plsRglm: Algorithmic insights and applications

Bertrand, F., Magnanensi, J., Meyer, N. and Maumy-Bertrand, M.

Edited: June 2014; Compiled: June 26, 2014

Contents

Abstract

The aim of the *plsRglm* package is to deal with incomplete, as well as complete, data sets through several new techniques which had not been implemented *R* before. It features several bootstrap techniques -including (Y,X) and (Y,T) resampling-, leave-one-out and -doubly- repeated k-fold cross-validation on complete or incomplete datasets and, to boot, the extension of the Partial Least Squares (PLS) Regression, denoted by PLSR, to the generalized linear regression (GLR) models including binary or ordinal logistic PLSR. This package also provides formula support, several new classes and their generics, custom GLR models and graphics to assess the bootstrap based significance of the predictors.

Availability: `plsRglm` is freely available from the *R* archive CRAN. The software is distributed under the GNU General Public License (version 3 or later) and is accompanied by vignettes, demos and example datasets.
Contact: fbertran@math.unistra.fr

1 Motivation

The analysis of data sets with a large number of variables is firmly increasing, especially in medicine or in biology. However, major issues, including a strong linear correlation between the predictors -or even worse more predictors than subjects- or missing data, require the use of more involved statistical methods rather than, for instance, the ordinary least squares estimation of the linear regression. One of these methods is the Partial Least Square (PLS) Regression, denoted by PLSR, which was first introduced by Wold et al. (1983) and Wold et al. (1984). This method was already implemented on *R*, for instance in *pls* or in *plsdepot*, but we provide useful additions to the several existing packages including missing data, GLR or bootstrap support.

None of them can deal with incomplete data sets and they don't offer many other selection criteria for the number of components than cross-validation. Furthermore, the existing packages do not provide bootstrap techniques to assess the significance of the predictors neither operate with PLS generalizes linear regression models (PLSGLR). Finally, we provide support for weighted PLS.

The aim of this package is to fill these gaps and to enable *R* users:

- to fit PLSR or PLSGLR regression models (Bastien et al. (2005)) to complete or incomplete datasets,
- to implement weighted PLSR or PLSGLR (Haaland and Howland (1998)),

- to carry out -doubly- cross-validation using various criteria,
- to apply bootstrap techniques (Lazraq et al. (2003) and Bastien et al. (2005)) in order to obtain confidence intervals on the original predictors, not only in PLSR models but also in PLSGLR models, and assess thier significance.

The authors applied these models to some datasets and several of them are available in this package including a study to predict the quality of some Bordeaux wines (Tenenhaus, 1998) -ordinal logistic PLSGLR model- or an allelotyping dataset (Meyer et al., 2010) -binomial logistic PLSGLR model- with success.

2 Theory

2.1 PLS Regression

For a more detailed description of the PLS Regression see Höskuldsson (1988) and Wold et al. (2001). There are several algorithms to fit PLSR models. One of the more commonly used one is based on the NIPALS (Nonlinear estimation by Iterative PArtial Least Squares) algorithm which was first introduced by Wold et al. (1966). This algorithm was initially developed to analyse incomplete data sets through a PCA (Principal Component Analysis) and boils down to repeatively fit univariate linear regressions.

Consider $\mathbf{y}$ the response variable and $\mathbf{X}$ the matrix of predictors $\mathbf{x}_1, \ldots, \mathbf{x}_j, \ldots, \mathbf{x}_p$. All these variables are supposed to be centered and scaled.

As for many other models, it is sometimes convenient to use a vector of 'prior weights' in the fitting process. Suach as weighted version of the PLS Regression was introduced by Haaland and Howland (1998).

Remark 1. As in PCA, scaling variables is quite frequent to balance the effects of each variable. However, PLSR could be performed as well on unscaled variables as soon as they are centered (Bastien et al., 2005).

The PLS Regression is a non-linear method that designs new orthogonal components, denoted by $\mathbf{t}_1, \ldots, \mathbf{t}_H$. These are linear combinations of the predictors $\mathbf{x}_j$ whose construction is carried out in order to maximize their covariance with $\mathbf{y}$. Let $\mathbf{T}$ be the matrix formed by these components $\mathbf{t}_1, \ldots, \mathbf{t}_H$. The model is then:

$$\mathbf{y} = \mathbf{T}^t\mathbf{c} + \epsilon, \tag{1}$$

where ϵ represent the residual vector and ${}^t\mathbf{c}$ the vector formed by the regression coefficients of the components $\mathbf{t}_h$, ^{t} representing the transpose.

Then, let $\mathbf{T} = \mathbf{X}\mathbf{W}^*$, where $\mathbf{W}^*$ is the matrix of the coefficients of the predictors for each component $\mathbf{t}_h$, $1 \leqslant h \leqslant H$. Thus (1) can be rewritten as:

$$\mathbf{y} = \mathbf{X}\mathbf{W}^{*t}c + \epsilon. \tag{2}$$

By developing the right member of the equation (2), we derive for each component y_i of $\mathbf{y}$ the following expression:

$$y_i = \sum_{h=1}^{H} \left(c_h w^*_{1h} x_{i1} + \ldots + c_h w^*_{ph} x_{ip}\right) + \epsilon_i, \tag{3}$$

H being the number of components in the final model with $H \leqslant \mathrm{rk}\,(\mathbf{X})$. The coefficients $c_h w^*_{jh}$, where $1 \leqslant j \leqslant p$ and $1 \leqslant h \leqslant H$, with the notation * from Wold et al. (2001), represents the relationship between $\mathbf{y}$ and the variables $\mathbf{x}_j$ through the components $\mathbf{t}_h$.

2.2 PLS generalized linear regression

The extension of the PLS to the generalized linear regression model (PLSGLR) is one of the most attractive feature of this new package. This method is well described by Bastien et al. (2005) and allows to take account of missing data. The PLSGLR regression of the response $\mathbf{y}$ on the centered variables $\mathbf{x}_1, \ldots, \mathbf{x}_j, \ldots, \mathbf{x}_p$ with H components $\mathbf{t}_h = w^*_{1h}\mathbf{x}_1 + \ldots + w^*_{ph}\mathbf{x}_p$ is written as follows:

$$g(\theta)_i = \sum_{h=1}^{H} \left(c_h \sum_{j=1}^{p} w^*_{jh} x_{ij} \right), \tag{4}$$

with θ:

- a conditional expectation of the variable $\mathbf{y}$ if its distribution is continuous,
- a probability vector if the distribution of $\mathbf{y}$ a discrete law with a finite support.

The components $\mathbf{t}_h$ are built to be orthogonal and the link function g is chosen according to the distribution of $\mathbf{y}$ to best fit the model to the data.

Remark 2.

1. Like in classical PLSR, the main idea is to build orthogonal components $\mathbf{t}_h$. However, because of the GLR structure, it is impossible, for each step of the algorithm -that adds a new component- to use the residuals ϵ_{h-1} of the multiple regression of $\mathbf{y}$ on $\mathbf{t}_1, \ldots, \mathbf{t}_{h-1}$. So, the algorithm includes at the h-step the previously computed components $\mathbf{t}_1, \ldots, \mathbf{t}_{h-1}$ as covariables when fitting the GLR of $\mathbf{y}$ on each of the predictor $\mathbf{x}_j$.
2. However, in order to ensure orthogonality between the different components, it is necessary to express each supplementary component in terms of residuals obtained by the Ordinary Least Squares (OLS) linear regression of the $\mathbf{x}_j$ on the previously found $\mathbf{t}_h$ components, noted $\tilde{\mathbf{x}}_{hj}$, that is:

$$t_{h+1} = \frac{1}{\sum_{j=1}^{p} a_{h+1,j}^2} \sum_{j=1}^{p} a_{h+1,j} \tilde{\mathbf{x}}_{hj} \tag{5}$$

   where $a_{h+1,j}$ is the regression coefficient of $\tilde{\mathbf{x}}_{hj}$ in the GLR of $\mathbf{y}$ on $\mathbf{t}_1, \ldots, \mathbf{t}_h$ and $\tilde{\mathbf{x}}_{hj}$. Indeed, by proceeding to a GLR of the variables $\mathbf{x}_j$ on the components $\mathbf{t}_h$ already build in order to extract the residuals, these ones would not necessarily be orthogonal to the components $\mathbf{t}_h$ since the parameter estimation is done by Maximum Likelihood (ML) and not by OLS.
3. Note that to compute $a_{h+1,j}$, one must be able to carry out the regression of $\mathbf{y}$ on $\mathbf{t}_1, \ldots, \mathbf{t}_h$ and $\mathbf{x}_j$. Let p_{hj} be the regression coefficients of $\mathbf{t}_h$ in the multiple linear regression by OLS of $\mathbf{x}_j$ on $\mathbf{t}_h$, then:

$$\begin{aligned}\mathbf{y} &= c_1\mathbf{t}_1 + c_2\mathbf{t}_2 + \ldots + c_h\mathbf{t}_h + a_{h+1,j}\mathbf{x}_j \\ &= c_1\mathbf{t}_1 + \ldots + c_h\mathbf{t}_h + a_{h+1,j}\left(p_{1j}\mathbf{t}_1 + \ldots + p_{hj}\mathbf{t}_h + \tilde{\mathbf{x}}_{hj}\right) \\ &= \left(c_1 + a_{h+1,j}p_{1j}\right)\mathbf{t}_1 + \ldots + a_{h+1,j}\tilde{\mathbf{x}}_{hj}\end{aligned}$$

4. We propose an enhanced version of PLSGLR so that a vector of 'prior weights' can be used in the fitting process, as Haaland and Howland (1998) did it for classical PLSR.

2.2.1 PLSGLR algorithm

These remarks being done, here is the algorithm implemented in this new package allowing to determine the PLS components $\mathbf{t}_h$ of a PLSGLR model.

- The variables $\mathbf{x}_j$ are centered and scaled.
- *Determination of the first component* $\mathbf{t}_1$*:*
  1. Compute the coefficients a_{1j} of $\mathbf{x}_j$ in the generalized linear regression of $\mathbf{y}$ on $\mathbf{x}_j$, $j = 1, \ldots, p$.
  2. Normalize the vector $\mathbf{a}_1 = (a_{1j})_{j=1,\ldots,p}$:

  $$\mathbf{w}_1 = \mathbf{a}_1 / \left\|\mathbf{a}_1\right\|_2.$$

  3. Determine the component $\mathbf{t}_1$:

  $$\mathbf{t}_1 = \mathbf{X}\mathbf{w}_1 / {}^t\mathbf{w}_1\mathbf{w}_1.$$

- *Determination of the second component* $\mathbf{t}_2$*:*
  1. Compute the coefficients a_{2j} of $\mathbf{x}_j$ in the generalized linear regression of $\mathbf{y}$ on $\mathbf{t}_1$ and $\mathbf{x}_j$, $j = 1, \ldots, p$.
  2. Normalize the vector $\mathbf{a}_2 = (a_{2j})_{j=1,\ldots,p}$:

  $$\mathbf{w}_2 = \mathbf{a}_2 / \left\|\mathbf{a}_2\right\|_2.$$

  3. Find the residual matrix $\mathbf{X}_1$ of the linear regression by OLS of $\mathbf{X}$ on $\mathbf{t}_1$.
  4. Determine the component $\mathbf{t}_2$:

  $$\mathbf{t}_2 = \mathbf{X}_1\mathbf{w}_2 / {}^t\mathbf{w}_2\mathbf{w}_2.$$

  5. Express $\mathbf{t}_2$ in terms of $\mathbf{X}$: $\mathbf{t}_2 = \mathbf{X}\mathbf{w}^*{}_2$.
- *Determination of the* h^{th} *component* $\mathbf{t}_h$*:*
  During the previous steps, the components $\mathbf{t}_1, \ldots, \mathbf{t}_{h-1}$ have been obtained. The component $\mathbf{t}_h$ is obtained by repeating the steps done for $\mathbf{t}_2$.
  1. Compute the coefficients a_{hj} of $\mathbf{x}_j$ in the GLR regression of $\mathbf{y}$ on $\mathbf{t}_1, \ldots, \mathbf{t}_{h-1}$ and $\mathbf{x}_j$, $j = 1, \ldots, p$.
  2. Normalize the vector $\mathbf{a}_h = (a_{hj})_{j=1,\ldots,p}$:

  $$\mathbf{w}_h = \mathbf{a}_h / \left\|\mathbf{a}_h\right\|_2.$$

  3. Find the residual matrix $\mathbf{X}_{h-1}$ of the linear regression by OLS of $\mathbf{X}$ on $\mathbf{t}_1, \ldots, \mathbf{t}_{h-1}$.

4. Determine the component $\mathbf{t}_h$:
$$\mathbf{t}_h = \mathbf{X}_{h-1}\mathbf{w}_h / {}^t\mathbf{w}_h\mathbf{w}_h.$$
5. Express $\mathbf{t}_h$ in terms of $\mathbf{X}$: $\mathbf{t}_h = \mathbf{X}\mathbf{w}^*{}_h$.

Remark 3. The algorithm shown above works even with incomplete data sets. Indeed, each coordinate t_{hi} can be expressed as $t_{hi} =^t \mathbf{x}_{h-1,i}\mathbf{w}_h / {}^t\mathbf{w}_h\mathbf{w}_h$ where $\mathbf{x}_{h-1,i}$ is the vector obtained as the transpose of the i^{th} line of $\mathbf{X}_{h-1}$. As a result, a component $\mathbf{t}_h$ can be seen as the slope of the fitted line of the univariate OLS linear regression without intercept of $\mathbf{x}_{h-1,i}$ on $\mathbf{w}_h$. Such a slope can be derived even when there are missing values in the dataset.

. Likewise, the denominator ${}^t\mathbf{w}_h\mathbf{w}_h$ will be calculate only on the available data.

2.2.2 Stopping criterion in the construction of components

It is one of the attractive features of our package. Indeed, it contains several criterion in order to determine the optimal number of components to build, some of them that did not already exist in other *R* packages. So, the user will obtain different criterion like the AIC, BIC, Q^2 by cross-validation, the number of miss-classified if the data are binary or ordinal discrete or the stop of significance of a component when no coefficient a_{h+1} is significant anymore (Bastien et al., 2005). The AIC and BIC obtained are, in case of a PLSR without missing data, those calculated with the corrected degrees of freedom (Krämer and Sugiyama, 2011). They are noted `AIC.dof` and `BIC.dof` when `AIC.naive` and `BIC.naive` are those calculated by treating each component as a classical variable, that is to say corresponding to one degree of freedom. In all the other cases, only the naive AIC and BIC are available.

2.3 Bootstrap

Another greatly attractive feature of this package consists in the implementation of several bootstrap techniques to compute confidence intervals and hence carry out significance tests of the predictors $\mathbf{x}_j$, $j = 1, \ldots, p$ of either a PLSR or a PLSGLR model.

We propose two kinds of bootstrap in the *plsRglm* package for either the PLSR models or the PLSGLR models:

- sample $(\mathbf{y}, \mathbf{X})$,
- sample $(\mathbf{y}, \mathbf{T})$.

The $(\mathbf{y}, \mathbf{X})$ bootstrap was introduced by (Lazraq et al., 2003) for PLSR models whereas the $(\mathbf{y}, \mathbf{T})$ was proposed by Bastien et al. (2005) for the PLSGLR setting.

The implementation is based on the `boot` function of the *boot* package Canty and Ripley (2014); Davison and Hinkley (1997) and designed, except for `permutation` bootstrap, to push forward any bootstrap option to the underlying the `boot` function. For instance, the `sim` option can be used to produce `ordinary` (the default), `parametric`, `balanced`, `permutation`, or `antithetic` bootstrap. Parallel computing options, including `parallel`, `ncpus` or `cl`, can also be used with a proper setting. The `strata` option is used in Bordeaux Wine Quality example.

2.3.1 Bootstrap in PLSR

Bootstrap $(\mathbf{y}, \mathbf{X})$ in PLSR

(Lazraq et al., 2003) proposed the following algorithm to perform the $(\mathbf{y}, \mathbf{X})$ bootstrap in PLSR.

First, here are some notations:

- Let L be the number of bootstrap samples:
- We note $\mathbf{w}_l^b : (\mathbf{y}_l^b, \mathbf{X}_l^b) = \left\{ \left(y_\alpha^b x_{1,\alpha}^b \ldots x_{p,\alpha}^b \right), \alpha = 1, \ldots, n \right\}$ the l^{th} bootstrap sample of size n got by sampling with replacement, $l = 1, \ldots, L$, with $\mathbf{y}_l^b \in M_{n\times 1}, \mathbf{X}_l^b \in M_{n\times p}$ *and* $\mathbf{w}_l^b \in M_{n\times(p+1)}$.
- For each of these bootstrap sample l, we do a PLS regression of $\mathbf{y}_l^b$ on $\mathbf{X}_l^b$, giving us the components $\mathbf{t}_{h,l}$, $h = 1, \ldots, H_l$ from which we obtain as a result, noted $\mathbf{B}_l^b$, the coefficient vector for the original variables, $\mathbf{B}_l^b = (b_{1l}^b, \ldots, b_{pl}^b)$.

The algorithm

```
1. Repeat for $l = 1, \ldots, L$:
    • Get a sample with replacement of size $n$: $\mathbf{w}_l^b = (\mathbf{y}_l^b, \mathbf{X}_l^b)$.
    • Calculate $\mathbf{B}_l^b$ the result of the PLSR of $\mathbf{y}_l^b$ on $\mathbf{X}_l^b$.
2. Repeat for $k = 1, \ldots, p$:
    • Let $E_k$ be the vector $(b_{k1}^b, b_{k2}^b, \ldots, b_{kL}^b) \in M_{1\times L}$, where $E_k$ is a bootstrap sample of size $L$ of $b_k$, the coefficient of $\mathbf{x}_k$ in the PLS regression of $\mathbf{y}$ on $\mathbf{X}$.
    • Get a confidence interval $I_k^b$ for $b_k$.
    • If $0 \in I_k^b$, remove the variable $\mathbf{x}_k$
3. Send back the list of significant predictors.
```

Remark 4.

1. The intervals obtained by the previous bootstrap techniques are designed to carry out pairwise or multiple comparisons and must be interpreted separately.
2. Several methods for the obtaining of confidence intervals are available in our package, we also can found the following construction: normal, basic, percentiles or BC_a (Efron and Tibshirani, 1993). Lazraq et al. (2003) use the method based on an asymptotic normal distribution with unknown parameters and also add a mean calculation of the b_{kl}^b and of the variances, in order to build a confidence interval with the help of normal quantiles.

Bootstrap $(\mathbf{y}, \mathbf{T})$ in PLSR

We propose the following algorithm to perform the $(\mathbf{y}, \mathbf{T})$ bootstrap in PLSR following what Bastien et al. (2005) introduced for the PLSGLR setting.

We suppose that the correct number m of components was retained for a PLSR model of $\mathbf{y}$ on $\mathbf{x}_1, \ldots, \mathbf{x}_p$.

Let $\widehat{F}_{(\mathbf{T}|\mathbf{y})}$ the empirical cumulative distribution function given the matrix $\mathbf{T}$ formed by the m PLS components and the response $\mathbf{y}$.

The algorithm

1. Get B samples of $\widehat{F}_{(\mathbf{T}|\mathbf{y})}$.
2. For all $b = 1, \ldots, B$, calculate:
$$\mathbf{c}^b = \left({}^t\mathbf{T}^b\mathbf{T}^b\right)^{-1t}\mathbf{T}^b\mathbf{y}^b \text{ et } \mathbf{b}^b = \mathbf{W}^*\mathbf{c}^b,$$
where $\left[\mathbf{T}^b, \mathbf{y}^b\right]$ is the b^{th} bootstrap sample, c^b the vector of coefficients of the components and $\mathbf{b}^b$ is the vector of coefficient of the p original predictors for this sample. Finally, given the fact that sampling is carried out with replacement on $\mathbf{y}$ and $\mathbf{T}$, the matrix $\mathbf{W}^*$ remains fixed during all the bootstrap re-sampling and represent also the weights of the predictors in the original model having m components.
3. For each $j = 1, \ldots, p$, denote by Φ_{b_j} the Monte-Carlo approximation of the cumulative distribution function of the bootstrap of b_j.
For each b_j, boxplots and confidence intervals can be derived using the percentiles of Φ_{b_j}. The confidence interval is also defined as $I_j(\alpha) = \left[\Phi_{b_j}^{-1}(\alpha), \Phi_{b_j}^{-1}(1-\alpha)\right]$ where $\Phi_{b_j}^{-1}(\alpha)$ and $\Phi_{b_j}^{-1}(1-\alpha)$ are the obtained values from the bootstrap cumulative distribution function so that a nominal level of confidence $100(1-2\alpha)$ is reached.
However, in order to improve the quality of the confidence interval in terms of coverage, that is to say the capacity of $I_j(\alpha)$ to give some precise coverage, it is possible to use many other construction techniques: normal, basic or BC_a (Efron and Tibshirani, 1993).

2.3.2 Bootstrap in PLSGLR

Bootstrap $(\mathbf{y}, \mathbf{X})$ in PLSGLR

We propose, following what (Lazraq et al., 2003) did for the PLSR models, the following algorithm to perform the $(\mathbf{y}, \mathbf{X})$ bootstrap in PLSGLR.

First, here are some notations:

- Let L be the number of bootstrap samples:
- We note $\mathbf{w}_l^b : \left(\mathbf{y}_l^b, \mathbf{X}_l^b\right) = \left\{\left(y_\alpha^b x_{1,\alpha}^b \ldots x_{p,\alpha}^b\right), \alpha = 1, \ldots, n\right\}$ the l^{th} bootstrap sample of size n got by sampling with replacement, $l = 1, \ldots, L$, with $\mathbf{y}_l^b \in M_{n\times 1}, \mathbf{X}_l^b \in M_{n\times p}\ and\ \mathbf{w}_l^b \in M_{n\times(p+1)}$.
- For each of these bootstrap sample l, we do a PLS regression of $\mathbf{y}_l^b$ on $\mathbf{X}_l^b$, giving us the components $\mathbf{t}_{h,l},\ h = 1, \ldots, H_l$ from which we obtain as a result, noted $\mathbf{B}_l^b$, the coefficient vector for the original variables, $\mathbf{B}_l^b = \left(b_{1l}^b, \ldots, b_{pl}^b\right)$.

The algorithm

1. Repeat for $l = 1, \ldots, L$:
    - Get a sample with replacement of size n: $\mathbf{w}_l^b = \left(\mathbf{y}_l^b, \mathbf{X}_l^b\right)$.
    - Calculate $\mathbf{B}_l^b$ the result of the PLSR of $\mathbf{y}_l^b$ on $\mathbf{X}_l^b$.
2. Repeat for $k = 1, \ldots, p$:
    - Let E_k be the vector $\left(b_{k1}^b, b_{k2}^b, \ldots, b_{kL}^b\right) \in M_{1\times L}$, where E_k is a bootstrap sample of size L of b_k, the coefficient of $\mathbf{x}_k$ in the PLS regression of $\mathbf{y}$ on $\mathbf{X}$.

    - Get a confidence interval I_k^b for b_k.
    - If $0 \in I_k^b$, remove the variable $\mathbf{x}_k$
3. Send back the list of significant predictors.

Bootstrap $(\mathbf{y}, \mathbf{T})$ in PLSGLR

Bastien et al. (2005) proposed a bootstrap $(\mathbf{y}, \mathbf{T})$ based algorithm to compute confidence intervals as well as carry out tests of significance for the predictors $\mathbf{x}_j$, $j = 1, \ldots, p$ in the PLSGLR setting.

We suppose that the correct number m of components was retained for a PLSGLR model of $\mathbf{y}$ on $\mathbf{x}_1, \ldots, \mathbf{x}_p$.

Let $\widehat{F}_{(\mathbf{T}|\mathbf{y})}$ the empirical cumulative distribution function given the matrix $\mathbf{T}$ formed by the m PLS components and the response $\mathbf{y}$.

The algorithm

1. Get B samples of $\widehat{F}_{(\mathbf{T}|\mathbf{y})}$.
2. For all $b = 1, \ldots, B$, calculate:
$$\mathbf{c}^b = \left({}^t\mathbf{T}^b\mathbf{T}^b\right)^{-1t}\mathbf{T}^b\mathbf{y}^b \text{ et } \mathbf{b}^b = \mathbf{W}^*\mathbf{c}^b,$$
where $\left[\mathbf{T}^b, \mathbf{y}^b\right]$ is the b^{th} bootstrap sample, c^b the vector of coefficients of the components and $\mathbf{b}^b$ is the vector of coefficient of the p original predictors for this sample. Finally, given the fact that the sampling is carried out with replacement on $\mathbf{y}$ and $\mathbf{T}$, the matrix $\mathbf{W}^*$ remains fixed during all the bootstrap re-sampling and represent also the weights of the predictors in the original model having m components.
3. For each $j = 1, \ldots, p$, denote by Φ_{b_j} the Monte-Carlo approximation of the cumulative distribution function of the bootstrap of b_j.
For each b_j, boxplots and confidence intervals can be derived using the percentiles of Φ_{b_j}. The confidence interval is also defined as $I_j(\alpha) = \left[\Phi_{b_j}^{-1}(\alpha), \Phi_{b_j}^{-1}(1-\alpha)\right]$ where $\Phi_{b_j}^{-1}(\alpha)$ and $\Phi_{b_j}^{-1}(1-\alpha)$ are the obtained values from the bootstrap cumulative distribution function so that a nominal level of confidence $100(1-2\alpha)$ is reached.
However, in order to improve the quality of the confidence interval in terms of coverage, that is to say the capacity of $I_j(\alpha)$ to give some precise coverage, it is possible to use many other construction techniques: normal, basic or BC_a (Efron and Tibshirani, 1993).

Remark 5.

1. The intervals obtained by the previous bootstrap techniques are designed to carry out pairwise or multiple comparisons and must be interpreted separately.
2. This algorithm has the benefit of being fast in terms of execution time since the re-sampling is done on the $\mathbf{t}_h$ and not on the $\mathbf{x}_j$. Furthermore, the estimation of $\mathbf{c}^b$ is done via the resolution of the normal equations, equations which give estimations in the PLS framework and not in the PLSGLR one (except in the case of the Gaussian distribution) in which the estimations are done by the maximum likelihood technique via an iterative method. It also seems clear that this algorithm is interesting because of its execution speedas well as its reduced sensitivity to the re-sampling. Indeed, some extreme replications can lead to a non-convergence of the algorithm during the maximization likelihood or more often a divergence of the estimates of the parameters leading to very important variability and hence to the potential non-significance of some variables. Some others methods have been developed in order to deal with these issues, for instance Moulton and Zeger (1991) have developed a bootstrap estimation method in the GLR framework consisting in only doing one step in the Newton-Raphson algorithm avoiding in this way these problems.

3 Applications

3.1 Data

The `plsRglm` package contains some interesting datasets including:

- the Cornell dataset (Kettaneh-Wold, 1992),
- a study on the pine processionary caterpillars (Tomassone et al., 1992),
- an allelotyping study on cancer cells dataset with missing values (Meyer et al., 2010),
- a Bordeaux wines quality study (Tenenhaus, 1998).

The package was also applied to the `Phenyl` and `Hyptis` datasets from the *chemometrics* and the `colonCA` dataset from the *colonCA* package.

3.2 PLS regression: Cornell

Cross-validation

In this example, we will use formula specification of the PLS model.

```
rm(list = ls())
library(plsRglm)
data(Cornell)
```

We use $k = 6$ balanced groups of 2 subjects to perform repeated k-fold cross validation. We set to 10, thanks to the option `nt=6`, the maximal number of components for the cross-validation function -`cv.plsR`- since the rank of the design matrix is equal to 6. The `grouplist` option enables the user to provide custom splits of the datasets on which cross validation will be carried out. As a consequence, one can use the *caret* (from Jed Wing et al., 2014) package to find balanced splits of the dataset into folds with respect to the response values.

```
cv.modpls<-cv.plsR(Y~.,data=Cornell,nt=6,K=6)
```

We sum up the results in a single table using the `summary`.

```
res.cv.modpls<-cvtable(summary(cv.modpls))
## ____************************************************____
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Component____ 5 ____
## ____Component____ 6 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****
##
##
## NK: 1
##
##
## CV Q2 criterion:
## 0 1
## 0 1
##
## CV Press criterion:
## 1 2 3 4
## 0 0 0 1
```

You can perform leave one out cross validation similar to the one that existed in previous versions of SIMCA by setting `TypeVC="standard"`. Two other options, `TypeVC="missing"` or `TypeVC="standard"`, exists to handle incomplete datasets. Indeed, of cross validation is required is that case, one needs to selects the way of predicting the response for left out observations. For complete rows, without any missing value, there are two different ways of computing these predictions. As a consequence, for mixed datasets, with complete and incomplete rows, there are two ways of computing prediction : either predicts any row as if there were missing values in it (`missingdata`) or selects the prediction method accordingly to the completeness of the row (`adaptative`).

```
res6<-plsR(Y~.,data=Cornell, nt=6, typeVC="standard", pvals.expli=TRUE)
## ____************************************************____
## ____TypeVC____ standard ____
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Component____ 5 ____
## ____Component____ 6 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****
colSums(res6$pvalstep)
## [1] 0 0 3 0 0 0
res6$InfCrit
##             AIC Q2cum_Y LimQ2_Y     Q2_Y PRESS_Y   RSS_Y   R2_Y
## Nb_Comp_0 82.01      NA      NA       NA      NA 467.797     NA
## Nb_Comp_1 53.15  0.8967  0.0975  0.89666  48.344  35.742 0.9236
## Nb_Comp_2 41.08  0.9175  0.0975  0.20211  28.519  11.067 0.9763
## Nb_Comp_3 32.06  0.9400  0.0975  0.27196   8.057   4.418 0.9906
## Nb_Comp_4 33.76  0.9197  0.0975 -0.33760   5.910   4.309 0.9908
## Nb_Comp_5 33.34  0.9281  0.0975  0.10506   3.857   3.522 0.9925
## Nb_Comp_6 35.26  0.9233  0.0975 -0.06792   3.761   3.496 0.9925
##           R2_residY RSS_residY PRESS_residY Q2_residY  LimQ2
## Nb_Comp_0        NA   11.00000           NA        NA     NA
## Nb_Comp_1    0.9236    0.84047      1.13679   0.89666 0.0975
## Nb_Comp_2    0.9763    0.26023      0.67060   0.20211 0.0975
## Nb_Comp_3    0.9906    0.10389      0.18945   0.27196 0.0975
## Nb_Comp_4    0.9908    0.10133      0.13896  -0.33760 0.0975
## Nb_Comp_5    0.9925    0.08282      0.09068   0.10506 0.0975
## Nb_Comp_6    0.9925    0.08221      0.08844  -0.06792 0.0975
##           Q2cum_residY AIC.std DoF.dof sigmahat.dof AIC.dof BIC.dof
## Nb_Comp_0           NA  37.010   1.000       6.5213 46.0709 47.7894
## Nb_Comp_1       0.8967   8.150   2.741       1.8665  4.5700  4.9558
## Nb_Comp_2       0.9175  -3.919   5.086       1.1825  2.1075  2.3949
## Nb_Comp_3       0.9400 -12.938   5.121       0.7488  0.8468  0.9628
## Nb_Comp_4       0.9197 -11.237   5.103       0.7387  0.8233  0.9358
## Nb_Comp_5       0.9281 -11.658   6.006       0.7096  0.7976  0.9198
## Nb_Comp_6       0.9233  -9.746   7.000       0.7633  0.9711  1.1360
##           GMDL.dof DoF.naive sigmahat.naive AIC.naive BIC.naive
## Nb_Comp_0    27.59         1         6.5213   46.0709   47.7894
## Nb_Comp_1    21.34         2         1.8906    4.1700    4.4588
## Nb_Comp_2    27.40         3         1.1089    1.5370    1.6861
## Nb_Comp_3    24.41         4         0.7431    0.7363    0.8256
## Nb_Comp_4    24.23         5         0.7846    0.8721    0.9965
## Nb_Comp_5    28.21         6         0.7662    0.8805    1.0228
## Nb_Comp_6    33.18         7         0.8362    1.1071    1.3049
##           GMDL.naive
## Nb_Comp_0      27.59
## Nb_Comp_1      18.38
## Nb_Comp_2      17.71
## Nb_Comp_3      19.01
## Nb_Comp_4      24.17
## Nb_Comp_5      28.64
## Nb_Comp_6      33.64
```

The number of significant predictors per components, which is a criteria of significance for Bastien et al. (2005), can be obtained via the following code:

```
res6<-plsR(Y~.,data=Cornell, nt=6, pvals.expli=TRUE)
## ____************************************************____
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Component____ 5 ____
## ____Component____ 6 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****
```

```
colSums(res6$pvalstep)
```

```
## [1] 0 0 3 0 0 0
```

The number of significant predictors within each component tell us to only build 3 components when the AIC criteria gives us 5 components and the BIC concludes to 5 components. The cross-validated Q^2_{cum} criterion advocates for retaining 3 components either for leave one out and 1 for 6-fold CV. The 6-fold CV cross-validation was run 100 times by randomly creating groups. Here are the command lines:

```
set.seed(123)
cv.modpls<-cv.plsR(Y~.,data=Cornell,nt=6,K=6,NK=100,random=TRUE)
```

```
res.cv.modpls=cvtable(summary(cv.modpls))
```

```
## ____************************************************____
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Component____ 5 ____
## ____Component____ 6 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****
##
##
## NK: 1,  2,  3,  4,  5,  6,  7,  8,  9,  10
## NK: 11,  12,  13,  14,  15,  16,  17,  18,  19,  20
## NK: 21,  22,  23,  24,  25,  26,  27,  28,  29,  30
## NK: 31,  32,  33,  34,  35,  36,  37,  38,  39,  40
## NK: 41,  42,  43,  44,  45,  46,  47,  48,  49,  50
## NK: 51,  52,  53,  54,  55,  56,  57,  58,  59,  60
## NK: 61,  62,  63,  64,  65,  66,  67,  68,  69,  70
## NK: 71,  72,  73,  74,  75,  76,  77,  78,  79,  80
## NK: 81,  82,  83,  84,  85,  86,  87,  88,  89,  90
## NK: 91,  92,  93,  94,  95,  96,  97,  98,  99,  100
##
##
## CV Q2 criterion:
##  0  1  2
##  0 89 11
##
## CV Press criterion:
##  1  2  3  4  5
##  0  0 33 55 12
```

The results, based on the use of the Q^2 criterion, (Fig. 1) confirm those of the first 6-fold CV cross validation: we decide to retain 1 components. Even in the linear case, cross validation should be repeated to select the number of components in a PLSR model.

```
plot(res.cv.modpls)
```

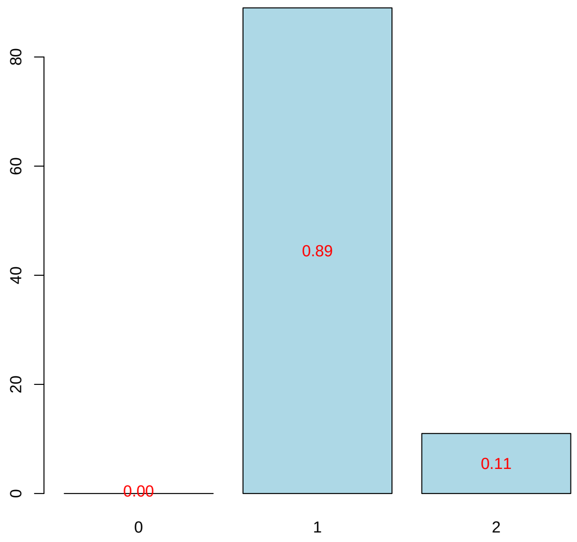

Figure 1: Nb components, 6-CV, n=100

Now, the PLSGLR regression is done in order to obtain these coefficients c_h and the intercept.

```
res<-plsR(Y˜.,data=Cornell,nt=1,pvals.expli=TRUE)

## ____************************************************____
## ____Component____ 1 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****

res

## Number of required components:
## [1] 1
## Number of successfully computed components:
## [1] 1
## Coefficients:
##                [,1]
## Intercept  92.4322
## X1        -14.8846
## X2         -0.5942
## X3        -25.5424
## X4         -5.1075
## X5         14.1877
## X6          5.5177
## X7        -44.9000
## Information criteria and Fit statistics:
##              AIC  RSS_Y   R2_Y R2_residY RSS_residY AIC.std DoF.dof
## Nb_Comp_0 82.01 467.80     NA        NA    11.0000   37.01   1.000
## Nb_Comp_1 53.15  35.74 0.9236    0.9236     0.8405    8.15   2.741
##           sigmahat.dof AIC.dof BIC.dof GMDL.dof DoF.naive
## Nb_Comp_0        6.521   46.07  47.789    27.59         1
## Nb_Comp_1        1.867    4.57   4.956    21.34         2
##           sigmahat.naive AIC.naive BIC.naive GMDL.naive
## Nb_Comp_0          6.521     46.07    47.789      27.59
## Nb_Comp_1          1.891      4.17     4.459      18.38
```

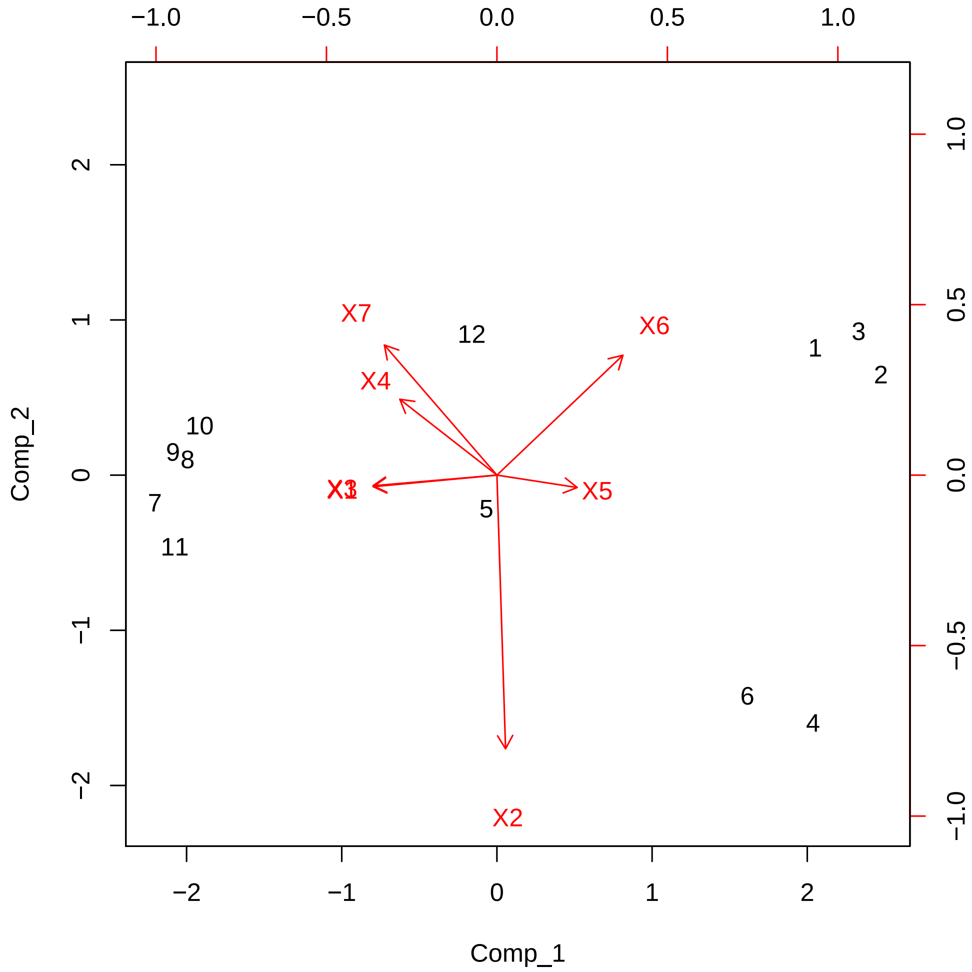

Figure 2: Biplot of the observations and the variables

It is also possible to obtain the matrix $\mathbf{W}^*$ with the following command line:

```
res$wwetoile
```

```
##    Coord_Comp_1
## X1     -0.43700
## X2     -0.03696
## X3     -0.43734
## X4     -0.36884
## X5      0.25772
## X6      0.51412
## X7     -0.38680
```

It is also possible to display the biplot of the observations and the predictors (Figure 2).

```
biplot(res6$tt,res6$pp)
```

Hard thresholding PLS regression and automatic selection of the number of components (Bastien et al. (2005)) is also available:

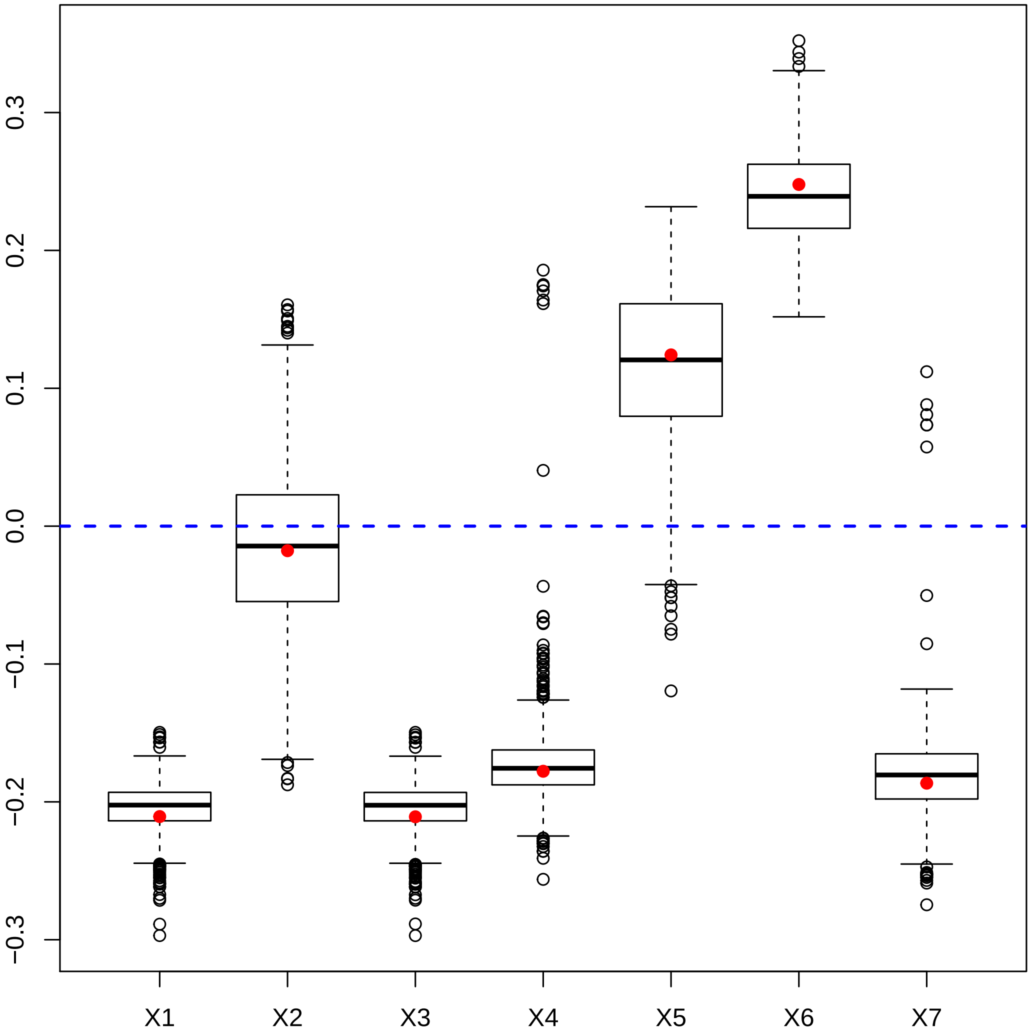

Figure 3: Bootstrap $(\mathbf{y}, \mathbf{X})$ distribution of the coefficients of the predictors

```
modpls2 <- plsR(Y~.,data=Cornell,6,sparse=TRUE)
modpls3 <- plsR(Y~.,data=Cornell,6,sparse=TRUE,sparseStop=FALSE)
```

Bootstrap $(\mathbf{y}, \mathbf{X})$

Graphical results of the bootstrap on the (Y, X): distributions of the estimators (see Figure 3) and CI (see Figure 4).

```
set.seed(123)
Cornell.bootYX1=bootpls(res,R=1000)
```

We do not bootstrap the intercept since the boostrap is done with the centered and scaled response and predictors. As a consequence we should exclude it from the boxplots using the option `indice=2:8` and must exclude it from the CI computations, if we request BC$_a$ ones, again with the option `indice=2:8`.

```
boxplots.bootpls(Cornell.bootYX1,indice=2:8)
```

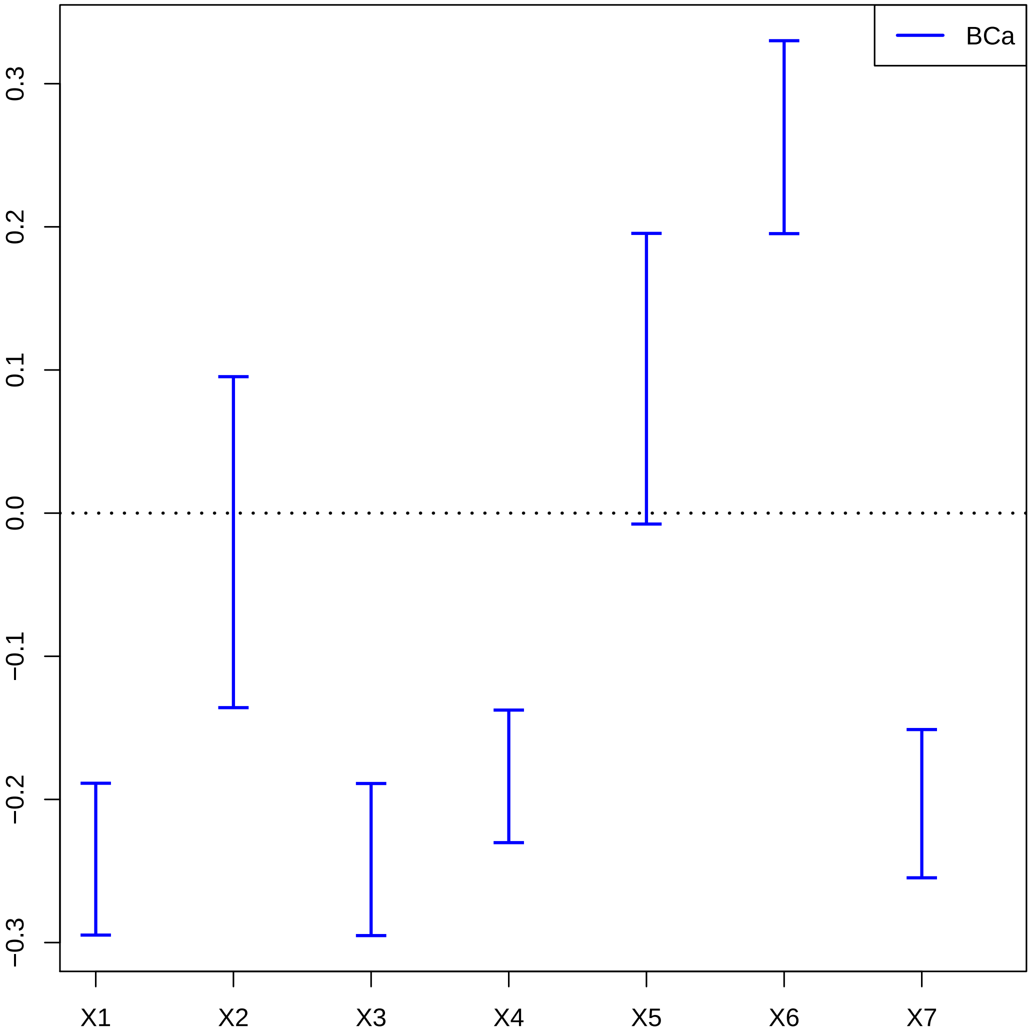

Figure 4: CI of the coefficients of the predictors, bootstrap $(\mathbf{y}, \mathbf{X})$, R=1000

```
temp.ci=confints.bootpls(Cornell.bootYX1,indice=2:8)
plots.confints.bootpls(temp.ci,typeIC="BCa",colIC=c("blue","blue","blue","blue"),
                       legendpos ="topright")
```

Bootstrap is perfomed using the *boot* package. It allows the user to apply the functions, including `jack.after.boot` or `plot.boot` (Figure 4), of this package to the bootstrapped PLSR or PLSGLR models.

```
plot(Cornell.bootYX1,index=2,jack=TRUE)
```

Using the `dataEllipse` of the *car* you can plot confidence ellipses for two parameters of the PLSR or PLSGLR models (Figure 6).

```
car::dataEllipse(Cornell.bootYX1$t[,2], Cornell.bootYX1$t[,3], cex=.3, levels=c(.5, .95, .99),
                 robust=T, xlab="X2", ylab="X3")
```

Bootstrap $(\mathbf{y}, \mathbf{T})$

Re-sampling on the couple (Y, T) (Bastien et al., 2005) is more stable and faster than the first one. We set at 1000 the number of re-sampling. CIs for each of the predictors (see Figure 8) and boxplots as well (see Figure 7).

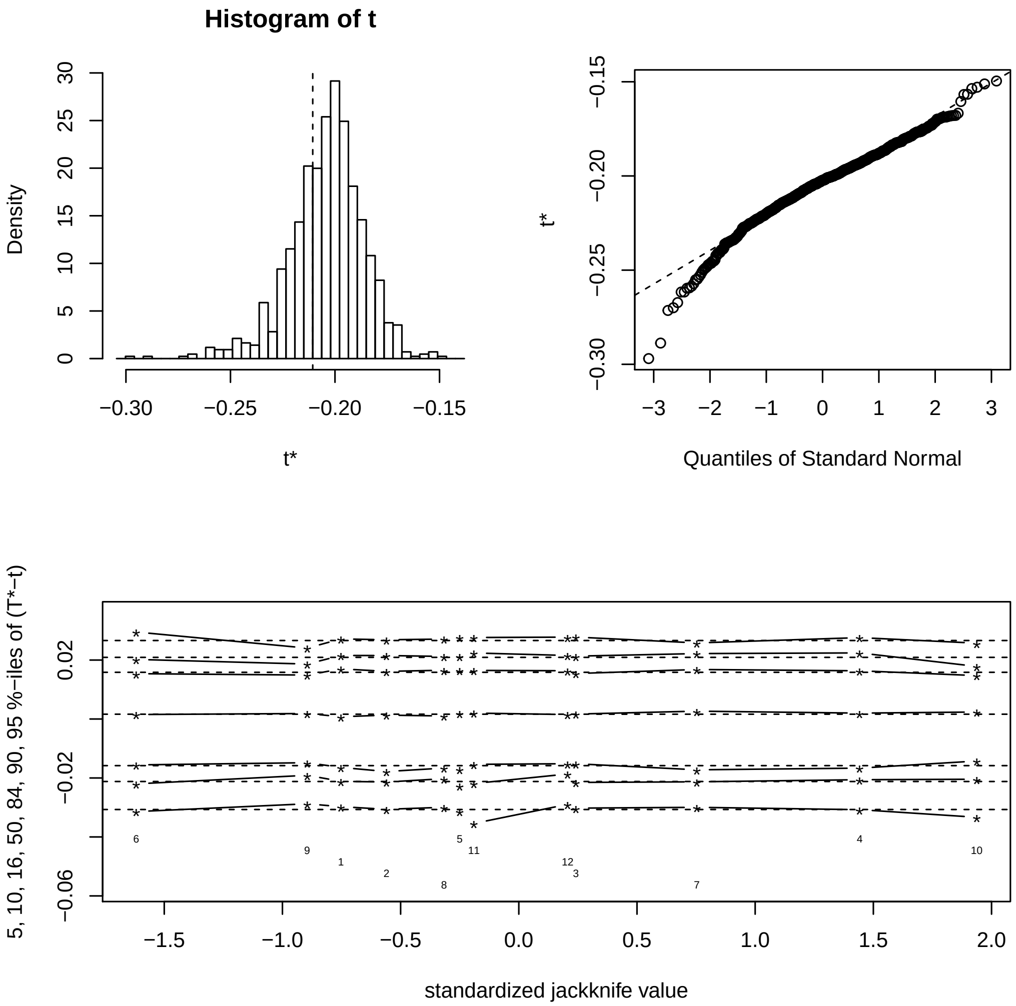

Figure 5: plot.boot, bootstrap $(\mathbf{y}, \mathbf{X})$, R=1000

```
set.seed(123)
Cornell.bootYT1=bootpls(res,typeboot="fmodel_np",R=1000)
```

```
boxplots.bootpls(Cornell.bootYT1,indices=2:8)
```

We do not bootstrap the intercept since the boostrap is done with the centered and scaled response and predictors. As a consequence we should exclude it from the boxplots using the option `indice=2:8` and must exclude it from the CI computations, if we request BC$_a$ ones, again with the option `indice=2:8`.

```
temp.ci=confints.bootpls(Cornell.bootYT1,indices=2:8)
plots.confints.bootpls(temp.ci,typeIC="BCa",colIC=c("blue","blue","blue","blue"),
                       legendpos ="topright")
```

Since after cross validation we an empirical distribution of the retained number of components, it makes sense to perform ($mathbfy,$ $mathbfT)$ bootstrap for any of these numbers of components and compare the resulting significance of the predictors at a 5% level. The `signpred` function can be used to plot a summary of this selection (Figure 9).

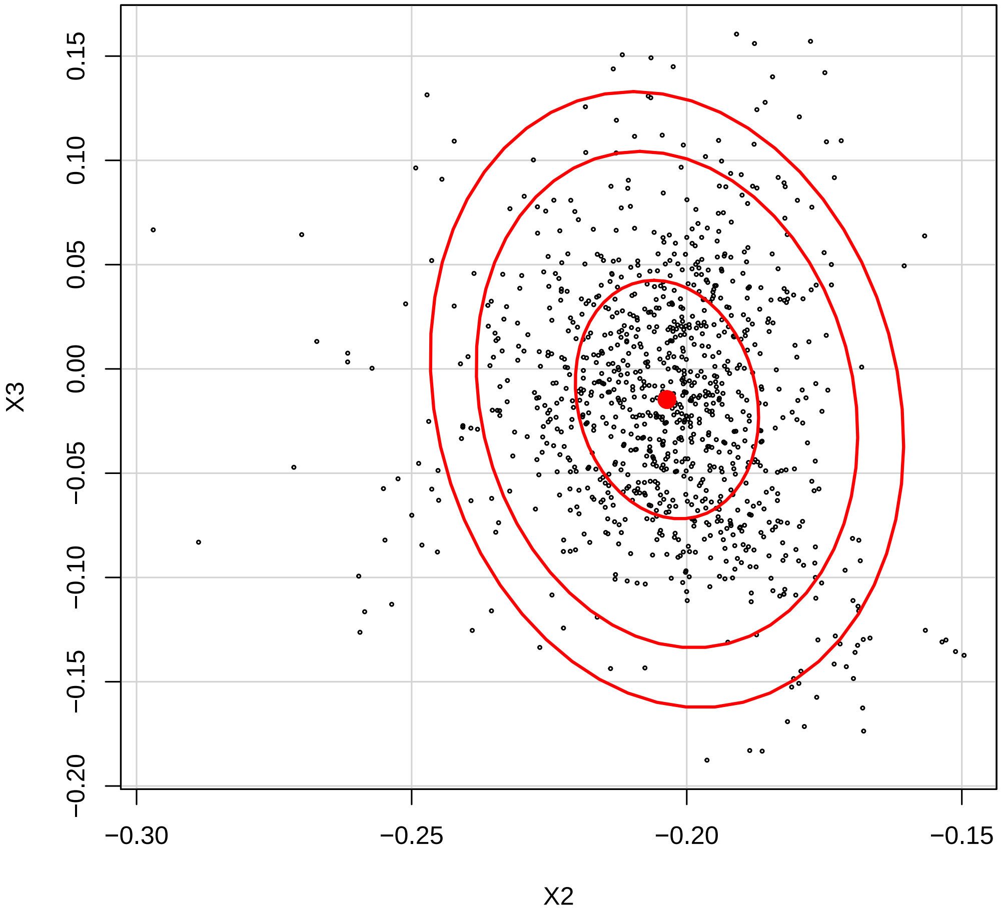

Figure 6: Confidence ellipse of the coefficients of the first two predictors, bootstrap $(\mathbf{y}, \mathbf{X})$, R=1000

In addition, one can compute an empirical measure of significance π_e by computing the weighted -with respect to the empirical distribution of components- average of the significance indicatrices. In that case, all the predictors are significant for the 1 and 2 components model and hence the empirical mesure of significance is equal to 1 for all of them.

```
res2<-plsR(Y~.,data=Cornell,nt=2)
Cornell.bootYT2=bootpls(res2,typeboot="fmodel_np",R=1000)
temp.ci2<-confints.bootpls(Cornell.bootYT2,indices=2:8)

ind.BCa.CornellYT1 <- (temp.ci[,7]<0&temp.ci[,8]<0)|(temp.ci[,7]>0&temp.ci[,8]>0)
ind.BCa.CornellYT2 <- (temp.ci2[,7]<0&temp.ci2[,8]<0)|(temp.ci2[,7]>0&temp.ci2[,8]>0)
```

```
(matind=(rbind(YT1=ind.BCa.CornellYT1,YT2=ind.BCa.CornellYT2)))

##       X1   X2   X3   X4    X5   X6   X7
## YT1 TRUE TRUE TRUE TRUE  TRUE TRUE TRUE
## YT2 TRUE TRUE TRUE TRUE FALSE TRUE TRUE

pi.e=prop.table(res.cv.modpls$CVQ2)[-1]%*%matind
pi.e
```

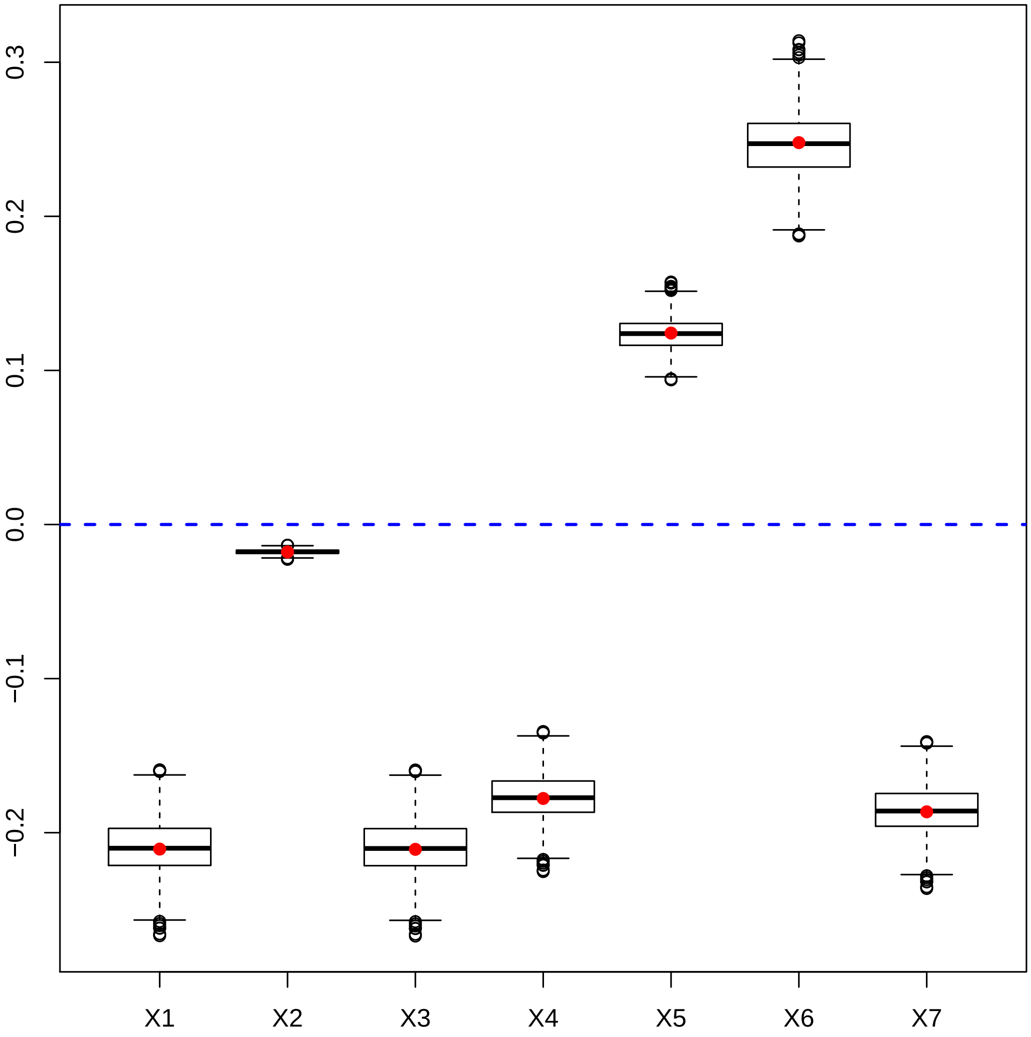

Figure 7: Bootstrap $(\mathbf{y}, \mathbf{T})$ distribution of the coefficients of the predictors

```
##      X1 X2 X3 X4   X5 X6 X7
## [1,]  1  1  1  1 0.89  1  1
signpred(t(matind),labsize=.5, plotsize = 12)
text(1:(ncol(matind))-.5,-.5,pi.e,cex=1.4)
mtext(expression(pi[e]),side=2,las=1,line=2,at=-.5,cex=1.4)
```

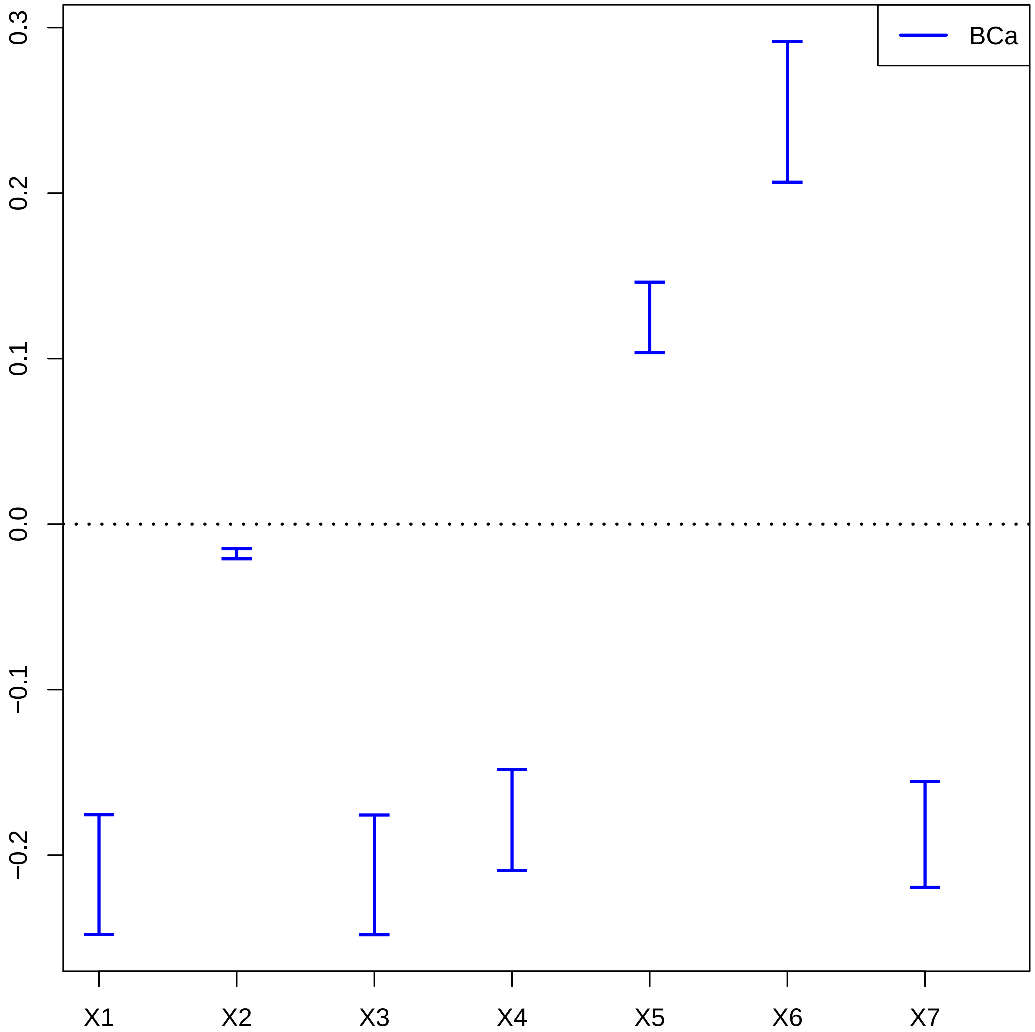

Figure 8: CI of the coefficients of the predictors, bootstrap $(\mathbf{y}, \mathbf{T})$, R=1000

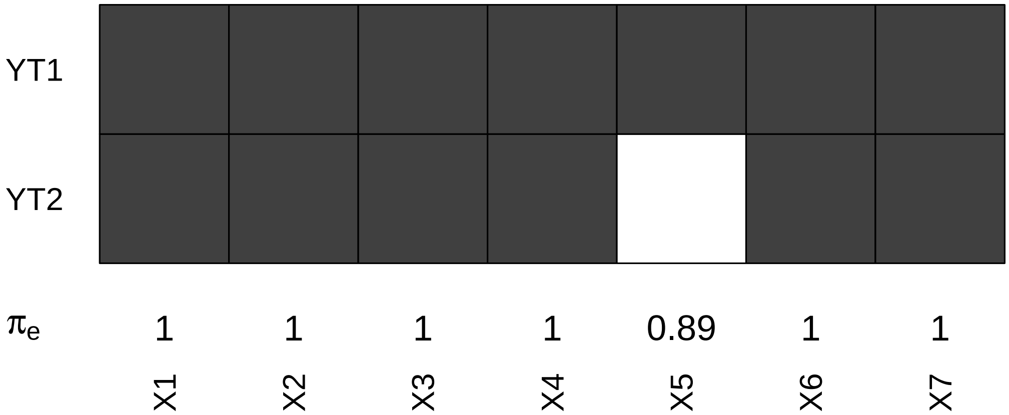

Figure 9: Significance of the predictors vs nbr of components, bootstrap $(\mathbf{y}, \mathbf{T})$, R=1000

3.3 PLS binary logistic regression: Microsatellites

In this section, we are going to deal with the allelotyping study dataset. This database, which is made with data collected on patients carrying a colon adenocarcinoma, has 104 observations on 33 binary qualitative explanatory variables representing the cancer stage according to the to Astler-Coller classification (Astler and Coller, 1954). This dataset, called aze, has some missing data due to technical limits. Finally, the variable $aze[,1]$ represent the Astler-Coller score and so will be our dependant variable in this analysis.

We first use the original one with missing values then an imputed dataset. We show some discrepancy in the estimates of the coefficients as well as in the significance testing procedure.

3.3.1 Method and Results: original dataset

Cross-validation

We decided to apply a binary logistic PLS regression with a $logit$ link-function on these data.

```
rm(list = ls())
library(plsRglm)
data(aze)
Xaze<-aze[,2:34]
yaze<-aze$y
```

First of all, a repeated k-fold cross validation is necessary to obtain the number of components to build. We decided to use $k = 8$ balanced groups of 13 subjects. We then chose to set to 10, thanks to the option `nt=10`, the maximal number of components for the cross-validation that the `cv.plsRglm` function would try to compute. According to field experts, this number of components should be greater to the real number of components featured in the dataset. The cross-validation step is performed by running the following command line.

```
cv.modpls<-cv.plsRglm(dataY=yaze,dataX=Xaze,nt=10,modele="pls-glm-logistic",K=8)
```

For PLS-GLR models, the cross-validation results can be summed up in a single table using the `summary`[1]. Results are obtained by the following command line.

```
res.cv.modpls=cvtable(summary(cv.modpls, MClassed = TRUE))
## ____************************************************____
## Only naive DoF can be used with missing data
##
## Family: binomial
## Link function: logit
##
## ____There are some NAs in X but not in Y____
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Component____ 5 ____
## ____Component____ 6 ____
## Warning:  glm.fit:  fitted probabilities numerically 0 or 1 occurred
## ____Component____ 7 ____
## Warning:  glm.fit:  fitted probabilities numerically 0 or 1 occurred
## ____Component____ 8 ____
## Warning:  glm.fit:  fitted probabilities numerically 0 or 1 occurred
## Warning:  glm.fit:  fitted probabilities numerically 0 or 1 occurred
## ____Component____ 9 ____
## Warning:  glm.fit:  algorithm did not converge
## Warning:  glm.fit:  fitted probabilities numerically 0 or 1 occurred
## Warning:  glm.fit:  algorithm did not converge
## Warning:  glm.fit:  fitted probabilities numerically 0 or 1 occurred
```

[1] for PLSR models the cross-validation results can be summed up in a single table using the functionsummary.

```
## ____Component____ 10 ____
## ____Predicting X with NA in X and not in Y____
## ****________________________________________________****
##
##
## NK: 1
## CV MissClassed criterion:
## 1 2 3 4 5
## 0 0 0 0 1
##
## CV Q2Chi2 criterion:
## 0
## 1
##
## CV PreChi2 criterion:
## 1
## 1
```

The number of significant predictors per components can be obtained via the following code:

```
res10<-plsRglm(yaze, Xaze, nt=10, modele="pls-glm-logistic", pvals.expli=TRUE)
## ____************************************************____
## Only naive DoF can be used with missing data
##
## Family: binomial
## Link function: logit
##
## ____There are some NAs in X but not in Y____
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Component____ 5 ____
## ____Component____ 6 ____
## Warning:  glm.fit:  fitted probabilities numerically 0 or 1 occurred
## ____Component____ 7 ____
## Warning:  glm.fit:  fitted probabilities numerically 0 or 1 occurred
## ____Component____ 8 ____
## Warning:  glm.fit:  fitted probabilities numerically 0 or 1 occurred
## Warning:  glm.fit:  fitted probabilities numerically 0 or 1 occurred
## ____Component____ 9 ____
## Warning:  glm.fit:  algorithm did not converge
## Warning:  glm.fit:  fitted probabilities numerically 0 or 1 occurred
## Warning:  glm.fit:  algorithm did not converge
## Warning:  glm.fit:  fitted probabilities numerically 0 or 1 occurred
## ____Component____ 10 ____
## ____Predicting X with NA in X and not in Y____
## ****________________________________________________****
colSums(res10$pvalstep)
## temppvalstep temppvalstep temppvalstep temppvalstep temppvalstep
##            1            3            0            0            0
## temppvalstep temppvalstep temppvalstep temppvalstep temppvalstep
##            0            0            0            0            0
```

The number of significant predictors within each component, which is a criteria of significance for Bastien et al. (2005), is implemented in the package with the options `sparse=TRUE` and `sparseStop=TRUE`.

```
modpls2 <- plsRglm(dataY=yaze,dataX=Xaze, nt = 10, modele = "pls-glm-logistic",sparse=TRUE,
                   sparseStop=TRUE)

## ____************************************************____
## Only naive DoF can be used with missing data
## sparse option cannot be used with missing data
##
## Family: binomial
## Link function: logit
##
## ____There are some NAs in X but not in Y____
## ____Component____ 1 ____
## ____Component____ 2 ____
## No more significant predictors (<0.05) found
## Warning only 2 components were thus extracted
## ____Predicting X with NA in X and not in Y____
## ****________________________________________________****
```

The number of significant predictors within each component tells us to only build 2 components while the AIC criteria gives us 6 components and the BIC concludes to 4 components. But for this study, the most important criteria was to minimize the miss classification rate after cross-validation, criteria which let us know to build 4 components, in agreement with BIC criteria. In order to confirm the choice of retaining 4 components, the cross-validation was run 100 times by randomly creating groups. Here are the command lines:

```
set.seed(123)
cv.modpls.logit<-cv.plsRglm(dataY=yaze,dataX=Xaze,nt=10,modele="pls-glm-logistic",K=8,NK=100)
```

```
res.cv.modpls.logit=cvtable(summary(cv.modpls.logit, MClassed = TRUE))

## ____************************************************____
## Only naive DoF can be used with missing data
##
## Family: binomial
## Link function: logit
##
## ____There are some NAs in X but not in Y____
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Component____ 5 ____
## ____Component____ 6 ____

## Warning:  glm.fit:  fitted probabilities numerically 0 or 1 occurred

## ____Component____ 7 ____

## Warning:  glm.fit:  fitted probabilities numerically 0 or 1 occurred

## ____Component____ 8 ____

## Warning:  glm.fit:  fitted probabilities numerically 0 or 1 occurred
## Warning:  glm.fit:  fitted probabilities numerically 0 or 1 occurred

## ____Component____ 9 ____

## Warning:  glm.fit:  algorithm did not converge
## Warning:  glm.fit:  fitted probabilities numerically 0 or 1 occurred
## Warning:  glm.fit:  algorithm did not converge
## Warning:  glm.fit:  fitted probabilities numerically 0 or 1 occurred

## ____Component____ 10 ____
## ____Predicting X with NA in X and not in Y____
## ****________________________________________________****
##
```

```
##
## NK: 1,  2,  3,  4,  5,  6,  7,  8,  9,  10
## NK: 11,  12,  13,  14,  15,  16,  17,  18,  19,  20
## NK: 21,  22,  23,  24,  25,  26,  27,  28,  29,  30
## NK: 31,  32,  33,  34,  35,  36,  37,  38,  39,  40
## NK: 41,  42,  43,  44,  45,  46,  47,  48,  49,  50
## NK: 51,  52,  53,  54,  55,  56,  57,  58,  59,  60
## NK: 61,  62,  63,  64,  65,  66,  67,  68,  69,  70
## NK: 71,  72,  73,  74,  75,  76,  77,  78,  79,  80
## NK: 81,  82,  83,  84,  85,  86,  87,  88,  89,  90
## NK: 91,  92,  93,  94,  95,  96,  97,  98,  99,  100
## CV MissClassed criterion:
##  1  2  3  4  5  6  7  8
##  1  0 35 38 14  6  4  2
##
## CV Q2Chi2 criterion:
##   0
## 100
##
## CV PreChi2 criterion:
##  1  2
## 94  6
```

The results (Fig. 10) confirm the results obtained during the original cross-validation and that's the reason why we decided to build 4 components. Here is also the model we will work with:

$$\mathbb{P}\left(y=1\right)=\frac{\exp\left(\mu+\sum_{h=1}^{4}c_h t_h\right)}{1+\exp\left(\mu+\sum_{h=1}^{4}c_h t_h\right)} \tag{6}$$

where t_h is the h^{th} component, c_h the coefficients of the logistic regression of the response variable y on the components t_h and μ the intercept.

```
plot(res.cv.modpls.logit)
```

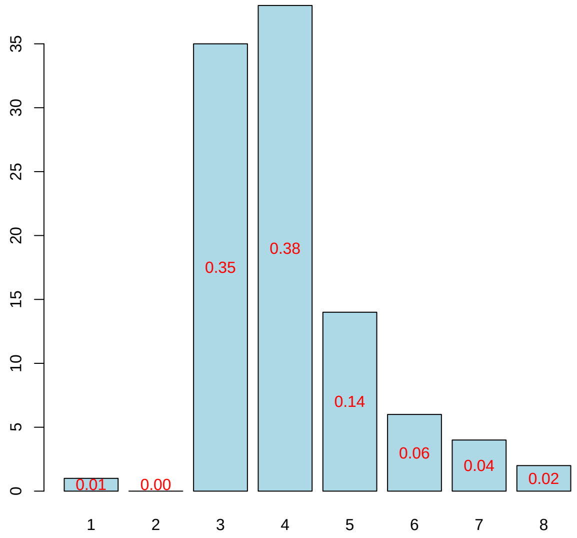

Figure 10: Nb components, 8-CV, n=100

Now, the PLSGLR regression is done in order to obtain these coefficients c_h and the intercept.

```
res<-plsRglm(yaze, Xaze, nt = 4, modele = "pls-glm-logistic", pvals.expli=TRUE)
## ____************************************************____
## Only naive DoF can be used with missing data
##
## Family: binomial
## Link function: logit
##
## ____There are some NAs in X but not in Y____
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Predicting X with NA in X and not in Y____
## ****________________________________________________****
res
## Number of required components:
## [1] 4
## Number of successfully computed components:
## [1] 4
## Coefficients:
##               [,1]
## Intercept -3.29706
## D2S138    -0.84808
## D18S61     2.66109
## D16S422   -0.75013
## D17S794    1.19845
## D6S264    -0.84513
## D14S65     0.42024
## D18S53    -0.19435
## D17S790   -0.94635
## D1S225    -0.18233
## D3S1282    0.30162
## D9S179     0.64299
## D5S430    -2.08075
## D8S283     0.08824
## D11S916    1.15044
## D2S159     0.60088
## D16S408    0.81723
## D5S346     1.82473
## D10S191   -0.50814
## D13S173    1.27486
## D6S275    -1.35727
## D15S127   -0.48758
## D1S305     1.56507
## D4S394    -1.10655
## D20S107   -0.88827
## D1S197    -0.59979
## D1S207     0.23937
## D10S192    1.18776
## D3S1283   -0.05031
## D4S414     0.88055
## D8S264     0.90852
## D22S928   -0.20802
## TP53      -1.36478
## D9S171     0.74161
## Information criteria and Fit statistics:
```

```
##              AIC   BIC Missclassed Chi2_Pearson_Y RSS_Y   R2_Y R2_residY
## Nb_Comp_0 145.8 148.5          49          104.0 25.91     NA        NA
## Nb_Comp_1 119.1 124.3          30          101.7 19.54 0.2461    -6.017
## Nb_Comp_2 106.0 113.9          20          111.0 16.17 0.3761   -11.620
## Nb_Comp_3 100.3 110.9          18          102.5 14.85 0.4269   -14.115
## Nb_Comp_4  96.2 109.4          20          122.8 13.74 0.4699   -19.633
##           RSS_residY
## Nb_Comp_0      25.91
## Nb_Comp_1     181.84
## Nb_Comp_2     327.02
## Nb_Comp_3     391.69
## Nb_Comp_4     534.68
## Model with all the required components:
##
## Call:  glm(formula = YwotNA ~ ., family = family, data = tttrain)
##
## Coefficients:
## (Intercept)         tt.1         tt.2         tt.3         tt.4
##      -0.297        1.427        0.510        0.690        0.793
##
## Degrees of Freedom: 103 Total (i.e. Null);  99 Residual
## Null Deviance:     144
## Residual Deviance: 86.2  AIC: 96.2
```

It is also possible to obtain the matrix $\mathbf{W}^*$ with the following command line:

```
res$wwetoile
```

```
##         Coord_Comp_1 Coord_Comp_2 Coord_Comp_3 Coord_Comp_4
## D2S138     -0.041632    -0.206274    -0.066401    -0.267206
## D18S61      0.325627     0.317242     0.318361     0.318081
## D16S422     0.040763    -0.131575    -0.256933    -0.231912
## D17S794     0.129158     0.192599     0.340246     0.102509
## D6S264     -0.103643    -0.214465    -0.135336    -0.088290
## D14S65      0.102355     0.078851     0.024787     0.006052
## D18S53      0.014590    -0.250126    -0.171388     0.164904
## D17S790    -0.168555    -0.261288     0.041658    -0.159789
## D1S225      0.034272    -0.093162     0.009514    -0.123996
## D3S1282    -0.071682    -0.164201     0.241973     0.215595
## D9S179      0.136638     0.100540     0.187466    -0.078107
## D5S430     -0.151642    -0.534737    -0.414466    -0.338288
## D8S283      0.191726    -0.027383    -0.142431    -0.148326
## D11S916     0.326263     0.162340     0.197944    -0.145698
## D2S159     -0.003301    -0.081425     0.305683     0.164072
## D16S408     0.099858     0.086526     0.103035     0.173948
## D5S346      0.400051     0.227688    -0.105805     0.370344
## D10S191     0.018664    -0.212931    -0.025305    -0.181209
## D13S173     0.221857     0.267187     0.201149     0.047438
## D6S275     -0.228091    -0.432388    -0.146102     0.011989
## D15S127     0.040485    -0.103365    -0.061795    -0.262021
## D1S305      0.253458     0.141652     0.293982     0.186037
## D4S394     -0.076095    -0.332878    -0.087742    -0.232867
## D20S107    -0.228181    -0.391943     0.059582     0.065595
## D1S197     -0.008654    -0.292466    -0.266137     0.057814
## D1S207      0.048839    -0.106744    -0.045262     0.165028
## D10S192     0.265045     0.218465     0.196497    -0.037039
## D3S1283    -0.130765    -0.335018     0.215132     0.231507
## D4S414      0.070732     0.007376     0.183889     0.242836
## D8S264      0.336656     0.120215    -0.175169    -0.003774
```

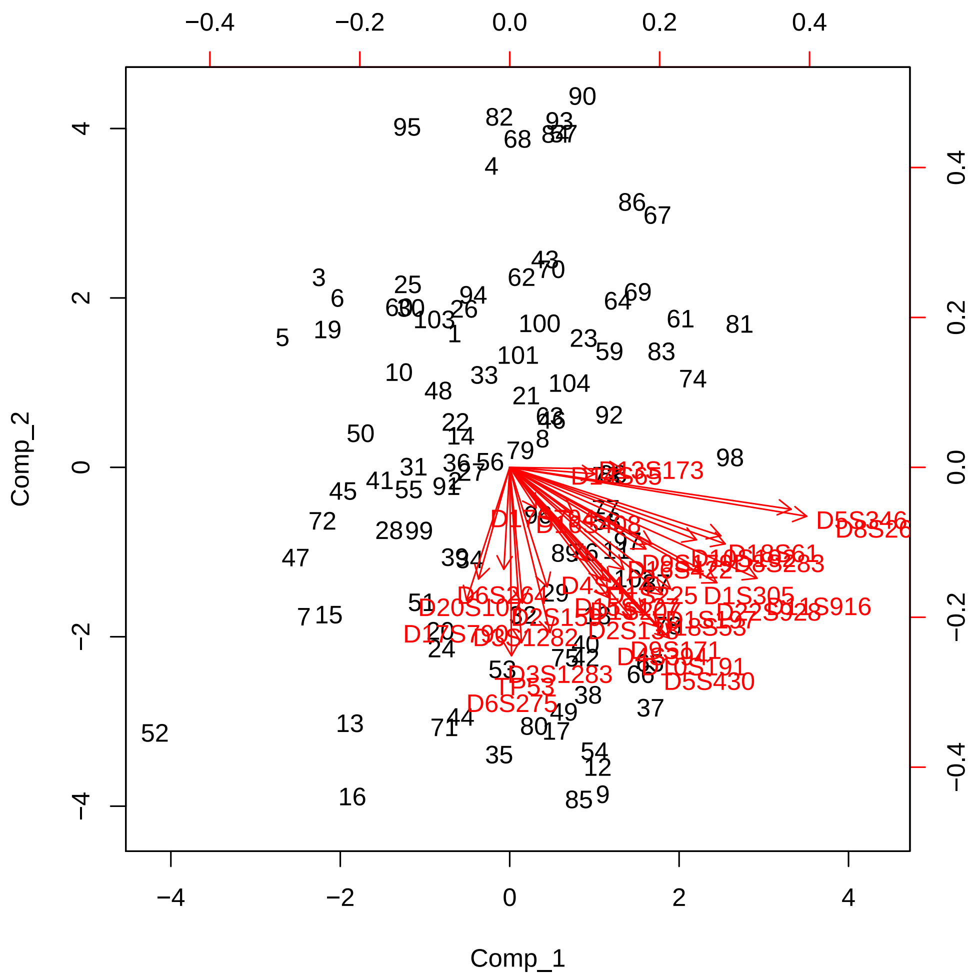

Figure 11: Biplot of the observations and the variables

```
## D22S928      0.151029     -0.030607     -0.120889     -0.278420
## TP53        -0.297981     -0.601905     -0.147448      0.293278
## D9S171       0.072332     -0.083747      0.220187      0.202930
```

It is also possible to display the biplot of the observations and the predictors (Figure 11).

```
biplot(res$tt,res$pp)
```

Then, in order to have results which are interpretable in practice, let run the following command line and so obtain the coefficients β_j of the predictors $\mathbf{x}_j$, $1 \leqslant j \leqslant 33$ of the final model.

```
res$Std.Coeffs
```

```
##                [,1]
## Intercept -0.29676
## D2S138    -0.42228
## D18S61     1.09847
## D16S422   -0.37014
## D17S794    0.59868
## D6S264    -0.42067
## D14S65      0.20819
## D18S53     -0.09420
## D17S790    -0.47171
## D1S225     -0.09032
## D3S1282     0.15200
## D9S179      0.31375
## D5S430     -1.04334
## D8S283      0.04376
## D11S916     0.56954
## D2S159      0.29492
## D16S408     0.39570
## D5S346      0.90770
## D10S191    -0.24304
## D13S173     0.62932
## D6S275     -0.63728
## D15S127    -0.24533
## D1S305      0.78444
## D4S394     -0.52349
## D20S107    -0.43230
## D1S197     -0.29928
## D1S207      0.11493
## D10S192     0.59593
## D3S1283    -0.02530
## D4S414      0.42423
## D8S264      0.41788
## D22S928    -0.10426
## TP53       -0.60130
## D9S171      0.37348
```

The options sparse and sparseStop allows enabling, or not, separately hard thresholding PLS regression or automatic selection of the number of components (Bastien et al. (2005)).

```
modpls3 <- plsRglm(dataY=yaze,dataX=Xaze, nt = 10, modele = "pls-glm-logistic",sparse=FALSE,
                   sparseStop=TRUE)
modpls4 <- plsRglm(dataY=yaze,dataX=Xaze, nt = 10, modele = "pls-glm-logistic",sparse=TRUE,
                   sparseStop=FALSE)
```

Bootstrap $(\mathbf{y}, \mathbf{X})$

However, what is really important is to know which of them are significantly different from zero. We can also answer to this question with the bootstrap techniques we insert in this package.

Let us begin with the bootstrap on the (Y, X). This method, which seems to be natural, has some trouble in this case. Indeed, with the help of the boxplot, we decided to only focus on the BC_a CI, because of the fact of the clearly non symmetrical distributions of the estimators (see Figure 22). And when we choose 3 components, some of the CI become disproportionate (see Figure 23), depriving us of any graphical interpretation. So, the only way to see if any predictors is significantly different from 0, is to use the function confints.bootpls, which will allow to see the values of the CI for the four different type of CI. Thanks to this function, we can see that only one predictor is significantly different from 0, namely D18S61.

```
set.seed(123)
aze.bootYX4=bootplsglm(res,typeboot="plsmodel",R=1000)
```

By default with PLSGLR models the option typeboot is set to typeboot="fmodel_np" -Bootstrap $(\mathbf{y}, \mathbf{T})$-, we change this setting using the option typeboot="plsmodel" -Bootstrap $(\mathbf{y}, \mathbf{X})$-.

Since the object res contained the results of the function plsRglm(yaze,Xaze,nt=4,modele= "pls-glm-logistic"), the statement aze.bootYX4=bootplsglm(res,typeboot="plsmodel",R=1000) is equivalent to the instruction aze.bootYX4=bootplsglm(plsRglm(yaze,Xaze,nt=4,modele="pls-glm-logistic"), typeboot="plsmodel", R=1000).

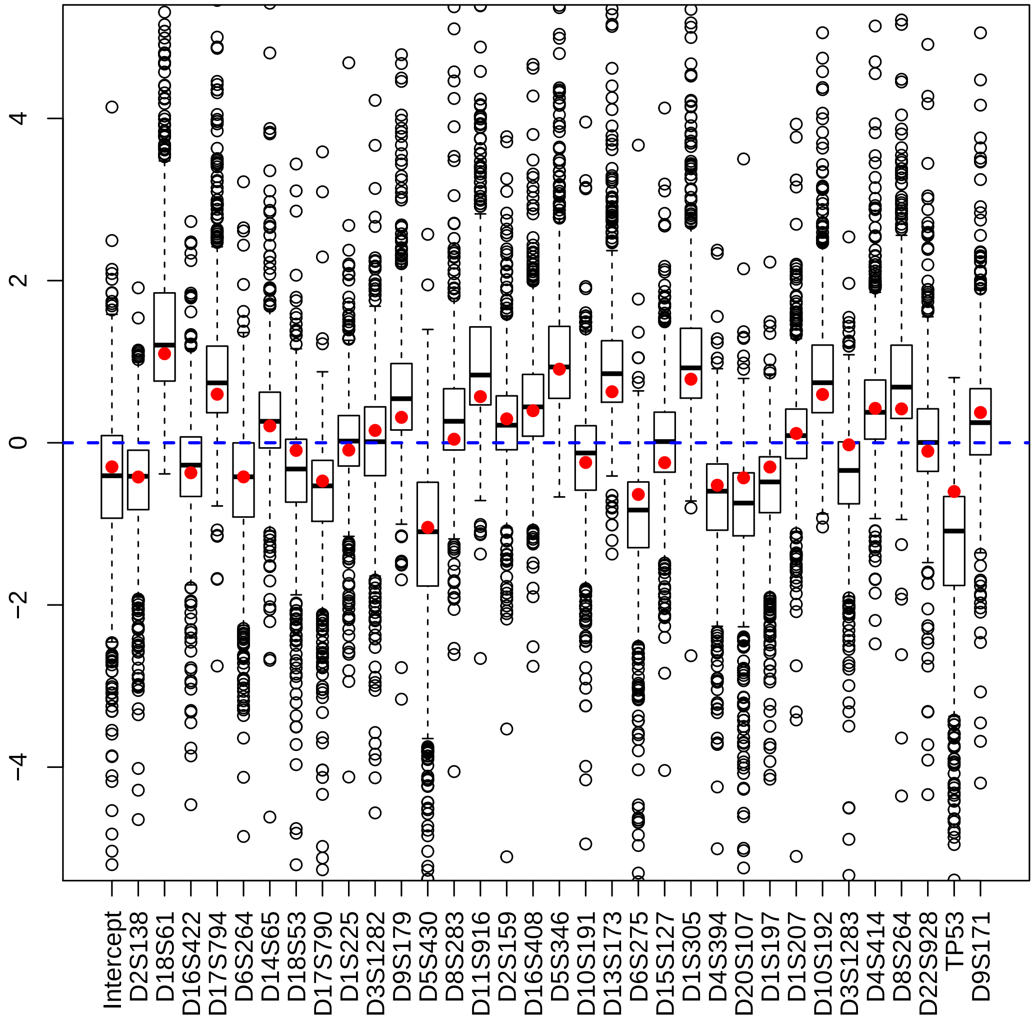

Figure 12: Bootstrap $(\mathbf{y}, \mathbf{X})$ distribution of the coefficients of the predictors, R=1000

```
boxplots.bootpls(aze.bootYX4,las=2,mar=c(5,2,1,1)+0.1)
```

```
temp.ci=confints.bootpls(aze.bootYX4)
## Warning:  extreme order statistics used as endpoints
## Warning:  extreme order statistics used as endpoints
plots.confints.bootpls(temp.ci,typeIC="BCa",colIC=c("blue","blue","blue","blue"),
                       legendpos ="topright",las=2,mar=c(5,2,1,1)+0.1)
```

Bootstrap $(\mathbf{y}, \mathbf{T})$

However, due to the problems of the previous results, we decided to choose the second type of bootstrap, that is to say the one which do re-sampling on the couple (Y, T) (Bastien et al., 2005). Indeed, it is more stable and faster than the first one. We set at 1000 the number of re-sampling. So we obtain a graphic representing the confidence intervals (CI) for each of the predictors (see Figure 14) and a boxplot as well (see Figure 14). These graphics were obtained by starting these command lines:

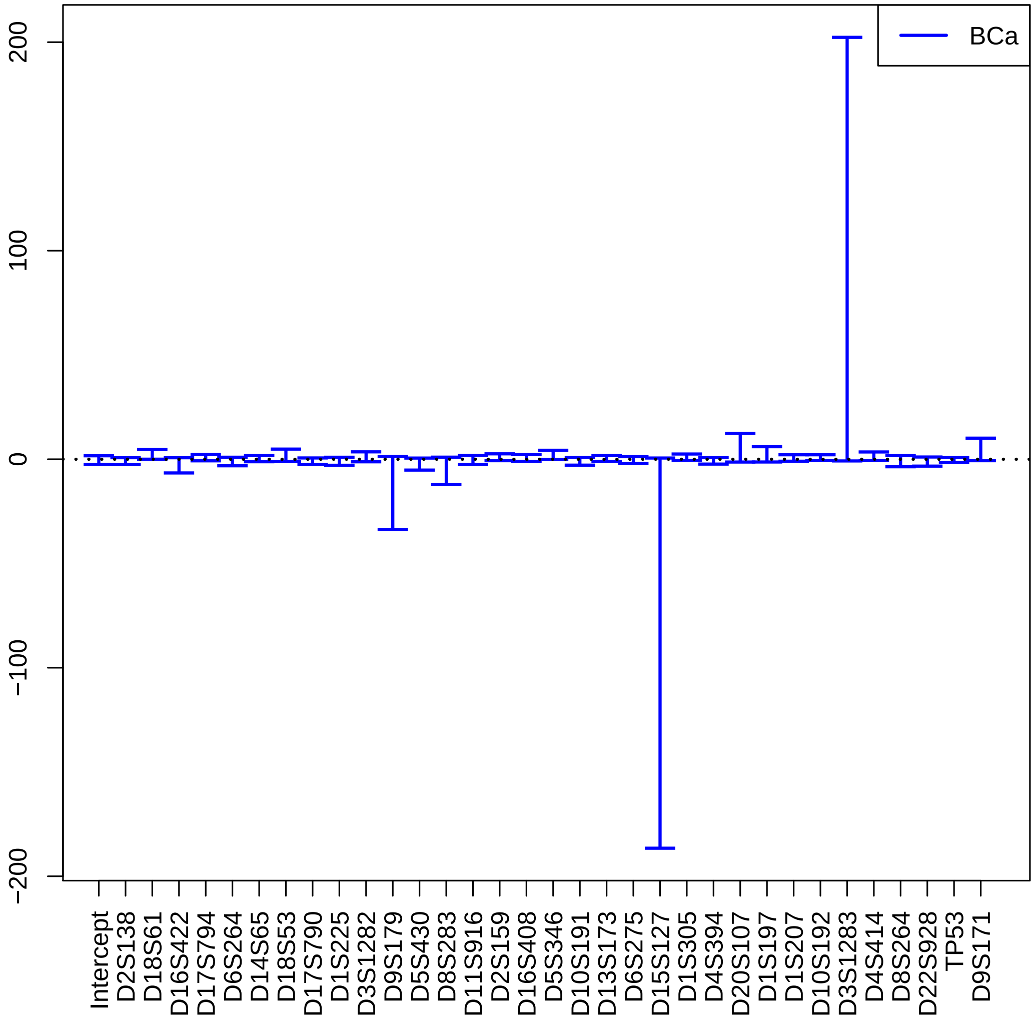

Figure 13: CI of the coefficients of the predictors, bootstrap $(\mathbf{y}, \mathbf{X})$, R=1000

```
set.seed(123)
aze.bootYT4=bootplsglm(res,R=1000)
```

By default with PLSGLR models the option `typeboot` is set to `typeboot="fmodel_np"` -Bootstrap $(\mathbf{y}, \mathbf{T})$-.

Since `res=plsRglm(yaze,Xaze,nt=4,modele="pls-glm-logistic")`, the statement `aze.bootYX4= bootplsglm(res,R=1000)` is equivalent to the instruction `aze.bootYX4=bootplsglm(plsRglm(yaze,Xaze,nt=4, modele="pls-glm-logistic"),R=1000)`.

```
boxplots.bootpls(aze.bootYT4,las=2,mar=c(5,2,1,1)+0.1)
```

```
temp.ci4=confints.bootpls(aze.bootYT4)
plots.confints.bootpls(temp.ci4,typeIC="BCa",colIC=c("blue","blue","blue","blue"),
                       legendpos ="topright",las=2,mar=c(5,2,1,1)+0.1)
```

Remark 6. In this paper, we decided to only focus on the *BCa* CI. But, with this package, it is naturally possible to obtain CI with percentile, normal or basic bootstrap as well.

With the help of Figure 15, we can see that only 9 predictors are not significantly different from 0. But, it could be interesting to display, through the model with 1 to 8 components, which of the predictors are significantly different

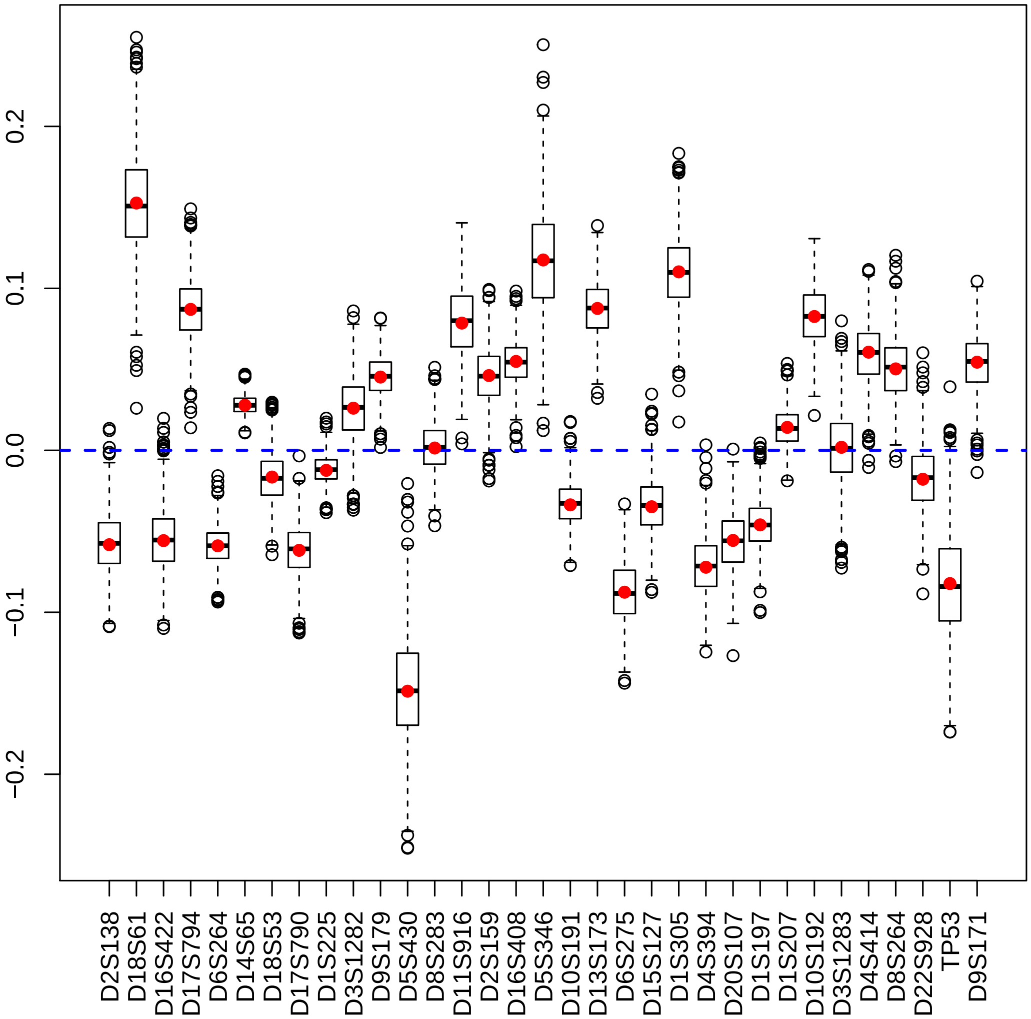

Figure 14: Bootstrap $(\mathbf{y}, \mathbf{T})$ distribution of the coefficients of the predictors, R=1000

from zero so that we could know if there is a stability of significant predictors or not (see Figure 16). A function is available in our package, called `signpred`, to do this kind of graphic.

Since after cross validation we an empirical distribution of the retained number of components, it makes sense to perform $(\mathbf{y}, \mathbf{T})$ bootstrap for any of these numbers of components and compare the resulting significance of the predictors at a 5% level. The `signpred` function can be used to plot a summary of this selection (Figure 16).

As we can see on the figure 16, there is few differences between the model with 3 and 4 components. Indeed, only 1 predictor, significant in the 3 components model, become non-significant in the 4 components model. Furthermore, only one predictor, non-significant in the 3 components model, become significant in the 4 components model.

In addition, one can compute an empirical measure of significance π_e by computing the weighted -with respect to the empirical distribution of components- average of the significance indicatrices. In that case, all the predictors are significant for the 1 and 2 components model and hence the empirical mesure of significance is equal to 1 for all of them.

The bootstrap technique used in this study, which is clearly faster and more stable than the other one, but the results between the two techniques are really different and so it could be interesting to confront them with the help of some simulations.

```r
res1<-plsRglm(yaze, Xaze, nt = 1, modele = "pls-glm-logistic")
res2<-plsRglm(yaze, Xaze, nt = 2, modele = "pls-glm-logistic")
res3<-plsRglm(yaze, Xaze, nt = 3, modele = "pls-glm-logistic")
```

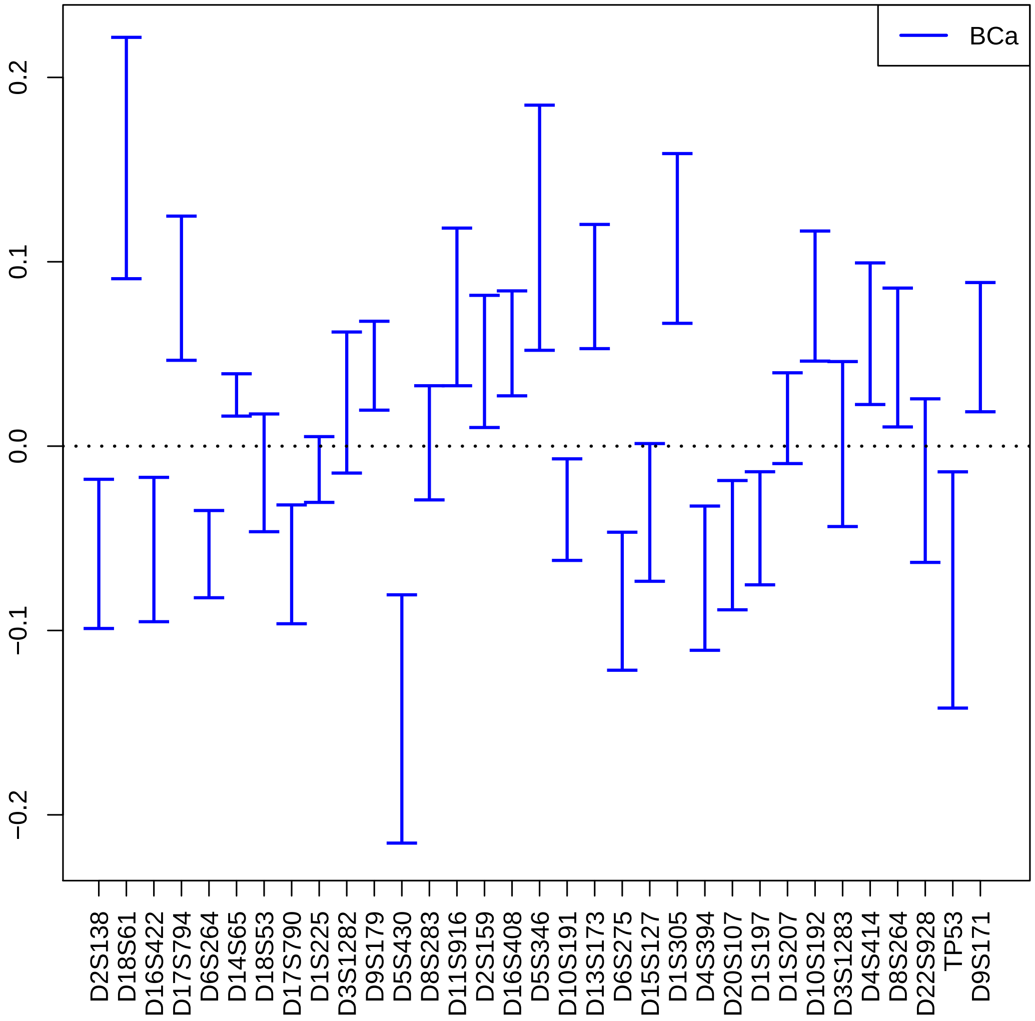

Figure 15: CI of the coefficients of the predictors, bootstrap $(\mathbf{y}, \mathbf{T})$, R=1000

```
res5<-plsRglm(yaze, Xaze, nt = 5, modele = "pls-glm-logistic")
res6<-plsRglm(yaze, Xaze, nt = 6, modele = "pls-glm-logistic")
res7<-plsRglm(yaze, Xaze, nt = 7, modele = "pls-glm-logistic")
res8<-plsRglm(yaze, Xaze, nt = 8, modele = "pls-glm-logistic")

aze.bootYT1=bootplsglm(res1,R=1000)
aze.bootYT2=bootplsglm(res2,R=1000)
aze.bootYT3=bootplsglm(res3,R=1000)
aze.bootYT5=bootplsglm(res5,R=1000)
aze.bootYT6=bootplsglm(res6,R=1000)
aze.bootYT7=bootplsglm(res7,R=1000)
aze.bootYT8=bootplsglm(res8,R=1000)

temp.ci1<-confints.bootpls(aze.bootYT1)
temp.ci2<-confints.bootpls(aze.bootYT2)
temp.ci3<-confints.bootpls(aze.bootYT3)
temp.ci5<-confints.bootpls(aze.bootYT5)
temp.ci6<-confints.bootpls(aze.bootYT6)
temp.ci7<-confints.bootpls(aze.bootYT7)
temp.ci8<-confints.bootpls(aze.bootYT8)

ind.BCa.azeYT1 <- (temp.ci1[,7]<0&temp.ci1[,8]<0)|(temp.ci1[,7]>0&temp.ci1[,8]>0)
ind.BCa.azeYT2 <- (temp.ci2[,7]<0&temp.ci2[,8]<0)|(temp.ci2[,7]>0&temp.ci2[,8]>0)
ind.BCa.azeYT3 <- (temp.ci3[,7]<0&temp.ci3[,8]<0)|(temp.ci3[,7]>0&temp.ci3[,8]>0)
ind.BCa.azeYT4 <- (temp.ci4[,7]<0&temp.ci4[,8]<0)|(temp.ci4[,7]>0&temp.ci4[,8]>0)
ind.BCa.azeYT5 <- (temp.ci5[,7]<0&temp.ci5[,8]<0)|(temp.ci5[,7]>0&temp.ci5[,8]>0)
ind.BCa.azeYT6 <- (temp.ci6[,7]<0&temp.ci6[,8]<0)|(temp.ci6[,7]>0&temp.ci6[,8]>0)
ind.BCa.azeYT7 <- (temp.ci7[,7]<0&temp.ci7[,8]<0)|(temp.ci7[,7]>0&temp.ci7[,8]>0)
ind.BCa.azeYT8 <- (temp.ci8[,7]<0&temp.ci8[,8]<0)|(temp.ci8[,7]>0&temp.ci8[,8]>0)
```

```
(matind=(rbind(YT1=ind.BCa.azeYT1,YT2=ind.BCa.azeYT2,YT3=ind.BCa.azeYT3,YT4=ind.BCa.azeYT4,
               YT5=ind.BCa.azeYT5,YT6=ind.BCa.azeYT6,YT7=ind.BCa.azeYT7,YT8=ind.BCa.azeYT8)))
```

```
##     D2S138 D18S61 D16S422 D17S794 D6S264 D14S65 D18S53 D17S790 D1S225
## YT1   TRUE   TRUE    TRUE    TRUE   TRUE   TRUE   TRUE    TRUE   TRUE
## YT2   TRUE   TRUE   FALSE    TRUE   TRUE   TRUE   TRUE    TRUE  FALSE
## YT3   TRUE   TRUE    TRUE    TRUE   TRUE   TRUE   TRUE    TRUE  FALSE
## YT4   TRUE   TRUE    TRUE    TRUE   TRUE   TRUE  FALSE    TRUE  FALSE
## YT5   TRUE   TRUE    TRUE    TRUE  FALSE   TRUE   TRUE    TRUE   TRUE
## YT6   TRUE   TRUE    TRUE    TRUE  FALSE  FALSE   TRUE    TRUE   TRUE
## YT7   TRUE   TRUE    TRUE    TRUE  FALSE  FALSE   TRUE    TRUE   TRUE
## YT8   TRUE   TRUE    TRUE    TRUE  FALSE  FALSE   TRUE    TRUE   TRUE
##     D3S1282 D9S179 D5S430 D8S283 D11S916 D2S159 D16S408 D5S346 D10S191
## YT1    TRUE   TRUE   TRUE   TRUE    TRUE   TRUE    TRUE   TRUE    TRUE
## YT2    TRUE   TRUE   TRUE   TRUE    TRUE   TRUE    TRUE   TRUE   FALSE
## YT3   FALSE   TRUE   TRUE  FALSE    TRUE  FALSE    TRUE   TRUE    TRUE
## YT4   FALSE   TRUE   TRUE  FALSE    TRUE   TRUE    TRUE   TRUE    TRUE
## YT5   FALSE   TRUE   TRUE  FALSE    TRUE   TRUE    TRUE   TRUE    TRUE
## YT6   FALSE  FALSE   TRUE  FALSE    TRUE   TRUE   FALSE   TRUE    TRUE
## YT7   FALSE   TRUE   TRUE  FALSE    TRUE   TRUE   FALSE   TRUE    TRUE
## YT8   FALSE   TRUE   TRUE  FALSE    TRUE   TRUE   FALSE   TRUE   FALSE
##     D13S173 D6S275 D15S127 D1S305 D4S394 D20S107 D1S197 D1S207 D10S192
## YT1    TRUE   TRUE    TRUE   TRUE   TRUE    TRUE   TRUE   TRUE    TRUE
## YT2    TRUE   TRUE   FALSE   TRUE   TRUE    TRUE   TRUE  FALSE    TRUE
## YT3    TRUE   TRUE   FALSE   TRUE   TRUE    TRUE   TRUE  FALSE    TRUE
## YT4    TRUE   TRUE   FALSE   TRUE   TRUE    TRUE   TRUE  FALSE    TRUE
## YT5    TRUE   TRUE    TRUE   TRUE   TRUE    TRUE   TRUE  FALSE    TRUE
## YT6    TRUE   TRUE    TRUE   TRUE   TRUE    TRUE   TRUE   TRUE    TRUE
## YT7    TRUE   TRUE    TRUE   TRUE   TRUE    TRUE   TRUE   TRUE    TRUE
## YT8    TRUE   TRUE    TRUE   TRUE   TRUE   FALSE   TRUE   TRUE    TRUE
##     D3S1283 D4S414 D8S264 D22S928  TP53 D9S171
## YT1    TRUE   TRUE   TRUE    TRUE  TRUE   TRUE
## YT2    TRUE   TRUE   TRUE    TRUE  TRUE  FALSE
## YT3   FALSE   TRUE   TRUE   FALSE  TRUE   TRUE
## YT4   FALSE   TRUE   TRUE   FALSE  TRUE   TRUE
## YT5   FALSE   TRUE   TRUE   FALSE  TRUE   TRUE
## YT6   FALSE   TRUE   TRUE    TRUE FALSE  FALSE
## YT7   FALSE   TRUE   TRUE   FALSE FALSE  FALSE
## YT8   FALSE   TRUE  FALSE    TRUE FALSE  FALSE
```

```
pi.e=prop.table(res.cv.modpls.logit$CVMC)%*%matind
pi.e
```

```
##      D2S138 D18S61 D16S422 D17S794 D6S264 D14S65 D18S53 D17S790 D1S225
## [1,]      1      1       1       1   0.74   0.88   0.62       1   0.27
##      D3S1282 D9S179 D5S430 D8S283 D11S916 D2S159 D16S408 D5S346 D10S191
## [1,]    0.01   0.94      1   0.01       1   0.65    0.88      1    0.98
##      D13S173 D6S275 D15S127 D1S305 D4S394 D20S107 D1S197 D1S207 D10S192
```

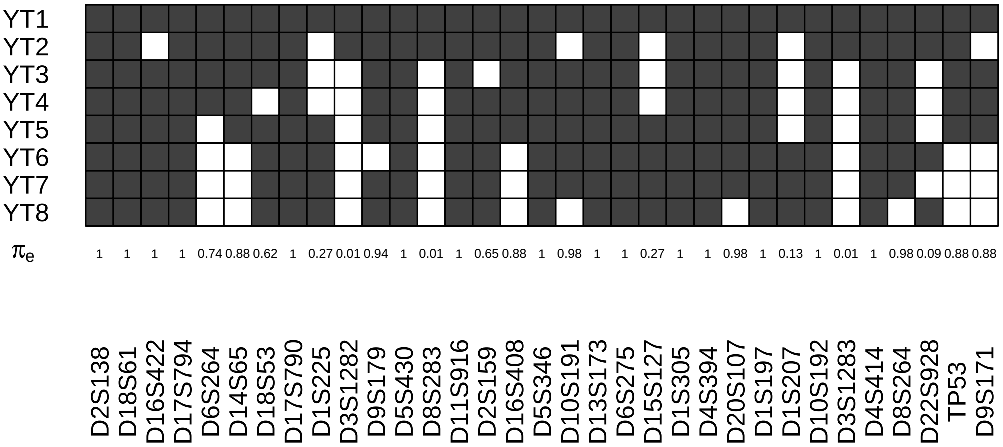

Figure 16: Significance of the predictors vs nbr of components, bootstrap $(\mathbf{y}, \mathbf{T})$, R=1000

```
## [1,]       1      1    0.27      1      1    0.98      1   0.13       1
##      D3S1283 D4S414 D8S264 D22S928 TP53 D9S171
## [1,]    0.01      1   0.98    0.09 0.88   0.88
```

```
signpred(t(matind),labsize=2, plotsize = 12)
text(1:(ncol(matind))-.5,-1,pi.e,cex=.5)
mtext(expression(pi[e]),side=2,las=1,line=2,at=-1)
```

3.3.2 Specifying families, links or custom GLRs

Using `family` and `link` options

The option `modele="pls-glm-logistic"` allows us to to fit a binary logistic with logit link GLR. It is a shortcut for setting these two options `modele="pls-glm-family"` and `family=binomial(link=logit)`.

```
data(aze)
Xaze<-aze[,2:34]
yaze<-aze$y
modpls <- plsRglm(yaze,Xaze,nt=10,modele="pls-glm-logistic",MClassed=TRUE,pvals.expli=TRUE)
modpls2 <- plsRglm(yaze,Xaze,nt=10,modele="pls-glm-family",family=binomial(link=logit),
                   MClassed=TRUE,pvals.expli=TRUE)
```

To replace the logit link with another, use `family=binomial(link=probit)`, `family=binomial(link=cauchit)`, `family=binomial(link=cloglog)`.

```
modpls3 <- plsRglm(yaze,Xaze,nt=10,modele="pls-glm-family",family=binomial(link=probit),
                   MClassed=TRUE,pvals.expli=TRUE)
modpls4 <- plsRglm(yaze,Xaze,nt=10,modele="pls-glm-family",family=binomial(link=cauchit),
                   MClassed=TRUE,pvals.expli=TRUE)
modpls5 <- plsRglm(yaze,Xaze,nt=10,modele="pls-glm-family",family=binomial(link=cloglog),
                   MClassed=TRUE,pvals.expli=TRUE)
```

For each of these other links, one should restart number of components selection using crossvalidation.

```
set.seed(123)
cv.modpls.probit<-cv.plsRglm(dataY=yaze,dataX=Xaze,nt=10,modele="pls-glm-family",
                             family=binomial(link=probit),K=8,NK=100)
```

```
cv.modpls.cauchit<-cv.plsRglm(dataY=yaze,dataX=Xaze,nt=10,modele="pls-glm-family",
                             family=binomial(link=cauchit),K=8,NK=100)
```

```
cv.modpls.cloglog<-cv.plsRglm(dataY=yaze,dataX=Xaze,nt=10,modele="pls-glm-family",
                             family=binomial(link=cloglog),,K=8,NK=100)
```

```
res.cv.modpls.probit=cvtable(summary(cv.modpls.probit, MClassed = TRUE))
## ____************************************************____
## Only naive DoF can be used with missing data
##
## Family: binomial
## Link function: probit
##
## ____There are some NAs in X but not in Y____
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Component____ 5 ____
## ____Component____ 6 ____
## ____Component____ 7 ____
## ____Component____ 8 ____
## ____Component____ 9 ____
## ____Component____ 10 ____
## ____Predicting X with NA in X and not in Y____
## ****________________________________________________****
##
##
## NK: 1,  2,  3,  4,  5,  6,  7,  8,  9,  10
## NK: 11,  12,  13,  14,  15,  16,  17,  18,  19,  20
## NK: 21,  22,  23,  24,  25,  26,  27,  28,  29,  30
## NK: 31,  32,  33,  34,  35,  36,  37,  38,  39,  40
## NK: 41,  42,  43,  44,  45,  46,  47,  48,  49,  50
## NK: 51,  52,  53,  54,  55,  56,  57,  58,  59,  60
## NK: 61,  62,  63,  64,  65,  66,  67,  68,  69,  70
## NK: 71,  72,  73,  74,  75,  76,  77,  78,  79,  80
## NK: 81,  82,  83,  84,  85,  86,  87,  88,  89,  90
## NK: 91,  92,  93,  94,  95,  96,  97,  98,  99,  100
## CV MissClassed criterion:
##  1  2  3  4  5  6  7  8  9
##  2  0 33 46 10  4  1  3  1
##
## CV Q2Chi2 criterion:
##   0
## 100
##
## CV PreChi2 criterion:
##  1  2
## 94  6
```

```
res.cv.modpls.cauchit=cvtable(summary(cv.modpls.cauchit, MClassed = TRUE))
## ____************************************************____
## Only naive DoF can be used with missing data
##
## Family: binomial
## Link function: cauchit
##
```

```
## ____There are some NAs in X but not in Y____
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Component____ 5 ____
## ____Component____ 6 ____
## ____Component____ 7 ____
## ____Component____ 8 ____
## Warning : reciprocal condition number of t(cbind(res$pp,tempppp)[XXNA[3,],,drop=FALSE])%*%cbind(res$pp,tempppp)[
## Warning only 8 components could thus be extracted
## ____Predicting X with NA in X and not in Y____
## ****________________________________________________****
##
##
## NK: 1,  2,  3,  4,  5,  6,  7,  8,  9,  10
## NK: 11,  12,  13,  14,  15,  16,  17,  18,  19,  20
## NK: 21,  22,  23,  24,  25,  26,  27,  28,  29,  30
## NK: 31,  32,  33,  34,  35,  36,  37,  38,  39,  40
## NK: 41,  42,  43,  44,  45,  46,  47,  48,  49,  50
## NK: 51,  52,  53,  54,  55,  56,  57,  58,  59,  60
## NK: 61,  62,  63,  64,  65,  66,  67,  68,  69,  70
## NK: 71,  72,  73,  74,  75,  76,  77,  78,  79,  80
## NK: 81,  82,  83,  84,  85,  86,  87,  88,  89,  90
## NK: 91,  92,  93,  94,  95,  96,  97,  98,  99,  100
## CV MissClassed criterion:
##  1  2  3  4  5  6  7  8
## 29  2 27 16  7 12  2  5
##
## CV Q2Chi2 criterion:
##   0
## 100
##
## CV PreChi2 criterion:
##   1
## 100
```

```
#res.cv.modpls.cloglog=cvtable(summary(cv.modpls.cloglog, MClassed = TRUE))
```

```
layout(matrix(1:4,nrow=2))
plot(res.cv.modpls.logit)
plot(res.cv.modpls.probit)
plot(res.cv.modpls.cauchit)
#plot(res.cv.modpls.cloglog)
layout(1)
```

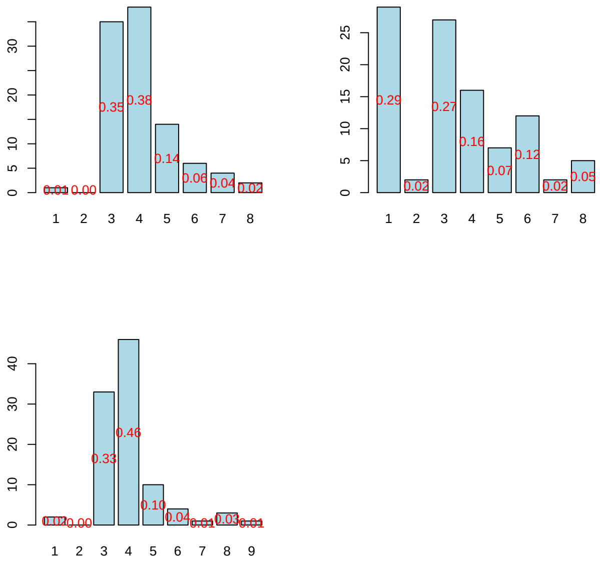

Figure 17: Nb components for logit, probit, cauchit and cloglog links, 8-CV, n=100

Since we are comparing glms with different link function a direct comparison of the estimates of the model, such as the one provided below, is not relevant. Instead we compare BC_a Bootstrap ($mathbfy,$ $mathbfT$) based significance of the variables and plot the results using the signpred function.

```
data.frame(logit=modpls2$Std.Coeffs,probit=modpls3$Std.Coeffs,
           cauchit=modpls4$Std.Coeffs,cloglog=modpls5$Std.Coeffs)
```

```
##               logit      probit  cauchit   cloglog
## Intercept -0.28153 -0.3118285 -2.68062 -0.754038
## D2S138    -1.24513 -0.7573422 -2.43563 -0.838584
## D18S61     1.91630  1.2102996  5.92632  1.226347
## D16S422   -0.13835 -0.4156485 -0.34574 -0.502377
## D17S794    1.75798  1.2475210  4.67314  0.455824
## D6S264    -0.04176  0.5459001 -2.87545  0.105627
## D14S65    -0.09021  0.0007938  2.23711  0.094785
## D18S53     0.23184 -0.3068814 -0.40407 -0.060258
## D17S790   -1.60461 -1.3850280 -3.08423 -0.070063
## D1S225    -0.07654 -0.3266758  0.19084 -0.346550
## D3S1282    0.07130  0.2660855  0.08213 -0.006113
## D9S179     1.25291  0.6626852  0.26260  0.440091
## D5S430    -0.39566 -1.0168237 -5.80080 -0.370428
## D8S283    -0.11684 -0.3377915  0.81736 -0.066913
## D11S916    0.10535  0.2376243  3.27206  0.586031
## D2S159     0.64625  0.6522306  4.30848  0.666585
## D16S408   -0.77149 -0.0858054  3.68491  0.399375
## D5S346     1.83010  1.2062642  4.21555  0.864934
## D10S191    0.21089  0.2002734 -2.00996 -0.197892
## D13S173    1.40402  0.8049591  1.00667  0.532997
## D6S275    -0.65335 -0.4995541 -3.77352 -0.571409
## D15S127   -0.63920 -0.4790719 -3.13018 -0.515623
## D1S305     1.31361  0.8739911  5.37317  0.898312
## D4S394    -2.43322 -0.9472823 -5.02391 -0.815028
## D20S107    0.01186  0.0007201 -3.86012 -0.979947
## D1S197    -1.11056 -0.5256046 -3.65892 -0.468815
## D1S207     1.21068  0.6104127  0.66750  0.309633
## D10S192    1.01538  0.7229599  4.47329  0.635891
## D3S1283    0.69431  0.1497382 -0.53379 -0.199955
## D4S414     1.11589  0.5535850  3.22387  0.509255
## D8S264    -0.47945  0.3727693  5.00007  0.182964
## D22S928   -0.86121 -0.5062403 -0.46118 -0.425737
## TP53      -1.16590 -0.3591004 -2.97188 -0.361731
## D9S171     0.02704 -0.1287678  2.49295  0.405163

temp.ci.logit<-confints.bootpls(aze.bootYT4)
aze.bootYT4.probit=bootplsglm(modpls3,R=1000)
temp.ci.probit<-confints.bootpls(aze.bootYT4.probit)
aze.bootYT4.cauchit=bootplsglm(modpls4,R=1000)
temp.ci.cauchit<-confints.bootpls(aze.bootYT4.cauchit)
aze.bootYT4.cloglog=bootplsglm(modpls5,R=1000)
temp.ci.cloglog<-confints.bootpls(aze.bootYT4.cloglog)

ind.BCa.logit <- (temp.ci.logit[,7]<0&temp.ci.logit[,8]<0)|
                  (temp.ci.logit[,7]>0&temp.ci.logit[,8]>0)
ind.BCa.probit <- (temp.ci.probit[,7]<0&temp.ci.probit[,8]<0)|
                   (temp.ci.probit[,7]>0&temp.ci.probit[,8]>0)
ind.BCa.cauchit <- (temp.ci.cauchit[,7]<0&temp.ci.cauchit[,8]<0)|
                    (temp.ci.cauchit[,7]>0&temp.ci.cauchit[,8]>0)
ind.BCa.cloglog <- (temp.ci.cloglog[,7]<0&temp.ci.cloglog[,8]<0)|
                    (temp.ci.cloglog[,7]>0&temp.ci.cloglog[,8]>0)

(matind=(rbind(logit=ind.BCa.logit,probit=ind.BCa.probit,cauchit=ind.BCa.cauchit,
               cloglog=ind.BCa.cloglog)))

##         D2S138 D18S61 D16S422 D17S794 D6S264 D14S65 D18S53 D17S790
## logit     TRUE   TRUE    TRUE    TRUE   TRUE   TRUE  FALSE    TRUE
## probit    TRUE   TRUE    TRUE    TRUE  FALSE  FALSE  FALSE    TRUE
## cauchit   TRUE   TRUE   FALSE    TRUE   TRUE   TRUE  FALSE    TRUE
## cloglog   TRUE   TRUE   FALSE   FALSE  FALSE  FALSE  FALSE   FALSE
##         D1S225 D3S1282 D9S179 D5S430 D8S283 D11S916 D2S159 D16S408
## logit    FALSE   FALSE   TRUE   TRUE  FALSE    TRUE   TRUE    TRUE
## probit    TRUE    TRUE   TRUE   TRUE  FALSE   FALSE   TRUE   FALSE
## cauchit  FALSE   FALSE  FALSE   TRUE  FALSE    TRUE  FALSE   FALSE
## cloglog   TRUE   FALSE   TRUE   TRUE  FALSE    TRUE   TRUE   FALSE
##         D5S346 D10S191 D13S173 D6S275 D15S127 D1S305 D4S394 D20S107
## logit     TRUE    TRUE    TRUE   TRUE   FALSE   TRUE   TRUE    TRUE
## probit    TRUE   FALSE    TRUE  FALSE    TRUE   TRUE   TRUE   FALSE
## cauchit   TRUE   FALSE   FALSE   TRUE   FALSE   TRUE   TRUE   FALSE
## cloglog   TRUE   FALSE    TRUE   TRUE    TRUE   TRUE   TRUE    TRUE
##         D1S197 D1S207 D10S192 D3S1283 D4S414 D8S264 D22S928   TP53
## logit     TRUE  FALSE    TRUE   FALSE   TRUE   TRUE   FALSE   TRUE
## probit    TRUE   TRUE    TRUE   FALSE  FALSE  FALSE    TRUE  FALSE
## cauchit   TRUE  FALSE    TRUE   FALSE  FALSE   TRUE   FALSE  FALSE
## cloglog   TRUE   TRUE    TRUE   FALSE  FALSE  FALSE    TRUE  FALSE
##         D9S171
## logit     TRUE
## probit   FALSE
## cauchit   TRUE
## cloglog  FALSE

signpred(t(matind),labsize=2, plotsize = 12)
```

Any of the GLM family implemented in *R*, and even user specified links, can be used by using the option `modele="pls-glm-family"` and setting `family` to the desired value. We reproduce the example given in the help of the `glm` of an user-specified link, a logit model for pdays, see Shaffer (2004).

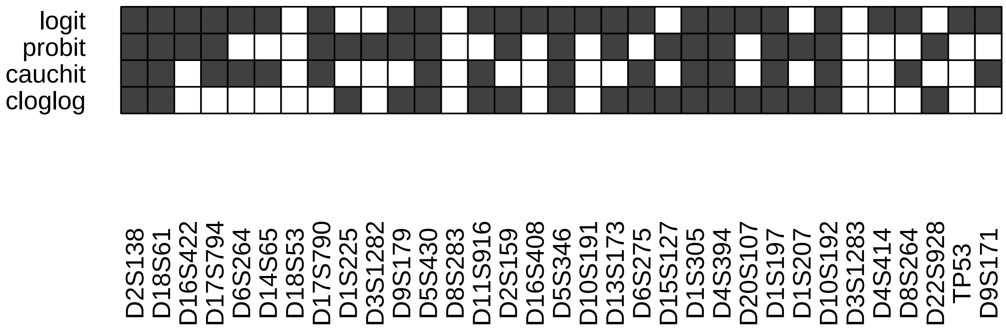

Figure 18: Significance of the predictors vs link, bootstrap $(\mathbf{y}, \mathbf{T})$, R=1000

```
logexp <- function(exposure = 1)
{
    linkfun <- function(mu) qlogis(mu^(1/exposure))
    linkinv <- function(eta)  plogis(eta)^exposure
    mu.eta <- function(eta) exposure * plogis(eta)^(exposure-1) *
      .Call(stats:::C_logit_mu_eta, eta, PACKAGE = "stats")
    valideta <- function(eta) TRUE
    link <- paste("logexp(", deparse(substitute(exposure)), ")",
                   sep="")
    structure(list(linkfun = linkfun, linkinv = linkinv,
                   mu.eta = mu.eta, valideta = valideta,
                   name = link),
              class = "link-glm")
}
binomial(logexp(3))
##
## Family: binomial
## Link function: logexp(3)
data(aze_compl)
Xaze_compl<-aze_compl[,2:34]
yaze_compl<-aze_compl$y
modplscustom <- plsRglm(yaze_compl,Xaze_compl,nt=10,modele="pls-glm-family",
family=binomial(link=logexp(3)),MClassed=TRUE,pvals.expli=TRUE)
## ____************************************************____
##
## Family: binomial
## Link function: logexp(3)
##
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Component____ 5 ____
## ____Component____ 6 ____
## ____Component____ 7 ____
## ____Component____ 8 ____
## ____Component____ 9 ____
## ____Component____ 10 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****
```

```
modplscustom
```

```
## Number of required components:
## [1] 10
## Number of successfully computed components:
## [1] 10
## Coefficients:
##               [,1]
## Intercept -0.17337
## D2S138    -0.89999
## D18S61     2.39045
## D16S422   -1.28883
## D17S794    1.63611
## D6S264     0.42436
## D14S65    -0.15137
## D18S53    -0.16509
## D17S790   -1.10990
## D1S225     0.24969
## D3S1282   -0.52479
## D9S179     0.44527
## D5S430    -0.67561
## D8S283     0.63264
## D11S916    0.45407
## D2S159     0.36409
## D16S408    0.48960
## D5S346     1.18813
## D10S191   -0.01616
## D13S173    0.45029
## D6S275    -0.65743
## D15S127    0.33754
## D1S305     1.04005
## D4S394    -0.86576
## D20S107   -1.01708
## D1S197    -1.45460
## D1S207     1.25697
## D10S192    1.14235
## D3S1283   -0.38086
## D4S414     0.21858
## D8S264     0.83803
## D22S928   -0.23684
## TP53      -2.09116
## D9S171     0.12095
## Information criteria and Fit statistics:
##              AIC   BIC Missclassed Chi2_Pearson_Y RSS_Y   R2_Y
## Nb_Comp_0  145.8 148.5          49         104.00 25.91     NA
## Nb_Comp_1  117.2 122.5          27         100.25 19.16 0.2605
## Nb_Comp_2  110.2 118.2          27          97.06 17.48 0.3255
## Nb_Comp_3  105.3 115.9          25          97.99 16.15 0.3769
## Nb_Comp_4  104.7 118.0          23          94.91 15.72 0.3933
## Nb_Comp_5  105.5 121.3          23          88.51 15.63 0.3970
## Nb_Comp_6  106.7 125.2          23          87.56 15.50 0.4020
## Nb_Comp_7  108.2 129.4          20          89.47 15.34 0.4080
## Nb_Comp_8  110.0 133.8          21          87.20 15.39 0.4062
## Nb_Comp_9  111.9 138.4          21          88.34 15.34 0.4082
## Nb_Comp_10 113.9 143.0          22          88.29 15.32 0.4086
##            R2_residY RSS_residY
## Nb_Comp_0         NA      25.91
## Nb_Comp_1     -4.698     147.66
## Nb_Comp_2     -6.576     196.33
## Nb_Comp_3     -7.429     218.43
## Nb_Comp_4     -8.714     251.72
## Nb_Comp_5     -9.176     263.69
## Nb_Comp_6     -9.344     268.05
## Nb_Comp_7     -9.561     273.66
## Nb_Comp_8     -9.945     283.62
## Nb_Comp_9    -10.010     285.31
## Nb_Comp_10   -10.041     286.12
## Model with all the required components:
##
## Call:  glm(formula = YwotNA ~ ., family = family, data = tttrain)
##
## Coefficients:
## (Intercept)         tt.1         tt.2         tt.3         tt.4
##      1.2522       1.1494       0.3483       0.7214       0.3567
##        tt.5         tt.6         tt.7         tt.8         tt.9
##      0.2635       0.2321       0.1859       0.1184       0.0845
##       tt.10
##      0.0570
##
## Degrees of Freedom: 103 Total (i.e. Null);  93 Residual
## Null Deviance:     144
## Residual Deviance: 91.9  AIC: 114
```

```
set.seed(123)
cv.modplscustom<-cv.plsRglm(yaze_compl,Xaze_compl,nt=10,modele="pls-glm-family",
                            family=binomial(link=logexp(3)),K=8,NK=100)
```

```
res.cv.modplscustom=cvtable(summary(cv.modplscustom,MClassed=TRUE))
## ____************************************************____
##
## Family: binomial
## Link function: logexp(3)
##
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Component____ 5 ____
## ____Component____ 6 ____
## ____Component____ 7 ____
## ____Component____ 8 ____
## ____Component____ 9 ____
## ____Component____ 10 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****
##
##
## NK: 1,  2,  3,  4,  5,  6,  7,  8,  9,  10
## NK: 11,  12,  13,  14,  15,  16,  17,  18,  19,  20
## NK: 21,  22,  23,  24,  25,  26,  27,  28,  29,  30
## NK: 31,  32,  33,  34,  35,  36,  37,  38,  39,  40
## NK: 41,  42,  43,  44,  45,  46,  47,  48,  49,  50
## NK: 51,  52,  53,  54,  55,  56,  57,  58,  59,  60
## NK: 61,  62,  63,  64,  65,  66,  67,  68,  69,  70
## NK: 71,  72,  73,  74,  75,  76,  77,  78,  79,  80
## NK: 81,  82,  83,  84,  85,  86,  87,  88,  89,  90
## NK: 91,  92,  93,  94,  95,  96,  97,  98,  99,  100
## CV MissClassed criterion:
##  1  2  3  4  5  6  7  8  9 10
## 31 13 33  3  3  1  5  2  5  4
##
## CV Q2Chi2 criterion:
##   0
## 100
##
## CV PreChi2 criterion:
##   1
## 100
```

```
plot(res.cv.modplscustom)
```

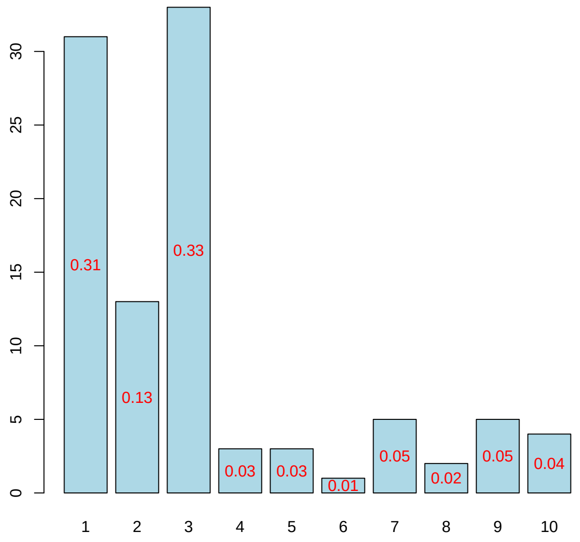

Figure 19: Nb components, 8-CV, n=100

3.3.3 Method and Results: imputed dataset

Cross-validation

We imputed a complete dataset aze_ compl from the raw dataset aze using the *mice* package van Buuren and Groothuis-Oudshoorn (2011) in order to copare the results of the PLSGLR algorithm on these two datasets.

```
rm(list = ls())
library(plsRglm)
data(aze_compl)
Xaze_compl<-aze_compl[,2:34]
yaze_compl<-aze_compl$y
```

Again we use a repeated k-fold cross validation to find the number of components to retain with $k = 8$ balanced groups of 13 subjects. We then again chose to set to 10, thanks to the option nt=10, the maximal number of components for the cross-validation that the cv.plsRglm function would try to compute. According to field experts, this number of components should be greater to the real number of components featured in the dataset. The cross-validation step is performed by running the following command line.

```
cv.modpls_compl<-cv.plsRglm(dataY=yaze_compl,dataX=Xaze_compl,nt=10,
                            modele="pls-glm-logistic",K=8)
```

For PLSGLR models, the cross-validation results can be summed up in a single table using the summary[2]. Results are obtained by the following command line.

```
res.cv.modpls_compl<-cvtable(summary(cv.modpls_compl, MClassed=TRUE))
## ____************************************************____
##
## Family: binomial
## Link function: logit
##
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Component____ 5 ____
## ____Component____ 6 ____
## ____Component____ 7 ____
## ____Component____ 8 ____
## ____Component____ 9 ____
## ____Component____ 10 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****
##
##
## NK: 1
## CV MissClassed criterion:
## 1 2
## 0 1
##
## CV Q2Chi2 criterion:
## 0
## 1
##
## CV PreChi2 criterion:
## 1
## 1
```

The number of significant predictors per components can be obtained via the following code:

[2] for PLSR models the cross-validation results can be summed up in a single table using the functionsummary.

```
res10_compl<-plsRglm(yaze_compl, Xaze_compl, nt=10, modele="pls-glm-logistic",
                     pvals.expli=TRUE)
```

```
## ____************************************************____
##
## Family: binomial
## Link function: logit
##
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Component____ 5 ____
## ____Component____ 6 ____
## ____Component____ 7 ____
## ____Component____ 8 ____
## ____Component____ 9 ____
## ____Component____ 10 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****
```

```
colSums(res10_compl$pvalstep)
```

```
## temppvalstep temppvalstep temppvalstep temppvalstep temppvalstep
##            2            1            0            0            0
## temppvalstep temppvalstep temppvalstep temppvalstep temppvalstep
##            0            0            0            0            0
```

The number of significant predictors within each component, which is a criteria of significance for Bastien et al. (2005), is implemented in the package with the options `sparse=TRUE` and `sparseStop=TRUE`.

```
modpls2_compl <- plsRglm(yaze_compl,Xaze_compl, nt = 10, modele = "pls-glm-logistic",
                         sparse=TRUE,sparseStop=TRUE)
```

```
## ____************************************************____
##
## Family: binomial
## Link function: logit
##
## ____Component____ 1 ____
## ____Component____ 2 ____
## Warning : 32 < 10^{-12}
## Warning only 2 components could thus be extracted
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****
```

The number of significant predictors within each component tells us to only build 2 components when the AIC criteria gives us 4 components and the BIC concludes to 3 components. But for this study, the most important criteria was to minimize the miss classification rate after cross-validation, criteria which let us know to build 3 -in agreement with BIC criteria-, 4 -in agreement with AIC criteria-, 8 or 10 components. In order to confirm the choice of retaining 3 components, the cross-validation was run 100 times by randomly creating groups. Here are the command lines:

```
set.seed(123)
cv.modpls_compl<-cv.plsRglm(dataY=yaze_compl,dataX=Xaze_compl,nt=10,
                            modele="pls-glm-logistic",K=8,NK=100)
```

```
res.cv.modpls_compl=cvtable(summary(cv.modpls_compl, MClassed = TRUE))
```

```
## ____************************************************____
##
## Family: binomial
## Link function: logit
##
```

```
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Component____ 5 ____
## ____Component____ 6 ____
## ____Component____ 7 ____
## ____Component____ 8 ____
## ____Component____ 9 ____
## ____Component____ 10 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****
##
##
## NK: 1,  2,  3,  4,  5,  6,  7,  8,  9,  10
## NK: 11,  12,  13,  14,  15,  16,  17,  18,  19,  20
## NK: 21,  22,  23,  24,  25,  26,  27,  28,  29,  30
## NK: 31,  32,  33,  34,  35,  36,  37,  38,  39,  40
## NK: 41,  42,  43,  44,  45,  46,  47,  48,  49,  50
## NK: 51,  52,  53,  54,  55,  56,  57,  58,  59,  60
## NK: 61,  62,  63,  64,  65,  66,  67,  68,  69,  70
## NK: 71,  72,  73,  74,  75,  76,  77,  78,  79,  80
## NK: 81,  82,  83,  84,  85,  86,  87,  88,  89,  90
## NK: 91,  92,  93,  94,  95,  96,  97,  98,  99,  100
## CV MissClassed criterion:
##  1  2  3  4  5  6  7  8  9 10
## 29  8 37  2  2  4  5  6  3  4
##
## CV Q2Chi2 criterion:
##   0
## 100
##
## CV PreChi2 criterion:
##   1
## 100
```

The results (Fig. 20) confirm the results obtained during the original cross-validation and that's the reason why we decided to retain a 3 components model. Please note that rather surprisingly, a 1 component model is favored in about 3 out of 10 crossvalidations whereas it was put forward by any of the other criteria. The binary logistic PLSGLR model is:

$$\mathbb{P}\left(y=1\right)=\frac{\exp\left(\mu+\sum_{h=1}^{3}c_h t_h\right)}{1+\exp\left(\mu+\sum_{h=1}^{3}c_h t_h\right)} \tag{7}$$

where t_h is the h^{th} component, c_h the coefficients of the logistic regression of the response variable y on the components t_h and μ the intercept.

```
plot(res.cv.modpls_compl)
```

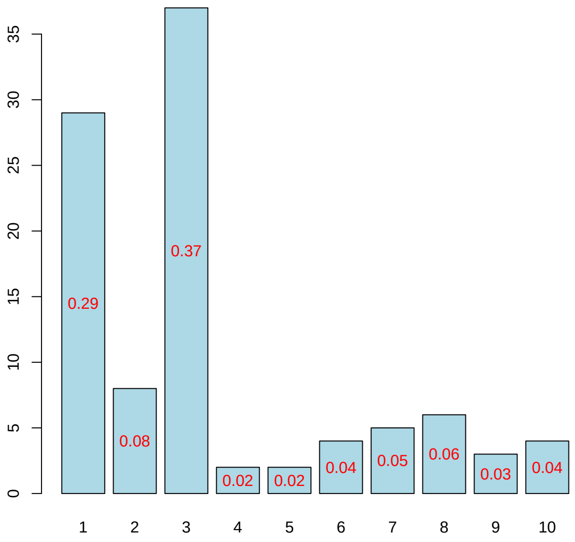

Figure 20: Nb components, 8-CV, n=100

Now, the PLSGLR regression is done in order to obtain these coefficients c_h and the intercept.

```
res_compl<-plsRglm(yaze_compl, Xaze_compl, nt = 3, modele = "pls-glm-logistic",
                   pvals.expli=TRUE)

## ____************************************************____
##
## Family: binomial
## Link function: logit
##
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****

res_compl$FinalModel

##
## Call:  glm(formula = YwotNA ~ ., family = family, data = tttrain)
##
## Coefficients:
## (Intercept)         tt.1         tt.2         tt.3
##      -0.230        1.360        0.451        0.735
##
## Degrees of Freedom: 103 Total (i.e. Null);  100 Residual
## Null Deviance:     144
## Residual Deviance: 97.2  AIC: 105
```

It is also possible to obtain the matrix $\mathbf{W}^*$ with the following command line:

```
res_compl$wwetoile

##         Coord_Comp_1 Coord_Comp_2 Coord_Comp_3
## D2S138      0.016418     -0.21375    -0.230977
## D18S61      0.305844      0.26024     0.475067
## D16S422    -0.013421     -0.15720    -0.166544
## D17S794     0.131059      0.19708     0.217451
## D6S264     -0.101889     -0.07257     0.287207
## D14S65     -0.173631     -0.14560     0.054495
## D18S53      0.051354     -0.07572     0.072797
## D17S790    -0.128149     -0.22194    -0.008700
## D1S225      0.064117     -0.10483     0.042804
## D3S1282    -0.129699     -0.25572    -0.031888
## D9S179      0.054226     -0.10657    -0.072031
## D5S430      0.013506     -0.24247    -0.167629
## D8S283      0.233736      0.02401     0.028430
## D11S916     0.430456      0.07599    -0.123851
## D2S159      0.097213     -0.02646     0.179212
## D16S408     0.046090     -0.02248    -0.107424
## D5S346      0.324886      0.07382    -0.002599
## D10S191    -0.005398     -0.15839     0.232896
## D13S173     0.109983      0.13870     0.126744
## D6S275     -0.245769     -0.32646     0.160458
## D15S127     0.118649     -0.05855    -0.046058
## D1S305      0.207923     -0.05727     0.131704
## D4S394     -0.053537     -0.30268    -0.164555
## D20S107    -0.125769     -0.31572    -0.259423
## D1S197     -0.089950     -0.43873    -0.514108
## D1S207      0.113068      0.01189     0.169432
## D10S192     0.173157      0.11215     0.203668
## D3S1283    -0.235752     -0.40593     0.007044
## D4S414     -0.119512     -0.20543     0.251344
## D8S264      0.286067      0.09273     0.010267
## D22S928     0.129148     -0.03919     0.042384
## TP53       -0.281527     -0.56362    -0.385875
## D9S171     -0.033558     -0.16804     0.129786
```

It is also possible to display the biplot of the observations and the predictors (Figure 21).

```
biplot(res_compl$tt,res_compl$pp)
```

Then, in order to have results which are interpretable in practice, let run the following command line and so obtain the coefficients β_j of the predictors $\mathbf{x}_j,\ 1 \leqslant j \leqslant 33$ of the final model.

```
res_compl$Std.Coeffs

##               [,1]
## Intercept -0.22969
## D2S138    -0.24396
## D18S61     0.88279
## D16S422   -0.21165
## D17S794    0.42711
## D6S264     0.03977
## D14S65    -0.26187
## D18S53     0.08920
## D17S790   -0.28091
## D1S225     0.07137
## D3S1282   -0.31532
## D9S179    -0.02729
## D5S430    -0.21431
## D8S283     0.34971
## D11S916    0.52884
## D2S159     0.25205
## D16S408   -0.02642
## D5S346     0.47338
## D10S191    0.09237
```

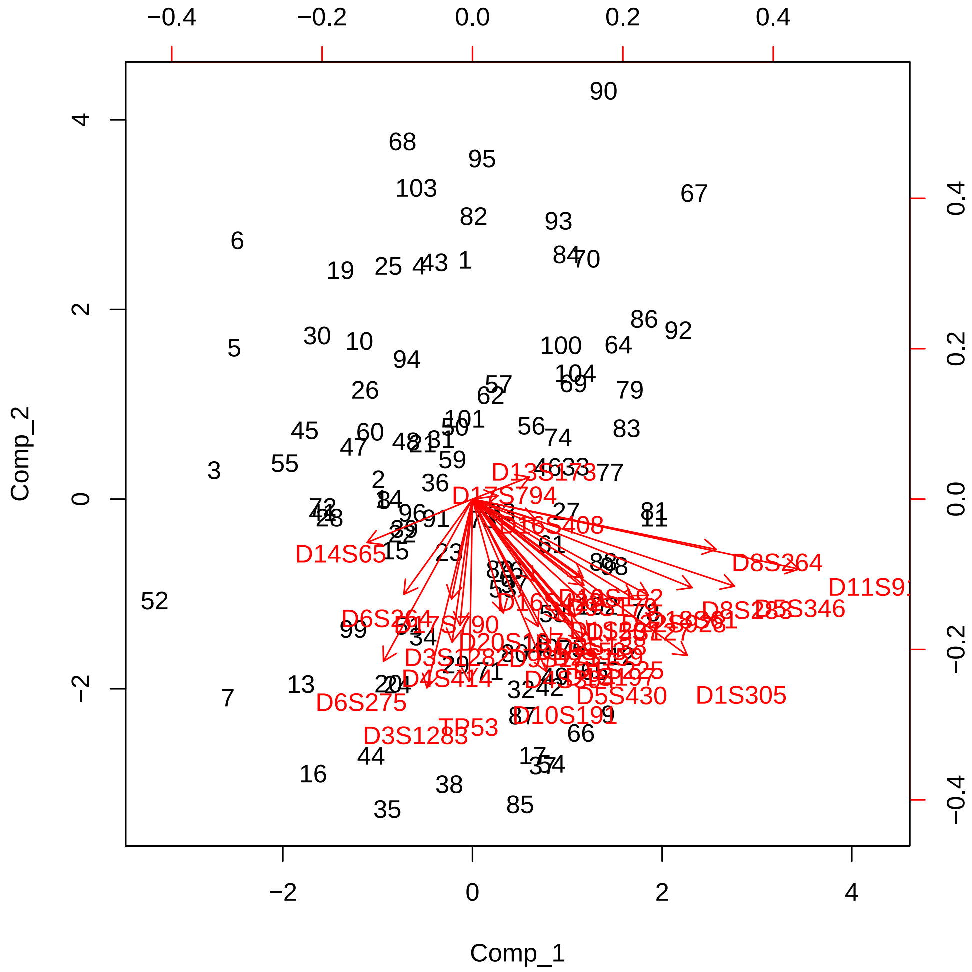

Figure 21: Biplot of the observations and the variables

```
## D13S173    0.30541
## D6S275    -0.36374
## D15S127    0.10112
## D1S305     0.35383
## D4S394    -0.33043
## D20S107   -0.50433
## D1S197    -0.69835
## D1S207     0.28374
## D10S192    0.43591
## D3S1283   -0.49877
## D4S414    -0.07054
## D8S264     0.43857
## D22S928    0.18916
## TP53      -0.92108
## D9S171    -0.02609
```

Hard thresholding PLS regression and automatic selection of the number of components (Bastien et al. (2005)) is also available:

```
modpls3_compl <- plsRglm(yaze_compl, Xaze_compl, nt = 10, modele = "pls-glm-logistic",
                         sparse=FALSE,sparseStop=TRUE)
modpls4_compl <- plsRglm(yaze_compl, Xaze_compl, nt = 10, modele = "pls-glm-logistic",
                         sparse=TRUE,sparseStop=FALSE)
```

Bootstrap $(\mathbf{y}, \mathbf{X})$

However, what is really important is to know which of them are significantly different from zero. We can also answer to this question with the bootstrap techniques we insert in this package.

Let us begin with the bootstrap on the (Y, X). This method, which seems to be natural, has some trouble in this case. Indeed, with the help of the boxplot, we decided to only focus on the BC_a CI, because of the fact of the clearly non symmetrical distributions of the estimators (see Figure 22). And when we choose 3 components, some of the CI become disproportionate (see Figure 23), depriving us of any graphical interpretation. So, the only way to see if any predictors is significantly different from 0, is to use the function `confints.bootpls`, which will allow to see the values of the CI for the four different type of CI. Thanks to this function, we can see that only one predictor is significantly different from 0, TP53.

```
set.seed(123)
aze_compl.bootYX3=bootplsglm(res_compl,typeboot="plsmodel",R=1000)
```

By default with PLSGLR models the option `typeboot` is set to `typeboot="fmodel_np"` -Bootstrap $(\mathbf{y}, \mathbf{T})$-, we change this setting using the option `typeboot="plsmodel"` -Bootstrap $(\mathbf{y}, \mathbf{X})$-.

```
boxplots.bootpls(aze_compl.bootYX3,las=2,mar=c(5,2,1,1)+0.1)
```

```
temp.ci=confints.bootpls(aze_compl.bootYX3)

## Warning:  extreme order statistics used as endpoints
## Warning:  extreme order statistics used as endpoints

plots.confints.bootpls(temp.ci,typeIC="BCa",colIC=c("blue","blue","blue","blue"),
                       legendpos ="topright",las=2,mar=c(5,2,1,1)+0.1)
```

Bootstrap $(\mathbf{y}, \mathbf{T})$

However, due to the problems of the previous results, we decided to choose the second type of bootstrap, that is to say the one which do re-sampling on the couple (Y, T) (Bastien et al., 2005). Indeed, it is more stable and faster than the first one. We set at 1000 the number of re-sampling. So we obtain a graphic representing the confidence intervals (CI) for each of the predictors (see Figure 25) and a boxplot as well (see Figure 24). These graphics were obtained by starting these command lines:

```
set.seed(123)
aze_compl.bootYT3=bootplsglm(res_compl,R=1000)
```

By default with PLSGLR models the option `typeboot` is set to `typeboot="fmodel_np"` -Bootstrap $(\mathbf{y}, \mathbf{T})$-.

```
boxplots.bootpls(aze_compl.bootYT3,las=2,mar=c(5,2,1,1)+0.1)
```

```
temp.ci3<-confints.bootpls(aze_compl.bootYT3)
plots.confints.bootpls(temp.ci3,typeIC="BCa",colIC=c("blue","blue","blue","blue"),
                       legendpos ="topright",las=2,mar=c(5,2,1,1)+0.1)
```

Remark 7. In this paper, we decided to only focus on the *BCa* CI. But, with this package, it is naturally possible to obtain CI with percentile, normal or basic bootstrap as well.

With the help of Figure 25, we can see that only 6 predictors is significantly different from 0. But, it could be interesting to display, through the models with 1 to 10 components, which of the predictors are significantly different from zero so that we could know if there is a stability of significant predictors or not (see Figure 26). A function is available in our package, called `signpred`, to do this kind of graphic.

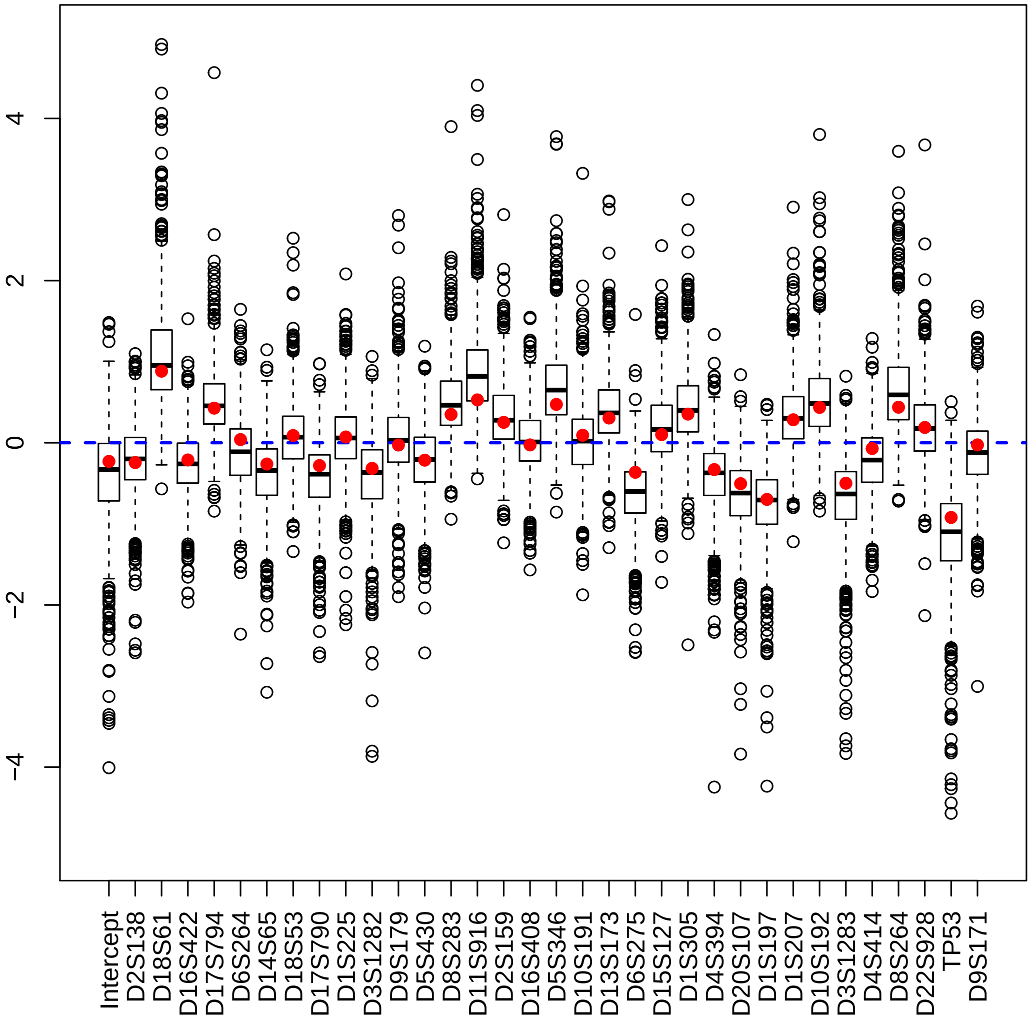

Figure 22: Bootstrap $(\mathbf{y}, \mathbf{X})$ distribution of the coefficients of the predictors, R=1000

As we can see on the figure 26, there are several differences between the model with 1 and 3 components. Indeed, 6 predictors, significant in the 1 component model, become non-significant in the 3 components model. During the cross-validation, 29 percents of results give 1 component and 37 percents give 4 components, representing than 66 percents of the results obtained during the 100 cross-validation made at the beginning.

The bootstrap technique used in this study, which is clearly faster and more stable than the other one, but the results between the two techniques are really different and so it could be interesting to confront them with the help of some simulations.

```
res_compl1<-plsRglm(yaze_compl, Xaze_compl, nt = 1, modele = "pls-glm-logistic")
res_compl2<-plsRglm(yaze_compl, Xaze_compl, nt = 2, modele = "pls-glm-logistic")
res_compl4<-plsRglm(yaze_compl, Xaze_compl, nt = 4, modele = "pls-glm-logistic")
res_compl5<-plsRglm(yaze_compl, Xaze_compl, nt = 5, modele = "pls-glm-logistic")
res_compl6<-plsRglm(yaze_compl, Xaze_compl, nt = 6, modele = "pls-glm-logistic")
res_compl7<-plsRglm(yaze_compl, Xaze_compl, nt = 7, modele = "pls-glm-logistic")
res_compl8<-plsRglm(yaze_compl, Xaze_compl, nt = 8, modele = "pls-glm-logistic")
res_compl9<-plsRglm(yaze_compl, Xaze_compl, nt = 9, modele = "pls-glm-logistic")
res_compl10<-plsRglm(yaze_compl, Xaze_compl, nt = 10, modele = "pls-glm-logistic")

aze_compl.bootYT1=bootplsglm(res_compl1,R=1000)
```

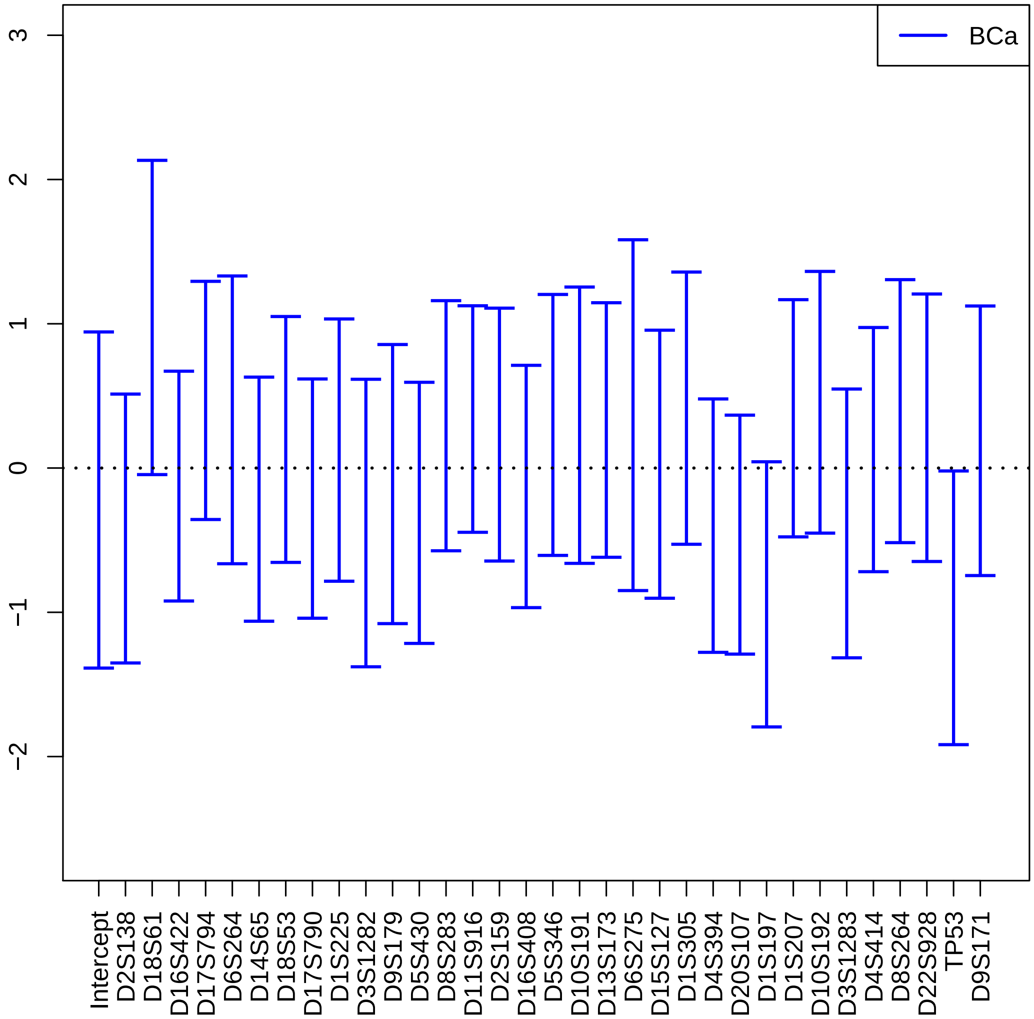

Figure 23: CI of the coefficients of the predictors, bootstrap $(\mathbf{y}, \mathbf{X})$, R=1000

```
aze_compl.bootYT2=bootplsglm(res_compl2,R=1000)
aze_compl.bootYT4=bootplsglm(res_compl4,R=1000)
aze_compl.bootYT5=bootplsglm(res_compl5,R=1000)
aze_compl.bootYT6=bootplsglm(res_compl6,R=1000)
aze_compl.bootYT7=bootplsglm(res_compl7,R=1000)
aze_compl.bootYT8=bootplsglm(res_compl8,R=1000)
aze_compl.bootYT9=bootplsglm(res_compl9,R=1000)
aze_compl.bootYT10=bootplsglm(res_compl10,R=1000)

temp.ci1<-confints.bootpls(aze_compl.bootYT1)
temp.ci2<-confints.bootpls(aze_compl.bootYT2)
temp.ci4<-confints.bootpls(aze_compl.bootYT4)
temp.ci5<-confints.bootpls(aze_compl.bootYT5)
temp.ci6<-confints.bootpls(aze_compl.bootYT6)
temp.ci7<-confints.bootpls(aze_compl.bootYT7)
temp.ci8<-confints.bootpls(aze_compl.bootYT8)
temp.ci9<-confints.bootpls(aze_compl.bootYT9)
temp.ci10<-confints.bootpls(aze_compl.bootYT10)
```

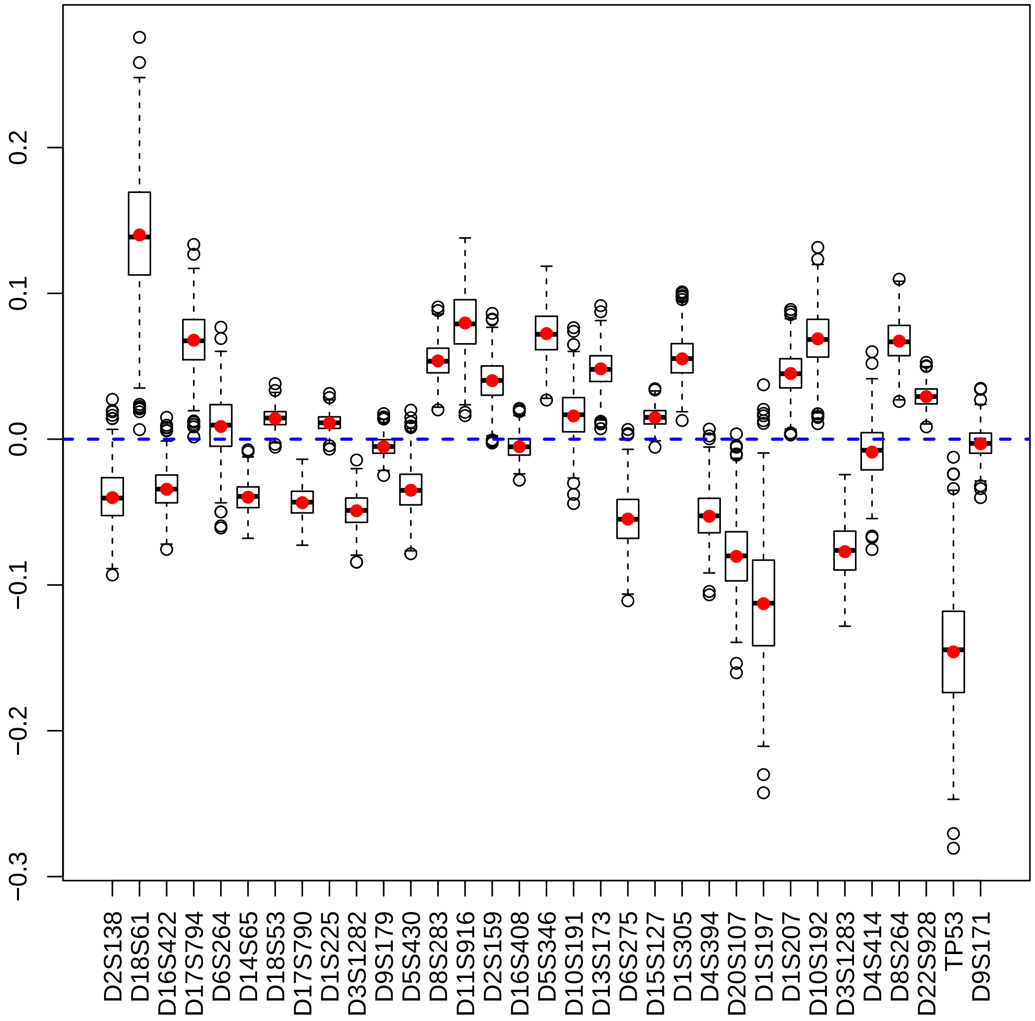

Figure 24: Bootstrap $(\mathbf{y}, \mathbf{T})$ distribution of the coefficients of the predictors, R=1000

```
ind.BCa.aze_complYT1 <- (temp.ci1[,7]<0&temp.ci1[,8]<0)|(temp.ci1[,7]>0&temp.ci1[,8]>0)
ind.BCa.aze_complYT2 <- (temp.ci2[,7]<0&temp.ci2[,8]<0)|(temp.ci2[,7]>0&temp.ci2[,8]>0)
ind.BCa.aze_complYT3 <- (temp.ci3[,7]<0&temp.ci3[,8]<0)|(temp.ci3[,7]>0&temp.ci3[,8]>0)
ind.BCa.aze_complYT4 <- (temp.ci4[,7]<0&temp.ci4[,8]<0)|(temp.ci4[,7]>0&temp.ci4[,8]>0)
ind.BCa.aze_complYT5 <- (temp.ci5[,7]<0&temp.ci5[,8]<0)|(temp.ci5[,7]>0&temp.ci5[,8]>0)
ind.BCa.aze_complYT6 <- (temp.ci6[,7]<0&temp.ci6[,8]<0)|(temp.ci6[,7]>0&temp.ci6[,8]>0)
ind.BCa.aze_complYT7 <- (temp.ci7[,7]<0&temp.ci7[,8]<0)|(temp.ci7[,7]>0&temp.ci7[,8]>0)
ind.BCa.aze_complYT8 <- (temp.ci8[,7]<0&temp.ci8[,8]<0)|(temp.ci8[,7]>0&temp.ci8[,8]>0)
ind.BCa.aze_complYT9 <- (temp.ci9[,7]<0&temp.ci9[,8]<0)|(temp.ci9[,7]>0&temp.ci9[,8]>0)
ind.BCa.aze_complYT10 <- (temp.ci10[,7]<0&temp.ci10[,8]<0)|(temp.ci10[,7]>0&temp.ci10[,8]>0)
```

```
(matind=(rbind(YT1=ind.BCa.aze_complYT1,YT2=ind.BCa.aze_complYT2,YT3=ind.BCa.aze_complYT3,
               YT4=ind.BCa.aze_complYT4,YT5=ind.BCa.aze_complYT5,YT6=ind.BCa.aze_complYT6,
               YT7=ind.BCa.aze_complYT7,YT8=ind.BCa.aze_complYT8,YT9=ind.BCa.aze_complYT9,
               YT10=ind.BCa.aze_complYT10)))

##      D2S138 D18S61 D16S422 D17S794 D6S264 D14S65 D18S53 D17S790 D1S225
## YT1    TRUE   TRUE    TRUE    TRUE   TRUE   TRUE   TRUE    TRUE   TRUE
```

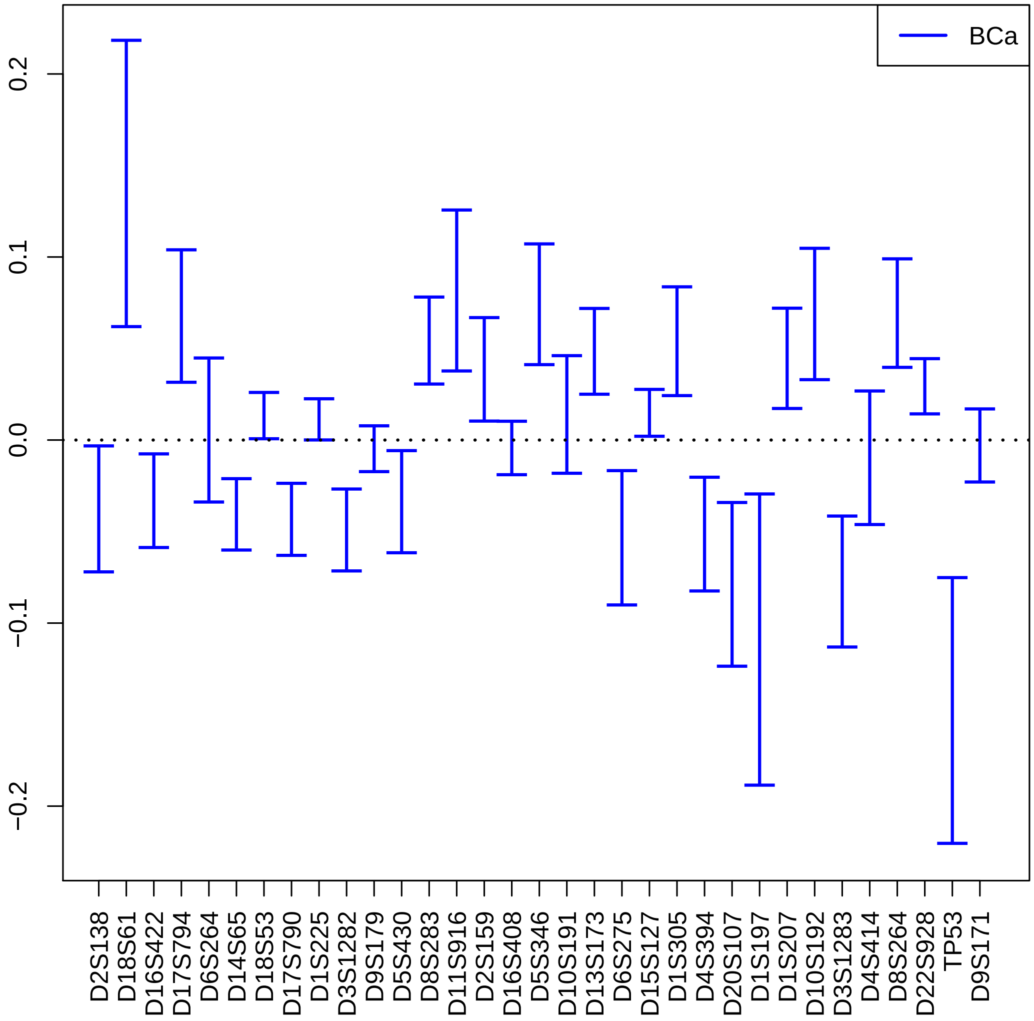

Figure 25: CI of the coefficients of the predictors, bootstrap $(\mathbf{y}, \mathbf{T})$, R=1000

```
## YT2   FALSE   TRUE    TRUE    TRUE   TRUE   TRUE  FALSE    TRUE  FALSE
## YT3    TRUE   TRUE    TRUE    TRUE  FALSE   TRUE   TRUE    TRUE   TRUE
## YT4    TRUE   TRUE   FALSE    TRUE  FALSE   TRUE   TRUE    TRUE   TRUE
## YT5    TRUE   TRUE    TRUE    TRUE  FALSE   TRUE  FALSE    TRUE  FALSE
## YT6    TRUE   TRUE    TRUE    TRUE  FALSE  FALSE  FALSE    TRUE  FALSE
## YT7    TRUE   TRUE   FALSE    TRUE  FALSE  FALSE  FALSE    TRUE  FALSE
## YT8    TRUE   TRUE   FALSE   FALSE  FALSE  FALSE  FALSE    TRUE  FALSE
## YT9    TRUE   TRUE   FALSE   FALSE  FALSE  FALSE  FALSE   FALSE  FALSE
## YT10   TRUE   TRUE   FALSE   FALSE  FALSE  FALSE  FALSE   FALSE  FALSE
##      D3S1282 D9S179 D5S430 D8S283 D11S916 D2S159 D16S408 D5S346 D10S191
## YT1     TRUE   TRUE   TRUE   TRUE    TRUE   TRUE    TRUE   TRUE    TRUE
## YT2     TRUE  FALSE  FALSE   TRUE    TRUE   TRUE    TRUE   TRUE    TRUE
## YT3     TRUE  FALSE   TRUE   TRUE    TRUE   TRUE   FALSE   TRUE   FALSE
## YT4     TRUE  FALSE   TRUE   TRUE    TRUE   TRUE   FALSE   TRUE   FALSE
## YT5     TRUE  FALSE   TRUE   TRUE    TRUE   TRUE   FALSE   TRUE   FALSE
## YT6     TRUE  FALSE   TRUE   TRUE   FALSE  FALSE   FALSE   TRUE   FALSE
## YT7     TRUE  FALSE   TRUE   TRUE   FALSE  FALSE   FALSE   TRUE   FALSE
## YT8     TRUE  FALSE  FALSE  FALSE   FALSE  FALSE   FALSE   TRUE   FALSE
## YT9    FALSE  FALSE  FALSE  FALSE   FALSE  FALSE   FALSE   TRUE   FALSE
## YT10   FALSE  FALSE  FALSE  FALSE   FALSE  FALSE   FALSE   TRUE   FALSE
##      D13S173 D6S275 D15S127 D1S305 D4S394 D20S107 D1S197 D1S207 D10S192
## YT1     TRUE   TRUE    TRUE   TRUE   TRUE    TRUE   TRUE   TRUE    TRUE
## YT2     TRUE   TRUE    TRUE   TRUE   TRUE    TRUE   TRUE   TRUE    TRUE
## YT3     TRUE   TRUE    TRUE   TRUE   TRUE    TRUE   TRUE   TRUE    TRUE
## YT4     TRUE   TRUE   FALSE   TRUE   TRUE    TRUE   TRUE   TRUE    TRUE
## YT5     TRUE   TRUE   FALSE   TRUE   TRUE    TRUE   TRUE   TRUE    TRUE
## YT6    FALSE   TRUE   FALSE   TRUE   TRUE    TRUE   TRUE   TRUE    TRUE
## YT7    FALSE   TRUE   FALSE   TRUE   TRUE    TRUE   TRUE   TRUE    TRUE
## YT8    FALSE   TRUE   FALSE   TRUE   TRUE    TRUE   TRUE   TRUE    TRUE
## YT9    FALSE   TRUE   FALSE   TRUE  FALSE    TRUE   TRUE   TRUE    TRUE
## YT10   FALSE  FALSE   FALSE   TRUE  FALSE    TRUE   TRUE   TRUE   FALSE
##      D3S1283 D4S414 D8S264 D22S928 TP53 D9S171
## YT1     TRUE   TRUE   TRUE    TRUE TRUE   TRUE
## YT2     TRUE   TRUE   TRUE    TRUE TRUE   TRUE
## YT3     TRUE  FALSE   TRUE    TRUE TRUE  FALSE
## YT4     TRUE  FALSE   TRUE   FALSE TRUE  FALSE
## YT5     TRUE  FALSE   TRUE   FALSE TRUE  FALSE
## YT6    FALSE  FALSE   TRUE   FALSE TRUE  FALSE
## YT7    FALSE  FALSE   TRUE   FALSE TRUE  FALSE
## YT8    FALSE  FALSE   TRUE   FALSE TRUE  FALSE
## YT9    FALSE  FALSE   TRUE   FALSE TRUE  FALSE
## YT10   FALSE  FALSE   TRUE   FALSE TRUE  FALSE
```

```
pi.e=prop.table(res.cv.modpls_compl$CVMC)%*%matind
pi.e
```

```
##      D2S138 D18S61 D16S422 D17S794 D6S264 D14S65 D18S53 D17S790 D1S225
## [1,]   0.92      1     0.8    0.87   0.37   0.78   0.68    0.93   0.68
##      D3S1282 D9S179 D5S430 D8S283 D11S916 D2S159 D16S408 D5S346 D10S191
## [1,]    0.93   0.29   0.79   0.87    0.78   0.78    0.37      1    0.37
##      D13S173 D6S275 D15S127 D1S305 D4S394 D20S107 D1S197 D1S207 D10S192
## [1,]    0.78   0.96    0.74      1   0.93       1      1      1    0.96
##      D3S1283 D4S414 D8S264 D22S928 TP53 D9S171
## [1,]    0.78   0.37      1    0.74    1   0.37
```

```
signpred(t(matind),labsize=2, plotsize = 12)
text(1:(ncol(matind))-.5,-1,pi.e,cex=.5)
mtext(expression(pi[e]),side=2,las=1,line=2,at=-1)
```

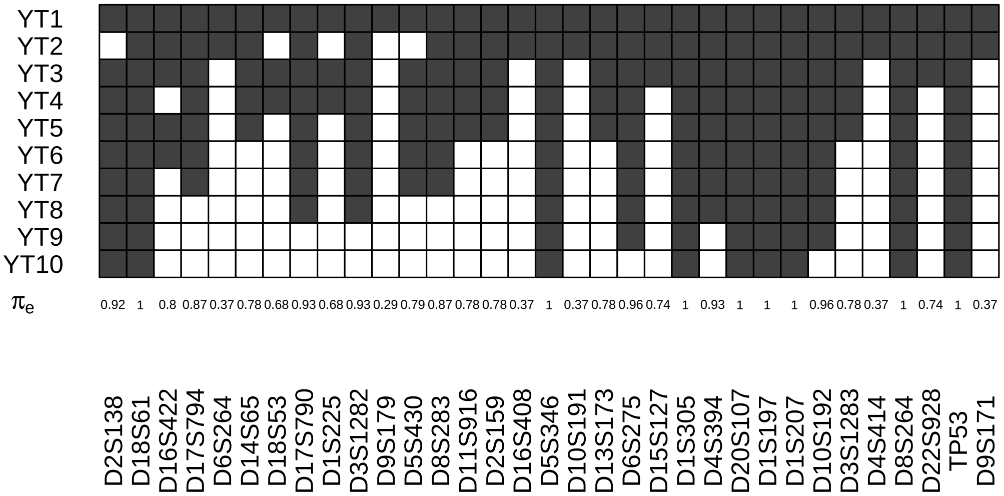

Figure 26: Significance of the predictors vs nbr of components, bootstrap $(\mathbf{y}, \mathbf{T})$, R=1000

3.4 PLS regression bis: Pine caterpillar

Cross-validation

```
rm(list = ls())
library(plsRglm)
data(pine)
Xpine<-pine[,1:10]
ypine<-pine[,11]
```

We use $k = 5$ unbalanced groups of 5 to 6 subjects to perform repeated k-fold cross validation. We set to 10, thanks to the option `nt=10`, the maximal number of components for the cross-validation function `-cv.plsR-` since the rank of the design matrix is equal to 10.

```
cv.modpls<-cv.plsR(ypine,Xpine,nt=10)
```

We sum up the results in a single table using the `summary`.

```
res.cv.modpls<-cvtable(summary(cv.modpls))

## ____************************************************____
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Component____ 5 ____
## ____Component____ 6 ____
## ____Component____ 7 ____
## ____Component____ 8 ____
## ____Component____ 9 ____
## ____Component____ 10 ____
## ____Predicting X without NA neither in X nor in Y____

## Loading required package:  plsdof
## Loading required package:  MASS

## ****________________________________________________****
##
##
## NK: 1
##
## CV Q2 criterion:
## 0 1
## 0 1
##
## CV Press criterion:
## 1 2 3 4 5 6 7
## 0 0 0 0 0 0 1
```

You can perform leave one out cross validation similar to the one that existed in previous versions of SIMCA by setting `TypeVC="standard"` as well the number of significant predictors per components Bastien et al. (2005).

```
res1<-plsR(ypine,Xpine, nt=10, typeVC="standard", pvals.expli=TRUE)

## ____************************************************____
## ____TypeVC____ standard ____
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Component____ 5 ____
## ____Component____ 6 ____
## ____Component____ 7 ____
## ____Component____ 8 ____
## ____Component____ 9 ____
## ____Component____ 10 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****
```

```
colSums(res1$pvalstep)
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0
```

```
res1$InfCrit
```

```
##              AIC  Q2cum_Y LimQ2_Y     Q2_Y PRESS_Y  RSS_Y   R2_Y
## Nb_Comp_0  82.42       NA      NA       NA      NA 20.800     NA
## Nb_Comp_1  63.62  0.38249  0.0975  0.38249  12.844 11.075 0.4676
## Nb_Comp_2  58.48  0.34836  0.0975 -0.05526  11.687  8.919 0.5712
## Nb_Comp_3  56.55  0.23688  0.0975 -0.17108  10.445  7.920 0.6192
## Nb_Comp_4  54.35  0.07000  0.0975 -0.21869   9.652  6.973 0.6648
## Nb_Comp_5  56.00 -0.07691  0.0975 -0.15796   8.074  6.899 0.6683
## Nb_Comp_6  57.70 -0.19969  0.0975 -0.11401   7.685  6.836 0.6714
## Nb_Comp_7  59.38 -0.27722  0.0975 -0.06463   7.277  6.770 0.6745
## Nb_Comp_8  61.21 -0.30603  0.0975 -0.02255   6.923  6.736 0.6762
## Nb_Comp_9  63.18 -0.39920  0.0975 -0.07134   7.217  6.730 0.6764
## Nb_Comp_10 65.16 -0.43744  0.0975 -0.02733   6.914  6.725 0.6767
##            R2_residY RSS_residY PRESS_residY Q2_residY  LimQ2
## Nb_Comp_0         NA      32.00           NA        NA     NA
## Nb_Comp_1     0.4676      17.04        19.76   0.38249 0.0975
## Nb_Comp_2     0.5712      13.72        17.98  -0.05526 0.0975
## Nb_Comp_3     0.6192      12.18        16.07  -0.17108 0.0975
## Nb_Comp_4     0.6648      10.73        14.85  -0.21869 0.0975
## Nb_Comp_5     0.6683      10.61        12.42  -0.15796 0.0975
## Nb_Comp_6     0.6714      10.52        11.82  -0.11401 0.0975
## Nb_Comp_7     0.6745      10.42        11.20  -0.06463 0.0975
## Nb_Comp_8     0.6762      10.36        10.65  -0.02255 0.0975
## Nb_Comp_9     0.6764      10.35        11.10  -0.07134 0.0975
## Nb_Comp_10    0.6767      10.35        10.64  -0.02733 0.0975
##            Q2cum_residY AIC.std DoF.dof sigmahat.dof AIC.dof BIC.dof
## Nb_Comp_0            NA   96.63   1.000       0.8062  0.6697  0.6992
## Nb_Comp_1       0.38249   77.83   3.176       0.5994  0.4048  0.4565
## Nb_Comp_2       0.34836   72.69   7.134       0.5762  0.4138  0.5212
## Nb_Comp_3       0.23688   70.77   8.778       0.5604  0.4071  0.5321
## Nb_Comp_4       0.07000   68.57   8.428       0.5222  0.3506  0.4548
## Nb_Comp_5      -0.07691   70.21   9.308       0.5286  0.3667  0.4846
## Nb_Comp_6      -0.19969   71.91   9.292       0.5260  0.3629  0.4795
## Nb_Comp_7      -0.27722   73.60   9.756       0.5285  0.3703  0.4938
## Nb_Comp_8      -0.30603   75.43  10.364       0.5338  0.3831  0.5171
## Nb_Comp_9      -0.39920   77.40  10.732       0.5378  0.3921  0.5329
## Nb_Comp_10     -0.43744   79.38  11.000       0.5407  0.3987  0.5446
##            GMDL.dof DoF.naive sigmahat.naive AIC.naive BIC.naive
## Nb_Comp_0    -3.605         1         0.8062    0.6697    0.6992
## Nb_Comp_1    -9.875         2         0.5977    0.3789    0.4113
## Nb_Comp_2    -6.986         3         0.5453    0.3243    0.3648
## Nb_Comp_3    -6.261         4         0.5226    0.3062    0.3557
## Nb_Comp_4    -8.153         5         0.4990    0.2867    0.3432
## Nb_Comp_5    -7.112         6         0.5055    0.3020    0.3715
## Nb_Comp_6    -7.233         7         0.5127    0.3187    0.4021
## Nb_Comp_7    -6.742         8         0.5204    0.3365    0.4347
## Nb_Comp_8    -6.038         9         0.5298    0.3572    0.4718
## Nb_Comp_9    -5.600        10         0.5410    0.3813    0.5140
## Nb_Comp_10   -5.288        11         0.5529    0.4076    0.5601
##            GMDL.naive
## Nb_Comp_0      -3.605
## Nb_Comp_1     -11.451
## Nb_Comp_2     -12.823
## Nb_Comp_3     -12.757
## Nb_Comp_4     -12.812
## Nb_Comp_5     -11.330
## Nb_Comp_6      -9.919
## Nb_Comp_7      -8.593
## Nb_Comp_8      -7.288
## Nb_Comp_9      -6.009
## Nb_Comp_10     -4.799
```

The number of significant predictors within each component tell us to only build 0 components when the AIC criteria gives us 4 components and the BIC concludes to 4 components. The cross-validated Q^2_{cum} criterion advocates for retaining 1 components either for leave one out and 1 for 5-fold CV. The 5-fold CV cross-validation was run 100 times by randomly creating groups. Here are the command lines:

```
set.seed(123)
cv.modpls<-cv.plsR(x11~.,data=pine,nt=10,NK=100)
```

```
res.cv.modpls=cvtable(summary(cv.modpls))
## ____************************************************____
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Component____ 5 ____
## ____Component____ 6 ____
## ____Component____ 7 ____
## ____Component____ 8 ____
## ____Component____ 9 ____
## ____Component____ 10 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****
##
##
## NK: 1,  2,  3,  4,  5,  6,  7,  8,  9,  10
## NK: 11,  12,  13,  14,  15,  16,  17,  18,  19,  20
## NK: 21,  22,  23,  24,  25,  26,  27,  28,  29,  30
## NK: 31,  32,  33,  34,  35,  36,  37,  38,  39,  40
## NK: 41,  42,  43,  44,  45,  46,  47,  48,  49,  50
## NK: 51,  52,  53,  54,  55,  56,  57,  58,  59,  60
## NK: 61,  62,  63,  64,  65,  66,  67,  68,  69,  70
## NK: 71,  72,  73,  74,  75,  76,  77,  78,  79,  80
## NK: 81,  82,  83,  84,  85,  86,  87,  88,  89,  90
## NK: 91,  92,  93,  94,  95,  96,  97,  98,  99,  100
##
##
## CV Q2 criterion:
##   0   1
##   0 100
##
## CV Press criterion:
##  1  2  3  4  5  6  7  8  9
## 17  5  9 24 17 11 15  1  1
```

The results, based on the use of the Q^2 criterion, (Fig. 27) confirm those of the first 5-fold CV cross validation: we decide to retain 1 components. Even in the linear case, cross validation should be repeated to select the number of components in a PLSR model.

```
plot(res.cv.modpls)
```

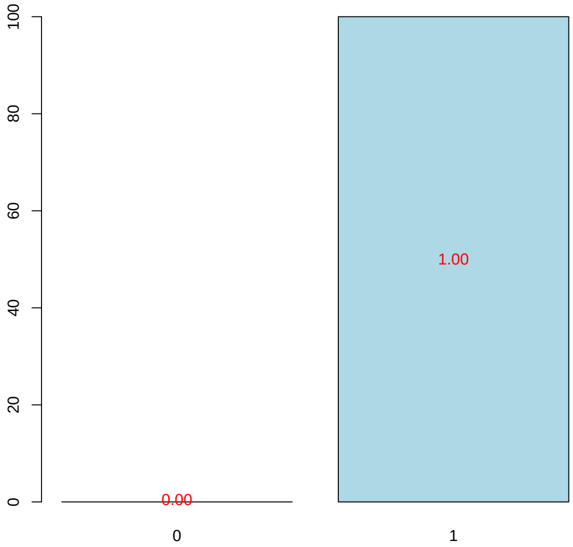

Figure 27: Nb components, 5-CV, n=100

Selected PLSR model for pine dataset.

```
res<-plsR(x11~.,data=pine,nt=1,pvals.expli=TRUE)
## ____************************************************____
## ____Component____ 1 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****
res
## Number of required components:
## [1] 1
## Number of successfully computed components:
## [1] 1
## Coefficients:
##                 [,1]
## Intercept  4.1382957
## x1        -0.0007545
## x2        -0.0114507
## x3        -0.0108577
## x4        -0.0631435
## x5        -0.0067332
## x6        -0.1459628
## x7        -0.2077726
## x8        -0.0430294
## x9        -0.2061096
## x10       -0.0887273
## Information criteria and Fit statistics:
##              AIC RSS_Y   R2_Y R2_residY RSS_residY AIC.std DoF.dof
## Nb_Comp_0 82.42 20.80     NA        NA      32.00   96.63   1.000
## Nb_Comp_1 63.62 11.07 0.4676    0.4676      17.04   77.83   3.176
```

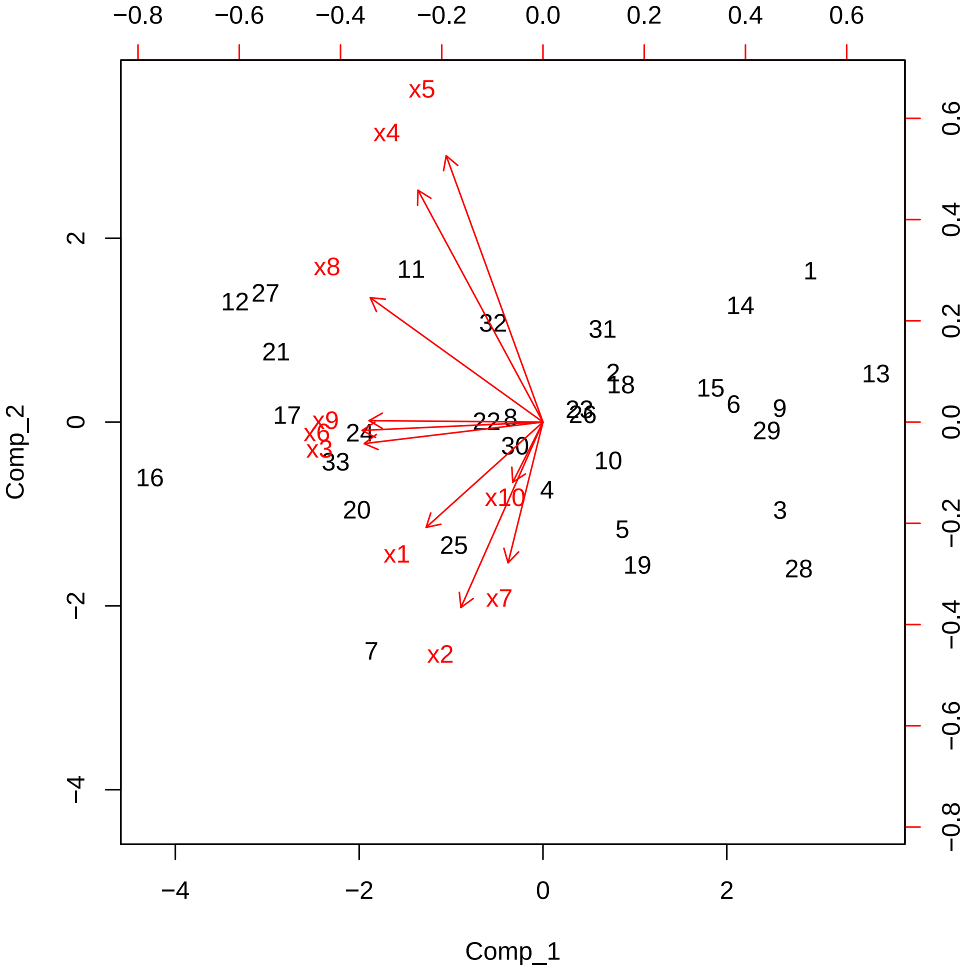

Figure 28: Biplot of the observations and the variables

```
##           sigmahat.dof AIC.dof BIC.dof GMDL.dof DoF.naive
## Nb_Comp_0       0.8062  0.6697  0.6992   -3.605         1
## Nb_Comp_1       0.5994  0.4048  0.4565   -9.875         2
##           sigmahat.naive AIC.naive BIC.naive GMDL.naive
## Nb_Comp_0         0.8062     0.6697    0.6992     -3.605
## Nb_Comp_1         0.5977     0.3789    0.4113    -11.451
```

It is also possible to display the biplot of the observations and the predictors (Figure 28).

```
biplot(res1$tt,res1$pp)
```

Bootstrap $(\mathbf{y}, \mathbf{X})$

Graphical results of the bootstrap on the (Y, X): distributions of the estimators (see Figure 29) and CI (see Figure 30).

```
set.seed(123)
Pine.bootYX1=bootpls(res,R=1000)
```

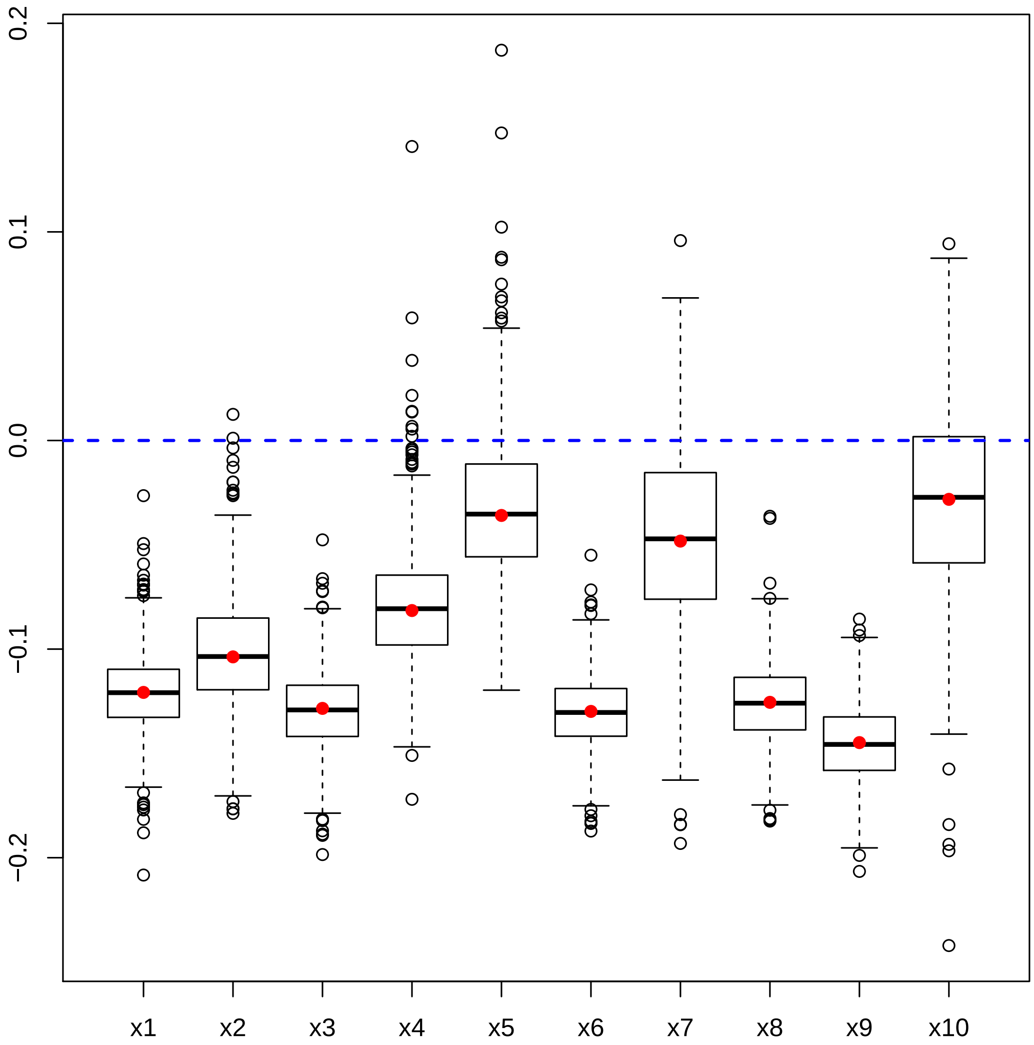

Figure 29: Bootstrap $(\mathbf{y}, \mathbf{X})$ distribution of the coefficients of the predictors, R=1000

We do not bootstrap the intercept since the boostrap is done with the centered and scaled response and predictors. As a consequence we should exclude it from the boxplots using the option `indice=2:8` and must exclude it from the CI computations, if we request BC$_a$ ones, again with the option `indice=2:8`.

```
boxplots.bootpls(Pine.bootYX1,indice=2:11)
```

```
temp.ci=confints.bootpls(Pine.bootYX1,indice=2:11)
plots.confints.bootpls(temp.ci,typeIC="BCa",colIC=c("blue","blue","blue","blue"),
                       legendpos ="topright")
```

Bootstrap is perfomed using the *boot* package. It allows the user to apply the functions, including `jack.after.boot` or `plot.boot` (Figure 30), of this package to the bootstrapped PLSR or PLSGLR models.

```
plot(Pine.bootYX1,index=2,jack=TRUE)
```

Using the `dataEllipse` of the *car* you can plot confidence ellipses for two parameters of the PLSR or PLSGLR models (Figure 32).

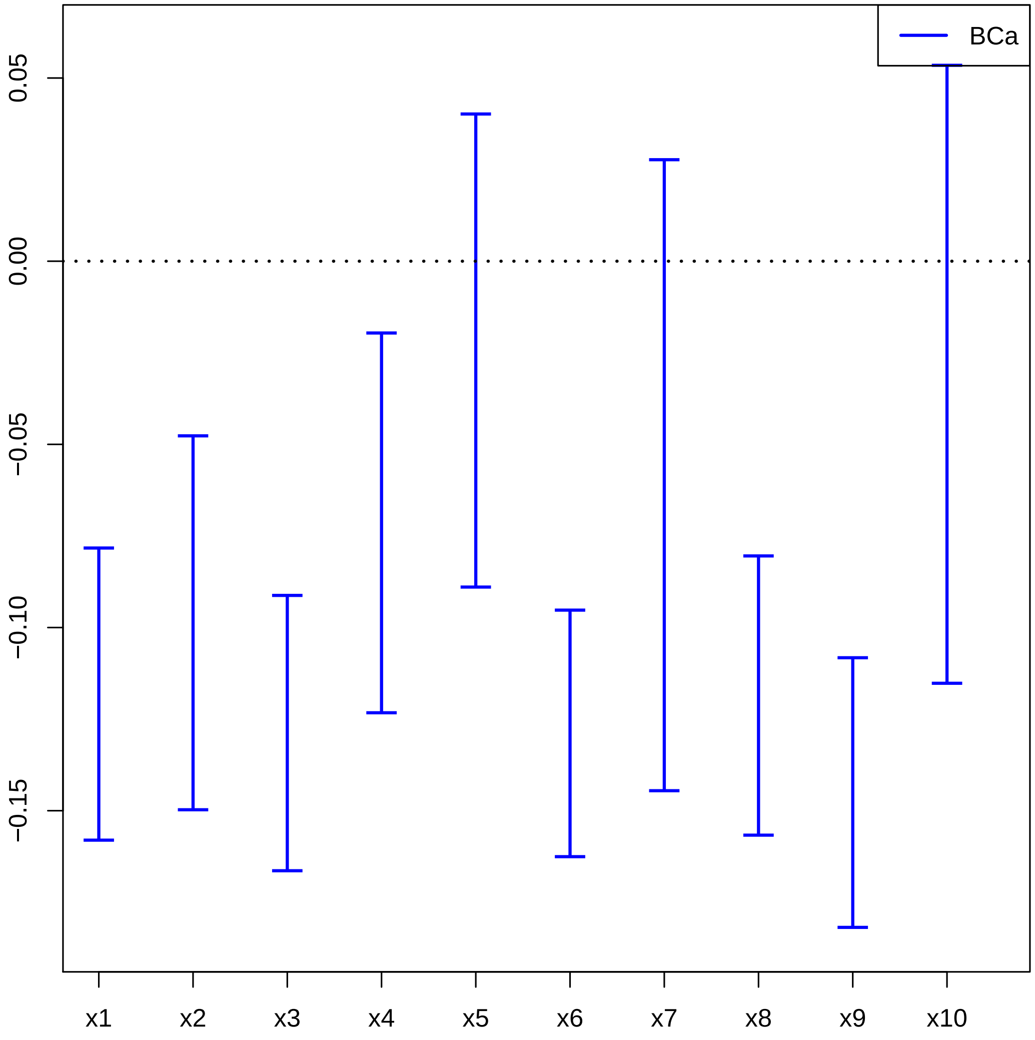

Figure 30: CI of the coefficients of the predictors, bootstrap $(\mathbf{y}, \mathbf{X})$, R=1000

```r
car::dataEllipse(Pine.bootYX1$t[,2], Pine.bootYX1$t[,3], cex=.3, levels=c(.5, .95, .99),
                 robust=T, xlab="X2", ylab="X3")
```

Bootstrap $(\mathbf{y}, \mathbf{T})$

Re-sampling on the couple (Y, T) (Bastien et al., 2005) is more stable and faster than the first one. We set at 1000 the number of re-sampling. CIs for each of the predictors (see Figure 34) and boxplots as well (see Figure 33).

```r
set.seed(123)
Pine.bootYT1=bootpls(res,typeboot="fmodel_np",R=1000)
```

```r
boxplots.bootpls(Pine.bootYT1,indices=2:11)
```

We do not bootstrap the intercept since the boostrap is done with the centered and scaled response and predictors. As a consequence we should exclude it from the boxplots using the option `indice=2:8` and must exclude it from the CI computations, if we request BC$_a$ ones, again with the option `indice=2:8`.

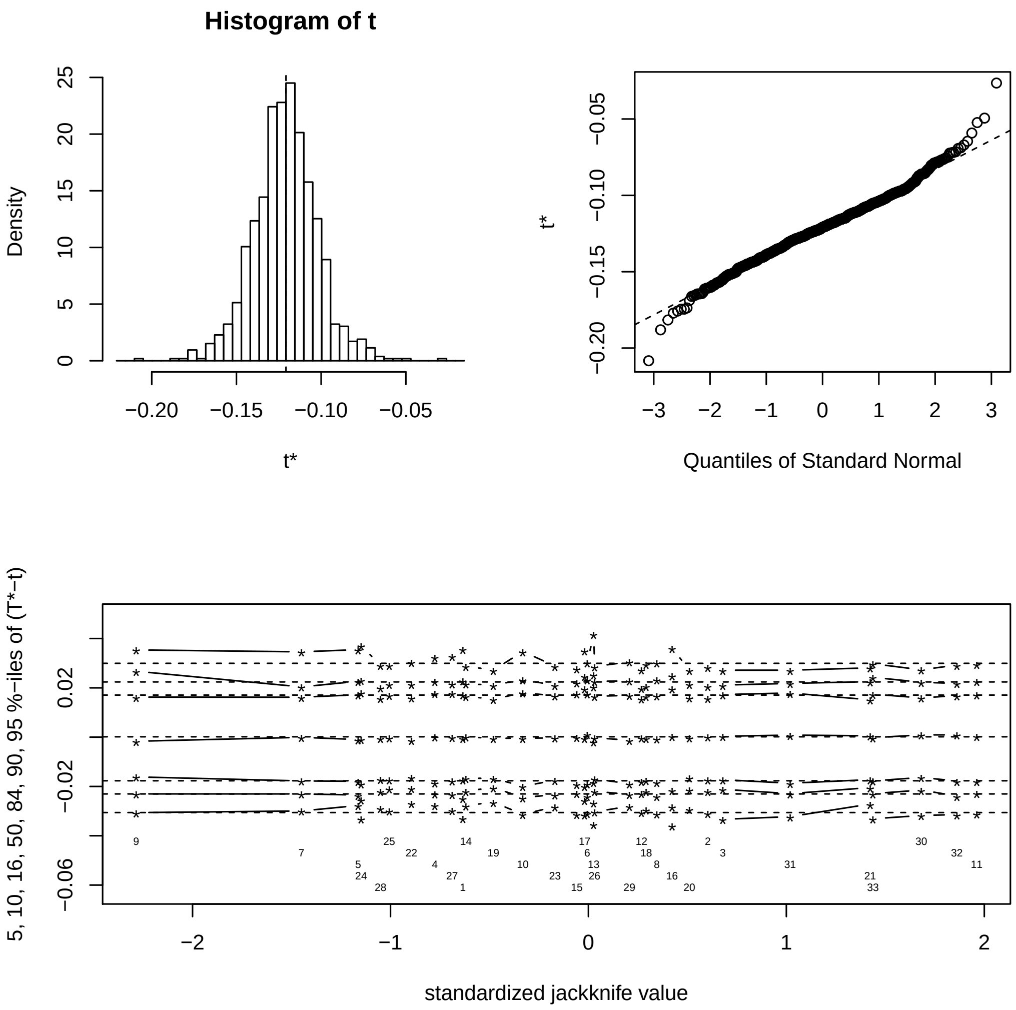

Figure 31: plot.boot, bootstrap $(\mathbf{y}, \mathbf{X})$, R=1000

```
temp.ci=confints.bootpls(Pine.bootYT1,indices=2:11)
plots.confints.bootpls(temp.ci,typeIC="BCa",colIC=c("blue","blue","blue","blue"),
                       legendpos ="topright")
```

```
ind.BCa.pineYT1 <- (temp.ci[,7]<0&temp.ci[,8]<0)|(temp.ci[,7]>0&temp.ci[,8]>0)
```

We display the significance of predictors using the `signpred` function.

```
(matind=(rbind(YT1=ind.BCa.pineYT1)))
##       x1   x2   x3   x4   x5   x6   x7   x8   x9  x10
## YT1 TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
pi.e=prop.table(res.cv.modpls$CVQ2)[-1]%*%matind
pi.e
##      x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
## [1,]  1  1  1  1  1  1  1  1  1   1
signpred(t(matind),labsize=.5, plotsize = 12)
text(1:(ncol(matind))-.5,-1,pi.e,cex=.75)
mtext(expression(pi[e]),side=2,las=1,line=2,at=-1)
```

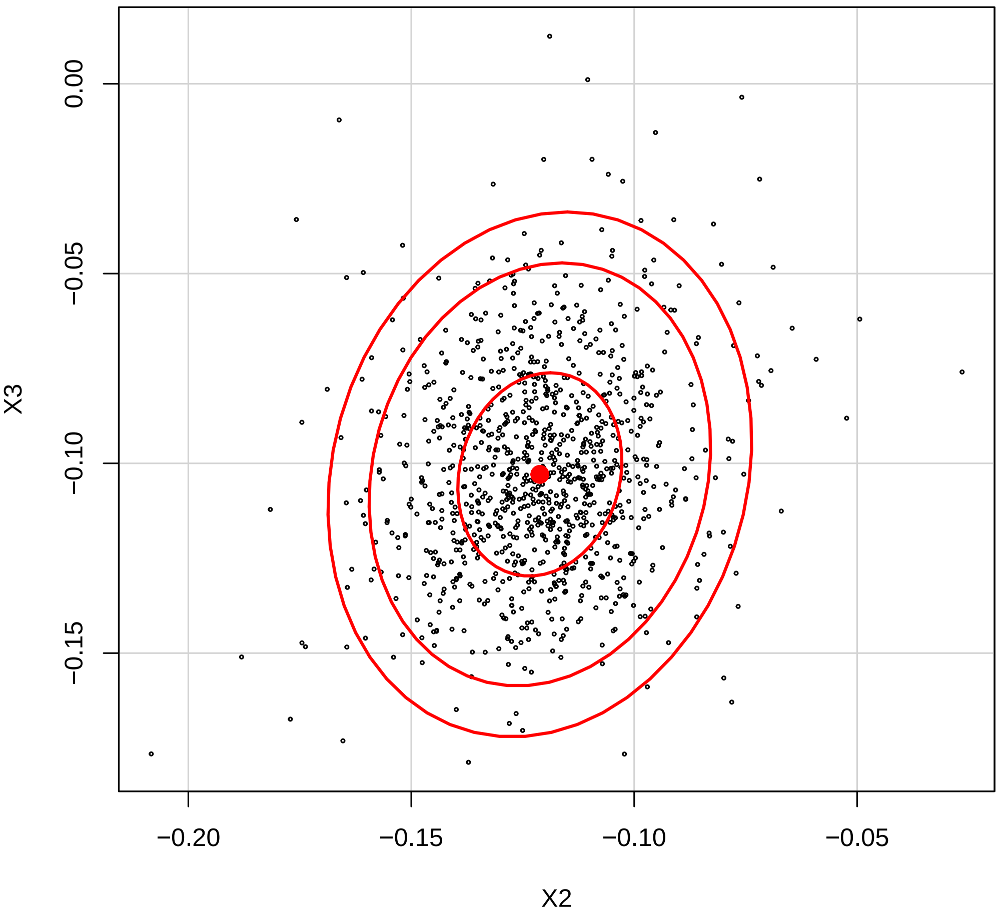

Figure 32: Confidence ellipse of the coefficients of the first two predictors, bootstrap $(\mathbf{y}, \mathbf{X})$, R=1000

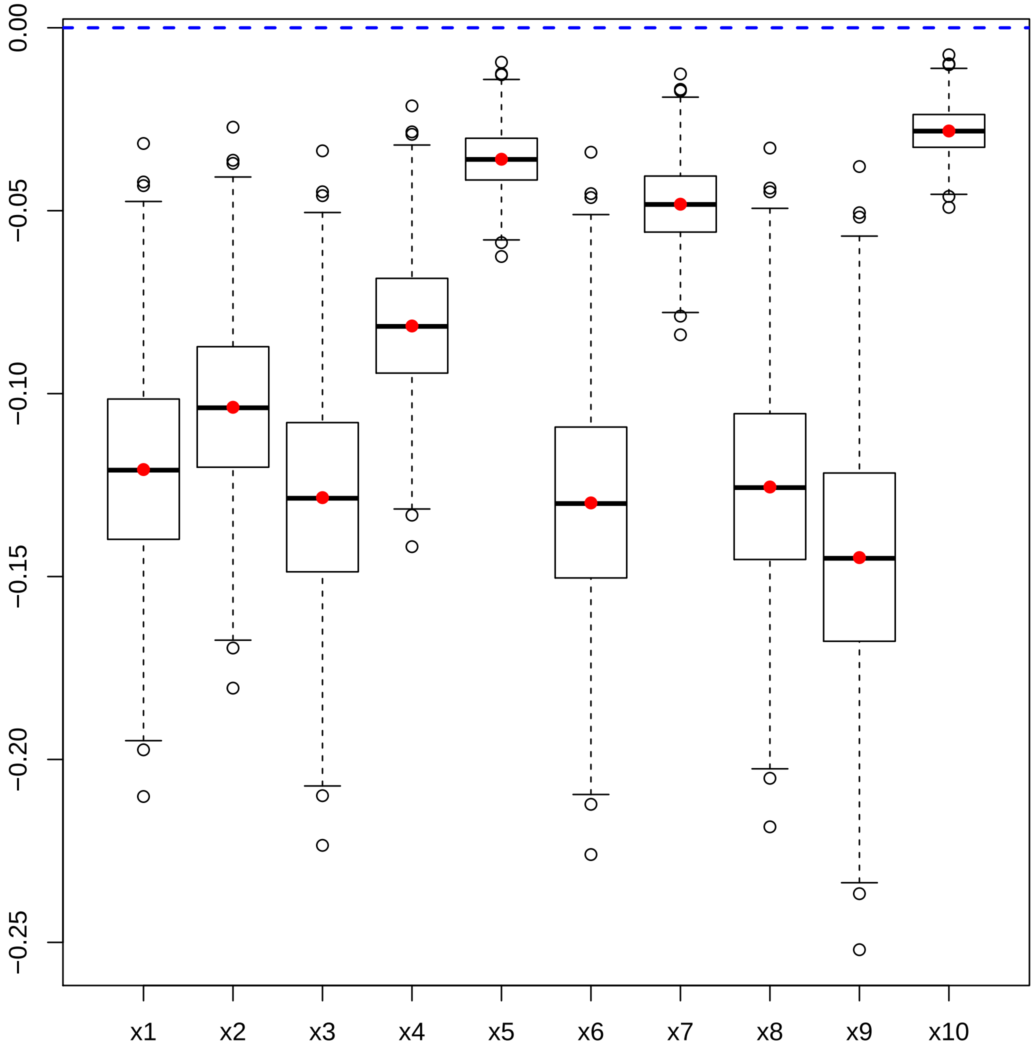

Figure 33: Bootstrap $(\mathbf{y}, \mathbf{T})$ distribution of the coefficients of the predictors, R=1000

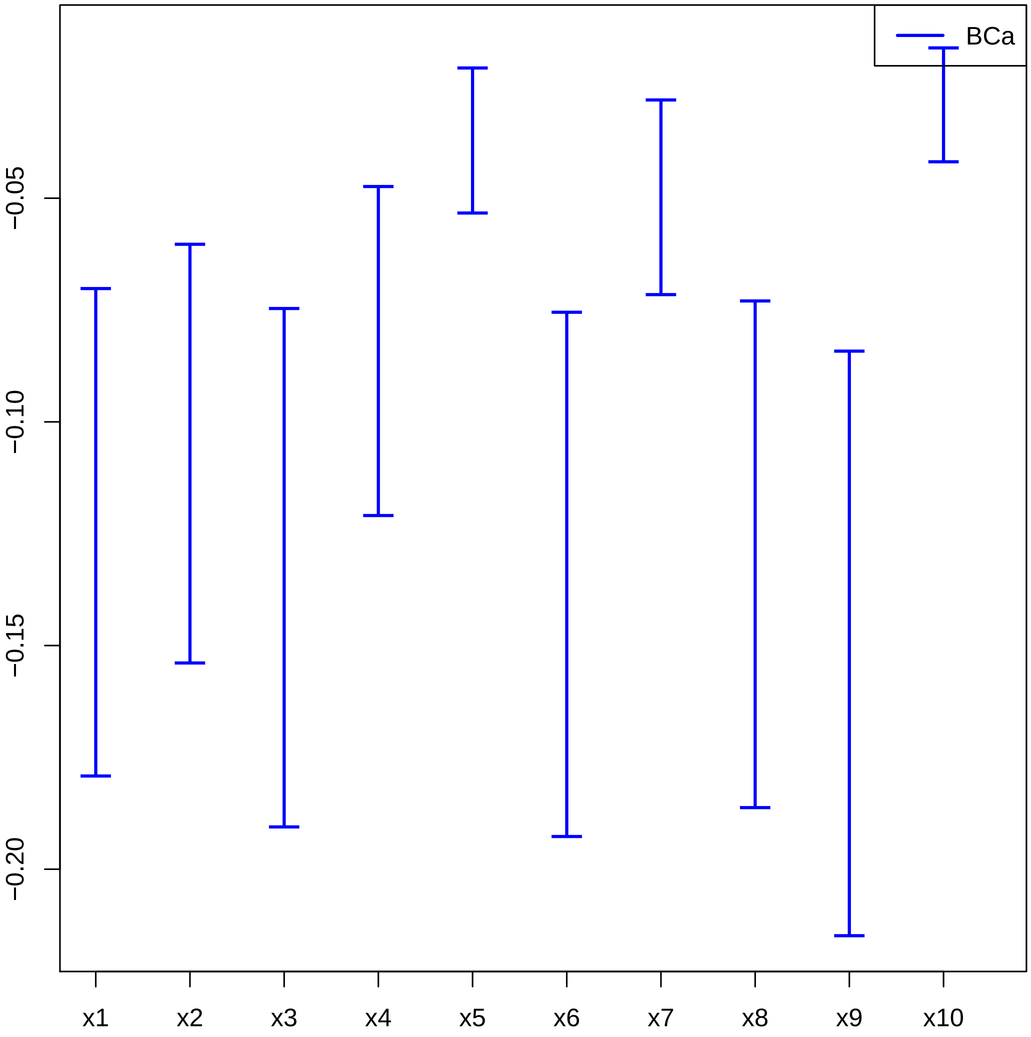

Figure 34: CI of the coefficients of the predictors, bootstrap $(\mathbf{y}, \mathbf{T})$, R=1000

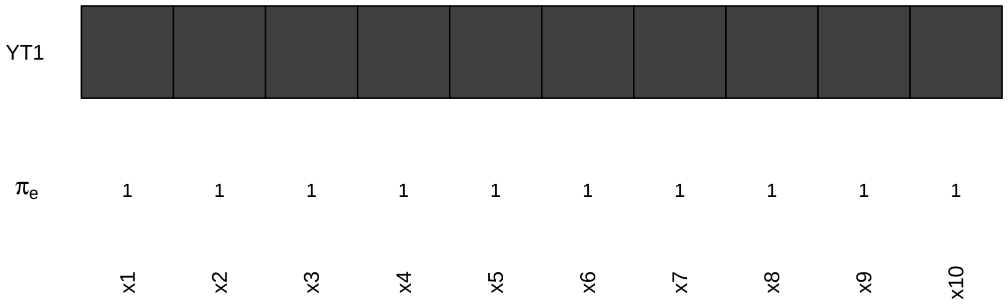

Figure 35: Significance of the predictors vs nbr of components, bootstrap $(\mathbf{y}, \mathbf{T})$, R=1000

3.5 PLS ordinal logistic regression: Bordeaux wine quality

Cross-validation

```
set.seed(12345)
data(bordeaux)
bordeaux$Quality<-factor(bordeaux$Quality,ordered=TRUE)
modpls1 <- plsRglm(Quality~.,data=bordeaux,4,modele="pls-glm-polr",pvals.expli=TRUE)

## ____************************************************____
##
## Model: pls-glm-polr
## Method: logistic
##
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Predicting X without NA neither in X or Y____
## ****________________________________________________****

modpls1

## Number of required components:
## [1] 4
## Number of successfully computed components:
## [1] 4
## Coefficients:
##                   [,1]
## 1|2         -85.50956
## 2|3         -80.55156
## Temperature   0.02427
## Sunshine      0.01379
## Heat         -0.08876
## Rain         -0.02590
## Information criteria and Fit statistics:
##             AIC   BIC Missclassed Chi2_Pearson_Y
## Nb_Comp_0 78.65 81.70          22         62.333
## Nb_Comp_1 36.50 41.08           6          9.357
## Nb_Comp_2 35.58 41.69           6          8.569
## Nb_Comp_3 36.27 43.90           7          8.281
## Nb_Comp_4 38.16 47.32           7          8.322
```

```
Xbordeaux<-bordeaux[,1:4]
ybordeaux<-bordeaux$Quality
modpls2 <- plsRglm(ybordeaux,Xbordeaux,4,modele="pls-glm-polr",pvals.expli=TRUE)

## ____************************************************____
##
## Model: pls-glm-polr
## Method: logistic
##
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****

modpls2

## Number of required components:
```

```
## [1] 4
## Number of successfully computed components:
## [1] 4
## Coefficients:
##                  [,1]
## 1|2         -85.50956
## 2|3         -80.55156
## Temperature   0.02427
## Sunshine      0.01379
## Heat         -0.08876
## Rain         -0.02590
## Information criteria and Fit statistics:
##             AIC   BIC Missclassed Chi2_Pearson_Y
## Nb_Comp_0 78.65 81.70          22         62.333
## Nb_Comp_1 36.50 41.08           6          9.357
## Nb_Comp_2 35.58 41.69           6          8.569
## Nb_Comp_3 36.27 43.90           7          8.281
## Nb_Comp_4 38.16 47.32           7          8.322
```

```
all(modpls1$InfCrit==modpls2$InfCrit)
```

```
## [1] TRUE
```

```
colSums(modpls2$pvalstep)
```

```
## temppvalstep temppvalstep temppvalstep temppvalstep
##            4            0            0            0
```

No discrepancy between formula specification (`formula` and `data`) and datasets (`dataY` and `dataX`) ones. Number of components to be retained:

- AIC → 2.
- BIC → 1.
- Non cross validated missclassed → 1.
- Non significant predictor criterion → 1.

```
set.seed(123)
cv.modpls<-cv.plsRglm(ybordeaux,Xbordeaux,nt=4,modele="pls-glm-polr",NK=100)
```

```
res.cv.modpls=cvtable(summary(cv.modpls, MClassed = TRUE))
```

```
## ____************************************************____
##
## Model: pls-glm-polr
## Method: logistic
##
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****
##
##
## NK: 1,  2,  3,  4,  5,  6,  7,  8,  9,  10
## NK: 11,  12,  13,  14,  15,  16,  17,  18,  19,  20
## NK: 21,  22,  23,  24,  25,  26,  27,  28,  29,  30
## NK: 31,  32,  33,  34,  35,  36,  37,  38,  39,  40
## NK: 41,  42,  43,  44,  45,  46,  47,  48,  49,  50
## NK: 51,  52,  53,  54,  55,  56,  57,  58,  59,  60
## NK: 61,  62,  63,  64,  65,  66,  67,  68,  69,  70
## NK: 71,  72,  73,  74,  75,  76,  77,  78,  79,  80
## NK: 81,  82,  83,  84,  85,  86,  87,  88,  89,  90
## NK: 91,  92,  93,  94,  95,  96,  97,  98,  99,  100
## CV MissClassed criterion:
##  1  2  3  4
## 84  7  7  2
##
## CV Q2Chi2 criterion:
##  0  1
## 98  2
##
## CV PreChi2 criterion:
##  1  2  3
## 24 73  3
```

According to the results of the cross validation procedure (Fig. 36), we retain a single component, which was also, by chance on this dataset, the BIC and raw cross-validation choices.

```
plot(res.cv.modpls)
```

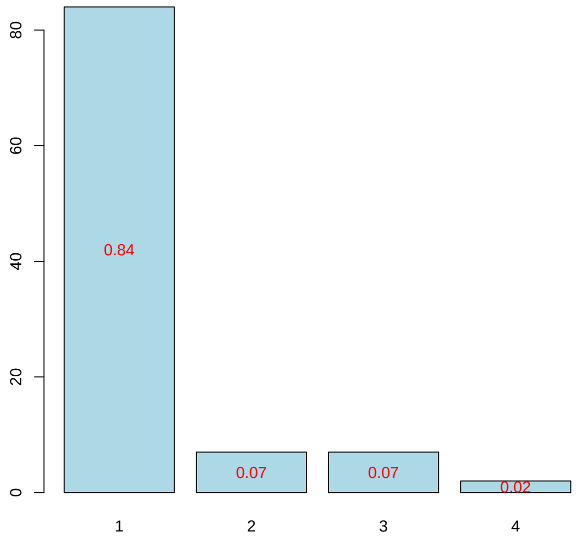

Figure 36: Nb components, 5-CV, n=100

Retained model according to cross validated missclassed criterion.

```
res<-plsRglm(ybordeaux,Xbordeaux,1,modele="pls-glm-polr")
## ____************************************************____
##
## Model: pls-glm-polr
## Method: logistic
##
## ____Component____ 1 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****
```

It is also possible to display the biplot of the observations and the predictors (Figure 37).

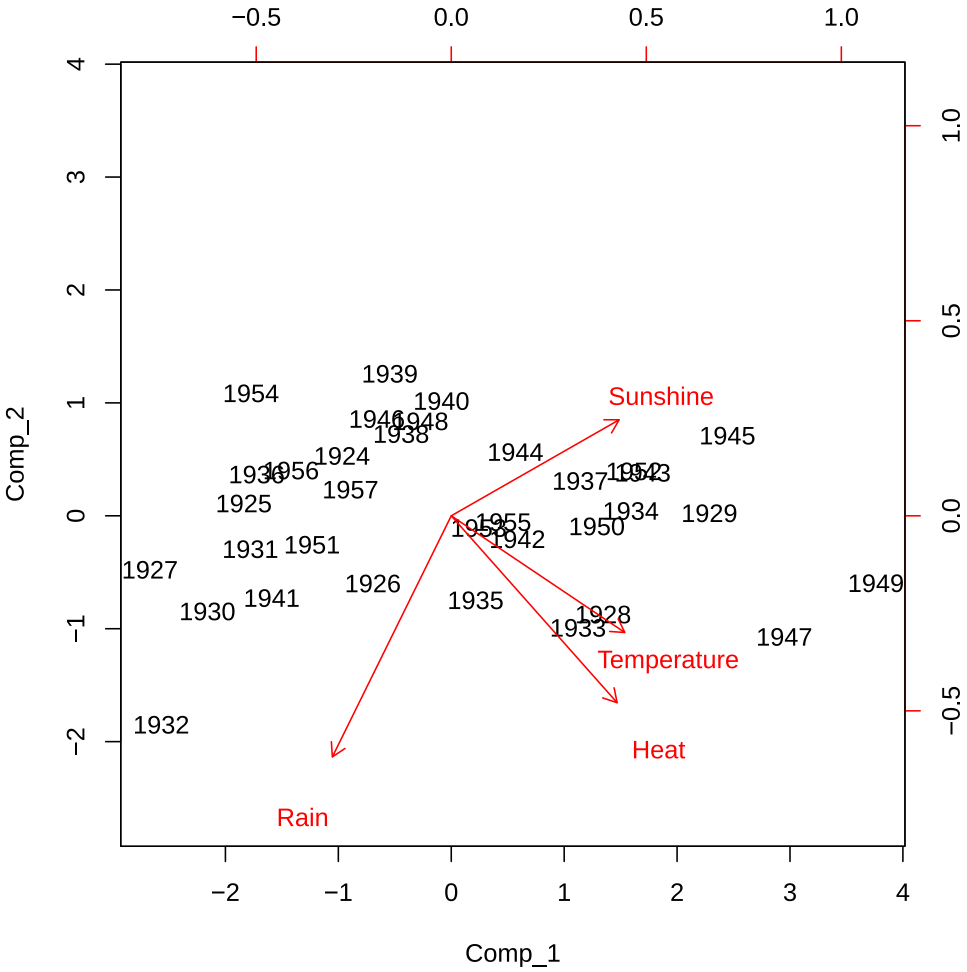

Figure 37: Biplot of the observations and the variables

```
biplot(modpls1$tt,modpls1$pp)
```

Application of the PLSGLR ordinal regression to an incomplete dataset.

```
XbordeauxNA<-Xbordeaux
XbordeauxNA[1,1] <- NA
modplsNA <- plsRglm(ybordeaux,XbordeauxNA,4,modele="pls-glm-polr")
## ____************************************************____
## Only naive DoF can be used with missing data
##
## Model: pls-glm-polr
## Method: logistic
##
## ____There are some NAs in X but not in Y____
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## Warning : reciprocal condition number of t(cbind(res$pp,temppp)[XXNA[1,],,drop=FALSE])%*%cbind(res$pp,temppp)[
## Warning only 3 components could thus be extracted
## ____Predicting X with NA in X and not in Y____
## ****________________________________________________****
```

```
modplsNA
```

```
## Number of required components:
## [1] 4
## Number of successfully computed components:
## [1] 3
## Coefficients:
##                  [,1]
## 1|2          -89.16630
## 2|3          -84.11693
## Temperature    0.02461
## Sunshine       0.01535
## Heat          -0.09543
## Rain          -0.02399
## Information criteria and Fit statistics:
##             AIC   BIC Missclassed Chi2_Pearson_Y
## Nb_Comp_0 78.65 81.70          22         62.333
## Nb_Comp_1 36.21 40.79           6          9.454
## Nb_Comp_2 35.30 41.40           5          8.235
## Nb_Comp_3 35.82 43.45           7          7.803
```

```
data.frame(formula=modpls1$Coeffs,datasets=modpls2$Coeffs,datasetsNA=modplsNA$Coeffs)
```

```
##                formula  datasets datasetsNA
## 1|2          -85.50956 -85.50956  -89.16630
## 2|3          -80.55156 -80.55156  -84.11693
## Temperature    0.02427   0.02427    0.02461
## Sunshine       0.01379   0.01379    0.01535
## Heat          -0.08876  -0.08876   -0.09543
## Rain          -0.02590  -0.02590   -0.02399
```

Bootstrap $(\mathbf{y}, \mathbf{X})$

CIs for each of the predictors (see Figure 39) and boxplots as well (see Figure 38) for ordinary balanced bootstrap.

```
bordeaux.bootYX1<- bootplsglm(res, typeboot = "plsmodel", sim="balanced", R=1000)
```

```
boxplots.bootpls(bordeaux.bootYX1)
```

```
temp.ci=confints.bootpls(bordeaux.bootYX1)
plots.confints.bootpls(temp.ci,typeIC="BCa",colIC=c("blue","blue","blue","blue"),
                       legendpos ="topright")
```

The `strata` option is an integer vector or factor specifying the strata for multi-sample problems. It ensures that, for a nonparametric bootstrap, the resampling are done within each of the specified strata. In our case it improves the results of the bootstrap as can been seen on the CIs for each of the predictors (see Figure 39) and boxplots as well (see Figure 38).

```
bordeaux.bootYX1strata<- bootplsglm(res,typeboot = "plsmodel", sim="balanced",
                                    R=1000, strata=unclass(ybordeaux))
```

```
boxplots.bootpls(bordeaux.bootYX1strata)
```

```
confints.bootpls(bordeaux.bootYX1strata)
```

```
##
## 1|2          -4.6062 1.1012 -3.0109 1.171 -5.7018 -1.5195 -4.5549
## 2|3          -1.1715 4.7396 -1.5958 3.227  1.3705  6.1936  1.2746
```

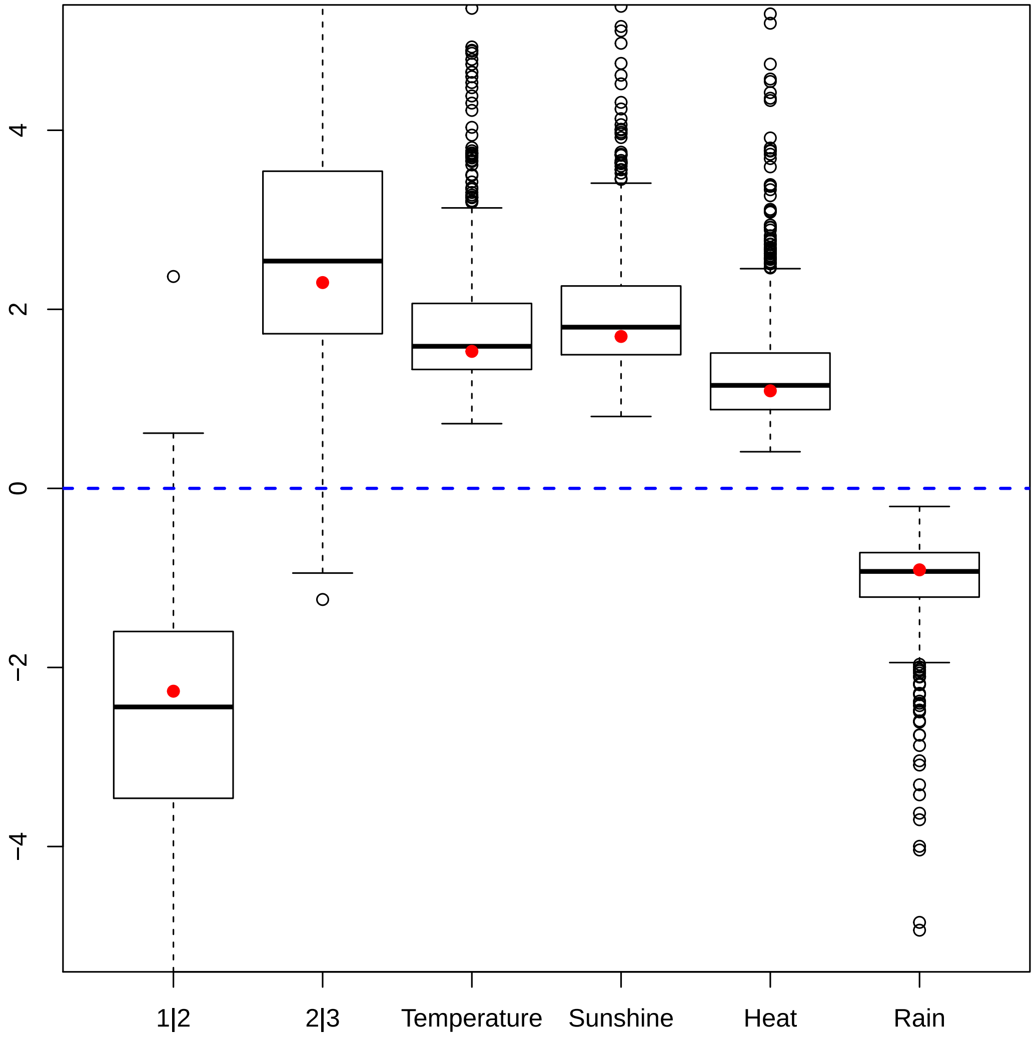

Figure 38: Bootstrap $(\mathbf{y}, \mathbf{T})$ distribution of the coefficients of the predictors, R=1000

```
## Temperature -0.8713 3.2450 -1.3148 2.061  0.9988  4.3751  0.9127
## Sunshine    -0.2018 3.0132 -0.3663 2.313  1.0806  3.7602  0.9906
## Heat        -0.7934 2.4686 -0.9826 1.602  0.5770  3.1618  0.5106
## Rain        -1.9133 0.3824 -1.3915 0.674 -2.4934 -0.4279 -2.2640
##
## 1|2         -1.3808
## 2|3          4.7353
## Temperature  3.0436
## Sunshine     3.0642
## Heat         2.5470
## Rain        -0.3995
## attr(,"typeBCa")
## [1] TRUE
plots.confints.bootpls(temp.ci,typeIC="BCa",colIC=c("blue","blue","blue","blue"),
                       legendpos ="topright")
```

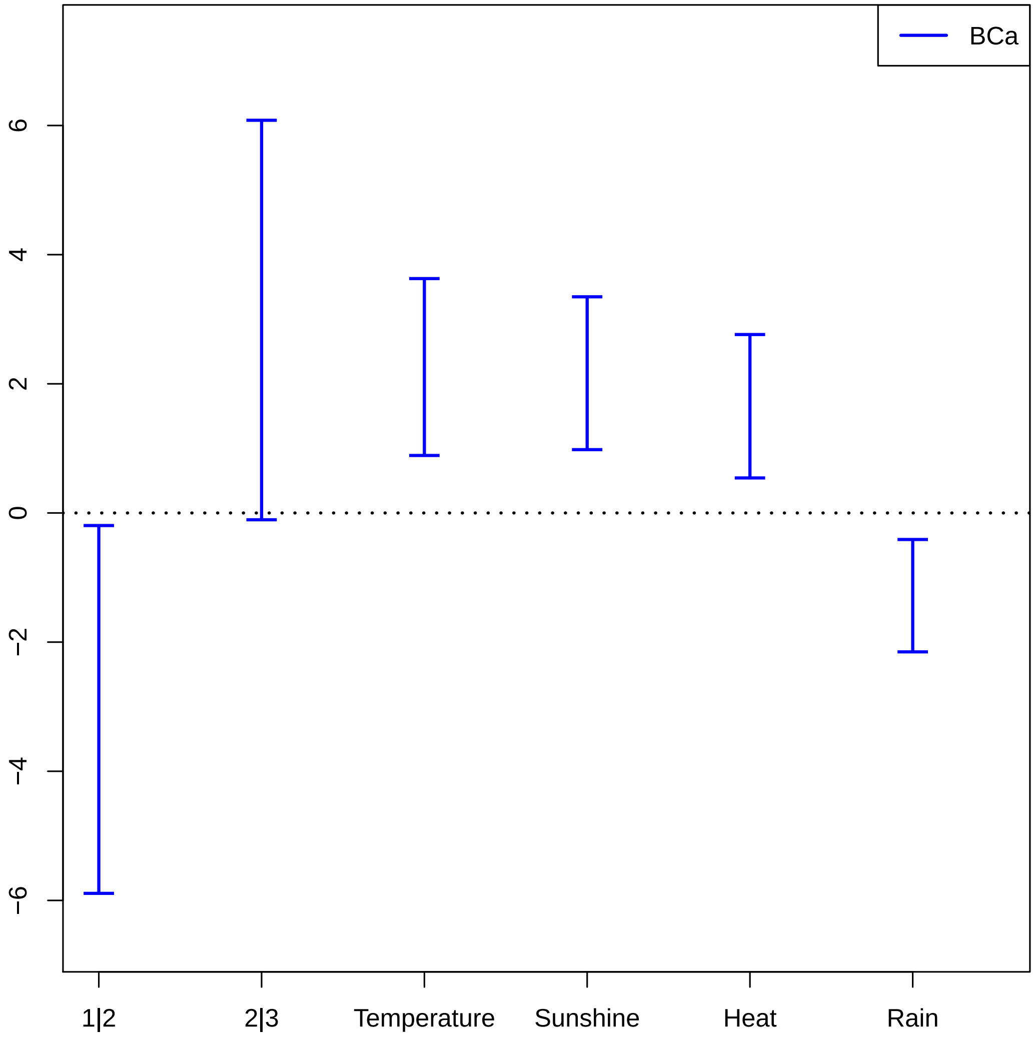

Figure 39: CI of the coefficients of the predictors, bootstrap $(\mathbf{y}, \mathbf{T})$, R=1000

Bootstrap $(\mathbf{y}, \mathbf{T})$

CIs for each of the predictors (see Figure 43) and boxplots as well (see Figure 42) for ordinary balanced bootstrap.

```
bordeaux.bootYT1<- bootplsglm(res,sim="balanced", R=1000)
```

```
boxplots.bootpls(bordeaux.bootYT1)
```

```
temp.ci=confints.bootpls(bordeaux.bootYT1)
plots.confints.bootpls(temp.ci,typeIC="BCa",colIC=c("blue","blue","blue","blue"),
                       legendpos ="topright")
```

Again the strata option improves the results of the bootstrap as can been seen on the CIs for each of the predictors (see Figure 45) and boxplots as well (see Figure 44).

```
bordeaux.bootYT1strata<- bootplsglm(res, sim="balanced", R=1000,
                                    strata=unclass(ybordeaux))
```

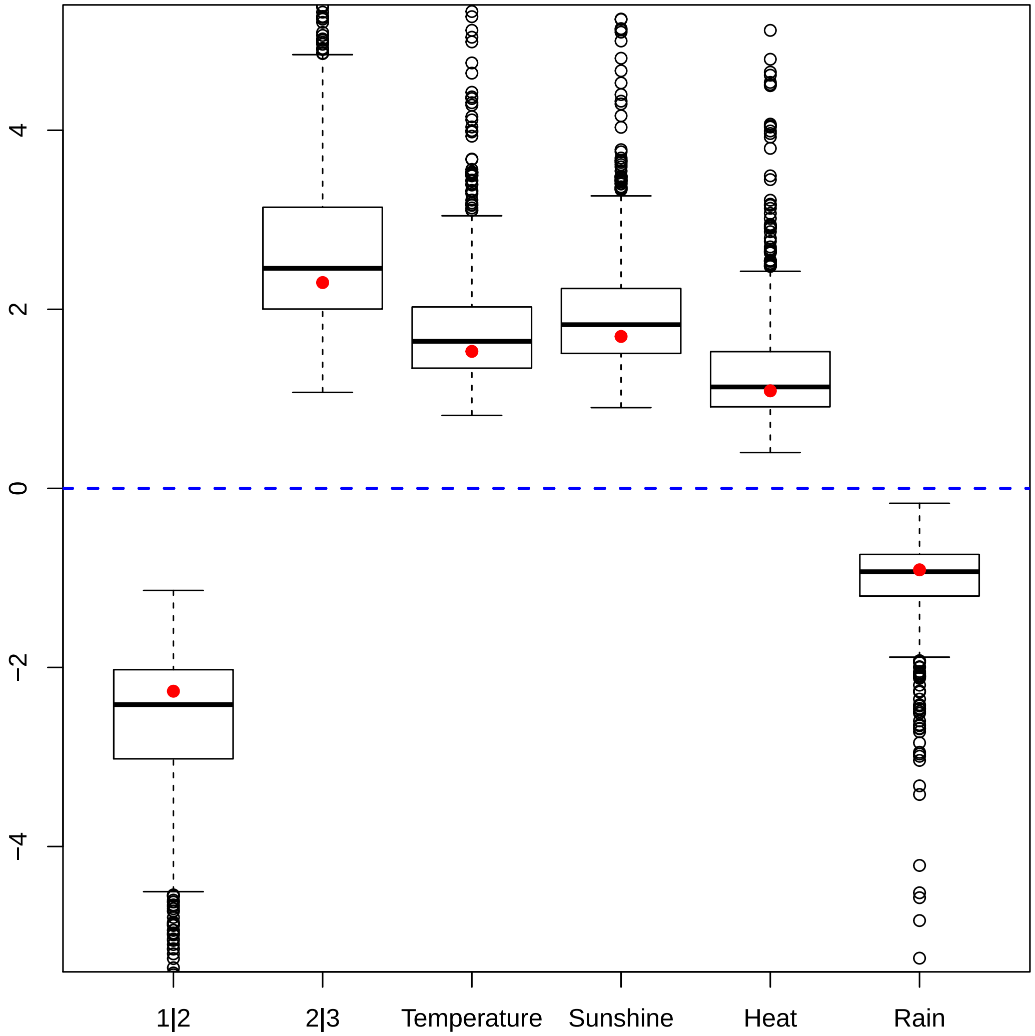

Figure 40: Bootstrap $(\mathbf{y}, \mathbf{T})$ distribution of the coefficients of the predictors, R=1000

```
boxplots.bootpls(bordeaux.bootYT1strata)
```

```
temp.cis <- confints.bootpls(bordeaux.bootYT1strata)
plots.confints.bootpls(temp.cis,typeIC="BCa",colIC=c("blue","blue","blue","blue"),
                         legendpos ="topright")
```

It could be interesting to display, through the models with 1 to 4 components, which of the predictors are significantly different from zero so that we could know if there is a stability of significant predictors or not. A function is available in our package, called `signpred`, to do this kind of graphic.

As we can see on Figures 46 and 47, there is a single difference between stratified bootstrap and regular one. Indeed, 1 predictor significant for stratified bootstrap of the 2 component model turn out to be non-significant for regular bootstrap in the 2 components model. During the cross-validation, 84 percents of results give 1 component and 7 percents give 2 components, representing than 91 percents of the results obtained during the 100 cross-validation made at the beginning.

The bootstrap technique used in this study, which is clearly faster and more stable than the other one, but the results between the $(\mathbf{y}, \mathbf{X})$ and $(\mathbf{y}, \mathbf{T})$ bootstrap techniques are really different and so it could be interesting to confront them with the help of some simulations.

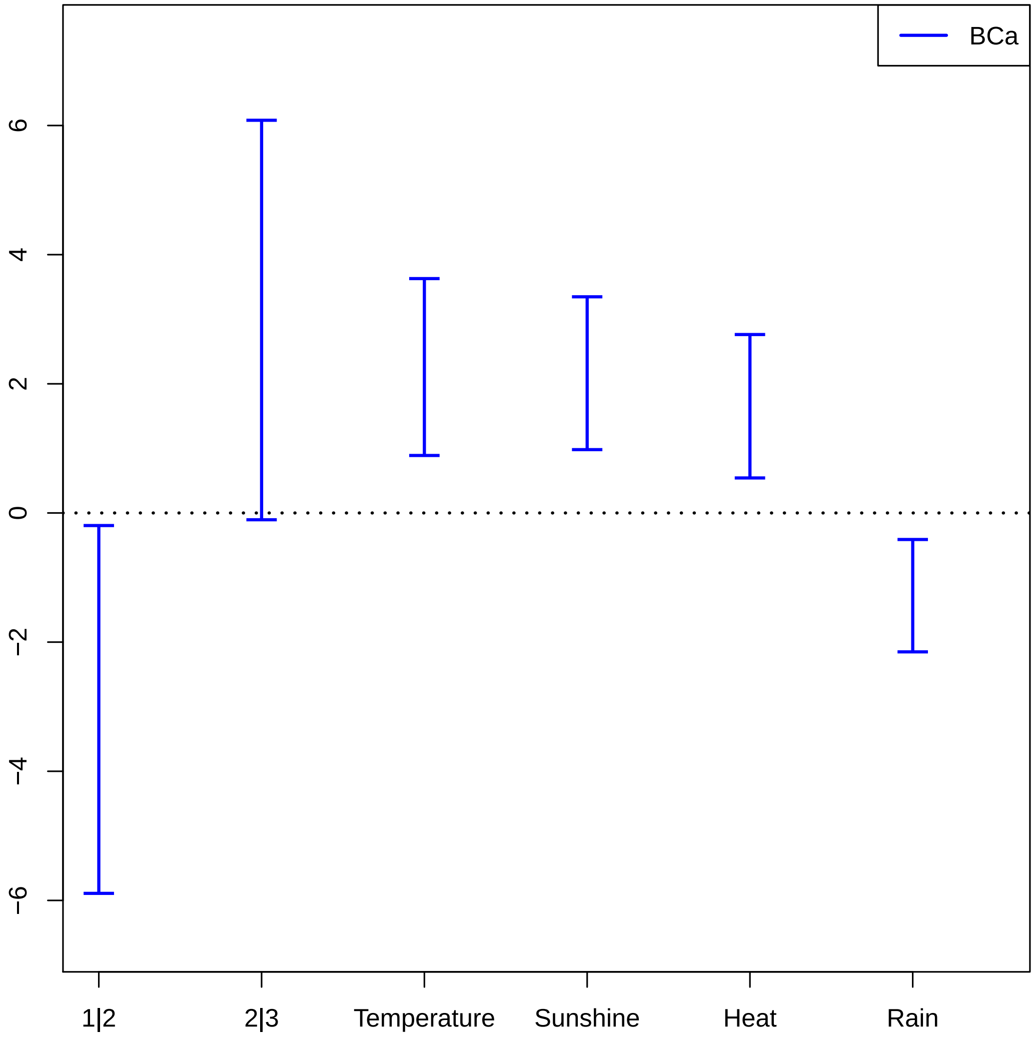

Figure 41: CI of the coefficients of the predictors, bootstrap $(\mathbf{y}, \mathbf{T})$, R=1000

```
res2<-plsRglm(ybordeaux,Xbordeaux,2,modele="pls-glm-polr")
res3<-plsRglm(ybordeaux,Xbordeaux,3,modele="pls-glm-polr")
res4<-plsRglm(ybordeaux,Xbordeaux,4,modele="pls-glm-polr")

bordeaux.bootYT2=bootplsglm(res2,sim="balanced", R=1000)
bordeaux.bootYT3=bootplsglm(res3,sim="balanced", R=1000)
bordeaux.bootYT4=bootplsglm(res4,sim="balanced", R=1000)
bordeaux.bootYT2s=bootplsglm(res2,sim="balanced", R=1000,strata=unclass(ybordeaux))
bordeaux.bootYT3s=bootplsglm(res3,sim="balanced", R=1000,strata=unclass(ybordeaux))
bordeaux.bootYT4s=bootplsglm(res4,sim="balanced", R=1000,strata=unclass(ybordeaux))

temp.ci2<-confints.bootpls(bordeaux.bootYT2)
temp.ci3<-confints.bootpls(bordeaux.bootYT3)
temp.ci4<-confints.bootpls(bordeaux.bootYT4)
temp.cis2<-confints.bootpls(bordeaux.bootYT2s)
temp.cis3<-confints.bootpls(bordeaux.bootYT3s)
temp.cis4<-confints.bootpls(bordeaux.bootYT4s)
```

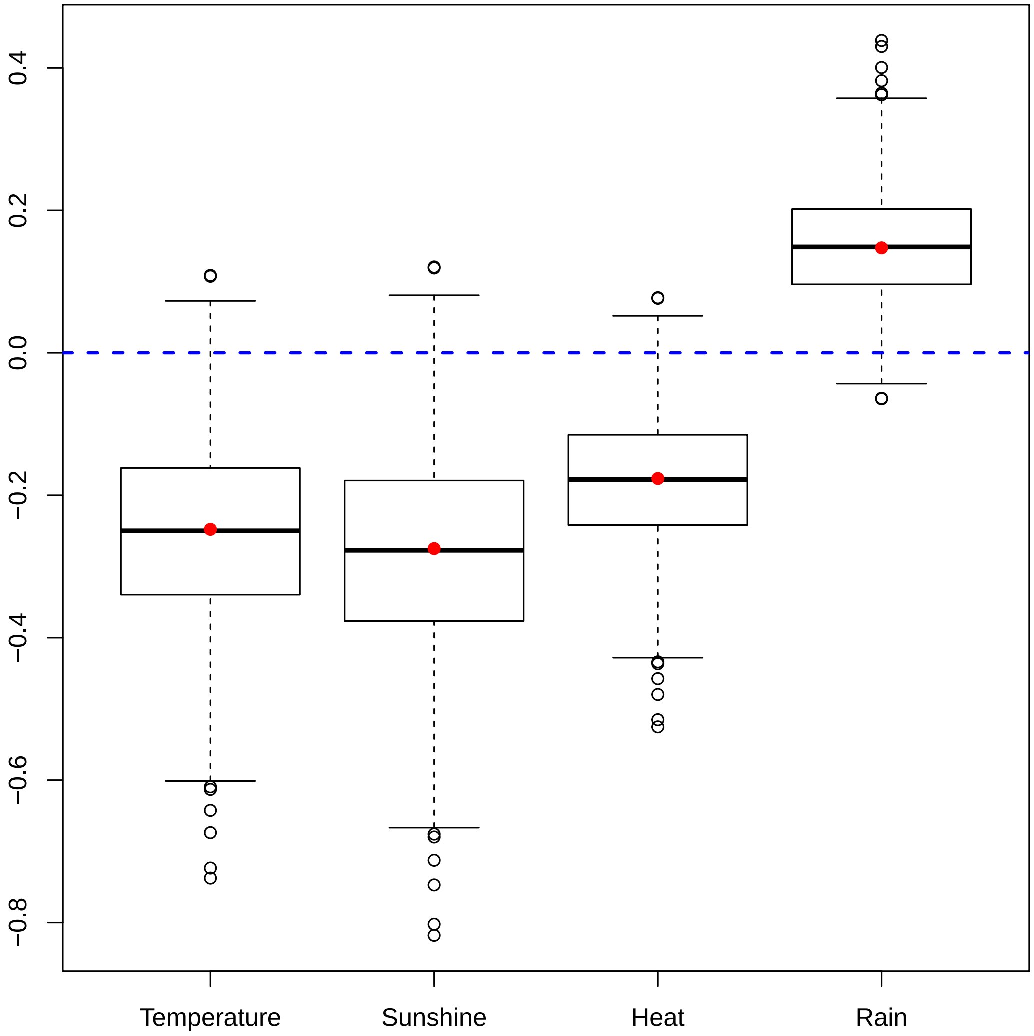

Figure 42: Bootstrap $(\mathbf{y}, \mathbf{T})$ distribution of the coefficients of the predictors, R=1000

```
ind.BCa.bordeauxYT1 <- (temp.ci[,7]<0&temp.ci[,8]<0)|(temp.ci[,7]>0&temp.ci[,8]>0)
ind.BCa.bordeauxYT2 <- (temp.ci2[,7]<0&temp.ci2[,8]<0)|(temp.ci2[,7]>0&temp.ci2[,8]>0)
ind.BCa.bordeauxYT3 <- (temp.ci3[,7]<0&temp.ci3[,8]<0)|(temp.ci3[,7]>0&temp.ci3[,8]>0)
ind.BCa.bordeauxYT4 <- (temp.ci4[,7]<0&temp.ci4[,8]<0)|(temp.ci4[,7]>0&temp.ci4[,8]>0)
ind.BCa.bordeauxYT1s <- (temp.cis[,7]<0&temp.cis[,8]<0)|(temp.cis[,7]>0&temp.cis[,8]>0)
ind.BCa.bordeauxYT2s <- (temp.cis2[,7]<0&temp.cis2[,8]<0)|(temp.cis2[,7]>0&temp.cis2[,8]>0)
ind.BCa.bordeauxYT3s <- (temp.cis3[,7]<0&temp.cis3[,8]<0)|(temp.cis3[,7]>0&temp.cis3[,8]>0)
ind.BCa.bordeauxYT4s <- (temp.cis4[,7]<0&temp.cis4[,8]<0)|(temp.cis4[,7]>0&temp.cis4[,8]>0)
```

```
(matind=(rbind(YT1=ind.BCa.bordeauxYT1,YT2=ind.BCa.bordeauxYT2,
               YT3=ind.BCa.bordeauxYT3,YT4=ind.BCa.bordeauxYT4)))

##     Temperature Sunshine  Heat  Rain
## YT1        TRUE     TRUE  TRUE  TRUE
## YT2       FALSE    FALSE FALSE FALSE
## YT3       FALSE    FALSE FALSE FALSE
## YT4       FALSE    FALSE FALSE FALSE

pi.e=prop.table(res.cv.modpls$CVMC)%*%matind
```

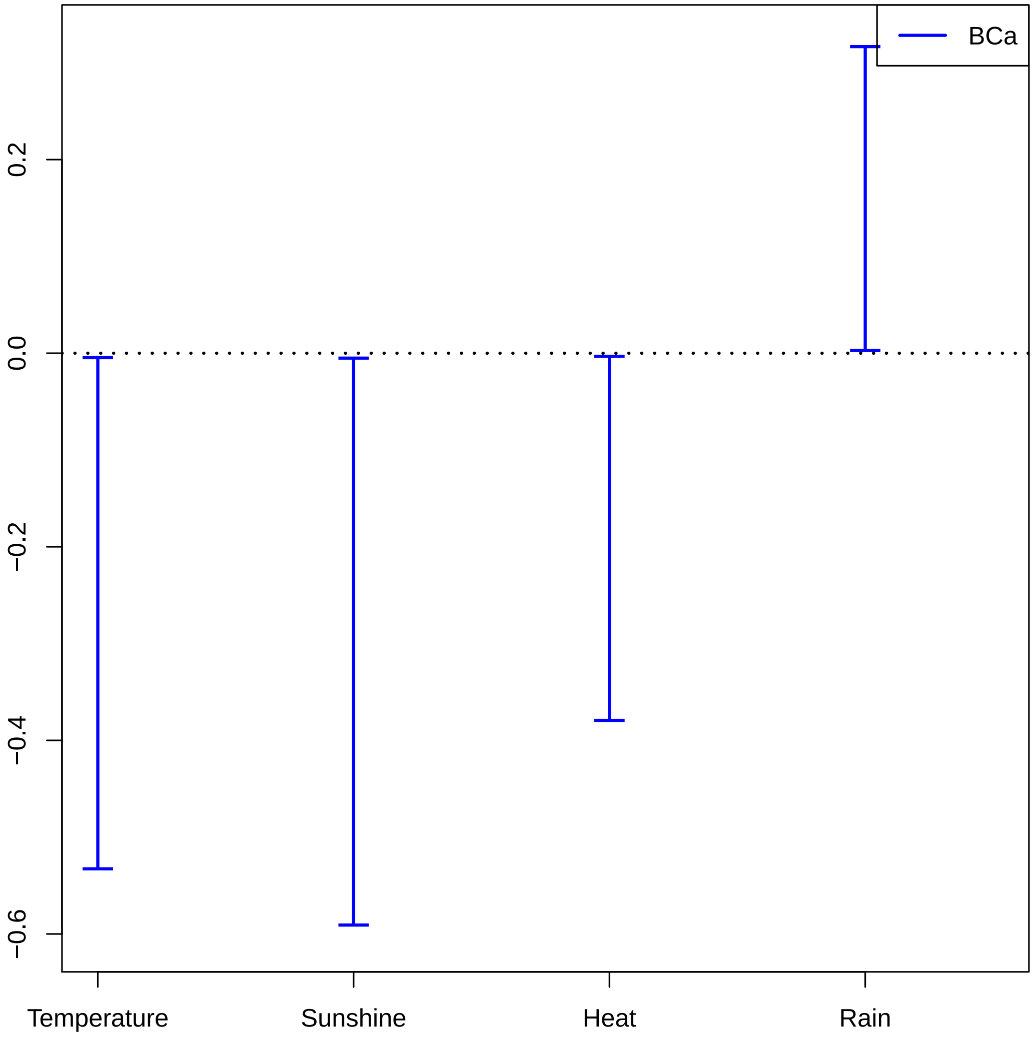

Figure 43: CI of the coefficients of the predictors, bootstrap $(\mathbf{y}, \mathbf{T})$, R=1000

```
pi.e
```

```
##      Temperature Sunshine Heat Rain
## [1,]        0.84     0.84 0.84 0.84
```

```
signpred(t(matind),labsize=.5, plotsize = 12)
mtext(expression(pi[e]),side=2,las=1,line=2,at=-1,cex=2)
text(1:(ncol(matind))-.5,-1,pi.e,cex=2)
text(1:(ncol(matind))-.5,-.5,c("Temp","Sun","Heat","Rain"),cex=2)
```

```
(matinds=(rbind(YT1=ind.BCa.bordeauxYT1s,YT2=ind.BCa.bordeauxYT2s,
               YT3=ind.BCa.bordeauxYT3s,YT4=ind.BCa.bordeauxYT4s)))
```

```
##     Temperature Sunshine  Heat  Rain
## YT1        TRUE     TRUE  TRUE  TRUE
## YT2        TRUE    FALSE FALSE FALSE
## YT3       FALSE    FALSE FALSE FALSE
## YT4       FALSE    FALSE FALSE FALSE
```

```
pi.es=prop.table(res.cv.modpls$CVMC)%*%matinds
pi.es
```

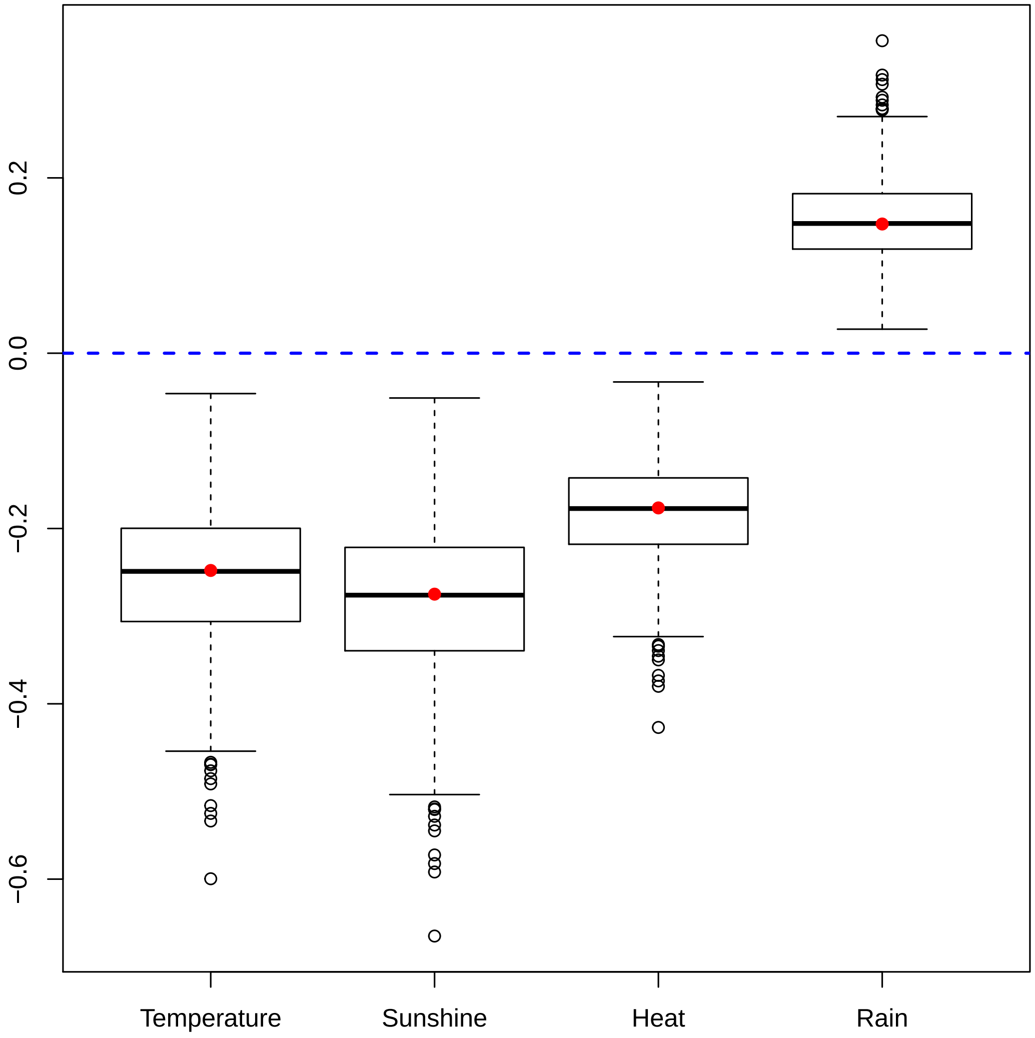

Figure 44: Bootstrap $(\mathbf{y}, \mathbf{T})$ distribution of the coefficients of the predictors, R=1000

```
##      Temperature Sunshine Heat Rain
## [1,]        0.91     0.84 0.84 0.84
```

```
signpred(t(matinds),pred.lablength=10,labsize=.5, plotsize = 12)
mtext(expression(pi[e]),side=2,las=1,line=2,at=-1,cex=2)
text(1:(ncol(matinds))-.5,-1,pi.es,cex=2)
text(1:(ncol(matinds))-.5,-.5,c("Temp","Sun","Heat","Rain"),cex=2)
```

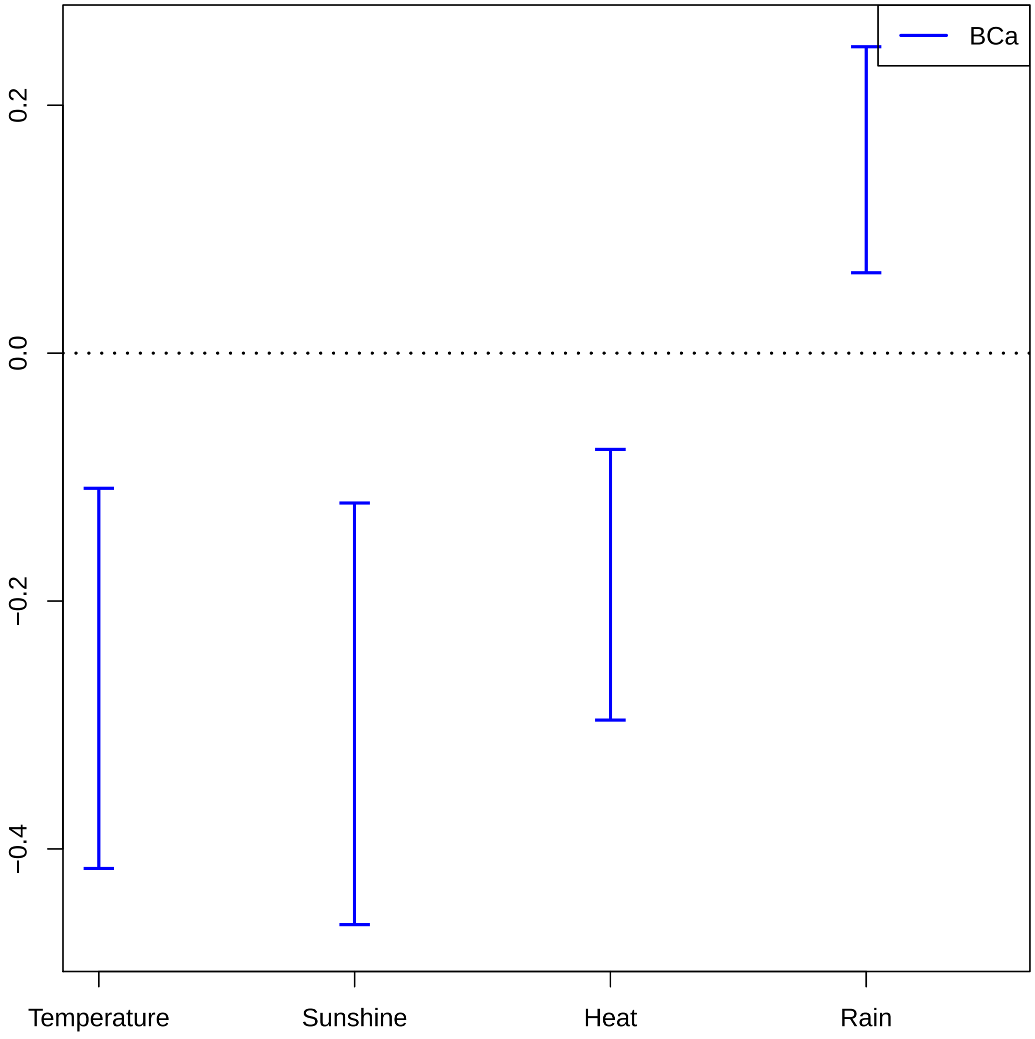

Figure 45: CI of the coefficients of the predictors, bootstrap $(\mathbf{y}, \mathbf{T})$, R=1000

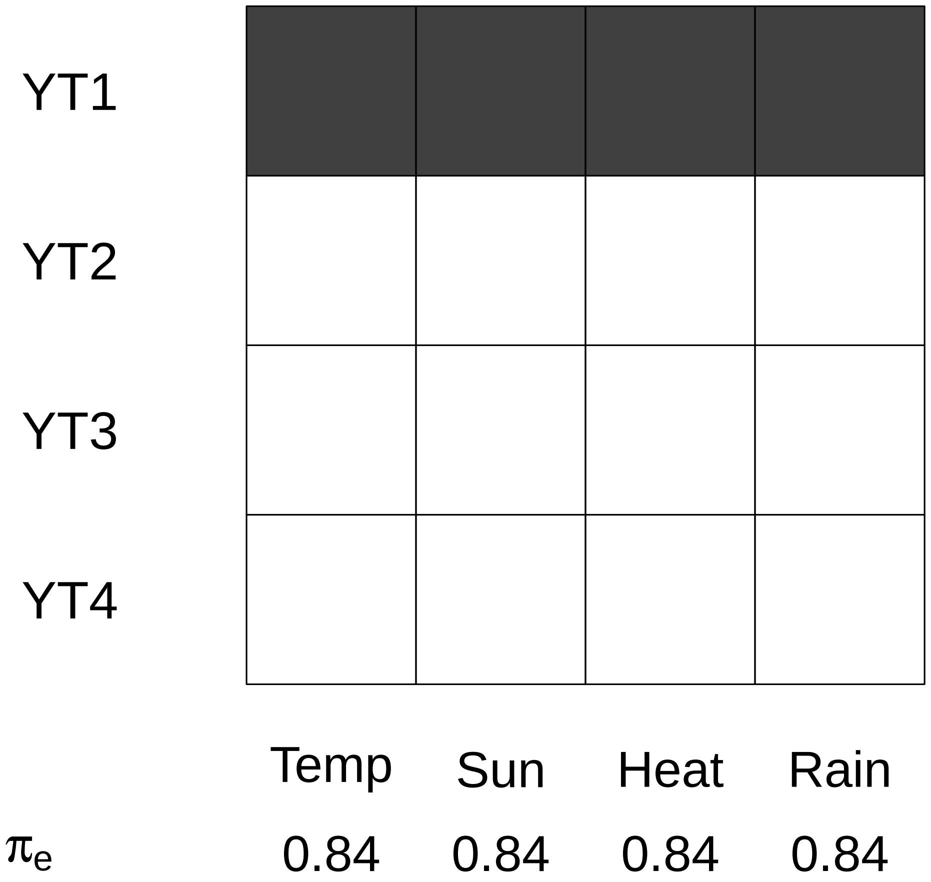

Figure 46: Significance of the predictors vs nbr of components, bootstrap $(\mathbf{y}, \mathbf{T})$, R=1000

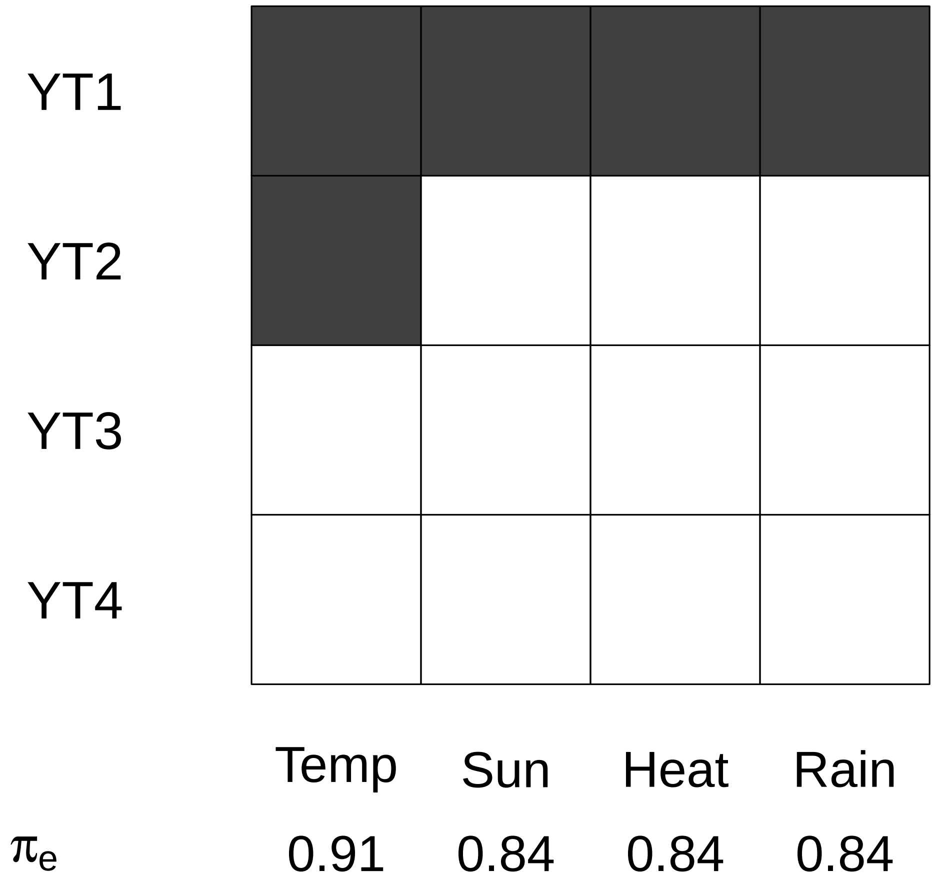

Figure 47: Significance of the predictors vs nbr of components, bootstrap $(\mathbf{y}, \mathbf{T})$, R=1000

3.6 PLS ordinal logistic regression: Hyptis

Cross-validation

```
rm(list = ls())
library(plsRglm)
data(hyptis,package="chemometrics")
yhyptis <- factor(hyptis$Group,ordered=TRUE)
Xhyptis <- as.data.frame(hyptis[,c(1:6)])
modpls <- plsRglm(yhyptis,Xhyptis,6,modele="pls-glm-polr",pvals.expli=TRUE)
```

```
## ____************************************************____
##
## Model: pls-glm-polr
## Method: logistic
##
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Component____ 5 ____
## ____Component____ 6 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****
```

```
modpls
```

```
## Number of required components:
## [1] 6
## Number of successfully computed components:
## [1] 6
## Coefficients:
##                 [,1]
## 1|2          8.96983
## 2|3         10.56293
## 3|4         12.25418
## Sabinene     0.18643
## Pinene      -3.00094
## Cineole      0.14989
## Terpinene   -0.05132
## Fenchone    -0.11987
## Terpinolene -0.18647
## Information criteria and Fit statistics:
##             AIC   BIC Missclassed Chi2_Pearson_Y
## Nb_Comp_0 86.87 91.08          20          60.00
## Nb_Comp_1 72.73 78.34          13          30.47
## Nb_Comp_2 71.10 78.11          12          27.69
## Nb_Comp_3 66.57 74.98          11          24.52
## Nb_Comp_4 67.29 77.10          10          24.43
## Nb_Comp_5 68.78 79.99          10          24.44
## Nb_Comp_6 70.70 83.31          10          24.63
```

```
colSums(modpls$pvalstep)
```

```
## temppvalstep temppvalstep temppvalstep temppvalstep temppvalstep
##            2            0            1            0            0
## temppvalstep
##            0
```

No discrepancy between formula specification (`formula` and `data`) and datasets (`dataY` and `dataX`) ones. Number of components to be retained:

- AIC → 3.
- BIC → 3.

- Non cross validated missclassed → 4.
- Non significant predictor criterion → 3.

One could have used the `grouplist` and the *caret* (from Jed Wing et al., 2014) package to provide custom balanced splits of the dataset into folds with respect to the response values.

```
set.seed(123)
cv.modpls<-cv.plsRglm(dataY=yhyptis,dataX=Xhyptis,nt=4,K=5,NK=100,modele="pls-glm-polr")
```

```
res.cv.modpls=cvtable(summary(cv.modpls,MClassed=TRUE))
## ____************************************************____
##
## Model: pls-glm-polr
## Method: logistic
##
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****
##
##
## NK: 1,  2,  3,  4,  5,  6,  7,  8,  9,  10
## NK: 11,  12,  13,  14,  15,  16,  17,  18,  19,  20
## NK: 21,  22,  23,  24,  25,  26,  27,  28,  29,  30
## NK: 31,  32,  33,  34,  35,  36,  37,  38,  39,  40
## NK: 41,  42,  43,  44,  45,  46,  47,  48,  49,  50
## NK: 51,  52,  53,  54,  55,  56,  57,  58,  59,  60
## NK: 61,  62,  63,  64,  65,  66,  67,  68,  69,  70
## NK: 71,  72,  73,  74,  75,  76,  77,  78,  79,  80
## NK: 81,  82,  83,  84,  85,  86,  87,  88,  89,  90
## NK: 91,  92,  93,  94,  95,  96,  97,  98,  99,  100
## CV MissClassed criterion:
##  1  2  3  4
## 20 27 39 14
##
## CV Q2Chi2 criterion:
##   0
## 100
##
## CV PreChi2 criterion:
##  1  2  3  4
## 66  0  7 27
```

The results (Fig. 48) confirm the results obtained during the original cross-validation to retain 3 components.

```
plot(res.cv.modpls)
```

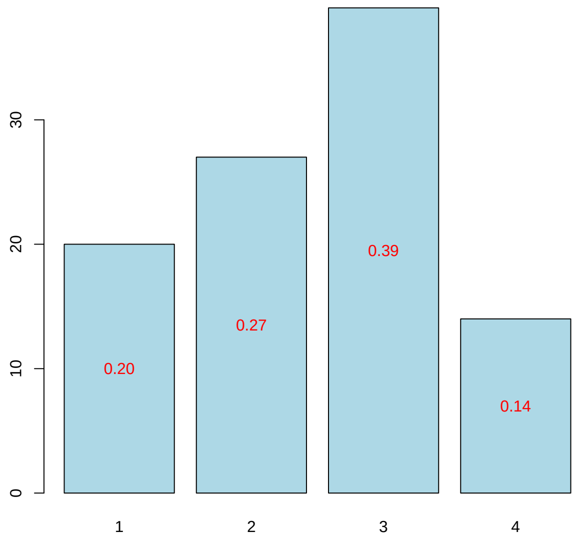

Figure 48: Nb components, 5-CV, n=100

```
modpls2 <- plsRglm(yhyptis,Xhyptis,3,modele="pls-glm-polr")
## ____************************************************____
##
## Model: pls-glm-polr
## Method: logistic
##
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****
modpls2
## Number of required components:
## [1] 3
## Number of successfully computed components:
## [1] 3
## Coefficients:
##                  [,1]
## 1|2          8.37257
## 2|3          9.92503
## 3|4         11.53479
## Sabinene     0.07356
## Pinene      -2.21204
## Cineole      0.06178
## Terpinene    0.00870
## Fenchone    -0.10557
## Terpinolene -0.20680
## Information criteria and Fit statistics:
##             AIC   BIC Missclassed Chi2_Pearson_Y
## Nb_Comp_0 86.87 91.08          20          60.00
## Nb_Comp_1 72.73 78.34          13          30.47
```

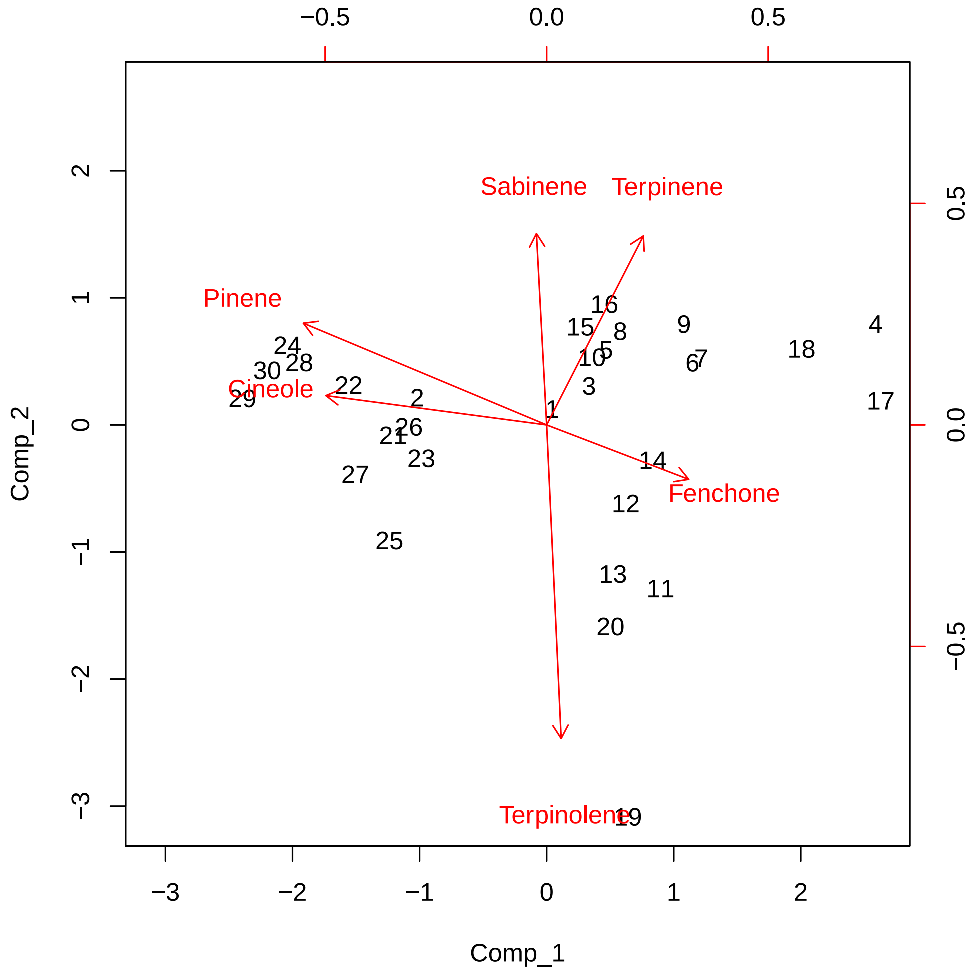

Figure 49: Biplot of the observations and the variables

```
## Nb_Comp_2 71.10 78.11          12          27.69
## Nb_Comp_3 66.57 74.98          11          24.52
table(yhyptis,predict(modpls2$FinalModel,type="class"))
##
## yhyptis 1 2 3 4
##       1 9 0 1 0
##       2 3 1 2 0
##       3 2 0 1 2
##       4 0 1 0 8
```

It is also possible to display the biplot of the observations and the predictors (Figure 49).

```
biplot(modpls2$tt,modpls2$pp)
```

Example of use of the `dataPredictY` option to predict the response values for a new dataset using the PLSGLR model.

```
modpls3 <- plsRglm(yhyptis[-c(1,11,17,22)],Xhyptis[-c(1,11,17,22),],3,modele="pls-glm-polr",
                   dataPredictY=Xhyptis[c(1,11,17,22),])
```

```
## ____************************************************____
##
## Model: pls-glm-polr
## Method: logistic
##
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****
```

```
modpls3$ValsPredictY
```

```
##           1       2        3         4
## 1  0.467585 0.36368 0.138012 0.0307262
## 11 0.326600 0.40462 0.214491 0.0542862
## 17 0.973745 0.02147 0.004034 0.0007501
## 22 0.009474 0.04144 0.204781 0.7443013
```

Compare with predicted categories to observed ones.

```
cbind(modpls3$ValsPredictYCat,yhyptis[c(1,11,17,22)])
```

```
##      [,1] [,2]
## [1,]    1    1
## [2,]    2    2
## [3,]    1    3
## [4,]    4    4
```

Bootstrap $(\mathbf{y}, \mathbf{X})$

In this example, we use permutation resampling and plot three types of bootstrap CI (normal, basic and percentile) by not specifying the `type` option and excluding BC_a CI computation with the `typeBCa=FALSE` option of the `confints.bootpls` function. We display CIs for each of the predictors (see Figure 51) and boxplots (see Figure 50).

```
hyptis.bootYX3<- bootplsglm(modpls2, typeboot="plsmodel", R=1000, strata=unclass(yhyptis),
                            sim="permutation")
rownames(hyptis.bootYX3$t0)<-c("1|2\n","2|3\n","3|4\n","Sabi\nnene","Pin\nene","Cine\nole",
                               "Terpi\nnene","Fenc\nhone","Terpi\nnolene")
```

```
boxplots.bootpls(hyptis.bootYX3,xaxisticks=FALSE,ranget0=TRUE)
```

Since we use permutation resampling, we have to spot the estimated values that do not lie within the bootstrap CI's limits. This procedure is much alike a significance test at the 5% level, since the conficence level of the CI is 95%.

```
plots.confints.bootpls(confints.bootpls(hyptis.bootYX3,typeBCa=FALSE),
                       legendpos = "bottomleft",xaxisticks=FALSE)
points(1:9,hyptis.bootYX3$t0,col="red",pch=19)
```

Bootstrap $(\mathbf{y}, \mathbf{T})$

Same specifications as for those used above for Bootstrap $(\mathbf{y}, \mathbf{X})$. We display CIs for each of the predictors (see Figure 53) and boxplots (see Figure 52).

```
hyptis.bootYT3<- bootplsglm(modpls2, R=1000, strata=unclass(yhyptis), sim="permutation")
rownames(hyptis.bootYT3$t0)<-c("Sabi\nnene","Pin\nene","Cine\nole","Terpi\nnene","Fenc\nhone",
                               "Terpi\nnolene")
```

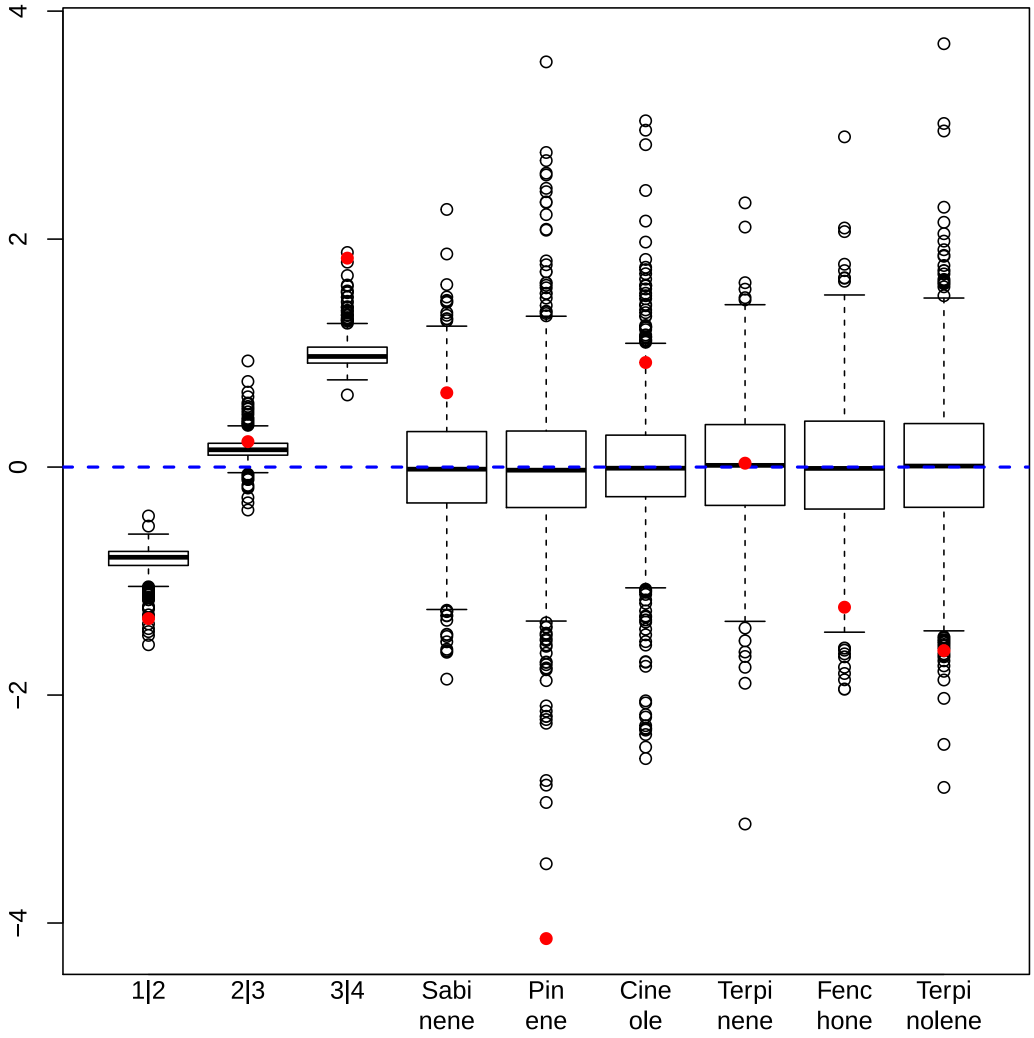

Figure 50: Bootstrap $(\mathbf{y}, \mathbf{T})$ distribution of the coefficients of the predictors, R=1000

```
boxplots.bootpls(hyptis.bootYT3,xaxisticks=FALSE,ranget0=TRUE)
```

Since we use permutation resampling, we have to spot the estimated values that do not lie within the bootstrap CI's limits. This procedure is much alike a significance test at the 5% level, since the conficence level of the CI is 95%.

```
plots.confints.bootpls(confints.bootpls(hyptis.bootYT3,typeBCa=FALSE), legendpos = "topright",
                       xaxisticks=FALSE)
points(1:6,hyptis.bootYT3$t0,col="red",pch=19)
```

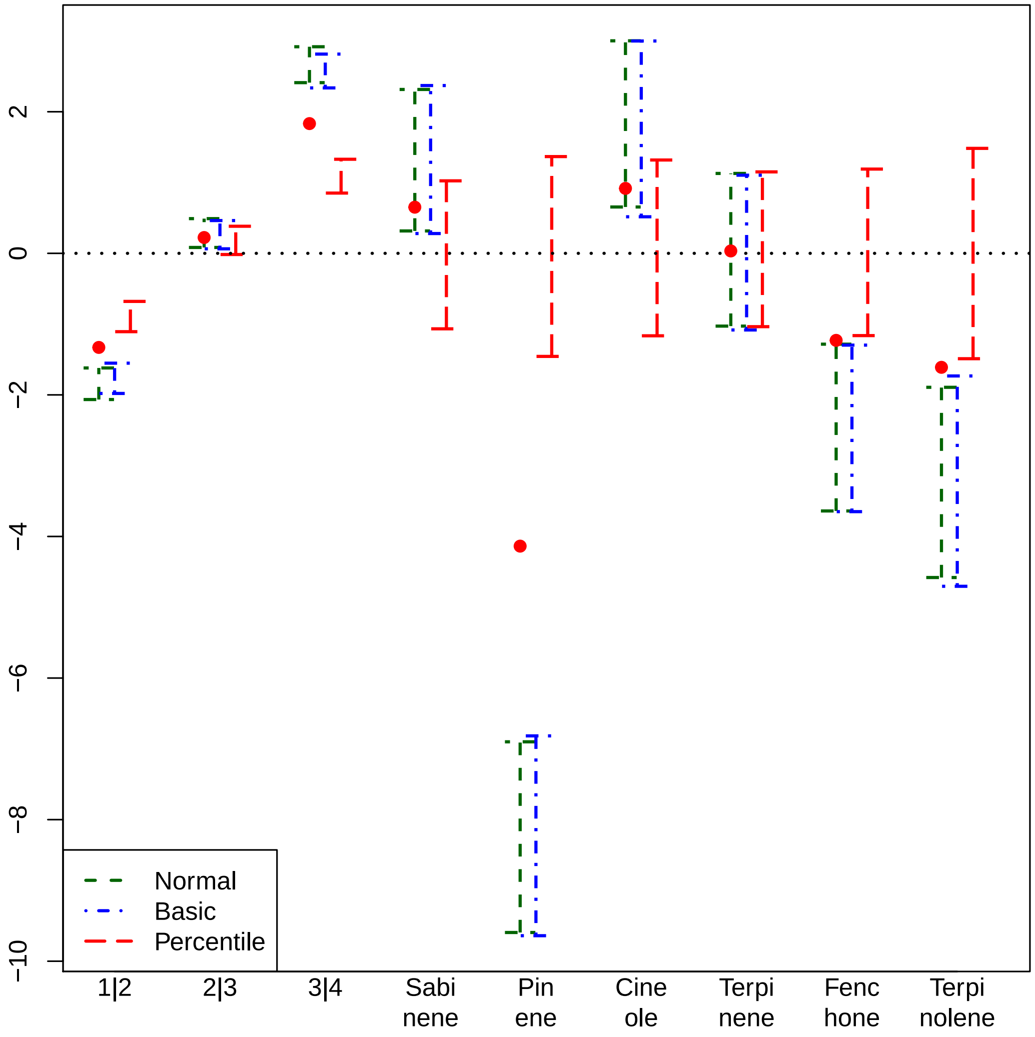

Figure 51: CI of the coefficients of the predictors, bootstrap $(\mathbf{y}, \mathbf{T})$, R=1000

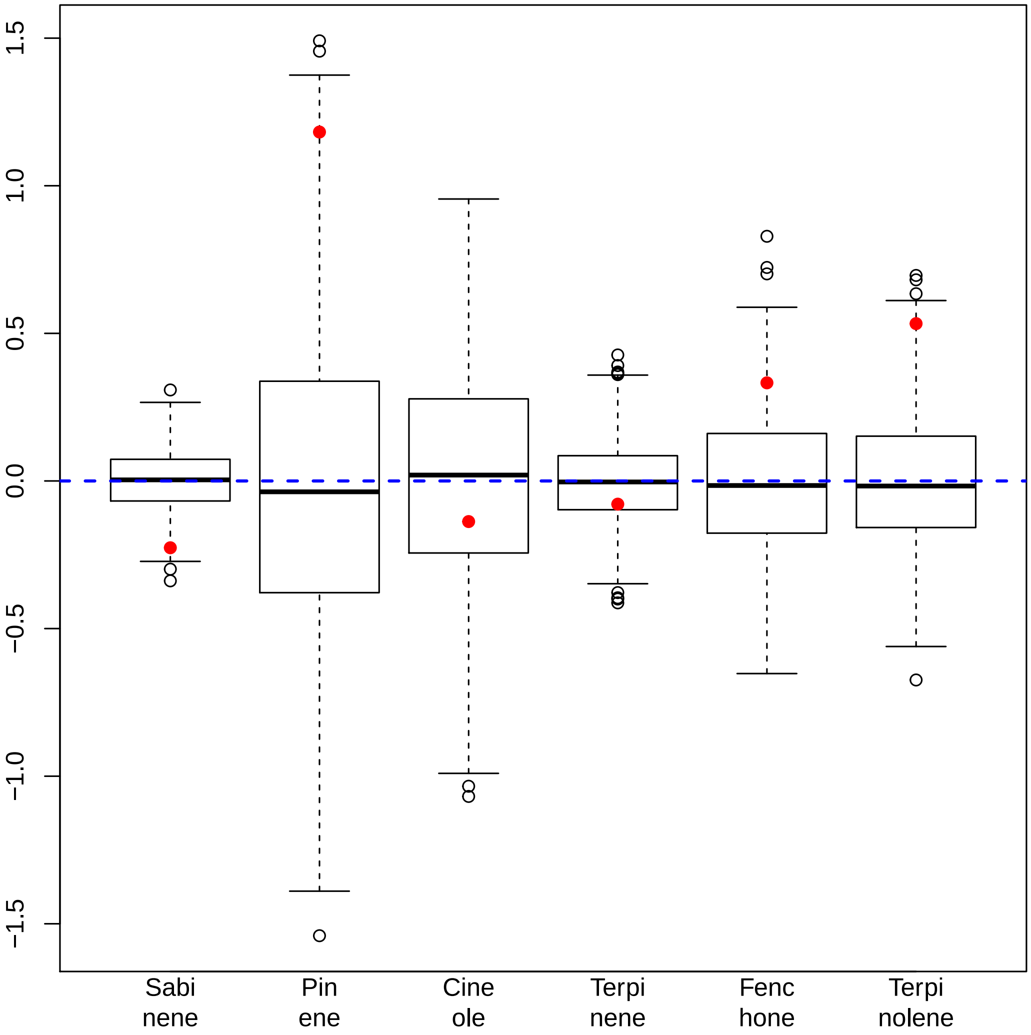

Figure 52: Bootstrap $(\mathbf{y}, \mathbf{T})$ distribution of the coefficients of the predictors, R=1000

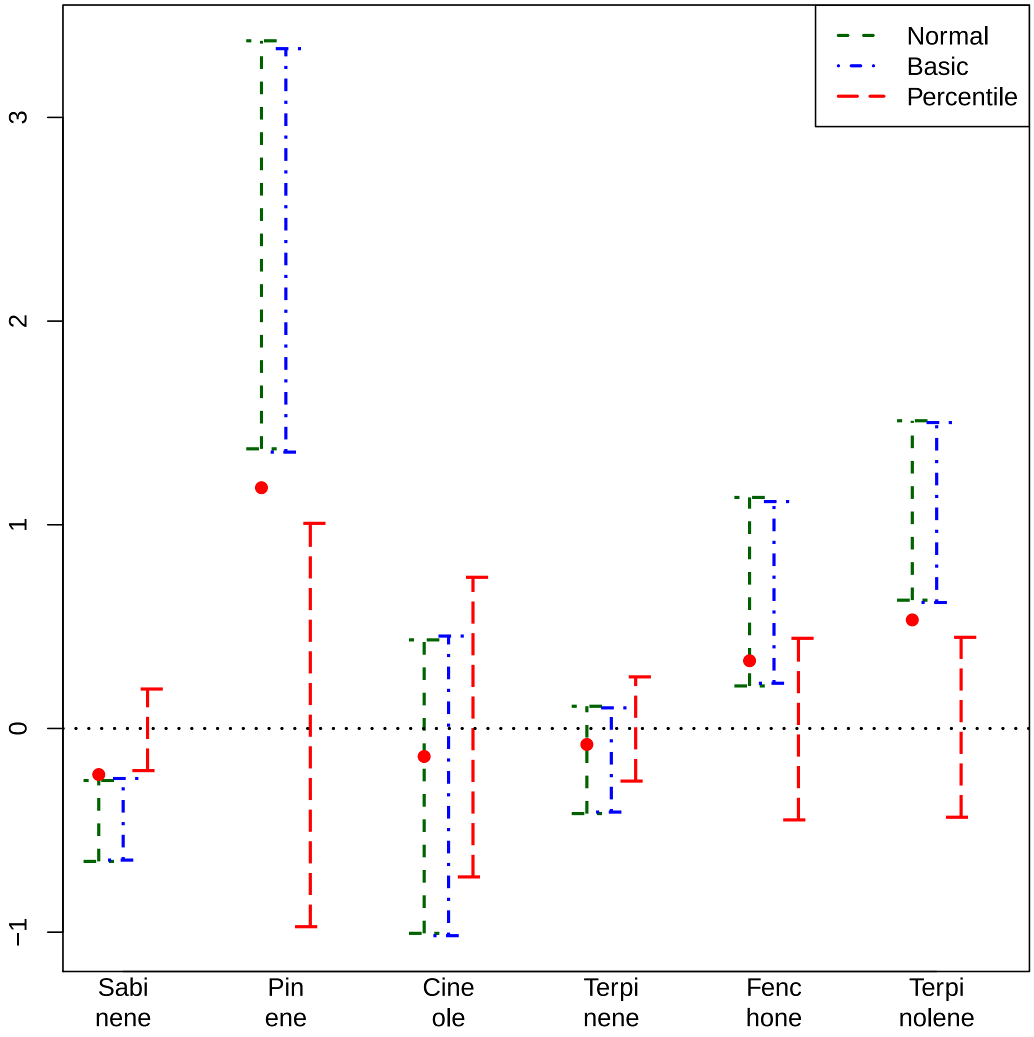

Figure 53: CI of the coefficients of the predictors, bootstrap $(\mathbf{y}, \mathbf{T})$, R=1000

3.7 PLS Poisson regression: Rock

Measurements on 48 rock samples from a petroleum reservoir. Twelve core samples from petroleum reservoirs were sampled by 4 cross-sections. Each core sample was measured for permeability, and each cross-section has total area of pores, total perimeter of pores, and shape. See `?rock` for more details.

Cross-validation

```
rm(list = ls())
library(plsRglm)
data(rock)
modpls <- plsRglm(area ~ ., data = rock,6,modele="pls-glm-family",family="poisson",
                  pvals.expli=TRUE)
## ____************************************************____
##
## Family: poisson
## Link function: log
##
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## Warning : 1 2 3 < 10^{-12}
## Warning only 3 components could thus be extracted
## ____Predicting X without NA neither in X or Y____
## ****________________________________________________****
modpls
## Number of required components:
## [1] 6
## Number of successfully computed components:
## [1] 3
## Coefficients:
##                 [,1]
## Intercept  7.6882295
## peri       0.0003431
## shape     -0.0344598
## perm       0.0005299
## Information criteria and Fit statistics:
##              AIC   BIC Chi2_Pearson_Y     RSS_Y   R2_Y R2_residY
## Nb_Comp_0 54058 54060          47100 338543101     NA        NA
## Nb_Comp_1 30974 30978          28213 173018641 0.4889    -7.307
## Nb_Comp_2 16196 16201          15194  79812992 0.7642    -7.307
## Nb_Comp_3 15863 15871          14805  80131766 0.7633    -7.307
##           RSS_residY
## Nb_Comp_0  3.385e+08
## Nb_Comp_1  2.812e+09
## Nb_Comp_2  2.812e+09
## Nb_Comp_3  2.812e+09
## Model with all the required components:
##
## Call:  glm(formula = YwotNA ~ ., family = structure(list(family = "poisson",
##     link = "log", linkfun = function (mu)
##     log(mu), linkinv = function (eta)
##     pmax(exp(eta), .Machine$double.eps), variance = function (mu)
##     mu, dev.resids = function (y, mu, wt)
##     {
##         r <- mu * wt
##         p <- which(y > 0)
##         r[p] <- (wt * (y * log(y/mu) - (y - mu)))[p]
##         2 * r
##     }, aic = function (y, n, mu, wt, dev)
##     -2 * sum(dpois(y, mu, log = TRUE) * wt), mu.eta = function (eta)
##     pmax(exp(eta), .Machine$double.eps), initialize = expression(
##         {
##             if (any(y < 0))
##                 stop("negative values not allowed for the 'Poisson' family")
##             n <- rep.int(1, nobs)
##             mustart <- y + 0.1
##         }), validmu = function (mu)
##     all(is.finite(mu)) && all(mu > 0), valideta = function (eta)
##     TRUE, simulate = function (object, nsim)
##     {
##         wts <- object$prior.weights
##         if (any(wts != 1))
##             warning("ignoring prior weights")
##         ftd <- fitted(object)
##         rpois(nsim * length(ftd), ftd)
##     }), .Names = c("family", "link", "linkfun", "linkinv", "variance",
## "dev.resids", "aic", "mu.eta", "initialize", "validmu", "valideta",
## "simulate"), class = "family"), data = structure(list(YwotNA = c(4990,
## 7002, 7558, 7352, 7943, 7979, 9333, 8209, 8393, 6425, 9364, 8624,
## 10651, 8868, 9417, 8874, 10962, 10743, 11878, 9867, 7838, 11876,
## 12212, 8233, 6360, 4193, 7416, 5246, 6509, 4895, 6775, 7894,
## 5980, 5318, 7392, 7894, 3469, 1468, 3524, 5267, 5048, 1016, 5605,
## 8793, 3475, 1651, 5514, 9718), tt.1 = c(0.780917208973395, 1.31628162567013,
## 1.25733473255635, 1.37685044684142, 1.40178986152356, 1.33369468759116,
## 1.48546879701717, 1.54536473907293, 0.943797704062212, 0.684614548016699,
## 1.55708156570345, 1.26154812044631, 1.13835622017351, 0.813891041348348,
## 1.24843216435484, 1.06707939674712, 1.56991647562089, 1.54095247422576,
## 1.73636047478215, 1.63188598759725, 0.978585290944047, 1.12423503501407,
## 1.45584001614919, 0.919654725426246, -0.906041393288676, -1.03154401191994,
## -0.732273781868988, -0.795512414771367, -1.00190749080515, -1.65126727435043,
## -1.28211600650803, -1.36108547021135, -1.25655828260807, -1.8293565897517,
## -1.20003533642413, -1.42133841866358, -0.384574137981601, -1.55603948879456,
## -0.467426254084444, -0.402630642425575, -2.21547743486887, -2.36912106672574,
## -2.41173526216677, -1.61384132437327, -1.04693074796276, -1.54634696628036,
## -0.831145438953197, -0.855628104069664), tt.2 = c(-0.810324943234362,
## -0.145745001768119, -0.052105922019788, -0.225382043607907, -0.15953323279508,
## -0.0328810351937208, 0.180198446788208, 0.124512823665787, 0.0321166647102502,
## -0.342534665485853, 0.308558738228635, 0.0608990556964554, 0.206924099741436,
## -0.040205541992077, 0.0675764284642719, -0.154868039799349, 0.400143439303565,
## 0.614019949650515, 0.516001376437726, 0.208228386578939, -0.252975716162375,
## 0.582440189331114, 0.64080766740926, -0.105260238577268, 0.257069610319131,
## -0.254872926283339, 0.0345029527187617, -0.254333280144128, 0.290385618365295,
## 0.242163693317762, 0.417084866638178, 0.20936424562948, 0.109298426683102,
## 0.174729248329808, -0.0218704538536954, 0.295542008801672, -1.18077496618469,
## -1.05526115135254, -1.30132384778801, -0.88330950088822, 0.657873326526871,
## 0.135085715602058, 1.05074086826238, 1.5115553530292, -0.539056555734109,
## -0.687138511892081, -0.440754488825967, -0.387311136647196),
##     tt.3 = c(0.0824307010502463, 0.0186666594555753, -0.0301625335456955,
##     0.0632888139586293, 0.0597345525997243, -0.00286016189976393,
##     -0.0291977764763363, 0.0072058013880108, -0.0355232043302926,
##     0.0130616759170939, 0.0537672437441116, 0.0490564311828598,
##     -0.073969282528354, -0.0874360998500556, 0.00532365589440289,
##     0.0260263534521784, -0.0351538352425509, -0.115348931275098,
##     -0.024591396729242, 0.0474749082789075, 0.093875625343324,
##     -0.142639441767272, -0.0638570344212428, 0.0269462103521066,
##     -0.0212287544999234, 0.112959543545675, 0.104811546989891,
##     0.182764563282465, 0.0933233701629197, -0.0830732542227469,
##     -0.0321807946782985, 0.0139499403715498, 0.140098543629951,
##     -0.0516483871977959, 0.200774187761041, 0.028884274511621,
##     -0.04283939218548, -0.4322131081358, -0.027045460099051,
##     -0.147786330912943, 0.0317445224418696, 0.161219720659487,
##     -0.157981984290508, -0.0756816841264136, 0.0391362102318692,
##     -0.0593670031981543, 0.0702066316501666, 0.0450541637573421
##     )), .Names = c("YwotNA", "tt.1", "tt.2", "tt.3"), row.names = c("1",
## "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13",
## "14", "15", "16", "17", "18", "19", "20", "21", "22", "23", "24",
## "25", "26", "27", "28", "29", "30", "31", "32", "33", "34", "35",
## "36", "37", "38", "39", "40", "41", "42", "43", "44", "45", "46",
## "47", "48"), class = "data.frame"), na.action = function (object,
##     ...)
## UseMethod("na.exclude"), model = TRUE, method = "glm.fit")
##
## Coefficients:
## (Intercept)         tt.1         tt.2         tt.3
##       8.821        0.196        0.442        0.368
##
## Degrees of Freedom: 47 Total (i.e. Null);  44 Residual
## Null Deviance:     53500
## Residual Deviance: 15300  AIC: 15900
```

```
colSums(modpls$pvalstep)
```

```
## temppvalstep temppvalstep temppvalstep
##            3            3            3
```

```
modpls2 <- plsRglm(area ~ .^2, data = rock,nt=6,modele="pls-glm-family",family="poisson",
                   pvals.expli=TRUE)
```

```
## ____************************************************____
##
## Family: poisson
## Link function: log
##
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Component____ 5 ____
## ____Component____ 6 ____
## ____Predicting X without NA neither in X or Y____
## ****________________________________________________****
```

```
modpls2
```

```
## Number of required components:
## [1] 6
## Number of successfully computed components:
## [1] 6
## Coefficients:
##                   [,1]
## Intercept   7.959e+00
## peri        2.233e-04
## shape      -6.074e-01
## perm        2.128e-04
## peri.shape  3.371e-04
## peri.perm   3.035e-07
## shape.perm -7.110e-04
## Information criteria and Fit statistics:
##             AIC   BIC Chi2_Pearson_Y     RSS_Y   R2_Y R2_residY
## Nb_Comp_0 54058 54060          47100 338543101     NA        NA
## Nb_Comp_1 26001 26005          23549 133178719 0.6066    -7.307
## Nb_Comp_2 15311 15316          14395  76726600 0.7734    -7.307
## Nb_Comp_3 12435 12442          11836  63448705 0.8126    -7.307
## Nb_Comp_4 12352 12362          11755  63173614 0.8134    -7.307
## Nb_Comp_5 12335 12346          11793  63832561 0.8114    -7.307
## Nb_Comp_6 12336 12349          11789  63892436 0.8113    -7.307
##           RSS_residY
## Nb_Comp_0  3.385e+08
## Nb_Comp_1  2.812e+09
## Nb_Comp_2  2.812e+09
## Nb_Comp_3  2.812e+09
## Nb_Comp_4  2.812e+09
## Nb_Comp_5  2.812e+09
## Nb_Comp_6  2.812e+09
## Model with all the required components:
##
## Call:  glm(formula = YwotNA ~ ., family = structure(list(family = "poisson",
##     link = "log", linkfun = function (mu)
##     log(mu), linkinv = function (eta)
##     pmax(exp(eta), .Machine$double.eps), variance = function (mu)
##     mu, dev.resids = function (y, mu, wt)
##     {
##         r <- mu * wt
##         p <- which(y > 0)
##         r[p] <- (wt * (y * log(y/mu) - (y - mu)))[p]
##         2 * r
##     }, aic = function (y, n, mu, wt, dev)
##     -2 * sum(dpois(y, mu, log = TRUE) * wt), mu.eta = function (eta)
##     pmax(exp(eta), .Machine$double.eps), initialize = expression(
##         {
##             if (any(y < 0))
##                 stop("negative values not allowed for the 'Poisson' family")
##             n <- rep.int(1, nobs)
##             mustart <- y + 0.1
##         }), validmu = function (mu)
##     all(is.finite(mu)) && all(mu > 0), valideta = function (eta)
##     TRUE, simulate = function (object, nsim)
##     {
##         wts <- object$prior.weights
##         if (any(wts != 1))
##             warning("ignoring prior weights")
##         ftd <- fitted(object)
##         rpois(nsim * length(ftd), ftd)
##     }), .Names = c("family", "link", "linkfun", "linkinv", "variance",
## "dev.resids", "aic", "mu.eta", "initialize", "validmu", "valideta",
## "simulate"), class = "family"), data = structure(list(YwotNA = c(4990,
## 7002, 7558, 7352, 7943, 7979, 9333, 8209, 8393, 6425, 9364, 8624,
## 10651, 8868, 9417, 8874, 10962, 10743, 11878, 9867, 7838, 11876,
## 12212, 8233, 6360, 4193, 7416, 5246, 6509, 4895, 6775, 7894,
## 5980, 5318, 7392, 7894, 3469, 1468, 3524, 5267, 5048, 1016, 5605,
## 8793, 3475, 1651, 5514, 9718), tt.1 = c(0.247812968144745, 1.31948245416588,
## 1.5660211902098, 1.10785096746952, 1.18417310234941, 1.51405861648906,
## 1.94390541309809, 1.76192086761077, 1.32481879895635, 0.636427700772978,
## 1.64209266934092, 1.2474833976862, 1.84175496629625, 1.37793333027851,
## 1.45621432954102, 1.05099400008355, 2.22162803252415, 2.84167905126443,
## 2.41141106771852, 1.67414182422787, 0.625745132343545, 2.48003469386001,
## 2.45513792723314, 0.935589228468705, -0.862834636247188, -1.42607338406595,
## -0.993022101807249, -1.26810622644613, -1.21552362104258, -1.85338390420457,
## -1.3133048924232, -1.57873675280058, -1.64474933068624, -2.19508646593659,
## -1.67313194731169, -1.65952781489026, -0.68576872359812, -1.68269236146371,
## -0.845313712523005, -0.358250701992002, -2.74647316489006, -3.06948544175051,
## -2.77843782961596, -1.27660100296348, -1.41315621926431, -1.99891481226837,
## -1.17850706281615, -1.15122961912554), tt.2 = c(-1.29246199258762,
## -0.441950695983101, -0.279909971492392, -0.581087791646522, -0.477688832053013,
## -0.258839609528562, 0.0483899599051332, -0.062857113475549, 0.0188387098006091,
## -0.539848932218849, 0.333457745721046, -0.0055814702777909, 0.244055171103785,
## -0.107731898163712, -0.0181557740546154, -0.337807004795206,
## 0.441227657498504, 0.838470293713883, 0.592450729285436, 0.0856267472870207,
## -0.436520984099486, 0.97560161725499, 0.995971757130343, -0.187182372783293,
## 0.695031580368585, -0.312531219179379, 0.269634831458982, -0.284516967533427,
## 0.711027952261481, 0.486386373495288, 0.940046675590924, 0.486319607923975,
## 0.267472090332844, 0.27345872065071, 0.0066543083365021, 0.63015918213636,
## -1.7010680042703, -1.86913545996363, -1.85621329444357, -1.32668290101343,
## 1.00902218351377, -0.259659086546611, 1.91817718088627, 3.35671623970287,
## -0.777451626576309, -1.14189659537463, -0.577858043704092, -0.489559673594226
## ), tt.3 = c(0.156529683350293, -0.0220865739843232, -0.259998367419295,
## 0.193666246534475, 0.209294977375201, -0.0987144859240145, -0.227063788126593,
## -0.0347888365954247, -0.0699115537870728, 0.104747534430973,
## 0.484937304861931, 0.396715970233717, -0.322698252937108, -0.396822190842004,
## 0.0879272550821946, 0.15850516599295, -0.157702093055386, -0.617636132850047,
## -0.0934821947185934, 0.295436850669696, 0.582923225109595, -0.582623079145618,
## -0.125986948904914, 0.268466128326307, 0.195888542820914, 0.581658286109338,
## 0.841479109839368, 1.05285481396931, 0.793647131528871, -0.509095756743188,
## 0.0133457559456633, 0.122169658449661, 0.784930213956892, -0.515412052925032,
## 1.06355867566627, 0.190564320803158, -0.346876836753519, -1.38480165368885,
## -0.323873689448953, -0.662392883224411, -0.374899721208408, -0.162215556255466,
## -1.40538374629324, -0.102696184184279, 0.106509848941297, -0.525552634326535,
## 0.368761098961588, 0.268197414382606), tt.4 = c(-0.423303552099416,
## -0.389935676309095, -0.207143121730113, -0.55730106044946, -0.52071127068616,
## -0.288393656553581, -0.223453477849056, -0.3703132042046, 0.154172979523637,
## 0.0156111821436785, -0.191065142029551, -0.149444062098099, 0.180222086752166,
## 0.241837936979652, -0.109300552919792, -0.155505624153573, -0.0724904469765964,
## 0.248983484994521, -0.130901246092597, -0.393587020926238, -0.199478180673511,
## 0.616351874324904, 0.340685537696661, 0.0117241770086005, 0.513173641847726,
## 0.136440460017053, 0.402558427099044, 0.205476518730575, 0.405918766128427,
## -0.0212401105634606, 0.333074814090876, 0.147952070980011, 0.122464720189116,
## -0.267199463083008, 0.0905346046641824, 0.116318322757945, 0.245479388503825,
## 1.02738206682214, 0.224625385160862, 0.480614534842434, -0.73742594020348,
## -1.16201018392752, -0.801563475488853, 0.610286290202808, 0.119826929928938,
## -0.0413328651163217, 0.194536649305572, 0.226846483438719), tt.5 = c(0.394502641654165,
## 0.0943694160330658, -0.0272348527263425, 0.203029847045477, 0.179066743645063,
## 0.0228314188525609, -0.0948978622164532, -0.00603113857523316,
## -0.0109252573644852, 0.148485377629984, 0.0812034852608167, 0.124122211560798,
## -0.141449903068604, -0.0923048620372805, 0.0352031056185367,
## 0.125748623314793, -0.145359701879434, -0.367249988949129, -0.159449662962322,
## 0.0683473822674748, 0.255696546052337, -0.311084066090505, -0.192630928858462,
## 0.123730379289237, 0.0362810285028479, -0.107223449750508, -0.0116993599650508,
## -0.0987375662735392, -0.0333888310818363, -0.00945991836048268,
## 0.0401949961805602, -0.0527232280027028, -0.185806164539218,
## -0.078183965271524, -0.271334031882573, -0.0597655389081256,
## 0.257952817083929, -0.131291270731274, 0.301431129758836, 0.0663031833982023,
## -0.119648990822969, -0.668977978599679, 0.37018771214068, 0.584026963140004,
## -0.038137893766655, -0.0800600018806026, -0.00957746130148813,
## -0.00808113256288684), tt.6 = c(0.107707049102867, -0.0265400823184502,
## 0.0217013926597069, -0.0692614209080914, -0.0807568328688751,
## -0.01693377363569, -0.00309864857548361, -0.0564263648621988,
## 0.0139815305326116, 0.0249206035878958, -0.164925157238094, -0.104624035552277,
## 0.0546870837954885, 0.0712924518570272, -0.0422941824611796,
## -0.0293989849679257, -0.00871994505030567, 0.142044297388914,
## -0.0348834220652955, -0.14095673019534, -0.0783959476776941,
## 0.162454873888663, 0.0384818061346711, -0.0365780015691075, -0.0290866171229789,
## 0.0471845803061495, -0.0386932293978416, 0.0280004365904804,
## -0.0601601091200545, -0.0464997160105712, -0.0336304755104258,
## -0.0247460391340078, -0.0155290463372893, -0.0396405302741342,
## 0.00798671328218103, -0.029860545558012, 0.207159891541222, -0.461138870160792,
## 0.25104552525793, 0.0787315423720673, -0.0196953054389839, 0.0992034848587048,
## -0.0150456743042856, 0.0887801726707807, 0.0918147817415529,
## 0.0421179726924038, 0.0726750780060747, 0.0555484200479957)), .Names = c("YwotNA",
## "tt.1", "tt.2", "tt.3", "tt.4", "tt.5", "tt.6"), row.names = c("1",
## "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13",
## "14", "15", "16", "17", "18", "19", "20", "21", "22", "23", "24",
## "25", "26", "27", "28", "29", "30", "31", "32", "33", "34", "35",
## "36", "37", "38", "39", "40", "41", "42", "43", "44", "45", "46",
## "47", "48"), class = "data.frame"), na.action = function (object,
##     ...)
## UseMethod("na.exclude"), model = TRUE, method = "glm.fit")
##
## Coefficients:
## (Intercept)         tt.1         tt.2         tt.3         tt.4
##      8.8155       0.1799       0.2100       0.2018       0.0468
##        tt.5         tt.6
##      0.0393       0.0221
##
## Degrees of Freedom: 47 Total (i.e. Null);  41 Residual
## Null Deviance:      53500
## Residual Deviance: 11800  AIC: 12300
```

```
colSums(modpls2$pvalstep)
```

```
## temppvalstep temppvalstep temppvalstep temppvalstep temppvalstep
##            6            6            6            6            6
## temppvalstep
##            0
```

According to IC criteria values, the quadratic model is worth its increase in complexity, so we keep it. Number of components to be retained:

- AIC → 5.
- BIC → 5.
- Non significant predictor criterion → 5.

```
set.seed(123)
cv.modpls2<-cv.plsRglm(area~.^2,data=rock,nt=6,modele="pls-glm-poisson",K=8,NK=100)
```

```
res.cv.modpls2=cvtable(summary(cv.modpls2))
```

```
## ____************************************************____
##
## Family: poisson
## Link function: log
##
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Component____ 5 ____
## ____Component____ 6 ____
## ____Predicting X without NA neither in X or Y____
## ****________________________________________________****
##
##
## NK: 1,  2,  3,  4,  5,  6,  7,  8,  9,  10
## NK: 11,  12,  13,  14,  15,  16,  17,  18,  19,  20
## NK: 21,  22,  23,  24,  25,  26,  27,  28,  29,  30
## NK: 31,  32,  33,  34,  35,  36,  37,  38,  39,  40
## NK: 41,  42,  43,  44,  45,  46,  47,  48,  49,  50
## NK: 51,  52,  53,  54,  55,  56,  57,  58,  59,  60
## NK: 61,  62,  63,  64,  65,  66,  67,  68,  69,  70
## NK: 71,  72,  73,  74,  75,  76,  77,  78,  79,  80
## NK: 81,  82,  83,  84,  85,  86,  87,  88,  89,  90
## NK: 91,  92,  93,  94,  95,  96,  97,  98,  99,  100
##
## CV Q2Chi2 criterion:
##  0  1  2
##  0  1 99
##
## CV PreChi2 criterion:
##  1  2  3
##  0  7 93
```

This time we use the $PRE\chi^2$ as a cross validation criterion. It is the analoguous to the PRESS in this Poisson regression setting. According to the results, we chose to retain 3 components. It is known that, in PLSGLR logistic binary setting, the AIC and BIC tend to select too many components.

```
plot(res.cv.modpls2,type="CVPreChi2")
```

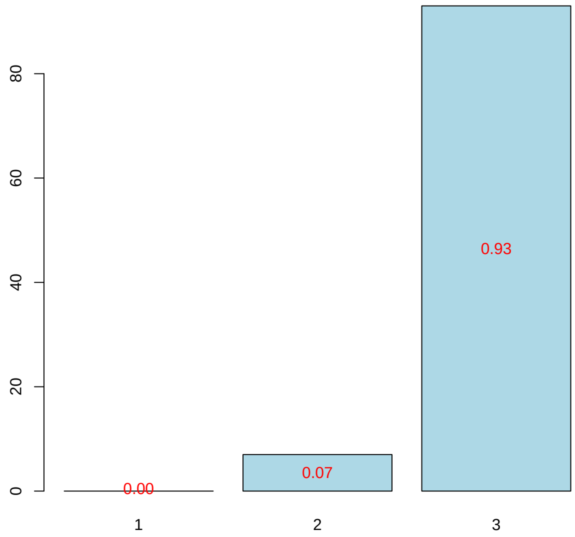

Figure 54: Nb components, 8-CV, n=100

The PLSGLR Poisson model with 3 components.

```
modpls3 <- plsRglm(area˜.ˆ2,data=rock,nt=3,modele="pls-glm-poisson")
## ____************************************************____
##
## Family: poisson
## Link function: log
##
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Predicting X without NA neither in X or Y____
## ****________________________________________________****
modpls3
## Number of required components:
## [1] 3
## Number of successfully computed components:
## [1] 3
## Coefficients:
##                  [,1]
## Intercept   7.884e+00
## peri        2.509e-04
## shape      -5.635e-01
## perm        3.351e-04
## peri.shape  2.884e-04
## peri.perm   2.510e-07
## shape.perm -7.008e-04
## Information criteria and Fit statistics:
##              AIC   BIC Chi2_Pearson_Y     RSS_Y   R2_Y R2_residY
## Nb_Comp_0 54058 54060          47100 338543101     NA        NA
## Nb_Comp_1 26001 26005          23549 133178719 0.6066    -7.307
## Nb_Comp_2 15311 15316          14395  76726600 0.7734    -7.307
## Nb_Comp_3 12435 12442          11836  63448705 0.8126    -7.307
##           RSS_residY
## Nb_Comp_0  3.385e+08
## Nb_Comp_1  2.812e+09
## Nb_Comp_2  2.812e+09
## Nb_Comp_3  2.812e+09
## Model with all the required components:
##
## Call:  glm(formula = YwotNA ˜ ., family = structure(list(family = "poisson",
##     link = "log", linkfun = function (mu)
##     log(mu), linkinv = function (eta)
##     pmax(exp(eta), .Machine$double.eps), variance = function (mu)
##     mu, dev.resids = function (y, mu, wt)
##     {
##         r <- mu * wt
##         p <- which(y > 0)
##         r[p] <- (wt * (y * log(y/mu) - (y - mu)))[p]
##         2 * r
##     }, aic = function (y, n, mu, wt, dev)
##     -2 * sum(dpois(y, mu, log = TRUE) * wt), mu.eta = function (eta)
##     pmax(exp(eta), .Machine$double.eps), initialize = expression(
##         {
##             if (any(y < 0))
##                 stop("negative values not allowed for the 'Poisson' family")
##             n <- rep.int(1, nobs)
##             mustart <- y + 0.1
##         }), validmu = function (mu)
##     all(is.finite(mu)) && all(mu > 0), valideta = function (eta)
##     TRUE, simulate = function (object, nsim)
##     {
##         wts <- object$prior.weights
##         if (any(wts != 1))
##             warning("ignoring prior weights")
##         ftd <- fitted(object)
##         rpois(nsim * length(ftd), ftd)
##     }), .Names = c("family", "link", "linkfun", "linkinv", "variance",
## "dev.resids", "aic", "mu.eta", "initialize", "validmu", "valideta",
## "simulate"), class = "family"), data = structure(list(YwotNA = c(4990,
## 7002, 7558, 7352, 7943, 7979, 9333, 8209, 8393, 6425, 9364, 8624,
## 10651, 8868, 9417, 8874, 10962, 10743, 11878, 9867, 7838, 11876,
## 12212, 8233, 6360, 4193, 7416, 5246, 6509, 4895, 6775, 7894,
## 5980, 5318, 7392, 7894, 3469, 1468, 3524, 5267, 5048, 1016, 5605,
## 8793, 3475, 1651, 5514, 9718), tt.1 = c(0.247812968144745, 1.31948245416588,
## 1.5660211902098, 1.10785096746952, 1.18417310234941, 1.51405861648906,
## 1.94390541309809, 1.76192086761077, 1.32481879895635, 0.636427700772978,
## 1.64209266934092, 1.2474833976862, 1.84175496629625, 1.37793333027851,
## 1.45621432954102, 1.05099400008355, 2.22162803252415, 2.84167905126443,
## 2.41141106771852, 1.67414182422787, 0.625745132343545, 2.48003469386001,
## 2.45513792723314, 0.935589228468705, -0.862834636247188, -1.42607338406595,
## -0.993022101807249, -1.26810622644613, -1.21552362104258, -1.85338390420457,
## -1.3133048924232, -1.57873675280058, -1.64474933068624, -2.19508646593659,
## -1.67313194731169, -1.65952781489026, -0.68576872359812, -1.68269236146371,
## -0.845313712523005, -0.358250701992002, -2.74647316489006, -3.06948544175051,
## -2.77843782961596, -1.27660100296348, -1.41315621926431, -1.99891481226837,
## -1.17850706281615, -1.15122961912554), tt.2 = c(-1.29246199258762,
## -0.441950695983101, -0.279909971492392, -0.581087791646522, -0.477688832053013,
## -0.258839609528562, 0.0483899599051332, -0.062857113475549, 0.0188387098006091,
## -0.539848932218849, 0.333457745721046, -0.0055814702777909, 0.244055171103785,
## -0.107731898163712, -0.0181557740546154, -0.337807004795206,
## 0.441227657498504, 0.838470293713883, 0.592450729285436, 0.0856267472870207,
## -0.436520984099486, 0.97560161725499, 0.995971757130343, -0.187182372783293,
## 0.695031580368585, -0.312531219179379, 0.269634831458982, -0.284516967533427,
## 0.711027952261481, 0.486386373495288, 0.940046675590924, 0.486319607923975,
## 0.267472090332844, 0.27345872065071, 0.0066543083365021, 0.63015918213636,
## -1.7010680042703, -1.86913545996363, -1.85621329444357, -1.32668290101343,
## 1.00902218351377, -0.259659086546611, 1.91817718088627, 3.35671623970287,
## -0.777451626576309, -1.14189659537463, -0.577858043704092, -0.489559673594226
## ), tt.3 = c(0.156529683350293, -0.0220865739843232, -0.259998367419295,
## 0.193666246534475, 0.209294977375201, -0.0987144859240145, -0.227063788126593,
## -0.0347888365954247, -0.0699115537870728, 0.104747534430973,
## 0.484937304861931, 0.396715970233717, -0.322698252937108, -0.396822190842004,
## 0.0879272550821946, 0.15850516599295, -0.157702093055386, -0.617636132850047,
## -0.0934821947185934, 0.295436850669696, 0.582923225109595, -0.582623079145618,
## -0.125986948904914, 0.268466128326307, 0.195888542820914, 0.581658286109338,
## 0.841479109839368, 1.05285481396931, 0.793647131528871, -0.509095756743188,
## 0.0133457559456633, 0.122169658449661, 0.784930213956892, -0.515412052925032,
## 1.06355867566627, 0.190564320803158, -0.346876836753519, -1.38480165368885,
## -0.323873689448953, -0.662392883224411, -0.374899721208408, -0.162215556255466,
## -1.40538374629324, -0.102696184184279, 0.106509848941297, -0.525552634326535,
## 0.368761098961588, 0.268197414382606)), .Names = c("YwotNA",
## "tt.1", "tt.2", "tt.3"), row.names = c("1", "2", "3", "4", "5",
## "6", "7", "8", "9", "10", "11", "12", "13", "14", "15", "16",
## "17", "18", "19", "20", "21", "22", "23", "24", "25", "26", "27",
## "28", "29", "30", "31", "32", "33", "34", "35", "36", "37", "38",
## "39", "40", "41", "42", "43", "44", "45", "46", "47", "48"), class = "data.frame"),
##     na.action = function (object, ...)
```

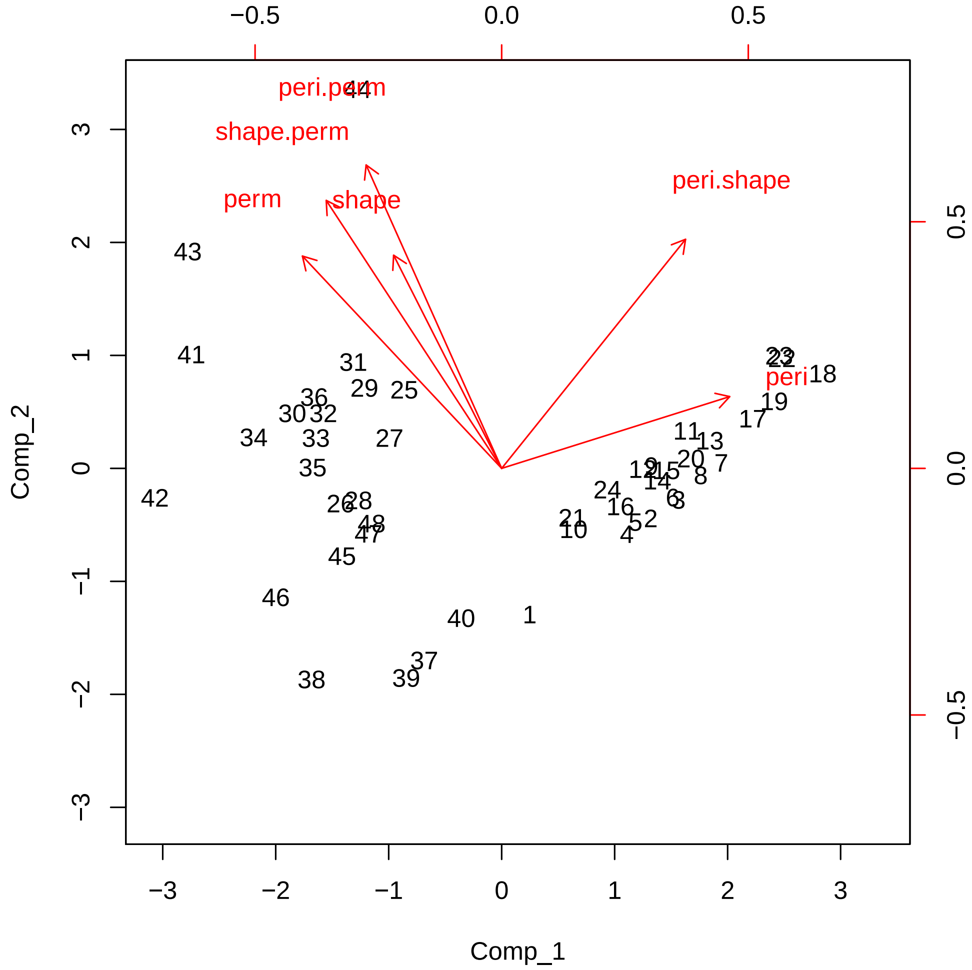

Figure 55: Biplot of the observations and the variables

```
##     UseMethod("na.exclude"), model = TRUE, method = "glm.fit")
##
## Coefficients:
## (Intercept)         tt.1         tt.2         tt.3
##       8.815        0.178        0.216        0.202
##
## Degrees of Freedom: 47 Total (i.e. Null);  44 Residual
## Null Deviance:     53500
## Residual Deviance: 11900  AIC: 12400
```

It is also possible to display the biplot of the observations and the predictors (Figure 55).

```
biplot(modpls3$tt,modpls3$pp)
```

Bootstrap $(\mathbf{y}, \mathbf{X})$

In this example, we use antithetic resampling and plot the four types of bootstrap CI by not specifying the type option. We display CIs for each of the predictors (see Figure 57) and boxplots (see Figure 56).

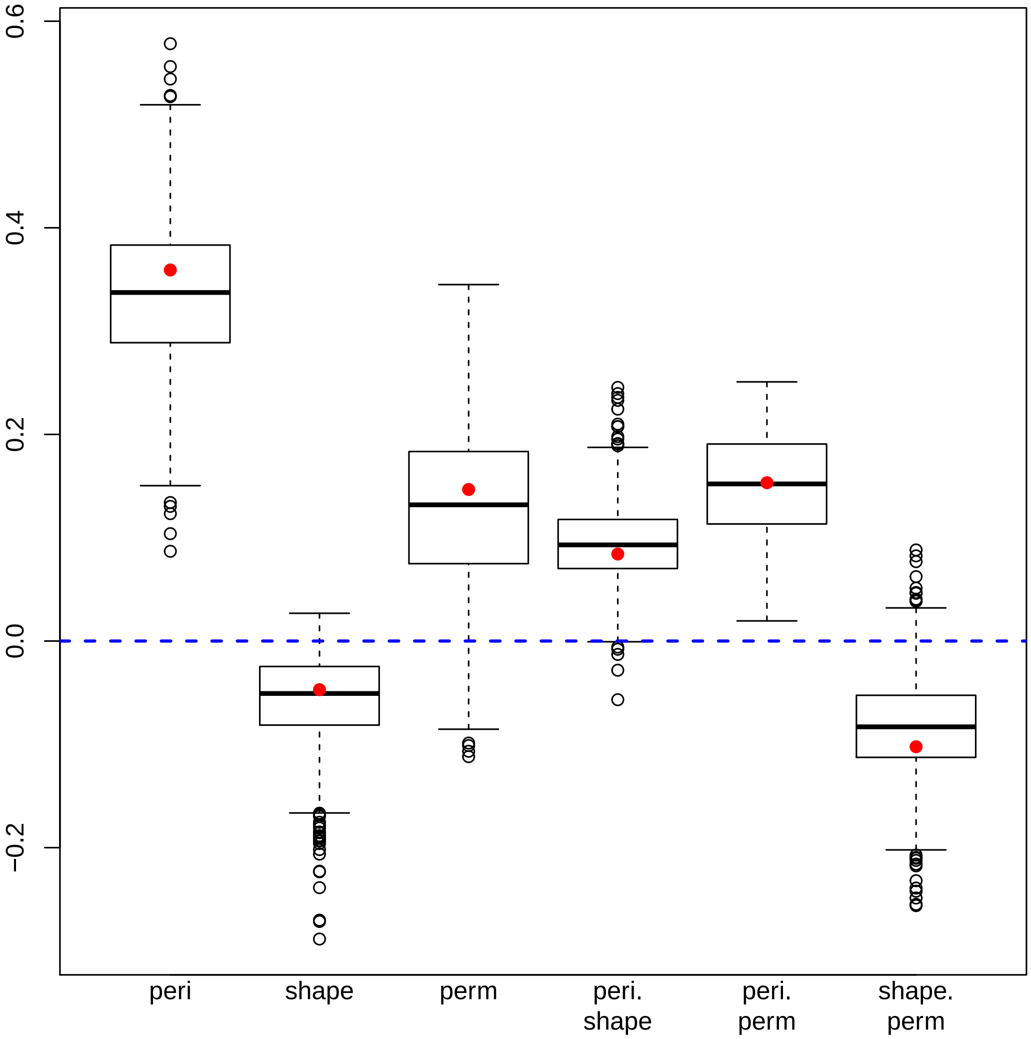

Figure 56: Bootstrap $(\mathbf{y}, \mathbf{T})$ distribution of the coefficients of the predictors, R=1000

```
rock.bootYX3<- bootplsglm(modpls3, typeboot="plsmodel", R=1000, sim="antithetic")
rownames(rock.bootYX3$t0)<-c("Intercept\n","peri\n","shape\n","perm\n","peri.\nshape",
                             "peri.\nperm","shape.\nperm")
```

```
boxplots.bootpls(rock.bootYX3,indice=2:7,xaxisticks=FALSE)
```

```
plots.confints.bootpls(confints.bootpls(rock.bootYX3), legendpos = "topright",
                       xaxisticks=FALSE,indice=2:7,type="BCa")
```

Bootstrap $(\mathbf{y}, \mathbf{T})$

Same specifications as for those used above for Bootstrap $(\mathbf{y}, \mathbf{X})$. We display CIs for each of the predictors (see Figure 59) and boxplots (see Figure 58). The `stabvalue` prevents the use of divergent estimates of models coefficients by discarding the bootstrap sample that led to such an instability in ML estimation.

```
rock.bootYT3<- bootplsglm(modpls3, R=1000, stabvalue=1e10, sim="antithetic")
rownames(rock.bootYT3$t0)<-c("peri\n","shape\n","perm\n","peri.\nshape","peri.\nperm",
                             "shape.\nperm")
```

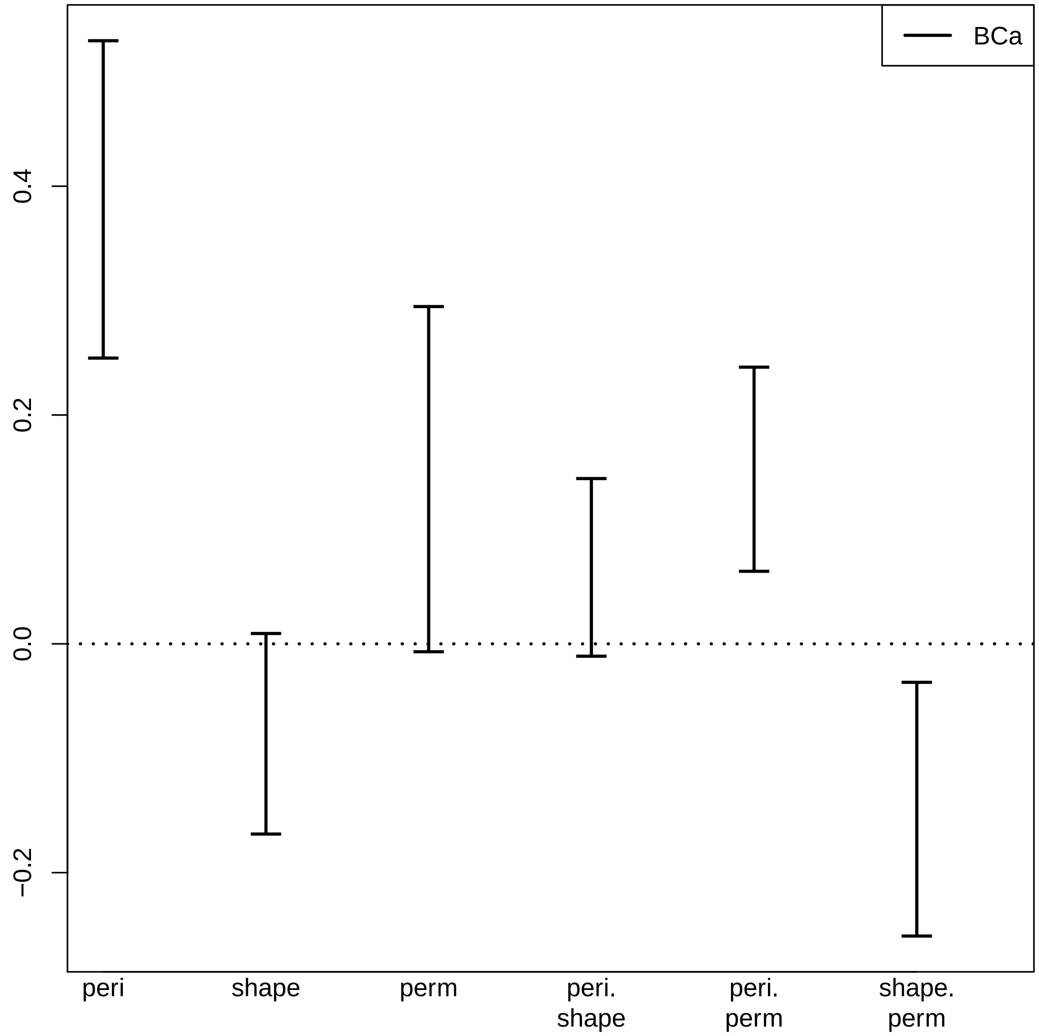

Figure 57: CI of the coefficients of the predictors, bootstrap $(\mathbf{y}, \mathbf{T})$, R=1000

```
boxplots.bootpls(rock.bootYT3,xaxisticks=FALSE,ranget0=TRUE)
```

```
plots.confints.bootpls(confints.bootpls(rock.bootYT3), legendpos = "topright",
                       xaxisticks=FALSE,type="BCa")
```

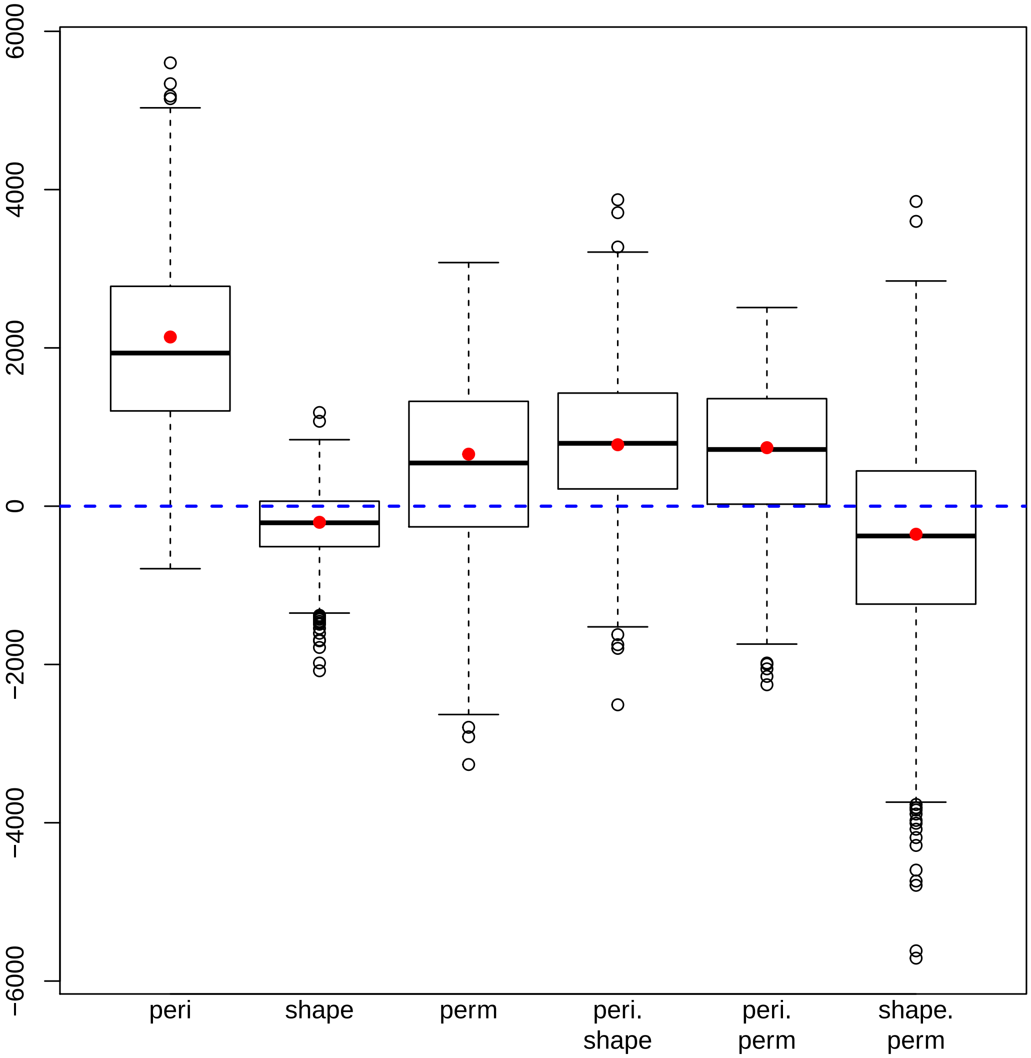

Figure 58: Bootstrap $(\mathbf{y}, \mathbf{T})$ distribution of the coefficients of the predictors, R=1000

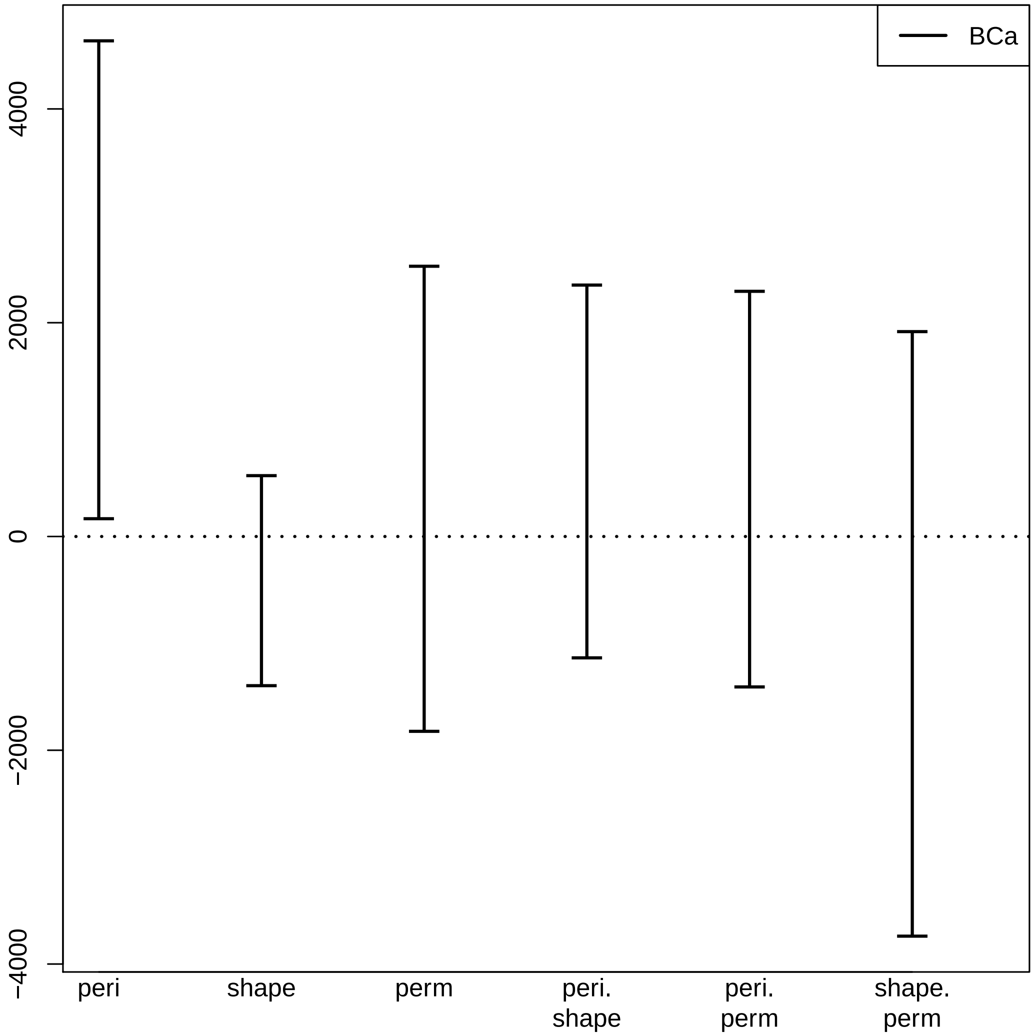

Figure 59: CI of the coefficients of the predictors, bootstrap $(\mathbf{y}, \mathbf{T})$, R=1000

4 Creating simulated datasets

To simulate PLSR datasets we implemented the results of Naes and Martens (1985) and Li et al. (2002) in the function `simul_data_YX` -multivariate response and predictors-. We then adapted it to get an univariate reponse and multivariate predictors in the function `simul_data_uniYX`. We derived logistic binary PLSGLR datasets by dichotomizing the PLSR ones.

4.1 Simulating PLSR datasets

Simulating, fitting and retrieving information criteria and leave one out cross validation results in two cases : 6 predictors (`dimX <- 6`) with 4 components (`Astar <- 4`) and 24 predictors (`dimX <- 6`) and 2 components (`Astar <- 2`).

```
rm(list = ls())
dimX <- 6
Astar <- 4
simul_data_UniYX(dimX,Astar)

##      Y     X1     X2     X3     X4     X5     X6
##  1.497 -4.331 -5.761  3.512 -6.811 -8.246  1.027

dataAstar <- as.data.frame(t(replicate(250,simul_data_UniYX(dimX,Astar))))
modpls <- plsR(Y~.,data=dataAstar,10,typeVC="standard")

## ____************************************************____
## ____TypeVC____ standard ____
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Component____ 5 ____
## ____Component____ 6 ____
## Warning : 1 2 3 4 5 6 < 10^{-12}
## Warning only 6 components could thus be extracted
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****

modpls

## Number of required components:
## [1] 10
## Number of successfully computed components:
## [1] 6
## Coefficients:
##                [,1]
## Intercept -0.002685
## X1        -0.591035
## X2         0.857294
## X3         0.954034
## X4         0.212147
## X5        -0.237734
## X6         0.031945
## Leave one out cross validated PRESS, Information criteria and Fit statistics:
##              AIC Q2cum_Y LimQ2_Y      Q2_Y  PRESS_Y     RSS_Y   R2_Y
## Nb_Comp_0 1701.8      NA      NA        NA       NA 13028.241     NA
## Nb_Comp_1 1221.3  0.8502  0.0975  0.850162 1952.128  1890.609 0.8549
## Nb_Comp_2  848.4  0.9658  0.0975  0.771734  431.562   422.053 0.9676
## Nb_Comp_3  375.1  0.9948  0.0975  0.847321   64.439    63.059 0.9952
## Nb_Comp_4 -258.9  0.9996  0.0975  0.920706    5.000     4.953 0.9996
## Nb_Comp_5 -258.1  0.9996  0.0975 -0.011338    5.009     4.929 0.9996
## Nb_Comp_6 -256.2  0.9996  0.0975 -0.007381    4.965     4.928 0.9996
##           R2_residY RSS_residY PRESS_residY Q2_residY  LimQ2
## Nb_Comp_0        NA  249.00000           NA        NA     NA
## Nb_Comp_1    0.8549   36.13393     37.30970  0.850162 0.0975
## Nb_Comp_2    0.9676    8.06641      8.24815  0.771734 0.0975
## Nb_Comp_3    0.9952    1.20520      1.23158  0.847321 0.0975
## Nb_Comp_4    0.9996    0.09467      0.09556  0.920706 0.0975
## Nb_Comp_5    0.9996    0.09420      0.09574 -0.011338 0.0975
## Nb_Comp_6    0.9996    0.09419      0.09489 -0.007381 0.0975
##           Q2cum_residY AIC.std DoF.dof sigmahat.dof  AIC.dof  BIC.dof
## Nb_Comp_0           NA   712.5   1.000       7.2334 52.53154 53.26854
## Nb_Comp_1       0.8502   231.9   3.673       2.7648  7.78706  8.18255
## Nb_Comp_2       0.9658  -141.0   4.445       1.3084  1.74908  1.85625
## Nb_Comp_3       0.9948  -614.2   4.912       0.5062  0.26230  0.28003
## Nb_Comp_4       0.9996 -1248.2   5.000       0.1419  0.02062  0.02204
## Nb_Comp_5       0.9996 -1247.5   7.000       0.1421  0.02085  0.02284
## Nb_Comp_6       0.9996 -1245.5   7.000       0.1421  0.02084  0.02283
##           GMDL.dof DoF.naive sigmahat.naive AIC.naive BIC.naive
## Nb_Comp_0   497.13         1         7.2334  52.53154  53.26854
## Nb_Comp_1   268.00         2         2.7611   7.68441   7.89917
## Nb_Comp_2    86.43         3         1.3072   1.72922   1.80143
## Nb_Comp_3  -144.74         4         0.5063   0.26044   0.27488
## Nb_Comp_4  -455.97         5         0.1422   0.02062   0.02205
## Nb_Comp_5  -444.99         6         0.1421   0.02068   0.02239
## Nb_Comp_6  -445.00         7         0.1424   0.02085   0.02285
##           GMDL.naive
## Nb_Comp_0     497.13
## Nb_Comp_1     263.26
## Nb_Comp_2      81.44
## Nb_Comp_3    -148.50
## Nb_Comp_4    -455.47
## Nb_Comp_5    -450.24
## Nb_Comp_6    -444.50
```

```
set.seed(123)
cv.modpls<-cv.plsR(Y~.,data=dataAstar,nt=10,K=10,NK=100)
```

The results, based on the use of the Q^2 criterion, (Fig. 60) retain 4 components as we aimed to simulate.

```
res.cv.modpls=cvtable(summary(cv.modpls))
```

```
## ____************************************************____
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Component____ 5 ____
## ____Component____ 6 ____
## Warning : 1 2 3 4 5 6 < 10^{-12}
## Warning only 6 components could thus be extracted
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****
##
##
## NK: 1,  2,  3,  4,  5,  6,  7,  8,  9,  10
## NK: 11,  12,  13,  14,  15,  16,  17,  18,  19,  20
## NK: 21,  22,  23,  24,  25,  26,  27,  28,  29,  30
## NK: 31,  32,  33,  34,  35,  36,  37,  38,  39,  40
## NK: 41,  42,  43,  44,  45,  46,  47,  48,  49,  50
## NK: 51,  52,  53,  54,  55,  56,  57,  58,  59,  60
## NK: 61,  62,  63,  64,  65,  66,  67,  68,  69,  70
## NK: 71,  72,  73,  74,  75,  76,  77,  78,  79,  80
## NK: 81,  82,  83,  84,  85,  86,  87,  88,  89,  90
## NK: 91,  92,  93,  94,  95,  96,  97,  98,  99,  100
##
##
## CV Q2 criterion:
##   0   1   2   3   4
##   0   0   0   0 100
##
## CV Press criterion:
##   1   2   3   4
##   0   0   0 100
```

```
plot(res.cv.modpls)
```

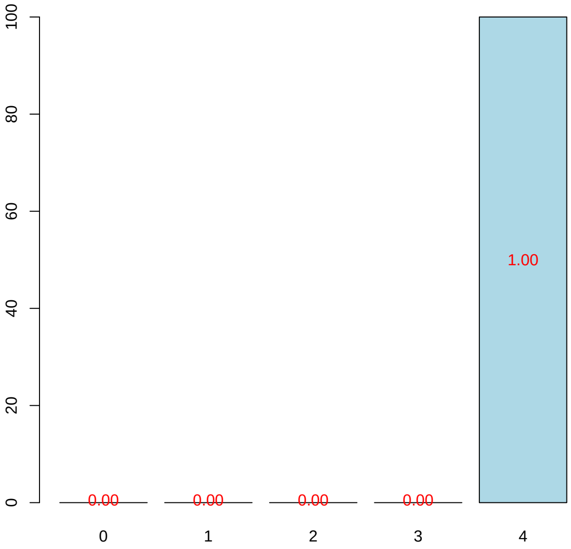

Figure 60: Nb components, 10-CV, n=100

```
rm(list = ls())
dimX <- 24
Astar <- 2
simul_data_UniYX(dimX,Astar)
```

```
##     Y    X1    X2    X3    X4    X5    X6    X7    X8    X9   X10   X11
## 5.208 2.208 2.224 1.774 2.196 2.215 1.775 2.196 2.189 1.770 2.201 2.212
##   X12   X13   X14   X15   X16   X17   X18   X19   X20   X21   X22   X23
## 1.770 2.225 2.224 1.775 2.201 2.194 1.764 2.196 2.217 1.763 2.211 2.214
##   X24
## 1.780
```

```
dataAstar2 <- as.data.frame(t(replicate(250,simul_data_UniYX(dimX,Astar))))
modpls2 <- plsR(Y˜.,data=dataAstar2,10,typeVC="standard")
```

```
## ____************************************************____
## ____TypeVC____ standard ____
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Component____ 5 ____
## ____Component____ 6 ____
## ____Component____ 7 ____
## ____Component____ 8 ____
## ____Component____ 9 ____
## ____Component____ 10 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****
```

```
modpls2
```

```
## Number of required components:
## [1] 10
## Number of successfully computed components:
## [1] 10
## Coefficients:
##               [,1]
## Intercept  0.01210
## X1         0.17939
## X2         0.39590
## X3         0.41792
## X4         1.78782
## X5         0.65121
## X6         0.17200
## X7         0.36370
## X8        -0.73833
## X9         1.11692
## X10        0.12659
## X11        1.21618
## X12        0.86323
## X13       -1.96520
## X14       -0.72444
## X15        0.52390
## X16       -0.23655
## X17        1.61505
## X18       -1.25339
## X19       -2.45084
## X20        0.64388
## X21        0.37566
## X22       -0.76251
## X23        0.46069
## X24        0.06193
## Leave one out cross validated PRESS, Information criteria and Fit statistics:
##               AIC Q2cum_Y LimQ2_Y     Q2_Y  PRESS_Y     RSS_Y   R2_Y
## Nb_Comp_0  1699.4      NA      NA       NA       NA 12900.449     NA
## Nb_Comp_1  1223.6  0.8477  0.0975  0.84774 1964.197  1908.309 0.8521
## Nb_Comp_2  -187.3  0.9995  0.0975  0.99646    6.764     6.701 0.9995
## Nb_Comp_3  -203.9  0.9994  0.0975 -0.07508    7.204     6.220 0.9995
## Nb_Comp_4  -203.5  0.9993  0.0975 -0.15684    7.196     6.182 0.9995
## Nb_Comp_5  -201.7  0.9992  0.0975 -0.17010    7.233     6.176 0.9995
## Nb_Comp_6  -199.8  0.9991  0.0975 -0.15470    7.131     6.175 0.9995
## Nb_Comp_7  -197.8  0.9990  0.0975 -0.14537    7.073     6.175 0.9995
## Nb_Comp_8  -195.8  0.9988  0.0975 -0.13746    7.024     6.175 0.9995
## Nb_Comp_9  -193.8  0.9987  0.0975 -0.13014    6.979     6.175 0.9995
## Nb_Comp_10 -191.8  0.9985  0.0975 -0.12363    6.939     6.175 0.9995
##            R2_residY RSS_residY PRESS_residY Q2_residY  LimQ2
## Nb_Comp_0         NA   249.0000           NA        NA     NA
## Nb_Comp_1     0.8521    36.8335      37.9122   0.84774 0.0975
## Nb_Comp_2     0.9995     0.1293       0.1305   0.99646 0.0975
## Nb_Comp_3     0.9995     0.1201       0.1391  -0.07508 0.0975
## Nb_Comp_4     0.9995     0.1193       0.1389  -0.15684 0.0975
## Nb_Comp_5     0.9995     0.1192       0.1396  -0.17010 0.0975
## Nb_Comp_6     0.9995     0.1192       0.1376  -0.15470 0.0975
## Nb_Comp_7     0.9995     0.1192       0.1365  -0.14537 0.0975
## Nb_Comp_8     0.9995     0.1192       0.1356  -0.13746 0.0975
## Nb_Comp_9     0.9995     0.1192       0.1347  -0.13014 0.0975
## Nb_Comp_10    0.9995     0.1192       0.1339  -0.12363 0.0975
##            Q2cum_residY AIC.std DoF.dof sigmahat.dof   AIC.dof  BIC.dof
## Nb_Comp_0            NA   712.5   1.000       7.1978 52.01627 52.74604
## Nb_Comp_1        0.8477   236.7   2.731       2.7724  7.80119  8.09692
## Nb_Comp_2        0.9995 -1174.2   3.000       0.1644  0.02745  0.02860
## Nb_Comp_3        0.9994 -1190.8  22.356       0.1649  0.02975  0.03832
## Nb_Comp_4        0.9993 -1190.4  23.557       0.1649  0.02985  0.03887
## Nb_Comp_5        0.9992 -1188.6  23.407       0.1647  0.02979  0.03873
## Nb_Comp_6        0.9991 -1186.6  23.659       0.1648  0.02984  0.03889
## Nb_Comp_7        0.9990 -1184.7  23.726       0.1648  0.02986  0.03894
## Nb_Comp_8        0.9988 -1182.7  23.749       0.1648  0.02986  0.03895
## Nb_Comp_9        0.9987 -1180.7  23.793       0.1649  0.02987  0.03898
## Nb_Comp_10       0.9985 -1178.7  23.597       0.1648  0.02983  0.03885
##            GMDL.dof DoF.naive sigmahat.naive AIC.naive BIC.naive
## Nb_Comp_0     495.4         1         7.1978  52.01627  52.74604
## Nb_Comp_1     266.2         2         2.7739   7.75635   7.97313
## Nb_Comp_2    -430.7         3         0.1647   0.02746   0.02860
## Nb_Comp_3    -336.4         4         0.1590   0.02569   0.02712
## Nb_Comp_4    -331.2         5         0.1588   0.02574   0.02751
## Nb_Comp_5    -332.0         6         0.1591   0.02592   0.02806
## Nb_Comp_6    -330.8         7         0.1594   0.02612   0.02863
## Nb_Comp_7    -330.4         8         0.1597   0.02633   0.02921
## Nb_Comp_8    -330.3         9         0.1601   0.02655   0.02979
## Nb_Comp_9    -330.1        10         0.1604   0.02676   0.03038
## Nb_Comp_10   -331.1        11         0.1607   0.02697   0.03098
##            GMDL.naive
## Nb_Comp_0       495.4
## Nb_Comp_1       264.4
## Nb_Comp_2      -430.2
## Nb_Comp_3      -433.4
## Nb_Comp_4      -428.4
## Nb_Comp_5      -422.8
## Nb_Comp_6      -417.1
## Nb_Comp_7      -411.6
## Nb_Comp_8      -406.1
## Nb_Comp_9      -400.6
## Nb_Comp_10     -395.3
```

```
set.seed(123)
cv.modpls2<-cv.plsR(Y~.,data=dataAstar2,nt=10,K=10,NK=100)
```

The results, based on the use of the Q^2 criterion, (Fig. 61) retain 2 components as we aimed to simulate.

```
res.cv.modpls2=cvtable(summary(cv.modpls2))

## ____************************************************____
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Component____ 5 ____
## ____Component____ 6 ____
## ____Component____ 7 ____
## ____Component____ 8 ____
## ____Component____ 9 ____
## ____Component____ 10 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****
##
##
## NK: 1,  2,  3,  4,  5,  6,  7,  8,  9,  10
## NK: 11,  12,  13,  14,  15,  16,  17,  18,  19,  20
## NK: 21,  22,  23,  24,  25,  26,  27,  28,  29,  30
## NK: 31,  32,  33,  34,  35,  36,  37,  38,  39,  40
## NK: 41,  42,  43,  44,  45,  46,  47,  48,  49,  50
## NK: 51,  52,  53,  54,  55,  56,  57,  58,  59,  60
## NK: 61,  62,  63,  64,  65,  66,  67,  68,  69,  70
## NK: 71,  72,  73,  74,  75,  76,  77,  78,  79,  80
## NK: 81,  82,  83,  84,  85,  86,  87,  88,  89,  90
## NK: 91,  92,  93,  94,  95,  96,  97,  98,  99,  100
##
##
## CV Q2 criterion:
##   0   1   2
##   0   0 100
##
## CV Press criterion:
##   1   2
##   0 100
plot(res.cv.modpls2)
```

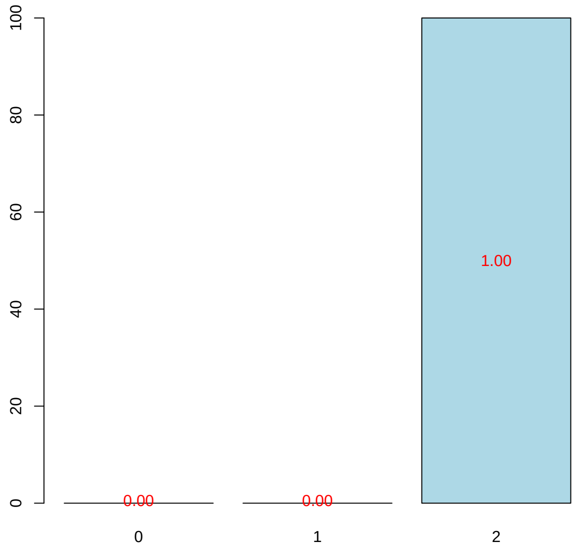

Figure 61: Nb components, 10-CV, n=100

4.2 Simulating logistic binary PLSGLR datasets

4.2.1 Continuous covariables

Simulating, fitting and retrieving information criteria and leave one out cross validation results.

```
ydataAstar2 <- dataAstar2[,1]
XdataAstar2 <- dataAstar2[,2:(dimX+1)]
ysimbin1 <- dicho(ydataAstar2)
res <- plsR(ysimbin1,XdataAstar2,10,typeVC="standard",MClassed=TRUE)
```

```
## ____************************************************____
## ____TypeVC____ standard ____
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Component____ 5 ____
## ____Component____ 6 ____
## ____Component____ 7 ____
## ____Component____ 8 ____
## ____Component____ 9 ____
## ____Component____ 10 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****
```

```
res$MissClassed
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
## [1,]  117   33    5   10    9    9    9    9    9     9     9
```

```
res
```

```
## Number of required components:
## [1] 10
## Number of successfully computed components:
## [1] 10
## Coefficients:
##               [,1]
## Intercept  0.47499
## X1         0.77223
## X2        -1.16091
## X3         2.97062
## X4         2.83936
## X5         0.03827
## X6        -0.52026
## X7         0.91260
## X8        -3.56136
## X9        -1.60262
## X10       -0.19633
## X11        1.88400
## X12       -1.05388
## X13       -0.29320
## X14       -1.17259
## X15        0.92728
## X16       -0.39698
## X17        0.51021
## X18       -2.52240
## X19        1.07467
## X20        0.90634
## X21        2.88766
## X22       -2.19775
## X23        0.08008
## X24       -0.96551
## Leave one out cross validated PRESS, Information criteria and Fit statistics:
```

```
##              AIC  Q2cum_Y LimQ2_Y    Q2_Y PRESS_Y RSS_Y   R2_Y
## Nb_Comp_0  365.9       NA      NA      NA      NA 62.24     NA
## Nb_Comp_1  178.8  0.52526  0.0975  0.5253   29.55 29.22 0.5305
## Nb_Comp_2  126.6  0.61435  0.0975  0.1877   23.74 23.53 0.6220
## Nb_Comp_3  114.0  0.57504  0.0975 -0.1020   25.93 22.18 0.6436
## Nb_Comp_4  114.4  0.49648  0.0975 -0.1848   26.29 22.05 0.6457
## Nb_Comp_5  116.3  0.42135  0.0975 -0.1492   25.34 22.03 0.6460
## Nb_Comp_6  118.2  0.33345  0.0975 -0.1519   25.38 22.03 0.6460
## Nb_Comp_7  120.2  0.23723  0.0975 -0.1444   25.21 22.03 0.6460
## Nb_Comp_8  122.2  0.13444  0.0975 -0.1348   25.00 22.03 0.6460
## Nb_Comp_9  124.2  0.02481  0.0975 -0.1267   24.82 22.03 0.6460
## Nb_Comp_10 126.2 -0.09037  0.0975 -0.1181   24.64 22.03 0.6460
##            MissClassed R2_residY RSS_residY PRESS_residY Q2_residY
## Nb_Comp_0          117        NA     249.00           NA        NA
## Nb_Comp_1           33    0.5305     116.90       118.21    0.5253
## Nb_Comp_2            5    0.6220      94.12        94.96    0.1877
## Nb_Comp_3           10    0.6436      88.75       103.72   -0.1020
## Nb_Comp_4            9    0.6457      88.21       105.15   -0.1848
## Nb_Comp_5            9    0.6460      88.15       101.37   -0.1492
## Nb_Comp_6            9    0.6460      88.14       101.54   -0.1519
## Nb_Comp_7            9    0.6460      88.14       100.87   -0.1444
## Nb_Comp_8            9    0.6460      88.14       100.02   -0.1348
## Nb_Comp_9            9    0.6460      88.14        99.31   -0.1267
## Nb_Comp_10           9    0.6460      88.14        98.55   -0.1181
##             LimQ2 Q2cum_residY AIC.std DoF.dof sigmahat.dof AIC.dof
## Nb_Comp_0      NA           NA   712.5   1.000       0.5000 0.25098
## Nb_Comp_1  0.0975      0.52526   525.4   2.666       0.3430 0.11939
## Nb_Comp_2  0.0975      0.61435   473.2   3.000       0.3080 0.09639
## Nb_Comp_3  0.0975      0.57504   460.6  23.642       0.3124 0.10720
## Nb_Comp_4  0.0975      0.49648   461.0  22.705       0.3108 0.10575
## Nb_Comp_5  0.0975      0.42135   462.9  22.771       0.3107 0.10572
## Nb_Comp_6  0.0975      0.33345   464.8  22.831       0.3108 0.10577
## Nb_Comp_7  0.0975      0.23723   466.8  22.914       0.3108 0.10584
## Nb_Comp_8  0.0975      0.13444   468.8  22.991       0.3109 0.10591
## Nb_Comp_9  0.0975      0.02481   470.8  23.119       0.3109 0.10602
## Nb_Comp_10 0.0975     -0.09037   472.8  23.207       0.3110 0.10609
##            BIC.dof GMDL.dof DoF.naive sigmahat.naive AIC.naive
## Nb_Comp_0   0.2545   -167.8         1         0.5000   0.25098
## Nb_Comp_1   0.1238   -257.2         2         0.3433   0.11877
## Nb_Comp_2   0.1004   -283.0         3         0.3086   0.09640
## Nb_Comp_3   0.1397   -244.2         4         0.3003   0.09163
## Nb_Comp_4   0.1366   -246.6         5         0.3000   0.09180
## Nb_Comp_5   0.1367   -246.6         6         0.3005   0.09247
## Nb_Comp_6   0.1368   -246.5         7         0.3011   0.09321
## Nb_Comp_7   0.1370   -246.3         8         0.3017   0.09396
## Nb_Comp_8   0.1372   -246.2         9         0.3024   0.09472
## Nb_Comp_9   0.1375   -245.9        10         0.3030   0.09548
## Nb_Comp_10  0.1377   -245.8        11         0.3036   0.09625
##            BIC.naive GMDL.naive
## Nb_Comp_0    0.25450     -167.8
## Nb_Comp_1    0.12209     -258.6
## Nb_Comp_2    0.10042     -282.5
## Nb_Comp_3    0.09671     -286.8
## Nb_Comp_4    0.09814     -284.9
## Nb_Comp_5    0.10010     -282.3
## Nb_Comp_6    0.10215     -279.8
## Nb_Comp_7    0.10422     -277.3
## Nb_Comp_8    0.10631     -274.9
## Nb_Comp_9    0.10841     -272.6
## Nb_Comp_10   0.11053     -270.3
```

Raw (non-constrained to lay between 0 and 1) estimations of probabilities are available in the `res$Probs` object. Truncated to $[0;1]$ estimations are available in the `res$Probs.trc` object.

```
res$Probs
res$Probs.trc
```

4.2.2 Dichotomous only covariables

For set the same frame as for the allelotyping study aze (Meyer et al., 2010), just dichotomize the predictors. Fitting and retrieving information criteria and leave one out cross validation results.

```
bindataAstar2 <- as.data.frame(dicho(dataAstar2))
resdicho <- plsR(Y~.,data=bindataAstar2,10,typeVC="standard",MClassed=TRUE)

## ____************************************************____
## ____TypeVC____ standard ____
## ____Component____ 1 ____
## ____Component____ 2 ____
## Warning : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 < 10^{-12}
## Warning only 2 components could thus be extracted
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****

resdicho$MissClassed

##      [,1] [,2] [,3]
## [1,]  117   13   13

resdicho

## Number of required components:
## [1] 10
## Number of successfully computed components:
## [1] 2
## Coefficients:
##                [,1]
## Intercept 0.011185
## X1        0.007899
## X2        0.007899
## X3        0.109405
## X4        0.007899
## X5        0.007899
## X6        0.109405
## X7        0.007899
## X8        0.007899
## X9        0.109405
## X10       0.007899
## X11       0.007899
## X12       0.109405
## X13       0.007899
## X14       0.007899
## X15       0.109405
## X16       0.007899
## X17       0.007899
## X18       0.109405
## X19       0.007899
## X20       0.007899
## X21       0.109405
## X22       0.007899
## X23       0.007899
## X24       0.109405
## Leave one out cross validated PRESS, Information criteria and Fit statistics:
##              AIC Q2cum_Y LimQ2_Y   Q2_Y PRESS_Y RSS_Y   R2_Y
## Nb_Comp_0 365.87      NA      NA     NA      NA 62.24     NA
## Nb_Comp_1  61.16  0.6999  0.0975 0.6999   18.68 18.25 0.7068
## Nb_Comp_2 -56.53  0.8106  0.0975 0.3689   11.52 11.31 0.8183
##           MissClassed R2_residY RSS_residY PRESS_residY Q2_residY
## Nb_Comp_0         117        NA     249.00           NA        NA
## Nb_Comp_1          13    0.7068      73.01        74.71    0.6999
## Nb_Comp_2          13    0.8183      45.23        46.08    0.3689
##            LimQ2 Q2cum_residY AIC.std DoF.dof sigmahat.dof AIC.dof
## Nb_Comp_0     NA           NA   712.5    1.00       0.5000 0.25098
## Nb_Comp_1 0.0975       0.6999   407.8    2.86       0.2712 0.07469
## Nb_Comp_2 0.0975       0.8106   290.1    3.00       0.2135 0.04632
##           BIC.dof GMDL.dof DoF.naive sigmahat.naive AIC.naive BIC.naive
## Nb_Comp_0 0.25450   -167.8         1         0.5000   0.25098   0.25450
## Nb_Comp_1 0.07765   -314.7         2         0.2713   0.07418   0.07626
## Nb_Comp_2 0.04825   -373.3         3         0.2140   0.04633   0.04826
##           GMDL.naive
## Nb_Comp_0     -167.8
## Nb_Comp_1     -316.9
## Nb_Comp_2     -372.8
```

Raw (non-constrained to lay between 0 and 1) estimations of probabilities are available in the `resdicho$Probs` object. Truncated to [0;1] estimations are available in the `resdicho$Probs.trc` object.

```
resdicho$Probs
resdicho$Probs.trc
```

5 Discussion

5.1 New classes

- `"plsRmodel"`: a PLSR model fitted with the `plsR` function.
- `"plsRglmmodel"`: a PLSR model fitted with the `plsRglm` function.
- `"coef.plsRmodel"`: coefficients of a PLSR model (of class `"plsRmodel"`) extracted with the `coef` function.
- `"coef.plsRglmmodel"`: coefficients of a PLSGLR model (of class `"plsRglmmodel"`) extracted with the `coef` function.
- `"summary.plsRmodel"`: summary of a PLSR model (of class `"plsRmodel"`) derived using the `summary` function.
- `"summary.plsRglmmodel"`: summary of a PLSGLR model (of class `"plsRglmmodel"`) derived using the `summary` function.
- `"cv.plsRmodel"`: crossvalidated PLSR model derived using the `cv.plsR` function.
- `"cv.plsRglmmodel"`: crossvalidated PLSGLR model derived using the `cv.plsRglm` function.
- `"summary.cv.plsRmodel"`: summary of a crossvalidated PLSR model (of class `"cv.plsRmodel"`) derived using the `summary` function.
- `"summary.cv.plsRglmmodel"`: summary of a crossvalidated PLSGLR model (of class `"cv.plsRglmmodel"`) derived using the `summary` function.
- `"table.summary.cv.plsRmodel"`: contingency table of the summary of a crossvalidated PLSR model (of class `"summary.cv.plsRmodel"`) derived using the `cvtable` function.
- `"table.summary.cv.plsRglmmodel"`: contingency table of the summary of a crossvalidated PLSGLR model (of class `"summary.cv.plsRglmmodel"`) derived using the `cvtable` function.

5.2 New generics

- `S3method("coef", "plsRmodel")`
- `S3method("coef", "plsRglmmodel")`
- `S3method("plot", "table.summary.cv.plsRglmmodel")`

- S3method("plot", "table.summary.cv.plsRmodel")
- S3method("print", "coef.plsRmodel")
- S3method("print", "coef.plsRglmmodel")
- S3method("print", "cv.plsRmodel")
- S3method("print", "cv.plsRglmmodel")
- S3method("summary", "cv.plsRmodel")
- S3method("summary", "cv.plsRglmmodel")
- S3method("predict", "plsRmodel")
- S3method("predict", "plsRglmmodel")
- S3method("print", "plsRmodel")
- S3method("print", "plsRglmmodel")
- S3method("summary", "plsRmodel")
- S3method("summary", "plsRglmmodel")
- S3method("print", "summary.plsRmodel")
- S3method("print", "summary.plsRglmmodel")

5.3 Validation of the results of the package

The package was tested in order to validate its results. The following lines of code generate outputs that can be checked against the relevant reference.

Comparing the PLSR results with SIMCA results in Tenenhaus's book (Tenenhaus, 1998)

```
rm(list = ls())
data(Cornell)
XCornell<-Cornell[,1:7]
yCornell<-Cornell[,8]
modpls<-plsR(yCornell,XCornell,3)

## ____************************************************____
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****

modpls

## Number of required components:
## [1] 3
## Number of successfully computed components:
## [1] 3
## Coefficients:
##              [,1]
## Intercept  92.676
## X1         -9.828
## X2         -6.960
## X3        -16.666
## X4         -8.422
## X5         -4.389
## X6         10.161
## X7        -34.529
## Information criteria and Fit statistics:
##             AIC   RSS_Y   R2_Y R2_residY RSS_residY AIC.std DoF.dof
## Nb_Comp_0 82.01 467.797     NA        NA    11.0000  37.010   1.000
## Nb_Comp_1 53.15  35.742 0.9236    0.9236     0.8405   8.150   2.741
## Nb_Comp_2 41.08  11.067 0.9763    0.9763     0.2602  -3.919   5.086
## Nb_Comp_3 32.06   4.418 0.9906    0.9906     0.1039 -12.938   5.121
##           sigmahat.dof AIC.dof BIC.dof GMDL.dof DoF.naive
## Nb_Comp_0       6.5213 46.0709 47.7894    27.59         1
## Nb_Comp_1       1.8665  4.5700  4.9558    21.34         2
## Nb_Comp_2       1.1825  2.1075  2.3949    27.40         3
## Nb_Comp_3       0.7488  0.8468  0.9628    24.41         4
##           sigmahat.naive AIC.naive BIC.naive GMDL.naive
## Nb_Comp_0         6.5213   46.0709   47.7894      27.59
## Nb_Comp_1         1.8906    4.1700    4.4588      18.38
## Nb_Comp_2         1.1089    1.5370    1.6861      17.71
## Nb_Comp_3         0.7431    0.7363    0.8256      19.01
```

```
modpls$uscores
```

```
##       [,1]     [,2]    [,3]
## 1   3.2183  2.05967  3.2801
## 2   2.9320  0.80716  0.4196
## 3   2.5502  0.38681 -1.4306
## 4   1.0869 -1.67717 -0.2158
## 5  -0.6309 -0.99337 -2.0551
## 6   0.8324 -1.37916  0.1155
## 7  -2.1261  0.13796  0.8375
## 8  -1.7443  0.43983  0.8989
## 9  -1.9670  0.21280  0.1701
## 10 -1.7125  0.35871  0.1068
## 11 -2.2851 -0.36863  0.2438
## 12 -0.1538  0.01538 -2.3710
```

```
modpls$pp
```

```
##      Comp_1   Comp_2  Comp_3
## X1 -0.45356 -0.04251  0.2730
## X2  0.03169 -1.00323  0.4493
## X3 -0.45436 -0.03900  0.2707
## X4 -0.35605  0.27807 -0.5332
## X5  0.29431 -0.04544 -0.4953
## X6  0.46197  0.43955  0.1054
## X7 -0.41254  0.47679 -0.3389
```

```
modpls$Coeffs
```

```
##              [,1]
## Intercept  92.676
## X1         -9.828
## X2         -6.960
## X3        -16.666
## X4         -8.422
## X5         -4.389
## X6         10.161
## X7        -34.529
```

```
modpls2<-plsR(yCornell,XCornell,4,typeVC="standard")
```

```
## ____************************************************____
## ____TypeVC____ standard ____
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****
```

```
modpls2
```

```
## Number of required components:
## [1] 4
## Number of successfully computed components:
## [1] 4
## Coefficients:
##              [,1]
## Intercept  92.480
## X1         -9.405
## X2         -6.941
## X3        -15.797
## X4         -8.769
## X5         -2.670
## X6         10.206
## X7        -32.368
## Leave one out cross validated PRESS, Information criteria and Fit statistics:
##             AIC Q2cum_Y LimQ2_Y    Q2_Y PRESS_Y   RSS_Y   R2_Y
## Nb_Comp_0 82.01      NA      NA      NA      NA 467.797     NA
## Nb_Comp_1 53.15  0.8967  0.0975  0.8967  48.344  35.742 0.9236
## Nb_Comp_2 41.08  0.9175  0.0975  0.2021  28.519  11.067 0.9763
## Nb_Comp_3 32.06  0.9400  0.0975  0.2720   8.057   4.418 0.9906
## Nb_Comp_4 33.76  0.9197  0.0975 -0.3376   5.910   4.309 0.9908
##           R2_residY RSS_residY PRESS_residY Q2_residY  LimQ2
## Nb_Comp_0        NA    11.0000           NA        NA     NA
## Nb_Comp_1    0.9236     0.8405       1.1368    0.8967 0.0975
## Nb_Comp_2    0.9763     0.2602       0.6706    0.2021 0.0975
## Nb_Comp_3    0.9906     0.1039       0.1895    0.2720 0.0975
## Nb_Comp_4    0.9908     0.1013       0.1390   -0.3376 0.0975
##           Q2cum_residY AIC.std DoF.dof sigmahat.dof AIC.dof BIC.dof
## Nb_Comp_0           NA  37.010   1.000       6.5213 46.0709 47.7894
## Nb_Comp_1       0.8967   8.150   2.741       1.8665  4.5700  4.9558
## Nb_Comp_2       0.9175  -3.919   5.086       1.1825  2.1075  2.3949
## Nb_Comp_3       0.9400 -12.938   5.121       0.7488  0.8468  0.9628
## Nb_Comp_4       0.9197 -11.237   5.103       0.7387  0.8233  0.9358
##           GMDL.dof DoF.naive sigmahat.naive AIC.naive BIC.naive
## Nb_Comp_0    27.59         1         6.5213   46.0709   47.7894
## Nb_Comp_1    21.34         2         1.8906    4.1700    4.4588
## Nb_Comp_2    27.40         3         1.1089    1.5370    1.6861
## Nb_Comp_3    24.41         4         0.7431    0.7363    0.8256
## Nb_Comp_4    24.23         5         0.7846    0.8721    0.9965
##           GMDL.naive
## Nb_Comp_0      27.59
## Nb_Comp_1      18.38
## Nb_Comp_2      17.71
## Nb_Comp_3      19.01
## Nb_Comp_4      24.17
```

```
modpls2$press.ind
```

```
##         [,1]      [,2]      [,3]      [,4]
## 1  0.5138445 2.765e-01 5.116e-02 0.0417100
## 2  0.0980315 4.634e-03 1.187e-03 0.0001432
## 3  0.0409695 2.452e-02 1.876e-02 0.0253325
## 4  0.2070668 2.516e-04 8.691e-04 0.0003176
## 5  0.1064217 1.681e-01 5.298e-02 0.0094285
## 6  0.1041921 1.059e-01 8.668e-03 0.0025259
## 7  0.0014598 7.563e-03 3.869e-03 0.0036997
## 8  0.0170245 8.904e-03 6.584e-03 0.0055489
## 9  0.0040074 3.409e-04 3.675e-04 0.0004262
## 10 0.0105422 1.559e-04 8.417e-05 0.0002713
## 11 0.0332006 4.808e-05 7.562e-03 0.0082795
## 12 0.0000273 7.365e-02 3.736e-02 0.0412779
```

```
modpls2$press.tot
```

```
## [1] 1.1368 0.6706 0.1895 0.1390
```

```
data(pine)
Xpine<-pine[,1:10]
ypine<-pine[,11]
modpls3<-plsR(ypine,Xpine,4)
```

```
## ____************************************************____
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****
```

```
modpls3
```

```
## Number of required components:
## [1] 4
## Number of successfully computed components:
## [1] 4
## Coefficients:
##                [,1]
## Intercept  8.350614
## x1        -0.002762
## x2        -0.037978
## x3         0.022306
## x4        -0.228437
## x5         0.073671
## x6         0.250399
## x7        -0.506743
## x8        -0.088513
## x9        -0.724570
## x10       -0.383506
## Information criteria and Fit statistics:
##             AIC  RSS_Y   R2_Y R2_residY RSS_residY AIC.std DoF.dof
## Nb_Comp_0 82.42 20.800     NA        NA      32.00   96.63   1.000
## Nb_Comp_1 63.62 11.075 0.4676    0.4676      17.04   77.83   3.176
## Nb_Comp_2 58.48  8.919 0.5712    0.5712      13.72   72.69   7.134
## Nb_Comp_3 56.55  7.920 0.6192    0.6192      12.18   70.77   8.778
## Nb_Comp_4 54.35  6.973 0.6648    0.6648      10.73   68.57   8.428
##           sigmahat.dof AIC.dof BIC.dof GMDL.dof DoF.naive
## Nb_Comp_0       0.8062  0.6697  0.6992   -3.605         1
## Nb_Comp_1       0.5994  0.4048  0.4565   -9.875         2
## Nb_Comp_2       0.5762  0.4138  0.5212   -6.986         3
## Nb_Comp_3       0.5604  0.4071  0.5321   -6.261         4
## Nb_Comp_4       0.5222  0.3506  0.4548   -8.153         5
##           sigmahat.naive AIC.naive BIC.naive GMDL.naive
## Nb_Comp_0         0.8062    0.6697    0.6992     -3.605
## Nb_Comp_1         0.5977    0.3789    0.4113    -11.451
## Nb_Comp_2         0.5453    0.3243    0.3648    -12.823
## Nb_Comp_3         0.5226    0.3062    0.3557    -12.757
## Nb_Comp_4         0.4990    0.2867    0.3432    -12.812
```

```
modpls3$Std.Coeffs
```

```
##              [,1]
## Intercept  0.0000
## x1        -0.4420
## x2        -0.3440
## x3         0.2638
## x4        -0.2949
## x5         0.3932
## x6         0.2228
## x7        -0.1176
## x8        -0.2582
## x9        -0.5091
## x10       -0.1219
```

```
modpls3$Coeffs
```

```
##                [,1]
## Intercept  8.350614
## x1        -0.002762
## x2        -0.037978
## x3         0.022306
## x4        -0.228437
## x5         0.073671
## x6         0.250399
## x7        -0.506743
## x8        -0.088513
## x9        -0.724570
## x10       -0.383506
```

```
modpls4<-plsR(ypine,Xpine,1)
```

```
## ____************************************************____
## ____Component____ 1 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****
```

```
modpls4
```

```
## Number of required components:
## [1] 1
## Number of successfully computed components:
## [1] 1
## Coefficients:
##                 [,1]
## Intercept  4.1382957
## x1        -0.0007545
## x2        -0.0114507
## x3        -0.0108577
## x4        -0.0631435
## x5        -0.0067332
## x6        -0.1459628
## x7        -0.2077726
## x8        -0.0430294
## x9        -0.2061096
## x10       -0.0887273
## Information criteria and Fit statistics:
##             AIC RSS_Y   R2_Y R2_residY RSS_residY AIC.std DoF.dof
## Nb_Comp_0 82.42 20.80     NA        NA      32.00   96.63   1.000
## Nb_Comp_1 63.62 11.07 0.4676    0.4676      17.04   77.83   3.176
##           sigmahat.dof AIC.dof BIC.dof GMDL.dof DoF.naive
## Nb_Comp_0       0.8062  0.6697  0.6992   -3.605         1
## Nb_Comp_1       0.5994  0.4048  0.4565   -9.875         2
##           sigmahat.naive AIC.naive BIC.naive GMDL.naive
## Nb_Comp_0         0.8062    0.6697    0.6992     -3.605
## Nb_Comp_1         0.5977    0.3789    0.4113    -11.451
```

```
modpls4$Std.Coeffs
```

```
##               [,1]
## Intercept  0.00000
## x1        -0.12076
## x2        -0.10373
## x3        -0.12843
## x4        -0.08151
## x5        -0.03593
## x6        -0.12987
## x7        -0.04823
## x8        -0.12552
## x9        -0.14482
## x10       -0.02821
```

```
modpls4$Coeffs
```

```
##                 [,1]
## Intercept  4.1382957
## x1        -0.0007545
## x2        -0.0114507
## x3        -0.0108577
## x4        -0.0631435
## x5        -0.0067332
## x6        -0.1459628
## x7        -0.2077726
## x8        -0.0430294
## x9        -0.2061096
## x10       -0.0887273
```

```
plsR(ypine,Xpine,10,typeVC="standard")$InfCrit
```

```
## ____************************************************____
## ____TypeVC____ standard ____
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Component____ 5 ____
## ____Component____ 6 ____
## ____Component____ 7 ____
## ____Component____ 8 ____
## ____Component____ 9 ____
## ____Component____ 10 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****
##              AIC  Q2cum_Y LimQ2_Y     Q2_Y PRESS_Y  RSS_Y   R2_Y
## Nb_Comp_0  82.42       NA      NA       NA      NA 20.800     NA
## Nb_Comp_1  63.62  0.38249  0.0975  0.38249  12.844 11.075 0.4676
## Nb_Comp_2  58.48  0.34836  0.0975 -0.05526  11.687  8.919 0.5712
## Nb_Comp_3  56.55  0.23688  0.0975 -0.17108  10.445  7.920 0.6192
## Nb_Comp_4  54.35  0.07000  0.0975 -0.21869   9.652  6.973 0.6648
## Nb_Comp_5  56.00 -0.07691  0.0975 -0.15796   8.074  6.899 0.6683
## Nb_Comp_6  57.70 -0.19969  0.0975 -0.11401   7.685  6.836 0.6714
## Nb_Comp_7  59.38 -0.27722  0.0975 -0.06463   7.277  6.770 0.6745
## Nb_Comp_8  61.21 -0.30603  0.0975 -0.02255   6.923  6.736 0.6762
## Nb_Comp_9  63.18 -0.39920  0.0975 -0.07134   7.217  6.730 0.6764
## Nb_Comp_10 65.16 -0.43744  0.0975 -0.02733   6.914  6.725 0.6767
##            R2_residY RSS_residY PRESS_residY Q2_residY  LimQ2
## Nb_Comp_0         NA      32.00           NA        NA     NA
## Nb_Comp_1     0.4676      17.04        19.76   0.38249 0.0975
## Nb_Comp_2     0.5712      13.72        17.98  -0.05526 0.0975
## Nb_Comp_3     0.6192      12.18        16.07  -0.17108 0.0975
## Nb_Comp_4     0.6648      10.73        14.85  -0.21869 0.0975
## Nb_Comp_5     0.6683      10.61        12.42  -0.15796 0.0975
## Nb_Comp_6     0.6714      10.52        11.82  -0.11401 0.0975
## Nb_Comp_7     0.6745      10.42        11.20  -0.06463 0.0975
## Nb_Comp_8     0.6762      10.36        10.65  -0.02255 0.0975
## Nb_Comp_9     0.6764      10.35        11.10  -0.07134 0.0975
## Nb_Comp_10    0.6767      10.35        10.64  -0.02733 0.0975
##            Q2cum_residY AIC.std DoF.dof sigmahat.dof AIC.dof BIC.dof
## Nb_Comp_0            NA   96.63   1.000       0.8062  0.6697  0.6992
## Nb_Comp_1       0.38249   77.83   3.176       0.5994  0.4048  0.4565
## Nb_Comp_2       0.34836   72.69   7.134       0.5762  0.4138  0.5212
## Nb_Comp_3       0.23688   70.77   8.778       0.5604  0.4071  0.5321
## Nb_Comp_4       0.07000   68.57   8.428       0.5222  0.3506  0.4548
## Nb_Comp_5      -0.07691   70.21   9.308       0.5286  0.3667  0.4846
## Nb_Comp_6      -0.19969   71.91   9.292       0.5260  0.3629  0.4795
## Nb_Comp_7      -0.27722   73.60   9.756       0.5285  0.3703  0.4938
## Nb_Comp_8      -0.30603   75.43  10.364       0.5338  0.3831  0.5171
## Nb_Comp_9      -0.39920   77.40  10.732       0.5378  0.3921  0.5329
## Nb_Comp_10     -0.43744   79.38  11.000       0.5407  0.3987  0.5446
##            GMDL.dof DoF.naive sigmahat.naive AIC.naive BIC.naive
## Nb_Comp_0    -3.605         1         0.8062    0.6697    0.6992
## Nb_Comp_1    -9.875         2         0.5977    0.3789    0.4113
## Nb_Comp_2    -6.986         3         0.5453    0.3243    0.3648
## Nb_Comp_3    -6.261         4         0.5226    0.3062    0.3557
## Nb_Comp_4    -8.153         5         0.4990    0.2867    0.3432
## Nb_Comp_5    -7.112         6         0.5055    0.3020    0.3715
## Nb_Comp_6    -7.233         7         0.5127    0.3187    0.4021
## Nb_Comp_7    -6.742         8         0.5204    0.3365    0.4347
## Nb_Comp_8    -6.038         9         0.5298    0.3572    0.4718
## Nb_Comp_9    -5.600        10         0.5410    0.3813    0.5140
## Nb_Comp_10   -5.288        11         0.5529    0.4076    0.5601
##            GMDL.naive
## Nb_Comp_0      -3.605
## Nb_Comp_1     -11.451
## Nb_Comp_2     -12.823
## Nb_Comp_3     -12.757
## Nb_Comp_4     -12.812
## Nb_Comp_5     -11.330
## Nb_Comp_6      -9.919
## Nb_Comp_7      -8.593
## Nb_Comp_8      -7.288
## Nb_Comp_9      -6.009
## Nb_Comp_10     -4.799
```

```
data(pine_full)
Xpine_full<-pine_full[,1:10]
ypine_full<-pine_full[,11]
modpls5<-plsR(log(ypine_full),Xpine_full,1)
```

```
## ____************************************************____
## ____Component____ 1 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****
```

```
modpls5
```

```
## Number of required components:
## [1] 1
## Number of successfully computed components:
## [1] 1
## Coefficients:
##                  [,1]
## Intercept  4.929e+00
## x1        -1.720e-03
## x2        -1.328e-02
## x3        -1.015e-02
## x4        -5.912e-02
## x5        -7.839e-05
## x6        -1.404e-01
## x7        -2.032e-01
## x8        -7.828e-02
## x9        -3.298e-01
## x10       -4.233e-01
## Information criteria and Fit statistics:
##              AIC  RSS_Y   R2_Y R2_residY RSS_residY AIC.std DoF.dof
## Nb_Comp_0 204.2 107.23     NA        NA      57.00   167.6   1.000
## Nb_Comp_1 187.7  77.87 0.2738    0.2738      41.39   151.0   3.678
##           sigmahat.dof AIC.dof BIC.dof GMDL.dof DoF.naive
## Nb_Comp_0        1.372   1.914   1.981    21.53         1
## Nb_Comp_1        1.186   1.521   1.705    16.08         2
##           sigmahat.naive AIC.naive BIC.naive GMDL.naive
## Nb_Comp_0          1.372     1.914     1.981      21.53
## Nb_Comp_1          1.179     1.439     1.537      14.46
```

```
modpls5$Std.Coeffs
```

```
##                 [,1]
## Intercept  0.0000000
## x1        -0.1566583
## x2        -0.0752549
## x3        -0.0709127
## x4        -0.0478586
## x5        -0.0003216
## x6        -0.0729950
## x7        -0.0337170
## x8        -0.1217004
## x9        -0.1355664
## x10       -0.0848364
```

```
modpls5$Coeffs
```

```
##                 [,1]
## Intercept  4.929e+00
## x1        -1.720e-03
## x2        -1.328e-02
## x3        -1.015e-02
## x4        -5.912e-02
## x5        -7.839e-05
## x6        -1.404e-01
## x7        -2.032e-01
## x8        -7.828e-02
## x9        -3.298e-01
## x10       -4.233e-01
```

```
cor(cbind(Xpine,ypine))
```

```
##              x1      x2      x3      x4       x5      x6       x7
## x1     1.00000  0.1205  0.5376  0.3211  0.28377  0.5147  0.26849
## x2     0.12052  1.0000  0.3219  0.1367  0.11342  0.3007 -0.15222
## x3     0.53756  0.3219  1.0000  0.4144  0.29492  0.9796  0.12847
## x4     0.32105  0.1367  0.4144  1.0000  0.90466  0.4393  0.05810
## x5     0.28377  0.1134  0.2949  0.9047  1.00000  0.3062 -0.07871
## x6     0.51467  0.3007  0.9796  0.4393  0.30623  1.0000  0.15068
## x7     0.26849 -0.1522  0.1285  0.0581 -0.07871  0.1507  1.00000
## x8     0.36015  0.2619  0.7590  0.7719  0.59620  0.8102  0.06001
## x9     0.36372  0.3257  0.8768  0.4596  0.26746  0.9085  0.06325
## x10   -0.09993  0.1293  0.2062 -0.0454 -0.02458  0.1301  0.13820
## ypine -0.53022 -0.4555 -0.5639 -0.3579 -0.15777 -0.5702 -0.21175
##              x8       x9      x10   ypine
## x1     0.36015  0.36372 -0.09993 -0.5302
## x2     0.26191  0.32567  0.12934 -0.4555
## x3     0.75896  0.87679  0.20618 -0.5639
## x4     0.77193  0.45959 -0.04540 -0.3579
## x5     0.59620  0.26746 -0.02458 -0.1578
## x6     0.81022  0.90853  0.13009 -0.5702
## x7     0.06001  0.06325  0.13820 -0.2117
## x8     1.00000  0.85364  0.05355 -0.5511
## x9     0.85364  1.00000  0.17452 -0.6359
## x10    0.05355  0.17452  1.00000 -0.1239
## ypine -0.55113 -0.63587 -0.12386  1.0000
```

```
XpineNAX21 <- Xpine
XpineNAX21[1,2] <- NA
modpls6<-plsR(ypine,XpineNAX21,4)
```

```
## ____************************************************____
## Only naive DoF can be used with missing data
## ____There are some NAs in X but not in Y____
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Predicting X with NA in X and not in Y____
## ****________________________________________________****
```

```
modpls6
```

```
## Number of required components:
## [1] 4
## Number of successfully computed components:
## [1] 4
## Coefficients:
##                [,1]
## Intercept  8.26449
## x1        -0.00280
## x2        -0.03772
## x3         0.02232
## x4        -0.23061
## x5         0.07459
## x6         0.25032
## x7        -0.43225
## x8        -0.09018
## x9        -0.73079
## x10       -0.36353
## Information criteria and Fit statistics:
##              AIC  RSS_Y   R2_Y R2_residY RSS_residY AIC.std
## Nb_Comp_0 82.42 20.800     NA        NA      32.00   96.63
## Nb_Comp_1 63.69 11.099 0.4664    0.4664      17.08   77.91
## Nb_Comp_2 58.35  8.886 0.5728    0.5728      13.67   72.57
## Nb_Comp_3 56.37  7.875 0.6214    0.6214      12.11   70.58
## Nb_Comp_4 54.02  6.904 0.6681    0.6681      10.62   68.24
```

```
modpls6$Std.Coeffs
```

```
##               [,1]
## Intercept  0.0000
## x1        -0.4482
## x2        -0.3419
## x3         0.2640
## x4        -0.2977
## x5         0.3981
## x6         0.2227
## x7        -0.1003
## x8        -0.2631
## x9        -0.5135
## x10       -0.1156
```

```
modpls6$YChapeau[1,]
```

```
##     1
## 2.063
```

```
modpls3$YChapeau[1,]
```

```
##     1
## 2.019
```

```
modpls6$CoeffC
```

```
## Coeff_Comp_Reg1 Coeff_Comp_Reg2 Coeff_Comp_Reg3 Coeff_Comp_Reg4
##          0.3259          0.3206          0.2795          0.3837
```

```
plsR(ypine,XpineNAX21,2,dataPredictY=XpineNAX21[1,])$ValsPredictY
```

```
## ____************************************************____
## Only naive DoF can be used with missing data
## ____There are some NAs in X but not in Y____
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Predicting X with NA in X and not in Y____
## ****________________________________________________****
##       [,1]
## [1,] 2.116
```

```
modpls7<-plsR(ypine,XpineNAX21,4,EstimXNA=TRUE)
```

```
## ____************************************************____
## Only naive DoF can be used with missing data
## ____There are some NAs in X but not in Y____
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Predicting X with NA in X and not in Y____
## ****________________________________________________****
```

```
modpls7$XChapeau
```

```
##      x1    x2       x3    x4     x5     x6    x7     x8    x9   x10
## 1  1135 17.71 -1.55495 4.618 17.306 0.8486 1.479  6.106 1.327 1.648
## 2  1289 28.04  7.90830 4.624 17.149 1.5220 1.603  6.882 1.661 1.629
## 3  1250 28.27  4.04605 2.767  9.770 1.1703 1.692  3.758 1.285 1.725
## 4  1303 27.63 17.52126 3.253 10.847 2.1871 1.737  6.421 2.063 1.892
## 5  1348 34.86  5.39130 3.930 12.914 1.3214 1.704  5.987 1.653 1.682
## 6  1266 28.96  0.65947 4.266 15.814 0.9760 1.594  5.455 1.287 1.567
## 7  1433 36.99 23.67253 3.129  8.218 2.6460 1.876  7.545 2.575 2.006
## 8  1405 37.58  5.32566 5.728 20.723 1.3677 1.611  8.157 1.715 1.468
## 9  1181 21.81  2.88849 3.335 11.169 1.1333 1.620  5.083 1.505 1.813
## 10 1276 27.24  9.65604 3.772 11.989 1.6470 1.682  6.686 1.941 1.849
## 11 1260 20.95 16.48623 5.668 18.533 2.2445 1.575 10.893 2.672 1.926
## 12 1395 27.99 31.86798 5.563 21.665 3.2947 1.651 10.645 2.660 1.738
## 13 1136 19.18  0.32731 3.036 11.267 0.9205 1.585  3.884 1.175 1.747
## 14 1239 25.77  1.37695 4.752 19.126 1.0315 1.525  5.742 1.175 1.475
## 15 1247 26.43  3.58144 4.147 15.176 1.1926 1.599  5.794 1.451 1.647
## 16 1502 38.07 29.19896 5.557 18.393 3.1295 1.764 11.388 2.985 1.830
## 17 1451 35.95 22.31620 5.619 20.183 2.6096 1.688 10.076 2.463 1.670
## 18 1253 25.63  5.23193 4.606 14.948 1.3638 1.607  7.598 1.921 1.790
## 19 1328 32.78  9.61238 3.044  9.807 1.5946 1.753  5.100 1.686 1.784
## 20 1430 36.32 18.92696 4.739 15.290 2.3502 1.747  9.031 2.447 1.800
## 21 1378 27.82 29.36000 5.108 18.668 3.1133 1.682 10.240 2.700 1.836
## 22 1318 28.05 16.03323 4.365 14.743 2.1271 1.678  8.066 2.207 1.829
## 23 1326 31.51  5.58479 4.967 17.261 1.3774 1.618  7.530 1.773 1.632
## 24 1383 31.25 21.66718 4.793 16.261 2.5509 1.707  9.287 2.495 1.828
## 25 1463 42.03 10.45350 4.878 16.759 1.7108 1.726  7.618 1.893 1.571
## 26 1302 29.44  6.08841 4.752 15.708 1.4213 1.628  7.677 1.907 1.729
## 27 1390 30.08 19.80133 6.653 23.023 2.4960 1.594 12.073 2.740 1.734
## 28 1236 28.53  0.08033 2.566  6.654 0.9087 1.723  4.044 1.475 1.869
## 29 1196 23.15  3.03239 3.307 10.802 1.1441 1.636  5.121 1.537 1.819
## 30 1265 24.41 14.03965 3.978 11.111 2.0070 1.696  8.365 2.462 2.036
## 31 1293 28.00  7.51088 5.069 19.458 1.5012 1.567  7.234 1.604 1.554
## 32 1250 22.53 10.77971 5.291 16.423 1.8201 1.591  9.906 2.473 1.924
## 33 1407 33.66 19.58236 5.059 16.211 2.4208 1.717  9.829 2.595 1.841
```

```
modpls7$XChapeauNA
```

```
##    [,1]  [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## 1     0 17.71    0    0    0    0    0    0    0     0
## 2     0  0.00    0    0    0    0    0    0    0     0
## 3     0  0.00    0    0    0    0    0    0    0     0
## 4     0  0.00    0    0    0    0    0    0    0     0
## 5     0  0.00    0    0    0    0    0    0    0     0
## 6     0  0.00    0    0    0    0    0    0    0     0
## 7     0  0.00    0    0    0    0    0    0    0     0
## 8     0  0.00    0    0    0    0    0    0    0     0
## 9     0  0.00    0    0    0    0    0    0    0     0
## 10    0  0.00    0    0    0    0    0    0    0     0
## 11    0  0.00    0    0    0    0    0    0    0     0
## 12    0  0.00    0    0    0    0    0    0    0     0
## 13    0  0.00    0    0    0    0    0    0    0     0
## 14    0  0.00    0    0    0    0    0    0    0     0
## 15    0  0.00    0    0    0    0    0    0    0     0
## 16    0  0.00    0    0    0    0    0    0    0     0
## 17    0  0.00    0    0    0    0    0    0    0     0
## 18    0  0.00    0    0    0    0    0    0    0     0
## 19    0  0.00    0    0    0    0    0    0    0     0
## 20    0  0.00    0    0    0    0    0    0    0     0
## 21    0  0.00    0    0    0    0    0    0    0     0
## 22    0  0.00    0    0    0    0    0    0    0     0
## 23    0  0.00    0    0    0    0    0    0    0     0
## 24    0  0.00    0    0    0    0    0    0    0     0
## 25    0  0.00    0    0    0    0    0    0    0     0
## 26    0  0.00    0    0    0    0    0    0    0     0
## 27    0  0.00    0    0    0    0    0    0    0     0
## 28    0  0.00    0    0    0    0    0    0    0     0
## 29    0  0.00    0    0    0    0    0    0    0     0
## 30    0  0.00    0    0    0    0    0    0    0     0
## 31    0  0.00    0    0    0    0    0    0    0     0
## 32    0  0.00    0    0    0    0    0    0    0     0
## 33    0  0.00    0    0    0    0    0    0    0     0
```

```
plsR(ypine,Xpine,10,typeVC="none")$InfCrit
plsR(ypine,Xpine,10,typeVC="standard")$InfCrit
plsR(ypine,Xpine,10,typeVC="adaptative")$InfCrit
plsR(ypine,Xpine,10,typeVC="missingdata")$InfCrit
plsR(ypine,XpineNAX21,10,typeVC="none")$InfCrit
plsR(ypine,XpineNAX21,10,typeVC="standard")$InfCrit
plsR(ypine,XpineNAX21,10,typeVC="adaptative")$InfCrit
plsR(ypine,XpineNAX21,10,typeVC="missingdata")$InfCrit
```

Comparing the ordinal logistic PLSR results with Tenenhaus (2005) and Bastien et al. (2005)

```
set.seed(12345)
data(bordeaux)
Xbordeaux<-bordeaux[,1:4]
ybordeaux<-factor(bordeaux$Quality,ordered=TRUE)
modpls <- plsRglm(ybordeaux,Xbordeaux,4,modele="pls-glm-polr")
```

```
## ____************************************************____
##
## Model: pls-glm-polr
## Method: logistic
##
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## ____Component____ 4 ____
## ____Predicting X without NA neither in X nor in Y____
## ****________________________________________________****
```

```
modpls
```

```
## Number of required components:
## [1] 4
## Number of successfully computed components:
## [1] 4
## Coefficients:
##                   [,1]
## 1|2         -85.50956
## 2|3         -80.55156
## Temperature   0.02427
## Sunshine      0.01379
## Heat         -0.08876
## Rain         -0.02590
## Information criteria and Fit statistics:
##             AIC   BIC Missclassed Chi2_Pearson_Y
## Nb_Comp_0 78.65 81.70          22         62.333
## Nb_Comp_1 36.50 41.08           6          9.357
## Nb_Comp_2 35.58 41.69           6          8.569
## Nb_Comp_3 36.27 43.90           7          8.281
## Nb_Comp_4 38.16 47.32           7          8.322
```

```
XbordeauxNA<-Xbordeaux
XbordeauxNA[1,1] <- NA
modplsNA <- plsRglm(ybordeaux,XbordeauxNA,10,modele="pls-glm-polr")
```

```
## ____************************************************____
## Only naive DoF can be used with missing data
##
## Model: pls-glm-polr
## Method: logistic
##
## ____There are some NAs in X but not in Y____
## ____Component____ 1 ____
## ____Component____ 2 ____
## ____Component____ 3 ____
## Warning : reciprocal condition number of t(cbind(res$pp,temppp)[XXNA[1,],,drop=FALSE])%*%cbind(res$pp,temppp)[
## Warning only 3 components could thus be extracted
## ____Predicting X with NA in X and not in Y____
## ****________________________________________________****
```

```
modplsNA
```

```
## Number of required components:
## [1] 10
## Number of successfully computed components:
## [1] 3
## Coefficients:
##                   [,1]
## 1|2         -89.16630
## 2|3         -84.11693
## Temperature   0.02461
## Sunshine      0.01535
## Heat         -0.09543
## Rain         -0.02399
## Information criteria and Fit statistics:
##             AIC   BIC Missclassed Chi2_Pearson_Y
## Nb_Comp_0 78.65 81.70          22         62.333
## Nb_Comp_1 36.21 40.79           6          9.454
## Nb_Comp_2 35.30 41.40           5          8.235
## Nb_Comp_3 35.82 43.45           7          7.803
```

5.4 Main features of the package

The usefulness of this new package on *R* is certain as complexity of the datasets is going to increase with the technological progress. Medicine, biology and chemistry are domains which will be most confronted to difficulties like strongly correlated predictors and often in number higher than the number of subjects. Indeed, these kinds of problems are inevitable during the establishment of a mixing model, spectrum analysis or genomic data analysis.

This package enable to analyze these databases in a much more comprehensive way, as we see in section Application, than what was already proposed with other *R* PLS packages. However, it's essential for the user to understand that methods and different criteria include in this package, like these to choose the best number of components or the bootstraps techniques, are quite approximate. Nevertheless, some of them seem to be quite good indicators.

The future aim is to develop some new criteria and methods to select a relevant number of components and assess significance of predictors more reliably, despite the high level of complexity of the dataset that statisticians will have to deal with in the near future.

6 Export results to LaTeX

Using the *xtable* package, the tables of results can easily and automatically be exported to LaTeX. This is interesting for those who strive to produce reproductible research studies. We provide here an example for the crossvalidation results.

```
library(xtable)
resCVtab1<-print(xtable(CVresults1[[1]][,c(1:6)],digits=c(0,1,1,0,0,-1,4),
                        caption="Cross-validation results, $k=8$, part one"))
```

```
## % latex table generated in R 3.1.0 by xtable 1.7-3 package
## % Thu Jun 26 10:39:00 2014
## \begin{table}[ht]
## \centering
## \begin{tabular}{rrrrrrr}
##   \hline
##  & AIC & BIC & MissClassed & CV\_MissClassed & Q2Chisqcum\_Y & limQ2 \\
##   \hline
## Nb\_Comp\_0 & 145.8 & 148.5 & 49 &  &  &  \\
##   Nb\_Comp\_1 & 119.1 & 124.3 & 30 & 50 & -2.8E+00 & 0.0975 \\
##   Nb\_Comp\_2 & 106.0 & 113.9 & 20 & 64 & -1.7E+01 & 0.0975 \\
##   Nb\_Comp\_3 & 100.3 & 110.9 & 18 & 48 & -1.9E+02 & 0.0975 \\
##   Nb\_Comp\_4 & 96.2 & 109.4 & 20 & 49 & -1.5E+04 & 0.0975 \\
##   Nb\_Comp\_5 & 94.2 & 110.0 & 18 & 47 & -2.5E+07 & 0.0975 \\
##   Nb\_Comp\_6 & 93.0 & 111.5 & 16 & 51 & -9.1E+10 & 0.0975 \\
##   Nb\_Comp\_7 & 94.1 & 115.3 & 17 & 47 & -1.2E+15 & 0.0975 \\
##   Nb\_Comp\_8 & 94.1 & 117.9 & 17 & 41 & -4.4E+21 & 0.0975 \\
##   Nb\_Comp\_9 & 93.4 & 119.9 & 16 & 43 & -4.2E+31 & 0.0975 \\
##   Nb\_Comp\_10 & 95.3 & 124.4 & 17 & 47 & -5.2E+45 & 0.0975 \\
##    \hline
## \end{tabular}
## \caption{Cross-validation results, $k=8$, part one}
## \end{table}
```

```
resCVtab2<-print(xtable(CVresults1[[1]][,c(7:11)],digits=c(0,-1,-1,1,1,3),
                        caption="Cross-validation results, $k=8$, part two"))
```

```
## % latex table generated in R 3.1.0 by xtable 1.7-3 package
## % Thu Jun 26 10:39:00 2014
## \begin{table}[ht]
## \centering
## \begin{tabular}{rrrrrr}
##   \hline
##  & Q2Chisq\_Y & PREChi2\_Pearson\_Y & Chi2\_Pearson\_Y & RSS\_Y & R2\_Y \\
##   \hline
## Nb\_Comp\_0 &  &  & 104.0 & 25.9 &  \\
##   Nb\_Comp\_1 & -2.8E+00 & 3.9E+02 & 101.7 & 19.5 & 0.246 \\
##   Nb\_Comp\_2 & -3.9E+00 & 5.0E+02 & 111.0 & 16.2 & 0.376 \\
##   Nb\_Comp\_3 & -9.4E+00 & 1.2E+03 & 102.5 & 14.9 & 0.427 \\
##   Nb\_Comp\_4 & -7.6E+01 & 7.9E+03 & 122.8 & 13.7 & 0.470 \\
##   Nb\_Comp\_5 & -1.7E+03 & 2.1E+05 & 148.7 & 13.0 & 0.497 \\
##   Nb\_Comp\_6 & -3.7E+03 & 5.4E+05 & 141.1 & 12.4 & 0.520 \\
##   Nb\_Comp\_7 & -1.4E+04 & 1.9E+06 & 149.1 & 12.2 & 0.531 \\
##   Nb\_Comp\_8 & -3.6E+06 & 5.3E+08 & 79.8 & 12.4 & 0.522 \\
##   Nb\_Comp\_9 & -9.5E+09 & 7.6E+11 & 73.3 & 12.1 & 0.532 \\
##   Nb\_Comp\_10 & -1.2E+14 & 9.0E+15 & 74.4 & 12.0 & 0.535 \\
##    \hline
## \end{tabular}
## \caption{Cross-validation results, $k=8$, part two}
## \end{table}
```

You can then either copy-paste this code into a LaTeXfile or apply the `\Sexpr` function to the two R objects `resCVtab1` and `resCVtab2` if you are using *knitr* .Rnw file.

	AIC	BIC	MissClassed	CV_MissClassed	Q2Chisqcum_Y	limQ2
Nb_Comp_0	145.8	148.5	49			
Nb_Comp_1	119.1	124.3	30	50	-2.8E+00	0.0975
Nb_Comp_2	106.0	113.9	20	64	-1.7E+01	0.0975
Nb_Comp_3	100.3	110.9	18	48	-1.9E+02	0.0975
Nb_Comp_4	96.2	109.4	20	49	-1.5E+04	0.0975
Nb_Comp_5	94.2	110.0	18	47	-2.5E+07	0.0975
Nb_Comp_6	93.0	111.5	16	51	-9.1E+10	0.0975
Nb_Comp_7	94.1	115.3	17	47	-1.2E+15	0.0975
Nb_Comp_8	94.1	117.9	17	41	-4.4E+21	0.0975
Nb_Comp_9	93.4	119.9	16	43	-4.2E+31	0.0975
Nb_Comp_10	95.3	124.4	17	47	-5.2E+45	0.0975

Table 1: Cross-validation results, $k = 8$, part one

With slightly more work, you get obtain these figures.

	Q2Chisq_Y	PREChi2_Pearson_Y	Chi2_Pearson_Y	RSS_Y	R2_Y
Nb_Comp_0			104.0	25.9	
Nb_Comp_1	-2.8E+00	3.9E+02	101.7	19.5	0.246
Nb_Comp_2	-3.9E+00	5.0E+02	111.0	16.2	0.376
Nb_Comp_3	-9.4E+00	1.2E+03	102.5	14.9	0.427
Nb_Comp_4	-7.6E+01	7.9E+03	122.8	13.7	0.470
Nb_Comp_5	-1.7E+03	2.1E+05	148.7	13.0	0.497
Nb_Comp_6	-3.7E+03	5.4E+05	141.1	12.4	0.520
Nb_Comp_7	-1.4E+04	1.9E+06	149.1	12.2	0.531
Nb_Comp_8	-3.6E+06	5.3E+08	79.8	12.4	0.522
Nb_Comp_9	-9.5E+09	7.6E+11	73.3	12.1	0.532
Nb_Comp_10	-1.2E+14	9.0E+15	74.4	12.0	0.535

Table 2: Cross-validation results, $k = 8$, part two

7 Session Information

```
## R version 3.1.0 (2014-04-10)
## Platform: x86_64-apple-darwin13.1.0 (64-bit)
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods
## [7] base
##
## other attached packages:
## [1] xtable_1.7-3  plsdof_0.2-6  MASS_7.3-33   plsRglm_1.0.0
## [5] knitr_1.6
##
## loaded via a namespace (and not attached):
##  [1] BiocStyle_1.2.0   bipartite_2.04    boot_1.3-11
##  [4] car_2.0-20        digest_0.6.4      evaluate_0.5.5
##  [7] fields_7.1        formatR_0.10      grid_3.1.0
## [10] highr_0.3         igraph_0.7.0      lattice_0.20-29
## [13] maps_2.3-7        mvtnorm_0.9-99992 nnet_7.3-8
## [16] permute_0.8-3     sna_2.3-2         spam_0.41-0
## [19] stringr_0.6.2     tools_3.1.0       vegan_2.0-10
```

References

V. B. Astler and F. A. Coller. The prognostic significance of direct extension of carcinoma of the colon and rectum. *Annals of surgery*, 139(6):846, 1954.

P. Bastien, V. Esposito-Vinzi, and M. Tenenhaus. PLS generalised linear regression. *Computational Statistics & Data Analysis*, 48(1):17–46, 2005.

A. Canty and B. D. Ripley. *boot: Bootstrap R (S-Plus) Functions*, 2014. R package version 1.3-11.

A. C. Davison and D. V. Hinkley. *Bootstrap Methods and Their Applications*. Cambridge University Press, Cambridge, 1997.

B. Efron and R. J. Tibshirani. *An introduction to the bootstrap*, volume 57. Chapman & Hall/CRC, 1993.

Max Kuhn. Contributions from Jed Wing, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, and the R Core Team. *caret: Classification and Regression Training*, 2014. URL http://CRAN.R-project.org/package=caret. R package version 6.0-30.

D. M. Haaland and J. D. T. Howland. Weighted partial least squares method to improve calibration precision for spectroscopic noise-limited data. In *The eleventh international conference on fourier transform spectroscopy*, volume 430 of *AIP Conference Proceedings*, pages 253–256. 1998.

A. Höskuldsson. PLS regression methods. *Journal of Chemometrics*, 2(3):211–228, 1988.

N. Kettaneh-Wold. Analysis of mixture data with partial least squares. *Chemometrics and Intelligent Laboratory Systems*, 14(1):57–69, 1992.

N. Krämer and M. Sugiyama. The degrees of freedom of partial least squares regression. *Journal of the American Statistical Association*, 106(494), 2011.

A. Lazraq, R. Cleroux, and J.-P. Gauchi. Selecting both latent and explanatory variables in the PLS1 regression model. *Chemometrics and Intelligent Laboratory Systems*, 66(2):117–126, 2003.

B. Li, J. Morris, and E. B. Martin. Model selection for partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, 64(1):79–89, 2002.

N. Meyer, M. Maumy-Bertrand, and F. Bertrand. Comparaison de variantes de régressions logistiques pls et de régression pls sur variables qualitatives: application aux données d'allélotypage. *Journal de la Société Française de Statistique*, 151(2):1–18, 2010.

L. H. Moulton and S. L. Zeger. Bootstrapping generalized linear models. *Computational Statistics & Data Analysis*, 11(1):53–63, 1991.

T. Naes and H. Martens. Comparison of prediction methods for multicollinear data. *Communications in Statistics - Simulation and Computation*, 14(3):545–576, 1985.

T. L. Shaffer. A unified approach to analyzing nest success. *Auk*, 121:526—-540, 2004.

M. Tenenhaus. *La régression PLS, Théorie et pratique*. Éditions Technip, Paris, 1998.

M. Tenenhaus. La régression logistique PLS. In J.-J. Droesbeke, M. Lejeune, and G. Saporta, editors, *Modèles statistiques pour données qualitatives*. Éditions Technip, Paris, 2005.

R. Tomassone, S. Audrain, E. Lesquoy-de Turckeim, and C. Millier. *La régression, nouveaux regards sur une ancienne méthode statistique*. Actualités scientifiques et agronomiques. Masson, Paris, 1992.

S. van Buuren and K. Groothuis-Oudshoorn. mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, 45(3):1–67, 2011. URL http://www.jstatsoft.org/v45/i03/.

H. Wold et al. Estimation of principal components and related models by iterative least squares. In P. R. Krishnaiaah, editor, *Multivariate analysis*, pages 391–420. Academic Press, New York, 1966.

S. Wold, H. Martens, and H. Wold. The multivariate calibration problem in chemistry solved by the PLS method. *Matrix pencils*, pages 286–293, 1983.

S. Wold, A. Ruhe, H. Wold, and W. J. Dunn, III. The collinearity problem in linear regression. the partial least squares (PLS) approach to generalized inverses. *SIAM Journal on Scientific and Statistical Computing*, 5(3):735–743, 1984.

S. Wold, M. Sjöström, and L. Eriksson. PLS-regression: a basic tool of chemometrics. *Chemometrics and intelligent laboratory systems*, 58(2):109–130, 2001.