# TDDE01 – Machine Learning
# Individual Lab Report 4

Martin Estgren <mares480>

December 4, 2016

## 1 Assignment 1

In this assignment we will be analyzing the *percentage of population living in standard metropolitan areas* (MET) over the *Per capita state and local public expenditures* (EX) by using *regression trees*.

We first plot the data (as seen in figure 1) and examine what kind of model could be of interest.

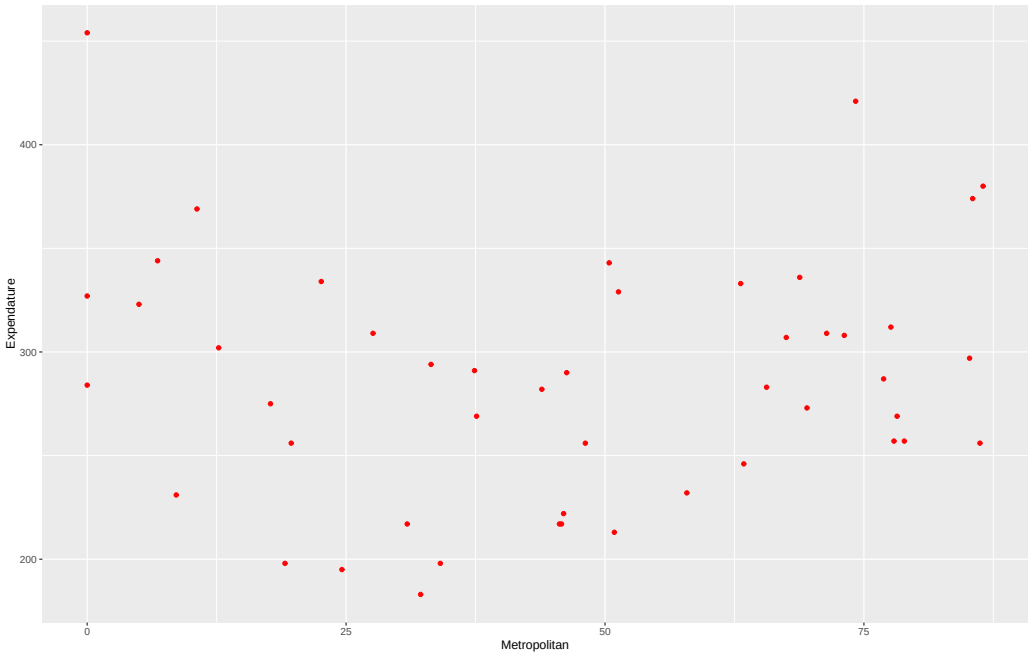



Figure 1: Raw data plot.

There is no observable pattern of significance in the figure. But a *tree model* would probably be a better model than *linear regression*. Because the response variable is continuous we decide on a *regression tree mode*.

The *regression tree mode* we decide on will use *cross validation* and the minimal number of observations in each leaf to 8. The resulting tree can be seen inf figure 2.

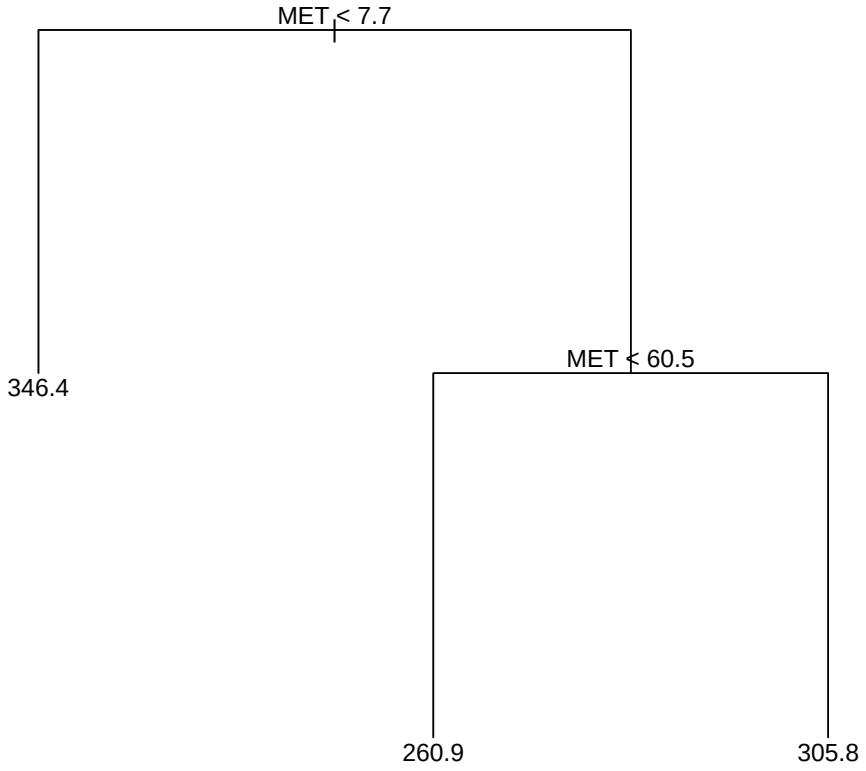



Figure 2: Optimal regression tree model

The optimal tree model resulted in three leafs. The model predictions can be seen in figure 3 and a histogram of residuals in figure 4.

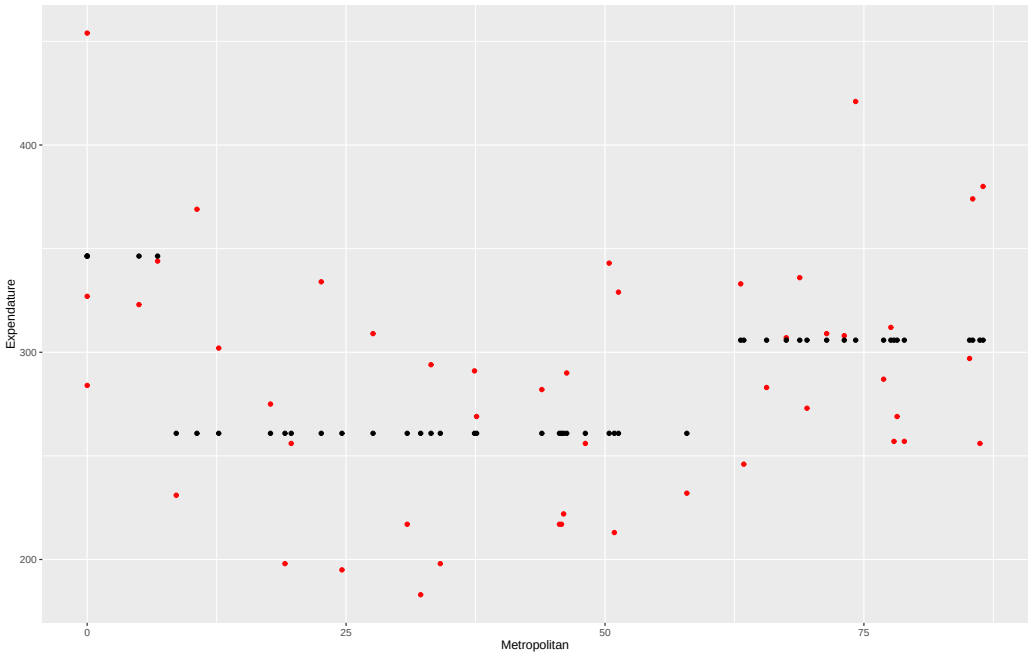



Figure 3: Predictions of the tree model

The number of labels (3) produced by the model is significantly less than the raw data. The labels seems to be located around the respective means of the raw data.

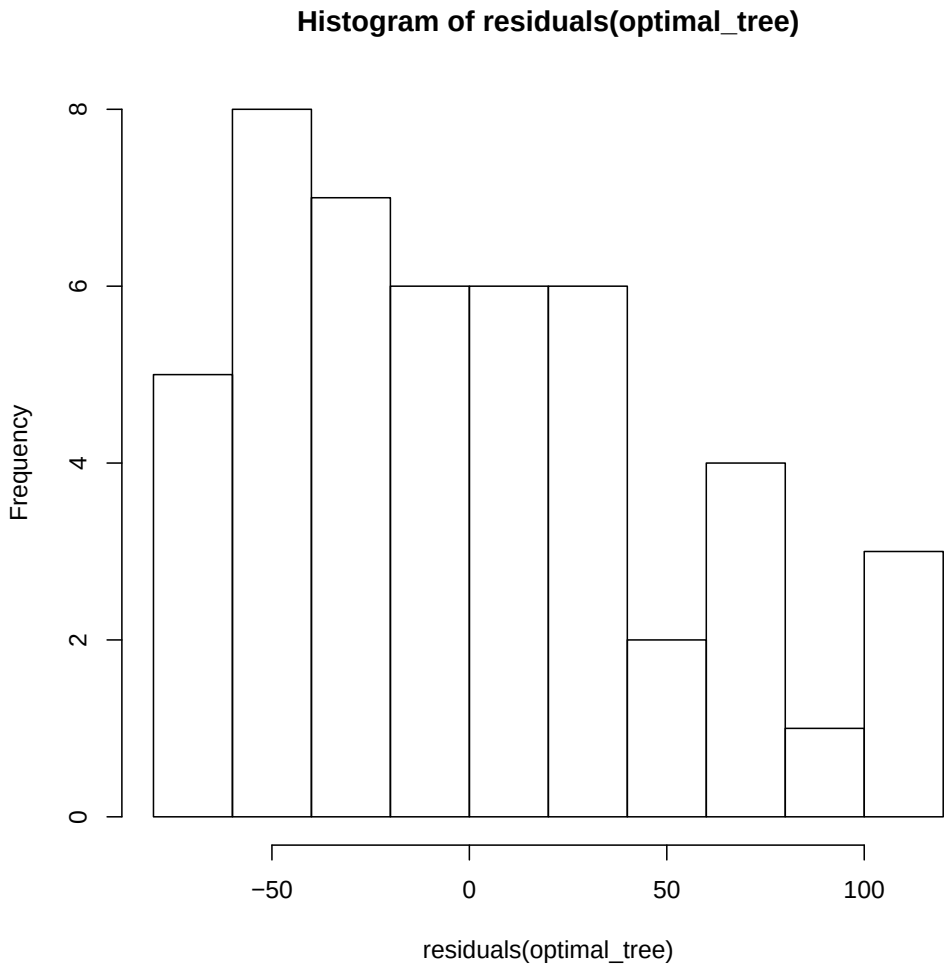



Figure 4: Histogram of the model residuals

Ideally, the histogram would show a normal distribution of residuals with a mean of 0. This is not what can be deduced from the model, although 48 observations are very few observations for a normal distribution to be observable with.

To mitigate this problem we use *bootstrapping* on the optimal tree model. We first try *non-parametric bootstrapping*. The result is plotted with a confidence level of 0.95 in figure 5

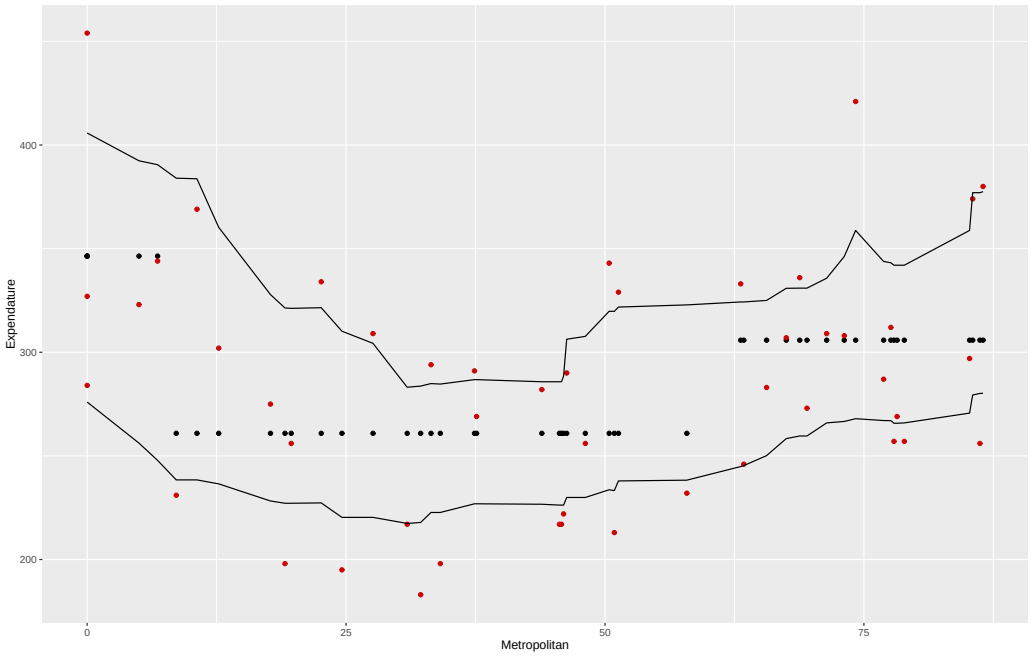



Figure 5: 0.95 confidence bound of the non-parametric bootstrap.

We can observe how most of the observations are located within the confidence bound but significantly more than 0.05 still outside. The result would indicate that the tree model isn't a perfect fit for the data.

We examine the same model, this time using *parametric bootstrap* where we assume the observations fall within the distribution:

$$Y \sim N(\mu_i, \theta^2) \tag{1}$$

The resulting plot can be seen in the figure 6.

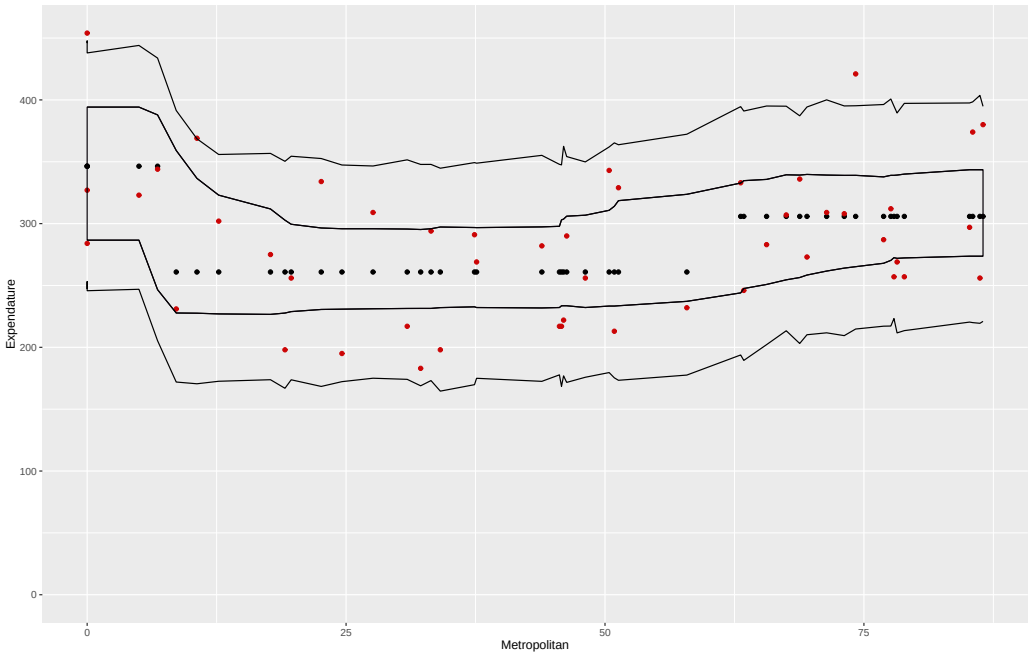



Figure 6: 0.95 confidence bound and predictions bound of the parametric bootstrap.

The 0.95 confidence bound is much tighter around the predictions compared to the *non-parametric bootstrap* with the *prediction bound* encompassing all-but a few outliers of the observations. The *prediction bound* can be seen as the 0.95 confidence bound of the assumed distribution. Looking back at the residuals in figure 4 we cant tell for sure if the data would follow a normal distribution.

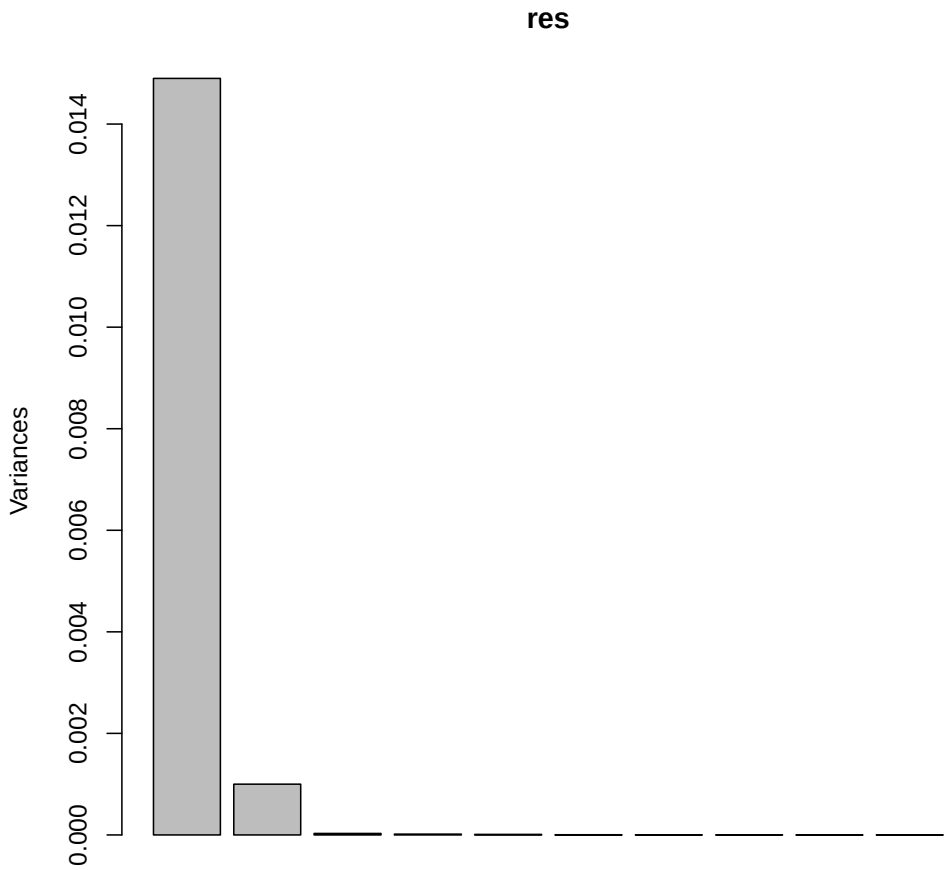
Comparing the *non-parametric* and the *parametric* bootstrap it seems like the *non-parametric* would be a better fit for the given data set.

# 2 Assignment 2

In this assignment we are tasked with analyzing a data set containing observations regarding different levels of *viscosity* and *near-infrared spectra* for many different *diesel fuels* using *PCA* and *ICA* (*Component Analysis Functions*).

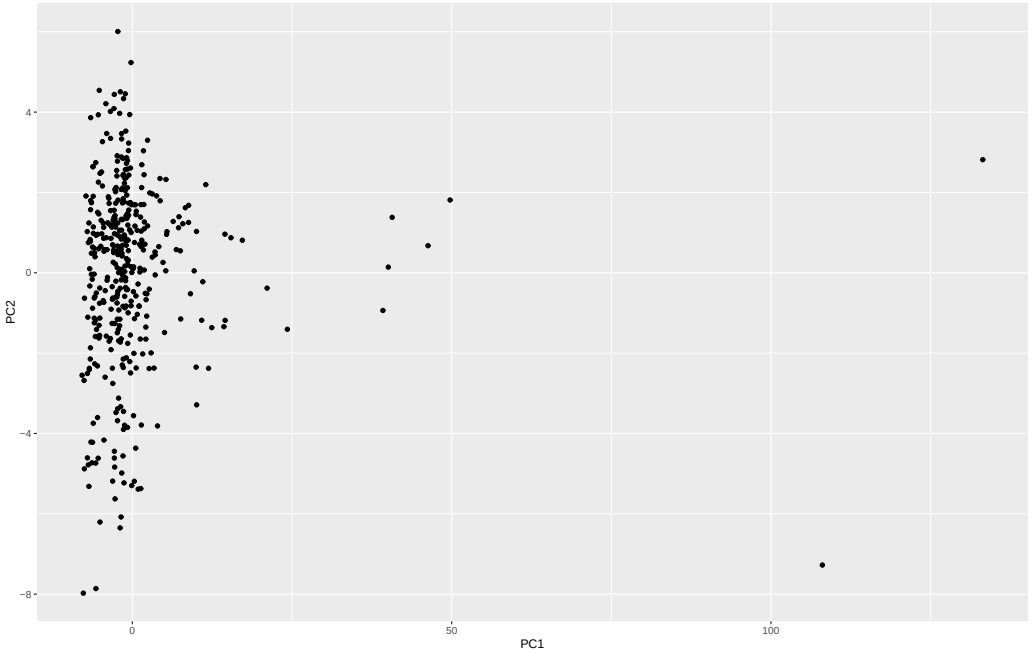
We start by centering the data around 0 and perform a *principal Component Analysis (PCA)* on the data set. The result can be seen in Figure 7.

Figure 7: PCA histogram of variance




Only two components are required to reach 99 % cumulative variance. The components are *X750 (PC1)* and *X752 (PC2)*. In Figure 8 the score for each of the two most significant PCs are displayed.

Figure 8: PCA score distribution




As can be seen, there is a large amount of independence between these two components on the X-axis while they are fairly compacted on the y-axis, ignoring the outliers. The outliers indicates unusual diesel fuels. We plot the traces for each of the two PCA with the highest variance impact on the variance. The resulting plots can be seen in Figures 9 and 10.

Figure 9: Trace plot of PC1

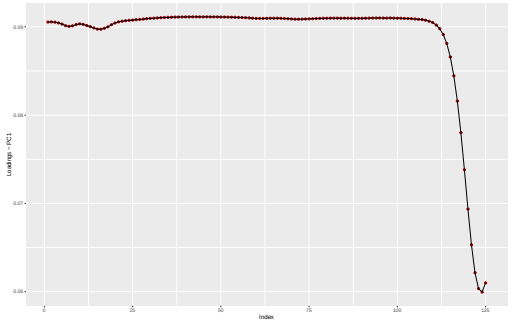
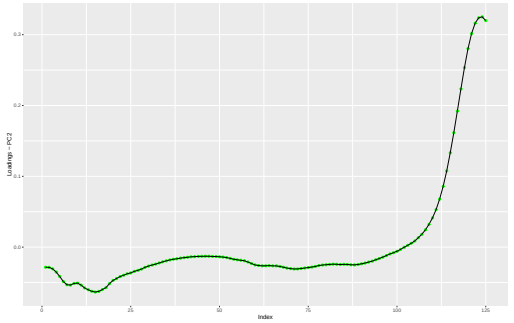


Figure 10: Trace plot of PC2



We can observe how the the components where the PC1 have a high correlation, PC2 have low and vice e versa.

We perform the same analysis again, this time using the *Independent component analysis* (ICA). In contrast to to PCA, we assume the components are independent.

The loadings are calculated with the function $\hat{W} = K\dot{W}$. The traces can be observed in Figure 11 and 12.

Figure 11: Trace plot of IC1

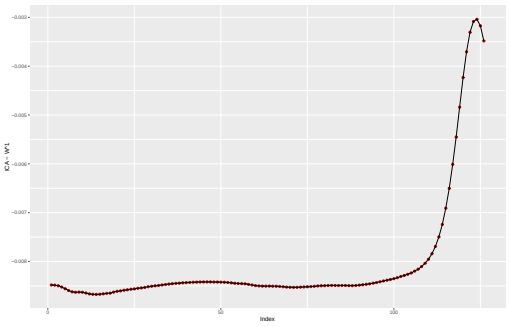
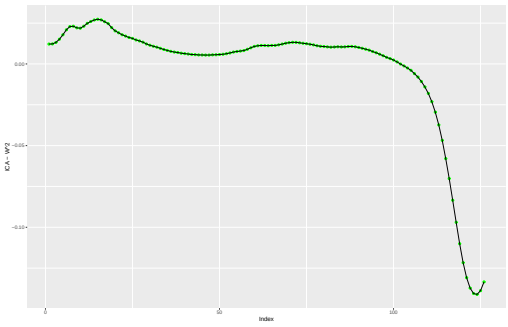


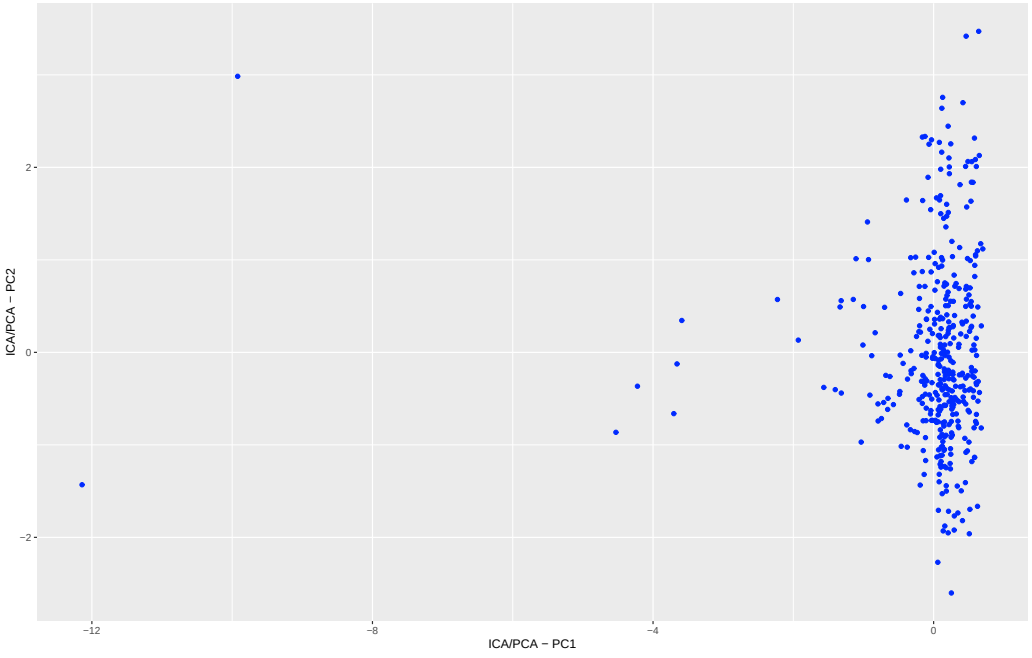
Figure 12: Trace plot of IC2



The results are quite similar to those found in PCA, but inverted. This is expected since we are looking for the independence instead of the correlation.

We now plot the scores found by doing ICA, which can be seen in Figure 13.

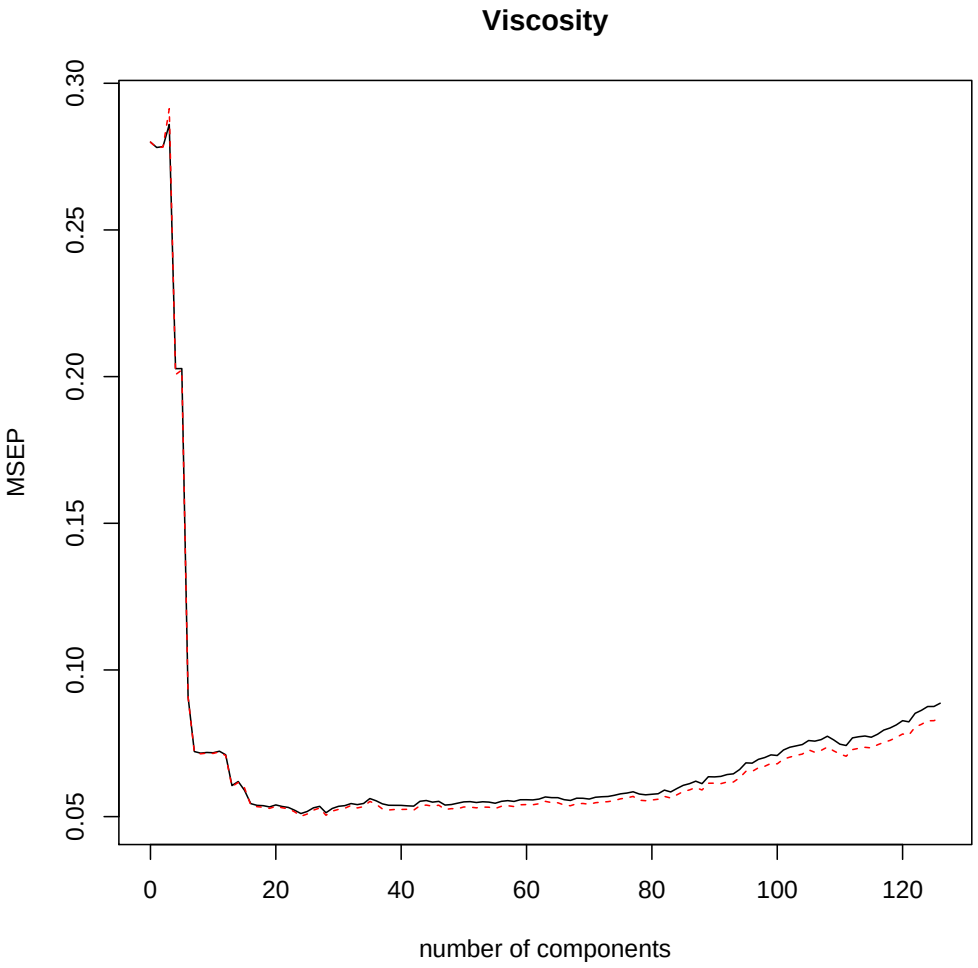
Figure 13: ICA score distribution




The score distribution is similar to the PCA analysis but with a different magnitude and mirrored Y-values. This could be explained by the analysis switching feature space.

We perform a *principal component regression* analysis with *cross validation* in order to examine the number of components that should be selected for our model. The result can be observed in figure 13

Figure 14: Mean squared predicted error over number of components




We can observe from the figure how the MSEP is high when few components are selected and how the mean squared error drastically decreases around 8 components. The number of components with the lowest mean squared error is around 20. The most reasonable number of components would be around 20.

# 3 Appendix: A - Code assignment 1

Listing 1: Code for assignment 1

```
library(tree)
library(boot)
library(ggplot2)
library(reshape2)
set.seed(12345)
data = read.csv2("State.csv", header = TRUE)
data = data[order(data$MET),]

controll = tree.control(nrow(data), minsize = 8)
fit = tree( EX ~ MET, data, control = controll)

fit.cv = cv.tree(fit)
best_k = fit.cv$size[which.min(fit.cv$dev)]
optimal_tree = prune.tree(fit, best=best_k)

setEPS()
postscript("A1_tree.eps")
plot(optimal_tree)
text(optimal_tree)
dev.off()

predictions = predict(optimal_tree, newdata=data)

fig_data = data.frame(x = data$MET, pred = predictions
   , orig = data$EX)
fig = ggplot(fig_data, aes(x, pred, orig)) +
  geom_point(aes(x,orig), colour = "#FF1111") +
#  geom_point(aes(x, pred)) +
  labs(x = "Metropolitan") +
  labs(y = "Expendature")
print(fig)
ggsave(file="A1_data.eps")

fig_data = data.frame(x = data$MET, pred = predictions
   , orig = data$EX)
fig = ggplot(fig_data, aes(x, pred, orig)) +
  geom_point(aes(x,orig), colour = "#FF1111") +
```

```
  geom_point(aes(x, pred)) +
  labs(x = "Metropolitan") +
  labs(y = "Expendature")
print(fig)
ggsave(file="A1_fit.eps")

setEPS()
postscript("A1_histogram_residuals.eps")
hist(residuals(optimal_tree))
dev.off()

 nonparama = function(data,index){
     sample = data[index,]
     controll = tree.control(nrow(sample), minsize =
        8)
     fit = tree( EX ~ MET, data=sample, control =
        controll)
     optimal_tree = prune.tree(fit, best=best_k)
     return(predict(optimal_tree, newdata=data))
 }

#
 set.seed(12345)
 nonparam_boot = boot(data, statistic = nonparama, R
    =1000)
 confidence_bound = envelope(nonparam_boot,level=0.95)
 predictions = predict(optimal_tree,data)


plot(nonparam_boot)

fig_data = data.frame(orig = data$EX, x=data$MET, pred
   =predictions, upper=confidence_bound$point[1,],
   lower=confidence_bound$point[2,])
fig = ggplot(fig_data, aes(x,predictions,upper,lower))
fig = fig +
    geom_point(aes(x, pred)) +
    geom_point(aes(x, orig),colour="#CC1111") +
    geom_line(aes(x,upper)) +
    geom_line(aes(x,lower)) +
    geom_ribbon(aes(x = x, ymin=lower, ymax=upper),
```

```
       alpha=0.05) +
    labs(x = "Metropolitan") +
    labs(y = "Expendature")
print(fig)
ggsave(file="A1_nonparametric.eps")

# lines(data$MET,confidence_bound$point[1,], col="Red
   ")
# lines(data$MET,confidence_bound$point[2,], col="Red
   ")

 parama_conf = function(data){
   controll = tree.control(nrow(data), minsize = 8)
   fit = tree( EX ~ MET, data=data, control = controll
      )
   optimal_tree = prune.tree(fit, best=best_k)
   return(predict(optimal_tree, newdata=data))
 }

 parama_predic = function(data){
   controll = tree.control(nrow(data), minsize = 8)
   fit = tree( EX ~ MET, data=data, control = controll
      )
   optimal_tree = prune.tree(fit, best=best_k)
   predictions = predict(optimal_tree, newdata=data)
   return(rnorm(nrow(data),predictions,sd(resid(fit)))
      )
 }

 random_predictions = function(data, model){
  sample = data.frame(MET=data$MET, EX=data$EX)
  sample$EX = rnorm(nrow(data), predict(model,newdata=
     data),sd(resid(model)))
  return(sample)
}

  set.seed(12345)
  param_boot_conf = boot(data, statistic = parama_conf
     , R=1000, mle = optimal_tree, ran.gen = random_
     predictions, sim = "parametric")
  confidence_bound_param = envelope(param_boot_conf,
```

```
     level=0.95)

plot(param_boot_conf)
 #
set.seed(12345)
param_boot_pred = boot(data, statistic = parama_predic
   , R=1000, mle = optimal_tree, ran.gen = random_
   predictions, sim = "parametric")
 prediction_bound_param = envelope(param_boot_pred,
    level=0.95)

plot(param_boot_pred)

predictions = predict(optimal_tree,data)
fig_data = data.frame(orig = data$EX, x=data$MET, pred
   =predictions, upper_c=confidence_bound_param$point
   [1,], lower_c=confidence_bound_param$point[2,],
   upper_p=prediction_bound_param$point[1,], lower_p=
   prediction_bound_param$point[2,])

 #
  fig = ggplot(fig_data, aes(orig,x,pred,upper_c,lower
     _c, upper_p, lower_p), xlab = "Metropolitan" ,
     ylab = "Predicted Expendature")
  fig = fig +
    geom_point(aes(x, pred)) +
    geom_point(aes(x, orig),colour="#CC1111") +
    geom_line(aes(x,upper_c)) +
    geom_line(aes(x,lower_c)) +
    geom_ribbon(aes(x = x, ymin=lower_c, ymax=upper_c)
       , alpha=0.05, colour = "#110011")+
    geom_line(aes(x,upper_p)) +
    geom_line(aes(x,lower_p))+
    geom_ribbon(aes(x = x, ymin=lower_p, ymax=upper_p)
       , alpha=0.05) +
    labs(x = "Metropolitan") +
    labs(y = "Expendature")
  print(fig)
  ggsave(file="A1_parametric.eps")
```

# 4 Appendix: B - Code assignment 2

Listing 2: Code for assignment

```
library(ggplot2)
library(fastICA)
library(pls)

fulldata = read.csv2("NIRSpectra.csv")
data = fulldata[, -ncol(fulldata)]

res = prcomp(data)
lambda = res$dev^2
#eigen
print(lambda)

#Variance
sprintf("%2.3f",lambda/sum(lambda)*100)

setEPS()
postscript('A2_pcahist.eps')
screeplot(res)
dev.off()

fig_data = data.frame(x=res$x[,1], y=res$x[,2], xlab =
    "PC1", ylab = "PC2")
fig = ggplot(fig_data) +
  geom_point(aes(x = x,y = y)) +
  labs(x = "PC1") +
  labs(y = "PC2")
print(fig)
ggsave(file="A2_pcascore.eps")

U = res$rotation
U = U[-nrow(U),]

fig_data = data.frame(x1=1:length(U[,1]), y1=U[,1])
fig = ggplot(fig_data, ylim = c(0,0.15), xlab = "Index
   " , ylab = "Loadings - PC1") +
  geom_point(aes(x1,y1), col="#AA1111") +
  geom_line((aes(x1,y1))) +
```

```
  labs(x = "Index") +
  labs(y = "Loadings - PC1")
print(fig)
ggsave(file="A2_trace_PC1.eps")


fig_data = data.frame(x1=1:length(U[,2]), y1=U[,2])
fig = ggplot(fig_data, ylim = c(0,0.15), xlab = "Index
   " , ylab = "Loadings - PC2") +
  geom_point(aes(x1,y1), col="#00FF11") +
  geom_line((aes(x1,y1))) +
  labs(x = "Index") +
  labs(y = "Loadings - PC2")
print(fig)
ggsave(file="A2_trace_PC2.eps")

set.seed(12345)
ica = fastICA(data,2)
um = ica$K %*% ica$W
fig_data = data.frame(x1=1:length(um[,1]), y1=um[,1])
fig = ggplot(fig_data, ylim = c(0,0.15), xlab = "Index
   ", ylab = "ICA - W^1") +
  geom_point(aes(x1,y1), col="#AA1111") +
  geom_line((aes(x1,y1))) +
  labs(x = "Index") +
  labs(y = "ICA - W^1")
print(fig)
ggsave(file="A2_trace_ICA1.eps")


fig_data = data.frame(x1=1:length(um[,2]), y1=um[,2])
fig = ggplot(fig_data, ylim = c(0,0.15), xlab = "Index
   ", ylab = "ICA - W^2") +
  geom_point(aes(x1,y1), col="#00FF11") +
  geom_line((aes(x1,y1))) +
  labs(x = "Index") +
  labs(y = "ICA - W^2")
print(fig)
ggsave(file="A2_trace_ICA2.eps")

fig_data = data.frame(xn=ica$S[,1], yn=ica$S[,2],xo=
```

```
   res$x[,1], yo=res$x[,2])
fig = ggplot(fig_data, ylim = c(0,0.15), xlab = "ICA/
   PCA␣-␣PC1", ylab = "ICA/PCA␣-␣PC2") +
  geom_point(aes(x = xn,y=yn), col="#0033FF") +
  geom_point(aes(x = xo, y=yo), col="#FF6666")  +
  labs(x = "ICA/PCA␣-␣PC1") +
  labs(y = "ICA/PCA␣-␣PC2")
print(fig)
ggsave(file="A2_icascore.eps")

set.seed(12345)
pcr.fit <- pcr(Viscosity ~ ., data=fulldata,
   validation="CV")

setEPS()
postscript("A2_viscosity.eps")
validationplot(pcr.fit,val.type = "MS")
dev.off()
```