Introduction to Machine Learning

Lab 4

Anton Persson, Emil Klasson Svensson, Mattias Karlsson, Rasmus Holm

2016-12-03

Contents

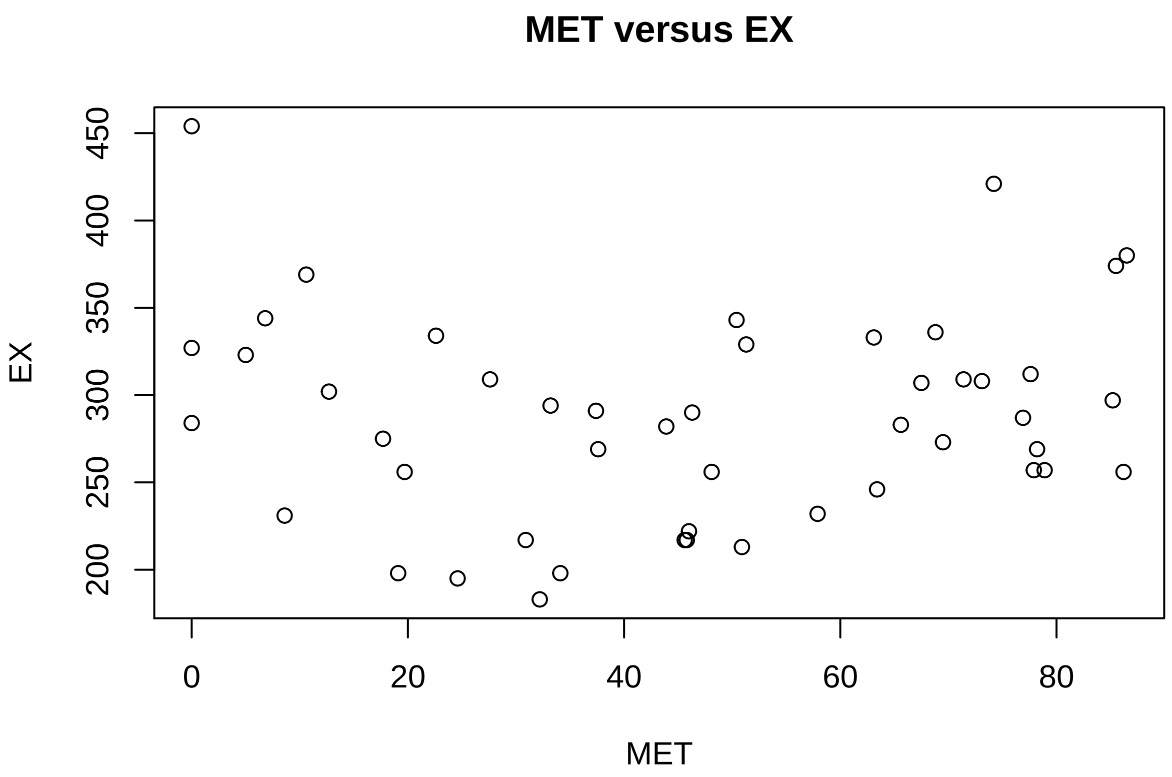
Assignment 1 **2**
1 2
2 2
3 5
4 6
5 7

Assignment 2 **8**
1 8
2 10
3 10
4 12

Appendix **14**
Code for Assignment 1 14
Code for Assignment 2 16

Assignment 1

1



The data from the plot above could probably be modelled reasonably well by a cubic spline function or a piecewise linear function with two knots.

2

To fit a regression tree to the data set we ran cross-validation in order to determine the optimal number of leaves. The outcome was three leaves which had the lowest deviance measure and that can be seen in figure 1. Four leaves had the same outcome but since we prefer a simpler model we picked three leaves.

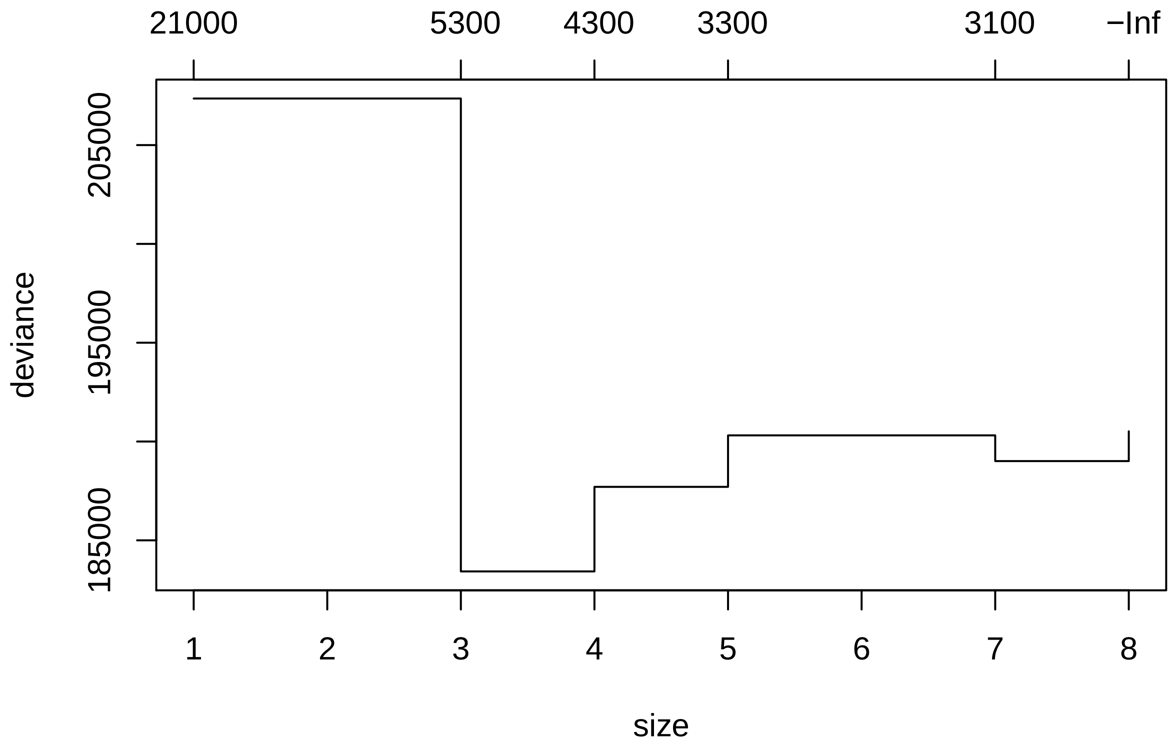


Figure 1: Deviance measure as the number of leaves increases.

The resulting tree can be seen in figure 2 which have divided the MET feature space into three intervals.

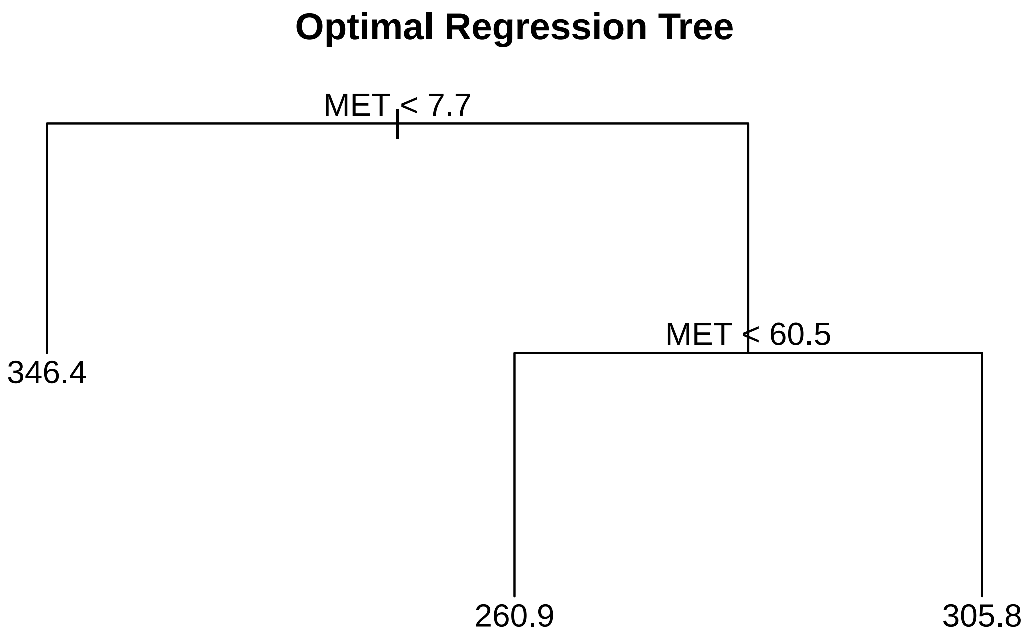


Figure 2: Regression tree where EX is response and MET is the explanatory variable.

The estimated values is seen in figure 3 and are decently matched with the observed values. However, it is clear that the model is too simplistic to represent the distribution from which the observations were generated.

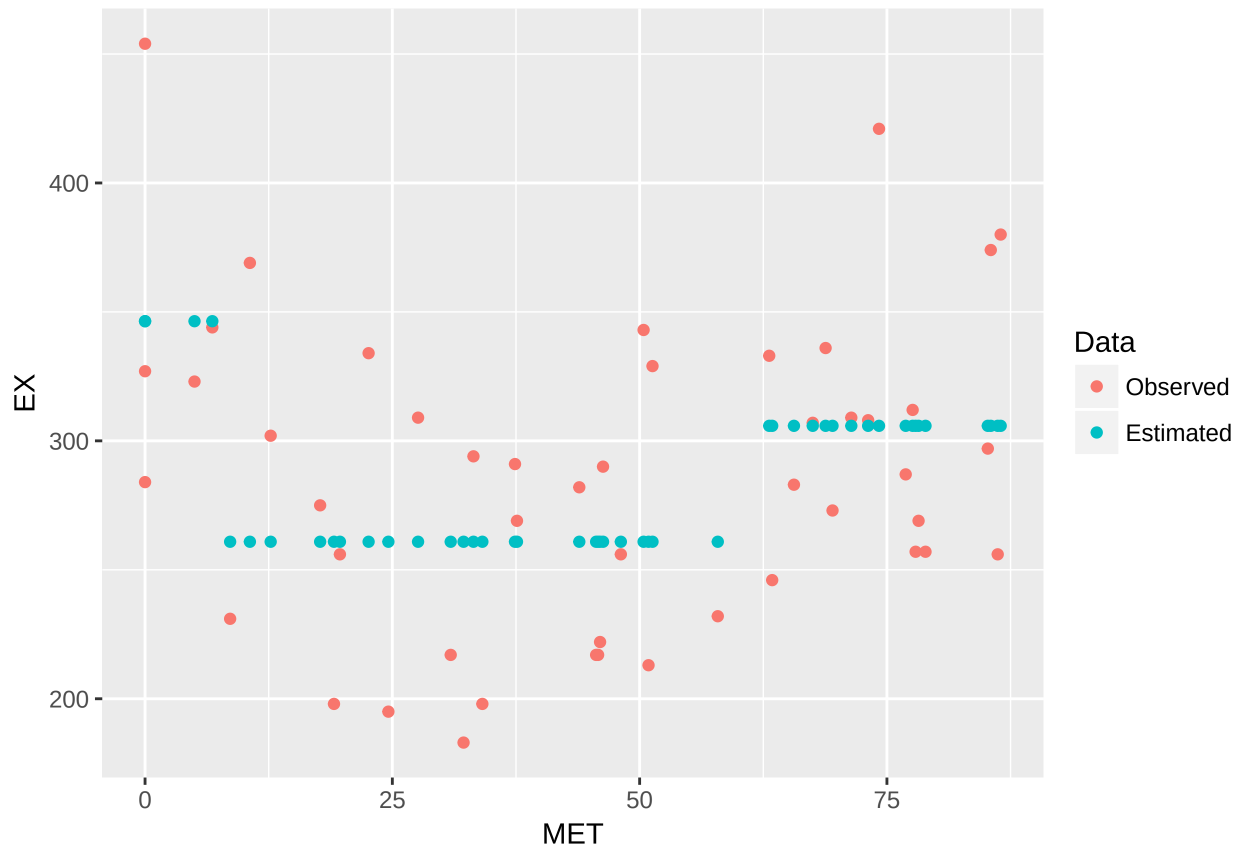


Figure 3: The esimated values by the regression tree.

The residuals in figure 4 resemble either a Gaussian or Gamma distribution quite well. Since the number of residuals are few, 48 to be precise, it is difficult to know exactly which distribution would match if any known at all.

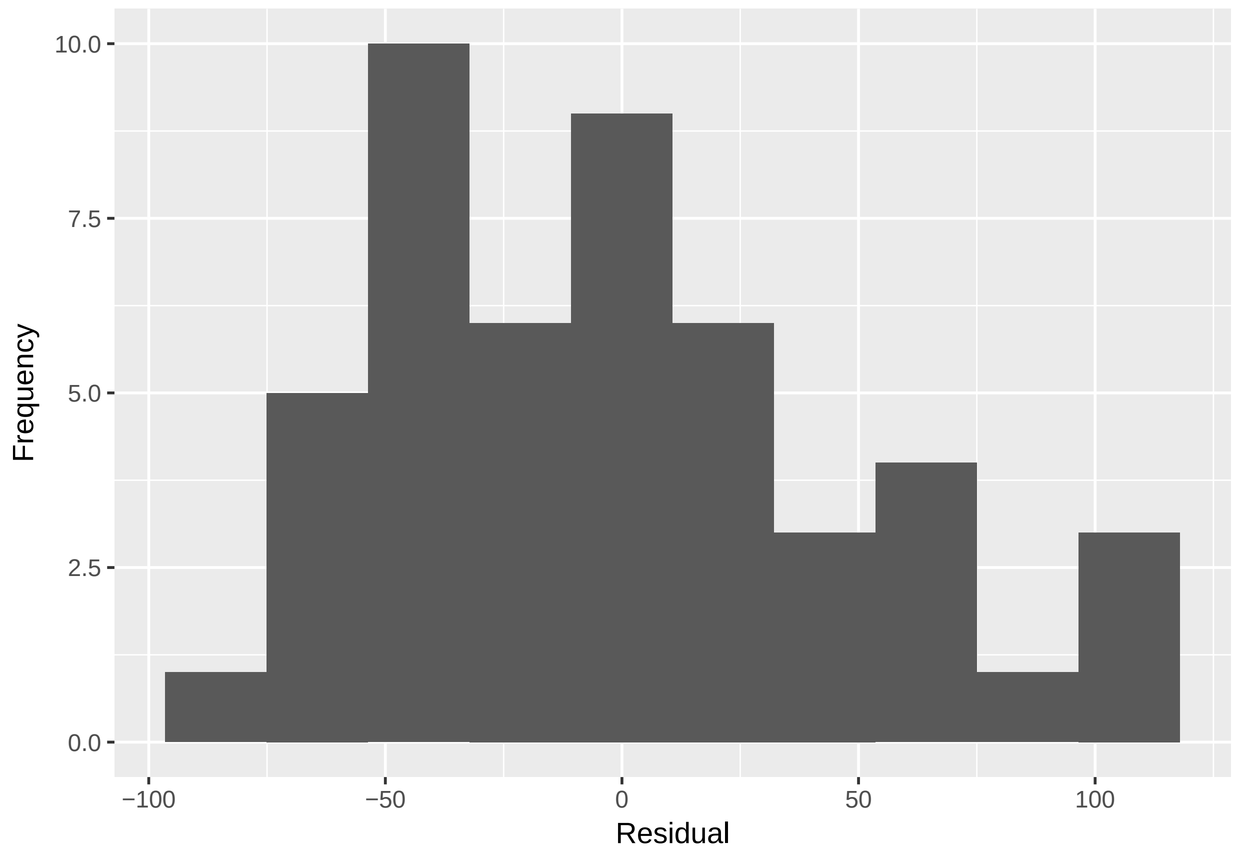


Figure 4: The distribution of the residuals by the regression tree.

3

We used non-parametric bootstrap in order to estimate the 95% confidence band for the regression tree which can be seen in figure 5.

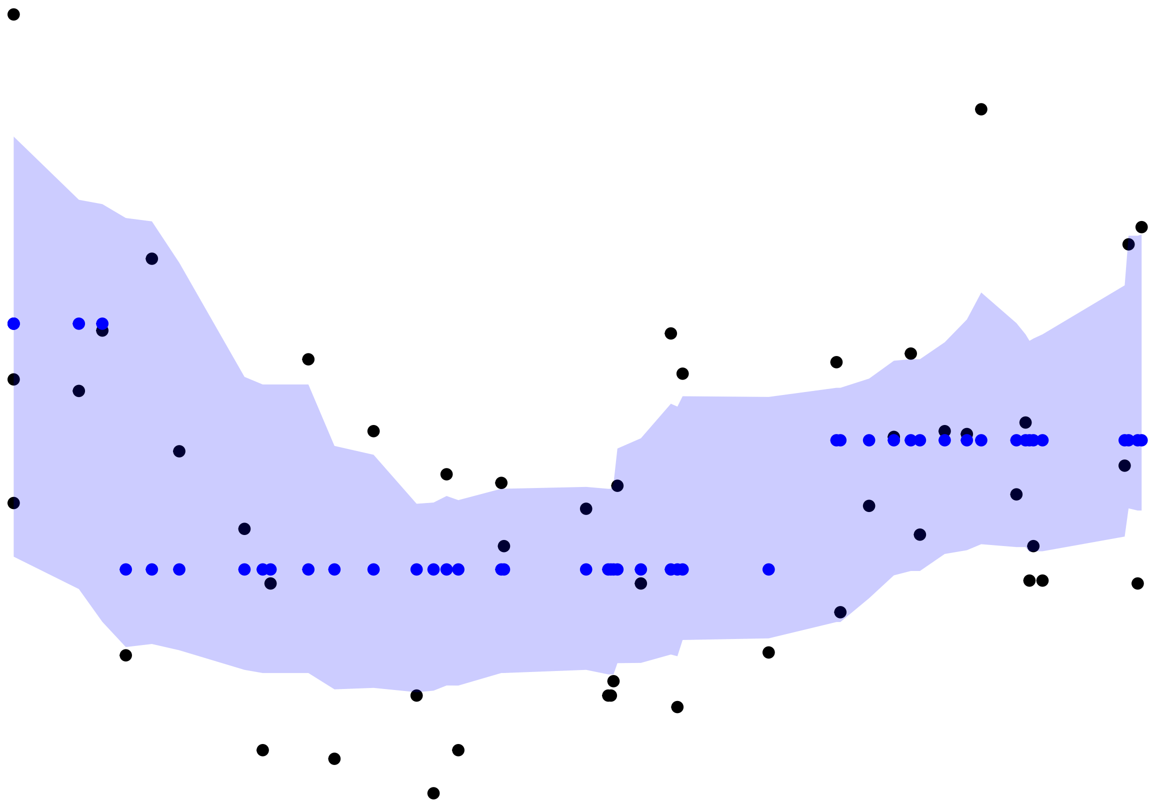


Figure 5: The confidence band estimated by non-parametric bootstrap for the regression tree.

The confidence bounds are bumpy and the width of the band varies dramatically with the x-values. The model is more confident in the middle and right part of the plot compared to high uncertainty of the mean prediction where values in x-axis are low. The bumpiness probably have to do with the lack of observations and we can see that outliers contribute to large spikes in the upper bound. Since the band is wide we conclude that the model is unreliable and too simplistic, as mentioned previously, to find a good estimate of the true mean.

4

Here we used the parametric bootstrap where we have assumed that $Y \sim \mathcal{N}(\mu_i, \sigma^2)$ where μ_i are the values in the tree leaves and σ^2 is the residual variance.

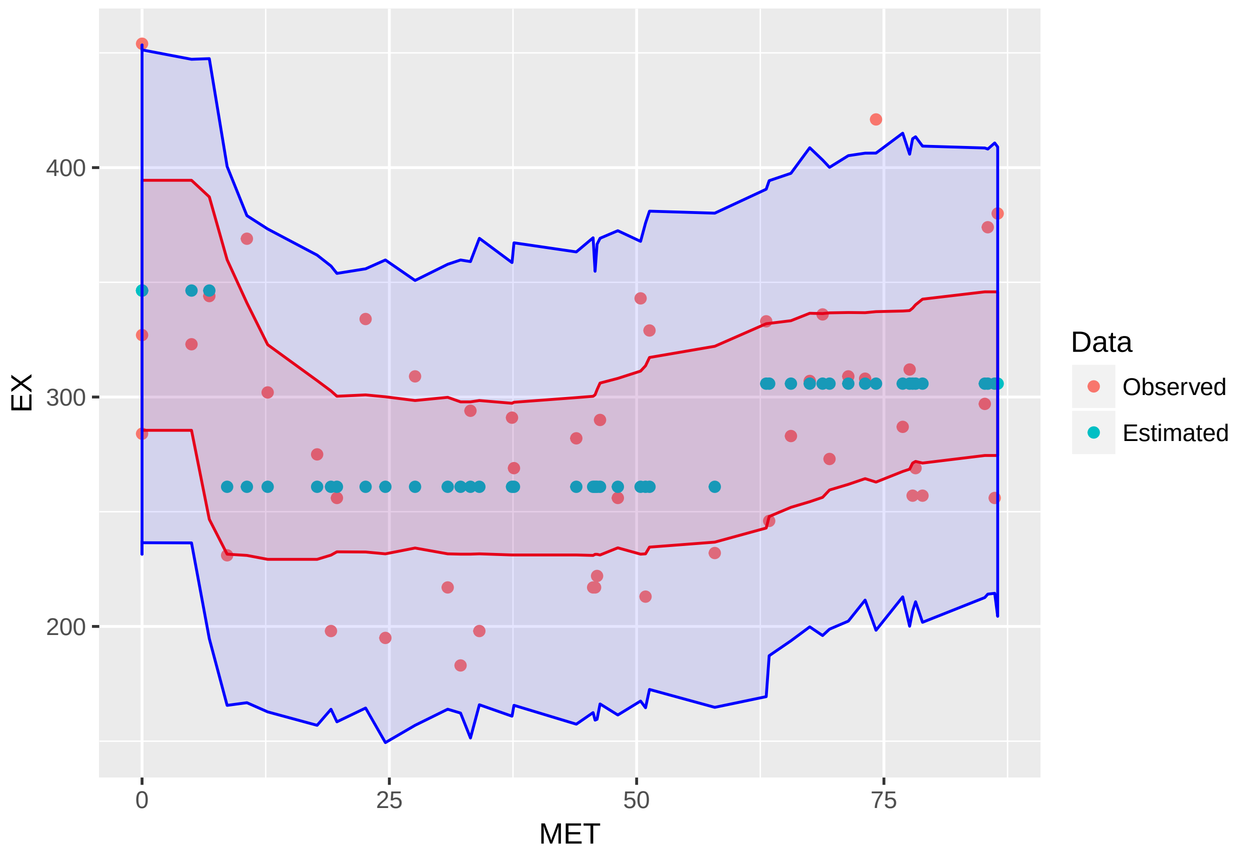


Figure 6: The 95% confidence and prediction bands found by parametric bootstrap after 1000 samples. The red band is the confidence band and the blue band is the prediction band.

We can see from figure 6 that the confidence band is more narrow, smoother, and consistent over the whole range on the x-axis compared to previous result because we have assumed a Gaussian distribution and introduced noise in our simulation. However, the confidence band are still wide and we support our previous conclusion that the model is unreliable. Since 5% of 48 is approximately 2 we are not surprised to find two observations lying outside the prediction band. That is because the 95% prediction bands specify the range in EX that an observation would lie in given its MET value with probability 95%, i.e. 5% probability to lie outside the prediction band.

5

Since we found the residuals to look similar to a Gaussian distribution the parametric bootstrap is more appropriate in the case and especially since the number of observations are so few it is helpful to explicitly guide the bootstrap method.

Assignment 2

1

In this assignment we have used the NIRspectra data set that contains near-infrared spectra and viscosity levels for a collection of diesel fuels.

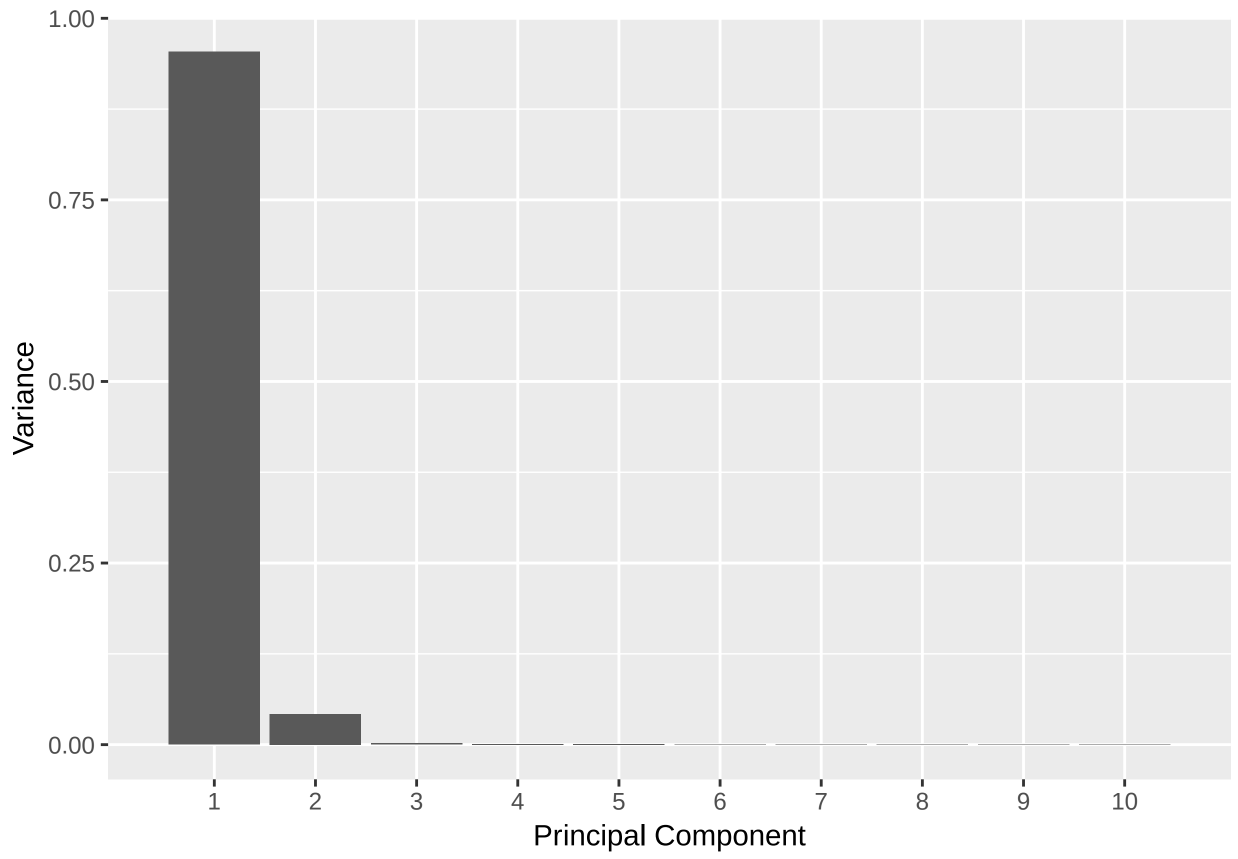


Figure 7: Variance explained by the first 10 principal components.

Figure 7 shows how much of the variance the first 10 principal components explain and we can clearly see that the first two explain most of it, over 99% to be more precise. This means that we will be extracting those two alone for further studies.

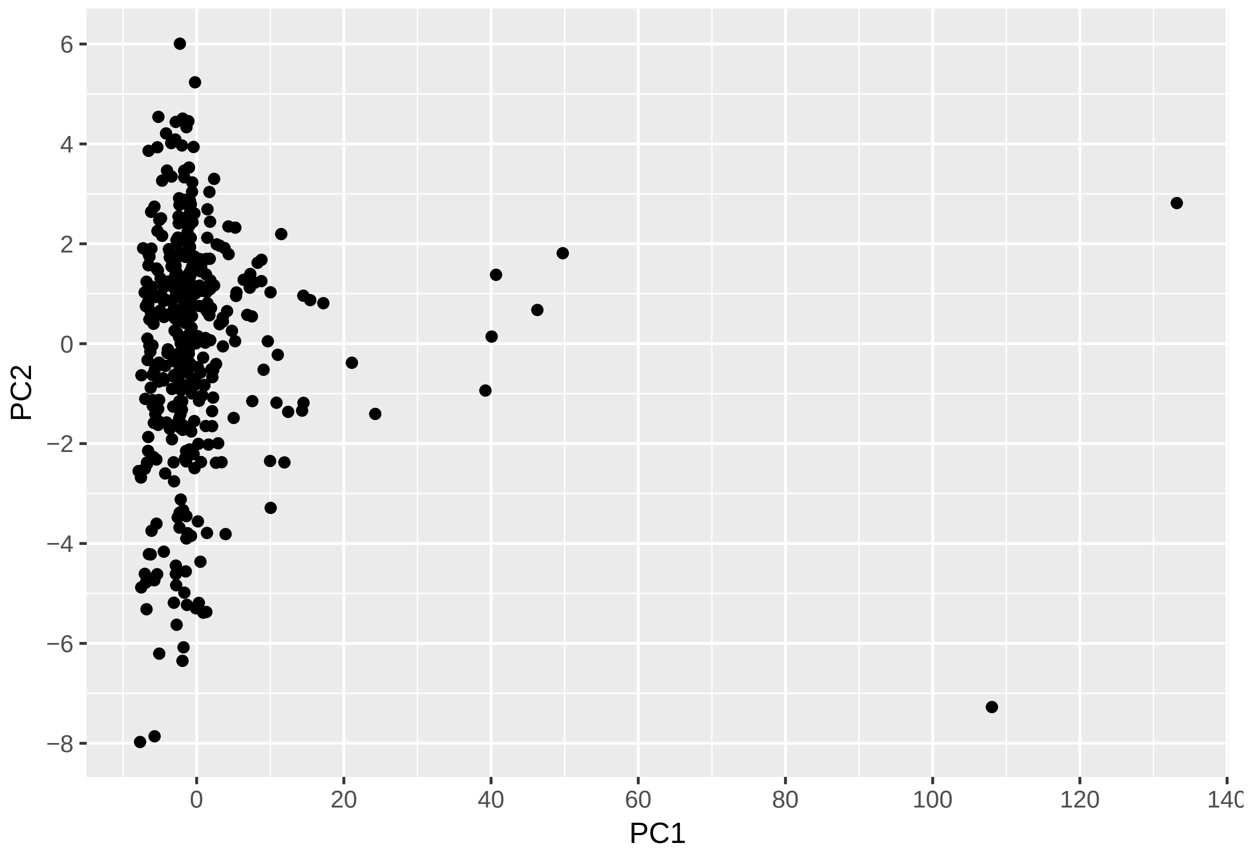


Figure 8: The data projected onto the first two principal components.

The data set is projected onto the first principal components (PCs) in figure 8 and we can detect outliers with respect to the first PC so there exists unusual diesel fuels in the data set.

2

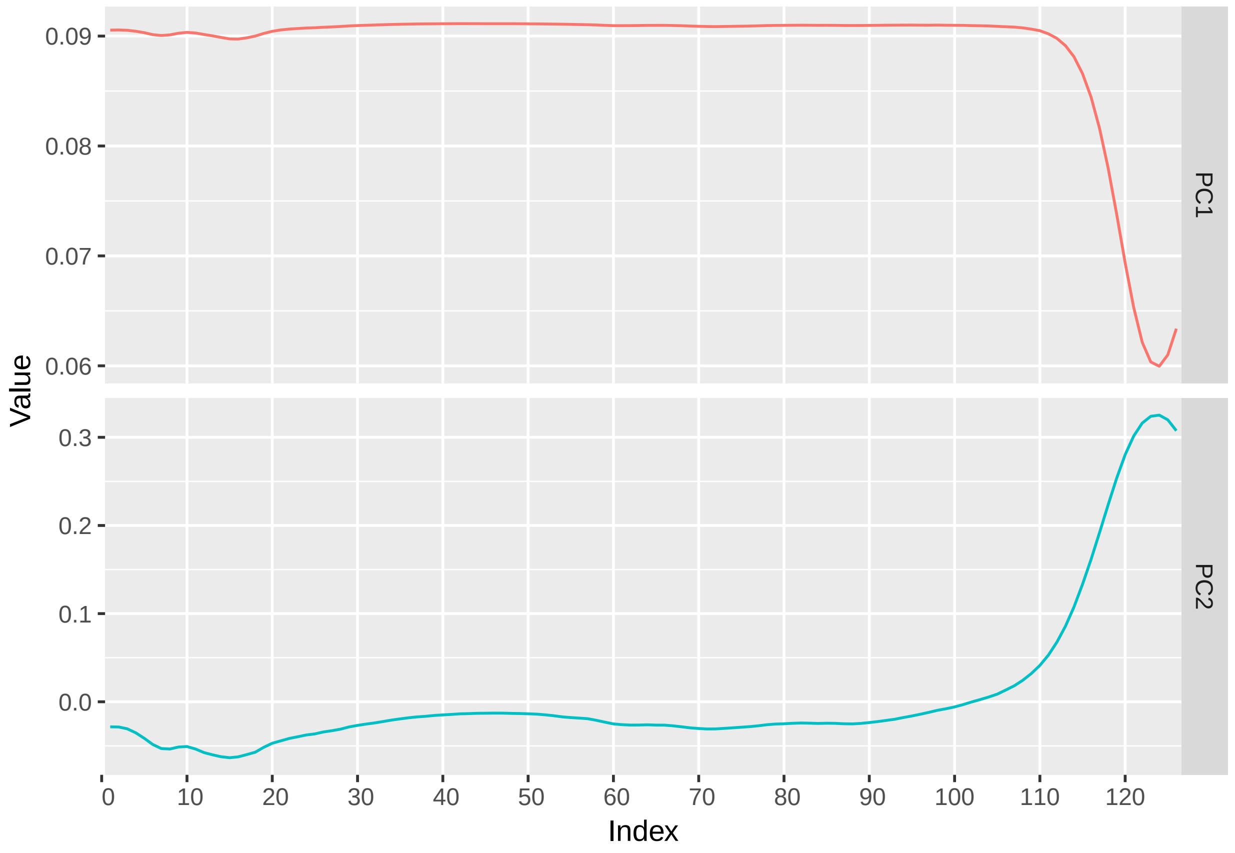


Figure 9: Trace plot of the first two principal components.

The trace plot in figure 9 shows that first PC gives a lot of weight to the 110 first variables and after there is a quick drop of in what the last 17 variables explain. In the second PC the situation is revesed and the last 17 variables are given more weight. The further away a weight is from zero the more dependence of that particular feature is explained by the PC.

3

Here we used independent component analysis (ICA) by finding 2 components using fastICA. In ICA we want to solve $X = SA$ where columns of S are the independent components (ICs) which is equivalent to $XW = S$ given $W = A^{-1}$. However, the algorithm uses a pre-whitening matrix K to project the data, i.e. XK, so the solution becomes $XKW = S$ where KW then are the variable loadings that basically correspond to the eigenvectors in PCA.

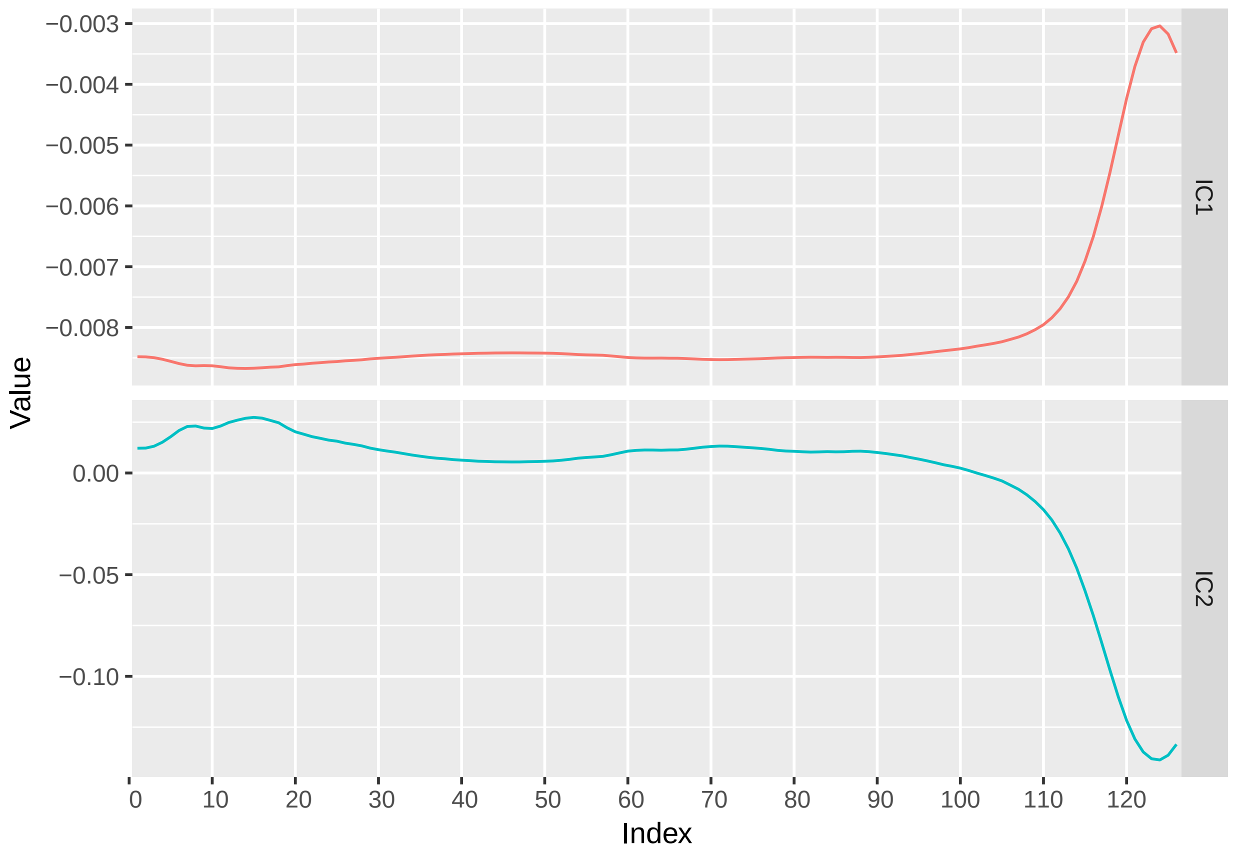


Figure 10: Traceplot of the first two independent components.

In figure 10 we can see that the first two components are similar to those found by PCA but with negative loadings of the features. Whether the loadings are negative or positive do not really matter, it just determines the direction of the vector, i.e. is the direction "*up*" or "*down*". IC1 is a linear combination of all the features with slightly less loadings in the last 17 variables. Since most loadings in IC2 are close to zero it is mostly a linear combination of the last 17 variables and the values are a lot higher than those found in IC1. This indicates that they complement each other.

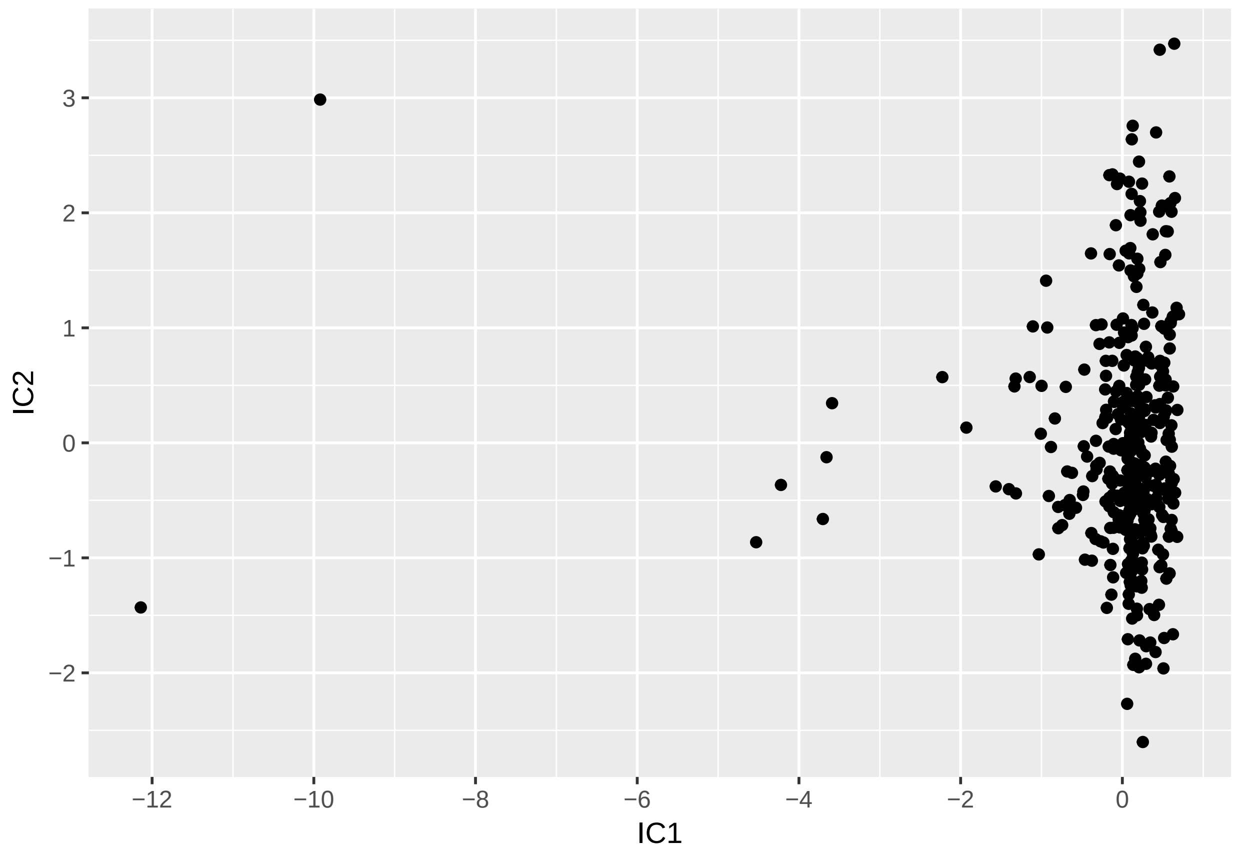


Figure 11: The data projected onto the first two independent components.

The score in figure 11 are very similar to the PCA-plot with different scales, but here the components are negative instead of positive which gives it a mirrored result.

4

In this exercise we have used principal component regression (PCR) in order to estimate the viscosity and let cross-validation estimate the optimal number of components to choose.

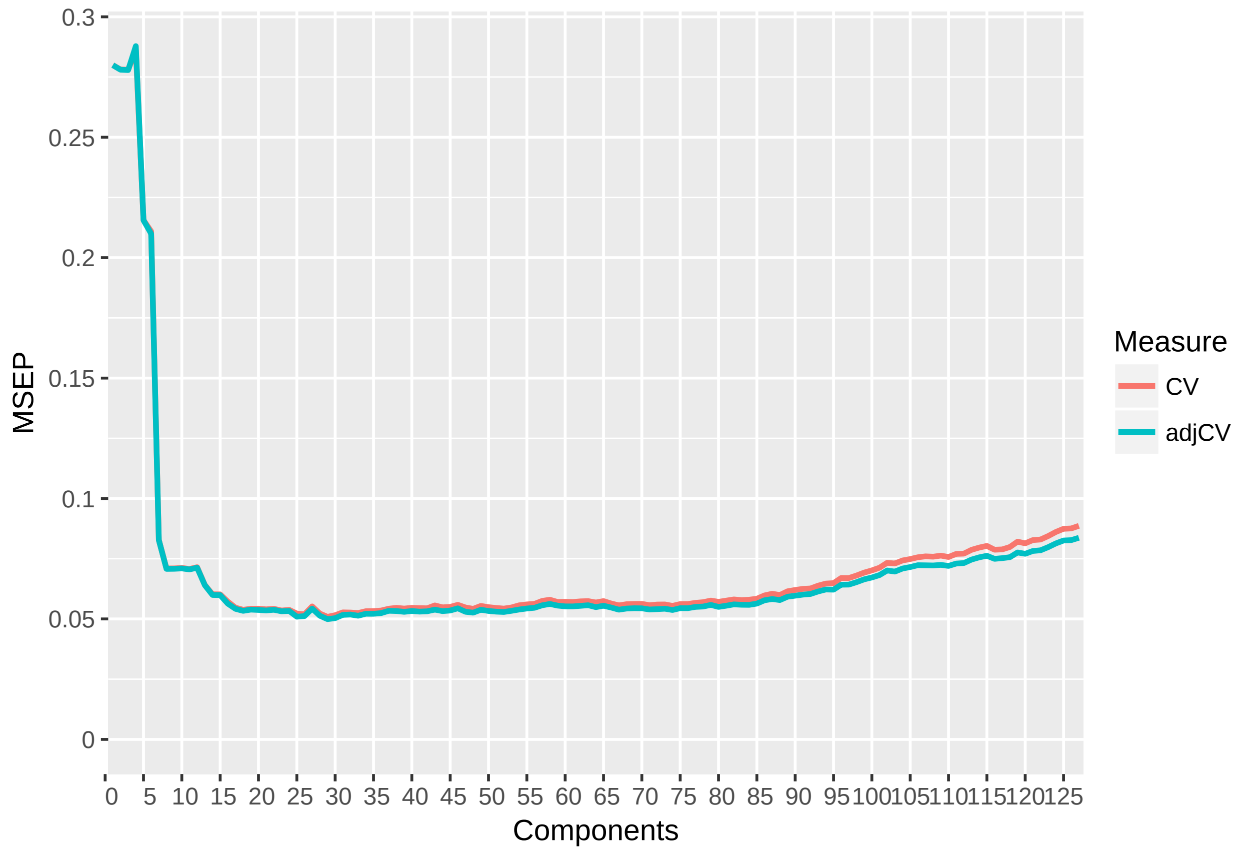


Figure 12: Mean Squared Error of Prediction against number of principal components.

Figure 12 shows how the mean squared error (MSE) changes as the number of components are increased. It is reasonable to select either 7 or around 17 components to have in the model since the MSE barely decrease as more components are added to the model. The choice of 7 or 17 components depends on the trade-off between a simpler model versus likely better predictions.

Appendix

Code for Assignment 1

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

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

library(tree)

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

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

plot(cv_regtree)

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

plot(tree_3leaves)
title("Optimal Regression Tree")
text(tree_3leaves, pretty=0)

library(ggplot2)
library(reshape2)

predicted <- predict(tree_3leaves, state)
plot_data <- data.frame(MET=state$MET, Observed=state$EX, Estimated=predicted)
plot_data <- melt(plot_data, id="MET", variable.name="Data", value.name="EX")

ggplot(plot_data) +
    geom_point(aes(x=MET, y=EX, color=Data))

residuals <- resid(tree_3leaves)
plot_data <- data.frame(resid=residuals)

ggplot(plot_data) +
    xlab("Residual") +
    ylab("Frequency") +
    geom_histogram(aes(resid), bins=10)

library(boot)

tree.fun <- function(data, ind){
    data <- data[ind,] #shuffle procedure
    trio <- tree(EX ~ MET, data = data, #fitting
                 control =
                     tree.control( minsize = 8, nobs = nrow(state))
                 )
    trio <- prune.tree(trio, best = 3)
    return( predict(trio, newdata = state) )
}
```

```r
set.seed(12345)
tree.boot <- boot(state, tree.fun, R = 1000)

State.plot <- data.frame(lower= envelope(tree.boot)$point[2,])
State.plot$upper <- envelope(tree.boot)$point[1,]
State.plot$predicted <- predict(tree_3leaves, state)
State.plot$EX <- state$EX
State.plot$MET <- state$MET

ggplot(data = State.plot) +
    geom_point(aes(x = MET, y = EX)) +
    geom_point(aes(x = MET, y = predicted), col = "blue") +
    geom_ribbon(aes(x = MET, ymin = lower, ymax = upper), alpha = 0.2, fill = "blue") +
    theme_void()

rng <- function(data, model) {
    n <- nrow(data)
    newdata <- data.frame(MET=data$MET, EX=data$EX)
    newdata$EX <- rnorm(n, predict(model, newdata=newdata),
                        sd(resid(model)))
    newdata
}

parametric.estimate.cb <- function(formula, original_data, leaves){
    formula <- formula
    original_data <- original_data
    leaves <- leaves

    function(data) {
        fit <- tree(formula, data=data, split="deviance",
                    control=tree.control(nobs=nrow(original_data), minsize=8))
        fit <- prune.tree(fit, best=leaves)
        prediction <- predict(fit, newdata=original_data)
        prediction
    }
}

parametric.estimate.pb <- function(formula, original_data, leaves){
    formula <- formula
    original_data <- original_data
    leaves <- leaves

    function(data) {
        fit <- tree(formula, data=data, split="deviance",
                    control=tree.control(nobs=nrow(original_data), minsize=8))
        fit <- prune.tree(fit, best=leaves)
        prediction <- predict(fit, newdata=original_data)
        rnorm(nrow(data), prediction, sd(resid(fit)))
    }
}

set.seed(12345)
f.cb <- parametric.estimate.cb(formula=EX ~ MET, original_data=state,
                                leaves=3)
fit  <- boot(state, statistic=f.cb, R=1000,
             mle=tree_3leaves, ran.gen=rng, sim="parametric")
confidence_bands <- envelope(fit, level=0.95)

set.seed(12345)
f.pb <- parametric.estimate.pb(formula=EX ~ MET, original_data=state,
                                leaves=3)
fit  <- boot(state, statistic=f.pb, R=1000,
             mle=tree_3leaves, ran.gen=rng, sim="parametric")
prediction_bands <- envelope(fit, level=0.95)

predicted <- predict(optimal_tree, state)
plot_data_est <- data.frame(MET=state$MET, Observed=state$EX, Estimated=predicted)
plot_data_est <- melt(plot_data_est, id="MET", variable.name="Data", value.name="EX")

plot_data_CB <- data.frame(MET=state$MET, CBU=confidence_bands$point[1,],
                           CBL=confidence_bands$point[2,])

plot_data_PB <- data.frame(MET=state$MET, PBU=prediction_bands$point[1,],
                           PBL=prediction_bands$point[2,])

ggplot() +
    geom_point(data=plot_data_est, aes(x=MET, y=EX, color=Data)) +
    geom_ribbon(data=plot_data_CB, aes(x=MET, ymin=CBL, ymax=CBU),
                color="red", alpha=0.1, fill="red") +
    geom_ribbon(data=plot_data_PB, aes(x=MET, ymin=PBL, ymax=PBU),
                color="blue", alpha=0.1, fill="blue")
```

Code for Assignment 2

```
library(ggplot2)
library(fastICA)
library(pls)
library(reshape2)

data <- read.csv2("../data/NIRSpectra.csv")

X <- scale(data[, -ncol(data)])
y <- data[, ncol(data)]

pca <- prcomp(X)

lambda <- pca$sdev^2
variances <- lambda / sum(lambda)

var99_comp_count <- which.max(cumsum(variances * 100) > 99)
components <- as.data.frame(pca$x[, 1:var99_comp_count])

pc_comps <- 1:10
plot_data <- data.frame(x=pc_comps, Variance=variances[pc_comps])
```

```r
ggplot(plot_data, aes(x=x, y=Variance)) +
    geom_bar(stat="identity") +
    scale_x_discrete(limits=pc_comps, labels=as.numeric(pc_comps)) +
    xlab("Principal Component")

ggplot(components) +
    geom_point(aes(x=PC1, y=PC2)) +
    scale_x_continuous(breaks=pretty(components$PC1, n=6)) +
    scale_y_continuous(breaks=pretty(components$PC2, n=6))

U <- pca$rotation

plot_data <- data.frame(x=1:nrow(U), PC1=U[, 1], PC2=U[, 2])
plot_data <- melt(plot_data, id="x")
names(plot_data) <- c("Index", "Component", "Value")
xlimits <- seq(0, nrow(U), by=10)

ggplot(plot_data) +
    geom_line(aes(x=Index, y=Value, color=Component), show.legend=FALSE) +
    scale_x_discrete(limits=xlimits) +
    facet_grid(Component ~ ., scales="free")

nir <- read.csv2("../data/NIRSpectra.csv", sep=";", header=T)
nir2 <- as.data.frame(scale(nir[, 1:126]))

set.seed(12345)
ica <- fastICA(nir2, n.comp=2,alg.typ="parallel",
               fun="logcosh", alpha=1, row.norm=F,verbose=F )

W_prime <- ica$K %*% ica$W
components <- as.data.frame(ica$S)
colnames(components) <- c("IC1", "IC2")

plot_data <- data.frame(Index=1:nrow(W_prime), IC1=W_prime[, 1], IC2=W_prime[, 2])
plot_data <- melt(plot_data, id="Index", variable.name="Component", value.name="Value")
xlimits <- seq(0, nrow(W_prime), by=10)

ggplot(plot_data) +
    geom_line(aes(x=Index, y=Value, color=Component), show.legend=FALSE) +
    scale_x_discrete(limits=xlimits) +
    facet_grid(Component ~ ., scales="free")

ggplot(components) +
    geom_point(aes(x=IC1, y=IC2)) +
    scale_x_continuous(breaks=pretty(components$IC1, n=6)) +
    scale_y_continuous(breaks=pretty(components$IC2, n=6))

set.seed(12345)
pcrfit <- pcr(Viscosity ~ ., data=data, scale=TRUE)
cvpcrfit <- crossval(pcrfit, segments=10, segment.type="random")

cv_scores <- t(matrix(MSEP(cvpcrfit)$val, nrow=2))
plot_data <- data.frame(cbind(1:ncol(data), cv_scores))
```

```r
colnames(plot_data) <- c("Components", "CV", "adjCV")
plot_data <- melt(plot_data, id="Components",
                  variable.name="Measure", value.name="MSEP")
xlimits <- seq(0, ncol(data), by=5)
ylimits <- seq(0, max(plot_data$MSEP) + 0.05, by=0.05)

ggplot(plot_data) +
    geom_line(aes(x=Components, y=MSEP, color=Measure), size=1) +
    scale_x_discrete(limits=xlimits) +
    scale_y_continuous(breaks=ylimits, labels=ylimits,
                       limits=c(0, max(plot_data$MSEP)))
```