The RGCCA package for Regularized/Sparse Generalized Canonical Correlation Analysis

2020-04-10

Contents

1 Introduction

A challenging problem in multivariate statistics is to study relationships between several sets of variables measured on the same set of individuals. In the literature, this paradigm can be stated under several names as “learning from multimodal data”, “data integration”, “data fusion” or “multiblock data analysis”. Typical examples are found in large variety of fields such as biology, chemistry, sensory analysis, marketing, food research, where the common general objective is to identify variables of each block that are active in the relationships with other blocks. For instance, neuroimaging is increasingly recognised as an intermediate phenotype to understand the complex path between genetics and behavioural or clinical phenotypes. In this imaging-genetics context, the goal is primarily to identify a set of genetic biomarker that explains some neuroimaging variability which implies some modification of the behavioural. A second application is found in molecular biology where the completion of the human genome sequence has shifted research efforts in genomics toward understanding the effect of sequence variation on gene expression, protein function, or other cellular mechanims. Both in the imaging-genetics and the multi-modal genetic context, it is crucial to perform multiple experiments (e.g. SNPs, functional MRI, behavioural data) on a single set of patients and the joint analysis of multiple datasets becomes more and more crucial. The RGCCA package aims to propose a unified and flexible framework for that purpose.

2 Multiblock data analysis with the RGCCA package

For the sake of comprehension of the use of the RGCCA package, the theoretical fundations of RGCCA and variations - that were previously published (Tenenhaus and Tenenhaus 2011 ; A. Tenenhaus and Tenenhaus 2014 ; Arthur Tenenhaus, Philippe, and Frouin 2015 ; Tenenhaus, Tenenhaus, and Groenen 2017) - are described.

We consider J data matrices $\mathbf{X}_1, \dots, \mathbf{X}_J$. Each $n \times p_j$ data matrix $\mathbf{X}_j = [\mathbf{x}_{j1}, \dots, \mathbf{x}_{jp_j}]$ is called a block and represents a set of p_j variables observed on n individuals. The number and the nature of the variables may differ from one block to another, but the individuals must be the same across blocks. We assume that all variables are centered. The objective of RGCCA is to find, for each block, a weighted composite of variables (called block component) $\mathbf{y}_j = \mathbf{X}_j\mathbf{a}_j, j = 1, \dots, J$ (where $\mathbf{a}_j$ is a column-vector with p_j elements) summarizing the relevant information between and within the blocks. The block components are obtained such that (i) block components explain well their own block and/or (ii) block components that are assumed to be connected are highly correlated. In addition, RGCCA integrates a variable selection procedure, called SGCCA, allowing the identification of the most relevant features. Finally, as a component-based method, RGCCA/SGCCA can provide users with graphical representations to visualize the sources of variability within blocks and the amount of correlation between blocks.

2.1 Regularized Generalized Canonical Correlation Analysis

The second generation RGCCA (Tenenhaus, Tenenhaus, and Groenen 2017) subsumes fifty years of multiblock component methods. It provides important improvements to the initial version of RGCCA (Tenenhaus and Tenenhaus 2011) and is defined as the following optimization problem:

$$\underset{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_J}{\text{maximize}} \sum_{j,k=1}^{J} c_{jk} g(\text{cov}(\mathbf{X}_j\mathbf{a}_j, \mathbf{X}_k\mathbf{a}_k)) \text{ s.t. } (1-\tau_j)\text{var}(\mathbf{X}_j\mathbf{a}_j) + \tau_j\|\mathbf{a}_j\|^2 = 1, j = 1, \dots, J \tag{1}$$

where:

- The scheme function g is any continuous convex function and allows to consider different optimization criteria. Typical choices of g are the identity (horst scheme, leading to maximizing the sum of covariances between block components), the absolute value (centroid scheme, yielding maximization of the sum of the absolute values of the covariances), the square function (factorial scheme, thereby maximizing the sum of squared covariances), or, more generally, for any even integer m, $g(x) = x^m$ (m-scheme, maximizing the power of m of the sum of covariances). The horst scheme penalizes structural negative correlation between block components while both the centroid scheme and the m-scheme enable two components to be negatively correlated. According to (Van de Geer 1984), a fair model is a model where all blocks contribute equally to the solution in opposition to a model dominated by only a few of the J sets. If fairness is a major objective, the user must choose $m = 1$. $m > 1$ is preferable if the user wants to discriminate between blocks. In practice, m is equal to 1, 2 or 4. The higher the value of m the more the method acts as block selector (Tenenhaus, Tenenhaus, and Groenen 2017).

- The design matrix $\mathbf{C}$ is a symmetric $J \times J$ matrix of nonnegative elements describing the network of connections between blocks that the user wants to take into account. Usually, $c_{jk} = 1$ for two connected blocks and 0 otherwise.
- The τ_j are called shrinkage parameters ranging from 0 to 1 and interpolate smoothly between maximizing the covariance and maximizing the correlation. Setting the τ_j to 0 will force the block components to unit variance ($\mathrm{var}(\mathbf{X}_j\mathbf{a}_j = 1)$), in which case the covariance criterion boils down to the correlation. The correlation criterion is better in explaining the correlated structure across datasets, thus discarding the variance within each individual dataset. Setting τ_j to 1 will normalize the block weight vectors ($\mathbf{a}_j^\top\mathbf{a}_j = 1$), which applies the covariance criterion. A value between 0 and 1 will lead to a compromise between the two first options and correspond to the following constraint $(1 - \tau_j)\mathrm{var}(\mathbf{X}_j\mathbf{a}_j) + \tau_j\|\mathbf{a}_j\|^2 = 1$ in (1). The choices $\tau_j = 1$, $\tau_j = 0$ and $0 < \tau_j < 1$ are respectively referred as Modes A, B and Ridge. In the RGCCA package, for each block, the determination of the shrinkage parameter can be made fully automatic by using the analytical formula proposed by (Schäfer and Strimmer 2005). Also, depending on the context, the shrinkage parameters should also be determined based on V-fold cross-validation. We can define the choice of the shrinkage parameters by providing interpretations on the properties of the resulting block components:
  - $\tau_j = 1$ yields the maximization of a covariance-based criterion. It is recommended when the user wants a stable component (large variance) while simultaneously taking into account the correlations between blocks. The user must, however, be aware that variance dominates over correlation.
  - $\tau_j = 0$ yields the maximization of a correlation-based criterion. It is recommended when the user wants to maximize correlations between connected components. This option can yield unstable solutions in case of multi-collinearity and cannot be used when a data block is rank deficient (e.g. $n < p_j$).
  - $0 < \tau_j < 1$ is a good compromise between variance and correlation: the block components are simultaneously stable and as well correlated as possible with their connected block components. This setting can be used when the data block is rank deficient.

From optimization problem (1), the term "generalized" in the acronym of RGCCA embraces at least three notions. The first one relates to the generalization of two-block methods - including Canonical Correlation Analysis (Hotelling 1936) Interbattery Factor Analysis (Tucker 1958) and Redundancy Analysis (Van den Wollenberg 1977) - to three or more sets of variables. The second one relates to the ability of taking into account some hypotheses on between-block connections: the user decides which blocks are connected and which ones are not. The third one relies on the choices of the shrinkage parameters allowing to capture both correlation or covariance-based criteria.

2.2 Variable selection in RGCCA: SGCCA

The quality and interpretability of the RGCCA block components $\mathbf{y}_j = \mathbf{X}_j\mathbf{a}_j, j = 1, \ldots, J$ are likely affected by the usefulness and relevance of the variables of each block. Accordingly, it is an important issue to identify within each block a subset of significant variables which are active in the relationships between blocks. SGCCA extends RGCCA to address this issue of variable selection. Specifically, RGCCA with all $\tau_j = 1$ equal to 1 is combined with an L1-penalty that gives rise to SGCCA (A. Tenenhaus et al. 2014). The SGCCA optimization problem is defined as follows:

$$\underset{\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_J}{\text{maximize}} \sum_{j,k=1}^{J} c_{jk} g(\mathrm{cov}(\mathbf{X}_j\mathbf{a}_j, \mathbf{X}_k\mathbf{a}_k)) \;\text{ s.t. }\; \|\mathbf{a}_j\|_2 = 1 \text{ and } \|\mathbf{a}_j\|_1 \leq s_j, j = 1, \ldots, J \tag{2}$$

where s_j is a user defined positive constant that determines the amount of sparsity for $\mathbf{a}_j, j = 1, \ldots, J$. The smaller the s_j, the larger the degree of sparsity for $\mathbf{a}_j$. The sparsity parameter s_j is usually set based on cross-validation procedures. Alternatively, values of s_j can simply be chosen to result in desired amounts of sparsity.

2.3 Higher stage block components

It is possible to obtain more than one block-component per block for RGCCA and SGCCA. Higher stage block components can be obtained using a deflation strategy. This strategy forces all the block components within a block to be uncorrelated. This deflation procedure can be iterated in a very flexible way. It is not necessary to keep all the blocks in the procedure at all stages: the number of components summarizing a block can vary from one block to another (see (Tenenhaus, Tenenhaus, and Groenen 2017) for details).

2.4 Implementation

The function `rgcca()` of the RGCCA package implements a monotonically convergent algorithm for the optimization problem (1) - i.e. the bounded criterion to be maximized increases at each step of the iterative procedure -, which hits at convergence a stationary point of (1). Two numerically equivalent approaches for solving the RGCCA optimization problem are available. A primal formulation described in (Tenenhaus, Tenenhaus, and Groenen 2017 ; Tenenhaus and Tenenhaus 2011) requires the handling of matrices of dimension $p_j \times p_j$. A dual formulation described in (Arthur Tenenhaus, Philippe, and Frouin 2015) requires the handling of matrices of dimension $n \times n$. Therefore, the primal formulation of the RGCCA algorithm will be used when $n > p_j$ and the dual form will be preferred when $n \leq p_j$. The `rgcca()` function of the RGCCA package implements these two formulations and selects automatically the best one. The SGCCA algorithm is similar to the RGCCA algorithm and keeps the same convergence properties. The algorithm associated with the optimization problem (2) is available through the function `rgcca()` with the sgcca option (type="sgcca")in the RGCCA package.

2.5 Special cases of RGCCA

RGCCA is a rich technique that encompasses a large number of multiblock methods that were published for fifty years. These methods are recovered with RGCCA by appropriately defining the triplet ($\mathbf{C}$, τ_j, g). Table 1 gives the correspondences between the triplet ($\mathbf{C}$, τ_j, g) and the associated methods. For a complete overview see (Tenenhaus, Tenenhaus, and Groenen 2017).

Table 1: Special cases of RGCCA in a situtation of $J \geq 2$ blocks. When $\tau_J + 1$ is introduced, it is assumed that $\mathbf{X}_1, \ldots, \mathbf{X}_J$ are connected to a $(J+1)$th block defined as the concatenation of the blocks, $\mathbf{X}_{J+1} = \left[\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_J\right]$ and that τ_{J+1} corresponds to the shrinkage parameter associated with $\mathbf{X}_{J+1}$.

Methods	$g(x)$	τ_j	$\mathbf{C}$
Canonical Correlation Analysis (Hotelling 1936)	x	$\tau_1 = \tau_2 = 0$	$\mathbf{C}_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
Interbattery Factor Analysis (Tucker 1958)	x	$\tau_1 = \tau_2 = 1$	$\mathbf{C}_1$
Redundancy Analysis (Van den Wollenberg 1977)	x	$\tau_1 = 1$ and $\tau_2 = 0$	$\mathbf{C}_1$
SUMCOR (Horst 1961)	x	$\tau_j = 0, j = 1, \ldots, J$	$\mathbf{C}_2 = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 \\ 1 & \cdots & 1 & 1 \end{pmatrix}$
SSQCOR (Kettenring 1971)	x^2	$\tau_j = 0, 1 \leq j \leq J$	$\mathbf{C}_2$
SABSCOR (Hanafi 2007)	\$abs(x) \$	$\tau_j = 0, 1 \leq j \leq J$	$\mathbf{C}_2$
SUMCOV-1 (Van de Geer 1984)	x	$\tau_j = 1, 1 \leq j \leq J$	$\mathbf{C}_2$
SSQCOV-1 (Hanafi and Kiers 2006)	x^2	$\tau_j = 1, 1 \leq j \leq J$	$\mathbf{C}_2$
SABSCOV-1 (Tenenhaus and Tenenhaus 2011 ; Kramer 2007)	$abs(x)$	$\tau_j = 1, 1 \leq j \leq J$	$\mathbf{C}_2$

Methods	$g(x)$	τ_j	$\mathbf{C}$
SUMCOV-2 (Van de Geer 1984)	x	$\tau_j = 0, 1 \le j \le J$	$\mathbf{C}_3 = \begin{pmatrix} 0 & 1 & \cdots & 1 \\ 1 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 \\ 1 & \cdots & 1 & 0 \end{pmatrix}$
SSQCOV-2 (Hanafi and Kiers 2006)	x^2	$\tau_j = 1, 1 \le j \le J$	$\mathbf{C}_3$
Generalized CCA (J.D. Carroll 1968)	x^2	$\tau_j = 0, 1 \le j \le J + 1$	$\mathbf{C}_4 = \begin{pmatrix} 0 & \cdots & 0 & 1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 1 \\ 1 & \cdots & 1 & 0 \end{pmatrix}$
Generalized CCA (JD Carroll 1968)	x^2	$\tau_j = 0, 0 \le j \le J_1 =$ 0 & $\tau_{J+1} = 0$ and $\tau_j = 1, J_1 + 1 \le j \le J$	$\mathbf{C}_4$
Hierarchical PCA (Wold, S. and Kettaneh, N. and Tjessem, K. 1996)	x^4	$\tau_j = 1, 1 \le j \le J$ and $\tau_{J+1} = 0$	$\mathbf{C}_4$
Multiple Co-Inertia Analysis (Chessel and Hanafi 1996)	x^2	$\tau_j = 1, 1 \le j \le J$ and $\tau_{J+1} = 0$	$\mathbf{C}_4$
PLS path modeling - mode B (Wold 1982)	$abs(x)$	$\tau_j = 0, 1 \le j \le J$	$c_{jk} = 1$ for two connected block and $c_{jk} = 0$ otherwise

For all the methods of Table 1, a single very simple monotonically convergent gradient-based algorithm is implemented within the RGCCA package and gives at convergence a solution of the stationary equations related to the optimization problem (1). In addition, SGCCA offers a sparse counterpart to all the covariance-based methods of RGCCA. From these perspectives, R/SGCCA provide a general framework for exploratory data analysis of multiblock datasets that has immediate practical consequences for a unified statistical analysis and implementation strategy.

The methods cited in Table 1 are recovered with the `rgcca()` function by appropriately tuning the arguments `connection`, `tau` and `scheme` associated with the triplet ($\mathbf{C}$, τ_j, g). All the methods of Table 1 are obtained as follows:

2.5.1 Principal Component Analysis.

Principal Component Analysis is defined as the following optimization problem

$$\underset{\mathbf{a}}{\text{maximize}} \;\; \text{var}\left(\mathbf{Xa}\right) \text{ s.t. } \|\mathbf{a}\| = 1 \tag{3}$$

and is obtained with the `rgcca()` function as follows:

```
# one block X
# Design matrix C
# Shrinkage parameters tau = c(tau1, tau2)

pca.with.rgcca = rgcca(blocks = list(X, X),
                       connection = matrix(c(0, 1, 1, 0), 2, 2),
                       tau = c(1, 1), type="rgcca")
```

or

```
pca.with.rgcca = rgcca(blocks = list(X), type="pca")
```

2.5.2 Canonical Correlation Analysis

Canonical Correlation Analysis is defined as the following optimization problem

$$\underset{\mathbf{a}_1,\mathbf{a}_2}{\text{maximize}} \;\; \text{cor}\left(\mathbf{X}_1\mathbf{a}_1, \mathbf{X}_2\mathbf{a}_2\right) \text{ s.t. } \text{var}(\mathbf{X}_1\mathbf{a}_1) = \text{var}(\mathbf{X}_2\mathbf{a}_2) = 1 \tag{4}$$

and is obtained with the `rgcca()` function as follows:

```
# X1 = Block1 and X2 = Block2
# Design matrix C
# Shrinkage parameters tau = c(tau1, tau2)

cca.with.rgcca = rgcca(blocks= list(X1, X2),
                       connection = matrix(c(0, 1, 1, 0), 2, 2),superblock=FALSE,
                       tau = c(0, 0), type="rgcca")
```

or

```
cca.with.rgcca = rgcca(blocks= list(X1, X2),
                       type="cca")
```

2.5.3 PLS regression (≈ Interbattery factor analysis)

PLS regression is defined as the following optimization problem

$$\underset{\mathbf{a}_1,\mathbf{a}_2}{\text{maximize}} \;\; \text{cov}\left(\mathbf{X}_1\mathbf{a}_1, \mathbf{X}_2\mathbf{a}_2\right) \text{ s.t. } \|\mathbf{a}_1\| = \|\mathbf{a}_2\| = 1 \tag{5}$$

and is obtained with the `rgcca()` function as follows:

```
# X1 = Block1 and X2 = Block2
# Design matrix C
# Shrinkage parameters tau = c(tau1, tau2)

pls.with.rgcca = rgcca(blocks= list(X1, X2),
                       connection= matrix(c(0, 1, 1, 0), 2, 2),superblock=FALSE,
                       tau = c(1, 1))
```

or

```
pls.with.rgcca = rgcca(blocks= list(X1, X2),type="pls")
```

2.5.4 Redundancy Analysis of $\mathbf{X}_1$ with respect to $\mathbf{X}_2$

Redundancy Analysis of $\mathbf{X}_1$ with respect to $\mathbf{X}_2$ is defined as the following optimization problem

$$\underset{\mathbf{a}_1,\mathbf{a}_2}{\text{maximize}} \;\; \text{cor}\left(\mathbf{X}_1\mathbf{a}_1, \mathbf{X}_2\mathbf{a}_2\right) \times \text{ var}\left(\mathbf{X}_1\mathbf{a}_1\right)^{1/2} \text{ s.t. } \|\mathbf{a}_1\| = \text{var}(\mathbf{X}_2\mathbf{a}_2) = 1 \tag{6}$$

and is obtained with the `rgcca()` function as follows:

```
# X1 = Block1 and X2 = Block2
# Design matrix C
# Shrinkage parameters tau = c(tau1, tau2)

ra.with.rgcca = rgcca(blocks= list(X1, X2),
                      connection = matrix(c(0, 1, 1, 0), 2, 2),
                      superblock=FALSE,
                      tau = c(1, 0))
```

2.5.5 Regularized Canonical Correlation Analysis (Vinod 1976 ; Shawe-Taylor and Cristianini 2004)

Regularized Canonical Correlation Analysis is defined as the following optimization problem

$$\underset{\mathbf{a}_1,\mathbf{a}_2}{\text{maximize}} \;\; \text{cov}\left(\mathbf{X}_1\mathbf{a}_1, \mathbf{X}_2\mathbf{a}_2\right) \text{ s.t. } \tau_j\|\mathbf{a}_j\|^2 + (1-\tau_j)\text{var}(\mathbf{X}_j\mathbf{a}_j) = 1, j = 1, 2 \tag{7}$$

and is obtained with the `rgcca()` function as follows:

```
# X1 = Block1 and X2 = Block2
# Design matrix C
# Shrinkage parameters tau = c(tau1, tau2)

rcca.with.rgcca = rgcca(blocks= list(X1, X2),
                        connection = matrix(c(0, 1, 1, 0), 2, 2),
                        superblock=FALSE,
                        tau = c(0<tau1<1, 0<tau2<1))
```

For various extreme cases $\tau_1 = 0$ or 1 and $\tau_2 = 0$ or 1, optimization problem (7) covers a situation which goes from Tucker's interbattery factor analysis to Canonical Correlation Analysis while passing through redundancy analysis. This framework corresponds exactly to the one proposed by (Borga, Landelius, and Knutsson 1997) and (Burnham, Viveros, and MacGregor 1996). Moreover, the special situation where $0 \leq \tau_1 \leq 1$ and $\tau_2 = 0$ which corresponds to a regularized version of redundancy analysis has been studied by (Takane and Hwang 2007) and by (Bougeard, Hanafi, and Qannari 2008) under the name "Continuum redundancy-PLS regression". When one block is reduced to only one variable, optimization problem (7) is equivalent to the simple continuum regression approach proposed by (Qannari and Hanafi 2005).

2.5.6 SUMCOV-1

SUMCOV-1 is defined as the following optimization problem

$$\underset{\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_J}{\text{maximize}} \sum_{j,k=1}^{J} \text{cov}(\mathbf{X}_j\mathbf{a}_j, \mathbf{X}_k\mathbf{a}_k) \;\text{ s.t }\; \|\mathbf{a}_j\| = 1, j = 1, \ldots, J \tag{8}$$

and is obtained with the `rgcca()` function as follows:

```
# X1 = Block1, ..., XJ = BlockJ
# J*J Design matrix connection
# Shrinkage parameters tau = c(tau1, ...,  tauJ)

sumcov.with.rgcca = rgcca(blocks= list(X1, ..., XJ),
                          connection = matrix(1, J, J),
                          tau = rep(1, J),
                          scheme = "horst",
                          superblock=FALSE
                          )
```

or

```
# X1 = Block1, ..., XJ = BlockJ
# J*J Design matrix connection
# Shrinkage parameters tau = c(tau1, ...,  tauJ)

sumcov.with.rgcca = rgcca(blocks= list(X1, ..., XJ), type="sumcov-1")
```

2.5.7 SSQCOV-1

SSQCOV-1 is defined as the following optimization problem

$$\underset{\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_J}{\text{maximize}} \sum_{j,k=1}^{J} \text{cov}^2(\mathbf{X}_j\mathbf{a}_j, \mathbf{X}_k\mathbf{a}_k) \;\; \text{s.t} \;\; \|\mathbf{a}_j\| = 1, j = 1, \ldots, J \tag{9}$$

and is obtained with the `rgcca()` function as follows:

```
# X1 = Block1, ..., XJ = BlockJ
# J*J Design matrix connection
# Shrinkage parameters tau = c(tau1, ...,  tauJ)

ssqcov.with.rgcca = rgcca(blocks= list(X1, ..., XJ),
                             connection = matrix(1, J, J),
                             tau = rep(1, J),
                             scheme = "factorial",
                             superblock=FALSE)
```

or

```
# X1 = Block1, ..., XJ = BlockJ
# J*J Design matrix connection
# Shrinkage parameters tau = c(tau1, ...,  tauJ)

ssqcov.with.rgcca = rgcca(blocks= list(X1, ..., XJ), type="ssqcov")
```

2.5.8 SABSCOV

SABSCOV is defined as the following optimization problem

$$\underset{\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_J}{\text{maximize}} \sum_{j,k=1}^{J} |\text{cov}(\mathbf{X}_j\mathbf{a}_j, \mathbf{X}_k\mathbf{a}_k)| \;\; \text{s.t} \;\; \|\mathbf{a}_j\| = 1, j = 1, \ldots, J \tag{10}$$

and is obtained with the `rgcca()` function as follows:

```
# X1 = Block1, ..., XJ = BlockJ
# J*J Design matrix connection
# Shrinkage parameters tau = c(tau1, ...,  tauJ)

```

```
sabscov.with.rgcca = rgcca(blocks= list(X1, ..., XJ),
                           connection = matrix(1, J, J),
                           tau = rep(1, J),
                           scheme = "centroid",
                           superblock=FALSE)
```

or

```
# X1 = Block1, ..., XJ = BlockJ
# J*J Design matrix connection
# Shrinkage parameters tau = c(tau1, ...,  tauJ)

sabscov.with.rgcca = rgcca(blocks= list(X1, ..., XJ),type="sabscov")
```

2.5.9 SUMCOR

SUMCOR is defined as the following optimization problem

$$\underset{\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_J}{\text{maximize}} \sum_{j,k=1}^{J} \text{cor}(\mathbf{X}_j\mathbf{a}_j, \mathbf{X}_k\mathbf{a}_k) \text{ s.t } \text{var}(\mathbf{X}_j\mathbf{a}_j) = 1, j = 1, \ldots, J \tag{11}$$

and is obtained with the `rgcca()` function as follows:

```
# X1 = Block1, ..., XJ = BlockJ
# J*J Design matrix C
# Shrinkage parameters tau = c(tau1, ...,  tauJ)

sumcor.with.rgcca = rgcca(blocks= list(X1, ..., XJ),
                          connection = matrix(1, J, J),
                          tau = rep(0, J),
                          scheme = "horst",
                          superblock=FALSE)
```

or

```
# X1 = Block1, ..., XJ = BlockJ
# J*J Design matrix connection
# Shrinkage parameters tau = c(tau1, ...,  tauJ)

sumcor.with.rgcca = rgcca(blocks= list(X1, ..., XJ),type="sumcor")
```

2.5.10 SSQCOR

SSQCOR is defined as the following optimization problem

$$\underset{\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_J}{\text{maximize}} \sum_{j,k=1;k\neq j}^{J} \text{cor}^2(\mathbf{X}_j\mathbf{a}_j, \mathbf{X}_k\mathbf{a}_k) \text{ s.t } \text{var}(\mathbf{X}_j\mathbf{a}_j) = 1, j = 1, \ldots, J \tag{12}$$

and is obtained with the `rgcca()` function as follows:

```
# X1 = Block1, ..., XJ = BlockJ
# J*J Design matrix connection
# Shrinkage parameters tau1, ...,  tauJ

ssqcor.with.rgcca = rgcca(blocks= list(X1, ..., XJ),
                      connection = matrix(1, J, J),
                      tau = rep(0, J),
                      scheme = "factorial",
                      superblock=FALSE)
```

or

```
# X1 = Block1, ..., XJ = BlockJ
# J*J Design matrix C
# Shrinkage parameters tau1, ...,  tauJ

ssqcor.with.rgcca = rgcca(blocks= list(X1, ..., XJ),
                      type="ssqcor")
```

2.5.11 SABSCOR

SABSCOR is defined as the following optimization problem

$$\underset{\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_J}{\text{maximize}} \sum_{j,k=1}^{J} |\text{cor}(\mathbf{X}_j\mathbf{a}_j, \mathbf{X}_k\mathbf{a}_k)| \text{ s.t } \text{var}(\mathbf{X}_j\mathbf{a}_j) = 1, j = 1, \ldots, J \tag{13}$$

and is obtained with the `rgcca()` function as follows:

```
# X1 = Block1, ..., XJ = BlockJ
# J*J Design matrix connection
# Shrinkage parameters tau = c(tau1, ...,  tauJ)

sabscor.with.rgcca = rgcca(blocks= list(X1, ..., XJ),
                     connection = matrix(1, J, J),
                      tau = rep(0, J),
                      scheme = "centroid",
                     superblock=FALSE)
```

or

```
# X1 = Block1, ..., XJ = BlockJ
# J*J Design matrix connection
# Shrinkage parameters tau = c(tau1, ...,  tauJ)

sabscor.with.rgcca = rgcca(blocks= list(X1, ..., XJ),
                     type="sabscor")
```

2.5.12 SUMCOV-2

SUMCOV-2 is obtained from following optimization problem

$$\underset{\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_J}{\text{maximize}} \sum_{j,k=1;j\neq k}^{J} \text{cov}(\mathbf{X}_j\mathbf{a}_j, \mathbf{X}_k\mathbf{a}_k) \;\text{ s.t }\; \|\mathbf{a}_j\| = 1, j = 1, \ldots, J \tag{14}$$

and is obtained with the `rgcca()` function as follows:

```
# X1 = Block1, ..., XJ = BlockJ
# J*J Design matrix C
C = matrix(1, J, J) ; diag(C) = 0
# Shrinkage parameters tau = c(tau1, ...,  tauJ)
maxbet.with.rgcca = rgcca(blocks= list(X1, ..., XJ),
                       connection = C,
                       tau = rep(1, J),
                       superblock=FALSE,
                       scheme = "horst")
```

or

```
# X1 = Block1, ..., XJ = BlockJ
# J*J Design matrix C
# C = matrix(1, J, J) ; diag(C) = 0
# Shrinkage parameters tau = c(tau1, ...,  tauJ)
maxbet.with.rgcca = rgcca(blocks= list(X1, ..., XJ),type="maxbet"
```

2.5.13 SSQCOV-2

SSQCOV-2 is obtained from following optimization problem

$$\underset{\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_J}{\text{maximize}} \sum_{j,k=1;j\neq k}^{J} \text{cov}^2(\mathbf{X}_j\mathbf{a}_j, \mathbf{X}_k\mathbf{a}_k) \;\text{ s.t }\; \|\mathbf{a}_j\| = 1, j = 1, \ldots, J \tag{15}$$

and is obtained with the `rgcca()` function as follows:

```
# X1 = Block1, ..., XJ = BlockJ
# J*J Design matrix C
C = matrix(1, J, J) ; diag(C) = 0
# Shrinkage parameters tau = c(tau1, ...,  tauJ)
maxbetb.with.rgcca = rgcca(blocks= list(X1, ..., XJ),
                       connection = C,
                       tau = rep(1, J),
                       superblock=FALSE,
                       scheme = "factorial")
```

or

```
maxbetb.with.rgcca = rgcca(blocks= list(X1, ..., XJ),type="maxbet-b")
```

or

```
maxbetb.with.rgcca = rgcca(blocks= list(X1, ..., XJ),type="ssqcov-2")
```

2.5.14 Generalized CCA (J.D. Carroll 1968)

For Carroll's Generalized Canonical Correlation Analysis (GCCA), a superblock $\mathbf{X}_{J+1} = [\mathbf{X}_1, \ldots, \mathbf{X}_J]$ defined as the concatenation of all the blocks is introduced. GCCA is defined as the following optimization problem

$$\underset{\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_J}{\text{maximize}} \sum_{j=1}^{J} \text{cor}^2(\mathbf{X}_j\mathbf{a}_j, \mathbf{X}_{J+1}\mathbf{a}_{J+1}) \;\text{ s.t }\; \text{var}(\mathbf{X}_j\mathbf{a}_j) = 1, j = 1, \ldots, J+1 \tag{16}$$

and is obtained with the `rgcca()` function as follows:

```
# X1 = Block1, ..., XJ = BlockJ, X_{J+1} = [X1, ..., XJ]
# (J+1)*(J+1) Design matrix C
C = matrix(c(0, 0, 0, ..., 0, 1,
             0, 0, 0, ..., 0, 1,
             0, 0, 0, ..., 0, 1,
                  ...
             1, 1, 1, ..., 1, 0), J+1, J+1)
# Shrinkage parameters tau = c(tau1, ...,  tauJ, tau_{J+1})
gcca.with.rgcca = rgcca(blocks= list(X1, ..., XJ, cbind(X1, ..., XJ)),
                        connection = C,
                        tau = rep(0, J+1),
                        superblock=FALSE,
                        scheme = "factorial")
```

or

```
gcca.with.rgcca = rgcca(blocks= list(X1, ..., XJ, cbind(X1, ..., XJ)),
                        connection = C,
                        superblock=FALSE,
                        tau = rep(0, J+1),
                        scheme = "factorial") #TODO
```

2.5.15 Multiple Co-Inertia Analysis

For Multiple Co-Inertia Analysis (MCOA) a superblock $\mathbf{X}_{J+1} = [\mathbf{X}_1, \ldots, \mathbf{X}_J$ defined as the concatenation of all the blocks is introduced. MCOA is defined as the following optimization problem

$$\underset{\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_J}{\text{maximize}} \sum_{j=1}^{J} \text{cor}^2(\mathbf{X}_j\mathbf{a}_j, \mathbf{X}_{J+1}\mathbf{a}_{J+1}) \times \text{var}(\mathbf{X}_j\mathbf{a}_j), \;\text{ s.t }\; \|\mathbf{a}_j\| = 1, j = 1, \ldots, J \;\text{ and }\; \text{var}(\mathbf{X}_{J+1}\mathbf{a}_{J+1}) = 1 \tag{17}$$

and is obtained with the `rgcca()` function as follows:

```
# X1 = Block1, ..., XJ = BlockJ, X_{J+1} = [X1, ..., XJ]
# (J+1)*(J+1) Design matrix C
C = matrix(c(0, 0, 0, ..., 0, 1,
             0, 0, 0, ..., 0, 1,
             0, 0, 0, ..., 0, 1,
                   ...
             1, 1, 1, ..., 1, 0), J+1, J+1)
# Shrinkage parameters tau = c(tau1, ...,  tauJ, tau_{J+1})
mcoa.with.rgcca = rgcca(blocks= list(X1, ..., XJ, cbind(X1, ..., XJ)),
                        connection= C,
                        superblock=FALSE,
                        tau = c(rep(1, J), 0),
                        scheme = "factorial")
```

2.5.16 Hierarchical Principal Component Analysis

For Hierarchical Principal Component Analysis (HPCA), a superblock $\mathbf{X}_{J+1} = [\mathbf{X}_1, \ldots, \mathbf{X}_J]$ defined as the concatenation of all the blocks is introduced. HPCA is defined as the following optimization problem

$$\underset{\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_J}{\text{maximize}} \sum_{j=1}^{J} \text{cov}^4(\mathbf{X}_j\mathbf{a}_j, \mathbf{X}_{J+1}\mathbf{a}_{J+1}), \text{ s.t } \|\mathbf{a}_j\| = 1, j = 1, \ldots, J \text{ and } \text{var}(\mathbf{X}_{J+1}\mathbf{a}_{J+1}) = 1 \tag{18}$$

and is obtained with the `rgcca()` function as follows:

```
# X1 = Block1, ..., XJ = BlockJ, X_{J+1} = [X1, ..., XJ]
# (J+1)*(J+1) Design matrix C
C = matrix(c(0, 0, 0, ..., 0, 1,
             0, 0, 0, ..., 0, 1,
             0, 0, 0, ..., 0, 1,
                  ...
             1, 1, 1, ..., 1, 0), J+1, J+1)
# Shrinkage parameters tau = c(tau1, ...,  tauJ, tau_{J+1})
hpca.with.rgcca = rgcca(blocks= list(X1, ..., XJ, cbind(X1, ..., XJ)),
                        connection= C,
                        tau = c(rep(1, J), 0),
                        superblock=FALSE,
                        #flexible design of the scheme function
                        scheme = function(x) x^4)
```

2.5.17 Principal Component Analysis (alternative formulation)

Let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_p]$ be a $n \times p$ data matrix. PCA can be defined as the following optimization problem

$$\underset{\mathbf{a}}{\text{maximize}} \sum_{j=1}^{p} \text{cor}^2(\mathbf{x}_j, \mathbf{X}\mathbf{a}) \text{ s.t } \text{var}(\mathbf{X}\mathbf{a}) = 1 \tag{19}$$

and is obtained with the `rgcca()` function as follows:

```
# one block X = [x1, ..., xJ] with J variables x1, ..., xJ
# (J+1)*(J+1) Design matrix C
C = matrix(c(0, 0, 0, ..., 0, 1,
             0, 0, 0, ..., 0, 1,
             0, 0, 0, ..., 0, 1,
                  ...
             1, 1, 1, ..., 1, 0), J+1, J+1)
# Shrinkage parameters tau = c(tau1, ...,  tauJ, tau_{J+1})
pca.with.rgcca = rgcca(list(x1, ..., xJ, X),
                       connection= C,
                       tau = c(rep(0, J+1), 0),
                       superblock=FALSE,
                       #flexible design of the scheme function
                       scheme = function(x) x^2)
```

3 Practical session

3.1 RGCCA for the Russett dataset.

In this section, we propose to reproduce some of the results presented in (Tenenhaus and Tenenhaus 2011) for the Russett data. The Russett dataset is available within the RGCCA package. The Russett data set (Russett 1964) are studied in (Gifi 1990). Russett collected this data to study relationships between Agricultural Inequality, Industrial Development and Political Instability.

```
library(RGCCA)
```

```
## Loading required package: MASS
```

```
data(Russett)
head(Russett)
```

```
##            gini farm rent gnpr labo inst ecks death demostab demoinst dictator
## Argentina  86.3 98.2 3.52 5.92 3.22 0.07 4.06  5.38        0        1        0
## Australia  92.9 99.6 3.27 7.10 2.64 0.01 0.00  0.00        1        0        0
## Austria    74.0 97.4 2.46 6.28 3.47 0.03 1.61  0.00        0        1        0
## Belgium    58.7 85.8 4.15 6.92 2.30 0.45 2.20  0.69        1        0        0
## Bolivia    93.8 97.7 3.04 4.19 4.28 0.37 3.99  6.50        0        0        1
## Brasil     83.7 98.5 2.31 5.57 4.11 0.45 3.91  0.69        0        1        0
```

The first step of the analysis is to define the blocks. Three blocks of variables have been defined for 47 countries. The variables that compose each block have been defined according to the nature of the variables.

- The first block $\mathbf{X}_1$ = [GINI, FARM, RENT] is related to “Agricultural Inequality”:
  - GINI = Inequality of land distribution,
  - FARM = % farmers that own half of the land (> 50),
  - RENT = % farmers that rent all their land.
- The second block $\mathbf{X}_2$ = [GNPR, LABO] describes “Industrial Development”:
  - GNPR = Gross national product per capita ($1955),
  - LABO = % of labor force employed in agriculture.
- The third one $\mathbf{X}_3$ = [INST, ECKS, DEAT] measures “Political Instability”:
  - INST = Instability of executive (45-61),
  - ECKS = Number of violent internal war incidents (46-61),
  - DEAT = Number of people killed as a result of civic group violence (50-62).
  - An additional variable DEMO describes the political regime: stable democracy, unstable democracy or dictatorship. The dummy variable “unstable democracy” has been left out because of redundancy.

The different blocks of variables $\mathbf{X}_1, \ldots, \mathbf{X}_J$ are arranged in the list format.

```
X_agric = as.matrix(Russett[,c("gini","farm","rent")])
X_ind = as.matrix(Russett[,c("gnpr","labo")])
X_polit = as.matrix(Russett[ , c("inst", "ecks",  "death",
                                 "demostab", "dictator")])
A = list(X_agric, X_ind, X_polit)
sapply(A, head)
```

```
## [[1]]
##            gini farm rent
## Argentina  86.3 98.2 3.52
## Australia  92.9 99.6 3.27
## Austria    74.0 97.4 2.46
## Belgium    58.7 85.8 4.15
## Bolivia    93.8 97.7 3.04
## Brasil    83.7 98.5 2.31
##
## [[2]]
##           gnpr labo
## Argentina 5.92 3.22
## Australia 7.10 2.64
## Austria   6.28 3.47
## Belgium   6.92 2.30
## Bolivia   4.19 4.28
## Brasil    5.57 4.11
##
## [[3]]
##           inst ecks death demostab dictator
## Argentina 0.07 4.06  5.38        0        0
## Australia 0.01 0.00  0.00        1        0
## Austria   0.03 1.61  0.00        0        0
## Belgium   0.45 2.20  0.69        1        0
## Bolivia   0.37 3.99  6.50        0        1
## Brasil    0.45 3.91  0.69        0        0
```

Preprocessing In order to ensure comparability between variables standardization is applied (zero mean and unit variance).

```
A = lapply(A, function(x) scale2(x, bias = TRUE))
```

We note that to make blocks comparable, a possible strategy is to standardize the variables and then to divide each block by the square root of its number of variables (Westerhuis, Kourti, and MacGregor 1998). This two-step procedure leads to $\mathrm{tr}(\mathbf{X}_j^t\mathbf{X}_j) = n$ for each block (i.e. the sum of the eigenvalues of the covariance matrix of $\mathbf{X}_j$ is equal to 1 whatever the block). Such a preprocessing is reached by setting the `scale` argument to `TRUE` (default value) in the `rgcca()` functions.

Definition of the design matrix C. From Russett's hypotheses, it is difficult for a country to escape dictatorship when its agricultural inequality is above-average and its industrial development below-average. These hypotheses on the relationships between blocks are depicted in Figure 1.

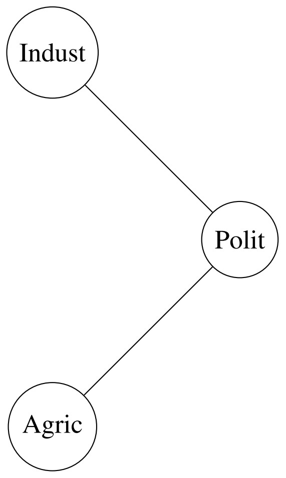


Figure 1: between-block connection.

and encoded through the design matrix $\mathbf{C}$; usually $c_{jk} = 1$ for two connected blocks and 0 otherwise. Therefore, we have decided to connect Agricultural Inequality to Political Instability ($c_{13} = 1$), Industrial Development to Political Instability ($c_{23} = 1$) and to not connect Agricultural Inequality to Industrial Development ($c_{12} = 0$). The resulting design matrix **C** is:

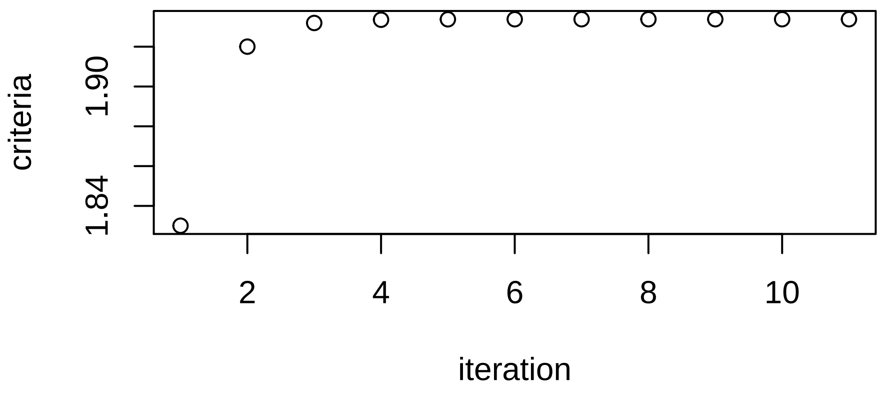
```
#Define the design matrix C.
C = matrix(c(0, 0, 1,
0, 0, 1,
1, 1, 0), 3, 3)

C
```

```
##      [,1] [,2] [,3]
## [1,]    0    0    1
## [2,]    0    0    1
## [3,]    1    1    0
```

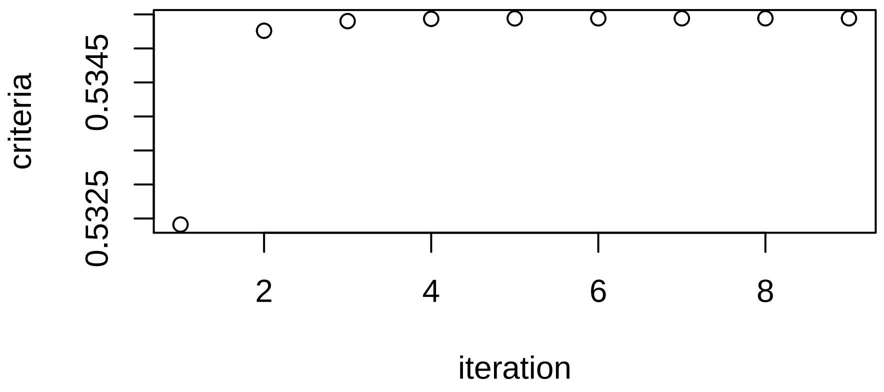
RGCCA using the pre-defined design matrix **C**, the factorial scheme ($g(x) = x^2$) and mode B for all blocks (full correlation criterion) is obtained by specifying appropriately the `C`, `scheme` and `tau` arguments of the `rgcca()` function. The `verbose` argument (default value = `TRUE`) indicates that the progress will be reported while computing and that a plot representing the convergence of the algorithm will be returned.

```
rgcca_B_factorial = rgcca(blocks=A, connection=C, tau = rep(0, 3), scheme = "factorial",
scale = FALSE, verbose = TRUE)
```

```
## Computation of the RGCCA block components based on the factorial scheme
## Shrinkage intensity paramaters are chosen manually
## Computation of the RGCCA block components #1 is under progress...
##  Iter:    1  Fit: 1.83005079  Dif:  0.38803933
##  Iter:    2  Fit: 1.92003517  Dif:  0.08998438
##  Iter:    3  Fit: 1.93192442  Dif:  0.01188925
##  Iter:    4  Fit: 1.93354278  Dif:  0.00161836
##  Iter:    5  Fit: 1.93376871  Dif:  0.00022593
##  Iter:    6  Fit: 1.93380060  Dif:  0.00003189
##  Iter:    7  Fit: 1.93380512  Dif:  0.00000452
##  Iter:    8  Fit: 1.93380576  Dif:  0.00000064
##  Iter:    9  Fit: 1.93380585  Dif:  0.00000009
##  Iter:   10  Fit: 1.93380586  Dif:  0.00000001
##  Iter:   11  Fit: 1.93380586  Dif:  0.00000000
## The RGCCA algorithm converged to a stationary point after 10 iterations
```



```
## Computation of the RGCCA block components #2 is under progress ...
##  Iter:    1  Fit: 0.53241157  Dif:  0.38795191
##  Iter:    2  Fit: 0.53525924  Dif:  0.00284766
##  Iter:    3  Fit: 0.53540108  Dif:  0.00014185
##  Iter:    4  Fit: 0.53543445  Dif:  0.00003337
##  Iter:    5  Fit: 0.53544109  Dif:  0.00000664
##  Iter:    6  Fit: 0.53544237  Dif:  0.00000129
##  Iter:    7  Fit: 0.53544262  Dif:  0.00000025
##  Iter:    8  Fit: 0.53544267  Dif:  0.00000005
##  Iter:    9  Fit: 0.53544268  Dif:  0.00000001
## The RGCCA algorithm converged to a stationary point after 8 iterations
```



The weight vectors, solution of the optimization problem (1) are obtained as:

```
rgcca_B_factorial$a # weight vectors
```

```
## [[1]]
##            [,1]      [,2]
## gini  1.826798 1.3217208
## farm -3.502036 0.7405909
## rent  1.361850 0.1314821
##
## [[2]]
##             [,1]     [,2]
## gnpr  0.4558005 2.825297
## labo -1.0178200 1.265225
##
## [[3]]
##                 [,1]        [,2]
## inst       0.3029039  0.74651348
## ecks      -0.2858136 -0.47831175
## death      0.1878383  2.68461838
## demostab   1.8674522 -0.46759739
## dictator  -0.5462041 -0.01118954
```

and correspond exactly to the weight vectors reported in (Tenenhaus and Tenenhaus 2011, see Figure 4). The block-components are also avalaible as output of `rgcca`. The first components of each block are given by:

```
Y = rgcca_B_factorial$Y
lapply(Y, head)
```

```
## [[1]]
##                 comp1      comp2
## Argentina  0.12892660  1.2856390
## Australia -0.01639636  1.6108710
## Austria   -1.41575574 -0.5539975
## Belgium    2.38653543  0.5866195
## Bolivia    0.43847422  1.8259136
## Brasil    -1.15744593  0.1988799
##
```

```
## [[2]]
##                 comp1      comp2
## Argentina  0.3340091 -0.8310223
## Australia  1.3652526  0.1262541
## Austria    0.2065837  0.4246859
## Belgium    1.6515657 -0.9868719
## Bolivia   -1.3949715 -1.7612656
## Brasil    -0.7152149  0.5055456
##
## [[3]]
##                 comp1       comp2
## Argentina -0.4534912  1.18490643
## Australia  1.4930824 -0.03462791
## Austria   -0.4399732 -1.27810261
## Belgium    1.5556128  0.62973048
## Bolivia   -0.7323576  1.97219793
## Brasil    -0.3981005 -0.67280389
```

Moreover, using the idea of Average Variance Explained (AVE), the following indicators of model quality are proposed:

- The AVE of block $\mathbf{X}_j$, denoted by AVE($\mathbf{X}_j$), is defined as:

$$\text{AVE}(\mathbf{X}_j) = 1/p_j \sum_{h=1}^{p_j} cor^2(\mathbf{x}_{jh}, \mathbf{y}_j) \tag{20}$$

AVE($\mathbf{X}_j$) varies between 0 and 1 and reflects the proportion of variance captured by $\mathbf{y}_j$.

- For all blocks:

$$\text{AVE(outermodel)} = \left(1/\sum_j p_j\right) \sum_j p_j \text{AVE}(\mathbf{X}_j) \tag{21}$$

- For the inner model:

$$\text{AVE(innermodel)} = \left(1/\sum_{j<k} c_{jk}\right) \sum_{j<k} c_{jk} \text{cor}^2(\mathbf{y}_j, \mathbf{y}_k) \tag{22}$$

These indicators of model quality can be obtained as follows:

```
rgcca_B_factorial$AVE
```

```
## $AVE_X
## $AVE_X[[1]]
## comp1 comp2
##     1     1
##
## $AVE_X[[2]]
## comp1 comp2
##     1     1
##
## $AVE_X[[3]]
## comp1 comp2
##     1     1
##
```

```
##
## $AVE_outer_model
## [1] 1 1
##
## $AVE_inner_model
## [1] 0.4834515 0.1338607
```

and corresponds exactly to the results reported in (Tenenhaus and Tenenhaus 2011, see last column of Table 7).

We mention that, for each block j, an "optimal" shrinkage parameter τ_j can be obtained from the Schafer and Strimmer analytical formula (Schäfer and Strimmer 2005) by using the `tau` argument of the rgcca funtion:

```
rgcca_optimal_factorial = rgcca(blocks=A, connection=C, tau = "optimal", scheme = "factoria
scale = FALSE, verbose = FALSE)
```

The optimal shrinkage parameters are given by:

```
rgcca_optimal_factorial$call$tau
```

```
##             [,1]       [,2]       [,3]
## [1,] 0.08853216 0.02703256 0.08422566
## [2,] 0.07206153 0.04272402 0.18870704
```

This automatic estimation of the shrinkage parameters allows one to come closer to the correlation criterion, even in the case of high multicollinearity or when the number of individuals is smaller than the number of variables.

At last, as a component-based method, RGCCA provides the users with graphical representations, including factor plot, correlation circle. This graphical displays allows visualizing the sources of variability within blocks, the relationships between variables within and between blocks and the amount of correlation between blocks. The graphical display of the countries obtained by crossing $\mathbf{X}_1\mathbf{a}_1$ = Agricultural Inequality and $\mathbf{X}_2\mathbf{a}_2$ = Industrial Development and marked with their political regime in 1960 is shown in below.

```
df = data.frame(political_regime =
factor(apply(Russett[, 9:11], 1, which.max),
labels = c("demostab", "demoinst", "dictator")),
comp1 = rgcca_B_factorial$Y[[1]][, 1],
comp2 = rgcca_B_factorial$Y[[2]][, 1])

p <- ggplot(df, aes(comp1, comp2)) +
geom_vline(xintercept = 0) +
geom_hline(yintercept = 0) +
ggtitle("Factor plot (Russett data)") +
geom_text(aes(colour = political_regime, label= rownames(df)),
vjust=0,nudge_y = 0.03,size = 3 )+
theme(legend.position="bottom", legend.box = "horizontal",
legend.title = element_blank())

p
```

Countries aggregate together when they share similarities. It may be noted that the lower left quadrant concentrates on dictatorships. It is difficult for a country to escape dictatorship when its industrial development is below-average and its agricultural inequality is above average. It is worth pointing out that some unstable democracies located in this quadrant (or close to it) became dictatorships for a period of time after 1960: Greece (1967-1974), Brazil (1964-1985), Chili (1973-1990), and Argentina (1966-1973).

Moreover, in the framework of consensus PCA and Hierarchical PCA, a superblock defined as the concatenation of all the blocks is also used and global components can be derived. The space spanned by the global components is viewed as a compromise space that integrated all the modalities and facilitates the visualization of the results and their interpretation.

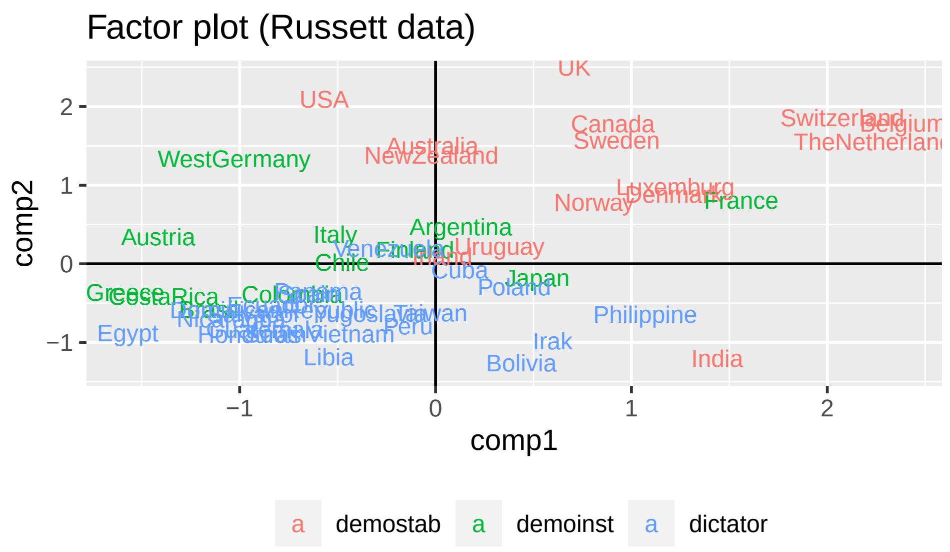


Figure 2: graphical display of the countries obtained by crossing y1 and y2 and labeled according to their political regime

Here, 2 components per block are specified using the `ncomp = rep(2, 4)` argument (default value `ncomp = rep(1, length(A))`, which gives one component per block).

```r
X_agric =as.matrix(Russett[,c("gini","farm","rent")])
X_ind = as.matrix(Russett[,c("gnpr","labo")])
X_polit = as.matrix(Russett[ , c("inst", "ecks",  "death",
"demostab", "demoinst", "dictator")])
A = list(X_agric = X_agric, X_ind = X_ind, X_polit = X_polit,
Superblock = cbind(X_agric, X_ind, X_polit)
)
#Define the design matrix (output = C)
C = matrix(c(0, 0, 0, 1,
0, 0, 0, 1,
0, 0, 0, 1,
1, 1, 1, 0), 4, 4)

rgcca.hpca = rgcca(blocks=A, connection=C, tau = c(1, 1, 1, 0), verbose = FALSE,
ncomp = rep(2, 4),
#flexible design of the scheme function
scheme = function(x) x^4,
scale = TRUE)
```

For HPCA, the countries can be represented in the space spanned by the two first global components.

```r
df1 = data.frame(political_regime =
factor(apply(Russett[, 9:11], 1, which.max),
labels = c("demostab", "demoinst", "dictator")),
rgcca.hpca$Y[[4]]
)

p1 <- ggplot(df1, aes(comp1, comp2)) +
geom_vline(xintercept = 0) +
geom_hline(yintercept = 0) +
ggtitle("Factor plot (Russett data) - Common Space") +
geom_text(aes(colour = political_regime, label= rownames(df1)),
vjust=0,nudge_y = 0.03,size = 3)+
theme(legend.position="bottom", legend.box = "horizontal",
legend.title = element_blank())

p1
```

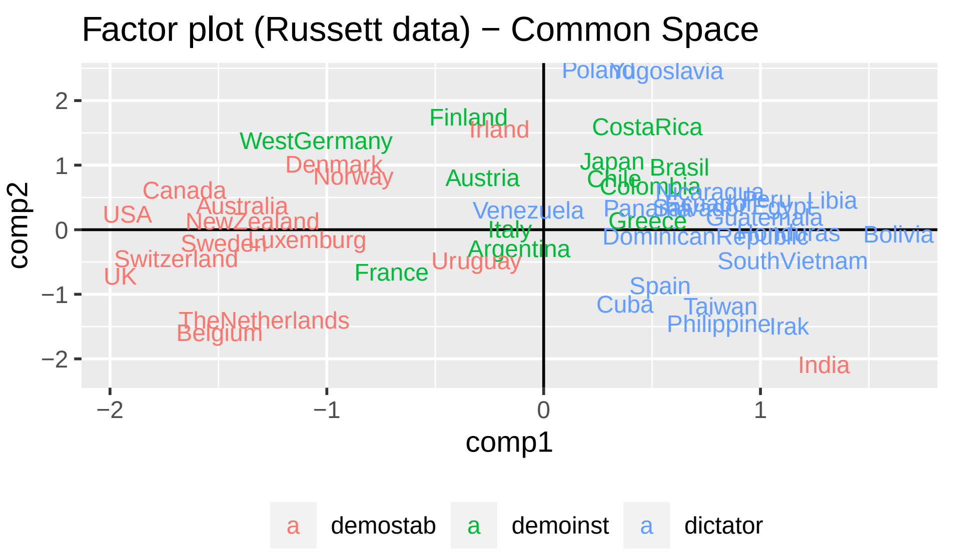


Figure 3: graphical display of the countries obtained by crossing the two first compoent of the superblock and labeled according to their political regime

Despite some overlap, the first global component exhibits a separation/continuum among regimes.

Moreover, the correlation circle plot highlights the contribution of each variable to the contruction of the global components. This figure shows the original variables projected on the compromise space.

```r
df2 = data.frame(comp1 = cor(Russett, rgcca.hpca$Y[[4]])[, 1],
comp2 = cor(Russett, rgcca.hpca$Y[[4]])[, 2],
BLOCK = rep(c("X1", "X2", "X3"), sapply(A[1:3], NCOL)))

# Circle
circleFun <- function(center = c(-1,1),diameter = 1, npoints = 100){
r = diameter / 2
tt <- seq(0,2*pi,length.out = npoints)
xx <- center[1] + r * cos(tt)
yy <- center[2] + r * sin(tt)
return(data.frame(x = xx, y = yy))
}

circle <- circleFun(c(0,0),2,npoints = 100)

p2 <-ggplot(df2, aes(comp1, comp2), colour = BLOCK) +
geom_path(aes(x,y), data=circle) +
geom_vline(xintercept = 0) +
geom_hline(yintercept = 0) +
ggtitle("Correlation Circle (Russett data) - Common Space") +
geom_text(aes(colour = BLOCK, label= rownames(df2)),
vjust=0,nudge_y = 0.03,size = 3)+
theme(legend.position="bottom", legend.box = "horizontal",
legend.title = element_blank())

p2
```

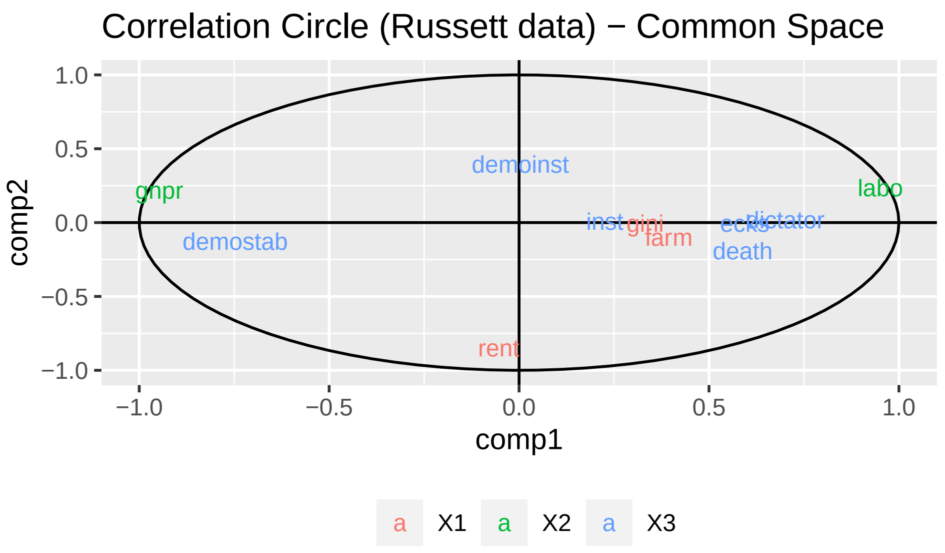


Figure 4: correlation circle plot - two first components of the superblock. Variables are colored according to their block membership

A variable that is highly expressed for a category of individuals is projected with a high weight (far from the origin) in the direction of that category.

Assessment of the reliability of parameter estimates. It is possible to use a bootstrap resampling method to assess the reliability of parameter estimates obtained using RGCCA. B bootstrap samples of the same size as the original data is repeatedly sampled with replacement from the original data. RGCCA is then applied to each bootstrap sample to obtain the R/SGCCA estimates. For RGCCA, we calculate the mean and variance of the estimates across the bootstrap samples, from which we derived, bootstrap confidence intervals, t-ratio and p-value (under the assumption that the parameter estimates exhibited asymptotic normality) to indicate how reliably parameters were estimated. Since several p-values are constructed simultaneously, Bonferroni or FDR corrections can be applied for controlling the Family-Wise Error Rate or the False Discovery Rate, respectively. This function is not yet available in the current version of the RGCCA package and is available upon request.

3.2 RGCCA (dual formulation) for the Glioma dataset

Biological problem. Brain tumors are the most common solid tumors in children and have the highest mortality rate of all pediatric cancers. Despite advances in multimodality therapy, children with pHGG invariably have an overall survival of around 20% at 5 years. Depending on their location (e.g. brainstem, central nuclei, or supratentorial), pHGG present different characteristics in terms of radiological appearance, histology, and prognosis. Our hypothesis is that pHGG have different genetic origins and oncogenic pathways depending on their location. Thus, the biological processes involved in the development of the tumor may be different from one location to another, as it has been frequently suggested.

Description of the data. Pretreatment frozen tumor samples were obtained from 53 children with newly diagnosed pHGG from Necker Enfants Malades (Paris, France) (Puget et al. 2012). The 53 tumors are divided into 3 locations: supratentorial (HEMI), central nuclei (MIDL), and brain stem (DIPG). The final dataset is organized in 3 blocks of variables defined for the 53 tumors: the first block $\mathbf{X}_1$ provides the expression of 15702 genes (GE). The second block

$\mathbf{X}_2$ contains the imbalances of 1229 segments (CGH) of chromosomes. $\mathbf{X}_3$ is a block of dummy variables describing the categorical variable location. One dummy variable has been left out because of redundancy with he others.

```r
# Download the dataset's package at http://biodev.cea.fr/sgcca/.
# --> gliomaData_0.4.tar.gz

require(gliomaData)
data(ge_cgh_locIGR)

A <- ge_cgh_locIGR$multiblocks
Loc <- factor(ge_cgh_locIGR$y)
levels(Loc) <- colnames(ge_cgh_locIGR$multiblocks$y)
A[[3]] = A[[3]][, -3]

# check dimensions of the blocks
sapply(A, dim)
```

Preprocessing To make variables comparable, standardization is applied (zero mean and unit variance). To make blocks comparable, we divide each block by the square root of its number of variables (Westerhuis, Kourti, and MacGregor 1998). Such a preprocessing is set by default (`scale = TRUE`) when using `rgcca` or `sgcca` and is applied here.

Path diagram. We propose to use the design matrix $\mathbf{C}$ represented in Figure 5.

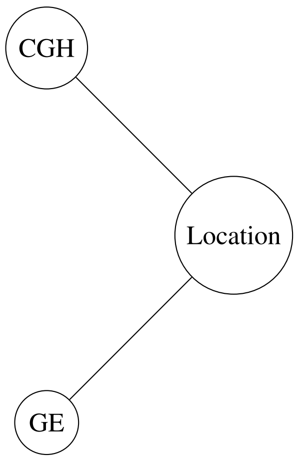


Figure 5: between-block connection for Glioma data.

This design is commonly used in many applications and is oriented toward the prediction of the location. This design does not impose any correlation between GE and CGH blocks and is encoded as follows:

```r
C <-  matrix(c(0, 0, 1, 0, 0, 1, 1, 1, 0), 3, 3)
C
```

```
##      [,1] [,2] [,3]
## [1,]    0    0    1
## [2,]    0    0    1
## [3,]    1    1    0
```

Determination of the shrinkage parameters. In this situation, the optimal shrinkage parameters should be chosen so as to minimize the test error rate determined by cross-validation (grid search). This has been done in (Arthur Tenenhaus, Philippe, and Frouin 2015). A function that implements this cross validation procedure is not yet included in the RGCCA package but can be found in http://biodev.cea.fr/sgcca/. Work is in progress to include this function within the RGCCA package. Due to the dimensionality of each block, the shrinkage parameters $\tau_1 = 1$, $\tau_2 = 1$ and $\tau_3 = 0$ are found to offer a good compromise between stability and correlation (Arthur Tenenhaus, Philippe, and Frouin 2015).

Notice that τ_3 was set to 0 because we were not looking for a block component $\mathbf{y}_3$ that explained its own block well (since $\mathbf{X}_3$ is a group coding matrix) but one that correlated with its neighboring components. In addition, usin the `ncomp` argument, one component per block have been asked (`ncomp = c(1, 1, 1)`).

From the dimension of each block ($n > p$ or $n \leq p$), the `rgcca()` function selects automatically the dual formulation for $\mathbf{X}_1$ and $\mathbf{X}_2$ and the primal one for $\mathbf{X}_3$.

```
rgcca.glioma = rgcca(blocks=A, connection=C, tau = c(1, 1, 0), ncomp = c(1, 1, 1),
                     scale = TRUE, scheme = "horst", verbose = FALSE)
```

The formulation used for each block is returned using the following command:

```
rgcca.glioma$primal_dual
```

The dual formulation make the RGCCA algorithm highly efficient even in a high dimensional setting.

```
system.time(rgcca(blocks=A, connection=C, tau = c(1, 1, 0), ncomp = c(1, 1, 1),
                  scheme = "horst", verbose = FALSE)
)
```

As previously, RGCCA enables visual inspection of the spatial relationships between classes. This facilitates assessment of the quality of the classification and makes it possible to readily determine which components capture the discriminant information.

```
df1 = data.frame(Loc = Loc,
                 GE1 = rgcca.glioma$Y[[1]][, 1],
                 CGH1 = rgcca.glioma$Y[[2]][, 1])

p1 <- ggplot(df1, aes(GE1, CGH1)) +
  geom_vline(xintercept = 0) +
  geom_hline(yintercept = 0) +
  ggtitle("Factor plot (Glioma data) - GE block") +
  geom_text(aes(colour = Loc, label= rownames(df1)),
            vjust=0,nudge_y = 0.03,size = 3 )+
  theme(legend.position="bottom", legend.box = "horizontal",
        legend.title = element_blank())

p1
```

In high-dimensional settings, the highest level of regularization ($\tau_j = 1$) has proven to be necessary or even insufficient. Additional penalties that promote sparsity are often required.

3.3 SGCCA for the Glioma dataset

The variable selection is achieved by adding, within the RGCCA optimization problem, a penalty promoting sparsity. For that purpose, an ℓ_1 penalization on the weight vectors $\mathbf{a}_1, \ldots, \mathbf{a}_J$ is applied. the `sparsity` argument of `rgcca()` vary between 0 and 1 (larger values of `sparsity` correspond to less penalization) and control the amount of sparsity of the weight vectors $\mathbf{a}_1, \ldots, \mathbf{a}_J$. If `sparsity` is a vector, ℓ_1-penalties are the same for all the weights corresponding to the same block but different components:

$$\forall h, \|\mathbf{a}_j^{(h)}\|_{\ell_1} \leq c_{1j}\sqrt{p_j}, \tag{23}$$

with p_j the number of variables of $\mathbf{X}_j$.

If `sparsity` is a matrix, each row h of `sparsity` defines the constraints applied to the weights corresponding to components h:

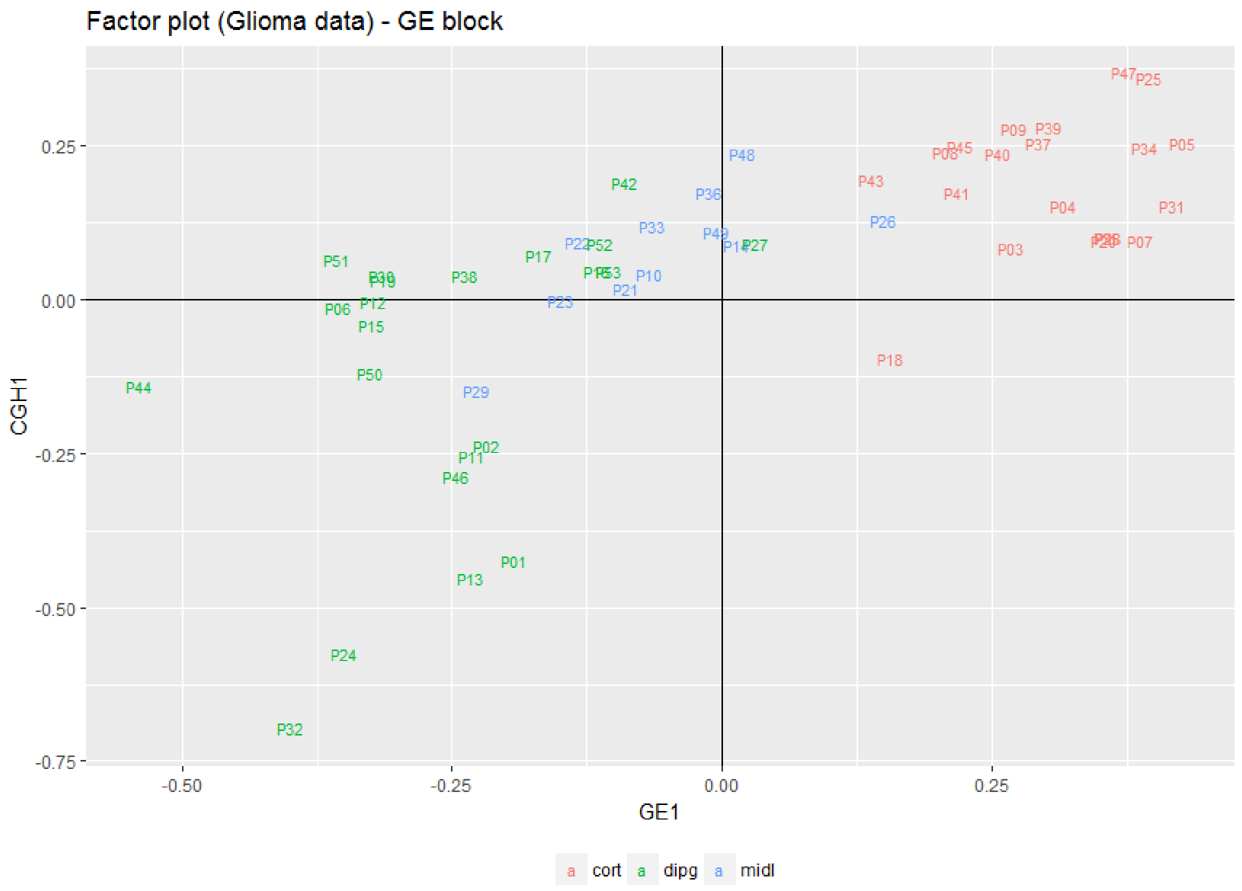


Figure 6: RGCCA dual formulation. Graphical display of the tumors obtained by crossing GE1 and CGH1 and labeled according to their location.

$$\forall h, \|\mathbf{a}_j^{(h)}\|_{\ell_1} \leq c_1[h,j]\sqrt{p_j}. \tag{24}$$

```
# sgcca algorithm
sgcca.glioma = rgcca(blocks=A, connection=C,type="sgcca", sparsity = c(.071,.2, 1),
                     ncomp = c(1, 1, 1),
                     scheme = "centroid",
                     scale = TRUE,
                     verbose = FALSE)
```

For GE, the number of non-zero elements of $\mathbf{a}_1$ is equal to 145 and is given by

```
nb_zero_GE = sum(sgcca.glioma$a[[1]] != 0)
```

For CGH, the number of non-zero elements of $\mathbf{a}_2$ is equal to 92 and is given by

```
nb_zero_CGH = sum(sgcca.glioma$a[[2]] != 0)
```

One component per block has been built (GE1 for $\mathbf{X}_1$ and CGH1 $\mathbf{X}_2$), and the graphical display of the tumors obtained by crossing GE1 and CGH1 and labeled according to their location is shown below.

```
df1 = data.frame(Loc = Loc,
                 GE1 = sgcca.glioma$Y[[1]][, 1],
                 CGH1 = sgcca.glioma$Y[[2]][, 1])

p1 <- ggplot(df1, aes(GE1, CGH1)) +
    geom_vline(xintercept = 0) +
    geom_hline(yintercept = 0) +
    ggtitle("Factor plot (Glioma data)") +
    geom_text(aes(colour = Loc, label= rownames(df1)),
              vjust=0,nudge_y = 0.03,size = 3 )+
    theme(legend.position="bottom", legend.box = "horizontal",
          legend.title = element_blank())

p1
```

We observe that the GE block contains much more discriminative information than the CGH block.

3.3.1 SGCCA with a superblock

In this subsection we consider the between-block connections described in Figure 8.

```
A <- ge_cgh_locIGR$multiblocks
Loc <- factor(ge_cgh_locIGR$y)
levels(Loc) <- colnames(ge_cgh_locIGR$multiblocks$y)
A[[3]] = A[[3]][, -3]
B = A
B[[3]] = Reduce("cbind", A[1:(length(A)-1)]) # superblock
B[[4]] = A[[3]]
sapply(B, dim) # check dimension
```

and encoded in the design matrix as follows:

```
nb_block = 4 #length(B)
C = matrix(0, nb_block, nb_block)
C[, nb_block-1] = 1 ; C[nb_block-1, ] = 1 ; diag(C) = 0
C
```

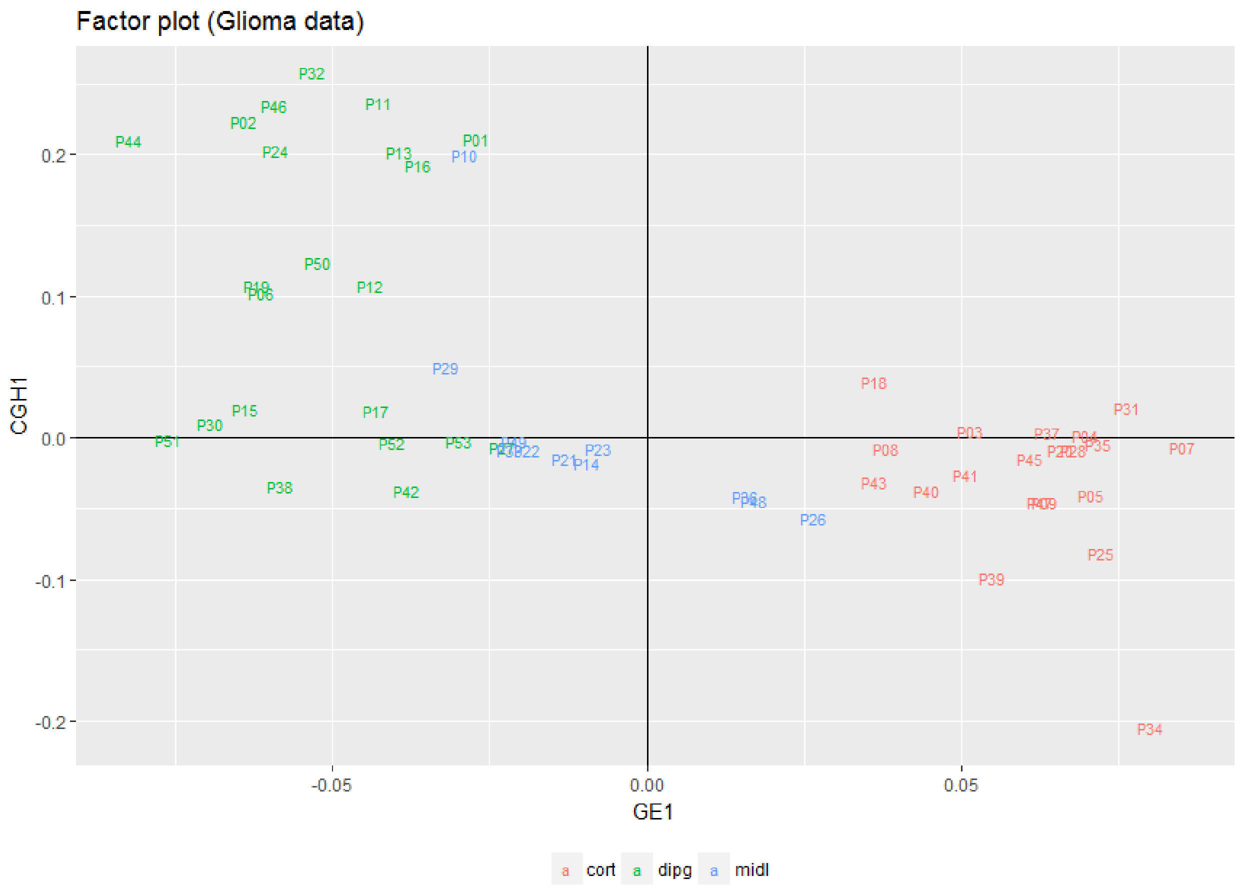


Figure 7: SGCCA. Graphical display of the tumors obtained by crossing GE1 and CGH1 and labeled according to their location.

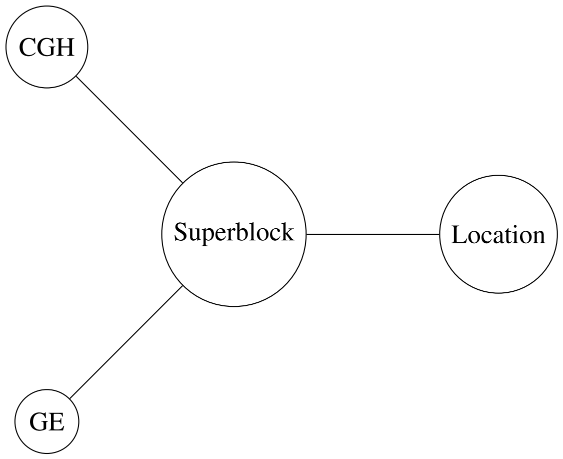


Figure 8: between-block connection for Glioma data with the superblock

```
##      [,1] [,2] [,3] [,4]
## [1,]    0    0    1    0
## [2,]    0    0    1    0
## [3,]    1    1    0    1
## [4,]    0    0    1    0
```

In that case, the call of `rgcca(,type="sgcca")` is as follows:

```
sgcca.glioma = rgcca(blocks=B, connection=C,type="sgcca",
                     sparsity = c(.071,.2, 1/sqrt(NCOL(B[[3]]))+.2, 1),
                     ncomp = c(rep(2, (length(B)-1)),1),
                     scheme = "factorial", scale = TRUE,
                     verbose = FALSE
)
```

and allows for the graphical representations of the individuals and the variables in a common space.

```
df1 = data.frame(Loc = Loc, sgcca.glioma$Y[[3]])

p1 <- ggplot(df1, aes(comp1, comp2)) +
    geom_vline(xintercept = 0) +
    geom_hline(yintercept = 0) +
    ggtitle("Factor plot (Glioma data) - super block") +
    geom_text(aes(colour = Loc, label= rownames(df1)),
              vjust=0,nudge_y = 0.03,size = 3)+
    theme(legend.position="bottom", legend.box = "horizontal",
          legend.title = element_blank())
p1
```

```
ind.select = unlist(lapply(sgcca.glioma$a[1:(length(B)-2)],
                           function(x) x[, 1]!=0 | x[, 2]!=0))
VAR.SELECT = B[[(length(B)-1)]][, ind.select]
BLOCK = rep("CGH", NCOL(VAR.SELECT))
BLOCK[grep("A", colnames(VAR.SELECT))] = "GE"

df = data.frame(comp1 = cor(VAR.SELECT, sgcca.glioma$Y[[(length(B)-1)]][, 1]),
                comp2 = cor(VAR.SELECT, sgcca.glioma$Y[[(length(B)-1)]][, 2]),
                BLOCK)
```

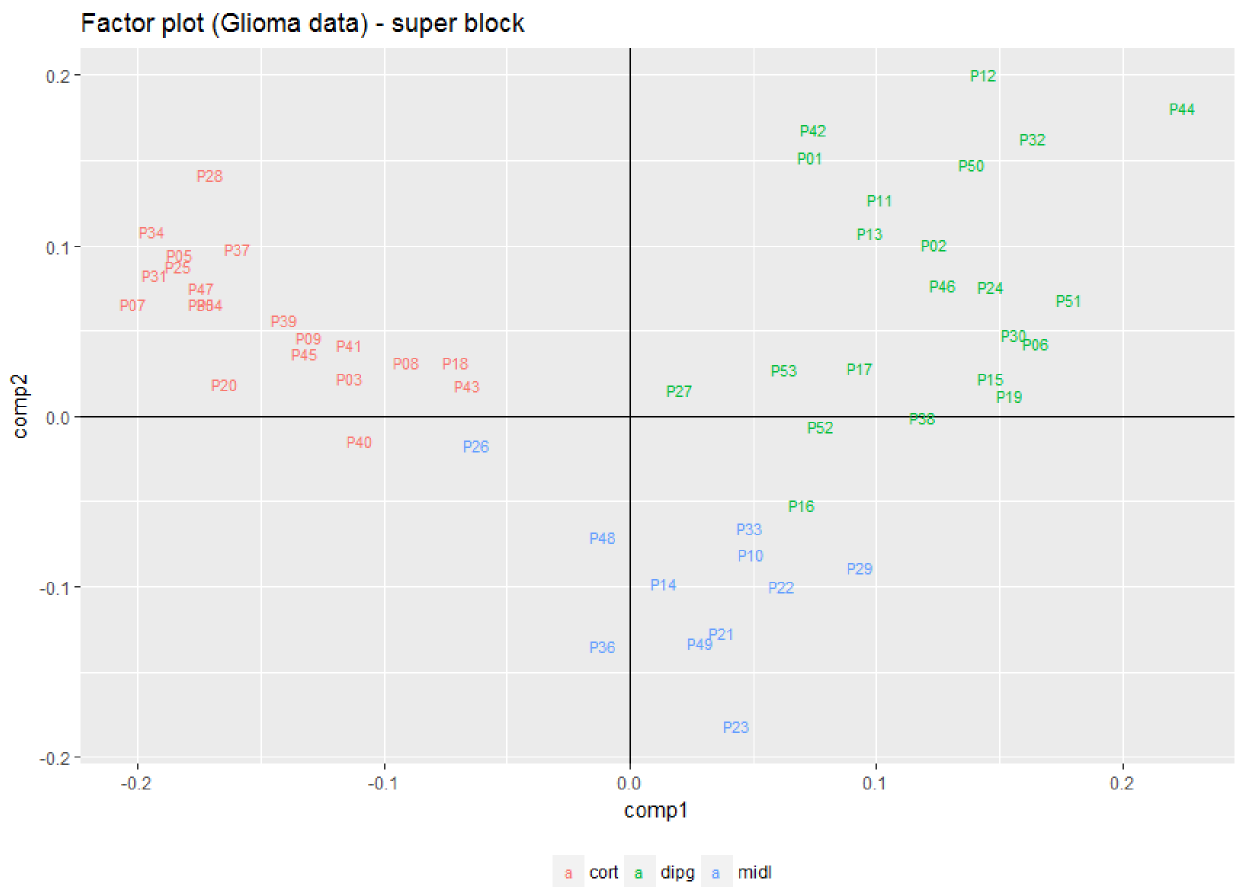


Figure 9: SGCCA with the superblock. Graphical display of the tumors obtained by crossing the two first superblock components and labeled according to their location.

```
# Circle
circleFun <- function(center = c(-1,1),diameter = 1, npoints = 100){
    r = diameter / 2
    tt <- seq(0,2*pi,length.out = npoints)
    xx <- center[1] + r * cos(tt)
    yy <- center[2] + r * sin(tt)
    return(data.frame(x = xx, y = yy))
}

circle <- circleFun(c(0,0),2,npoints = 100)

p <-ggplot(df, aes(comp1, comp2), colour = BLOCK) +
    geom_path(aes(x,y), data=circle) +
    geom_vline(xintercept = 0) +
    geom_hline(yintercept = 0) +
    ggtitle("Correlation Circle") +
    geom_text(aes(colour = BLOCK, label= rownames(df)),
              vjust=0,nudge_y = 0.03,size = 2)+
    theme(legend.position="bottom", legend.box = "horizontal",
          legend.title = element_blank())

p
```

Moreover, it is possible to use a bootstrap resampling method to assess the stability of the variable selection process of SGCCA. B (e.g. 500) bootstrap samples of the same size as the original data is repeatedly sampled with replacement from the original data. SGCCA is then applied to each bootstrap sample to obtain the SGCCA estimates. The percentage of times a specific variable had a non-null weight across bootstrap sample can be derived. Work is in progress to include this function within the RGCCA package but an R code is given below.

```
nb_boot = 500 # number of bootstrap samples
J = length(B)-2
STAB = list()
for (j in 1:J)
  STAB[[j]] = matrix(0, nb_boot, NCOL(B[[j]]))

sparsity = c(.071,.2, 1/sqrt(NCOL(B[[3]]))+.2, 1)

B = lapply(B, cbind)
for (i in 1:nb_boot){
  ind = sample(NROW(B[[1]]), replace = TRUE)
  Bscr = lapply(B, function(x) x[ind, ])
  res = sgcca(Bscr, C, sparsity = sparsity, ncomp = c(rep(1, length(B))),
              scheme = "factorial", scale = TRUE)
  for (j in 1:J) STAB[[j]][i, ] = res$a[[j]]
}

for (j in 1:J) colnames(STAB[[j]]) = colnames(B[[j]])

top = 30
count = lapply(STAB, function(x) apply(x, 2, function(y) sum(y!=0)))
count_sort = lapply(count, function(x) sort(x, decreasing = TRUE))
```

The 30 top variables for GE and CGH are represented below:

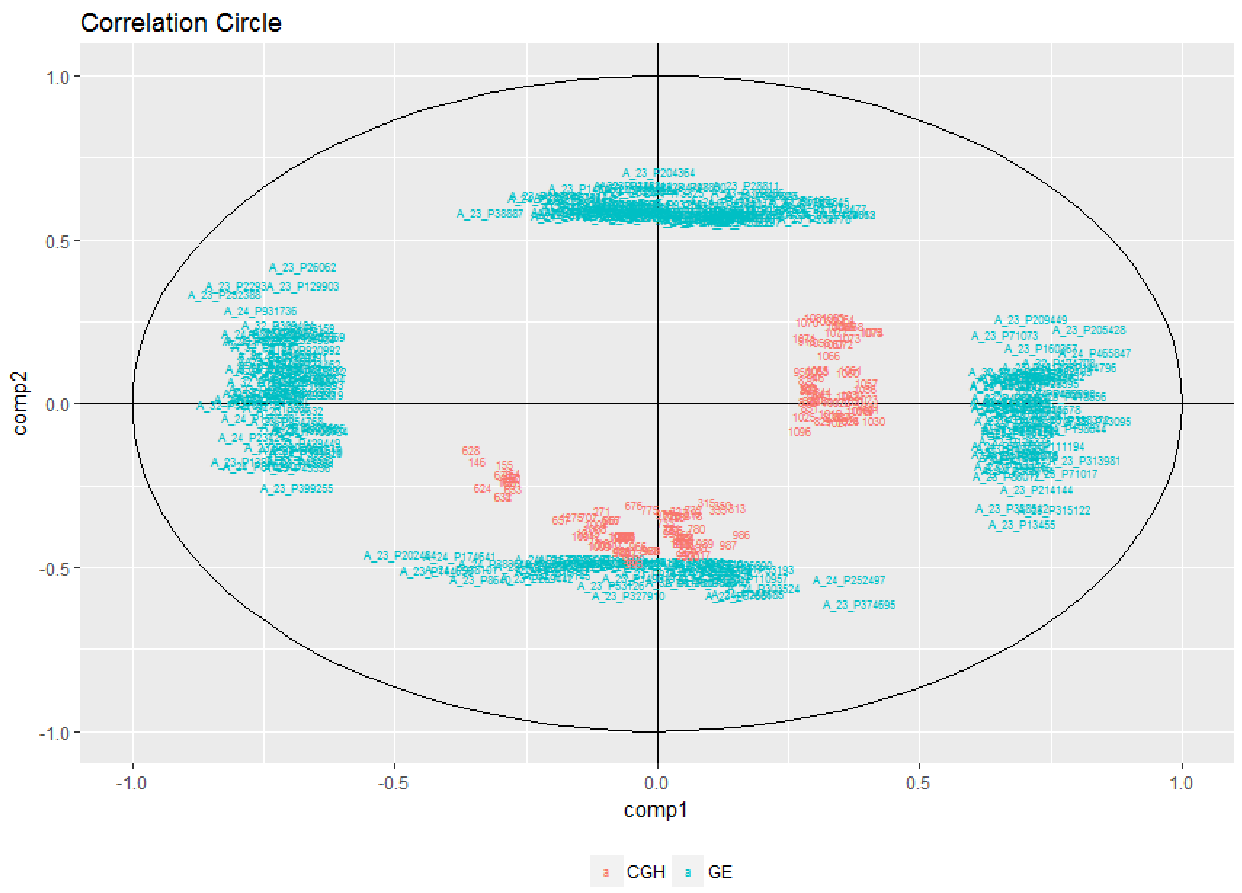


Figure 10: SGCCA with the superblock. correlation circle plot - two first components of the superblock. Variables are colored according to their block membership.

```
stabGE = data.frame(GE_symbol = names(count_sort[[1]])[1:top],
                    Count = count_sort[[1]][1:top]/nb_boot)

g1 <- ggplot(stabGE, aes(x = reorder(GE_symbol, Count), y = Count)) +
      coord_flip() + geom_bar(stat = "identity")
g1
```

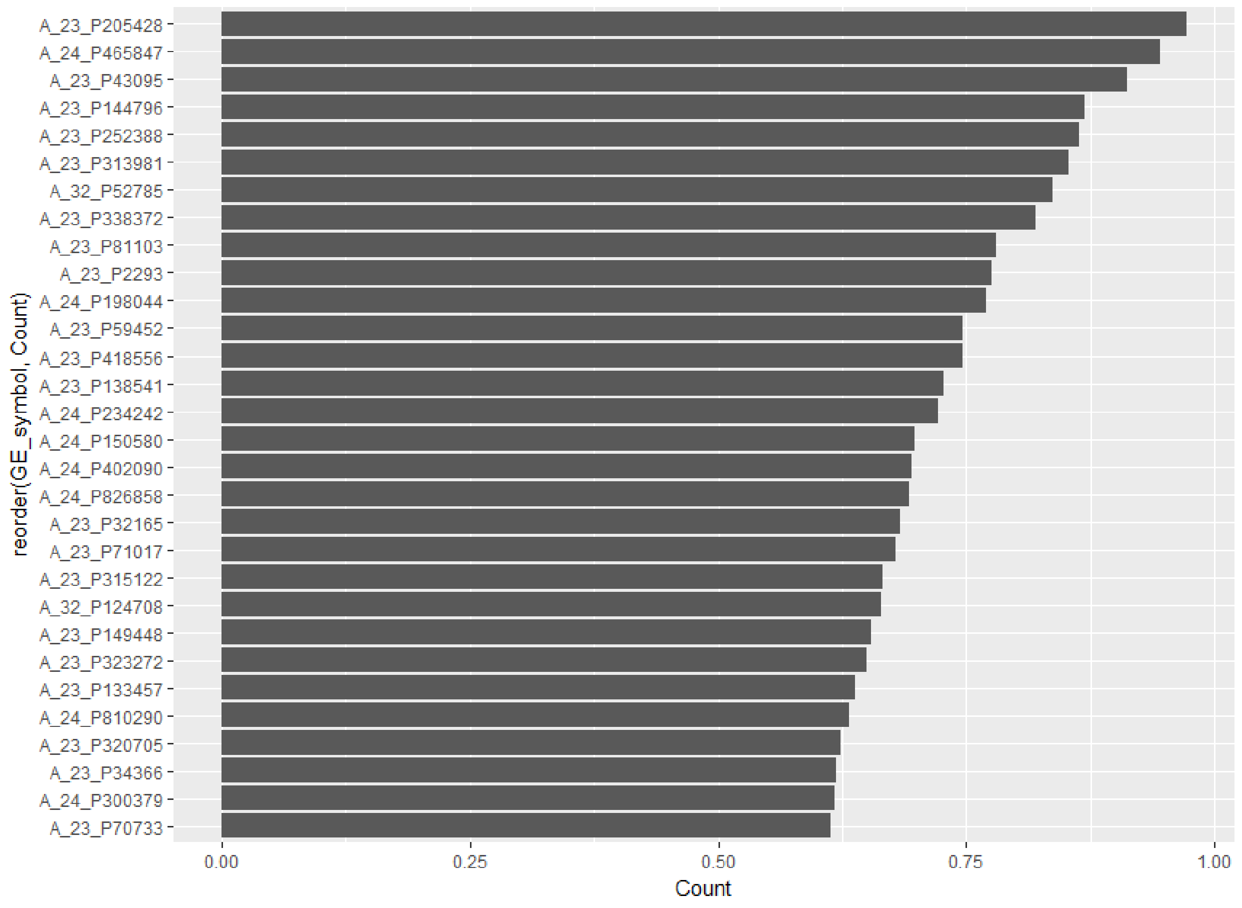


Figure 11: Stability selection for GE.

```
stabCGH = data.frame(CGH_symbol = names(count_sort[[2]])[1:top],
                     Count = count_sort[[2]][1:top]/nb_boot)

g2 <- ggplot(stabCGH, aes(x = reorder(CGH_symbol, Count), y = Count)) +
      coord_flip() + geom_bar(stat = "identity")
g2
```

At last, the McKeon's measure quantifies the homogeneity of a set of block components and is defined by the following formula:

$$r_I(\mathbf{X}_1\mathbf{a}_1, \dots, \mathbf{X}_J\mathbf{a}_J) = \frac{\frac{1}{(J(J-1)/2} \sum_{j<k} \operatorname{cov}(\mathbf{X}_j\mathbf{a}_j, \mathbf{X}_k\mathbf{a}_k)}{1/J \sum_j \operatorname{var}(\mathbf{X}_j\mathbf{a}_j)} \tag{25}$$

Formula (25) allows evaluating the homogeneity of the solution of any multiblock component methods. The McKeon

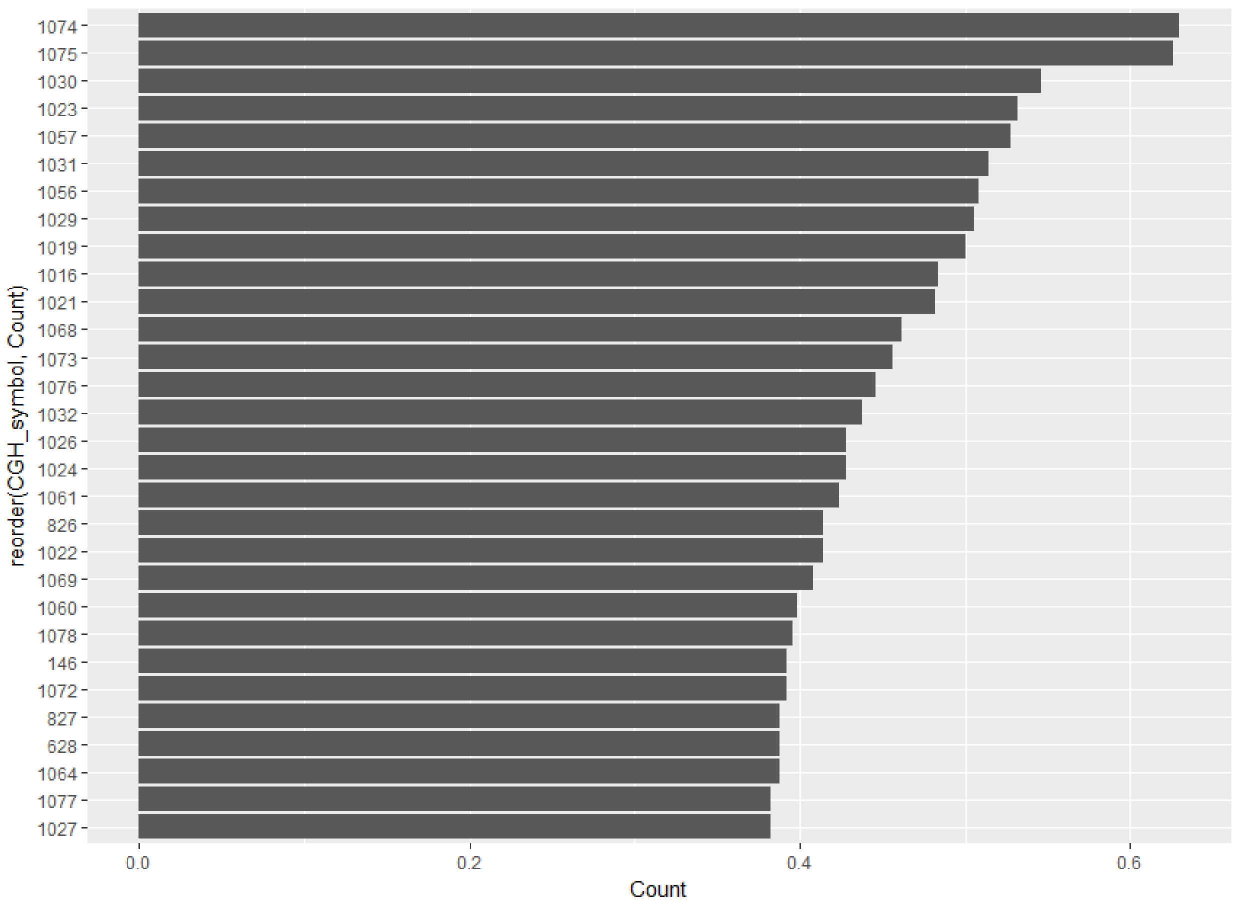


Figure 12: Stability selection for CGH.

measure (not yet implemented within the RGCCA package but easily derived from the `rgcca()` function, see below) can be used to evaluate the proximities of the 3 blocks. The computation of the McKeon's measure was carried out using RGCCA (full between-block connections, $\tau_j = 1$ for all blocks, and $g(x) = x$ for a fair analysis).

```
rI = function(A){
    J = length(A)
    res = rgcca(blocks=A, scale = TRUE, verbose = FALSE)
    Y = Reduce("cbind", res$Y)
    rI = 1/(J-1)*(cov2(rowSums(Y))/sum(apply(Y, 2, cov2))-1)
}


X1 = B[[1]][, unique(which(sgcca.glioma$a[[1]]!=0, arr.ind = TRUE)[, 1])]
X2 = B[[2]][, unique(which(sgcca.glioma$a[[2]]!=0, arr.ind = TRUE)[, 1])]
X3 = B[[4]]
A = list(GE = X1, CGH = X2, LOC = X3)

J = length(A)
M = matrix(0, J, J)
colnames(M) = rownames(M) = names(A)
for (i in 1:J){
  for (j in i:J){
    M[i, j] = rI(A[c(i, j)])
    M[j, i] = M[i, j]
  }
}

M
```

```
> M
           GE       CGH       LOC
GE  1.0000000 0.6687260 0.7886703
CGH 0.6687260 1.0000000 0.3741613
LOC 0.7886703 0.3741613 1.0000000
```

This suggests that the tumor location in the brain is more dependent on gene expression than on chromosomal imbalances and that this dependency is stronger that the dependency between gene expression and chromosomal imbalances.

4 Plots

4.1 Multiblock data analysis with the RGCCA package

We consider J data matrices X1 ,. . . , XJ. Each n × pj data matrix Xj = [xj1, . . . , xjpj] is called a block and represents a set of pj variables observed on n individuals. The number and the nature of the variables may differ from one block to another, but the individuals must be the same across blocks. We assume that all variables are centered. The objective of RGCCA is to find, for each block, a weighted composite of variables (called block component) yj = Xj . aj, j = 1 ,. . . , J (where aj is a column-vector with pj elements) summarizing the relevant information between and within the blocks. The block components are obtained such that (i) block components explain well their own block and/or (ii) block components that are assumed to be connected are highly correlated. In addition, RGCCA integrates a variable selection procedure, called SGCCA, allowing the identification of the most relevant features.

```
## [1] "/home/ecamenen/bin/RGCCA/vignettes"
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
## [[4]]
## NULL
##
## [[5]]
## NULL
```

4.2 Load the inputs

4.2.1 Load the blocks

The blocks are loaded with the function `load_blocks`. The first argument of this function (`superblock`) required a bolean giving the presence (TRUE) / absence (FALSE) of a superblock. The second one corresponds to a character giving the list of the file path separated by a comma (argument `file`). By default, the name of the blocks corresponds to those of the files (`names` argument) and could be set. By default, the tabulation is used as a column separator (`sep` argument) and the first row is considered as a header (`header` parameter).

```
library(rlang)
#>
#> Attaching package: 'rlang'
#> The following object is masked from 'package:igraph':
#>
#>     is_named
# Warning : separators by default are tabulation
blocks = load_blocks(file = "data/agriculture.tsv,data/industry.tsv,data/politic.tsv")
```

4.2.2 Load the groups of response and the connection between blocks

The connection between the blocks will be used by the RGCCA and must be set by `set_connection` function. A group of samples will be used to color them in the samples plot and must be set by `load_response` function. For both functions, the `blocks` parameter, set at the previous step, is required. The other parameters are optional. The user could import a file containing either (`file` parameter) : (i) a symmetric matrix with 1 giving a connection between two blocs, or 0 otherwise; (ii) a univariate vector (qualitative or quantitative) or a disjunctive table for the response. By default, the column separator is the tabulation and could be set (`sep` argument). For the `load_response`, a header could be specified (`header` parameter).

```
# Optional parameters
response <- connection <- NULL

# Uncomment the parameters below to try without this settings
response = load_response(blocks = blocks, file = "data/response.tsv")
connection = load_connection(file = "data/connection.tsv")
```

4.2.3 View the inputs

Table 2: agriculture

	gini	farm	rent
Argentina	86.3	98.2	3.52
Australia	92.9	99.6	3.27
Austria	74	97.4	2.46
Belgium	58.7	85.8	4.15
Bolivia	93.8	97.7	3.04
Brasil	83.7	98.5	2.31

Table 3: industry

	gnpr	labo
Argentina	5.92	3.22
Australia	7.1	2.64
Austria	6.28	3.47
Belgium	6.92	2.3
Bolivia	4.19	4.28
Brasil	5.57	4.11

Table 4: politic

	inst	ecks	death	demostab	demoinst	dictator
Argentina	0.07	4.06	5.38	0	1	0
Australia	0.01	0	0	1	0	0
Austria	0.03	1.61	0	0	1	0
Belgium	0.45	2.2	0.69	1	0	0
Bolivia	0.37	3.99	6.5	0	0	1
Brasil	0.45	3.91	0.69	0	1	0

Table 5: response

Argentina	demoinst
Australia	demostab
Austria	demoinst
Belgium	demostab
Bolivia	dictator
Brasil	demoinst

Table 6: connection

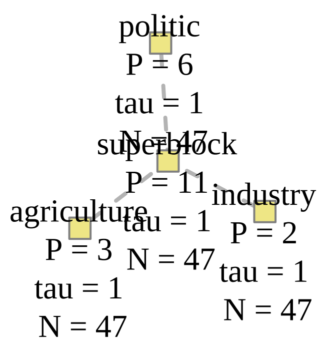
0	0	0	1
0	0	0	1
0	0	0	1
1	1	1	0

4.3 Run S/RGCCA

SGCCA is run from the RGCCA package by using two components for a biplot visualization. The S/RGCCA function doesn't names the blocks in their outputs. This step is required to generate biplots.

```r
rgcca_out = rgcca(blocks = blocks)
plot(rgcca_out,type="network")
```

Common rows between blocks : 47



4.4 Vizualise the analysis

Both the samples and the variables could be visualized by using biplots functions (respectively `plot_ind` and `plot_var_2D`). Histograms are used to visualized in decreasing order the variables with the higher weights and the blocks with the higher Average Variance Explained (AVE).

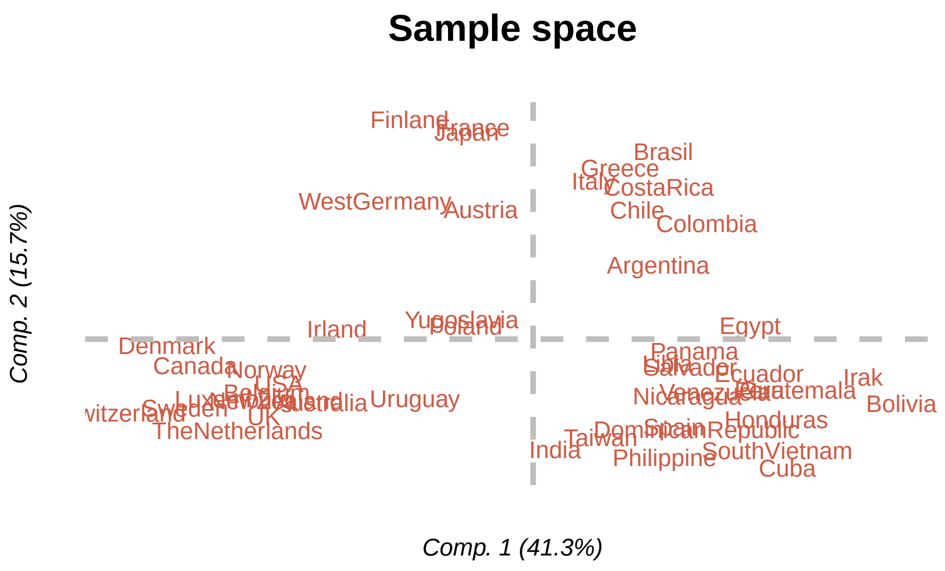
These functions take the results of a sgcca or a rgcca (`rgcca` parameter) and the components to visualize : either `compx` and `compy` for biplots or `comp` for histograms. By default, `compx` = `comp` = 1 and `compy` = 2. The presence or the absence of a superblock among the analysis could be specified for `plot_var_2D` and `plot_var_1D` to color the variables according to their blocks. By default, the last block is plotted, corresponding to the superblock if selected (`i_block` parameter). `plot_var_2D`, which is a corcircle plot, required the `blocks` for the correlation with the selected component. `plot_var_2D` could use the response variable to color the samples by groups. By default, the first 100th higher weights are used for the `plot_var_1D` and could be set by using the `n_mark` argument.

```r
comp1 = 1
comp2 = 2
nmark = 100
```

4.4.1 With the superblock, by default on the first and the second components

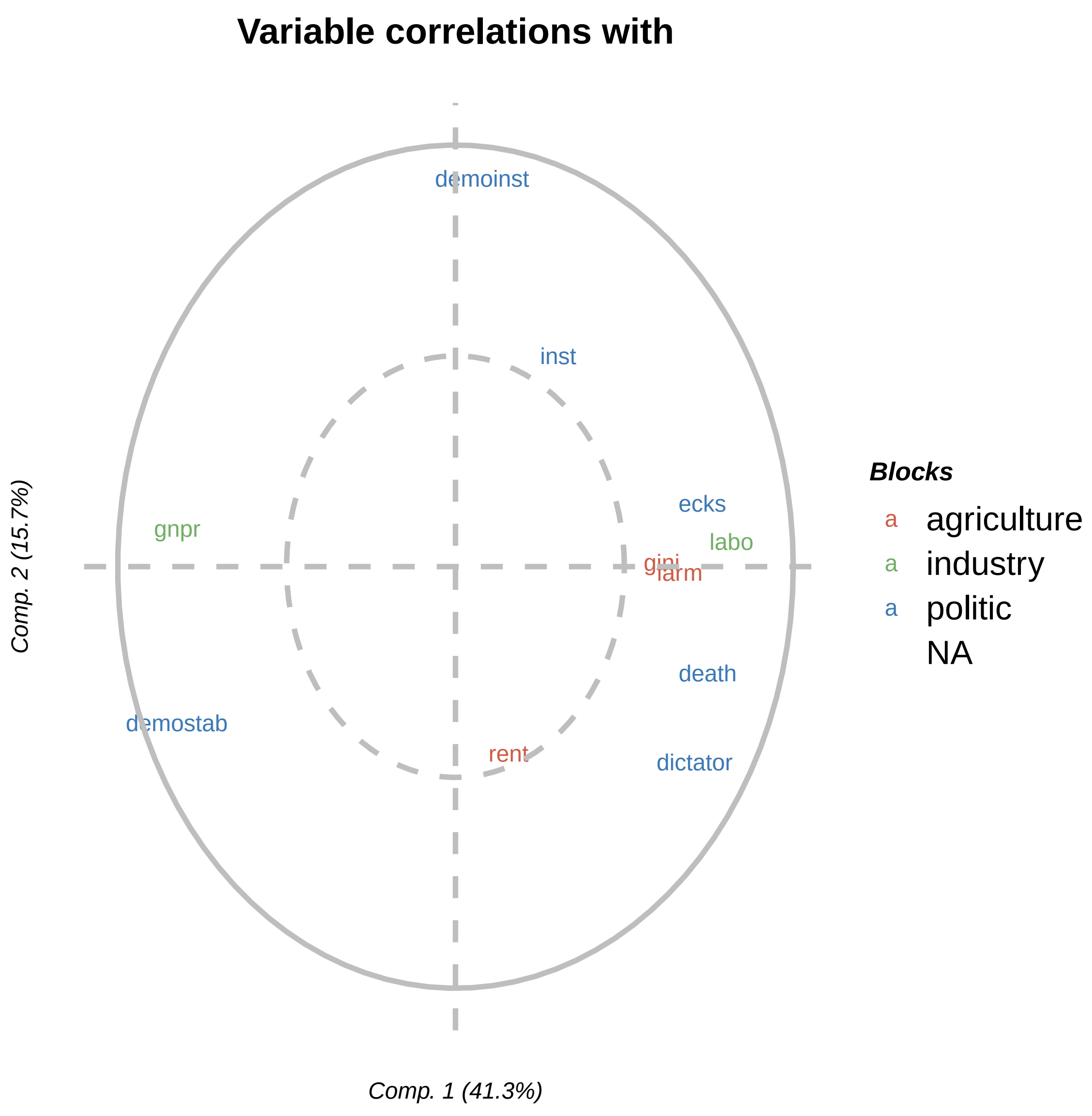
4.4.1.1 Samples plot

```r
plot(rgcca_out,type="ind")
```



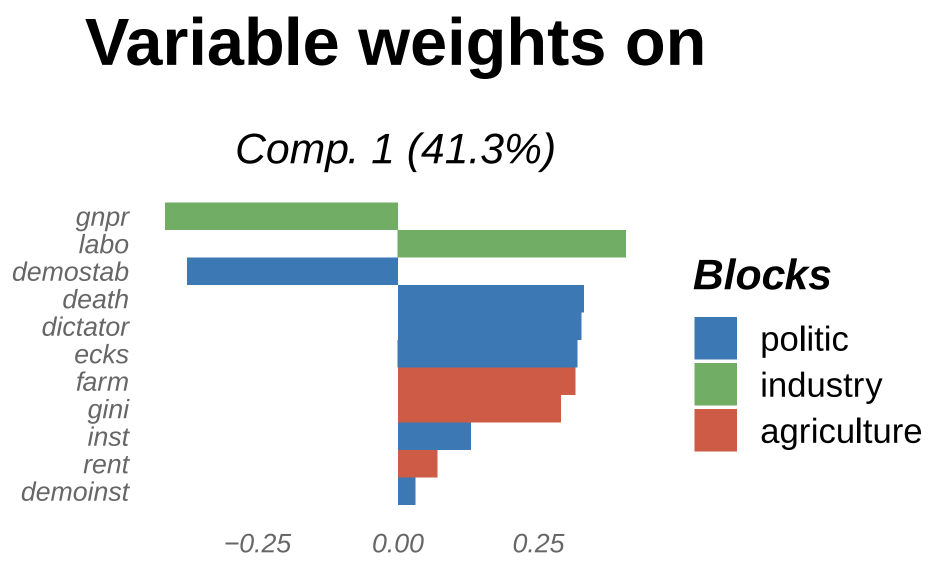
4.4.1.2 Corcircle plot

```r
plot(rgcca_out,type="var")
```



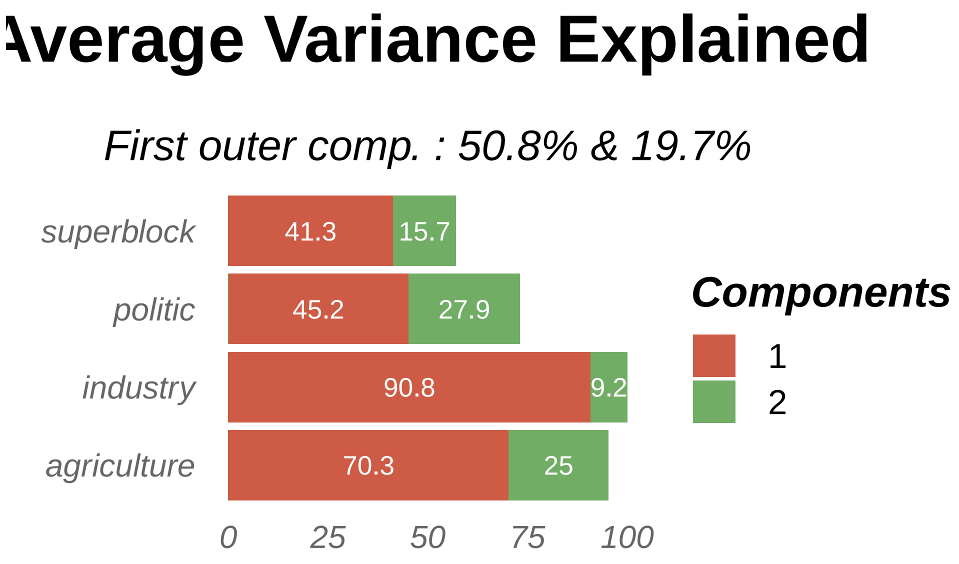
4.4.1.3 Fingerprint plot

```
plot(rgcca_out,type="weight")
```



4.4.1.4 Best explained blocks

```r
plot(rgcca_out,type="ave")
```

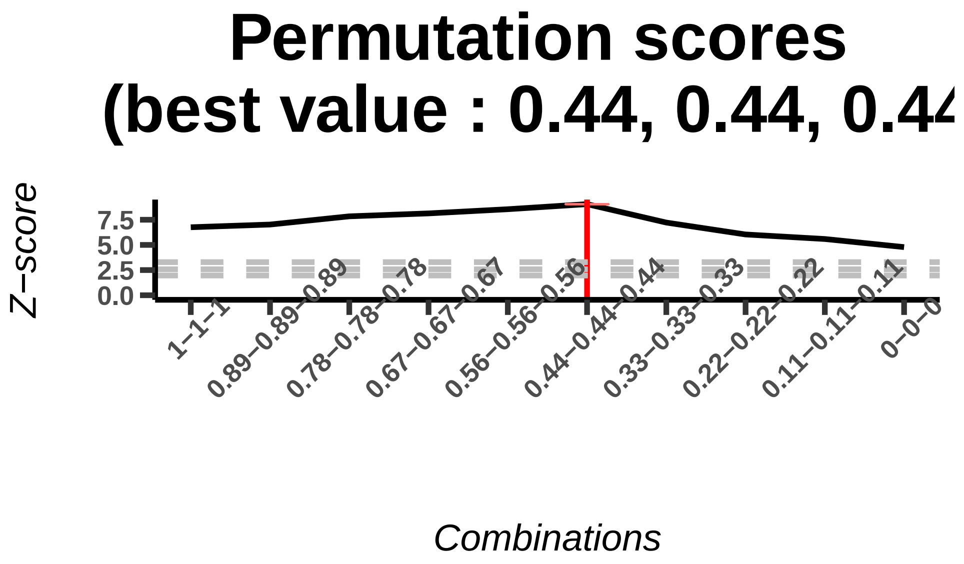


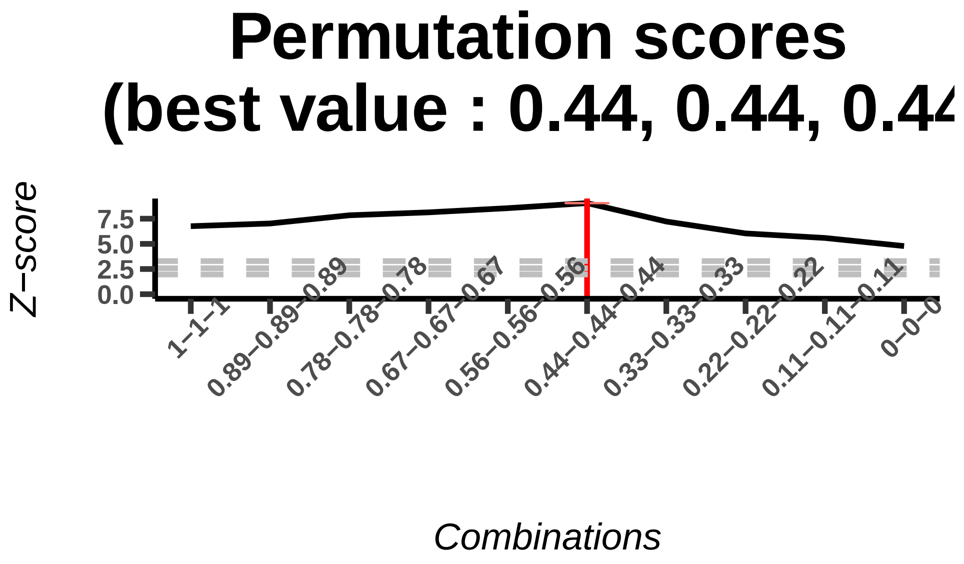
4.4.1.5 Bootstrap

```r
boot <- bootstrap(rgcca_out, n_boot = 2, n_cores = 2)
#> Bootstrap in progress...OK.
#plot(boot, n_cores = 2,type="2D")
#plot(boot, n_cores = 2,type="2D")
```

4.4.1.6 Permutation

```r
perm <- rgcca_permutation(blocks, nperm = 5, n_cores = 1)
#> Permutation in progress...OK.
head(order_df(cbind(perm$penalties, perm$zstat), ncol(perm$penalties) + 1))
#>      agriculture  industry   politic
#> [1,]   0.4444444 0.4444444 0.4444444 9.048892
#> [2,]   0.5555556 0.5555556 0.5555556 8.547246
#> [3,]   0.6666667 0.6666667 0.6666667 8.129590
#> [4,]   0.7777778 0.7777778 0.7777778 7.839339
#> [5,]   0.3333333 0.3333333 0.3333333 7.218131
#> [6,]   0.8888889 0.8888889 0.8888889 7.024268
plot(perm)
```

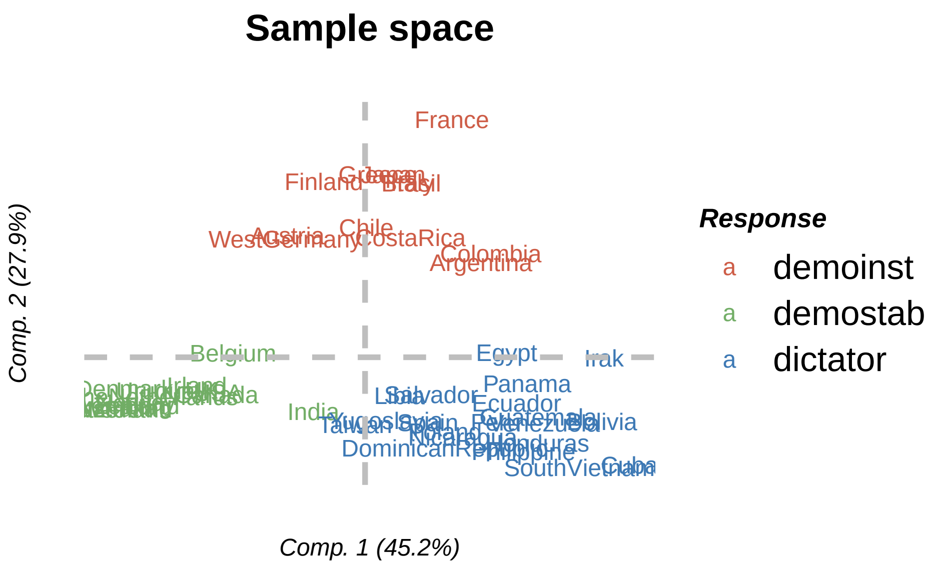




4.4.2 With the politic block, on the 2nd and the 3rd components

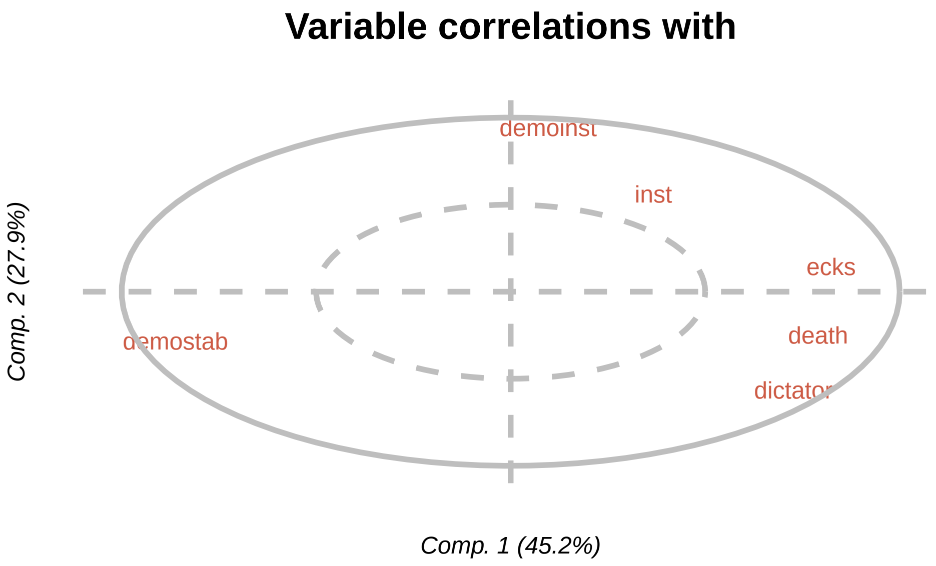
4.4.2.1 Samples plot

```
plot(x = rgcca_out,
    resp = response,
    compx = comp1,
    compy = comp2,
    i_block = 3, type="ind")
```



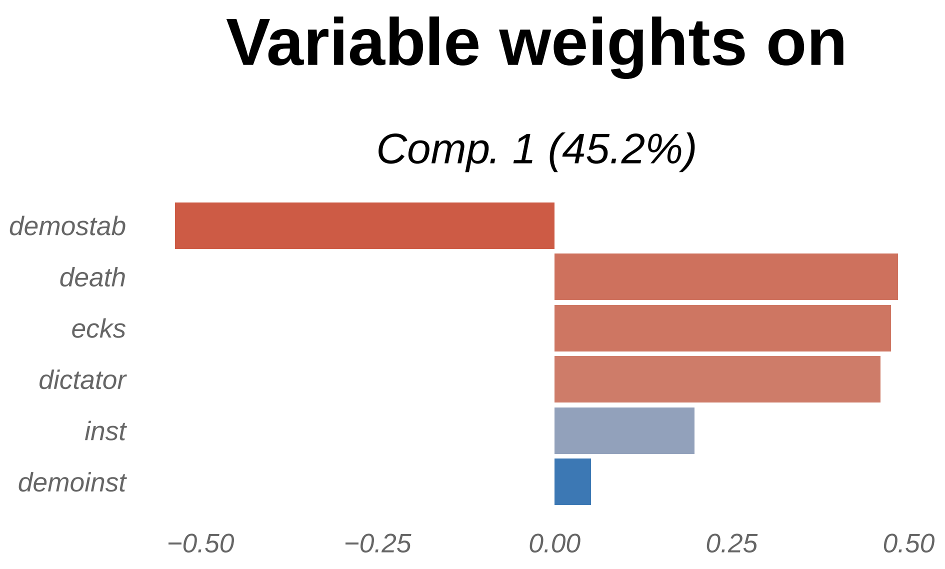
4.4.2.2 Corcircle plot

```
plot(
    x = rgcca_out,
    compx = comp1,
    compy = comp2,
    i_block = 3,type="var")
```



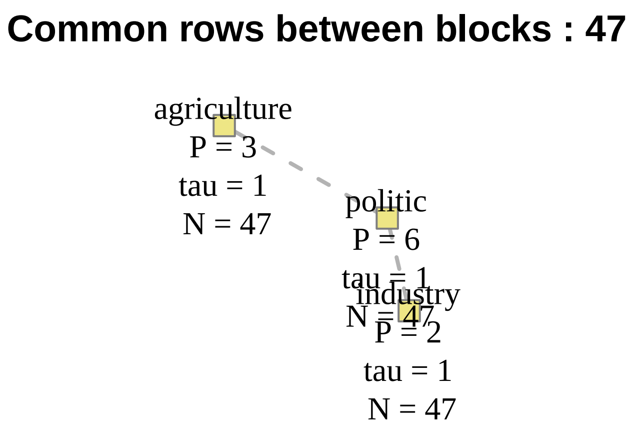
4.4.2.3 Fingerprint plot

```
plot(
    x = rgcca_out,
    compx = comp1,
    n_mark = nmark,
    i_block = 3,
    type = "weight")
```

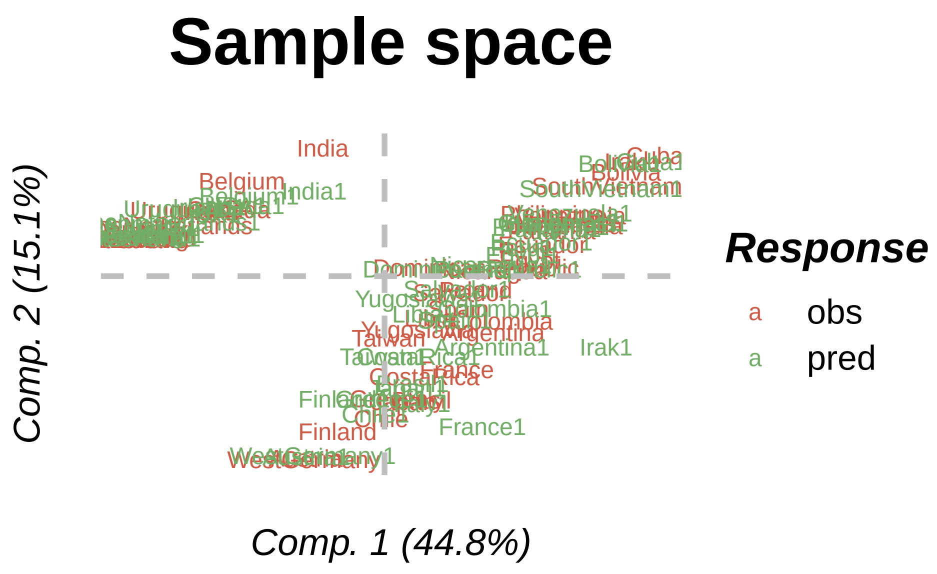


4.4.2.4 Supervized mode

```r
rgcca_out = rgcca(blocks = blocks, response = 3)
plot(rgcca_out,type="network")
```



```r
loo <- rgcca_crossvalidation(rgcca_out, n_cores = 2)
# Cross-validation score (leave-one-out)
round(loo$scores, 3)
#> [1] 0.092
plot_ind(rgcca_out, predicted = loo)
```



5 Dealing with missing data

5.1 Introduction

Multi-source data often have block-wise missing structure, i.e., data in one or more sources may be completely unobserved for a sample. The probability to observe block-wise missing structure increases with the number of sources. It is therefore mandatory to properly handle this block-wise missing structure within the framework of RGCCA. In literature, many solutions exist to deal with missing data: NIPALS-type algorithms, and imputation methods (simple or multiple). Until now, these solutions were not applied to integrative analysis.

In this section, we present several solutions and offer a methodology for choosing the best solution for missing-data according to the dataset. All the tools are available in RGCCA package.

5.2 An example dataset: Russett

The example dataset is Russett.

```r
library(RGCCA)
data(Russett)
head(Russett)
```

The first step of the RGCCA is to define the blocks of variables. Three blocks of variables have been defined for 47 countries. The variables that compose each block have been defined according to the nature of the variables.

```r
X_agric = as.matrix(Russett[,c("gini","farm","rent")])
X_ind = as.matrix(Russett[,c("gnpr","labo")])
X_polit = as.matrix(Russett[ , c("inst", "ecks",  "death")])
A = list(X_agric, X_ind, X_polit)
```

5.3 Creating and vizualizing dataset with missing values

5.3.1 Creating a dataset with missing values from a complete dataset

The function createNA transforms a complete dataset into a dataset with missing values. The parameter pNA specifies the probability of missing values around the dataset (pNA is a proportion), by block (if pNA is a vector of length(blocks)) or by variable (if pNA is a list of length (blocks)). The minimum number of complete individuals can be specified in the option “nAllRespondants”. Two options are available for the missing data: either they are missing by block (typeNA=“block”) or randomly in the block (typeNA=“ponc”)

```
res_create=createNA(blocks=A,pNA=0.2,seed=41)
names(res_create)
#> [1] "dat"          "naPos"        "subjectKept"
A_missing=res_create$dat
lapply(A_missing,head)
#> [[1]]
#>           gini farm rent
#> Argentina 86.3 98.2 3.52
#> Australia 92.9 99.6 3.27
#> Austria   74.0 97.4 2.46
#> Belgium   58.7 85.8 4.15
#> Bolivia   93.8 97.7 3.04
#> Brasil      NA   NA   NA
#>
#> [[2]]
#>           gnpr labo
#> Argentina 5.92 3.22
#> Australia 7.10 2.64
#> Austria     NA   NA
#> Belgium   6.92 2.30
#> Bolivia   4.19 4.28
#> Brasil    5.57 4.11
#>
#> [[3]]
#>           inst ecks death
#> Argentina 0.07 4.06  5.38
#> Australia   NA   NA    NA
#> Austria     NA   NA    NA
#> Belgium   0.45 2.20  0.69
#> Bolivia     NA   NA    NA
#> Brasil      NA   NA    NA
```

createNA return a list containing the data with missing values, the position of missing values and the complete subjects kept in the analysis.

5.3.2 Vizualizing a dataset with missing values, and getting the pattern of missing values

To visualize a dataset with missing values, the function get_patternNA can be used. It also return the percent of missing values per variables and per block.

```
patternNA=get_patternNA(blocks=A_missing)
#> Warning: Warnings in check_blocks(A):
#>  blocks of the list had no names. The blocks were named block1,... blockJ in the order
names(patternNA)
```

```
#> [1] "pctNA"                              "pctNAbyBlock"
#> [3] "completeSubjectByBlock"             "completeSubjects"
#> [5] "numberOfMissingBlocksPerSubject" "blocks"
```

```
A_missing2=createNA(blocks=A,typeNA="byvar",
                     pNA=list(0.1*1:3,rep(0,2),rep(0.1,3)))$dat
d=get_patternNA(A_missing2)
#> Warning: Warnings in check_blocks(A):
#>  blocks of the list had no names. The blocks were named block1,... blockJ in the order
```

5.4 Using complete data only

The complete data approach is the less risky, but can conduct to a large loss of data, potentially leading to less significant results. It is the one used by default in most software solutions.

```
A_inter=intersection(A_missing)
lapply(A_inter,dim)
#> [[1]]
#> [1] 25  3
#>
#> [[2]]
#> [1] 25  2
#>
#> [[3]]
#> [1] 25  3
```

5.5 Using NIPALS algorithm

5.5.1 Principle

This approach is based on NIPALS algorithm and PLS-PM approach, and consists in re-writing the algorithm by ignoring the missing values. Thus, calculating a matricial product without taking into account missing values is equivalent to replace the missing value by zero, regression on missing value removes the observation, and correlations are made pairwise. Thus, contrarily to the complete data approach, all existing data are taken into account during the analysis.

The resulting RGCCA can be entering method="nipals" in usual rgcca

5.5.2 R Code

```
rgcca_nipals=rgcca(blocks =A_missing,superblock=F,verbose=FALSE,method="nipals")
```

5.6 Using k-Nearest Neighbors

We chose here the hotdeck procedure based on the k nearest neighbors. Note that each block X_j is previously centered and scaled. The neighborood notion is based on the euclidean distance between individuals weighted by the number of variables (in order to give the same weight to each block). Thus, the distance between two individuals i_1 and i_2 can be formalized as:

$$d(i_1, i_2) = \sum_{j=1}^{J} \left(\frac{1}{card(P_j)} \sum_{p \in P_j} (\mathbf{X}_j(i_1, p) - \mathbf{X}_j(i_2, p))^2 \right)$$

P_j being the set of variables where i_1 and i_2 are not missing. For each missing value, the distance between the related individual and all the complete individuals are calculated.

```
rgcca_knnA=rgcca(blocks=A_missing,superblock=F,method="knnA")
rgcca_knn3=rgcca(blocks=A_missing,superblock=F,method="knn3")
```

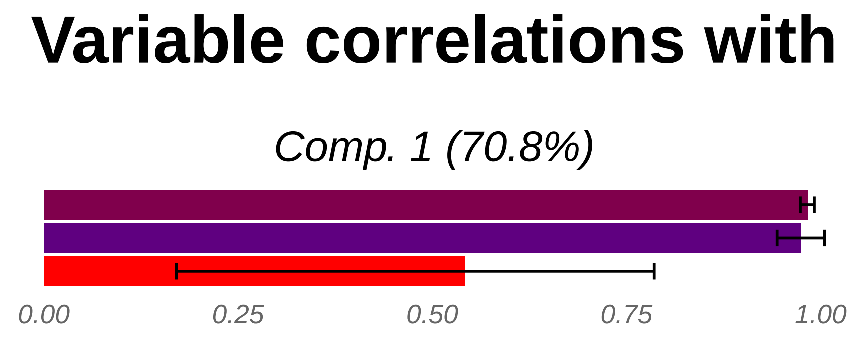
5.6.1 Multiple imputation

Here, we chose the 5-NN. RGCCA is run M (=5) times on the M resulted imputed dataset.

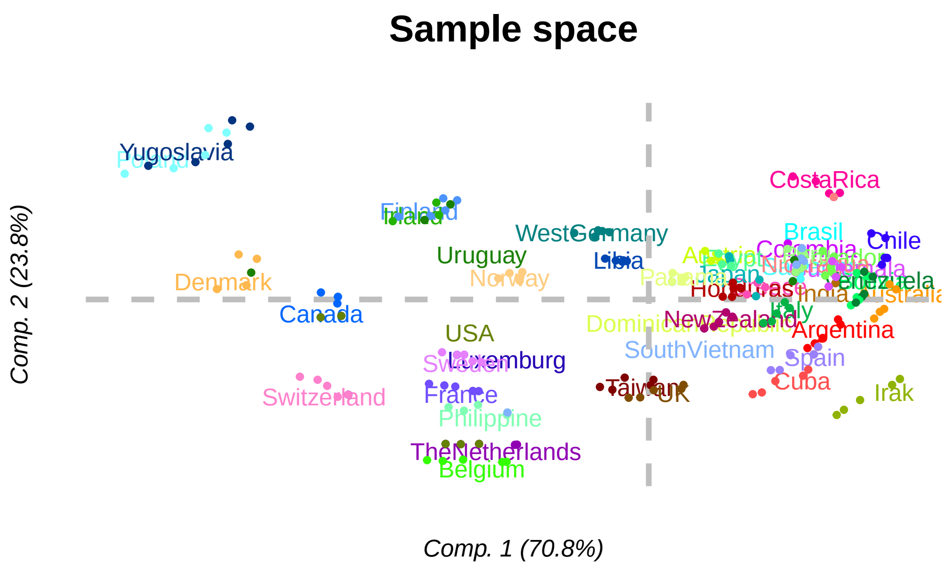
In order to represent RGCCA, a "reference" representation space must be chosen in order to project all the M results. For example, it can be the space of the RGCCA imputed by the 1-nearest neighbor. We choose here to take the space of the RGCCA of imputation by the average of all individuals weighted by the inverse of their distance with the individual with missing data. Regarding the pool of sample plot, procrustean rotation is run for superimpose each of the M RGCCA individual plot on the reference representation space. Thus, each individual correspond to M points. They can be linked by segments or summarized in an ellipse in order to represent the variability due to the estimation of missing data. Regarding the pool of variable plot, only the representation variable space is shown in order to not overload the graph.

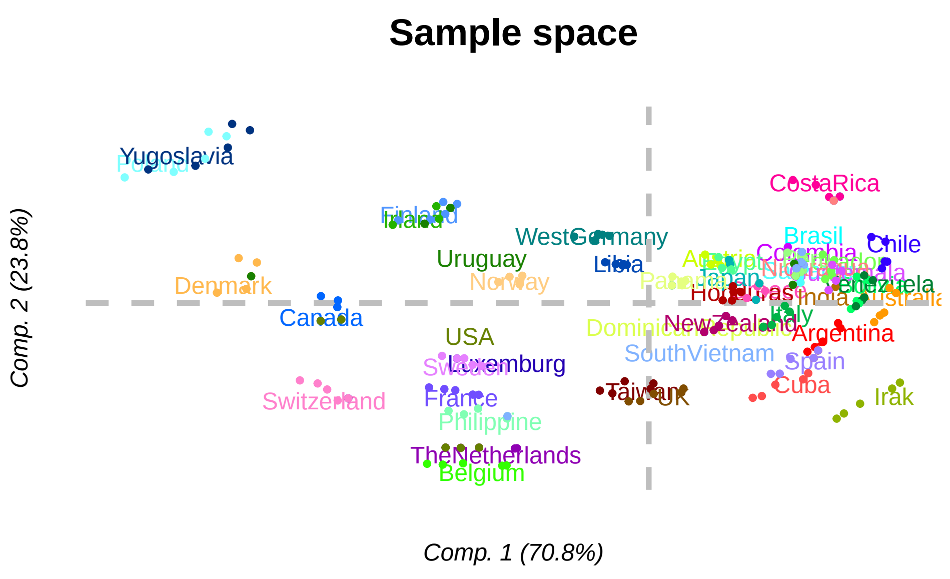
Regarding the relevant variables, the standard deviation of the correlations between their scores and first and second axis were added.

```
mi_rgcca=MIRGCCA(blocks=A_missing,k=3,ni=5,tau=rep(1,3))
 plot(mi_rgcca,type="cor")
```



```
 plot(mi_rgcca,type="ind")
```





5.7 Which method to use?

Among all these methods for dealing with missing data, we suspect that the best is dataset-depending, and the ideal is to compare the method results to choose the more adapted to the dataset.

5.7.1 Simulations on complete dataset according to this pattern

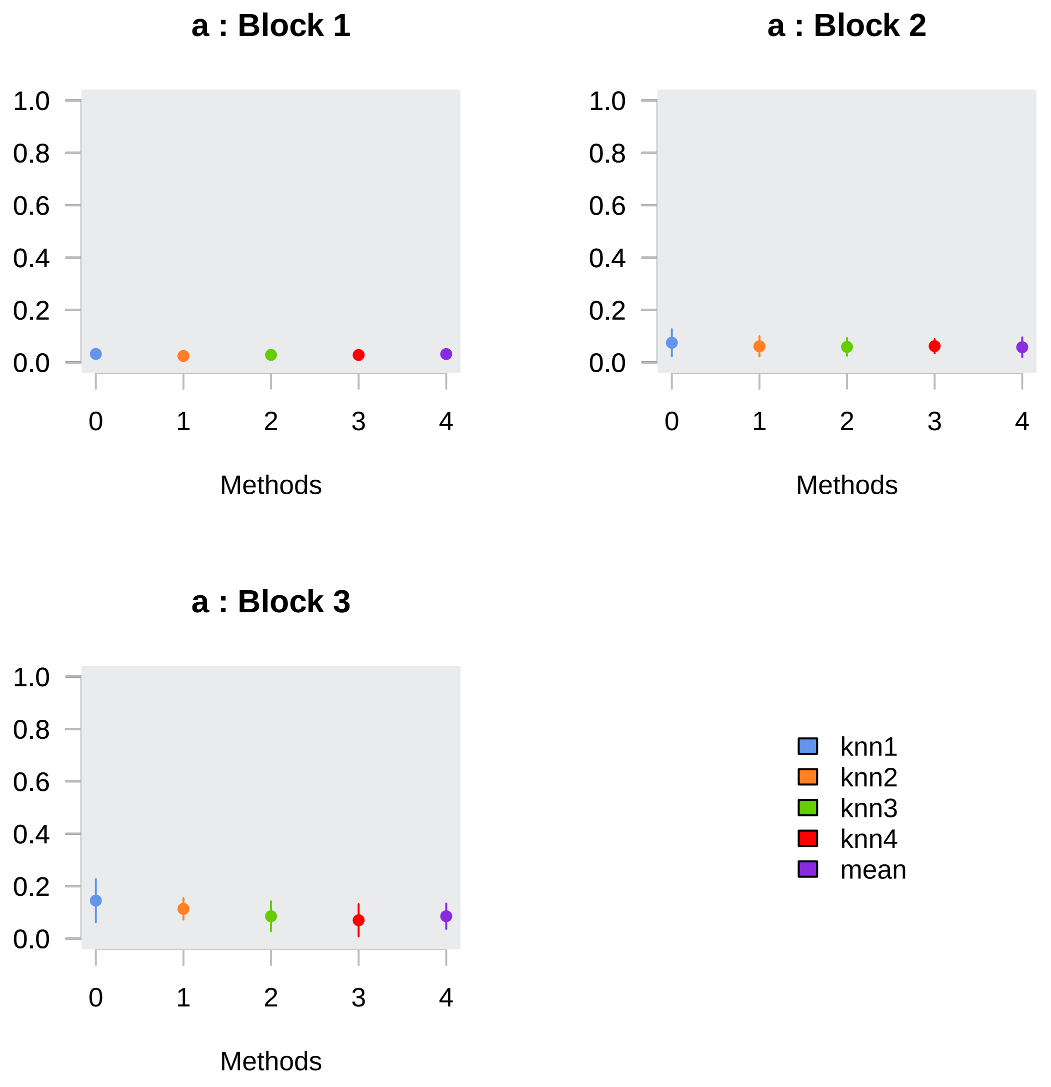
To select the more appropriate method for a given dataset, simulations are made on the complete part of the dataset. This is based on the assumptions that the missing values were randomly distributed, and that the complete part is a representative part of the dataset.

whichNAmethod simulates nDatasets (20 by default) with missing values according to a specific pattern of missing values (patternNA) and typeNA (“block”,“byvar” or “ponc”). The plot function allows to select one of the following indicators

The idea is to compare the results of the different “missing methods” to the original results (known, as the initial dataset is complete)

- output="a": distance between the weights of missing and original RGCCA
- output="rv": rv coefficient for all individuals on the 1-2 map between missing and original RGCCA (complete method can not be run with this option)
- output="rvComplete": rv coefficient for complete individuals only on the 1-2 map between missing and reference case
- output="bm": percentage of similar top ten of variables

```
res=whichNAmethod(A_missing,listMethods=c("knn1","knn2","knn3","knn4","mean"),
                  patternNA=patternNA$pctNAbyBlock,nDatasets = 5)
plot(res,output="a",ylim=c(0,1))
```

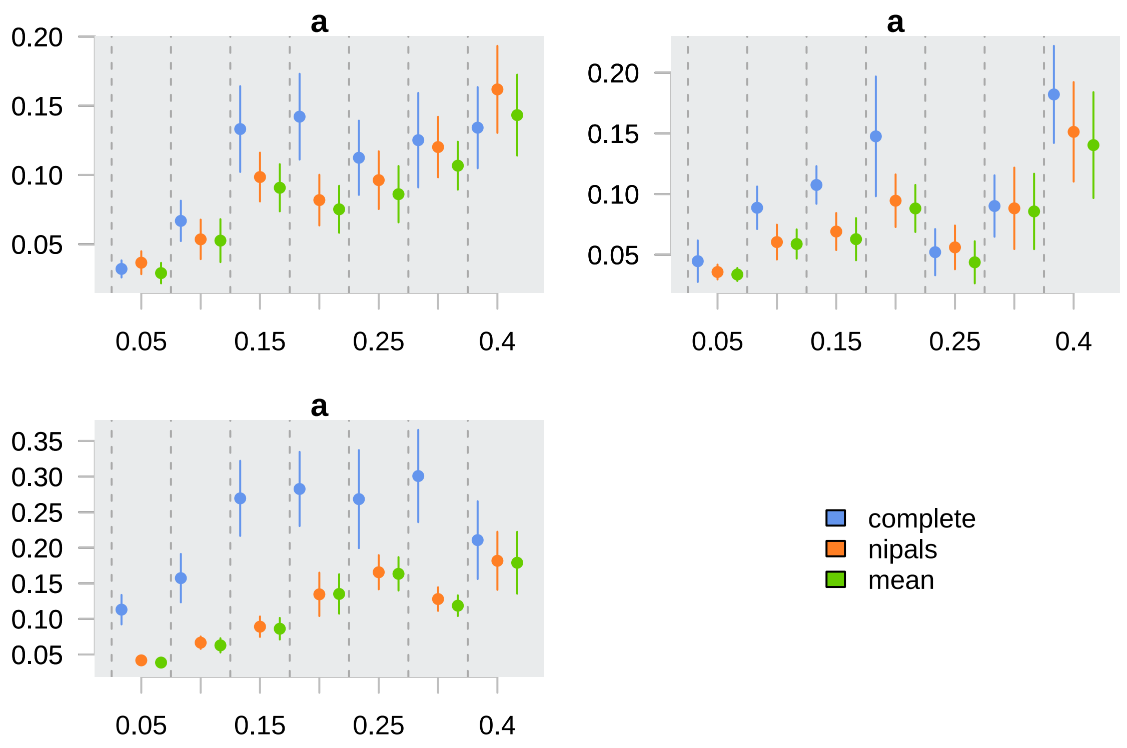


In this case, the best method for the similarities between weights is

5.8 Behavior with increasing missing data proportion

```
names(A)=c("agri","ind","polit")
listResults=naEvolution(blocks=A,listMethods=c("complete","nipals","mean"),
        prctNA=c(0.05,0.1,0.15,0.2,0.25,0.3,0.4),typeNA="ponc",ncomp=rep(1,3),
       ,nDatasets = 10)
```

```
plot(listResults,output="a",bars = "stderr")
```



6 Conclusion

This package gathers fifty years of multiblock component methods and offers a unified implementation stategy for these methods. Work is in progress to include within the RGCCA package:

- a function that fully implements the cross validation procedure.
- a bootstrap resampling procedure for assessing the realiability of the parameters estimates of RGCCA and the stability of the selected variables for SGCCA.
- Dedicated functions for graphical representations of the ouptut of RGCCA (factor plot, correlation circle).
- multiblock data faces two types of missing data structure: (i) if an observation i has missing values on a whole block j and (ii) if an observation i has some missing values on a block j (but not all). For these two situations, it is possible to exploit the algorithmic solution proposed for PLS path modeling to deal with missing data (see (Tenenhaus et al. 2005), page 171). Work is in progress to implement this missing data solution within the RGCCA package.
- At last, RGCCA for multigroup data (A. Tenenhaus et al. 2014) and for multiway data (Arthur Tenenhaus, Le Brusquet, and Lechuga 2015) has been proposed but not yet implemented in the RGGCA package. These two methods remain to be integrated into the package.

References

Borga, M., T. Landelius, and H. Knutsson. 1997. "A Unified Approach to PCA, PLS, MLR and CCA."

Bougeard, S., M. Hanafi, and E.M. Qannari. 2008. "Continuum redundancy-PLS regression: a simple continuum approach." *Computational Statistics and Data Analysis* 52: 3686–96.

Burnham, A.J., R. Viveros, and J.F. MacGregor. 1996. “Frameworks for latent variable multivariate regression.” *Journal of Chemometrics* 10: 31–45.

Carroll, J.D. 1968. “A generalization of canonical correlation analysis to three or more sets of variables.” In *Proceeding 76th Conv. Am. Psych. Assoc.*, 227–28.

Carroll, JD. 1968. “Equations and Tables for a Generalization of Canonical Correlation Analysis to Three or More Sets of Variables.” *Unpublished Companion Paper to Carroll, JD*.

Chessel, D., and M. Hanafi. 1996. “Analyse de la co-inertie de K nuages de points.” *Revue de Statistique Appliquée* 44: 35–60.

Gifi, A. 1990. *Nonlinear multivariate analysis*. John Wiley & Sons, Chichester, UK.

Hanafi, M. 2007. “PLS Path modelling: computation of latent variables with the estimation mode B.” *Computational Statistics* 22: 275–92.

Hanafi, M., and H.A.L. Kiers. 2006. “Analysis of K sets of data, with differential emphasis on agreement between and within sets.” *Computational Statistics and Data Analysis* 51: 1491–1508.

Horst, P. 1961. “Relations among m sets of variables.” *Psychometrika* 26: 126–49.

Hotelling, H. 1936. “Relation Between Two Sets of Variates.” *Biometrika* 28: 321–77.

Kettenring, J.R. 1971. “Canonical analysis of several sets of variables.” *Biometrika* 58: 433–51.

Kramer, N. 2007. “Analysis of high-dimensional data with partial least squares and boosting.” In *Doctoral dissertation, Technischen Universitat Berlin*.

Puget, S., C. Philippe, D. A Bax, B. Job, P. Varlet, M.-P. Junier, F. Andreiuolo, et al. 2012. “Mesenchymal transition and PDGFRA amplification/mutation are key distinct oncogenic events in pediatric diffuse intrinsic pontine gliomas.” *PloS One* 7 (2): e30313.

Qannari, E.M., and M. Hanafi. 2005. “A simple continuum regression approach.” *Journal of Chemometrics* 19: 387–92.

Russett, B.M. 1964. “Inequality and Instability: The Relation of Land Tenure to Politics.” *World Politics* 16:3: 442–54.

Schäfer, J., and K. Strimmer. 2005. “A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics.” *Statistical Applications in Genetics and Molecular Biology* 4 (1): Article 32.

Shawe-Taylor, J., and N. Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA.

Takane, Y., and H. Hwang. 2007. “Regularized linear and kernel redundancy analysis.” *Computational Statistics and Data Analysis* 52: 394–405.

Tenenhaus, A., C. Philippe, V. Guillemot, K.-A. Lê Cao, J. Grill, and V. Frouin. 2014. “Variable Selection for Generalized Canonical Correlation Analysis.” *Biostatistics* 15(3): 569–83.

Tenenhaus, Arthur, Laurent Le Brusquet, and Gisela Lechuga. 2015. “Multiway Regularized Generalized Canonical Correlation Analysis.” In *47ème Journées de Statistique, Lille, France*.

Tenenhaus, Arthur, Cathy Philippe, and Vincent Frouin. 2015. “Kernel Generalized Canonical Correlation Analysis.” *Computational Statistics & Data Analysis* 90. Elsevier: 114–31.

Tenenhaus, A., and M. Tenenhaus. 2011. “Regularized Generalized Canonical Correlation Analysis.” *Psychometrika* 76: 257–84.

———. 2014. “Regularized Generalized Canonical Correlation Analysis for multiblock or multigroup data analysis.” *European Journal of Operational Research* 238: 391–403.

Tenenhaus, M., V. Esposito Vinzi, Y.-M. Chatelin, and C. Lauro. 2005. “PLS path modeling.” *Computational Statistics and Data Analysis* 48: 159–205.

Tenenhaus, M., A. Tenenhaus, and PJF. Groenen. 2017. "Regularized generalized canonical correlation analysis: A framework for sequential multiblock component methods." *Psychometrika*, in press.

Tucker, L.R. 1958. "An inter-battery method of factor analysis." *Psychometrika* 23: 111–36.

Van de Geer, J.P. 1984. "Linear relations among k sets of variables." *Psychometrika* 49: 70–94.

Van den Wollenberg, A.L. 1977. "Redudancy analysis: an alternative for canonical correlation analysis." *Psychometrika* 42: 207–19.

Vinod, H.D. 1976. "Canonical ridge and econometrics of joint production." *Journal of Econometrics* 4: 147–66.

Westerhuis, J.A, T. Kourti, and J.F. MacGregor. 1998. "Analysis of multiblock and hierarchical PCA and PLS models." *Journal of Chemometrics* 12: 301–21.

Wold, H. 1982. "Soft Modeling: The Basic Design and Some Extensions." In *in Systems under indirect observation, Part 2, K.G. Jöreskog and H. Wold (Eds), North-Holland, Amsterdam*, 1–54.

Wold, S. and Kettaneh, N. and Tjessem, K.. 1996. "Hierarchical multiblock PLS and PC models for easier model interpretation and as an alternative to variable selection." *Journal of Chemometrics* 10: 463–82.