MIRI
Data Mining & Business Intelligence

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Session 4.2
Partial Least Squares Regression 1

Course on Multivariate Modeling

Tomàs Aluja-Banet

tomas.aluja@upc.edu

imp information matters
INFORMATION MODELING & PROCESSING

Limitations of CCA and IBA

- Both proposed methods deliver up to rank(V_{XY}) solutions. If Y is of dimension 1, only one solution is proposed.
- Usually $p > q$, hence we don't use all explanatory potential of X to explain the response variables.
- We are assuming $n > (p,q)$. The number of solutions is limited to min $(p,q,n-1)$.
- Usually the explanatory variables are correlated, leading to unstable coefficients (if MVR, CCA) difficult to interpret (coefficients are interpretable when their sign coincide with their correlation).
- CCA, IBA, RDA (Redundancy Analysis), … are intended for complete data, they don't allow missing values

- Partial Least Squares Regression was proposed to deal with all these problems. They were initiated by Herman Wold, Svante Wold and Harald Martens (Wold, Martens and Wold, 1983).



- PLS is not based in a criterion to optimize (like the previous methods), rather it is based in an iterative algorithm (which converges). Its solution is equivalent to that of previous methods.

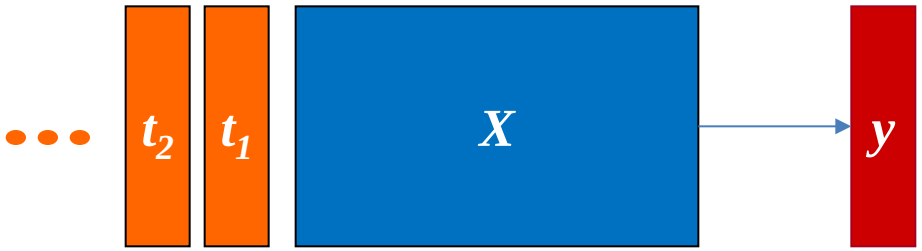
PLS Regression

- Relate a set of predictor variables X to a set of response variables Y
- Missing data is permitted
- The number of X variables can be very large, more than the number of observations $(p,q) >> n$
- Multicollinearity present for the set of predictors

PLS Regression

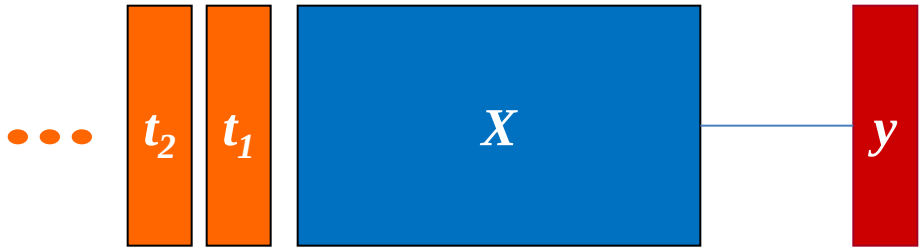
We will differentiate two cases

1. When Y has only one response = **PLS 1**
2. When Y has more than one response (general case) = **PLS 2**



1. We want m orthogonal components t_h, as correlated as possible with y and as explanatory as possible of the X matrix
   The number m of components is obtained by crossvalidation
2. Then, we will regress the y vector on the m components t_h
3. Finally, we will express the regression equation in terms of the original x_j variables

PLS1 Algorithm



We build a first PLSR component

$$t_1 = w_{11}x_1 + \cdots + w_{1p}x_p$$

$$w_{1j} = \frac{\mathrm{cov}(x_j, y)}{\sqrt{\sum_{j=1}^{p} \mathrm{cov}^2(x_j, y)}}$$

Normality constraint on w_1

the second component

Then we deflate y respect to t_1

$$y = c_1 t_1 + y_1$$

c_1 regression coef. of y respect to t_1

y_1 residuals of the OLS of y respect to t_1

$$y = c_1 w_{11} x_1 + \cdots + c_1 w_{1p} x_p + y_1$$

Also, we deflate x_j respecto to t_1

$$x_j = p_{1j} t_1 + x_{1j} \quad j = 1, \ldots, p$$

The second component is built as a linear composite of x_{1j} variables to explain the residual y_1

$$t_2 = w_{21} x_{11} + \cdots + w_{2p} x_{1p}$$

$$w_{2j} = \frac{\operatorname{cov}(x_{1j}, y_1)}{\sqrt{\sum_{j=1}^{p} \operatorname{cov}^2(x_{1j}, y_1)}}$$

PLS1 algorithm

X and y centered (eventually std.)

$$X_0 = X, y_0 = y$$
$$for\ h = 1, 2, \ldots, a$$
$$w_h = X'_{h-1} y_{h-1} / y'_{h-1} y_{h-1}$$
$$\|w_h\| = 1$$
$$t_h = X_{h-1} w_h / w'_h w_h$$
$$p_h = X'_{h-1} t_h / t'_h t_h$$
$$X_h = X_{h-1} - t_h p'_h$$
$$c_h = y'_{h-1} t_h / t'_h t_h$$
$$y_h = y_{h-1} - c_h t_h$$

a rank of X

if missing data:

← Regression of every col. of X_{h-1} on y_{h-1}

← Regression of every row of X_{h-1} on w_h

← Deflation of X

← Deflation of y

$$t'_h y_h = 0 \quad t'_h X_h = 0 \Rightarrow t'_h t_{h+1} = 0$$

PLSR algorithm

If there are no missing values, then, components t_h will be orthogonal

Hence, it is equivalent to perform the deflation component by component or all components altogether:

$$X_h = X_{h-1} - t_h p_h' = X - t_1 p_1' - \cdots - t_h p_h' = X - \hat{X}_h$$

$$\hat{X}_h = T_h P_h'$$

$$T_h = \begin{bmatrix} t_1 & \cdots & t_h \end{bmatrix}$$

$$P_h = \begin{bmatrix} p_1 & \cdots & p_h \end{bmatrix}$$

$$y_h = y_{h-1} - c_h t_h = y - c_1 t_1 - \cdots - c_h t_h = y - \hat{y}_h$$

$$\hat{y}_h = T_h \mathbf{c}_h$$

$$\mathbf{c}_h = \begin{bmatrix} c_1 \\ \vdots \\ c_h \end{bmatrix}$$

Modeling y as function of the original variables

$$\hat{y} = c_1 t_1 + \cdots + c_h t_h = T_h \mathbf{c}_h$$

$$T_h = [t_1, t_2, \ldots, t_h]$$

$$\mathbf{c}_h = [c_1, c_2, \ldots, c_h]$$

$$X_1 = X - t_1 p_1' = X(I - w_1 p_1')$$

$$X_2 = X_1 - t_2 p_2' = X_1(I - w_2 p_2')$$

$$\vdots$$

$$X_h = X \underbrace{\prod_{j=1}^{h} (I - w_j p_j')}$$

orthogonal projector on $[t_1,\ldots,t_h]^{\perp}$

$$t_1 = X w_1 = X w_1^* \qquad w_1 = w_1^*$$

$$t_2 = X_1 w_2 = X(I - w_1 p_1') w_2 = X w_2^*$$

$$t_3 = X_2 w_3 = X_1(I - w_2 p_2') w_3 = X(I - w_1 p_1')(I - w_2 p_2') w_3 = X w_3^*$$

$$\vdots$$

$$t_h = X_{h-1} w_h = X \prod_{j=1}^{h-1} (I - w_j p_j') w_h = X w_h^*$$

$$T_h = [t_1, t_2, \ldots, t_h] = X W_h^*$$

projection vectors $[w^*_1,\ldots,w^*_h]$

W_h^* projection matrix

$$\hat{y} = T_h \mathbf{c}_h = X W_h^* \mathbf{c}_h = X b_h$$

Number of components

The number of components are taken by crossvalidation (usually LOO)

$$\hat{y}_h = T_h \mathbf{c}_h \qquad T_h = [t_1, t_2, \ldots, t_h] \qquad \mathbf{c}_h = [c_1, c_2, \ldots, c_h]$$

$$RSS_h = \sum_{i=1}^{n} (y_i - \hat{y}_{hi})^2$$

$$\hat{y}_{h(-i)} = T_{h(-i)} \mathbf{c}_{h(-i)} = X_{(-i)} b_{h(-i)}$$

$$PRESS_h = \sum_{i=1}^{n} (y_i - \hat{y}_{h(-i)})^2$$

$$RMSEP_{cv} = \frac{PRESS_h}{n}$$

$$R^2_{cv} = 1 - \frac{PRESS_h}{\sum_i (y_i - \bar{y})^2}$$

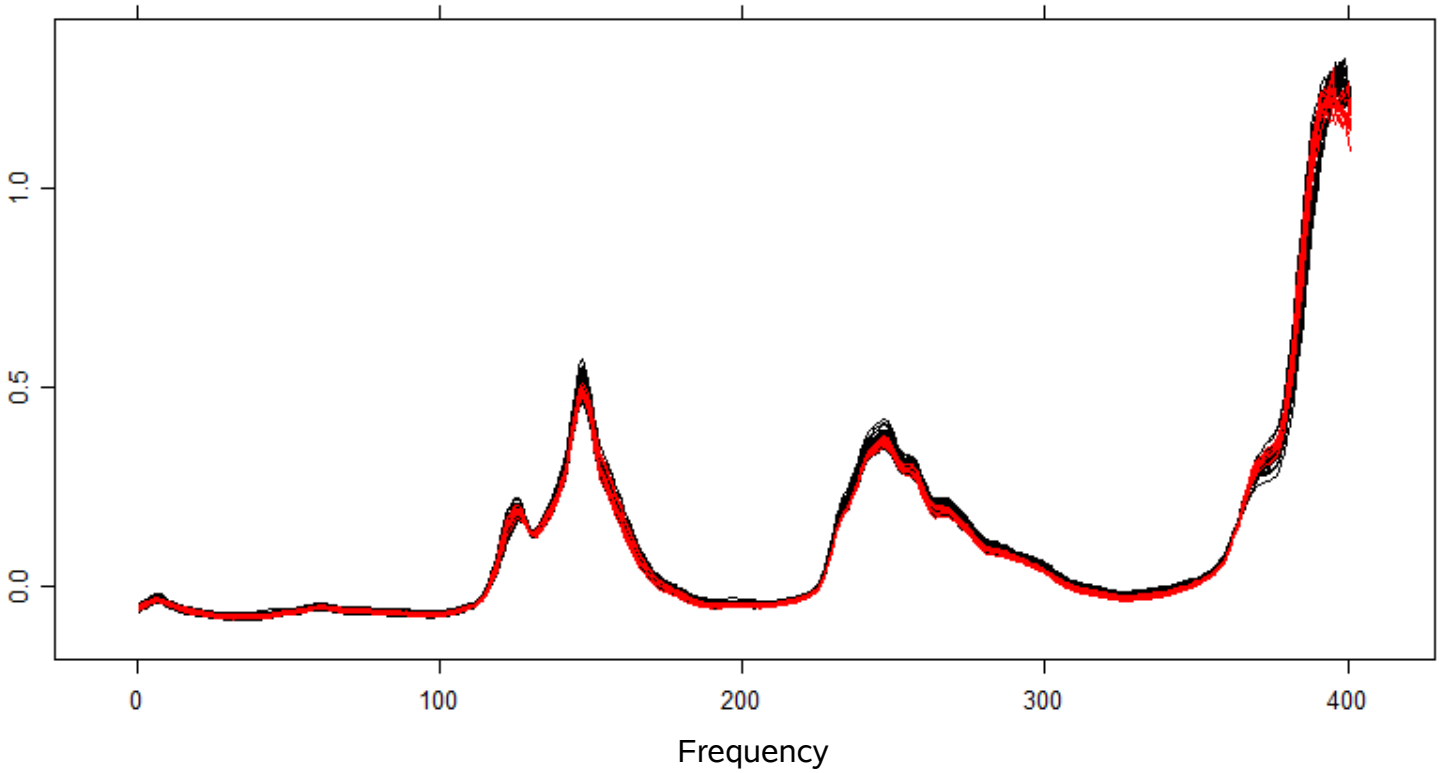
A component is taken if the $RMSEP$ decreases (or the $R_{cv}{}^2$ increases)

y is explained using the coefficients with the original variables x_j, but the predictions and confidence intervals are obtained from the regression of y on the t_h components with the usual formulae

Octane problem

- A typical problem of PLS1. We want to predict the Octane number of a gasoline from the NIR (Near Infra Red spectrum) of gasolines.
- The higher the octane number, the less likely is the fuel to ignite prematurely in the engine's cycle and cause the engine damage.
- Measuring the Octane number: The most common type of octane rating worldwide is the Research Octane Number (RON). RON is determined by running the fuel in a test engine with a variable compression ratio under controlled conditions, and comparing the results with those for mixtures of iso-octane and n-heptane.
- Infrared (IR) light is electromagnetic radiation with a wavelength longer than that of visible light, measured from the nominal edge of visible red light at 0.74 micrometers, and extending conventionally to 300 micrometres. Microscopically, IR light is typically emitted or absorbed by molecules when they change their rotational-vibrational movements. The infrared portion of the electromagnetic spectrum is usually divided into three regions; the near-, mid- and far- infrared, named for their relation to the visible spectrum.
- Spectroscopy: Is a technique which can be used to identify molecules by analysis of their constituent bonds. Each chemical bond in a molecule vibrates at a frequency which is characteristic of that bond. A group of atoms in a molecule may have multiple modes of oscillation caused by the stretching and bending motions of the group as a whole. The vibrational frequencies of most molecules correspond to the frequencies of infrared light. Typically, the technique is used to study organic compounds. This can be used to gain information about the sample composition in terms of chemical groups present and also its purity.

NIR spectrum

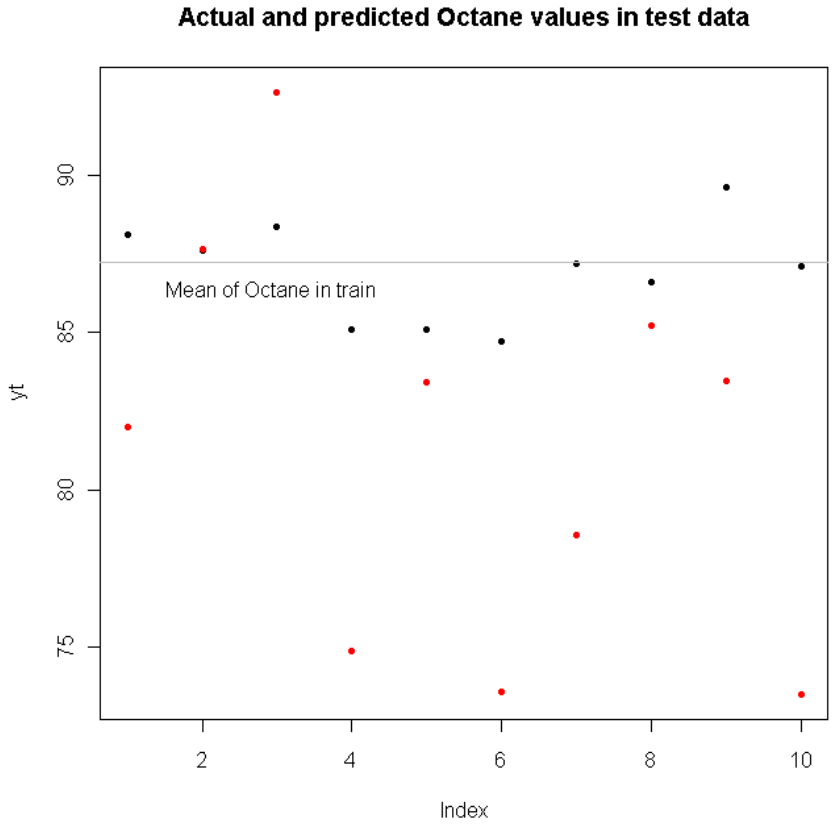


We have 60 gasolines, from which we have measured their octane number and their NIR spectrum (900nm – 1700nm), having 401 frequencies per each gasoline

We will use the first 50 gasolines as training sample, and the last 10 as test (holdout sample, in red)

Predicting the Octane by MR

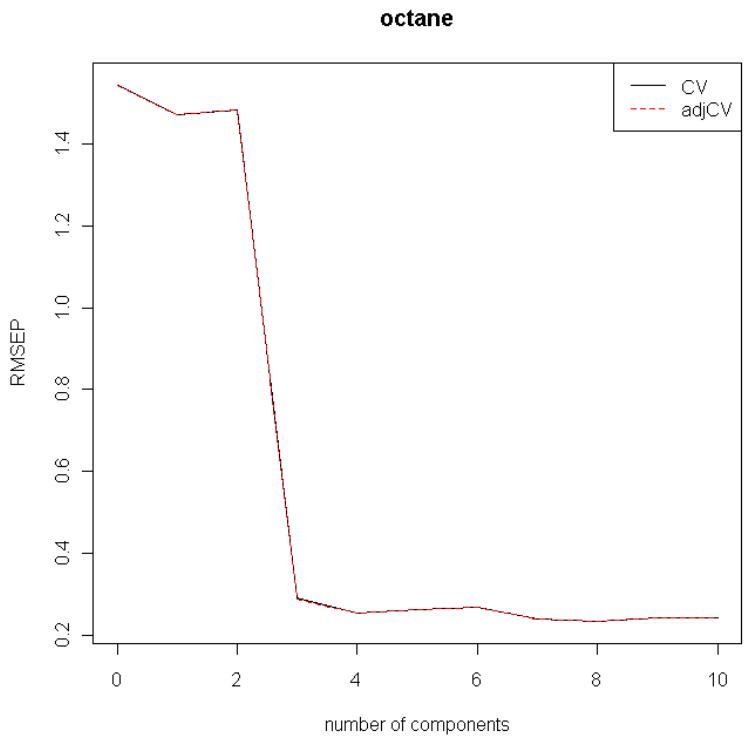
```
r=lm(octane~.,data=gasTrain)

Residual standard error: NaN on 0 degrees of freedom
Multiple R-squared:     1,      Adjusted R-squared:   NaN
F-statistic:   NaN on 49 and 0 DF,  p-value: NA
```



R2 test = -24.66982

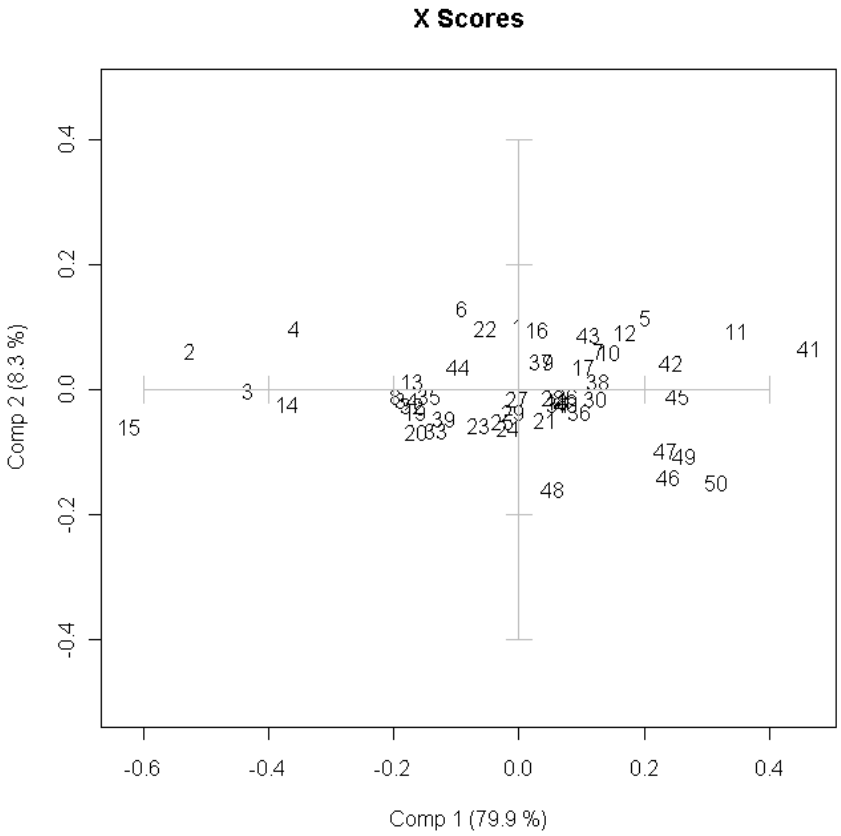
PCR of Gasoline Data

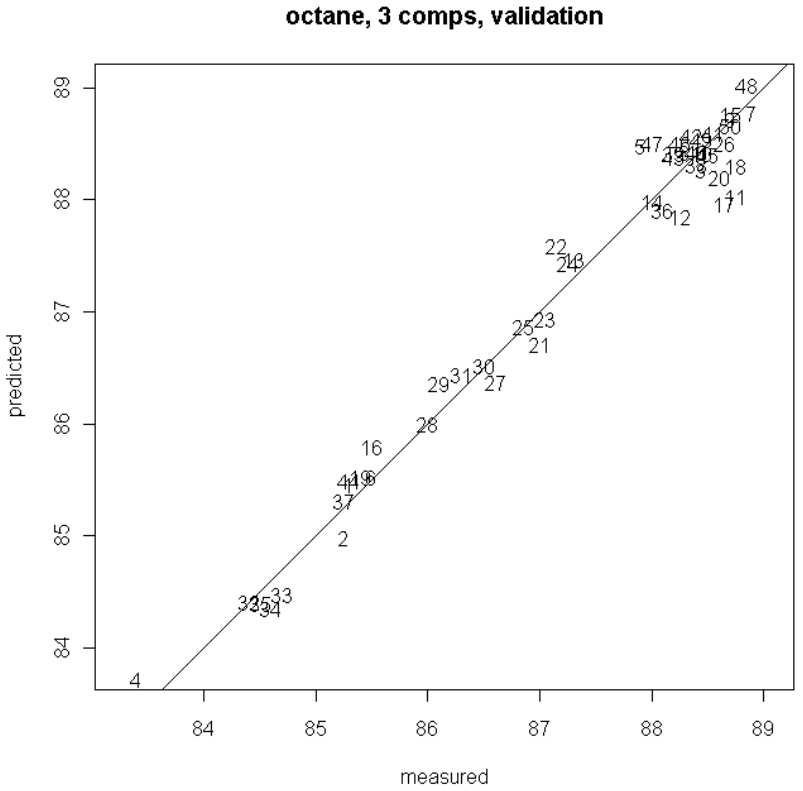
```
p1 <- pcr(octane ~ ., ncomp = 10, data = gasoline[1:50,], validation = "LOO")
```



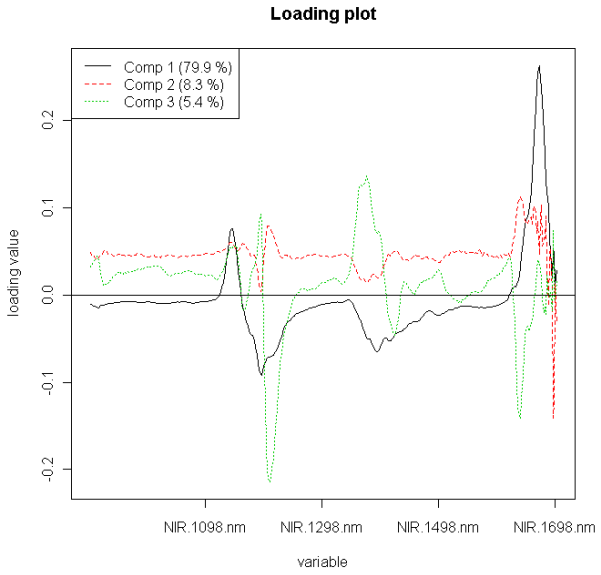

```
> explvar(p1)
    Comp 1     Comp 2     Comp 3     Comp 4     Comp 5
Comp 6     Comp 7
79.8586603  8.2639500  5.4171903  3.0034945  1.1963215
    Comp 6     Comp 7     Comp 8     Comp 9    Comp 10
 0.6397503  0.3691514  0.3127762  0.2171267  0.1417888
```

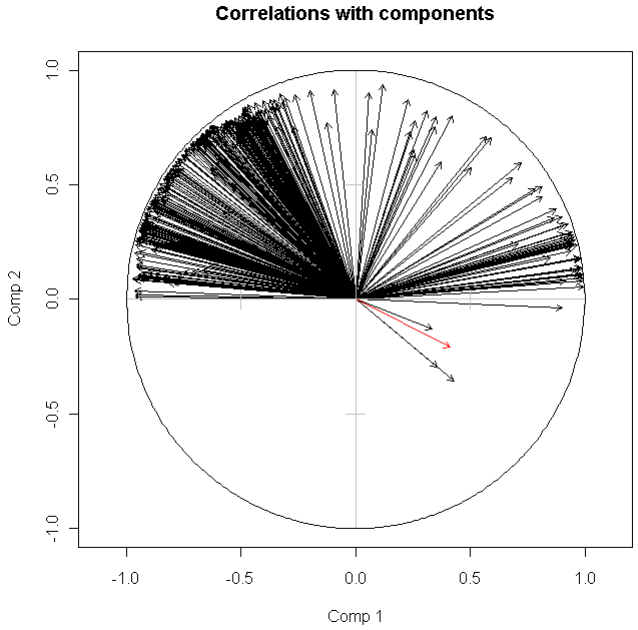
The gasoline training data

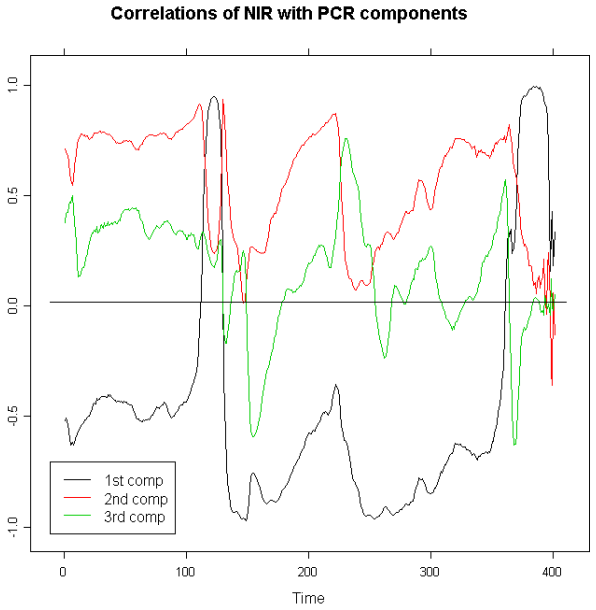




Loadings and correlations

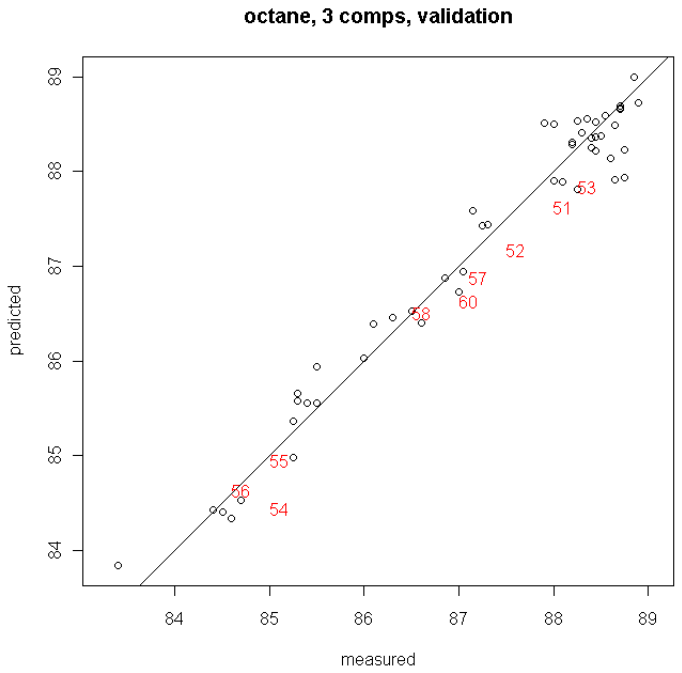






The PCR model

```
lm(formula = y ~ p1$scores[, 1:3])

Residuals:
     Min       1Q   Median       3Q      Max
-0.59006 -0.17399 -0.02148  0.13610  0.71821

Coefficients:
                       Estimate Std. Error  t value Pr(>|t|)
(Intercept)            87.22400    0.03869 2254.691  < 2e-16 ***
p1$scores[, 1:3]Comp 1  2.89707    0.17958   16.132  < 2e-16 ***
p1$scores[, 1:3]Comp 2 -4.57060    0.55825   -8.187 1.57e-10 ***
p1$scores[, 1:3]Comp 3 23.47029    0.68950   34.040  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2735 on 46 degrees of freedom
Multiple R-squared:  0.97,      Adjusted R-squared: 0.968
F-statistic: 495.3 on 3 and 46 DF,  p-value: < 2.2e-16
```



R2 test = 0.906

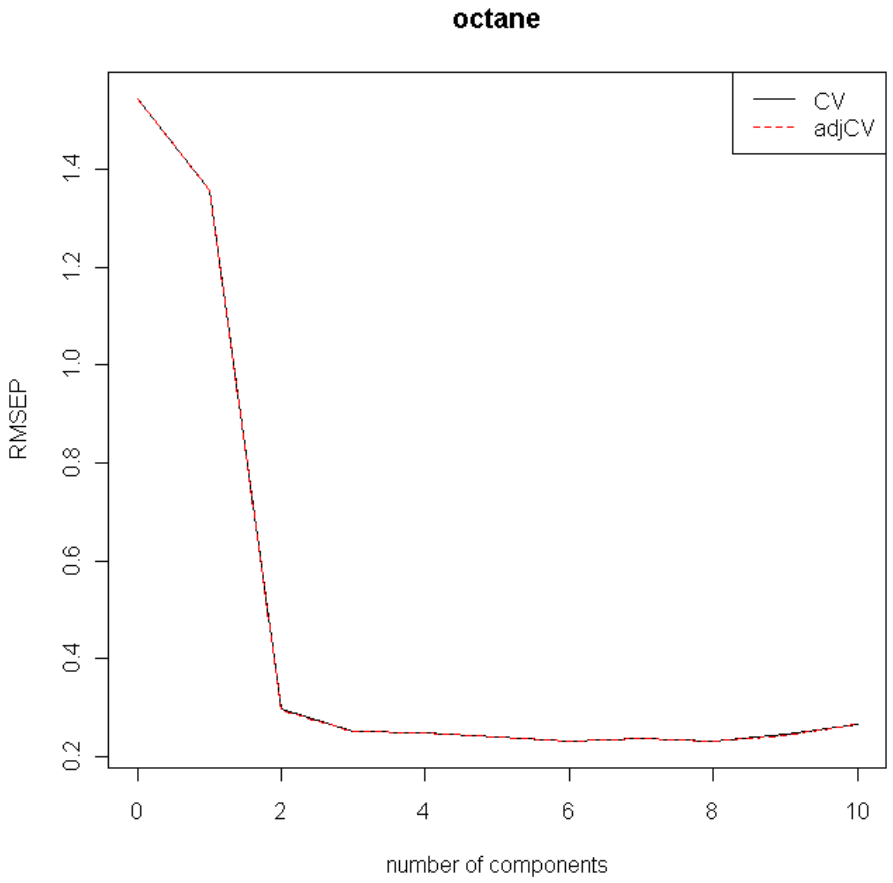
Predicting the test data

```
pred_test= predict(p1, ncomp = 3, newdata = gasTest)
```

The PLS1 solution

```
p1 <- plsr(octane ~ ., ncomp = 10, data = gasoline[1:50,], validation = "LOO")

Data:   X dimension: 50 401
        Y dimension: 50 1
Fit method: kernelpls
Number of components considered: 10

VALIDATION: RMSEP
Cross-validated using 50 leave-one-out segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
CV           1.545    1.357   0.2966   0.2524   0.2476   0.2398   0.2319
adjCV        1.545    1.356   0.2947   0.2521   0.2478   0.2388   0.2313
       7 comps  8 comps  9 comps  10 comps
CV      0.2386   0.2316   0.2449    0.2673
adjCV   0.2377   0.2308   0.2438    0.2657

TRAINING: % variance explained
        1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
X         78.17    85.58     93.4    96.06    96.94    97.89    98.38    98.85
octane    29.39    96.85     97.9    98.26    98.86    98.96    99.09    99.16
        9 comps  10 comps
X         99.02      99.2
octane    99.28      99.4
```

Selecting the number of components



Scores and Loadings

Scores

	Comp 1	Comp 2	Comp 3
1	-0.043556	-0.094168	0.059305
2	-0.472974	-0.016080	0.047717
3	-0.328194	0.125279	0.033588
4	-0.365999	-0.108083	0.052645
5	0.183252	0.010075	0.115155
6	-0.114041	-0.077129	0.093467
7	0.133122	0.044850	0.085409
8	-0.133321	0.092581	0.021212
9	0.068114	0.059107	0.072572
10	0.139518	0.027115	0.083531
11	0.294524	-0.029108	0.107154
12	0.145236	-0.010240	0.093560
13	-0.133728	0.037990	0.019048
14	-0.280500	0.107539	0.019261
15	-0.469787	0.188087	-0.003875
16	-0.013416	-0.085857	0.049918
17	0.099299	0.016226	0.058205
18	0.074893	0.046022	0.013879
19	-0.165865	-0.042503	-0.043131
20	-0.108124	0.084011	-0.032410

Loadings:

	Comp 1	Comp 2	Comp 3
NIR.900.nm	-0.01228	0.013708	0.05488
NIR.902.nm	-0.01155	0.016208	0.05340
NIR.904.nm	-0.01240	0.017443	0.05333
NIR.906.nm	-0.01347	0.021443	0.05482
NIR.908.nm	-0.01421	0.024035	0.05145
NIR.910.nm	-0.01543	0.025779	0.05081
NIR.912.nm	-0.01512	0.028922	0.05054
NIR.914.nm	-0.01573	0.024221	0.05083
NIR.916.nm	-0.01484	0.017316	0.05177
NIR.918.nm	-0.01386	0.008606	0.04991

$$X = TP'$$

```
lm(as.matrix(scale(X,scale=F))~
p1$scores)$coefficients  =    t(p1$loadings)
```

Yloadings

$$y = Tc$$

Yloadings: vector of c_h coeffcients, they allow to obtain the response from the PLS1 components

```
YLoadings:
Loadings:
       Comp 1 Comp 2 Comp 3 Comp 4 Comp 5 Comp 6 Comp 7 Comp 8 Comp 9 Comp 10
octane  4.345 19.097  2.369  3.262  5.490  2.678  4.025  2.822  7.830  7.079

                Comp 1  Comp 2  Comp 3  Comp 4  Comp 5  Comp 6  Comp 7  Comp 8
SS loadings     18.879 364.680   5.613  10.643  30.135   7.172  16.204   7.963
Proportion Var 18.879 364.680   5.613  10.643  30.135   7.172  16.204   7.963
Cumulative Var 18.879 383.559 389.172 399.815 429.950 437.121 453.325 461.288
```

```
lm(scale(y,scale=F)~p1$scores)$coefficients  =  p1$Yloadings
mean(y)+p1$scores[,1:2] %*% t(p1$Yloadings)[1:2] = p1$fitted[,,2]
```

Weights

$t_h = X_h w_h$

Loading.weights: matrix of w_h weights, they allow to find the PLS1 components if we were the deflacted X_h matrices

$T_h = X W_h^*$

Projection: matrix of w^*_h weights, they allow to find the PLS1 components.

```
scale(X,scale=F) %*% p1$projection
        Comp 1       Comp 2       Comp 3       Comp 4        Comp 5
1  -0.04355584 -0.094168122  0.059304733 -0.027416642  0.0048984359
2  -0.47297433 -0.016079722  0.047716531  0.004372191  0.0399353301
3  -0.32819384  0.125278987  0.033587783 -0.006358846  0.0083348621
4  -0.36599869 -0.108082608  0.052645154 -0.006288151  0.0073721207
5   0.18325237  0.010075274  0.115155405 -0.036793820 -0.0577039128
6  -0.11404121 -0.077129469  0.093466900 -0.026031651 -0.0023009946
7   0.13312203  0.044850102  0.085408581  0.008776140 -0.0124748634
8  -0.13332103  0.092581339  0.021212249 -0.001376719 -0.0145579276
9   0.06811378  0.059106888  0.072572426  0.011140154 -0.0203520345
10  0.13951770  0.027115291  0.083531131  0.029514858  0.0017714020
11  0.29452440 -0.029108496  0.107153819  0.054814779  0.0432770141
12  0.14523614 -0.010239936  0.093559990  0.005444119  0.0302146759
13 -0.13372843  0.037990197  0.019047594 -0.011617040  0.0252440834
14 -0.28049983  0.107538573  0.019261058  0.028024508 -0.0052030380
```

Coefficients

$$\hat{y}_h = Xb_h$$

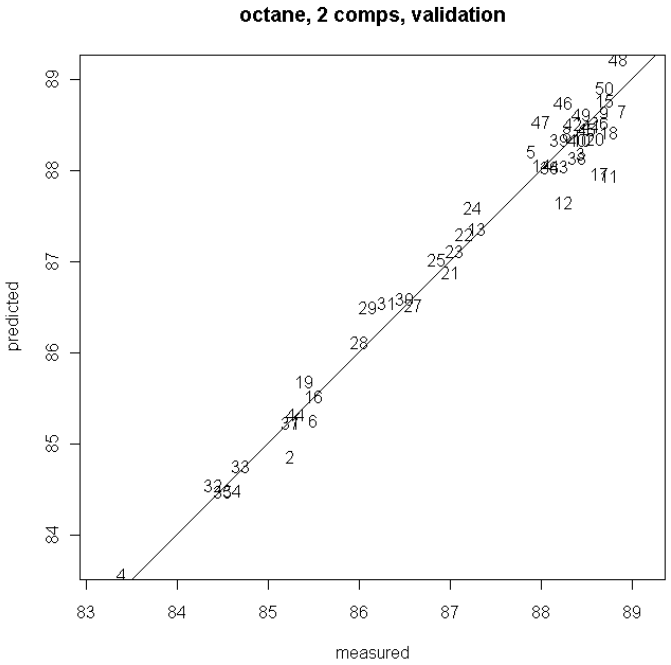
`Coefficients`: matrix of b_h coefficients, they allow to find the response from the (centered) X matrix

```
> p1$coefficients[,,5]
NIR.900.nm NIR.902.nm NIR.904.nm NIR.906.nm NIR.908.nm NIR.910.nm
 0.2479725  0.2690807  0.2905750  0.4321829  0.4536824  0.6020430
NIR.912.nm NIR.914.nm NIR.916.nm NIR.918.nm NIR.920.nm NIR.922.nm
 0.7029799  0.6941555  0.4874730  0.2779346 -0.0630905 -0.1383925
NIR.924.nm NIR.926.nm NIR.928.nm NIR.930.nm NIR.932.nm NIR.934.nm
-0.1787988 -0.2848536 -0.1901487 -0.1924330 -0.2034100 -0.1270751
NIR.936.nm NIR.938.nm NIR.940.nm NIR.942.nm NIR.944.nm NIR.946.nm
-0.1411060 -0.0512798  0.0009791  0.0245459 -0.0159240  0.0106277
NIR.948.nm NIR.950.nm NIR.952.nm NIR.954.nm NIR.956.nm NIR.958.nm
 0.0611463  0.0452529  0.1408813  0.1132197  0.1077124  0.1632347
NIR.960.nm NIR.962.nm NIR.964.nm NIR.966.nm NIR.968.nm NIR.970.nm
 0.2205944  0.2304395  0.3255974  0.3434137  0.3800613  0.4050635
NIR.972.nm NIR.974.nm NIR.976.nm NIR.978.nm NIR.980.nm NIR.982.nm
 0.4572848  0.4486218  0.5607187  0.5634353  0.6828395  0.6465780
```

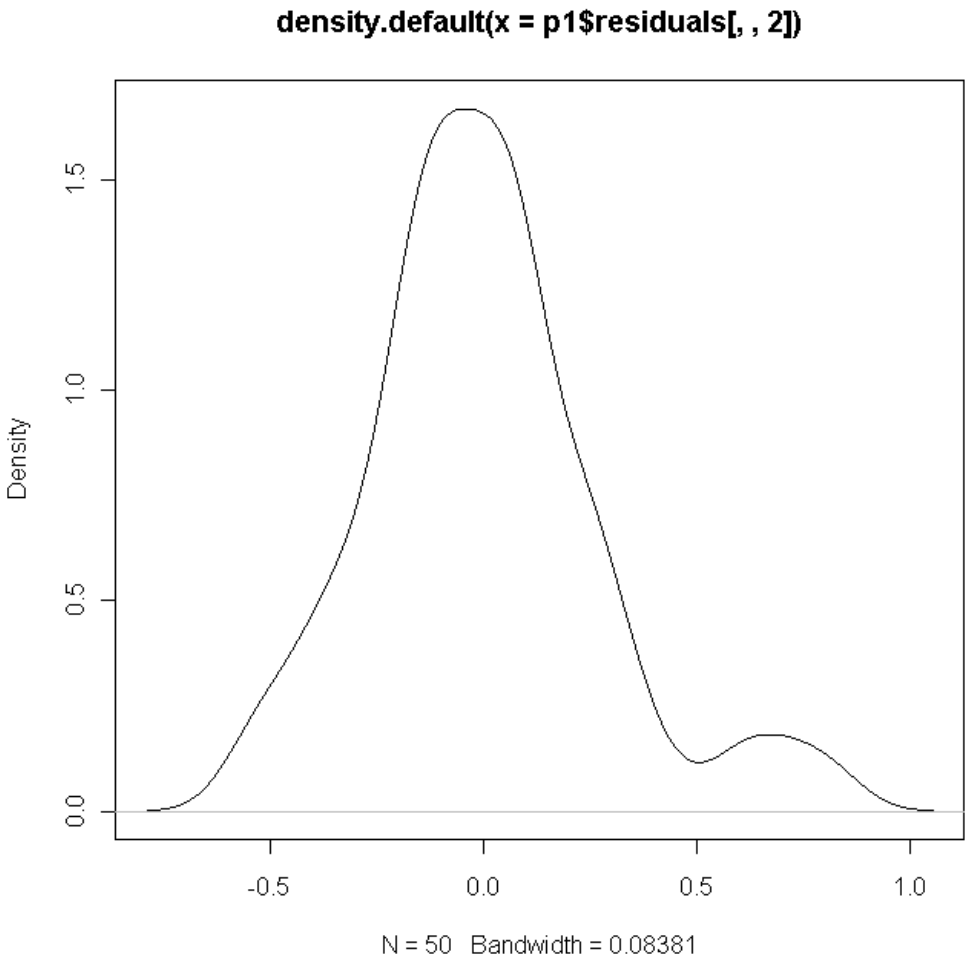
```
mean(y)+as.matrix(scale(X,scale=F)) %*% p1$coefficients[,,2] =
p1$fitted.values[,,2]
```

Fitted values

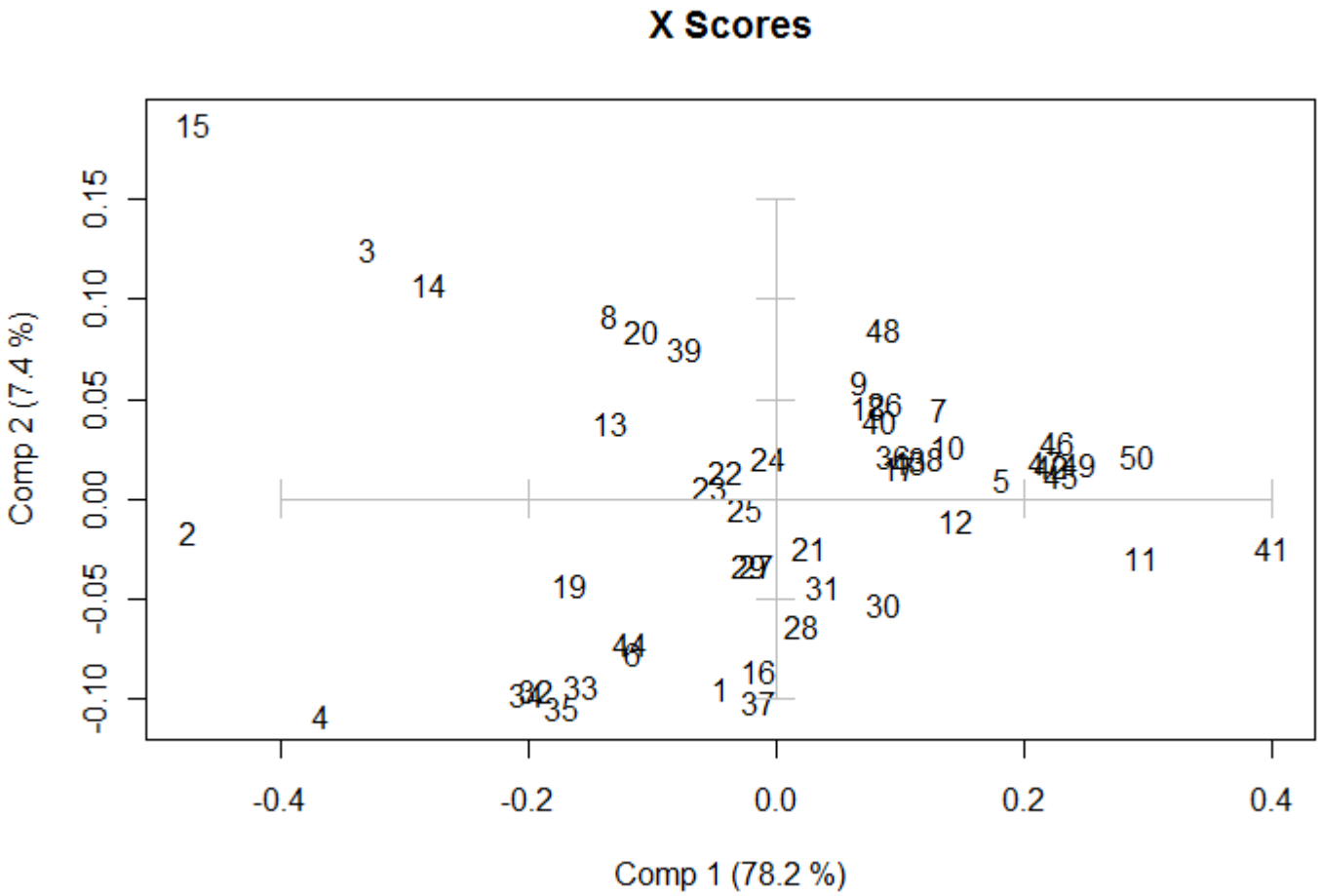
```
$fitted.values
 1     2     3     4     5     6     7     8     9    10    11    12
85.24 84.86 88.19 83.57 88.21 85.26 88.66 88.41 88.65 88.35 87.95 87.66
   13    14    15    16    17    18    19    20    21    22    23    24
87.37 88.06 88.77 85.53 87.97 88.43 85.69 88.36 86.89 87.31 87.12 87.59
   25    26    27    28    29    30    31    32    33    34    35    36
87.02 88.53 86.52 86.11 86.51 86.60 86.54 84.55 84.76 84.49 84.48 88.04
   37    38    39    40    41    42    43    44    45    46    47    48
85.24 88.15 88.34 88.34 88.49 88.51 88.05 85.33 88.46 88.75 88.54 89.23
   49    50
88.62 88.91
```



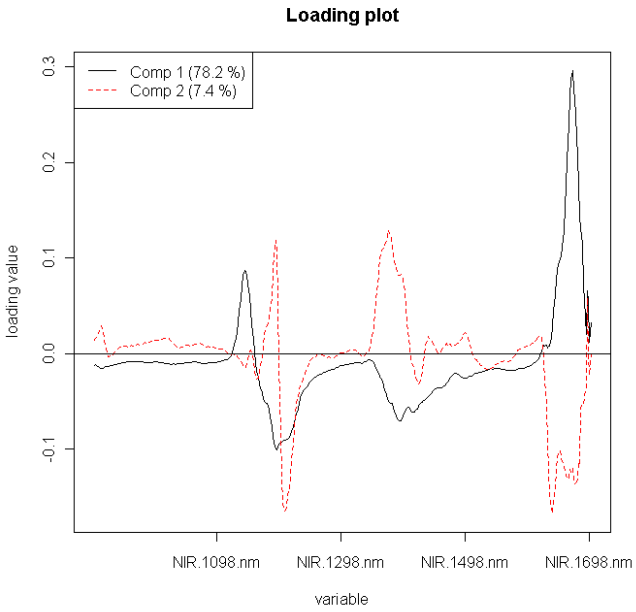
the residuals

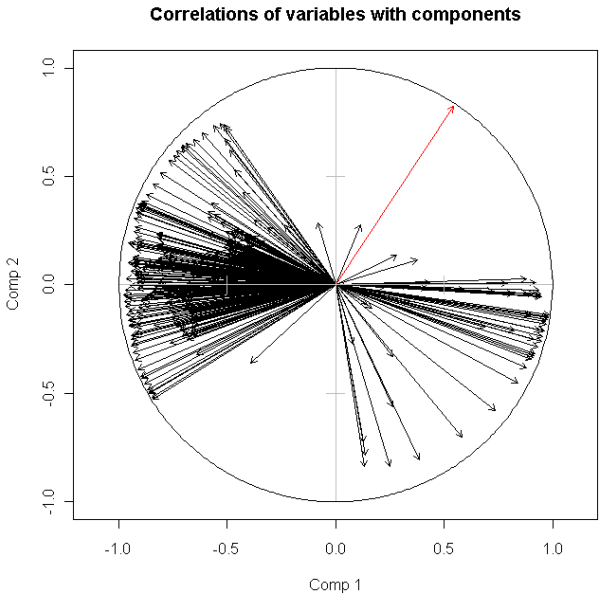


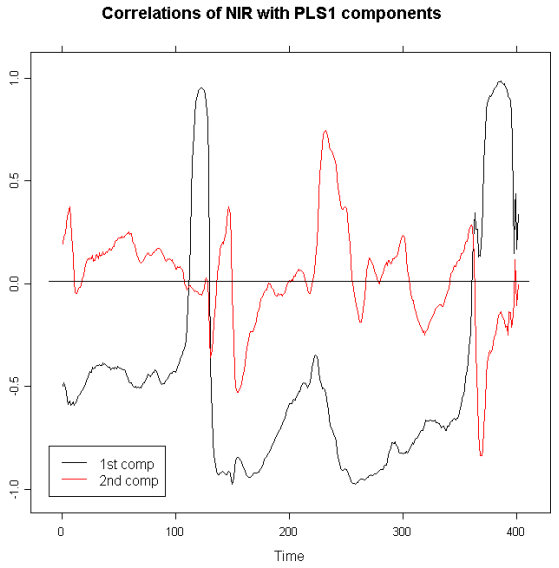
The training data: Scores plot



Loadings and correlations

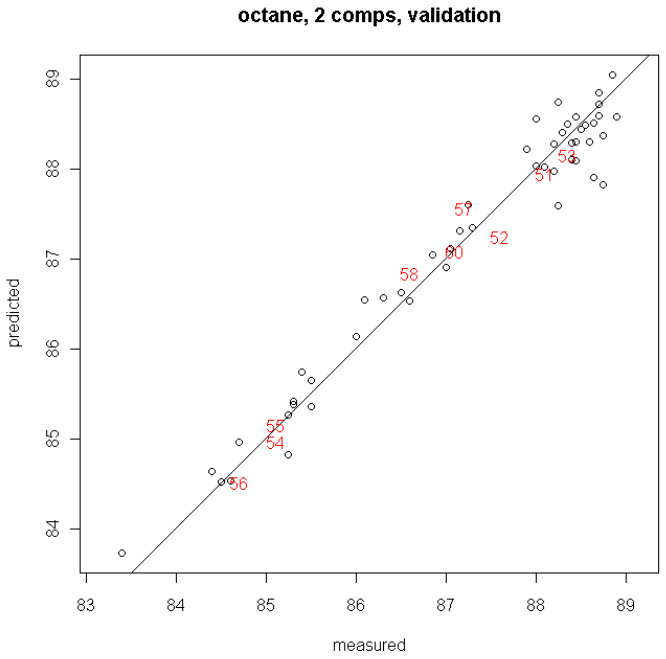




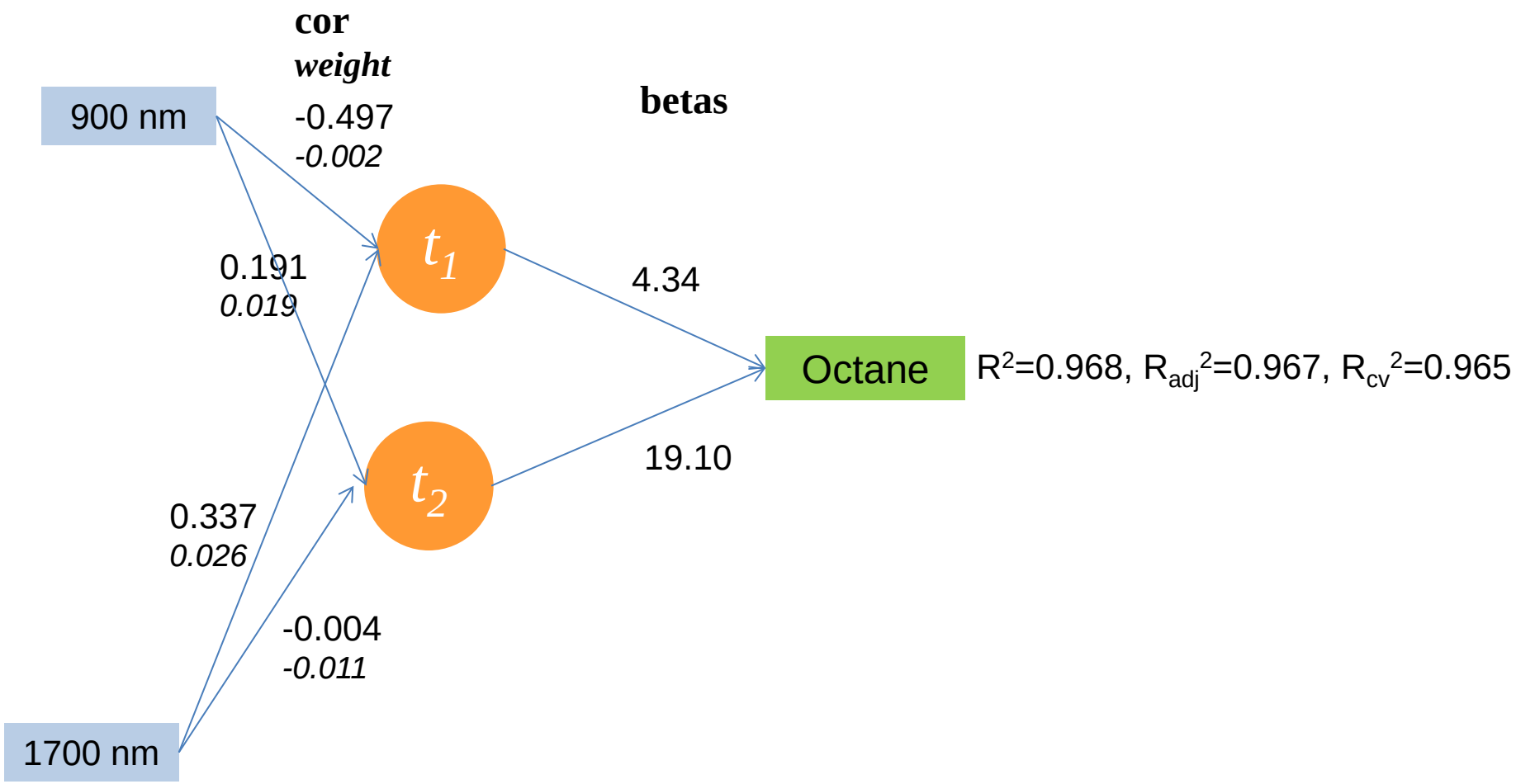


The PLS1 model

```
lm(formula = y ~ p1$scores[, 1:2])

Residuals:
     Min       1Q   Median       3Q      Max
-0.53706 -0.16282 -0.02681  0.11005  0.80215

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)            87.22400    0.03921 2224.53   <2e-16 ***
p1$scores[, 1:2]Comp 1  4.34504    0.20755   20.93   <2e-16 ***
p1$scores[, 1:2]Comp 2 19.09659    0.60209   31.72   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2773 on 47 degrees of freedom
Multiple R-squared: 0.9685,     Adjusted R-squared: 0.9671
F-statistic: 722.1 on 2 and 47 DF,  p-value: < 2.2e-16
```



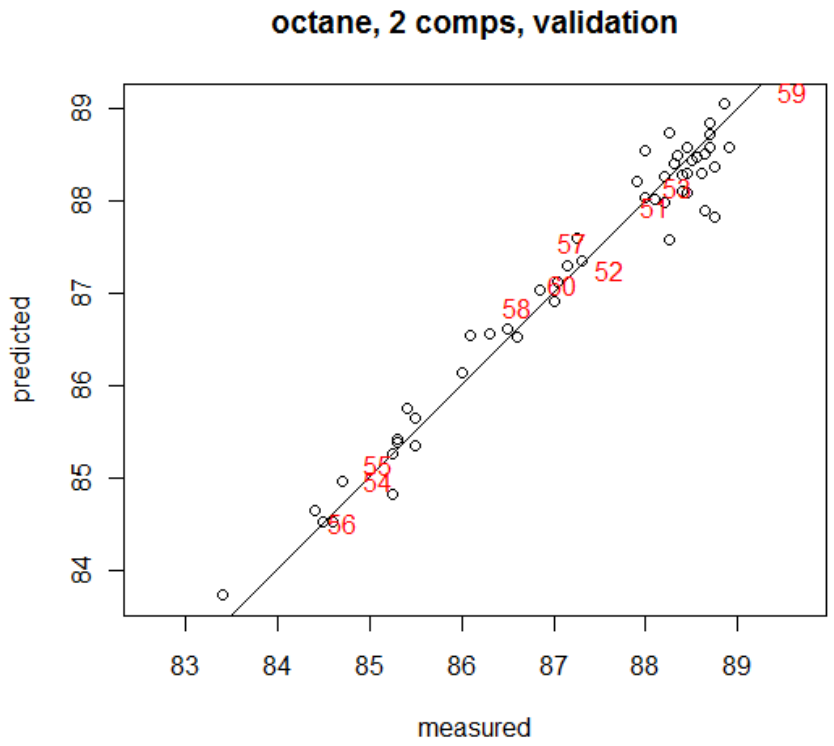
The PLS1 obtained octane model



Prediction test sample

Predicting the test data

```
pred_test <- predict(p1, ncomp = 2, newdata = gasTest)
```



$R^2_{test} = 0.974$