eNote 7

PLSR, Partial Least Squares Regression

Indhold

7.1 Reading material

The similar section from eNote4 on PCR, where the general reading for these methods are given, are repeated here:

PCR and the other biased regression methods presented in this course (PLS, Ridge and Lasso) are all together with even more methods (as e.g. MLR=OLS) introduced in each of the three books

- The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Second Edition, February 2009, Trevor Hastie, Robert Tibshirani, Jerome Friedman
- Ron Wehrens (2012). Chemometrics With R: Multivariate Data Analysis in the Natural Sciences and Life Sciences. Springer, Heidelberg.(Chapter 8 and 9)
- K. Varmuza and P. Filzmoser (2009). Introduction to Multivariate Statistical Analysis in Chemometrics, CRC Press. (Chapter 4)

The latter two ones are directly linked with R-packages, and here we will most directly use the latter. These things are NOT covered by the Lattin book, so we give here a reading list for the most relevant parts of chapter 4 of the Varmuza and Filzmoser book, when it comes to syllabus content for course 27411:

- Section 4.1 (4p) (Concepts - ALL models)
- Section 4.2.1-4.2.3 (6p)(Errors in (ALL) models)
- Section 4.2.5-4.2.6 (3.5p)(CV+bootstrap - ALL models)
- [Section 4.3.1-4.3.2.1 (9.5p)(Simple regression and MLR (=OLS))]
- [Section 4.5.3 (1p) (Stepwise Variable Selection in MLR)]
- Section 4.6 (2p) (PCR)
- Section 4.7.1 (3.5p) (PLS)
- Section 4.8.2 (2.5p) (Ridge and Lasso)
- Section 4.9-4.9.1.2 (5p) (An example using PCR and PLS)
- Section 4.9.1.4-4.9.1.5 (5p) (An example using Ridge and Lasso)
- Section 4.10 (2p) (Summary)

7.2 Presentation material

PLS1 method:

1. Centering and (possibly) scaling.
2. Do p simple regressions: x_1 on y, x_2 on y, ..., x_p on y.
3. Use these p regression coefficients $(w_1, w_2, \ldots, w_p)$ to define the first PLS-component, $t_1 = w_1x_1 + w_2x_2 + \cdots w_px_p$
4. Do the simple regression of y on t_1
5. Do the simple regressions of $x_1, x_2, \ldots, x_p$ on t_1
6. Repeat 2-5 on residuals for as many components as wanted/needed.
7. Choose number of components by Cross-Validation

PLS1

- PLS is a *canonical covariance* method

- PLS1 finds X-components with maximal Y-covariance:

$$\max_{||\alpha||=1} \text{Cov}^2(y, X\alpha)$$

- Or equivalently:

$$\max_{||\alpha||=1} \text{Corr}^2(y, X\alpha)\text{Var}(X\alpha)$$

- PCR is a *canonical variance* method
- PCR finds X-components with maximal Y-variance:

$$\max_{||\alpha||=1} \text{Var}(X\alpha)$$

PLS versus PCR

- PLS uses the y-information for building components, PCR does not
- PLS and PCR often predicts on a similar level of error
- PLS often does so with fewer components

How to do it?(same as for PCR)

1. Explore data
2. Do modelling (choose number of components, consider variable selection)
3. Validate (residuals, outliers, influence etc)
4. Iterate e.g. on 2. and 3.
5. Interpret, conclude, report.
6. If relevant: predict future values.

Cross Validation ("Full")

- Leave out one of the observations
- Fit a model on the remaining(reduced) data
- Predict the left out observation by the model: $\hat{y}_{i,val}$

- Do this in turn for ALL observations AND calculate the overall performance of the model:

$$\text{RMSEP} = \sqrt{\sum_i^n (y_i - \hat{y}_{i,val})^2/n}$$

(Root Mean Squared Error of Prediction)

Cross Validation ("Full")

Choose the optimal number of components:

- The one with overall minimal error
- The first local mininum
- In Hastie et al: the smallest number within the uncertainties of the overall minimum one.

Resampling

- Cross-Validation (CV)
- Jackknifing (Leave-on-out CV)
- Bootstrapping
- A good generic approach:
  - Split the data into a TRAINING and a TEST set.
  - Use Cross-validation on the TRAINING data
  - Check the model performance on the TEST-set
  - MAYBE: REPEAT all this many times (Repeated Double Cross Validation)

Cross Validation - principle

- Minimizes the expected prediction error:

$$\text{Squared Prediction error} = \text{Bias}^2 + \text{Variance}$$

- Including "many"PLS-components: LOW bias, but HIGH variance

- Including ”few”PLS-components: HIGH bias, but LOW variance
- Choose the best compromise!
- Note: Including ALL components = MLR (when $n > p$)

Validation - exist on different levels

1. Split in 3: Training(50%), Validation(25%) and Test(25%)
   - Requires many observations - Rarely used
2. Split in 2: Calibration/training (67%) and Test(33%) - us CV/bootstrap within the training
   - more commonly used
3. No ”fixed split”, but repeated splits by CV/bootstrap, and then CV within each training set (”Repeated double CV”)
4. No split, but using (one level of) CV/bootstrap.
5. Just fitting on all - and checking the error.

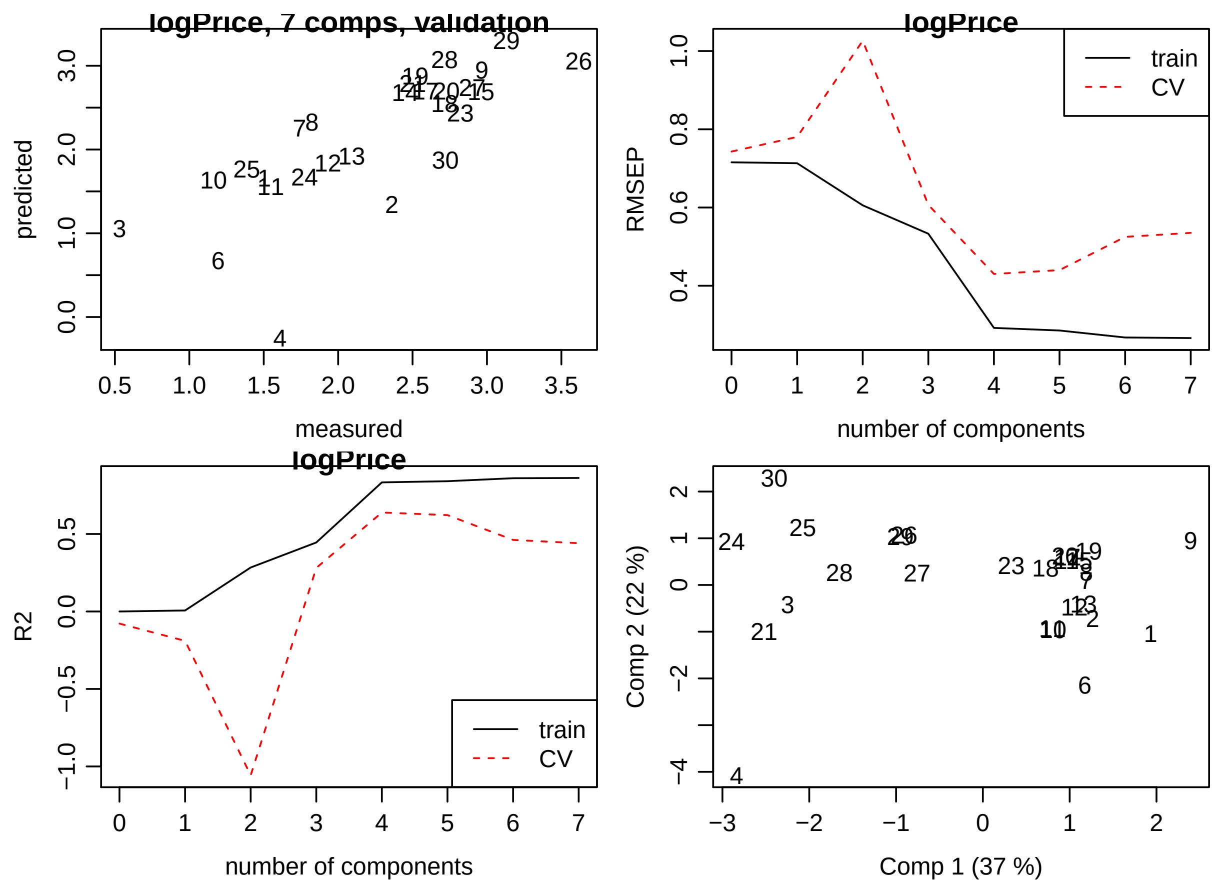
7.3 Motivating example - Leslie Salt again

```
# Example: using Salt data:
saltdata <- read.table("LESLIE_SALT.csv", header = TRUE,
                       sep = ";", dec = ".")
saltdata$logPrice <- log(saltdata$Price)
```

```
# Define the X-matrix as a matrix in the data frame:
saltdata$X <- as.matrix(saltdata[, 2:8])

# First of all we consider a random selection of 4 properties  as a TEST set
saltdata$train <- TRUE
saltdata$train[sample(1:length(saltdata$train), 4)] <- FALSE

saltdata_TEST <- saltdata[saltdata$train == FALSE,]
saltdata_TRAIN <- saltdata[saltdata$train == TRUE,]
```

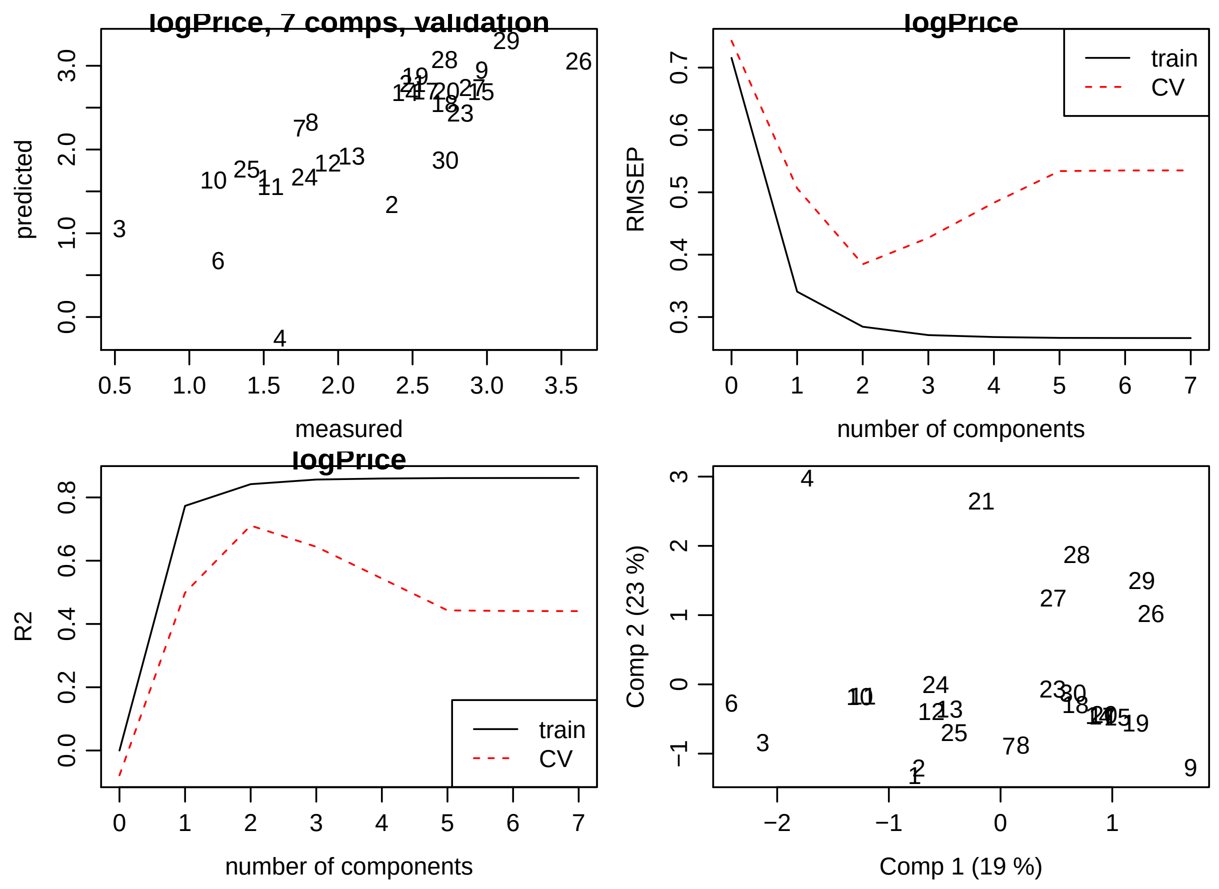
Now all the work is performed on the TRAIN data set

```
# First:  1) Explore the data - we allready did this previously,
# so no more of that here

# Next: 2) Model the data, first PCR
# Run the PCR with maximal/large number of components using pls package
# library(pls)
mod_pcr <- pcr(logPrice ~ X , ncomp = 7, data = saltdata_TRAIN,
               validation = "LOO", scale = TRUE, jackknife = TRUE)

# Initial set of plots:
par(mfrow=c(2,2))
plot(mod_pcr, labels = rownames(saltdata_TRAIN), which = "validation")
plot(mod_pcr, "validation", estimate = c("train", "CV"),
     legendpos = "topright")
plot(mod_pcr, "validation", estimate = c("train", "CV"),
     val.type = "R2", legendpos = "bottomright")
pls::scoreplot(mod_pcr, labels = rownames(saltdata_TRAIN))
```

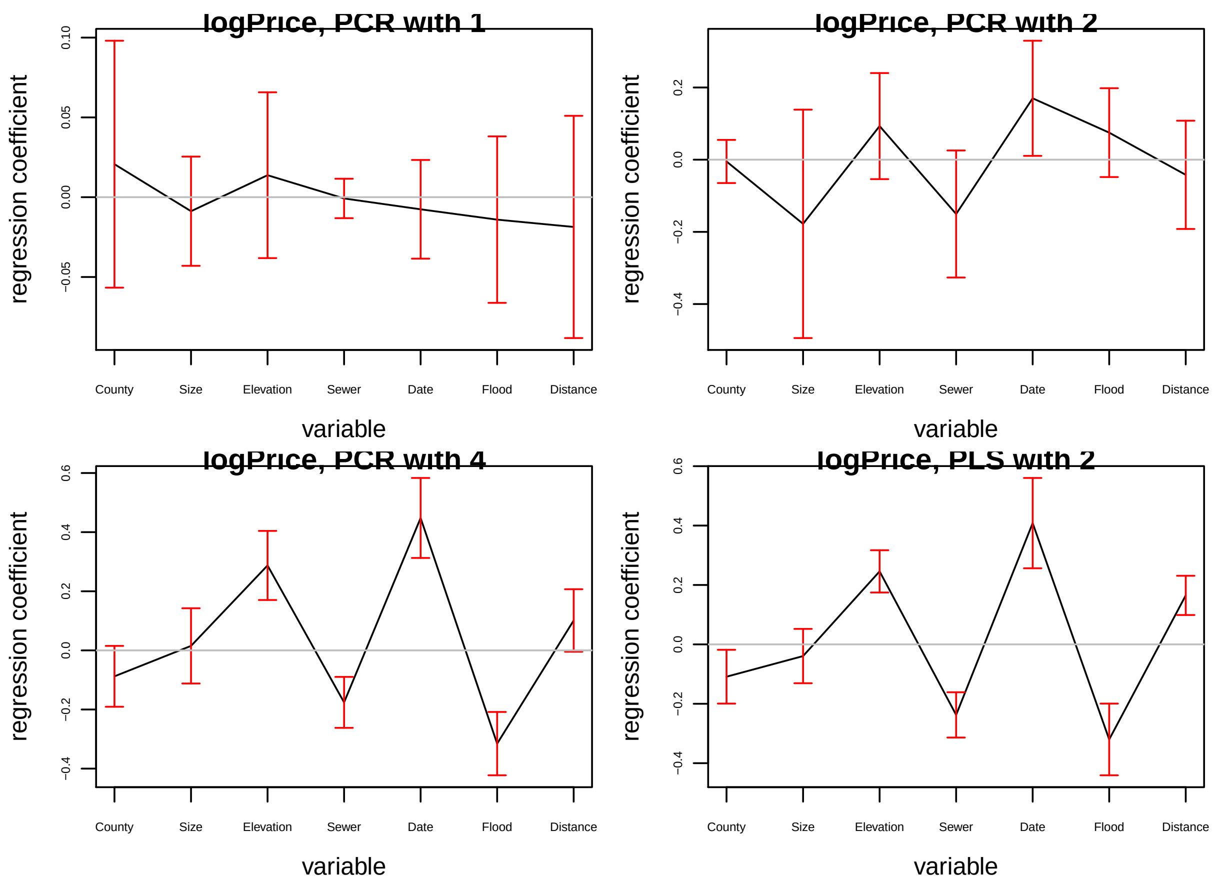


Note how the first PC-component does not bring anything for the prediction of the logPrice.

```r
# Now with pls:
mod_pls <- plsr(logPrice ~X , ncomp = 7, data = saltdata_TRAIN,
                validation = "LOO", scale = TRUE, jackknife = TRUE)

# Initial set of plots:
par(mfrow=c(2, 2))
plot(mod_pls, labels = rownames(saltdata_TRAIN), which = "validation")
plot(mod_pls, "validation", estimate = c("train", "CV"),
     legendpos = "topright")
plot(mod_pls, "validation", estimate = c("train", "CV"),
     val.type ="R2", legendpos = "bottomright")
pls::scoreplot(mod_pls, labels = rownames(saltdata_TRAIN))
```



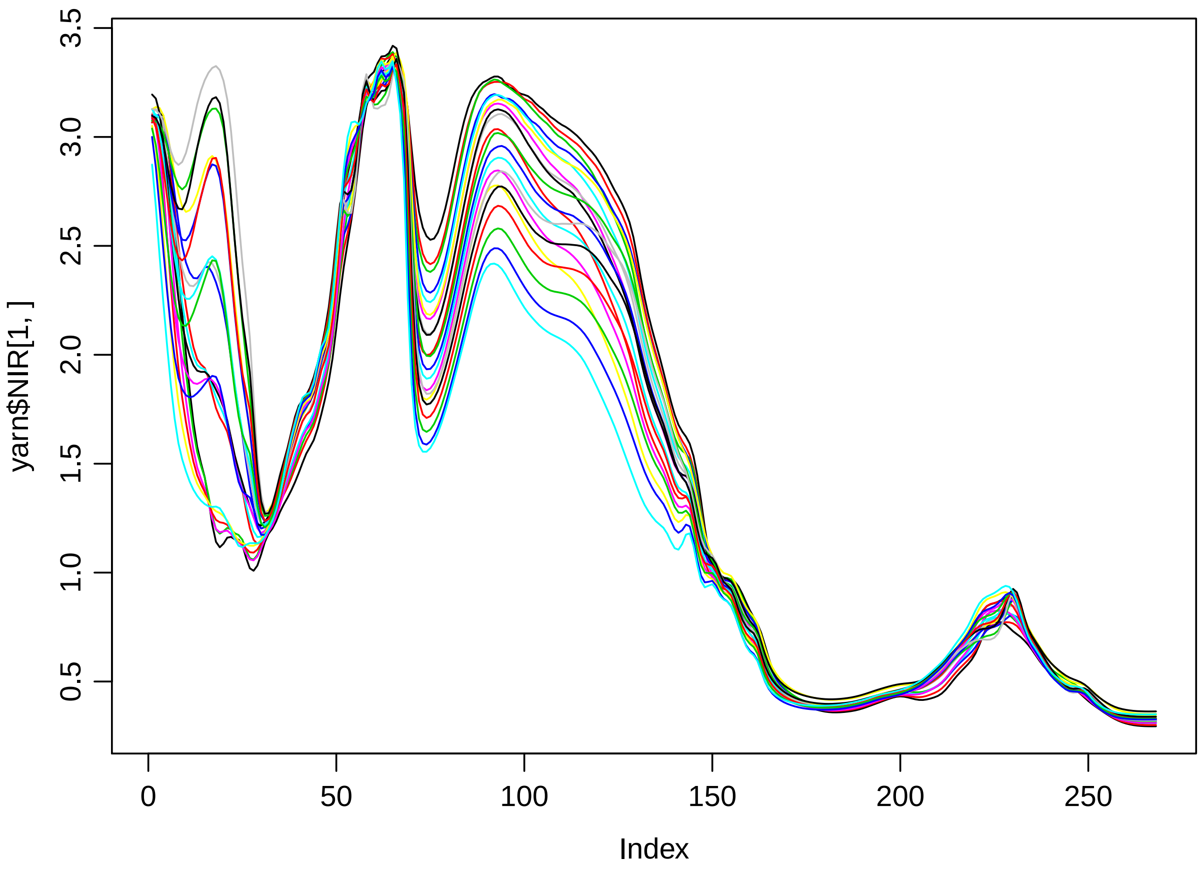
```
# Let us look at the coefficients of the final models:
# Plot coefficients with uncertainty from Jacknife:
par(mfrow=c(2, 2))
coefplot(mod_pcr, ncomp = 1, se.whiskers = TRUE, labels = prednames(mod_pcr),
         cex.axis =0.5, main = "logPrice, PCR with 1")
coefplot(mod_pcr, ncomp = 2, se.whiskers = TRUE, labels = prednames(mod_pcr),
         cex.axis = 0.5, main = "logPrice, PCR with 2")
coefplot(mod_pcr, ncomp = 4, se.whiskers = TRUE, labels = prednames(mod_pcr),
         cex.axis = 0.5, main = "logPrice, PCR with 4")
coefplot(mod_pls, ncomp = 2, se.whiskers = TRUE, labels = prednames(mod_pls),
         cex.axis = 0.5, main = "logPrice, PLS with 2")
```



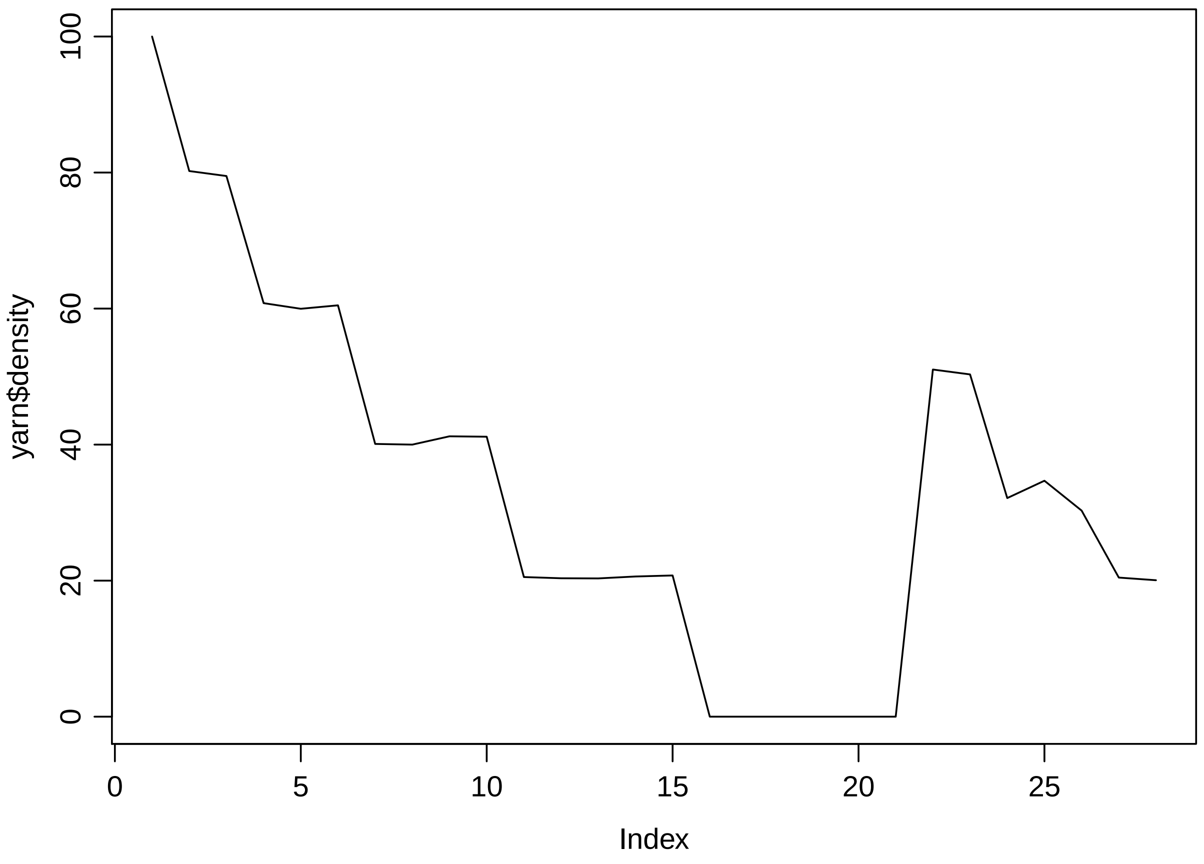
7.4 NIR Example: yarn data

```
#library(chemometrics)

data(yarn)
#summary(yarn)
?yarn
str(yarn)


'data.frame': 28 obs. of  3 variables:
 $ NIR    : num [1:28, 1:268] 3.07 3.07 3.08 3.08 3.1 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : NULL
 $ density: num  100 80.2 79.5 60.8 60 ...
 $ train  : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
```

```
# Plotting of the 21 individual NIR spectra"
max_X <- max(yarn$NIR)
min_X <- min(yarn$NIR)
par(mfrow = c(1,1))
plot(yarn$NIR[1,], type = "n", ylim = c(min_X, max_X))
for (i in 1:21) lines(yarn$NIR[i,], col = i)
```

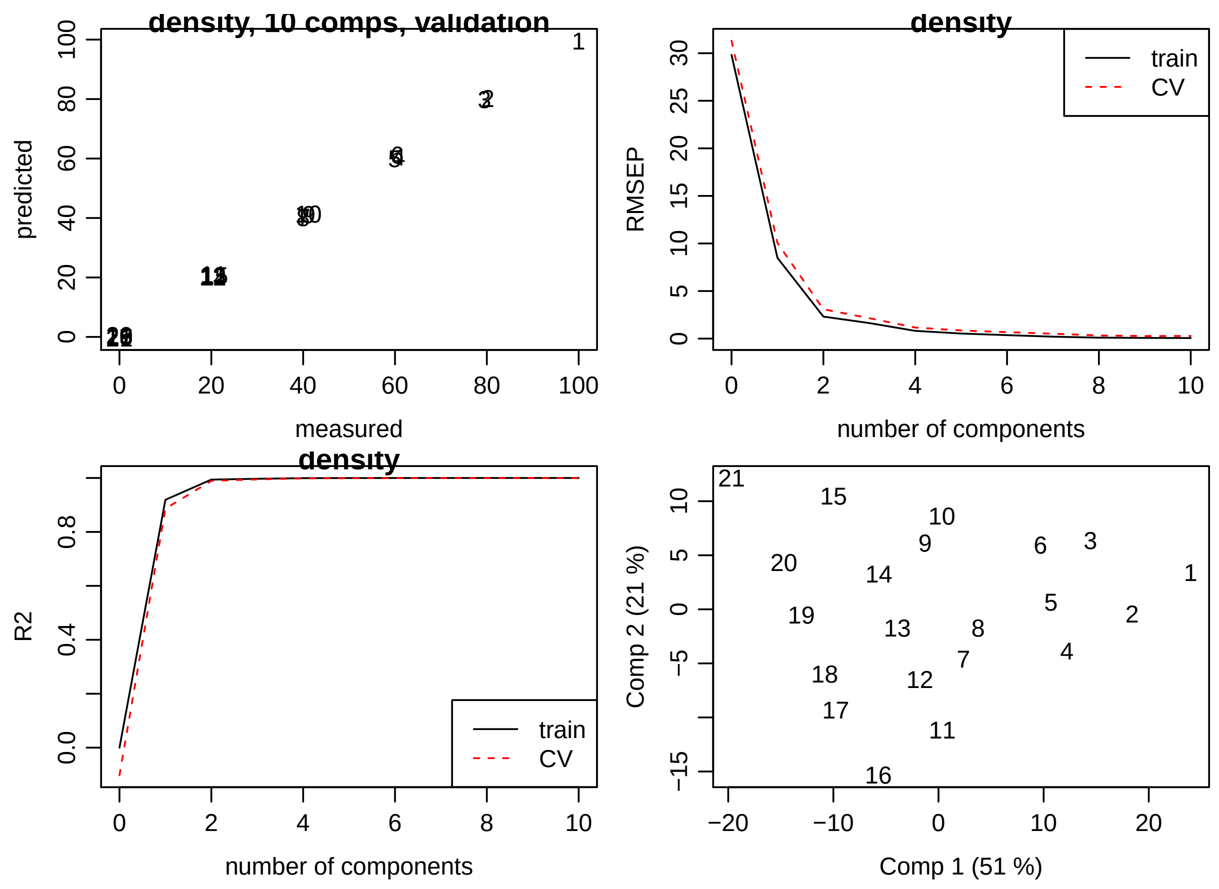


```
# Plot of y:
plot(yarn$density,type="n")
lines(yarn$density)
```

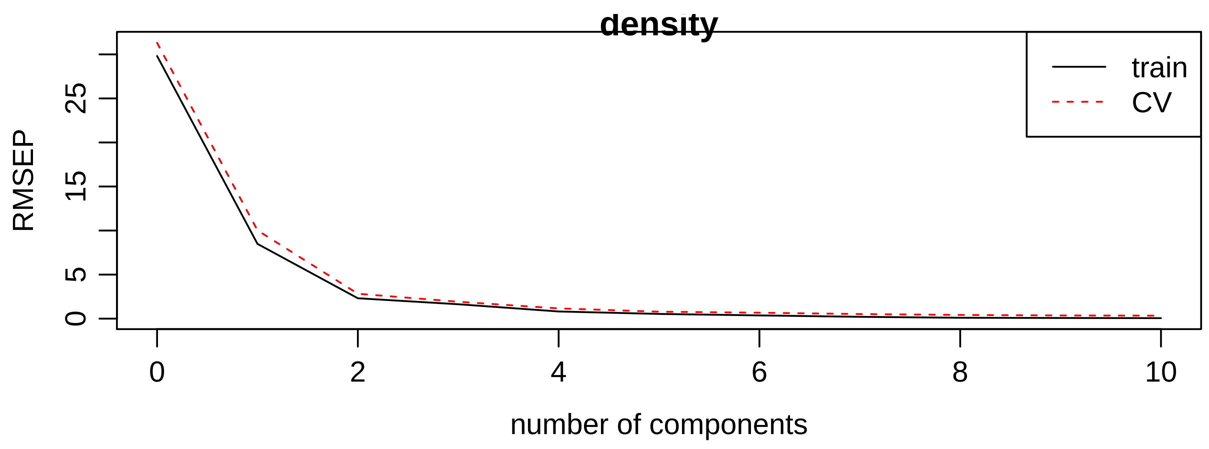


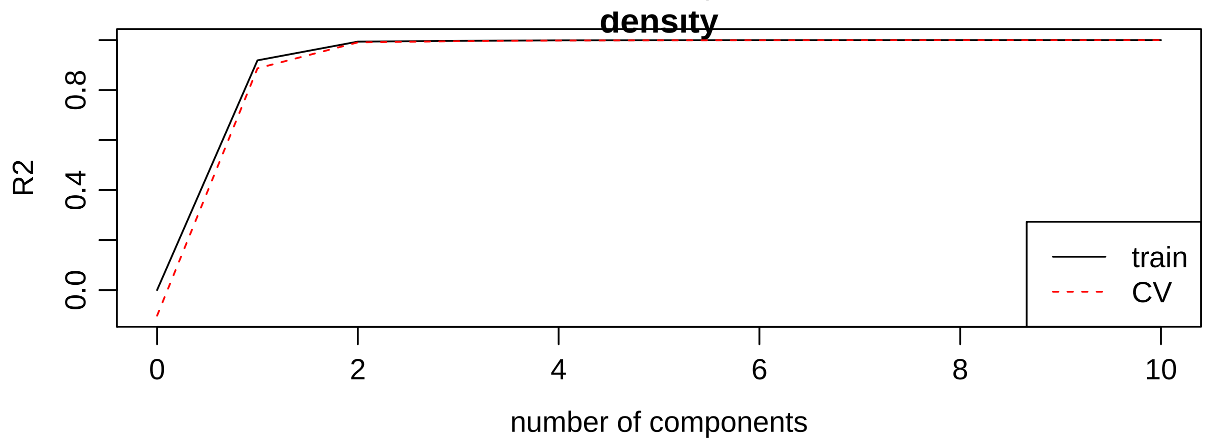
```
yarn_TEST <- yarn[yarn$train == FALSE,]
yarn_TRAIN <- yarn[yarn$train == TRUE,]

# Pls:
mod_pls <- plsr(density ~ NIR , ncomp = 10, data = yarn_TRAIN,
                validation = "LOO", scale = TRUE, jackknife = TRUE)
```

```
# Initial set of plots:
par(mfrow = c(2, 2))
plot(mod_pls, labels = rownames(yarn_TRAIN), which = "validation")
plot(mod_pls, "validation", estimate=c("train", "CV"),
     legendpos = "topright")
plot(mod_pls, "validation", estimate=c("train", "CV"),
     val.type = "R2", legendpos = "bottomright")
pls::scoreplot(mod_pls, labels = rownames(yarn_TRAIN))
```

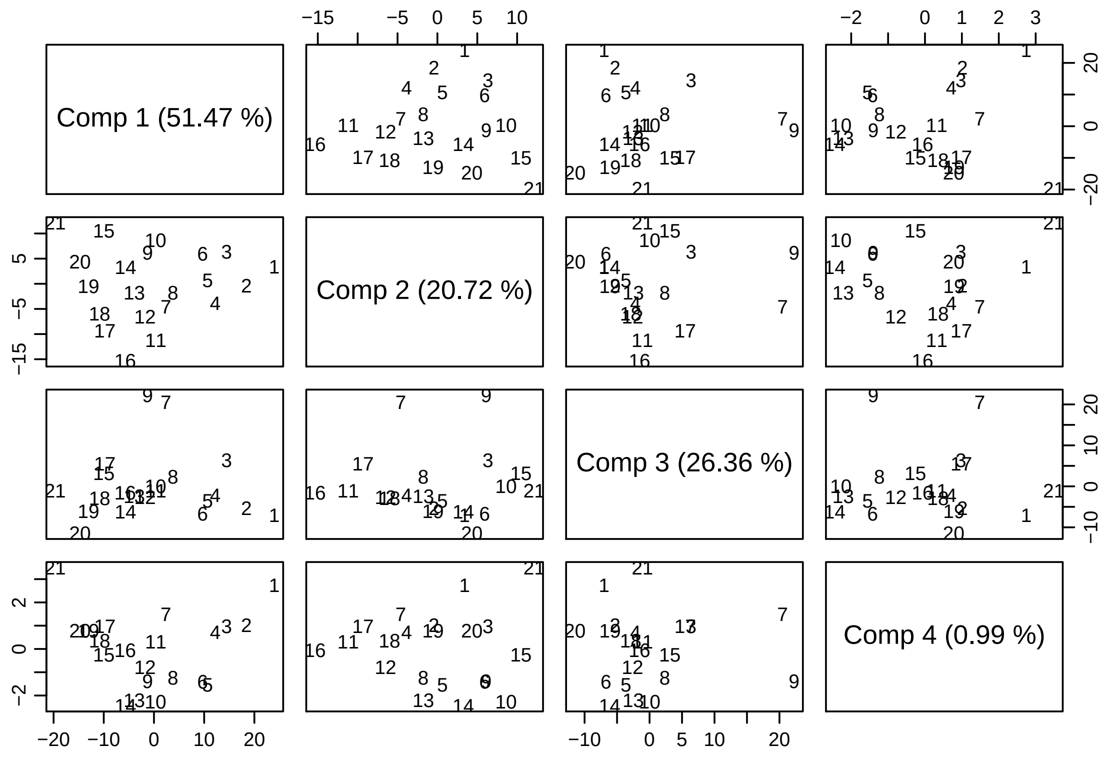


```
# Choice of components:

# what would segmented CV give:
mod_segCV <- plsr(density ~ NIR , ncomp = 10, data = yarn_TRAIN, scale = TRUE,
                  validation="CV", segments = 5, segment.type = c("random"),
                  jackknife = TRUE)

# Initial set of plots:
par(mfrow=c(2, 1))
plot(mod_segCV, "validation", estimate = c("train", "CV"),
     legendpos = "topright")
plot(mod_segCV, "validation", estimate = c("train", "CV"),
     val.type = "R2", legendpos = "bottomright")
```

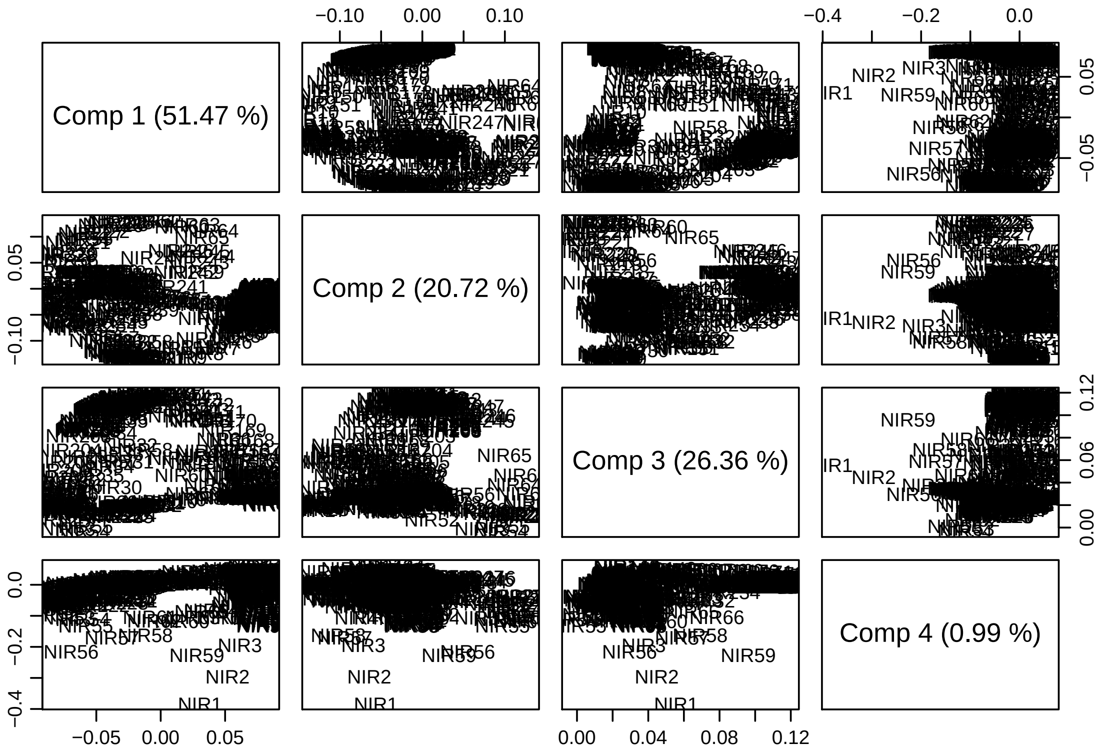




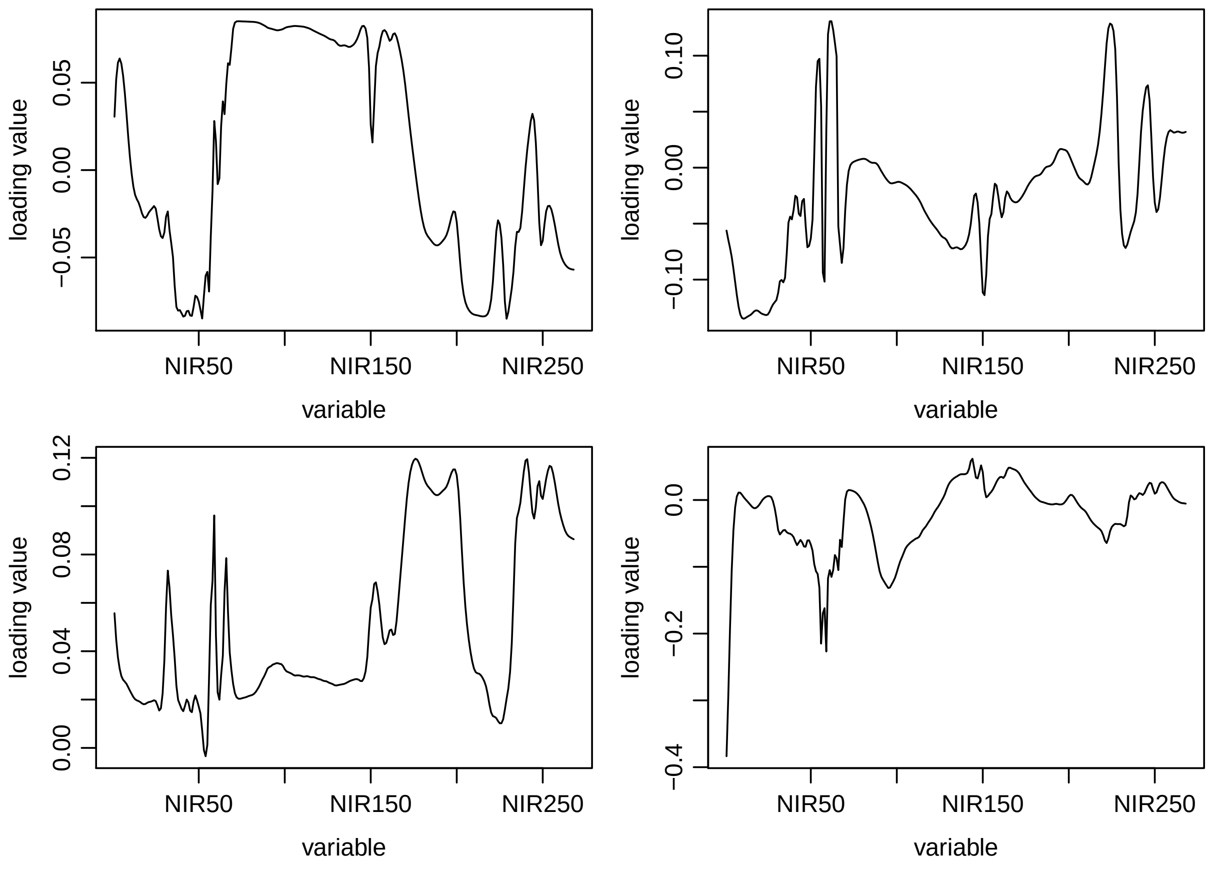
```
# Let us look at some more components:

# Scores:
pls::scoreplot(mod_pls, comps = 1:4, labels = rownames(yarn_TRAIN))
```



```
#Loadings:
loadingplot(mod_pls, comps = 1:4, scatter = TRUE, labels = prednames(mod_pls))
```



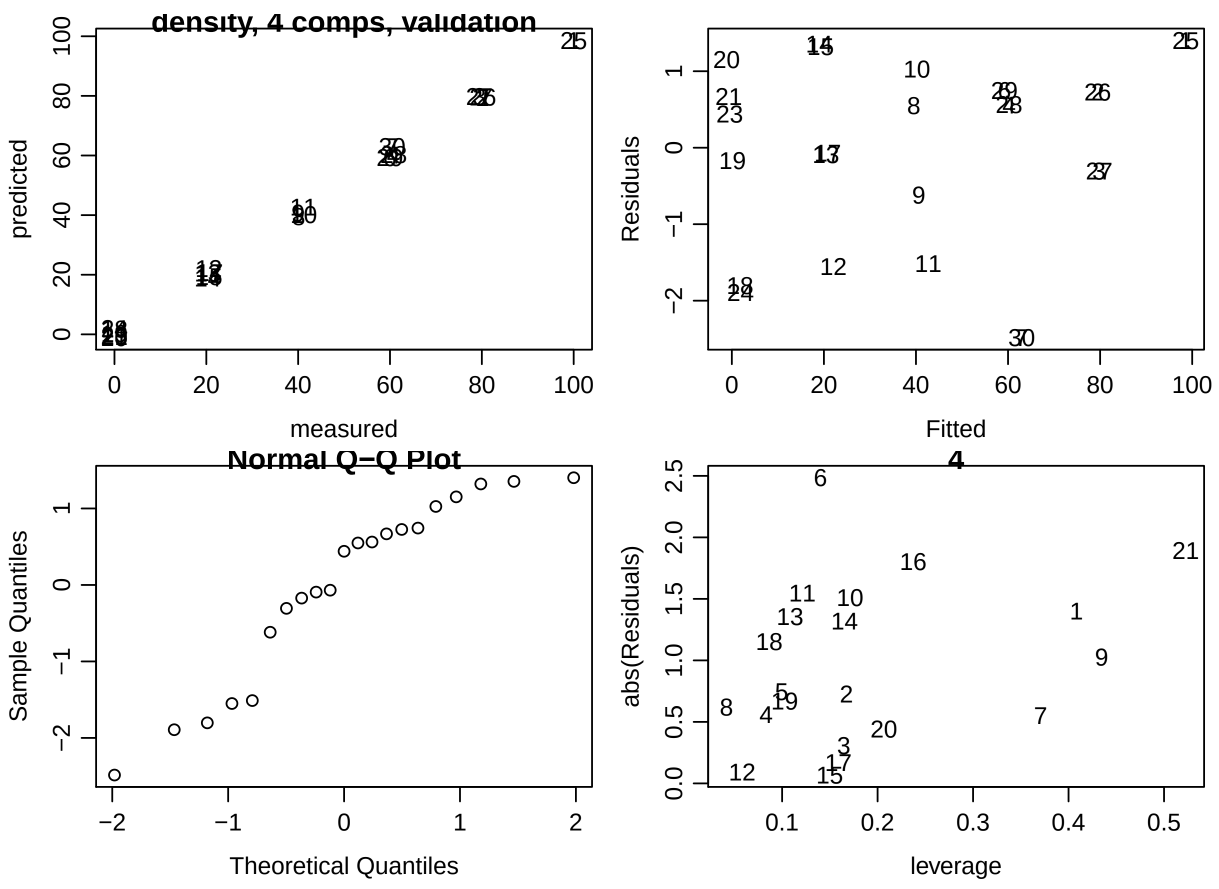
```
#Loadings, one by one - the spectral view:
par(mfrow=c(2, 2))
loadingplot(mod_pls, comps = 1, scatter = FALSE, labels = prednames(mod_pls))
loadingplot(mod_pls, comps = 2, scatter = FALSE, labels = prednames(mod_pls))
loadingplot(mod_pls, comps = 3, scatter = FALSE, labels = prednames(mod_pls))
loadingplot(mod_pls, comps = 4, scatter = FALSE, labels = prednames(mod_pls))
```



```
# We choose 4 components
mod4 <- plsr(density ~ NIR , ncomp = 4, data = yarn_TRAIN, validation = "LOO",
             scale = TRUE, jackknife = TRUE)
```

```
# Then 3) Validate:
# Let's validate som more: using 4 components
# We take the predicted and hence the residuals from the predplot function
# Hence these are the (CV) VALIDATED versions!

par(mfrow=c(2, 2))
obsfit <- predplot(mod4, labels = rownames(saltdata_TRAIN),
                   which = "validation")
Residuals <- obsfit[,1] - obsfit[,2]
plot(obsfit[,2], Residuals, type = "n", xlab = "Fitted", ylab = "Residuals")
text(obsfit[,2], Residuals, labels = rownames(saltdata_TRAIN))

qqnorm(Residuals)
```

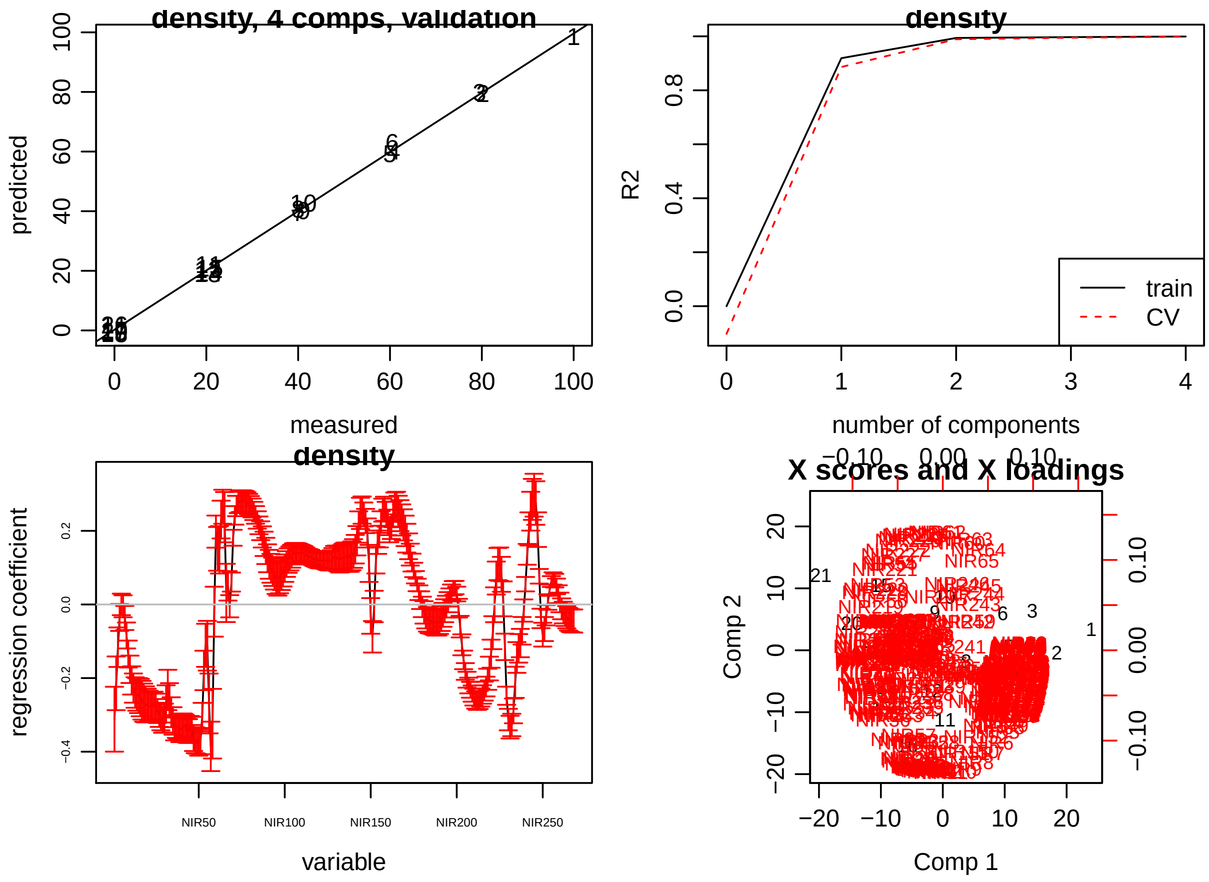
```
# To plot residuals against X-leverage, we need to find the X-leverage:
# AND then find the leverage-values as diagonals of the Hat-matrix:
# Based on fitted X-values:
Xf <- scores(mod4)
H <- Xf %*% solve(t(Xf) %*% Xf) %*% t(Xf)
leverage <- diag(H)

plot(leverage, abs(Residuals), type = "n", main = "4")
text(leverage, abs(Residuals), labels = rownames(yarn_TRAIN))
```



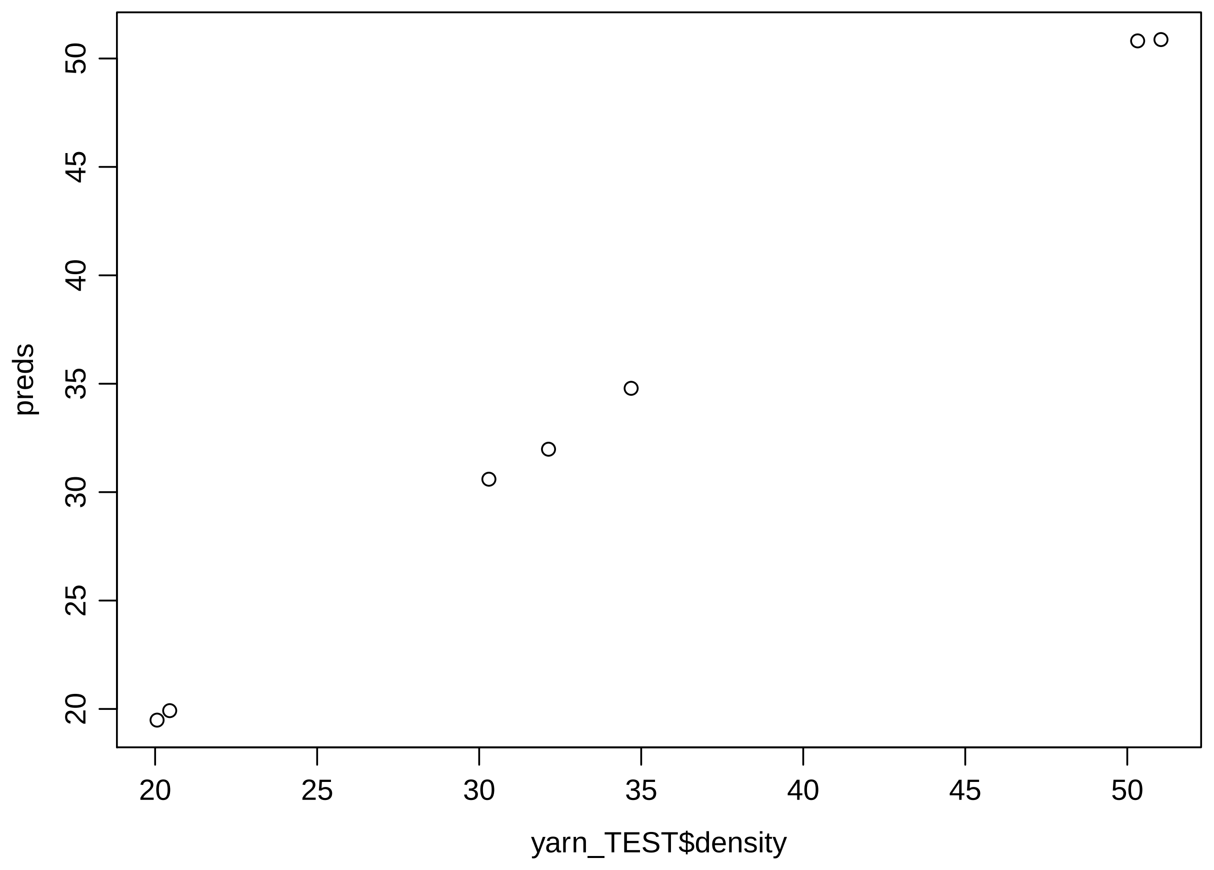
This set of four plots could be reproduced for other values of k. One could also redo some leaving out certain observations

```
# Now let's look at the results  -  4) "interpret/conclude"

par(mfrow=c(2,2))
# Plot coefficients with uncertainty from Jacknife:
```

```
obsfit <- predplot(mod4, labels = rownames(yarn_TRAIN),
                   which = "validation")
abline(lm(obsfit[,2] ~ obsfit[,1]))
plot(mod4, "validation", estimate = c("train", "CV"), val.type = "R2",
     legendpos = "bottomright")

coefplot(mod4, se.whiskers = TRUE, labels = prednames(mod4), cex.axis = 0.5)
biplot(mod4)
```



```
# And now let's try to 5) predict the 4 data points from the TEST set:
preds <- predict(mod4, newdata = yarn_TEST, ncomp = 4)
plot(yarn_TEST$density, preds)
```



```
rmsep <- sqrt(mean((yarn_TEST$density - preds)^2))
rmsep


[1] 0.3781003
```

7.5 Exercises

|||| Exercise 1 NIR data on Degree of Esterification

Use the Pectin data to be downloaded from file sharing in CN on

relating the degree of esterification (DE) for pectins to their NIR spectra as obtained by diffuse reflectance technique.

The datafile contains data for 70 different granulated pectin samples with 1 Y-variable (DE) and 1039 regressor X-variables in the NIR region (3996 cm^{-1} - 12004 cm^{-1}).

a) Setup and validate a PLS-model following the standard steps as shown in the PCR approach.

b) Compare with PCR results also!

c) Remember to validate and check for outliers.

d) Are any spectral ranges more important than others to predict DE?

e) How precise can DE be predicted by the models?

f) Define your own test set by a random selection of 23 samples:

```
# Exercise data: Pectin data:
pectin <- read.table("Pectin_and_DE.txt", header = TRUE,
                     sep = ";", dec = ",")
pectin$NIR <- as.matrix(pectin[,-1])

length(pectin$DE)

[1] 70

# First of all we consider a random selection of 23
# observation as a TEST set
pectin$train <- TRUE
pectin$train[sample(1:length(pectin$DE),23)] <- FALSE
```

```
pectin_TEST <- pectin[pectin$train==FALSE,]
pectin_TRAIN <- pectin[pectin$train==TRUE,]

# The training X-data is 47 rows and 1039 columns:
dim(pectin_TRAIN$NIR)

[1]   47 1039
```