

# Introduction to Machine Learning

Lab 4

*Rasmus Holm*

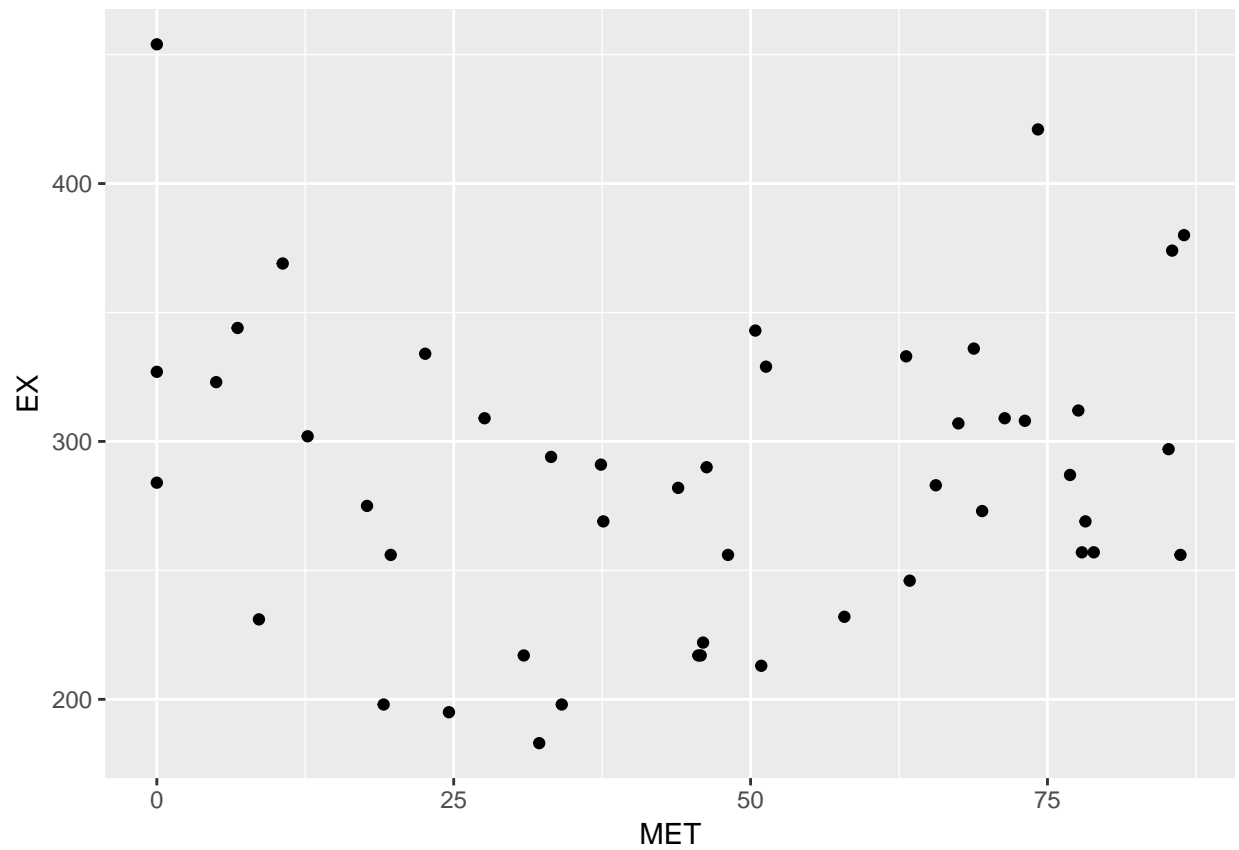
*2016-11-25*

## Contents

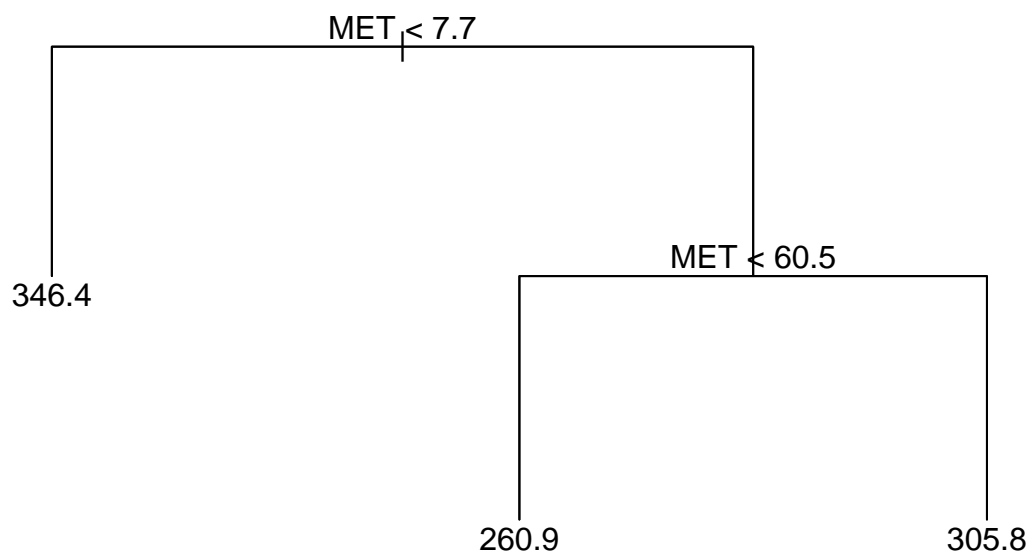
|                                 |           |
|---------------------------------|-----------|
| <b>Assignment 1</b>             | <b>2</b>  |
| 1 . . . . .                     | 2         |
| 2 . . . . .                     | 3         |
| 3 . . . . .                     | 5         |
| 4 . . . . .                     | 5         |
| 5 . . . . .                     | 5         |
| <b>Assignment 2</b>             | <b>6</b>  |
| 1 . . . . .                     | 6         |
| 2 . . . . .                     | 9         |
| 3 . . . . .                     | 10        |
| 4 . . . . .                     | 12        |
| <b>Appendix</b>                 | <b>13</b> |
| Code for Assignment 1 . . . . . | 13        |
| Code for Assignment 2 . . . . . | 13        |

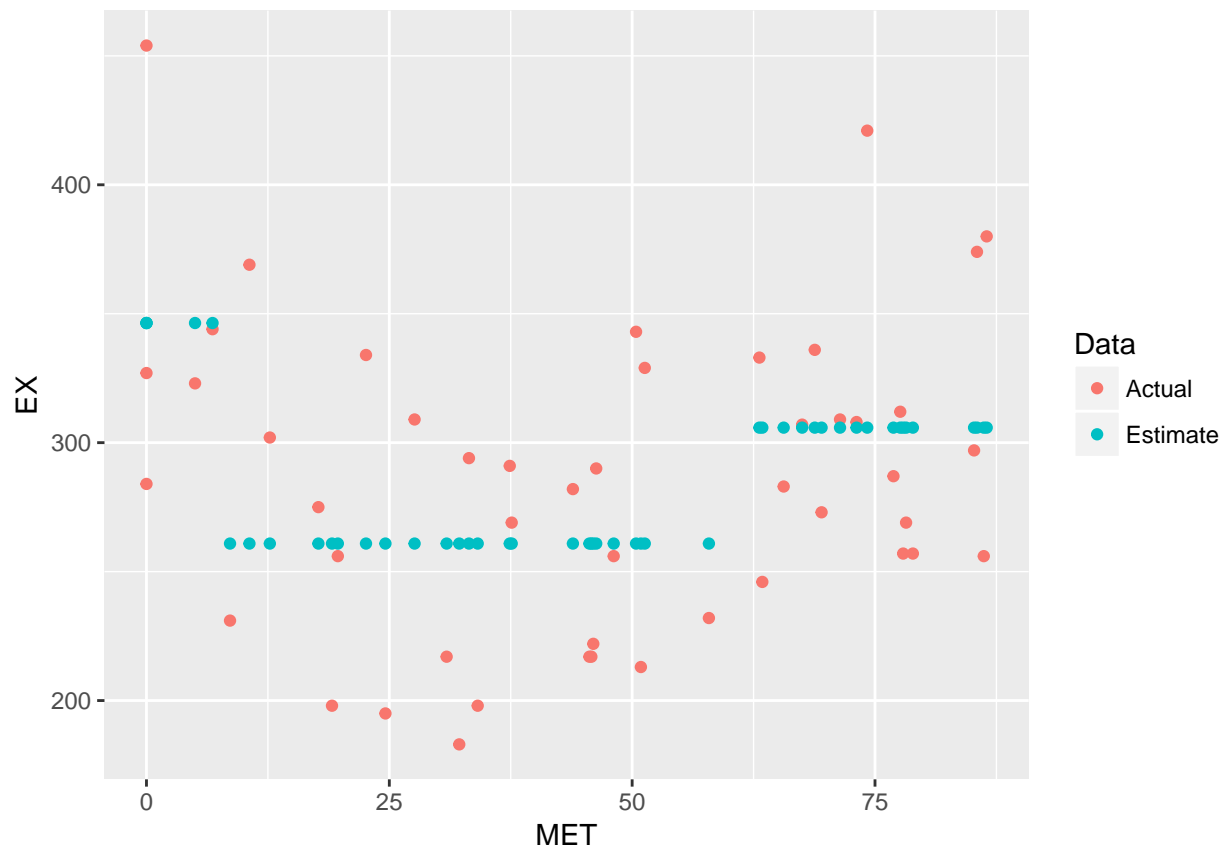
# Assignment 1

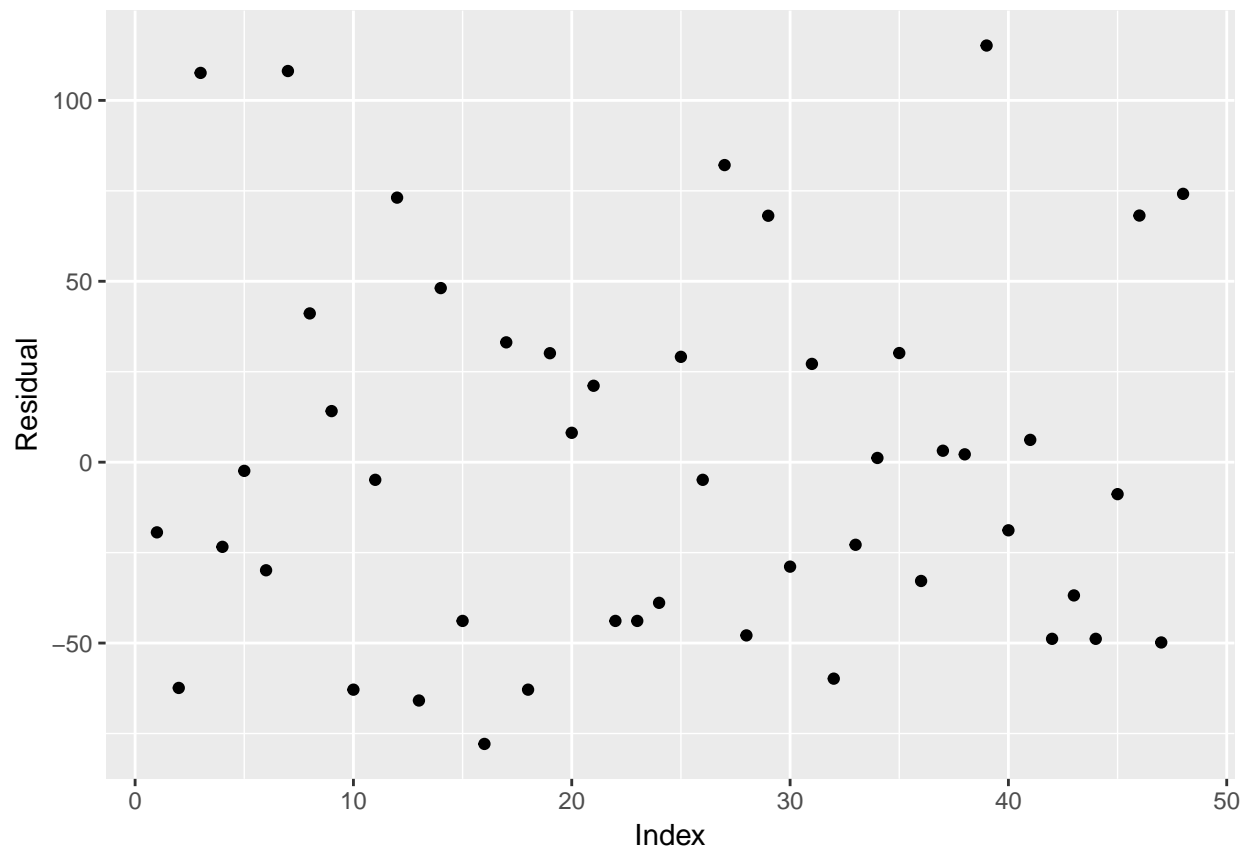
1



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







3

4

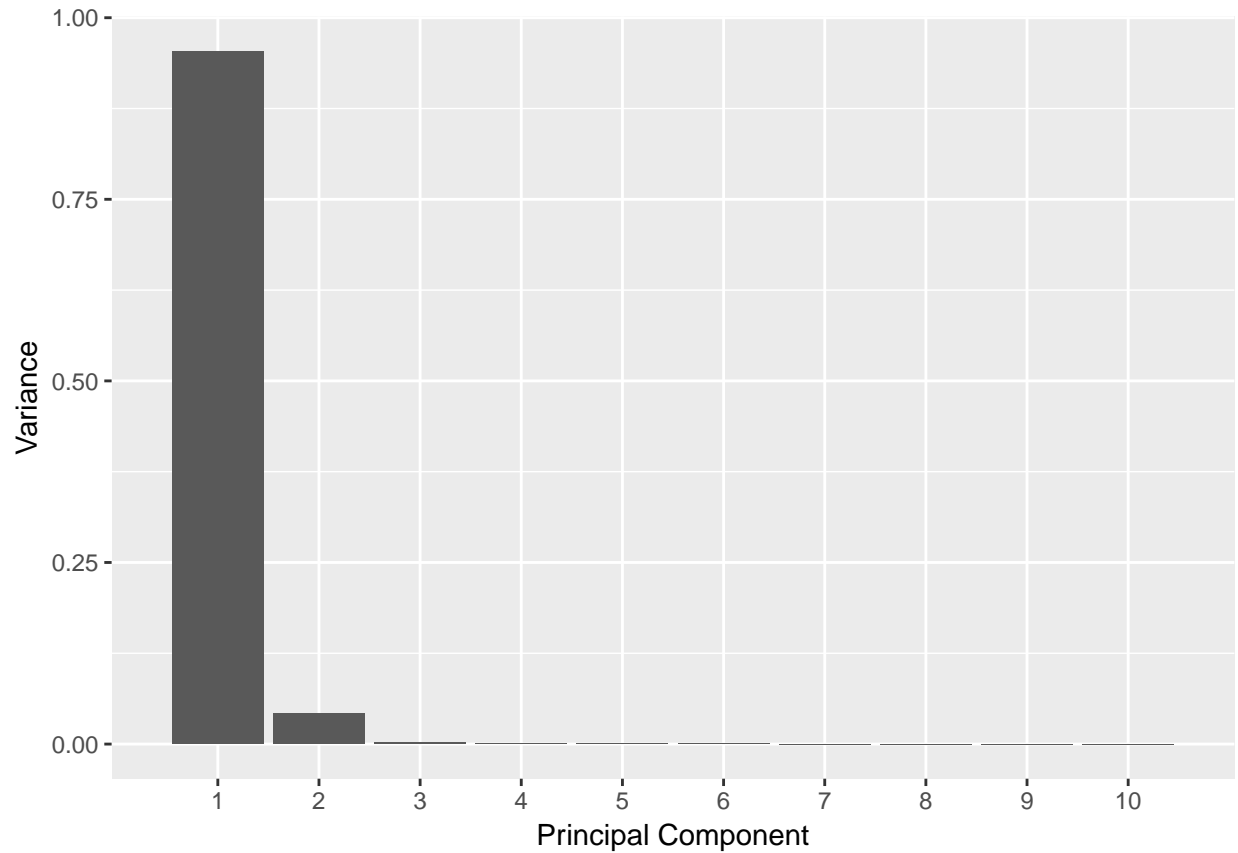
5

## 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

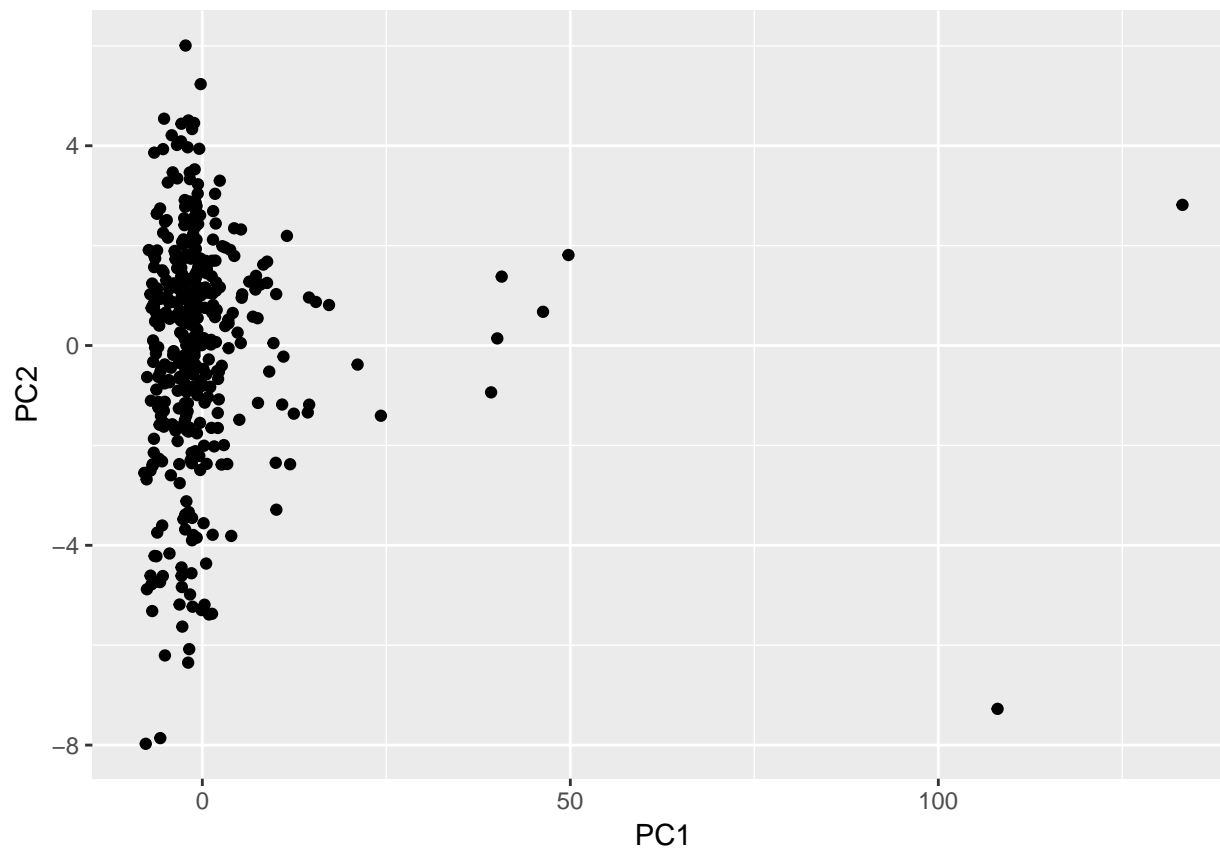
1

```
#> [1] "95.385" "4.229" "0.186" "0.084" "0.072" "0.021" "0.007"
#> [8] "0.003" "0.003" "0.002" "0.001" "0.001" "0.001" "0.001"
#> [15] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
#> [22] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
#> [29] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
#> [36] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
#> [43] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
#> [50] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
#> [57] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
#> [64] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
#> [71] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
#> [78] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
#> [85] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
#> [92] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
#> [99] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
#> [106] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
#> [113] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
#> [120] "0.000" "0.000" "0.000" "0.000" "0.000" "0.000" "0.000"
#> [1] "0.954" "0.996" "0.998" "0.999" "1.000" "1.000" "1.000" "1.000"
#> [9] "1.000" "1.000" "1.000" "1.000" "1.000" "1.000" "1.000" "1.000"
#> [17] "1.000" "1.000" "1.000" "1.000" "1.000" "1.000" "1.000" "1.000"
#> [25] "1.000" "1.000" "1.000" "1.000" "1.000" "1.000" "1.000" "1.000"
#> [33] "1.000" "1.000" "1.000" "1.000" "1.000" "1.000" "1.000" "1.000"
#> [41] "1.000" "1.000" "1.000" "1.000" "1.000" "1.000" "1.000" "1.000"
#> [49] "1.000" "1.000" "1.000" "1.000" "1.000" "1.000" "1.000" "1.000"
#> [57] "1.000" "1.000" "1.000" "1.000" "1.000" "1.000" "1.000" "1.000"
#> [65] "1.000" "1.000" "1.000" "1.000" "1.000" "1.000" "1.000" "1.000"
#> [73] "1.000" "1.000" "1.000" "1.000" "1.000" "1.000" "1.000" "1.000"
#> [81] "1.000" "1.000" "1.000" "1.000" "1.000" "1.000" "1.000" "1.000"
#> [89] "1.000" "1.000" "1.000" "1.000" "1.000" "1.000" "1.000" "1.000"
#> [97] "1.000" "1.000" "1.000" "1.000" "1.000" "1.000" "1.000" "1.000"
#> [105] "1.000" "1.000" "1.000" "1.000" "1.000" "1.000" "1.000" "1.000"
#> [113] "1.000" "1.000" "1.000" "1.000" "1.000" "1.000" "1.000" "1.000"
#> [121] "1.000" "1.000" "1.000" "1.000" "1.000" "1.000" "1.000" "1.000"
```



*Figure 1: Variance explained by the first 10 principal components.*

Figure 1 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 should be extracted without much loss of information, combined they explain over 99% of the variance.

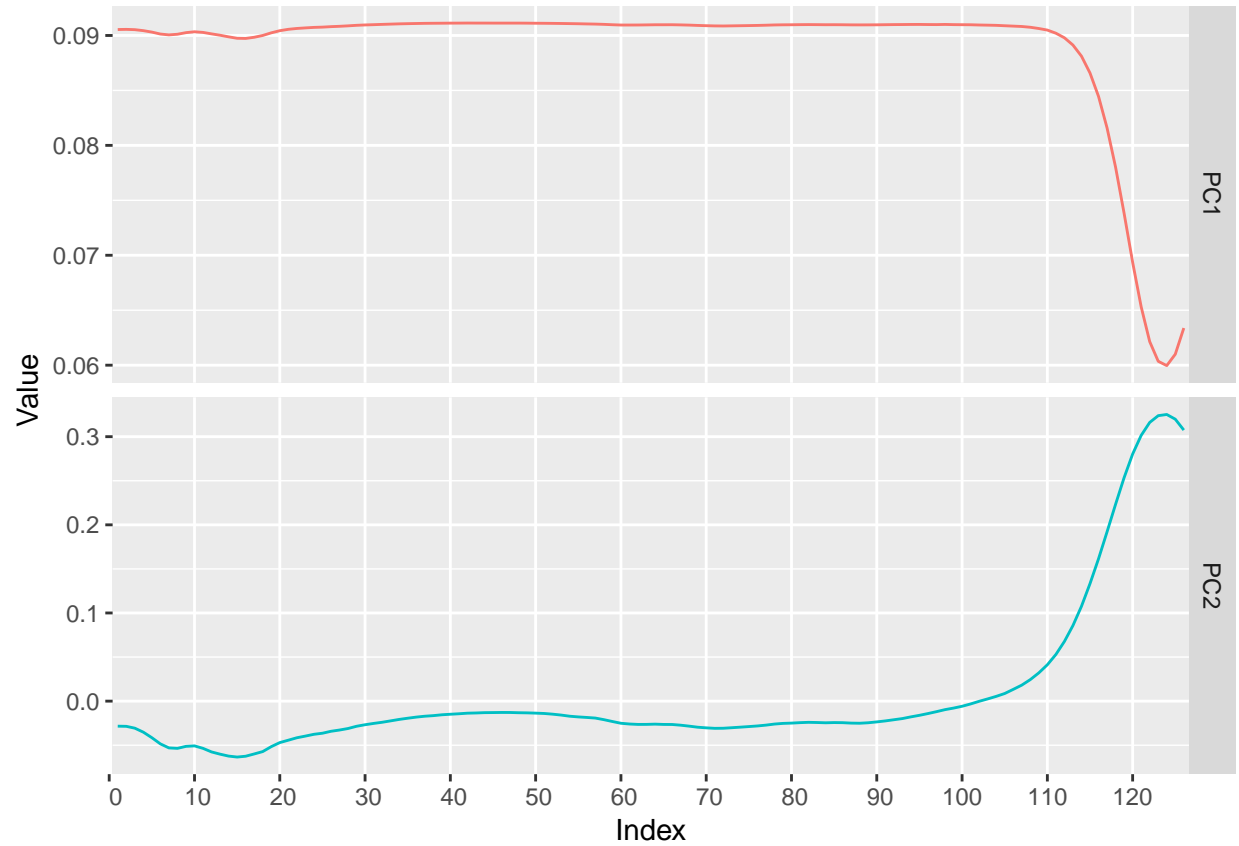


*Figure 2: The data projected onto the first two principal components.*

From figure 2 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.

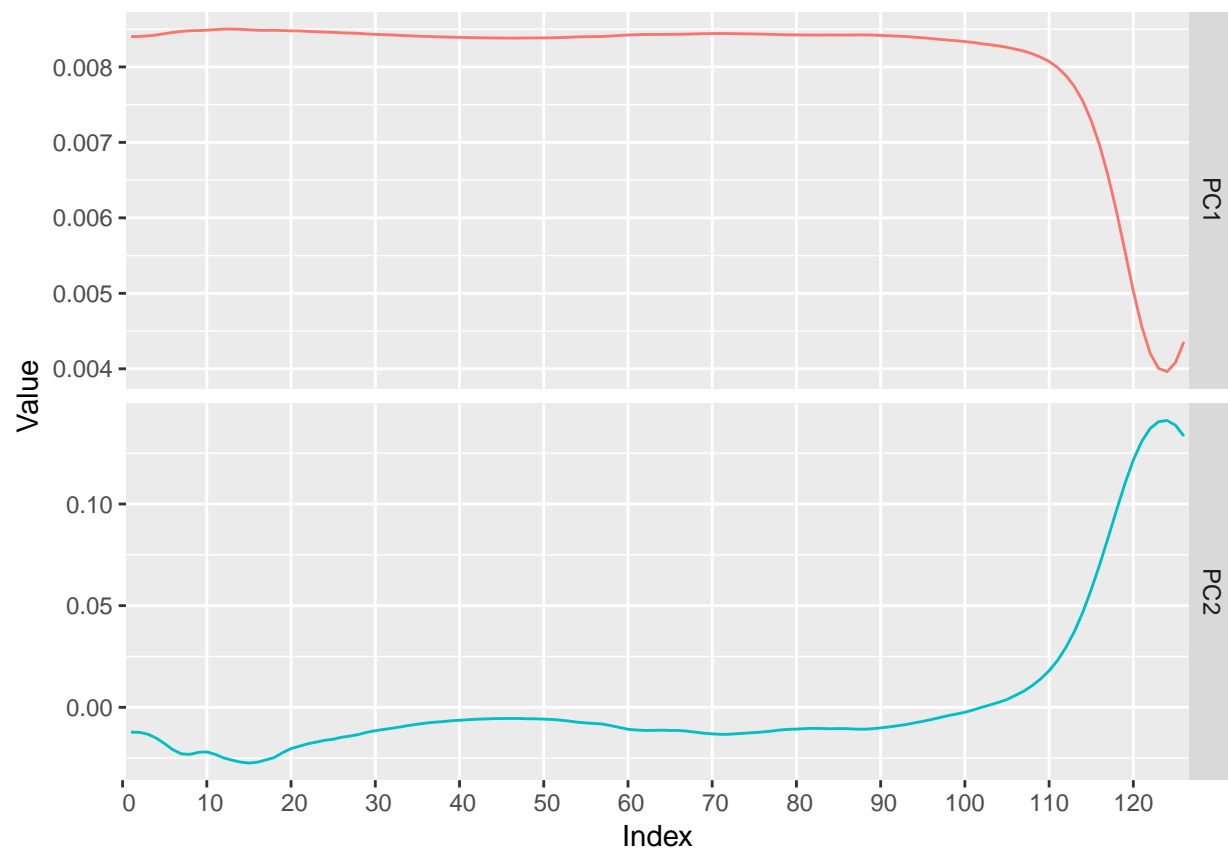


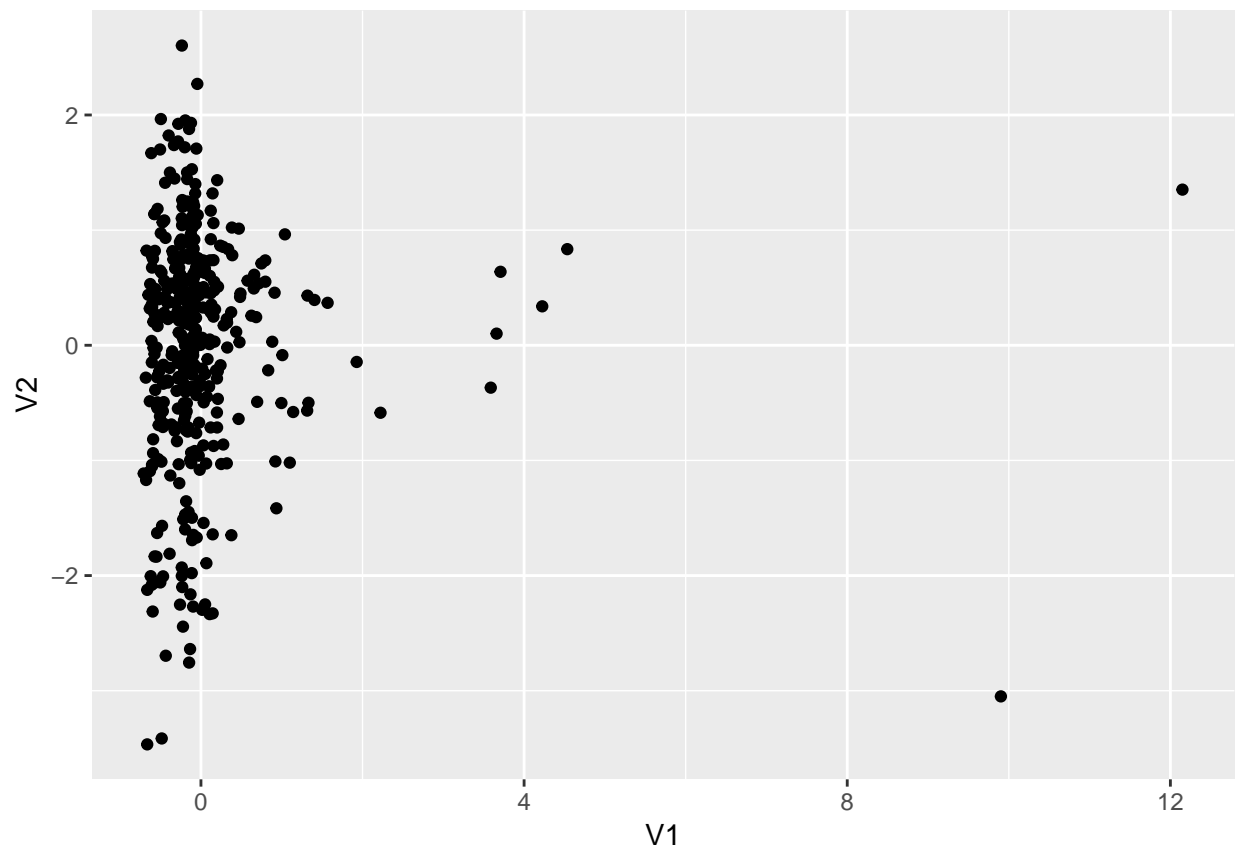
2



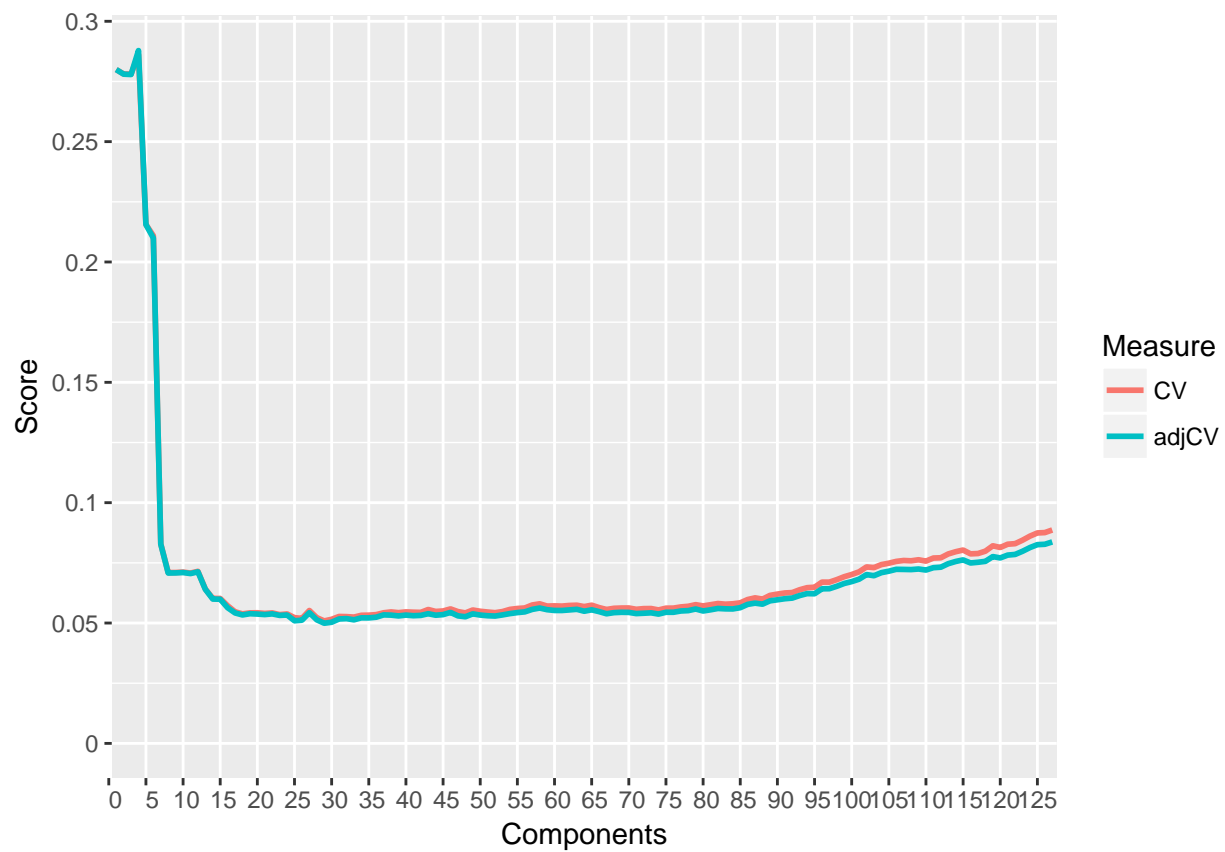
From the traceplot in figure ?? we can observe that the first and second PC are combinations of different variables. The first PC are a combination of all variables but less of the last ones while the second PC is mostly a combination of the last ones.

3





4



# Appendix

## Code for Assignment 1

```
library(ggplot2)
library(tree)
library(reshape)

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

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

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.vars="MET")
names(plot_data) <- c("MET", "Data", "EX")

ggplot(plot_data) +
  geom_point(aes(x=MET, y=EX, color=Data))
residuals <- resid(optimal_tree)

plot_data <- data.frame(x=1:length(residuals), y=residuals)

ggplot(plot_data) +
  xlab("Index") +
  ylab("Residual") +
  geom_point(aes(x=x, y=y))
```

## 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)]
```

```

traceplot <- function(n, pc1, pc2) {
  plot_data <- data.frame(x=1:n, PC1=pc1, PC2=pc2)
  plot_data <- melt(plot_data, id="x")
  names(plot_data) <- c("Index", "Component", "Value")
  xlimits <- seq(0, n, 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")
}

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])
sprintf("%2.3f", variances * 100)
sprintf("%2.3f", cumsum(variances))
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
traceplot(nrow(U), U[, 1], U[, 2])

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)
traceplot(nrow(W_prime), W_prime[, 1], W_prime[, 2])
ggplot(components) +
  geom_point(aes(x=V1, y=V2))

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")
names(plot_data)[ncol(plot_data)] <- "Score"
xlimits <- seq(0, ncol(data), by=5)

```

```
ylimits <- seq(0, max(plot_data$Score) + 0.05, by=0.05)

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