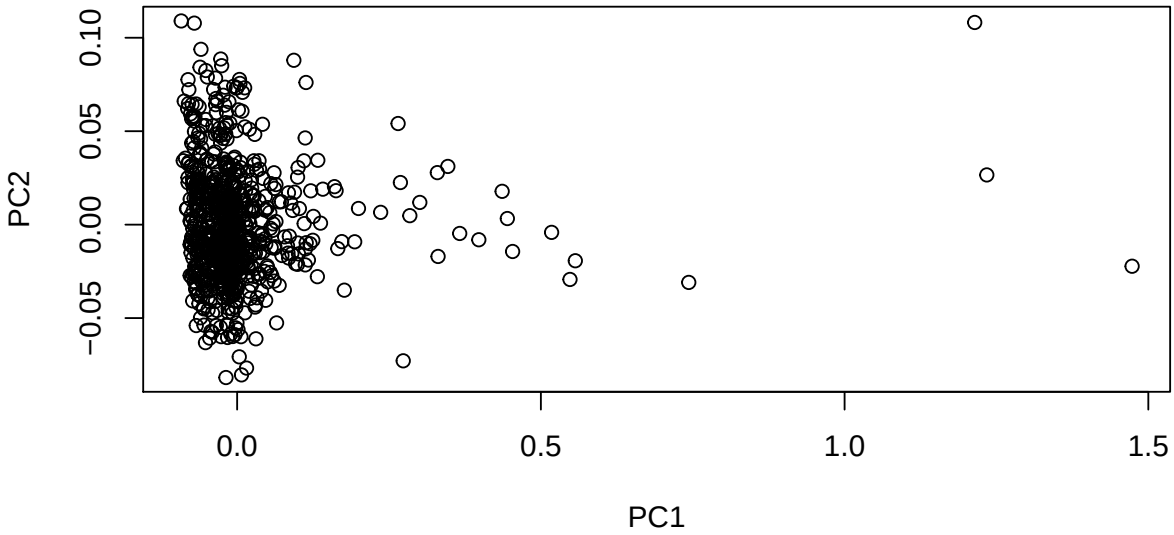
The data analysed in the second assignment consists of near-infrared spectra and viscosity levels for a collection of diesel fuels. The objective with the assignment is to examine how the measured spectra can be used in order to predict the viscosity.

2.1

A standard PCA with all features included is performed to determine how many principal components that are needed to explain at least 99% of the total variance. How much of the variation that is explained by each feature is presented with the following graph.



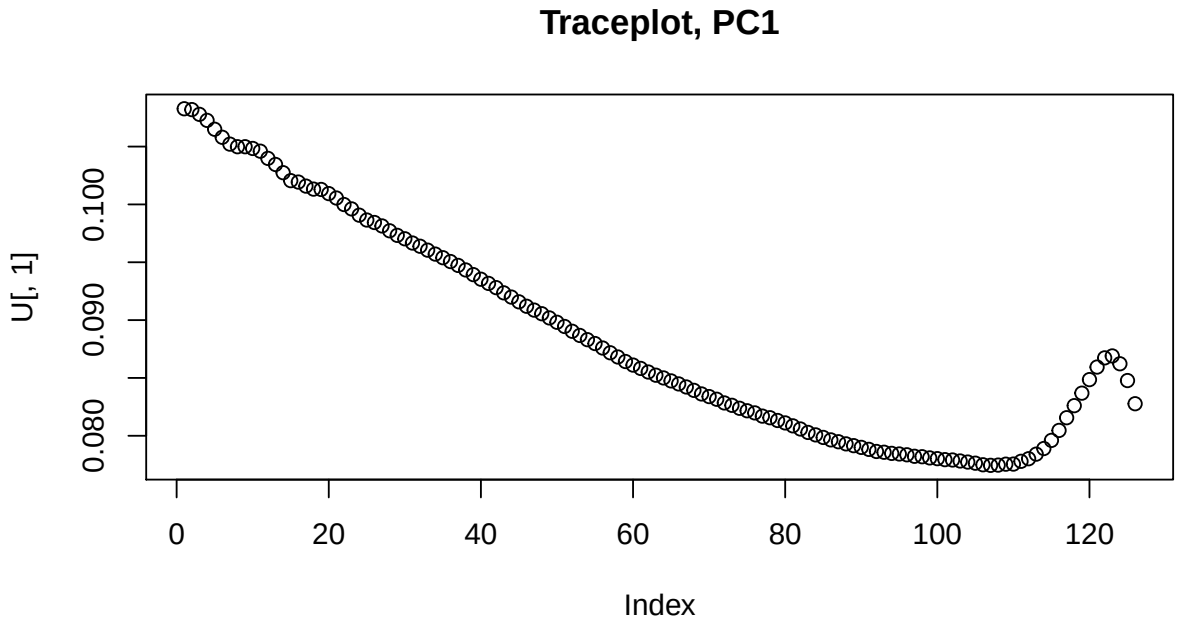
The amount of principal components needed to explain at least 99% of the total variation is concluded to be two. The scores for the respective feature for these principal components are investigated more closely by the next plot.

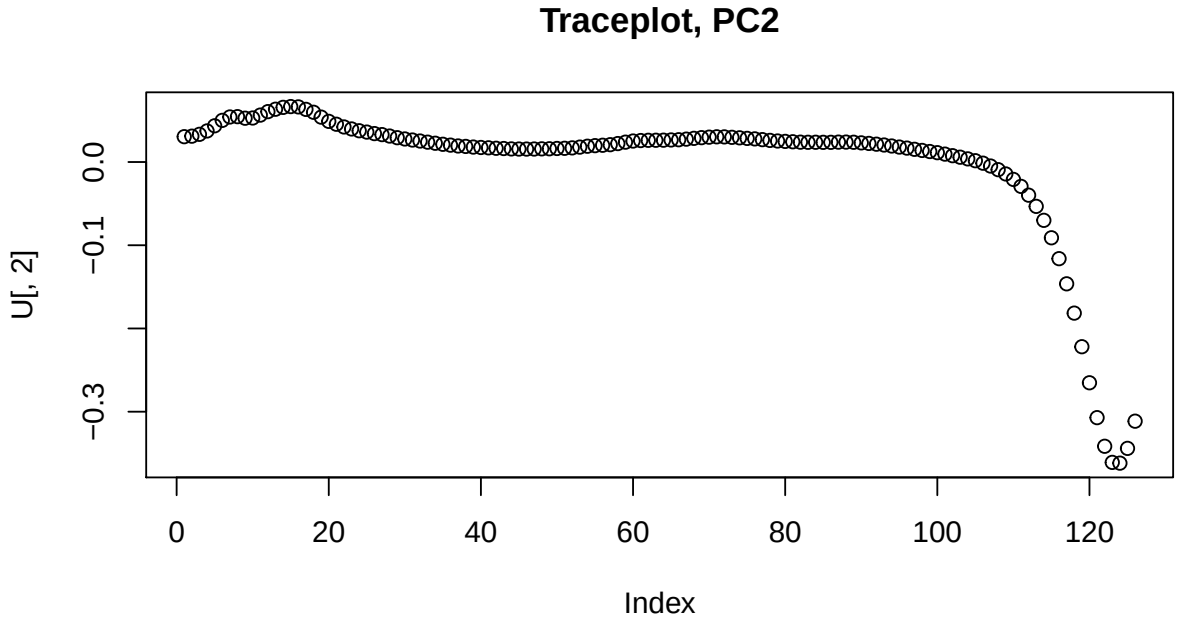


A cluster of points can be seen to the left in the graph and some outlying points at the right end of the graph. The outlying values x-wise can be interpreted to be the features with high scores for PC1. The valeus with low or high values y-wise are features with high scores for PC2.

2.2

The loadings of the components PC1 and PC2 are here visualised by so called trace plots.





For the first principal component, PC1, it can be seen that all of the factors seem to have significant effect on the component. For the second principal component, PC2, the majority of the factors are very close to zero. This implies that PC2 mainly is explained by those few factors whose loadings not are zero or close to zero.

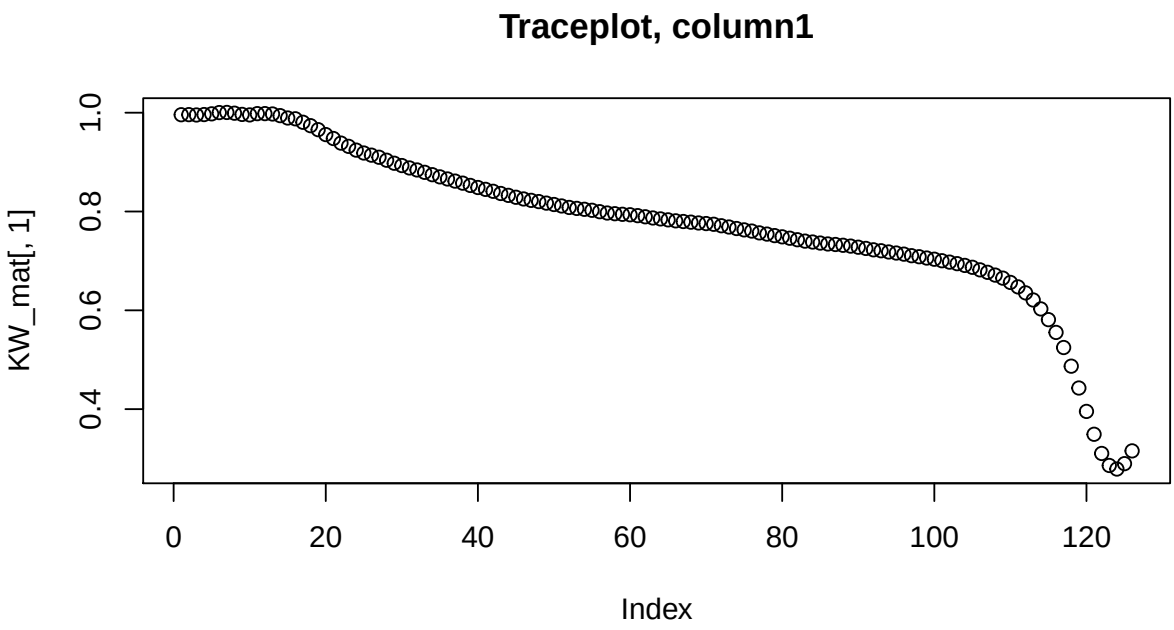
2.3

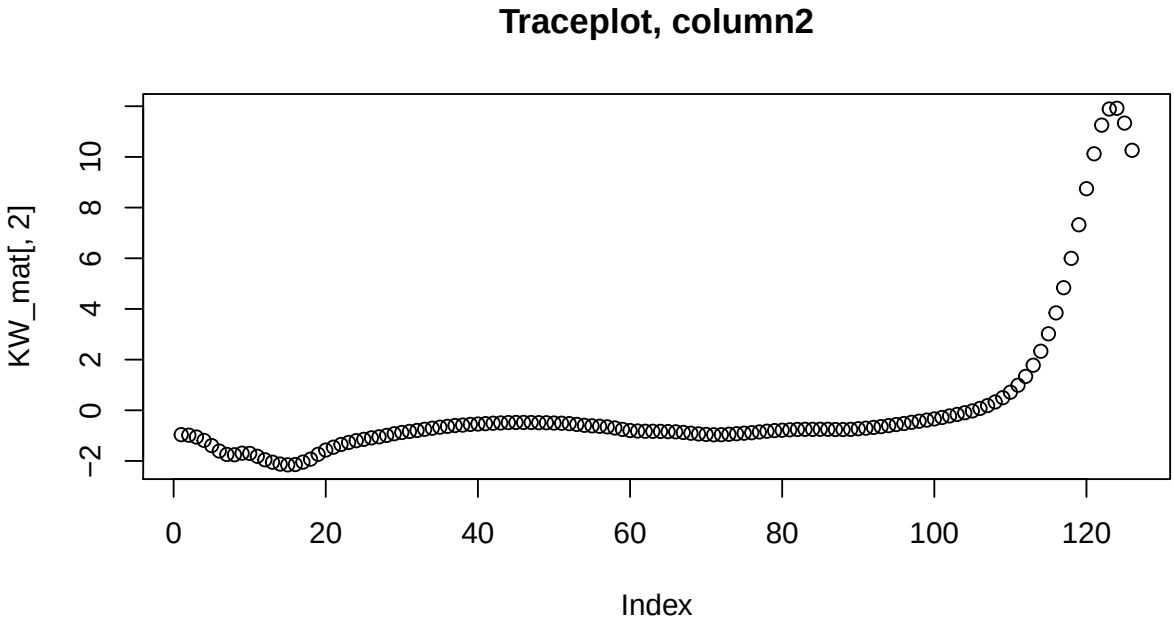
The next model that is conducted is a Independent Component Analysis, ICA, with two components.

a) The matrix W is presented by the table below. W is a weighting matrix that is used to minimize the mutual information between variables.This means that role the of the W matrix is to make the components in the model uncorrelated. An interpretation of the values in the matrix gives that the first component needs to be transformed quite much since the absolute value of first value in the matrix is high, almost equal to one. The second component is connected to a low value, almost equal to zero, and that means a small transformation is needed for this component.

```
##            [,1]       [,2]
## [1,] -0.9991790 -0.0405144
## [2,] -0.0405144  0.9991790
```

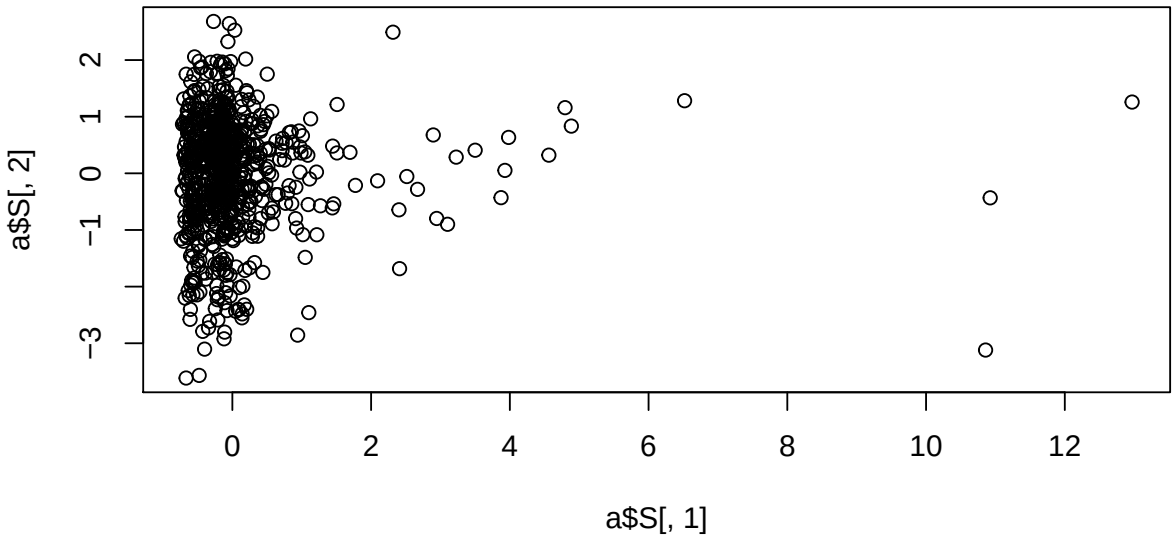
b) The next matrix presented is the W' matrix. The role of this matrix is to give the rotated, uncorrelated, loadings for the components of the ICA model. The two columns of the W' matrix are here presented by trace plots.





When comparing with the trace plots in 2.2 it can be noted that a similar conclusion can be drawn for the respecitive component. For the first component all factors have non-zero loadings. For the second component the majority of the factors have loadings close to zero and therefore only a few factors explains the component.

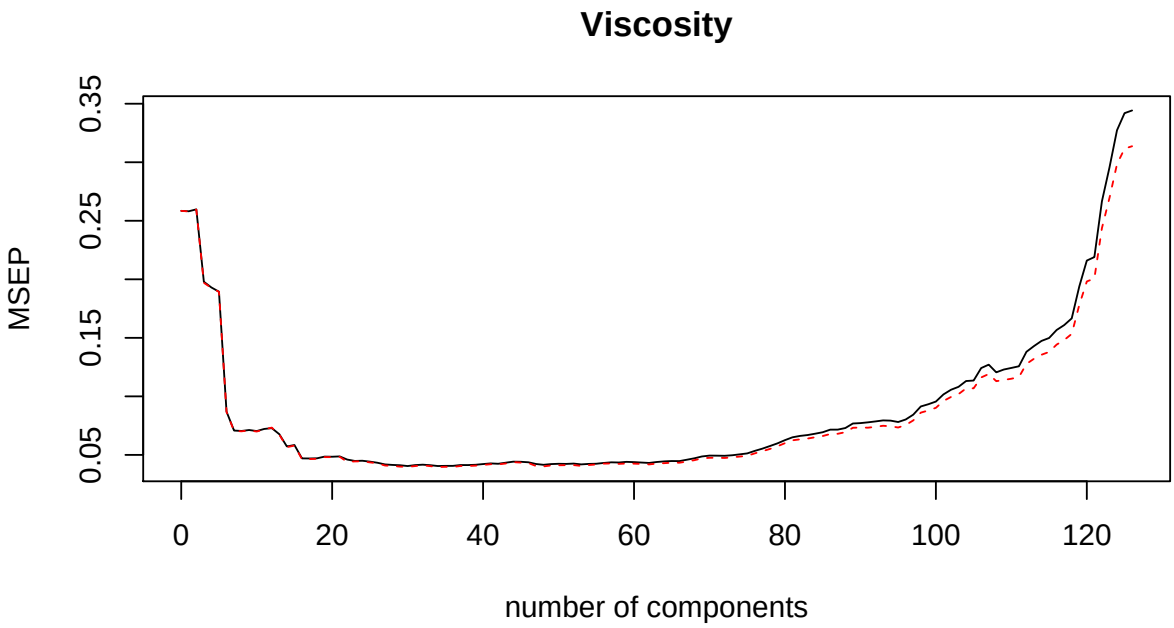
c) A score plot for the components in the ICA model



A comparsion with the score plot in 2.1 gives that the scores for the components in the ICA model is rather similar to those for the first two components of the standard PCA.

2.4 - 2.5

Before fitting a PCR and PLS model the data is divided into a training and a test set. The respective data set contains 50% each of the original data set.
A PCR model is fitted where the number of components in the model is selected by cross-validation. A low mean-square prediction error is wanted and the number of components that corresponds to the lowest error is chosen. To examine this further the dependence between the mean-square prediction error and the number of components is plotted.



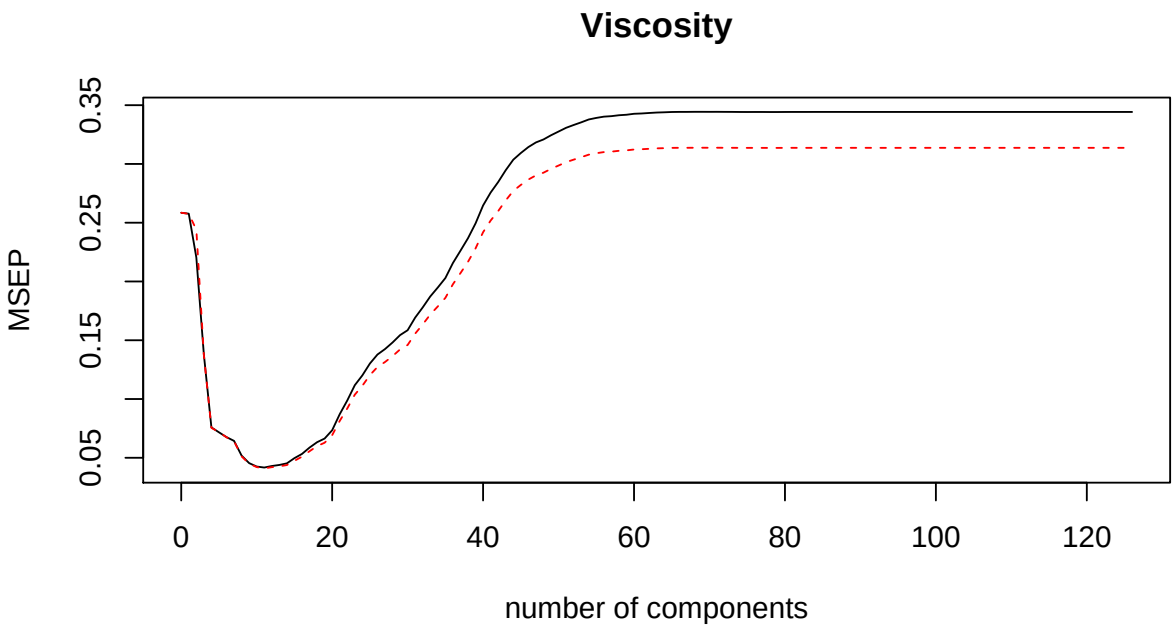
The value for the mean-square prediction error term is approximately the same for a quite longe range of components. For the number of components from about 25 up to around 60 it is a very slight difference between the mean-square prediction errors. Since there is no reason to include more components if they don't improve the results, a reasonable choice could be to select 25 components.

The test set is used to evaluate the selected, optimal, model with 25 components. This is done by calculating the mean-square error for the model when applied on the test set.

```
## [1] 0.06783242
```

2.6

The workflow for 2.5 is repeated in 2.6, with the difference that a PLS model is fitted instead of a PCR.



The lowest mean-square prediction error is given when the number of components is equal to 12. When the model is used on test data the mean-square error is calculated to be equal to 0.0673106. Compared to the mean-square error obtained for the PCR model so is the value slightly lower for the PLS model. The PLS model therefore seems to work equally well as the PCR model, even though it only uses 12 components compared to the 25 components used by the PCR model.

Appendix

R-code

```
state <- read.csv("C:/Users/Gustav/Documents/Machine-Learning/Lab 4/State.csv", sep=";")
library(ggplot2)
ggplot(state, aes(x=MET, y=EX)) + geom_point()
library(tree)
set.seed(12345)
fit2 <- tree(EX ~ MET, data=state, control=tree.control(nobs=48, minsize=2))
cv_fit <- cv.tree(fit2)
plot(cv_fit$size, cv_fit$dev, type="b")
pruneFit2 <- prune.tree(fit2, best=3)
plot(pruneFit2)
text(pruneFit2, pretty=0)
cv_pred <- predict(pruneFit2)
fit2_resid <- state$EX - cv_pred
myhist <- hist(fit2_resid)
par(mfrow=c(1,2))
plot(state$MET, state$EX, col="blue")
```

```
points(state$MET, cv_pred, col="red")

multiplier <- myhist$counts / myhist$density
mydensity <- density(fit2_resid)
mydensity$y <- mydensity$y * multiplier[1]

plot(myhist)
lines(mydensity)
par(mfrow=c(1,1))
# 1.3
# Non-parametric bootstrap
# 95 % confidence bands
library(boot)
data2=state[order(state$MET),]#reordering data according to MET
# computing bootstrap samples
f=function(data, ind){
  data1=data[ind,]# extract bootstrap sample
  #fit regression tree
  res=tree(EX ~ MET, data=data1, control=tree.control(nobs=48, minsize=2))
  fit=prune.tree(res, best=3)
  #predict values for all Area values from the original data
  priceP=predict(fit,newdata=data2)
  return(priceP)
}
res=boot(data2, f, R=1000) #make bootstrap

# Create lower and upper bound.
e=envelope(res)

fit=prune.tree(fit2, best=3)
priceP=predict(fit)

plot(state$MET, state$EX, pch=21, bg="black", col="red")
points(state$MET,priceP, col="blue", bg="blue", pch=21) #plot fitted line
#plot cofidence bands
points(data2$MET,e$point[2,], type="l", col="red", lwd=2)
points(data2$MET,e$point[1,], type="l", col="red", lwd=2)

# Parametric bootstrap
# 95 % confidence and prediction bands
# Assumes that Y follows the normal distribution
mle=prune.tree(fit2, best=3)
rng=function(data, mle) {
  data1=data.frame(EX=data$EX,
                   MET=data$MET, data=data)
  n=length(data$EX)
  data1$EX=rnorm(n,predict(mle,newdata=data1),sd(residuals(mle)))
  return(data1)
}
# Prediction bands
f1=function(data1){
  res=tree(EX ~ MET, data=data1, control=tree.control(nobs=48, minsize=2))
  fit=prune.tree(res, best=3)
```

```
  n=length(data1$EX)
  predictsP=predict(fit, newdata=data2)
  predictedP=rnorm(n,predictsP,sd(residuals(mle)))
  return(predictedP)
}
# Confidence bands
f2=function(data1){
  res=tree(EX ~ MET, data=data1, control=tree.control(nobs=48, minsize=2))
  fit=prune.tree(res, best=3)
  predictedP=predict(fit, newdata=data2)
  return(predictedP)
}


res1=boot(data2, statistic=f1, R=1000, mle=mle, ran.gen=rng , sim="parametric")
res2=boot(data2, statistic=f2, R=1000, mle=mle, ran.gen=rng , sim="parametric")

e2=envelope(res1)
e3=envelope(res2)

fit=prune.tree(fit2, best=3)
pred2=predict(fit)
plot(state$MET, state$EX, pch=21, bg="black", col="red", ylim=c(100, 550))
points(state$MET,pred2, col="blue") #plot fitted line
#plot cofidence bands
points(data2$MET,e2$point[2,], type="l", col="red", lwd=2)
points(data2$MET,e2$point[1,], type="l", col="red", lwd=2)
points(data2$MET,e3$point[2,], type="l", col="blue", lwd=2)
points(data2$MET,e3$point[1,], type="l", col="blue", lwd=2)
spectra <- read.csv("C:/Users/Gustav/Documents/Machine-Learning/Lab 4/NIRSpectra.csv", sep=";")
data_a <- spectra
data_a$Viscosity=c()
data_a$ID=c()
res=prcomp(data_a)
lambda=res$sdev^2
#proportion of variation explained by each feature
plot(sprintf("%2.3f",lambda/sum(lambda)*100), ylab="Variation explained (%)")
# Scores in coordinates of PC1 and PC2
plot(res$x[,1], res$x[,2], xlab="PC1", ylab="PC2")
U=res$rotation
plot(U[,1], main="Traceplot, PC1")
plot(U[,2],main="Traceplot, PC2")

library(fastICA)

set.seed(12345)
a <- fastICA(data_a, 2, alg.typ = "parallel", fun = "logcosh", alpha = 1,
             method = "R", row.norm = FALSE, maxit = 200, tol = 0.0001, verbose = TRUE) #ICA

W_mat <- a$W
W_mat
KW_mat <- a$K %*% a$W
# plot the columns as trace plots
```

```r
plot(KW_mat[,1], main="Traceplot, column1")
plot(KW_mat[,2],main="Traceplot, column2")
# The score plot
plot(a$S[,1], a$S[,2])
dat <- as.matrix(spectra)
n=dim(dat)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=dat[id,]
test=dat[-id,]

library(pls)
train <- data.frame(train[, 2:128])
test <- data.frame(test[, 2:128])
set.seed(12345)
pcr.fit=pcr(Viscosity~., data=train, validation="CV")
validationplot(pcr.fit,val.type="MSEP")

pcr.fit1=pcr(Viscosity~., 25,data=train, validation="none")
pcr.pred <- predict(pcr.fit1, newdata=test, ncomp = 25)
pcr.mse <- 1/length(na.omit(test$Viscosity)) * sum((test$Viscosity - pcr.pred)^2, na.rm=TRUE)
pcr.mse
set.seed(12345)
plsr.fit=plsr(Viscosity~., data=train, validation="CV")
validationplot(plsr.fit,val.type="MSEP")
plsr.fit1=plsr(Viscosity~., 12,data=train, validation="none")
plsr.pred <- predict(plsr.fit1, newdata=test, ncomp = 12)
plsr.mse <- 1/length(na.omit(test$Viscosity))* sum((test$Viscosity - plsr.pred)^2, na.rm=TRUE)
##
```