Introduction to Machine Learning

Lab 4

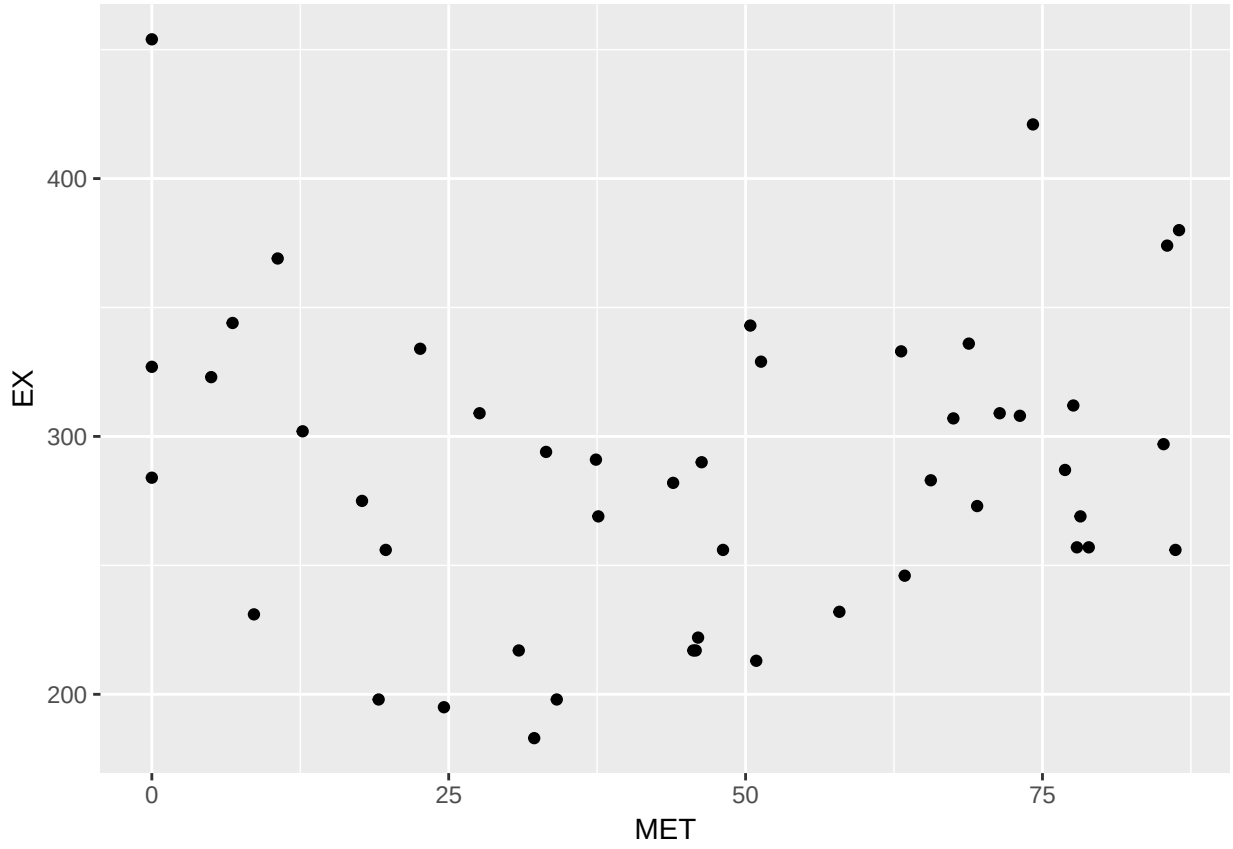
Rasmus Holm

2016-11-30

Contents

Assignment 1

1



The data looks like it could be modelled reasonably by a cubic spline function.

2

Here I have modelled the data using a regression tree by letting cross-validation choose the number of leaves.

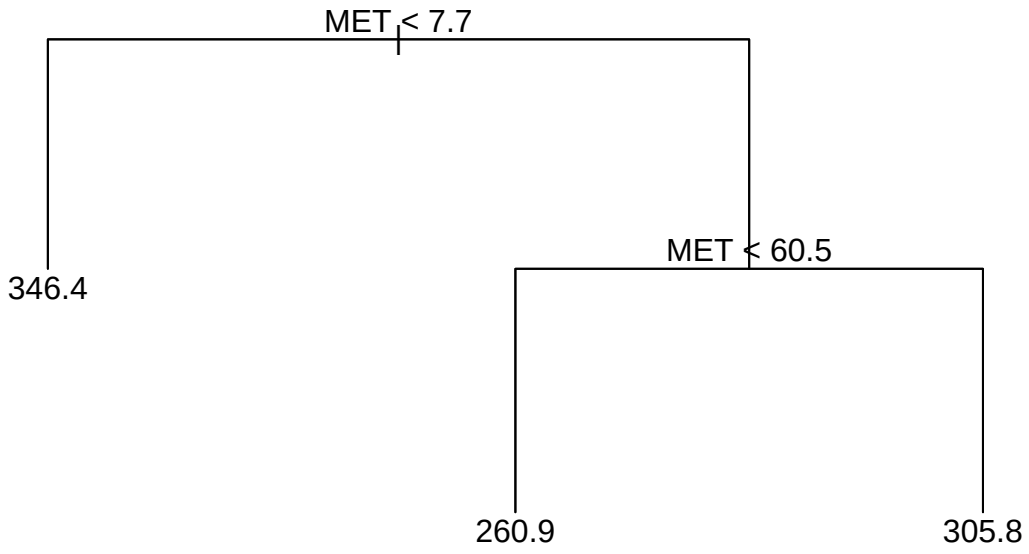


Figure 1: Regression tree fit.

Figure 1 shows the resulting tree and it turned out that the optimal tree has 3 leaves.

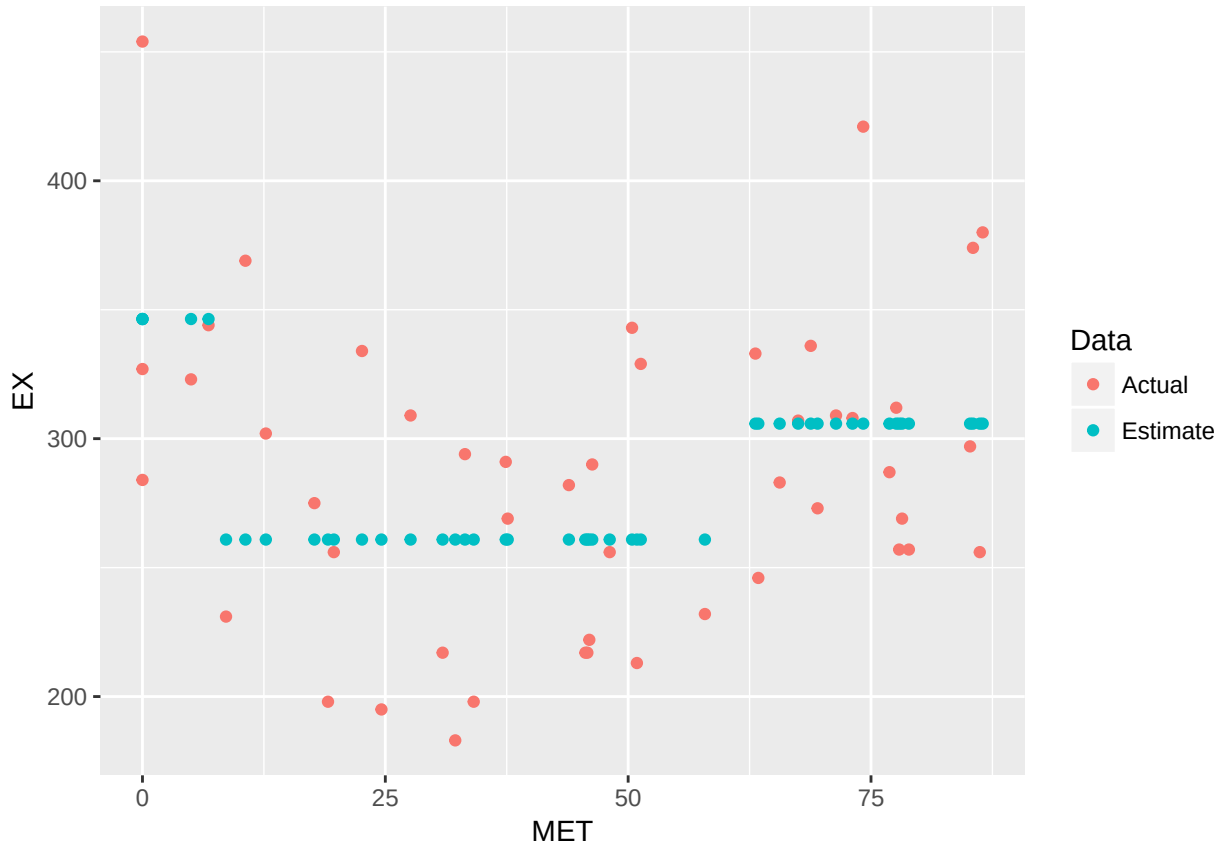


Figure 2: Shows the fitted values from the regression tree.

The fitted values can be seen in figure 2 and they have a decent resemblance of the original data but the model seems to be too simplistic to capture the underlying distribution the data set was generated from.

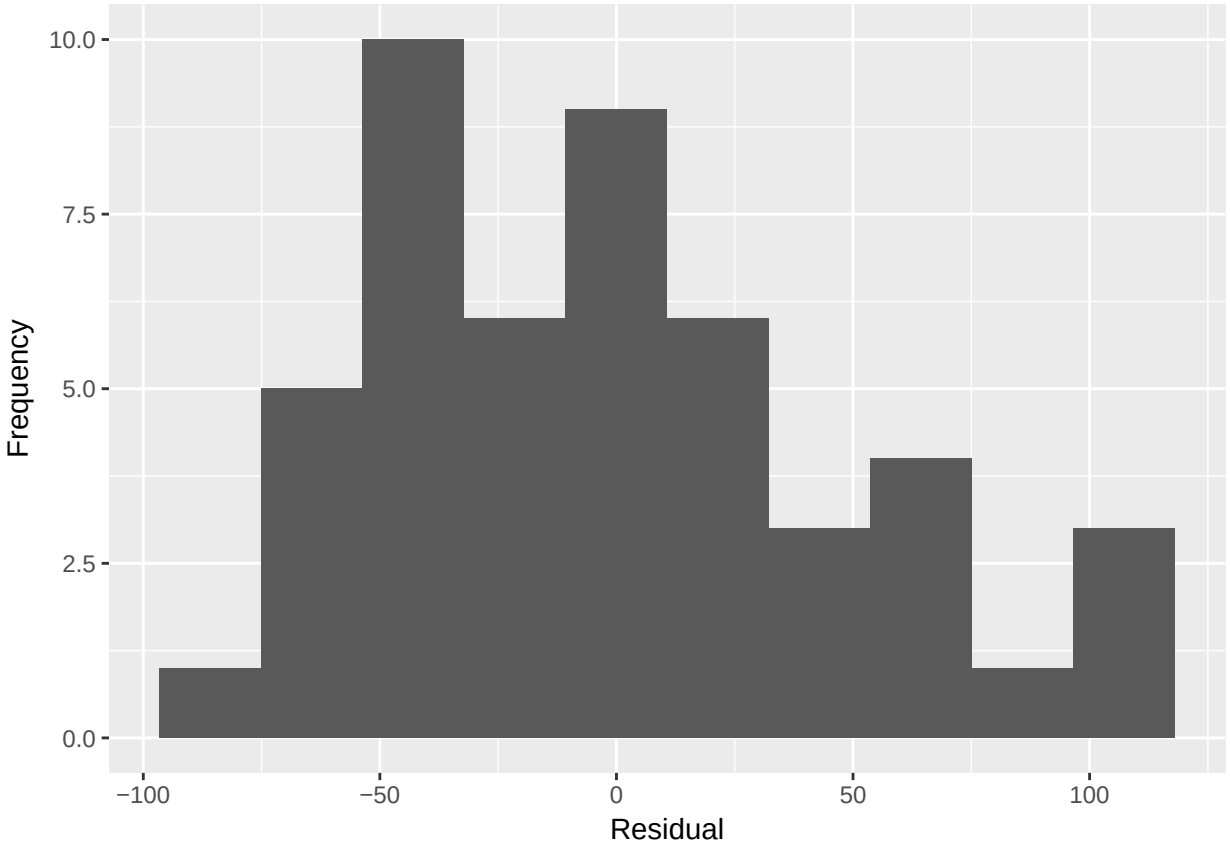


Figure 3: The distribution of the residuals from the regression tree.

We can see from figure 3 that the residuals have approximately a Gaussian distribution or some form of right-tailed Gamma distribution. Since there are only 48 observations in the data set it is difficult to get a good approximation of the distribution.

3

In this exercise I have used non-parametric bootstrap in order to approximate the 95% confidence bands for the regression model.

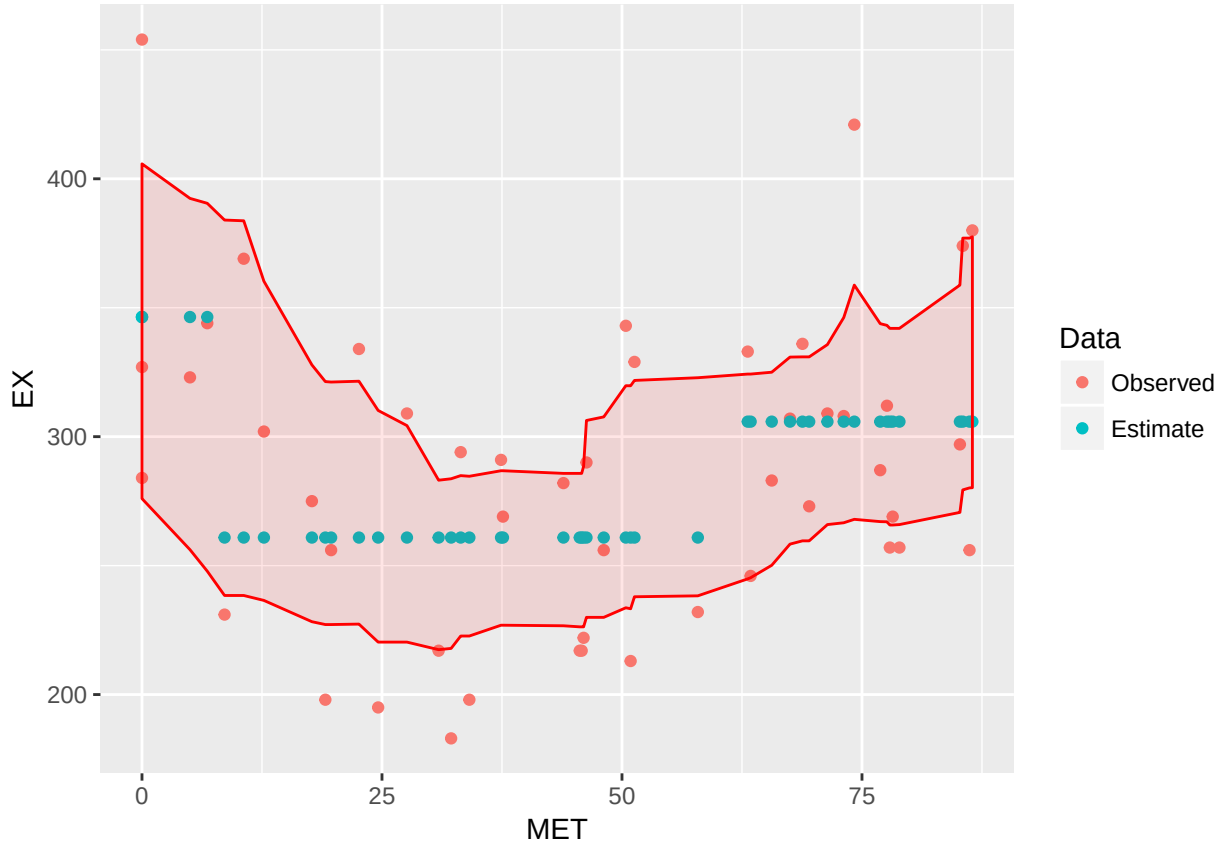


Figure 4: The confidence bands found by non-parametric bootstrap after 1000 samples.

The confidence bands are quite wide in figure 4 which indicates that the model cannot capture the data particular well since it is unsure where its mean prediction should be. The model is more certain in the middle values of the MET feature and very uncertain at low values of MET.

We can also see that the confidence bands are bumpy which I would assume have to do with the data density in our data set. Since we only have 48 observations it is not possible to have a high granularity in the confidence bands but by increasing the number of bootstrap samples it will probably become more smooth. However, it will never be completely smooth since there are simply not enough data to be able to capture the models uncertainty for all x-values.

4

Here I have instead used the parametric bootstrap where I 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.

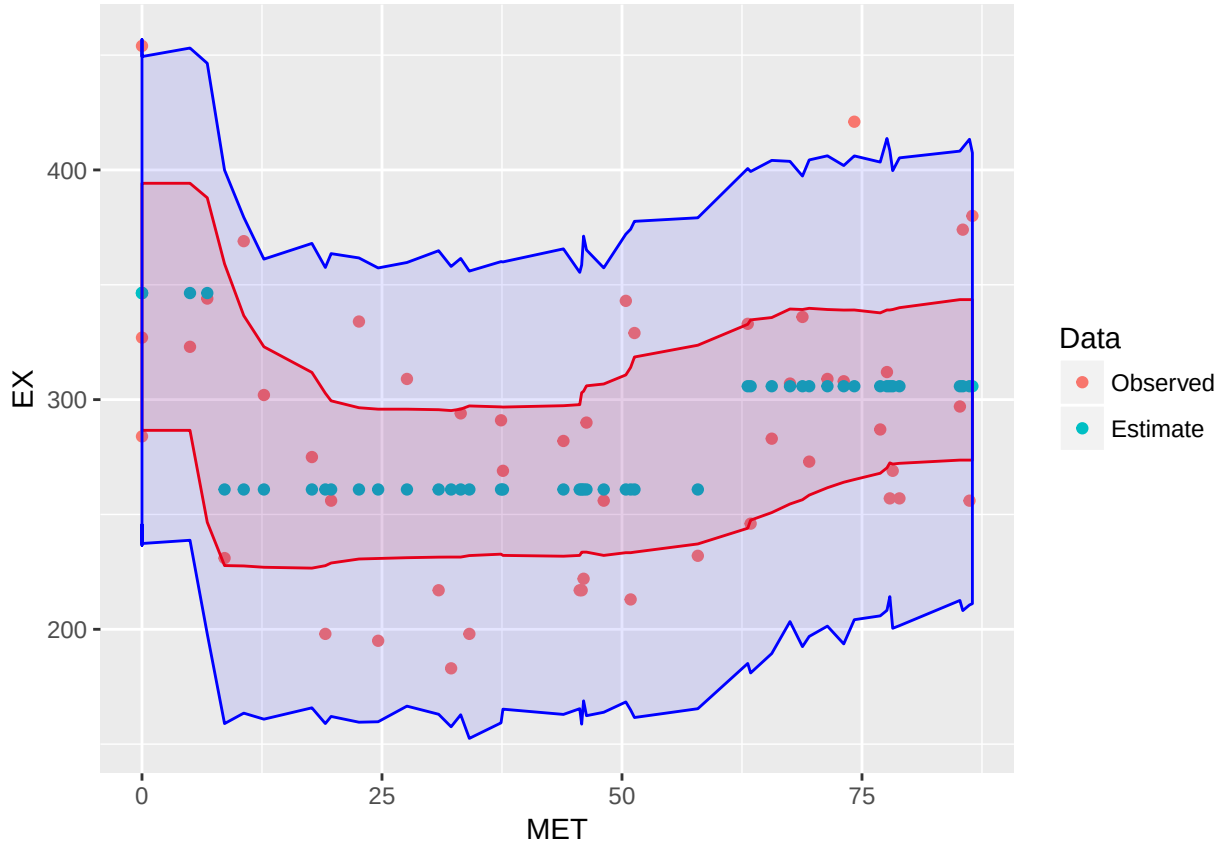


Figure 5: The confidence and prediction bands found by parametric bootstrap after 1000 samples. The red bands are the confidence bands and the blue bands are the prediction bands.

We can see from figure 5 that, as expected, the prediction bands are wider than the confidence bands because not only do we consider the uncertainty of the regression model but also the uncertainty of the data itself. Compared to the previous result, the confidence bands are narrower and much more consistent over the whole range of the MET feature. The bands are also a lot smoother than previously but still pretty wide which indicates that the regression model is not very well suited for this data set. Two observations are outside the prediction bands and 5% of 48 is approximately 2 so it is as we would expect.

5

Given that the residuals looked to have a Gaussian/Gamma distribution it would be more appropriate to use parametric bootstrap so we could explicitly guide the method to approximate the confidence and prediction bands.

Assignment 2

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

1

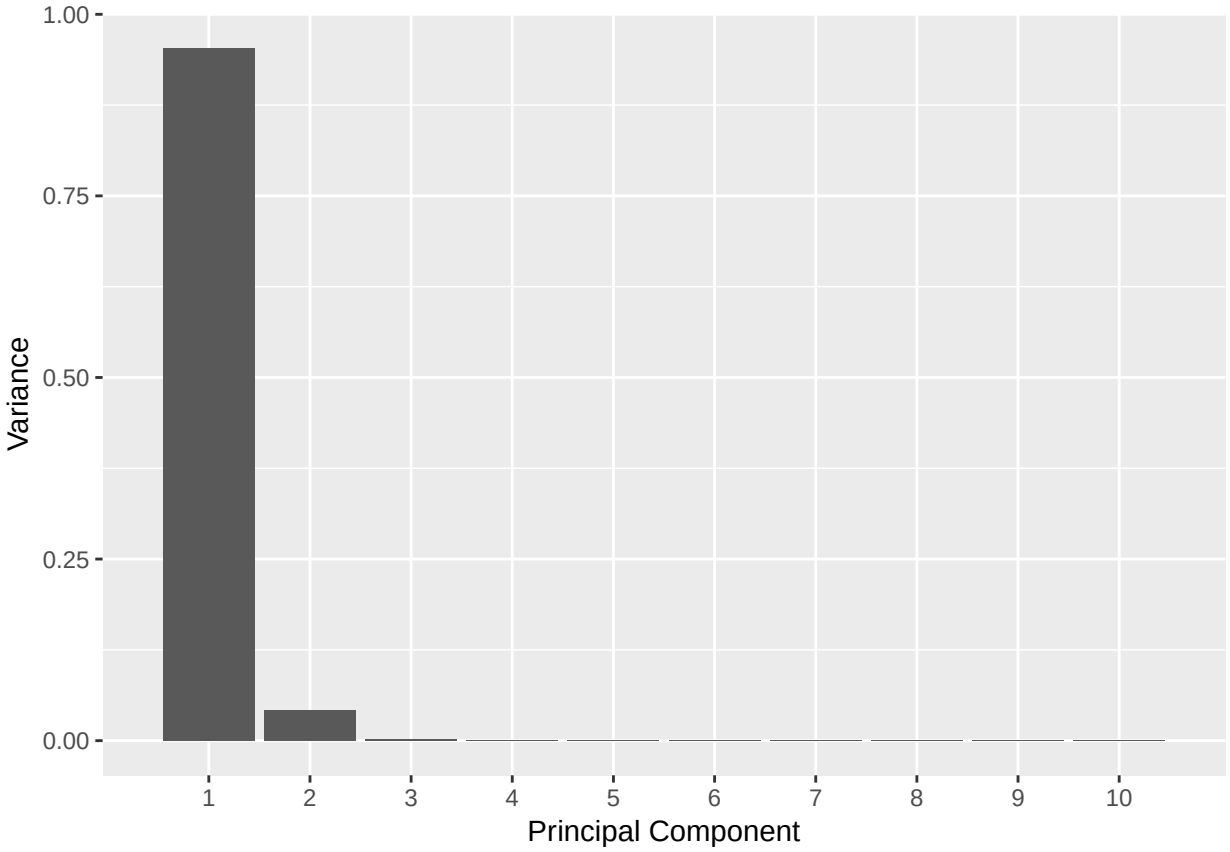


Figure 6: Variance explained by the first 10 principal components.

Figure 6 shows that basically all the variance can be explained by the first 5 principal components (PCs). The first two are clearly the most important and can be extracted without much loss of information, combined they explain over 99% of the variance.

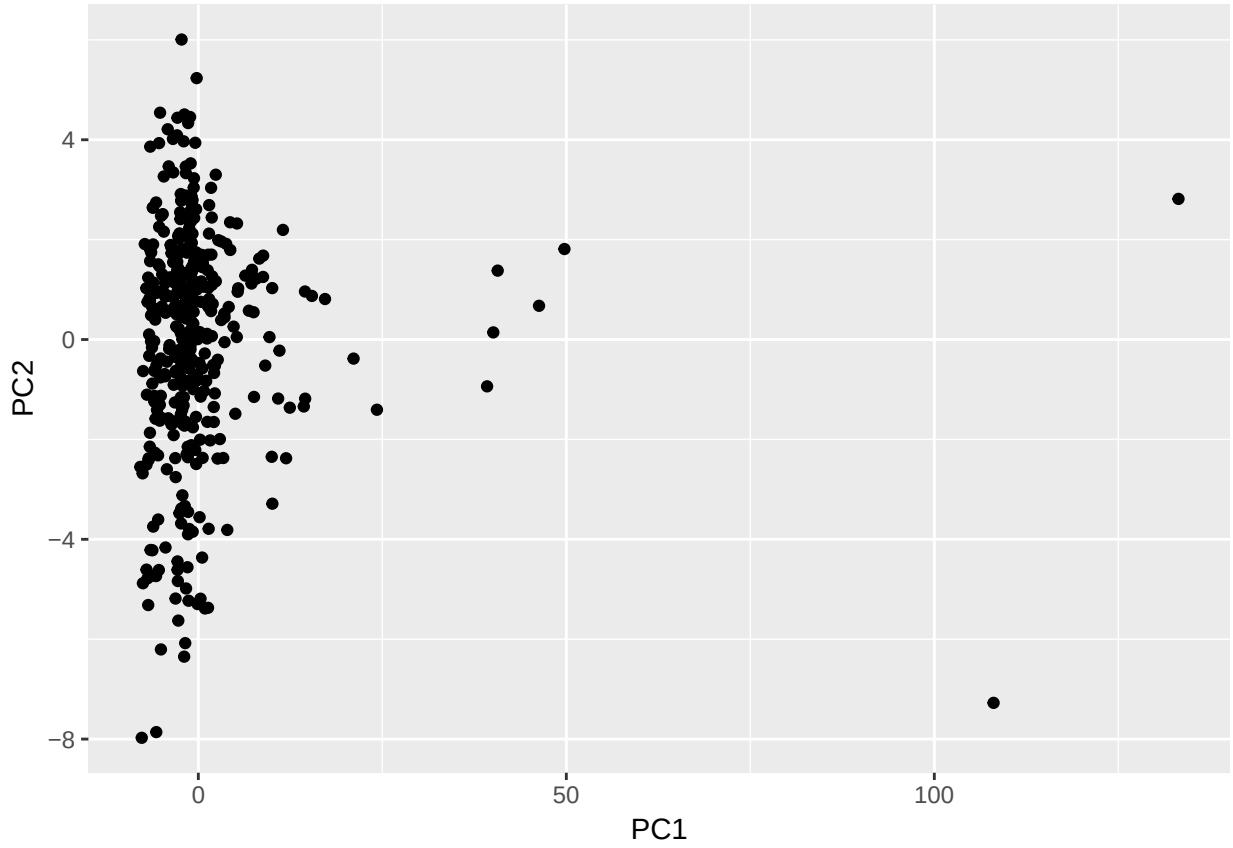


Figure 7: The data projected onto the first two principal components.

From figure 7 there are clear outliers with very high values in the first PC. Most observations have a low value but rather spread out values in the second PC which is surprising since the first PC are supposed to explain most of the variance. However, since the scales are very different it make sense.

2

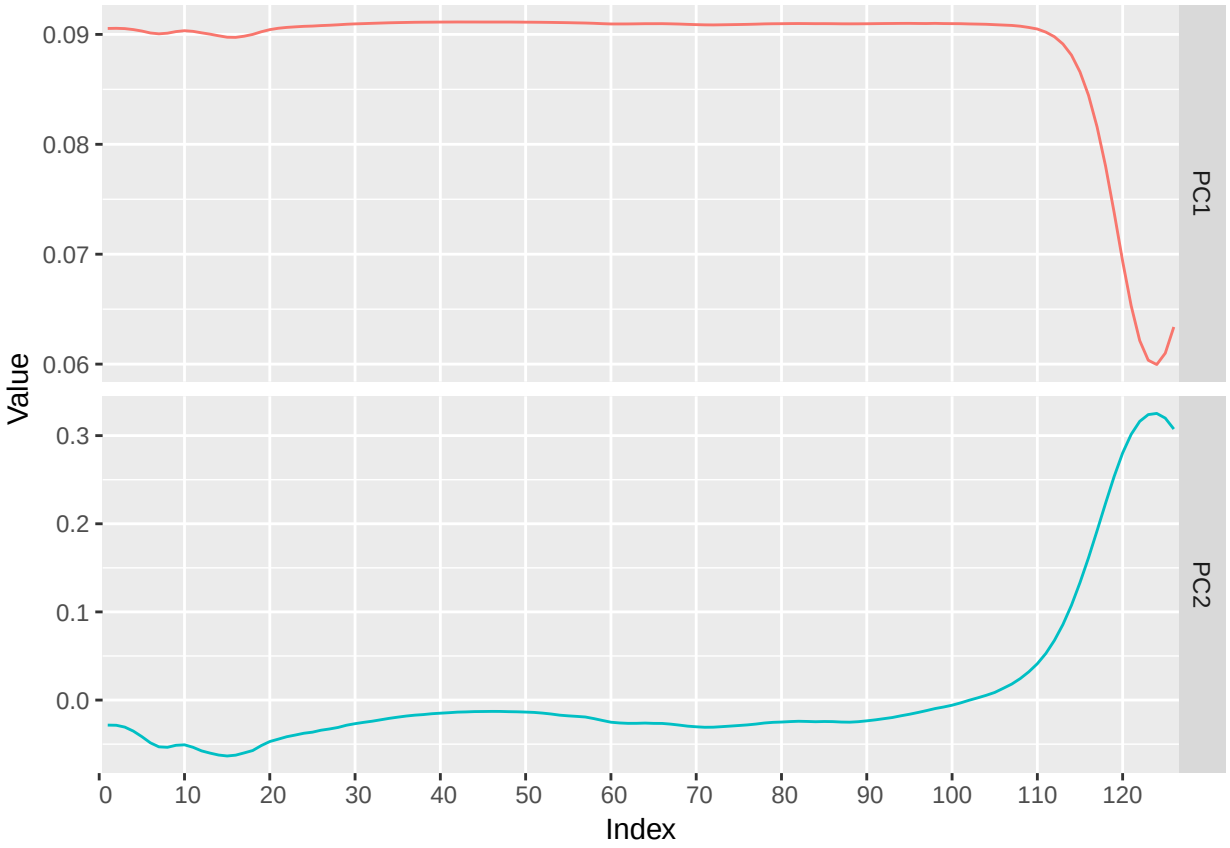


Figure 8: Trace plot of the first two principal components.

From the trace plot in figure 8 we can observe that the first and second PCs are combinations of basically all the variables to different extent. The first PC has relatively high combination of all variables but slightly less of the last ones while the second PC is mostly a combination of the last ones.

3

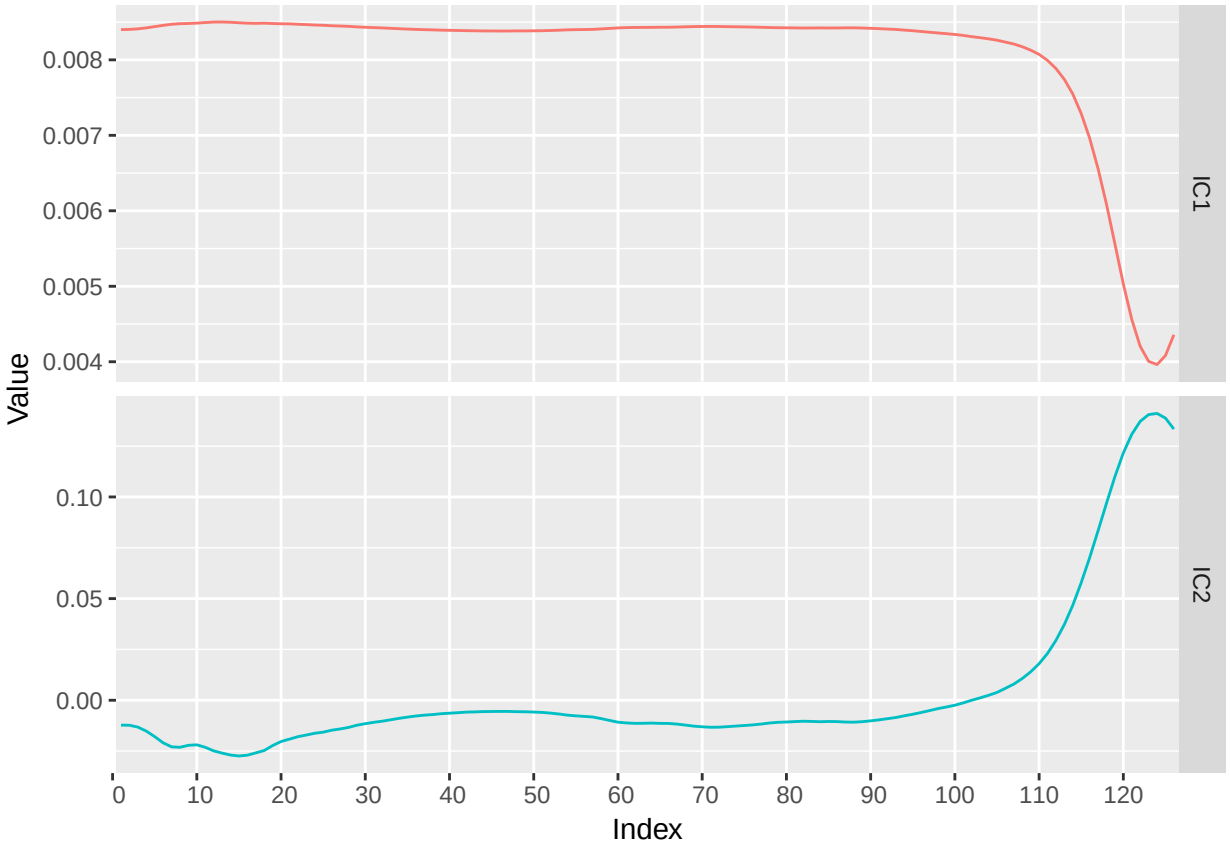


Figure 9: Traceplot of the first two independent components.

Similarly from the PC components in figure 8, we can see that the independent components (ICs) in figure 9 have opposite shapes. The second IC is mostly a combination of the last variables while the first IC explains more information from the other variables.

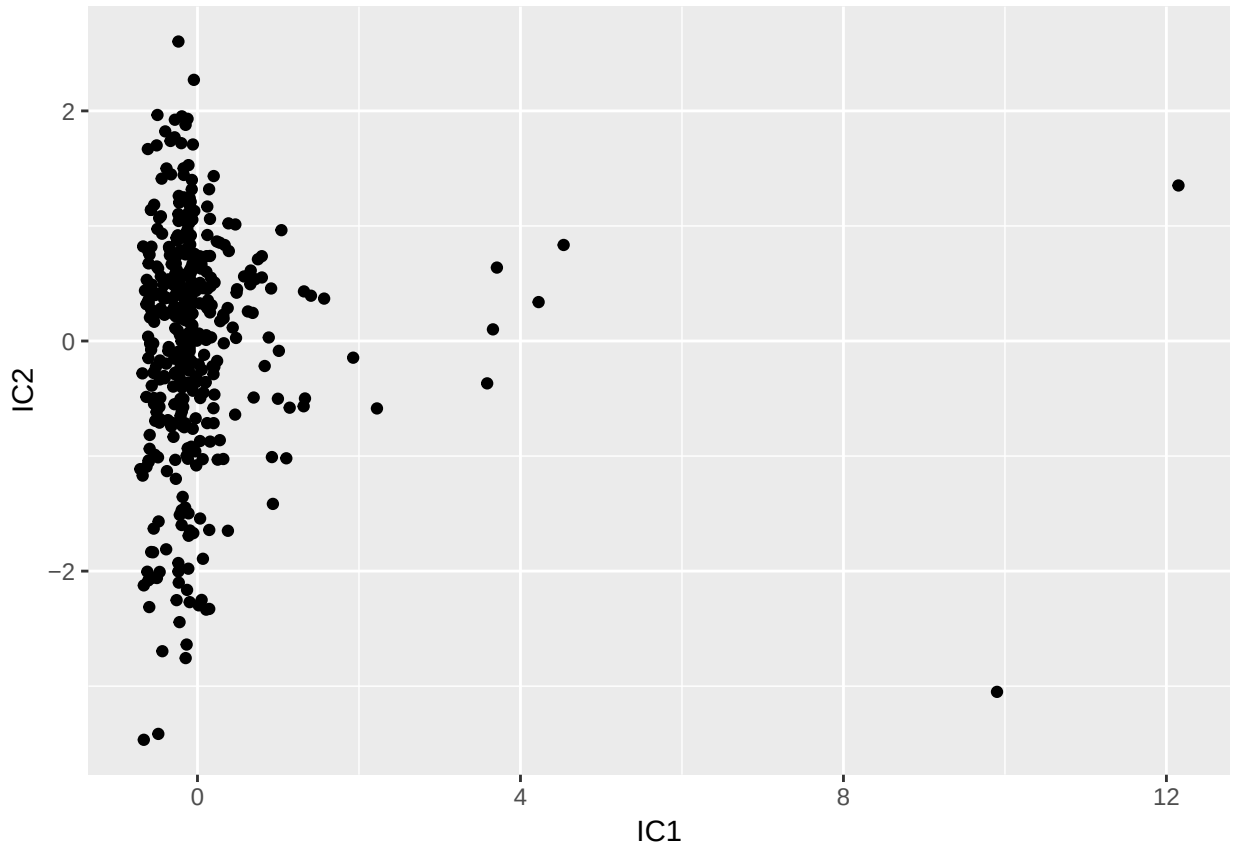


Figure 10: The data projected onto the first two independent components.

The scores in figure 10 have similar shape to that of the principal components (fig. 7) but with very different scales. Outliers can clearly be detected in the data.

4

In this exercise I have used principal component regression (PCR) in order to estimate the viscosity.

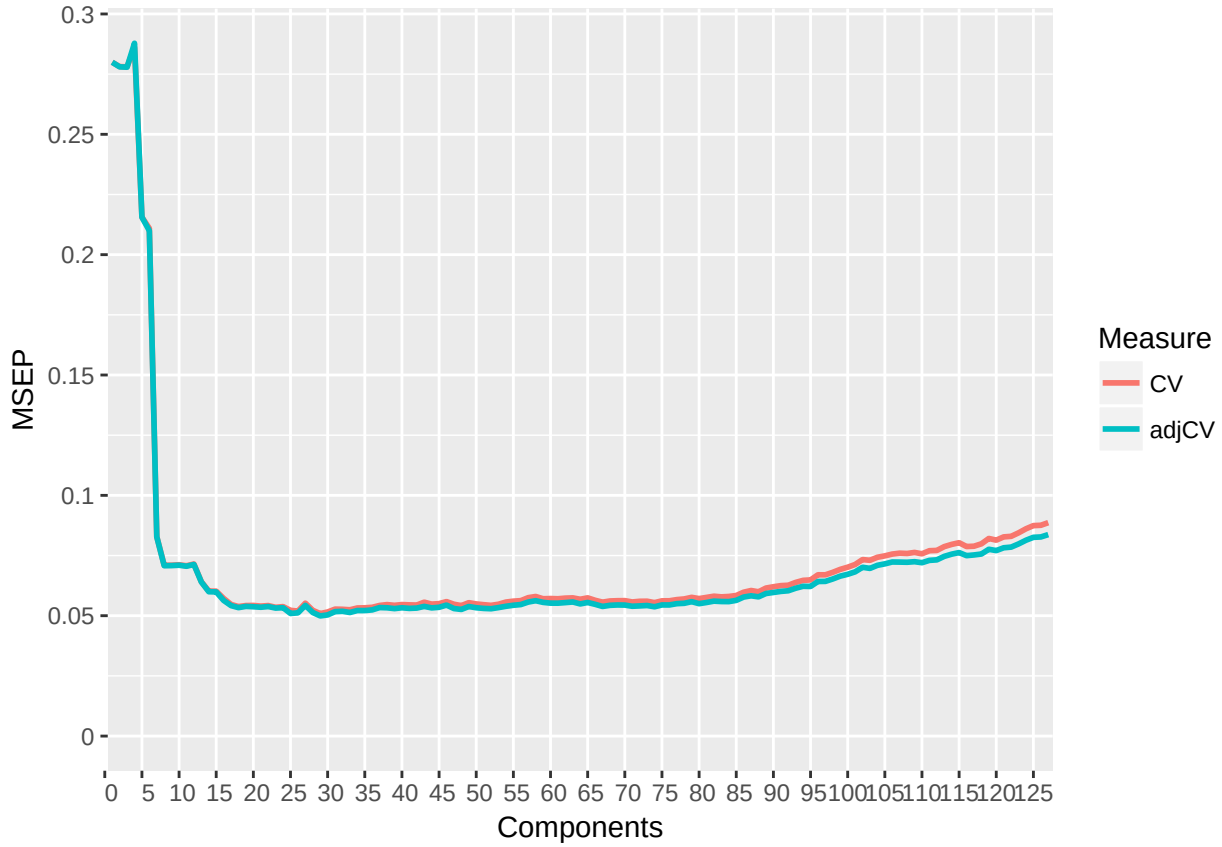


Figure 11: Mean Squared Error of Prediction against number of principal components.

Figure 11 shows how the mean squared error changes when varying the number of PCs in PCR using cross validation. The optimal number of PCs are probably either around 7 or 17 depending on how much you value fewer features and a simpler model.

Appendix

Code for Assignment 1

```
library(ggplot2)
library(tree)
library(reshape2)
library(boot)

data <- read.csv2("../data/State.csv", header=TRUE, sep=";")
data <- data[order(data$MET),]
ggplot(data) +
    geom_point(aes(x=MET, y=EX))

treefit <- tree(EX ~ MET, data=data, split="deviance",
                control=tree.control(nobs=nrow(data), minsize=8))

set.seed(12345)
treefit.cv <- cv.tree(treefit, FUN=prune.tree, K=10)
optimal_leaf_count <- treefit.cv$size[which.min(treefit.cv$dev)]

optimal_tree <- prune.tree(treefit, best=optimal_leaf_count)
plot(optimal_tree)
text(optimal_tree, pretty=0)
predicted <- predict(optimal_tree, data)
plot_data <- data.frame(MET=data$MET, Actual=data$EX, Estimate=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(optimal_tree)
plot_data <- data.frame(resid=residuals)

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

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

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

```
f <- nonparametric.estimate(formula=EX ~ MET, original_data=data,
                            leaves=optimal_leaf_count)

set.seed(12345)
fit <- boot(data, f, R=1000)
confidence_bands <- envelope(fit, level=0.95)
predicted <- predict(optimal_tree, data)
plot_data_est <- data.frame(MET=data$MET, Observed=data$EX,
                            Estimate=predicted)
plot_data_est <- melt(plot_data_est, id="MET",
                      variable.name="Data", value.name="EX")

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

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=data,
                               leaves=optimal_leaf_count)
fit  <- boot(data, statistic=f.cb, R=1000,
             mle=optimal_tree, 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=data,
                               leaves=optimal_leaf_count)
fit  <- boot(data, statistic=f.pb, R=1000,
             mle=optimal_tree, ran.gen=rng, sim="parametric")
prediction_bands <- envelope(fit, level=0.95)
predicted <- predict(optimal_tree, data)
plot_data_est <- data.frame(MET=data$MET, Observed=data$EX, Estimate=predicted)
plot_data_est <- melt(plot_data_est, id="MET", variable.name="Data", value.name="EX")

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

plot_data_PB <- data.frame(MET=data$MET, PBU=prediction_bands$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)

## Eigenvalues
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])

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")


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")

set.seed(12345)
ica <- fastICA(X, var99_comp_count, alg.typ="parallel", fun="logcosh", alpha=1,
               method="R", row.norm=FALSE, maxit=200, tol=1e-06, verbose=FALSE)

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))

set.seed(12345)
pcrfit <- pcr(Viscosity ~ ., data=data, scale=TRUE)
cvpcrfit <- crossval(pcrfit, segments=10, segment.type="random")
cv_scores <- t(matrix(MSEP(cvpcrfit)$val, nrow=2))
plot_data <- data.frame(cbind(1:ncol(data), cv_scores))
colnames(plot_data) <- c("Components", "CV", "adjCV")
plot_data <- melt(plot_data, id="Components",
                  variable.name="Measure", value.name="MSEP")
xlimits <- seq(0, ncol(data), by=5)
ylimits <- seq(0, max(plot_data$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)))
```