Lab 4 732A95 Introduction to machine learning

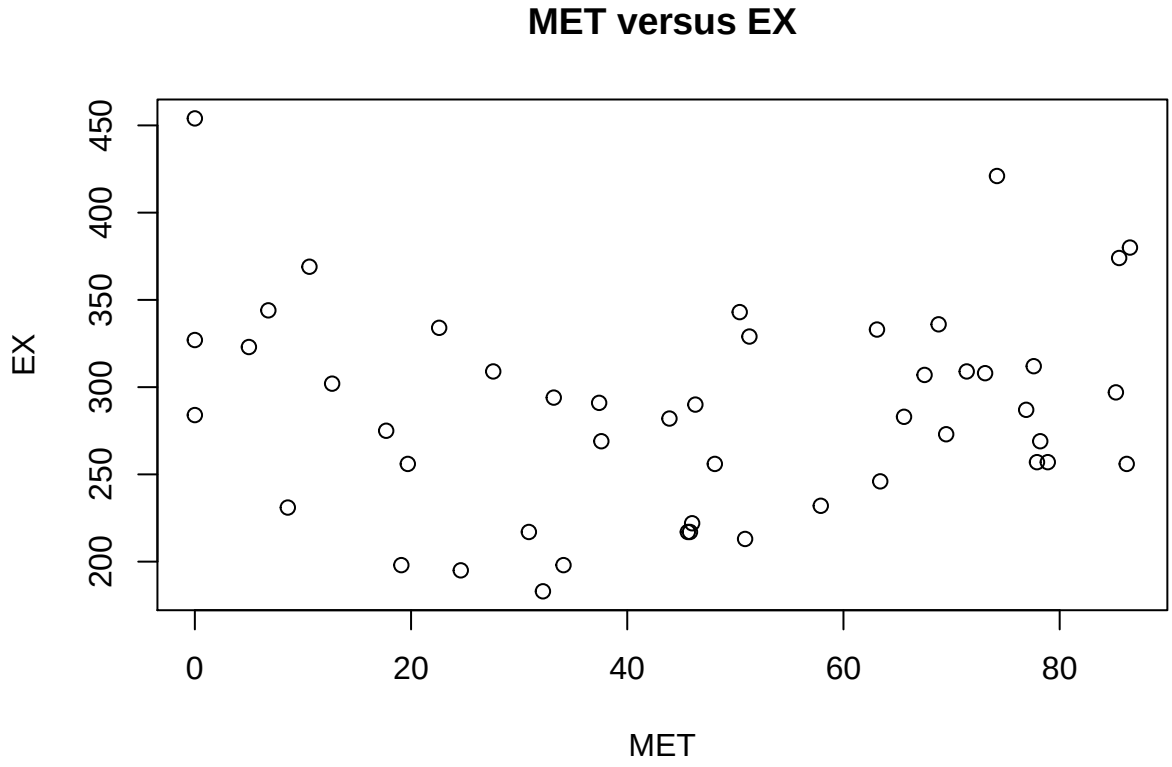
Anton Persson antpe404

30 november 2016

Assignment 1 Uncertainity estimation

Assignment 1.1

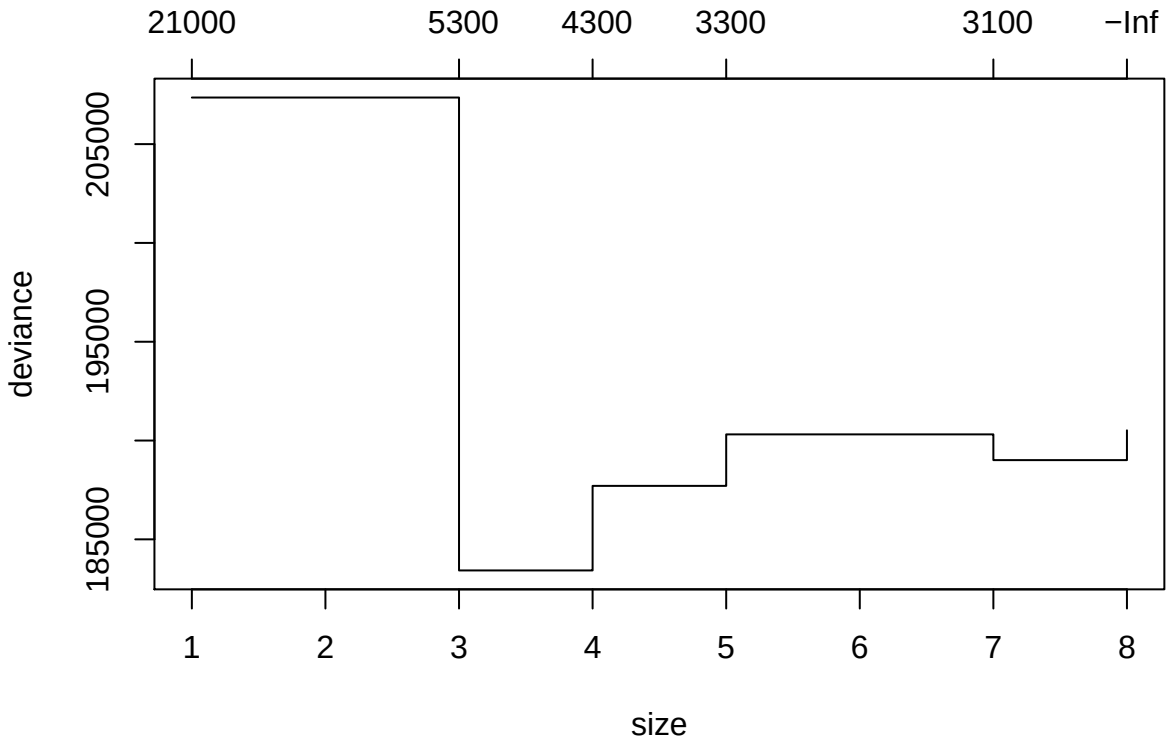
The requested plot of variable *EX* versus *MET* is shown below.



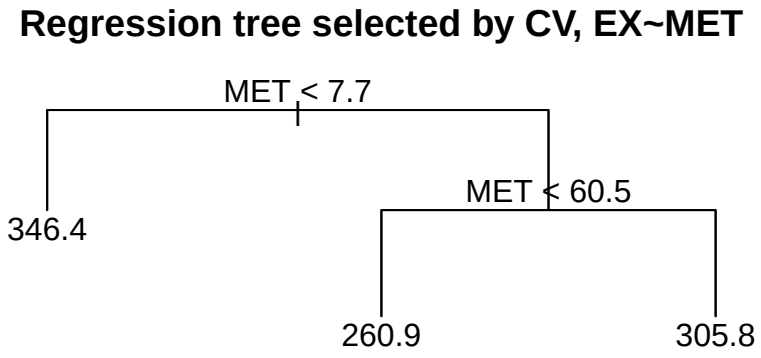
I do not find any total obvious modelation for the data shown above. A cubic spline might work out okey. I piecewise linear model with one knot would maybe be decent as well.

Assignment 1.2

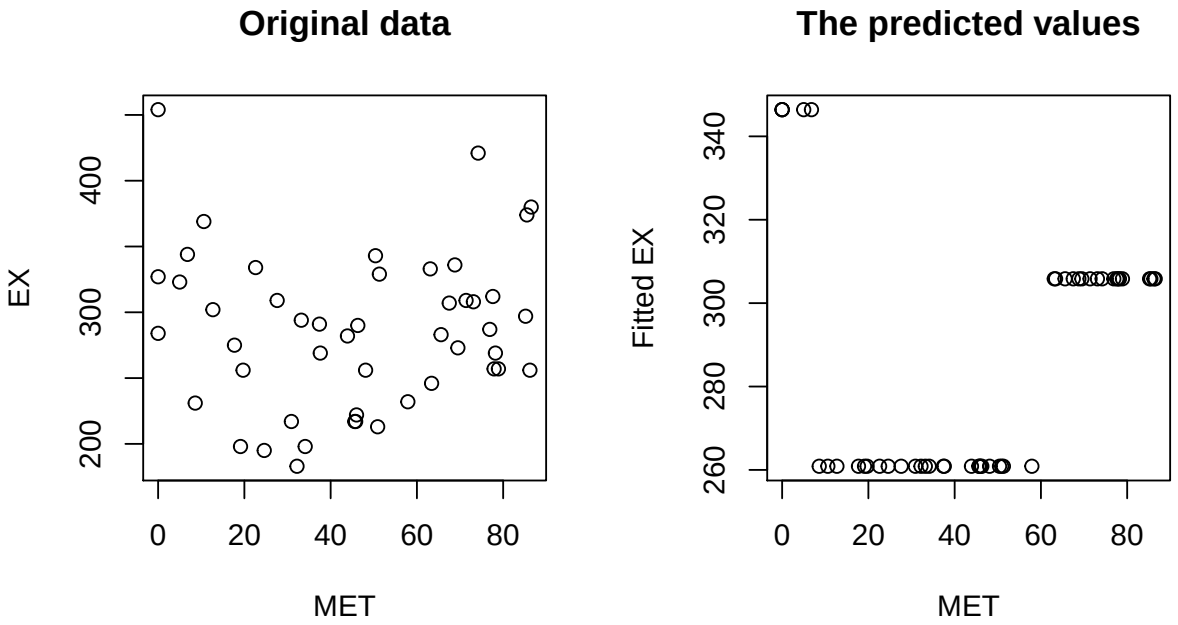
I set the options as in the instruction and ran the tree. Since the task was about to select the number of leaves by cross validation, I start off by showing the result of that procedure.



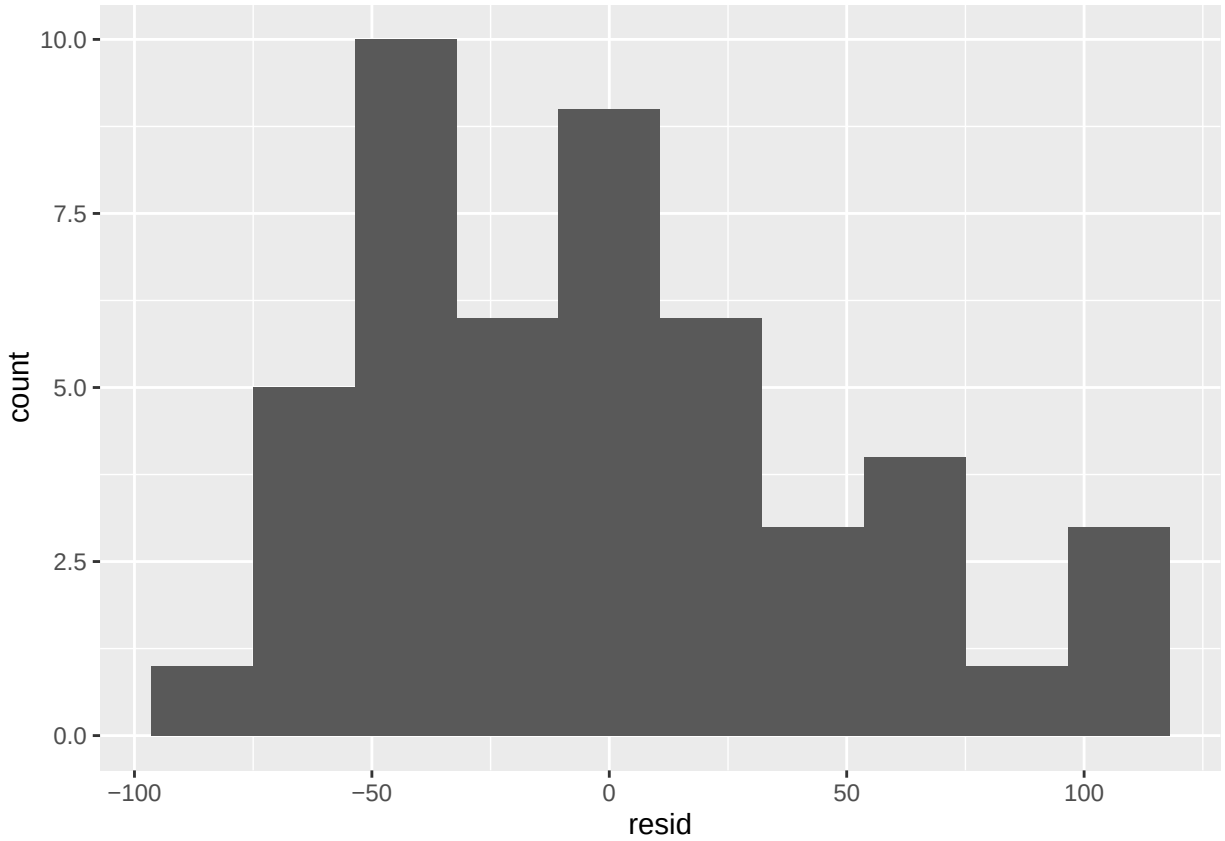
From the plot above I do conclude that three or four leaves seems to minimize the deviance. Since it's reasonable to also minimize the complexity, I therefore decide to select a tree with three leaves. The tree is shown below.



The tree above has three leaves, depending on the levels of *MET* on the points 7.7 and 60.5. The original data and the fitted data are plotted below.



The quality of the fit obviously doesn't look super good, which is intuitive just by looking at the original data. But at least I can see the pattern in the prediction the with high values to the left in the plot, lower in the middle and medium to the right. It's a underfit model however, its' confident bands wouldn't look good. The histogram of the residuals is presented below.

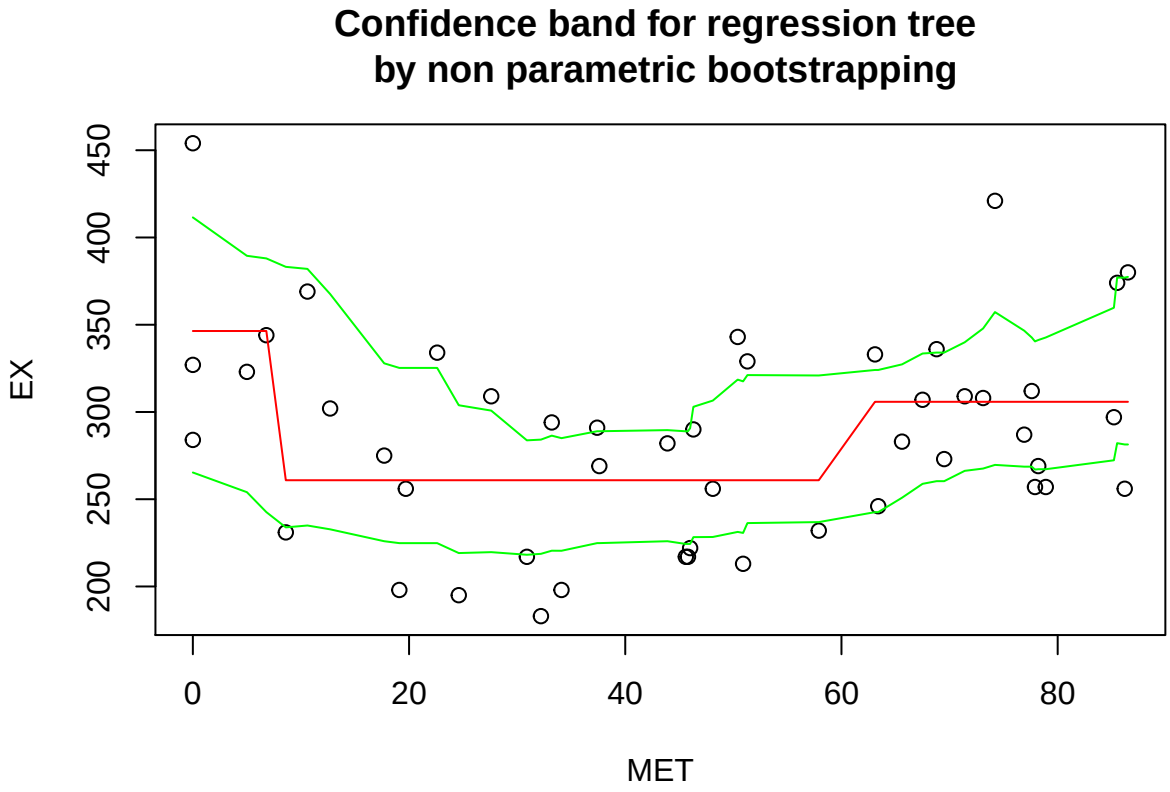


From the histogram above it's hard to tell whether the residuals are normally distributed or not. There are very few observations, which means that I can't expect to see a clear distribution anyhow. It could be a normal distribution, or maybe some gamma distribution.

If I compare the scatterplots above, I can see where the big residuals are coming from. Not all observation with MET values between 8 and 60 has low EX values for example, and not all observations with low values of MET has high EX.

Assignment 1.3

I used the lectures from this course and some code chunks from the slides to come up with a non parametric bootstrap method. The result, i.e. the plot with the fitted values and the confidence bands, is presented below.

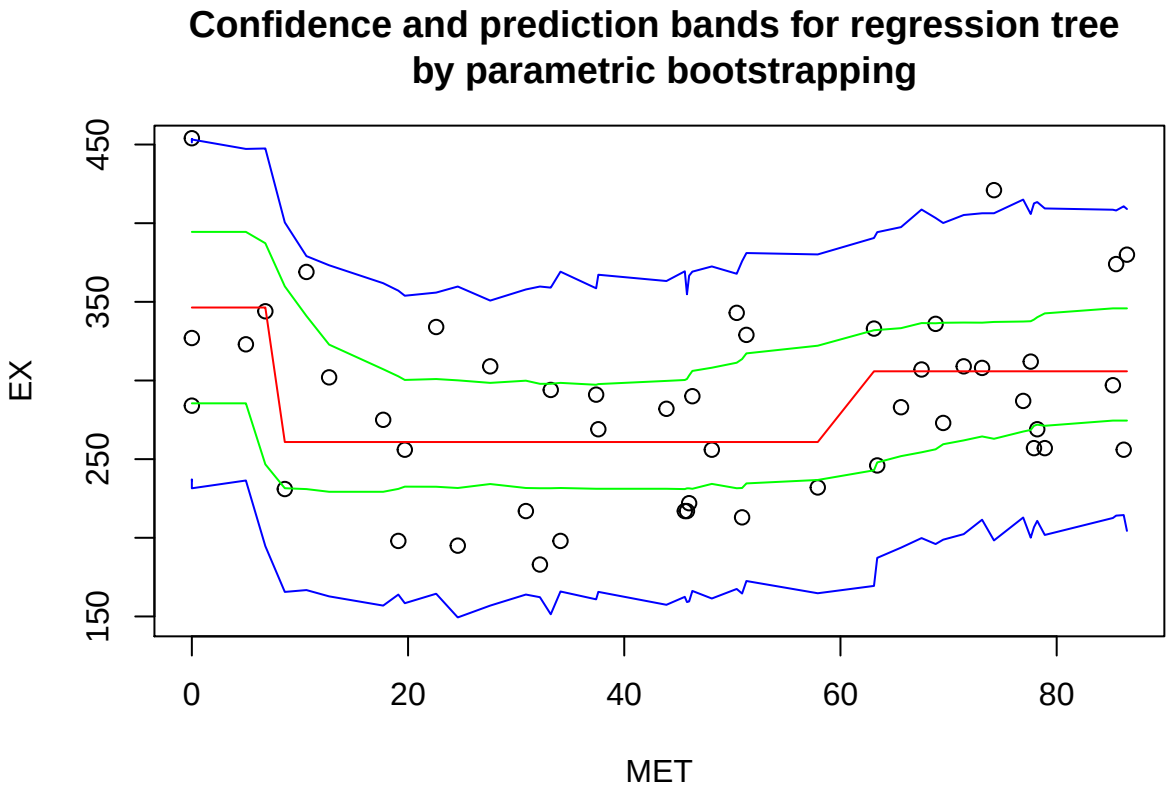


The band is quite bumpy. That is probably because some of the big outliers doesn't appear in all models created by bootstraping. Most of the trees seem to fit observations with MET [30,50] in the same way, that's why it has a more narrow band there, while it's much broader for example in the beginning of the series where we've got one obvious outlier.

The result from assignment 1.2, which is the red line in the plot above, doesn't seem very reliable. The confidant band is quite broad, and the confident band is supposed to show the true value of EX given MET. It's not supposed to cover 95 % of the observations.

Assignment 1.4

I did the parametric bootstrapping procedure to get both the confidence- and prediction band by using the code shells available in the slide for the course. The results for the parametric bootstrap are presented below.



In the plot above, the green lines represent the confidence band and the blue lines represents the prediction bands. The confidence bands is clearly more narrow here, when I've used parametric bootstraping. Regarding this plot, the regression tree in assignment 2 (still red line in the plot above) looks more reliable. It does look like two of the observations aren't covered by the prediction band. That's as close as I can come to 5 % of the data, since the dataset only consists of 48 observations. The predicition band is practically saying that a new observation with a given MET value will be within this span, why it's reasonable to find out that two observations are outside in this case. If all observations would be inside the prediction band by a wide margin, or ten observation would be outside, I couldn't say it's defintely wrong but it would be worrying.

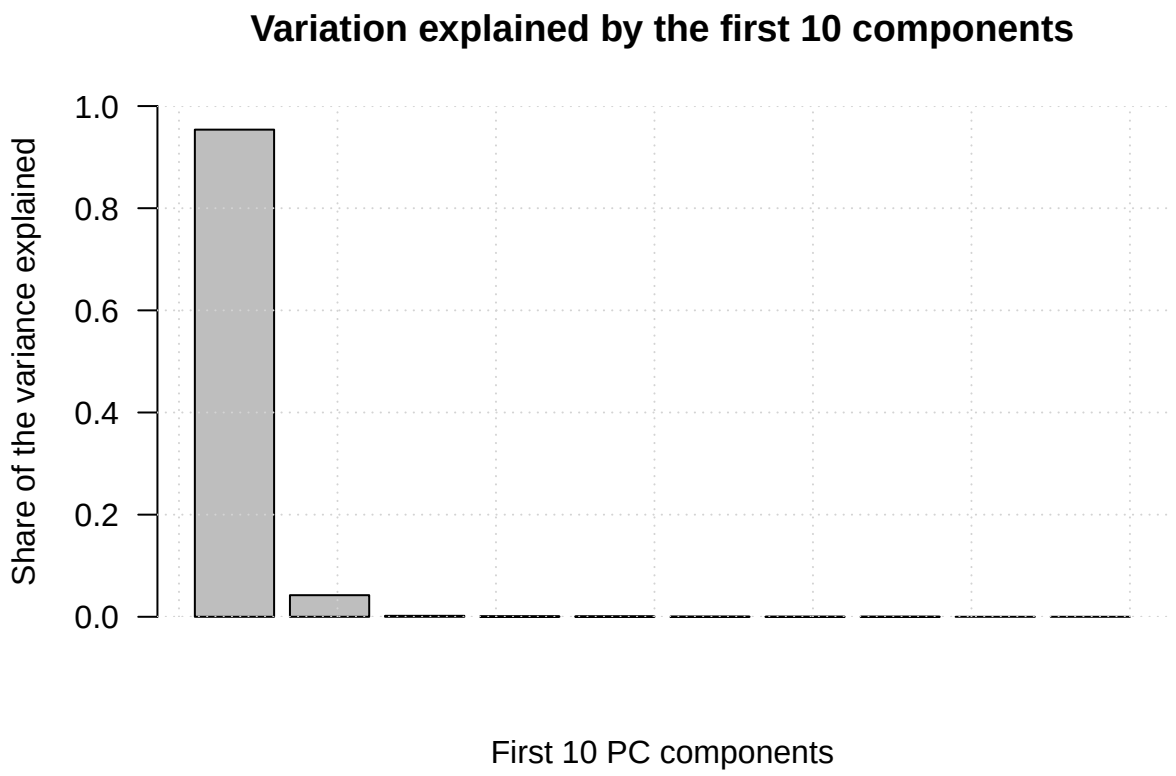
Assignment 1.5

Parametric bootstrapping has the advantage of working even for small samples, which I do have in this case. The non parametric bootstraping method doesn't require that I can tell which distribution I'm working with. When I did the parametric boorstrapping in assignment 1.4 I assumed $Y \sim \mathcal{N}(\mu_i, \sigma^2)$ but as I commented on the histogram in assignment 1.2, I'n not sure of that. The parametric bootstrap is shown to be more accurate if the assumed (or known of course) distribution is correct. In my case the parametric bootstrapping in 1.4 gives better results, which I suppose might indicate that I did the right thing to say that the residuals shown in the histogram in assignment 1.2 might come from a normal distribution. Overall I'd say that the parametric bootstraping, i.e. what's done in assignment 1.4, is more appropriate in this case.

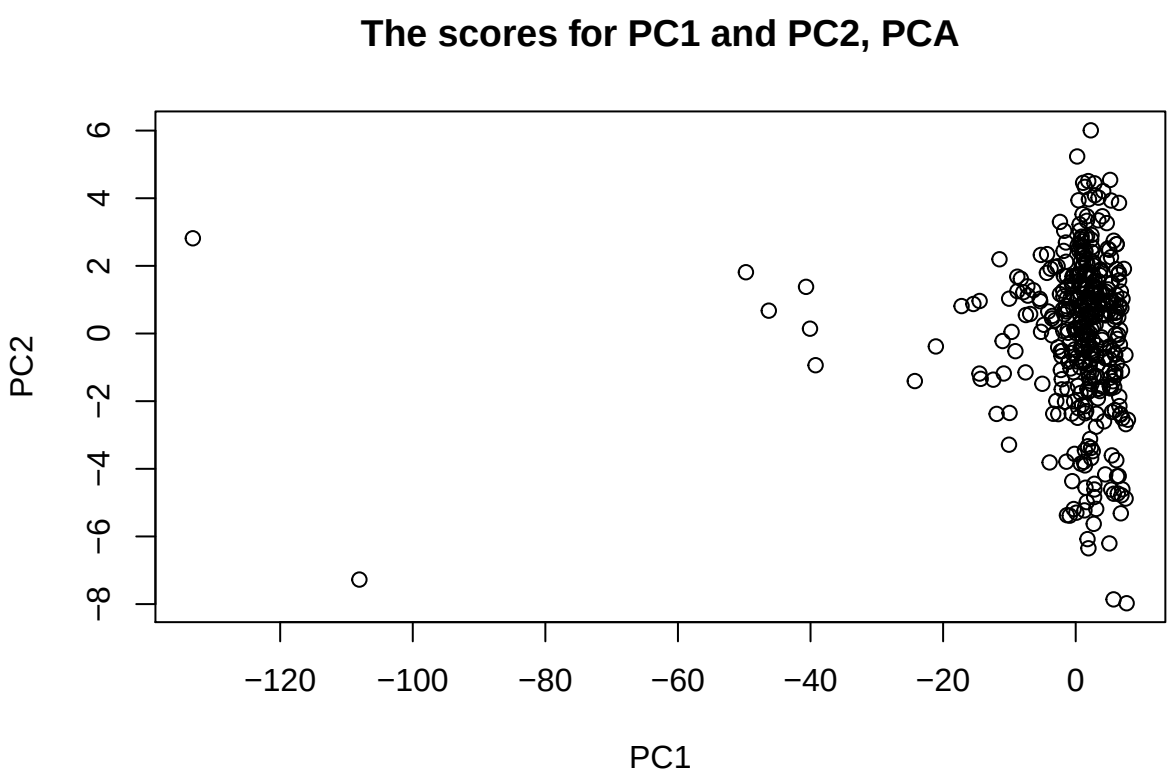
Assignment 2 Principal components

Assignment 2.1

I didn't use the prcomp() function but did the calcualtions myself. The plot explaining how much of the variation the first ten principal components explains is shown below.



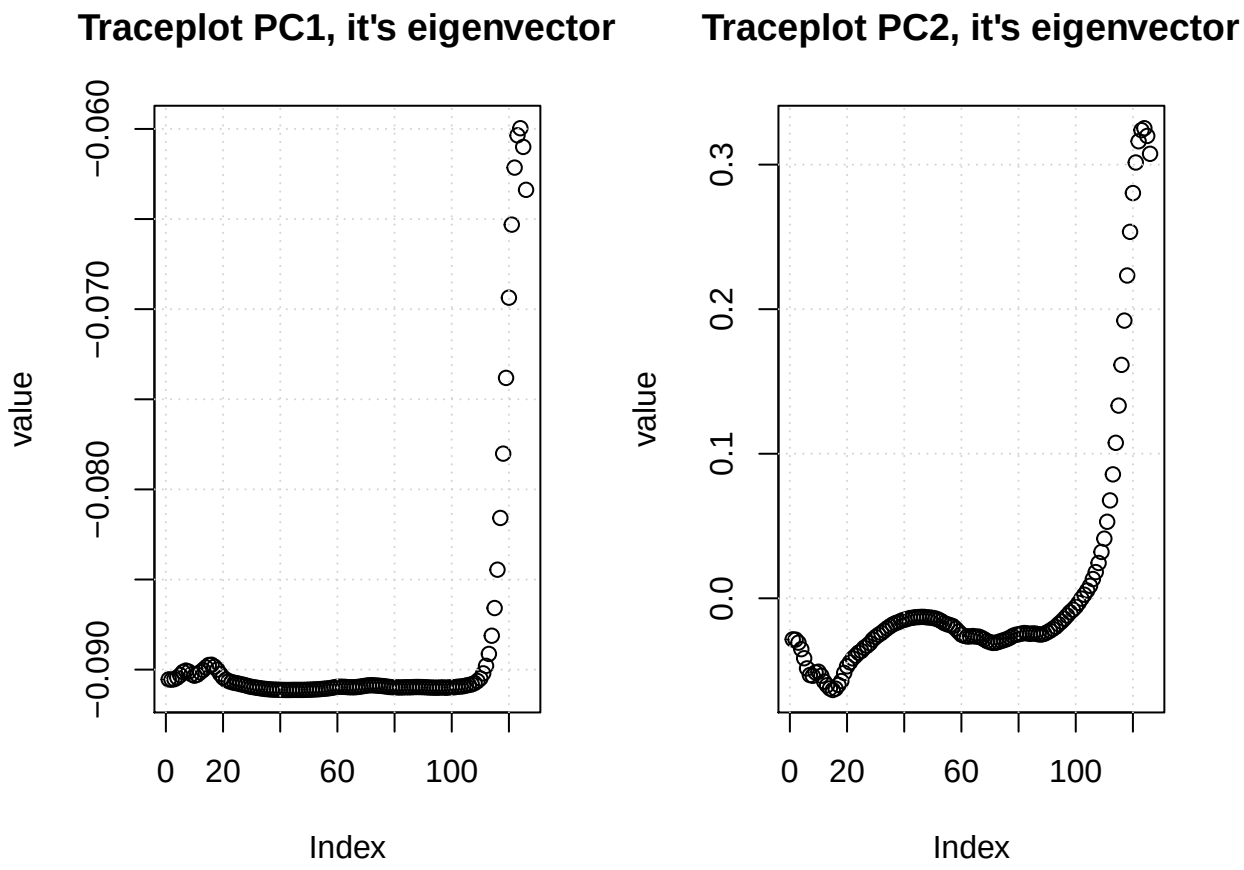
The plot above tells me that the first component explains very much, about 95 %. The plot tells me that one or two PC:s should be extracted, depending on how much variance I need to explain. The instructions tell me to explain at least 99 %, which means that I choose 2 components. That is, the first two bars in the plot above explains at least 99 %. The requested plot of the scores for the two PC:s is shown below.



The plot of the scores for PC1 and PC2 shows that there are a few observations of unusual diesel fuels. There are two major outliers accorinding to PC1 especially. Those two seems to be more normal according to PC2, even though the observatin on the left bottom of the plot can be seen as an outlier in both PCs. Yes, it seems to a few unusual fuels, escpecially on the PC1.

Assignment 2.2

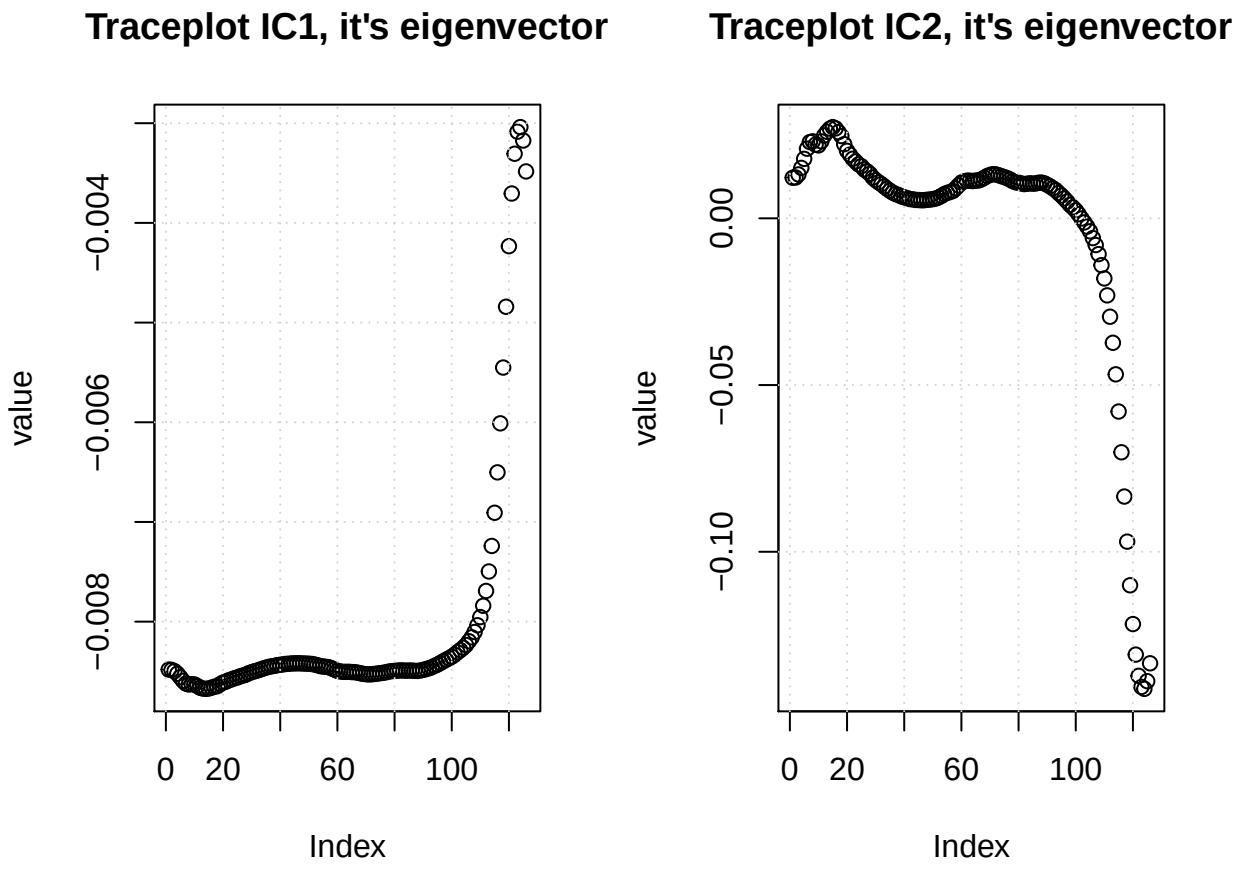
The trace plots for the two selected PCs are shown below.



From the left plot above, the loadings for PC1, I conclude that all original variables contribute to PC1. That can be seen since all 126 variable indexes on the X axis has a value separated from zero. For PC2 on the other side, not all original variables contributes to explain PC2. PC2 seems to be explained mainly by the first 25 and the last 20 original variables, i.e. those who are separated from 0 in the plot.

Assignment 2.3a

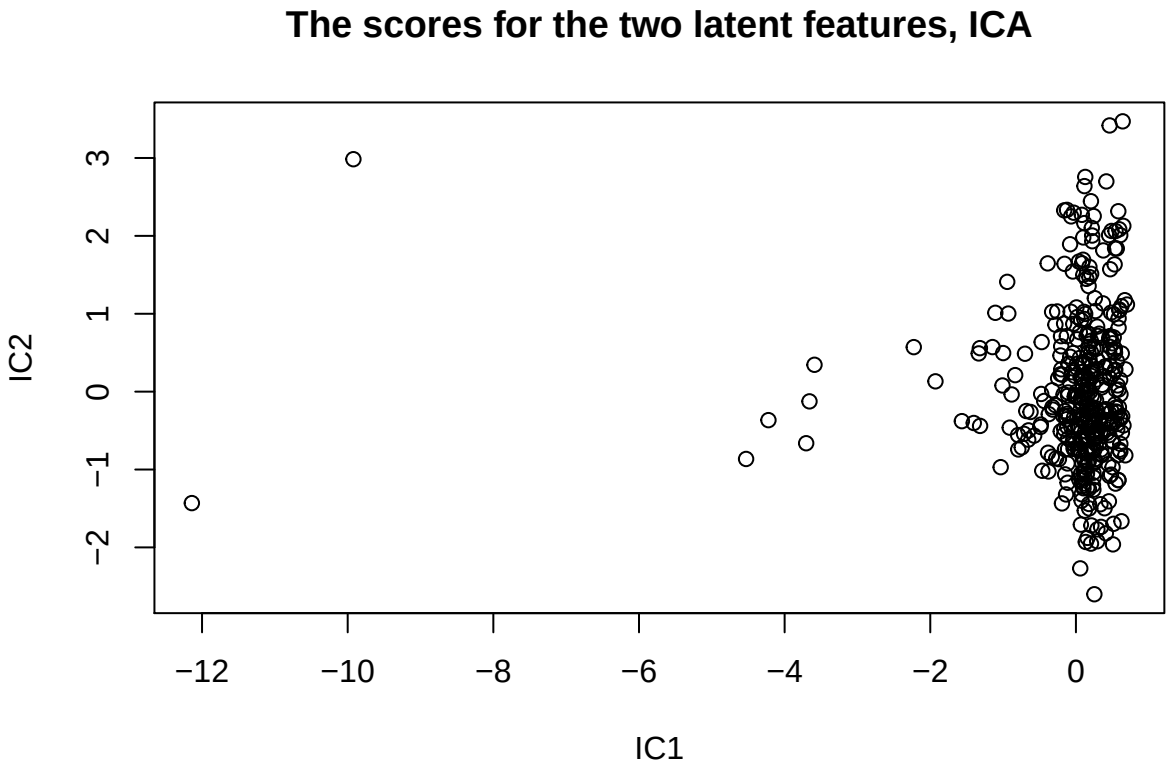
I ran the fastICA procedure and did the calculations as instructed, and the resulting traceplots is presented below. To campare them to the PCA-traceplots I decided to plot all four plots next to each other.



The traceplots above for the fastICA model look very much the same as the traceplots for the two principal components. I think that W' is the loadings of the ICA-model, representing the same thing as the eigenvectors in the U matrix in PCA.

Assignment 2.3b

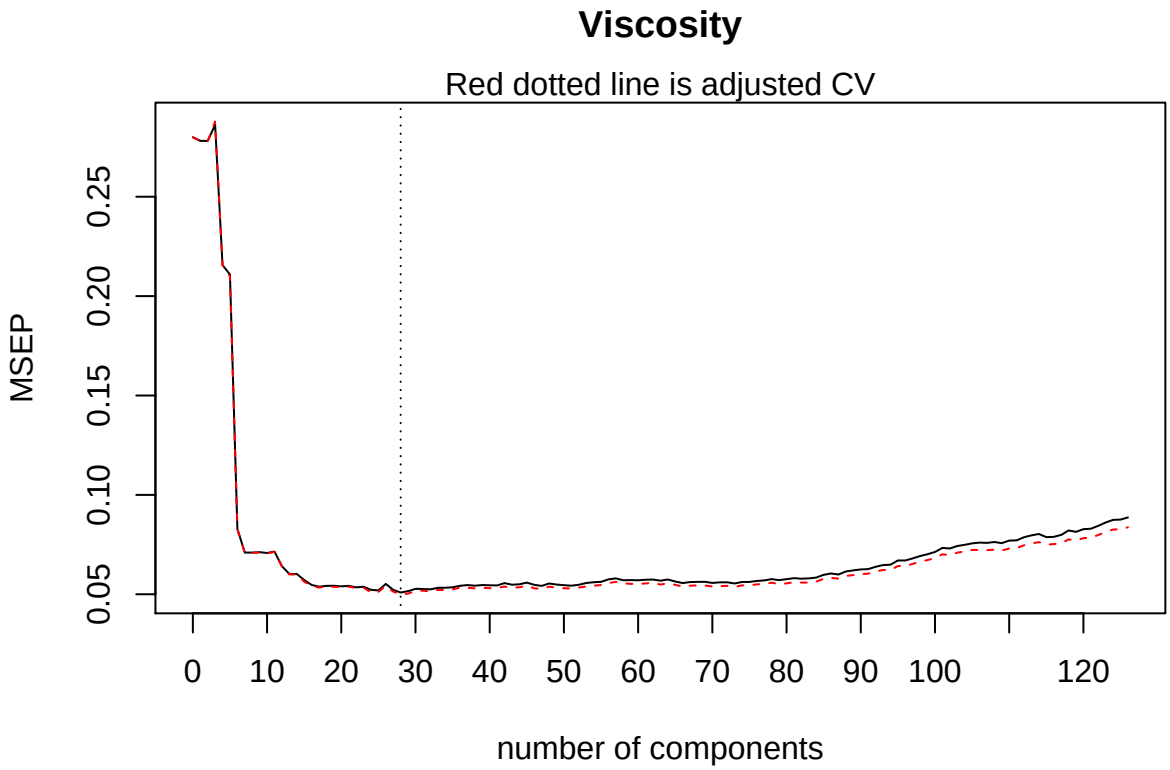
The scores for the first two latent features from the fastICA model, i.e. IC1 and IC2, are shown below.



The scores for IC1 and IC2 looks much the same as for PC1 and PC2. Note that the difference is the reversed values in IC2 compared to PC2. That could also be seen when comparing the trace plots, and it doesn't matter in this case.

Assignment 2.4

The PCR model where I chose the number of PCs by cross validation was done using the built in pcr function in R. The required plot is provided below.



The MSEP gets minimized when I select 28 components. That is shown in the plot, and the exact values for all number of components can be extracted by the call MSEP() in R. What's reasonable to select is not necessary the number of combonents, since I still do want to reduce the number of components. I'd probably go with either 7 or 16 components. That minimizes the MSEP and still reduce the number of components substantially.

Appendix

Below I attached the code for solving this lab.

```
#1.1
state<-read.csv2("data/State.csv", sep=";", header=T)

state<-state[order(state$MET, decreasing=T), ] #Reordrar efter MET
plot(state$MET, state$EX, xlab="MET", ylab="EX", main="MET versus EX")

#1.2
library(tree)

regtree<-tree(formula=EX~MET, data=state, minsize=8, split="deviance")

#plot(regtree)
set.seed(12345)
cv_regtree<-cv.tree(regtree)

plot(cv_regtree)

tree_3leaves<-prune.tree(regtree, best=3)

plot(tree_3leaves)
title("Regression tree selected by CV, EX~MET")
text(tree_3leaves, pretty=0)

par(mfrow=c(1,2))
plot(state$MET, state$EX, main="Original data", xlab="MET", ylab="EX") #Originaldata
plot(state$MET,predict(tree_3leaves),xlab="MET", ylab="Fitted EX", main="The predicted values")

library(ggplot2)
ggplot(data=data.frame(resid=residuals(tree_3leaves)))+geom_histogram(aes(x=resid), bins=10)

#1.3
library(boot)
#regtree<-tree(formula=EX~MET, data=state, minsize=8, split="deviance")

#Prunat till 3 löv---------------------------------
func_CI_tree<-function(data, ind){
  data1<-data[ind,]# extract bootstrap sample
  trees_3<-prune.tree(tree(formula=EX~MET, data=data1, minsize=8, split="deviance"), best=3)
  #fit pruned tree model. I make the tree and prune it in the same code line.
  priceP<-predict(trees_3,newdata=state) #predict values from the original data
  return(priceP)
}
set.seed(12345)
bootstrap_CI_unpara<-boot(data=state, statistic=func_CI_tree, sim="ordinary", R=1000)
#bootstrap_CI

CI<-envelope(bootstrap_CI_unpara, level=.95)

plot(state$MET,state$EX, xlab="MET", ylab="EX",
     main="Confidence band for regression tree \n by non parametric bootstrapping") #originaldata
```

```r
points(state$MET,predict(tree_3leaves), col="red", type="l") #Trädet
points(state$MET, CI$point[1,], col="green", type="l") #CI$point[1,]->Upper
points(state$MET, CI$point[2,], col="green", type="l") #CI$point[2,]->Lower

#1.4
#Börjar med min random generator
random_gen<-function(data, mle){
  data1<-data.frame(EX=state$EX, MET=state$MET)
  n=nrow(data1)
  data1$EX<-rnorm(n=n, mean=predict(tree_3leaves, newdata=data1), sd=sd(residuals(tree_3leaves)))
  return(data1)
}

#Min motsvarande f1 enligt slide 29
funk_CI_tree<-function(state){
  regtrees<-tree(formula=EX~MET, data=state, minsize=8, split="deviance") #fit tree model
  trees_3<-prune.tree(regtrees, best=3)
  preds<-predict(trees_3, newdata=state)
  return(preds)
}

set.seed(12345)
bootstrap_CI_para<-boot(data=state, statistic=funk_CI_tree,
                  sim="parametric", R=1000, mle=tree_3leaves, ran.gen = random_gen)

interval_CI<-envelope(bootstrap_CI_para, level=.95)

#Fortsätter nu med att göra PRED INTERVAL enligt slide 32.

funk_PI_tree<-function(state){
  regtrees<-tree(formula=EX~MET, data=state, minsize=8, split="deviance") #fit tree model

  trees_3<-prune.tree(regtrees, best=3) #prunes tree tol 3 leaves
  preds<-predict(trees_3,newdata=state)

  n<-nrow(state)
  generated_predictions<-rnorm(n=n,mean=preds, sd=sd(residuals(trees_3)))
  #slumpar nf med mean av värdena
  #och samma varians som residualerna har.
  #Alltså genererar slumpmässiga observation utifrån mina
  #predictioner.
  return(generated_predictions)
}
set.seed(12345)
bootstrap_PI<-boot(data=state, statistic=funk_PI_tree, sim="parametric", R=1000, mle=tree_3leaves,
                ran.gen=random_gen)

interval_PI<-envelope(bootstrap_PI, level=.95)

plot(state$MET,state$EX, xlab="MET", ylab="EX",
     main="Confidence and prediction bands for regression tree \n by parametric bootstrapping",
     ylim=c(min(interval_PI$point[2,]), 450))
```

```
points(state$MET,predict(tree_3leaves), col="red", type="l") #Trädet, alltså fitten
points(state$MET, interval_CI$point[1,], col="green", type="l") #CI upper
points(state$MET, interval_CI$point[2,], col="green", type="l") #CI lower
points(state$MET, interval_PI$point[1,], col="blue", type="l") #PI upper
points(state$MET, interval_PI$point[2,], col="blue", type="l") #PI lower

#2.1
nir<-read.csv2("data/NIRspectra.csv", sep=";", header=T)
#PC<-prcomp(nir[, 1:126], scale=T)
#lambda<-(PC$sdev^2)
#sprintf("%2.4f",lambda/sum(lambda)*100)

##############Ass 2.1, egen PCA
nir2<-as.data.frame(scale(nir[, 1:126])) #Tar bort y-variablen här
Y<-as.data.frame(scale(nir[, 127])) #Lägger Y här.
covar<-cov(nir2)#Cov av X-variablerna
eigen_stuff<-eigen(covar)
eigenvalues<-(eigen_stuff$values)
eigen_vectors<-eigen_stuff$vectors #första kol innebär förta egenvektorn
PC_proportion<-eigenvalues/sum(eigenvalues) #Detta är hur mkt de olika PC förklarar.
#sprintf("%2.4f", PC_proportion) #Här skriver jag ut dem. Första förklarar 95.38 %.

i<-1
PCAs<-integer(0)
while(sum(PCAs)<.99){
  PCAs[i]<-PC_proportion[i]
  i<-i+1
}
number_of_components<-length(PCAs)

#PLOTTEN
barplot(PC_proportion[1:10], ylim=c(0,1), ylab="Share of the variance explained",
 xlab="First 10 PC components", las=1)
grid()
title("Variation explained by the first 10 components")

#Z = X U slide 14
X<-as.matrix(nir2)
U<-eigen_vectors[,1:number_of_components]
Z<-as.data.frame(X %*% U)
colnames(Z)<-c("PC1", "PC2")

#Nedan är scoresen för PC1.
plot(x=Z$PC1, y=Z$PC2, xlab=colnames(Z)[1], ylab=colnames(Z)[2],
     main="The scores for PC1 and PC2, PCA" )

###############2.2
par(mfrow=c(1,2))
plot(U[,1], main="Traceplot PC1, it's eigenvector", ylab="value")
grid()
plot(U[,2], main="Traceplot PC2, it's eigenvector", ylab="value")
grid()
```

```
#############2.3
library(fastICA)
#fastICA(X, 2, alg.typ = "parallel", fun = "logcosh", alpha = 1,
 #       method = "R", row.norm = FALSE, maxit = 200, tol = 0.0001, verbose = TRUE)
set.seed(12345)
ica_model<-fastICA(nir2, n.comp=number_of_components,alg.typ="parallel",
                   fun="logcosh", alpha=1, row.norm=F,verbose=T )

#2.3a
#ica_model$K
#ica_model$W

W_prim<-ica_model$K%*%ica_model$W #126x2/2x2->126x2
#W_prim

par(mfrow=c(1,2))

plot(W_prim[,1], main="Traceplot IC1, it's eigenvector", ylab="value")
grid()
plot(W_prim[,2], main="Traceplot IC2, it's eigenvector", ylab="value")
grid()

Z_ica<-as.data.frame(X %*% W_prim) #395x126 126x2
colnames(Z_ica)<-c("IC1", "IC2")

#Nedan är scoresen för ICA
par(mfrow=c(1,1))
plot(x=Z_ica$IC1, y=Z_ica$IC2, xlab=colnames(Z_ica)[1],
     ylab=colnames(Z_ica)[2], main="The scores for the two latent features, ICA")

#2.4
library(pls)

#Detta är PCR enligt slides.
set.seed(12345)
pcr_model<-pcr(Viscosity ~ ., data=nir, scale=TRUE, validation="CV")
#pcr_model
#MSEP(pcr_model)#Detta är vad validationplotten plottar

validationplot(pcr_model,val.type="MSEP", xaxt="n") #xaxt tar bort x-axeln
axis(1, at = seq(0, 120, by = 10), las=1)
abline(v=28, lty=3)
mtext("Red dotted line is adjusted CV")

#plot(MSEP(pcr_model))
```