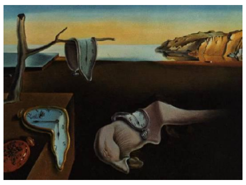


Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. doi: 10.18637/jss.v000.i00

# The RGCCA package for Regularized/Sparse Generalized Canonical Correlation Analysis

**FirstName LastName**
University/Company

**Second Author**
Affiliation

**Third Author**
Universitat Autònoma
de Barcelona

**Abstract**

The RGCCA package aims to propose a unified and flexible framework for multiblock component methods. The RGCCA package aims to propose a unified and flexible framework for multiblock component methods. The RGCCA package aims to propose a unified and flexible framework for multiblock component methods. The RGCCA package aims to propose a unified and flexible framework for multiblock component methods. The RGCCA package aims to propose a unified and flexible framework for multiblock component methods. The RGCCA package aims to propose a unified and flexible framework for multiblock component methods. The RGCCA package aims to propose a unified and flexible framework for multiblock component methods.

*Keywords*: Multiblock component methods, RGCCA, data integration.

## 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 scientific literature, this paradigm can be stated under several names as “learning from multimodal data”, “data integration”, “multiview data”, “multisource data”, “data fusion” or “multiblock data analysis”. Appropriate statistical methods and dedicated softwares able to cope with these multiple highly multivariate datasets constitute key issues for effective analysis leading to valuable knowledge. Regularized Generalized Canonical Correlation Analysis is a unified and flexible framework for multiblock component methods. The RGCCA package implements this framework and its usefulness is illustrated in this paper.

## 2. Optimization background

The goal of this section is to present the optimization framework under which most of the algorithms proposed in the RGCCA framework were designed. The RGCCA framework gathers several algorithm that has already been presented in (Tenenhaus and Tenenhaus 2011, Tenenhaus and Tenenhaus (2014), Tenenhaus, Philippe, and Frouin (2015), Tenenhaus, Tenenhaus, and Groenen (2017)). It is recalled here for a broader class of constraints.

The RGCCA framework relies on a master algorithm for maximizing a continuously differentiable multi-convex function $f(\mathbf{a}_1, \ldots, \mathbf{a}_J) : \mathbb{R}^{p_1} \times \ldots \times \mathbb{R}^{p_J} \rightarrow \mathbb{R}$ (i.e. for each $j$, $f$ is a convex function of $\mathbf{a}_j$ while all the other $\mathbf{a}_k$ are fixed) under the constraint that each $\mathbf{a}_j$ belongs to a compact set $\Omega_j \subset \mathbb{R}^{p_j}$. This general optimization problem can be formulated as follows:

$$\max_{\mathbf{a}_1, \ldots, \mathbf{a}_J} f(\mathbf{a}_1, \ldots, \mathbf{a}_J) \text{ s.t. } \mathbf{a}_j \in \Omega_j, \ j = 1, \ldots, J. \tag{1}$$

For such function defined over a set of parameter vectors $(\mathbf{a}_1, \ldots, \mathbf{a}_J)$, we make no difference between the notations $f(\mathbf{a}_1, \ldots, \mathbf{a}_J)$ and $f(\mathbf{a})$, where $\mathbf{a}$ is the column vector $\mathbf{a} = \left(\mathbf{a}_1^\top, \ldots, \mathbf{a}_J^\top\right)^\top$ of size $p = \sum_{j=1}^J p_j$. Moreover, the vertical concatenation of column vectors is denoted $\mathbf{a} = (\mathbf{a}_1; \ldots; \mathbf{a}_J)$ for the sake of simplification of notation.

### 2.1. Algorithm

A simple, monotonically and globally convergent algorithm is presented for solving optimization problem (1). The maximization of the function $f$ defined over different parameter vectors $(\mathbf{a}_1, \ldots, \mathbf{a}_J)$, is approached by updating each of the parameter vectors in turn, keeping the others fixed. This update rule was recommended in (De Leeuw 1994) and is called Block Relaxation or cyclic Block Coordinate Ascent (BCA).

Let $\nabla_j f(\mathbf{a})$ be the partial gradient of $f(\mathbf{a})$ with respect to $\mathbf{a}_j$. We assume $\nabla_j f(\mathbf{a}) \neq \mathbf{0}$ in this manuscript. This assumption is not too binding as $\nabla_j f(\mathbf{a}) = \mathbf{0}$ characterizes the global minimum of $f(\mathbf{a}_1, \ldots, \mathbf{a}_J)$ with respect to $\mathbf{a}_j$ when the other vectors $\mathbf{a}_1, \ldots, \mathbf{a}_{j-1}, \mathbf{a}_{j+1}, \ldots, \mathbf{a}_J$ are fixed.

We want to find an update $\hat{\mathbf{a}}_j \in \Omega_j$ such that $f(\mathbf{a}) \leq f(\mathbf{a}_1, ..., \mathbf{a}_{j-1}, \hat{\mathbf{a}}_j, \mathbf{a}_{j+1}, ..., \mathbf{a}_J)$. As $f$ is a continuously differentiable multi-convex function and considering that a convex function lies above its linear approximation at $\mathbf{a}_j$ for any $\tilde{\mathbf{a}}_j \in \Omega_j$, the following inequality holds:

$$f(\mathbf{a}_1, ..., \mathbf{a}_{j-1}, \tilde{\mathbf{a}}_j, \mathbf{a}_{j+1}, \ldots, \mathbf{a}_J) \geq f(\mathbf{a}) + \nabla_j f(\mathbf{a})^\top (\tilde{\mathbf{a}}_j - \mathbf{a}_j). \tag{2}$$

On the right-hand side of the inequality (2), only the term $\nabla_j f(\mathbf{a})^\top \tilde{\mathbf{a}}_j$ is relevant to $\tilde{\mathbf{a}}_j$ and the solution that maximizes the minorizing function over $\tilde{\mathbf{a}}_j \in \Omega_j$ is obtained by considering the following optimization problem:

$$\hat{\mathbf{a}}_j = \underset{\tilde{\mathbf{a}}_j \in \Omega_j}{\operatorname{argmax}} \nabla_j f(\mathbf{a})^\top \tilde{\mathbf{a}}_j := r_j(\mathbf{a}). \tag{3}$$

The entire algorithm is subsumed in Algorithm 1.

**Algorithm 1** Algorithm for the maximization of a continuously differentiable multi-convex function

```
1: Result: $\mathbf{a}_1^s, \dots, \mathbf{a}_J^s$ (approximate solution of (1))
2: Initialization: choose random vector $\mathbf{a}_j^0 \in \Omega_j, j = 1, \dots, J, \varepsilon$;
3: $s = 0$ ;
4: repeat
5:   for $j = 1$ to $J$ do
6:       $\mathbf{a}_j^{s+1} = r_j\left(\mathbf{a}_1^{s+1}, \dots, \mathbf{a}_{j-1}^{s+1}, \mathbf{a}_j^s, \dots, \mathbf{a}_J^s\right).$    (4)
7:   end for
8:   $s = s + 1$ ;
9: until $f(\mathbf{a}_1^{s+1}, \dots, \mathbf{a}_J^{s+1}) - f(\mathbf{a}_1^s, \dots, \mathbf{a}_J^s) < \varepsilon$
```

We need to introduce some extra notations to present the convergence properties of Algorithm 1: $\Omega = \Omega_1 \times \ldots \times \Omega_J$, $\mathbf{a} = (\mathbf{a}_1; \ldots; \mathbf{a}_J) \in \Omega$, $c_j : \Omega \mapsto \Omega$ is an operator defined as $c_j(\mathbf{a}) = (\mathbf{a}_1; \ldots; \mathbf{a}_{j-1}; r_j(\mathbf{a}); \mathbf{a}_{j+1}; \ldots; \mathbf{a}_J)$ with $r_j(\mathbf{a})$ introduced in equation (3) and $c : \Omega \mapsto \Omega$ is defined as $c = c_J \circ c_{J-1} \circ \ldots \circ c_1$, where $\circ$ stands for the composition operator. Using the operator $c$, the «for loop» inside Algorithm 1 can be replaced by the following recurrence relation: $\mathbf{a}^{s+1} = c(\mathbf{a}^s)$. The convergence properties of Algorithm 1 are summarized in the following proposition:

**Proposition 2.1.** *Let* $\{\mathbf{a}^s\}_{s=0}^{\infty}$ *be any sequence generated by the recurrence relation* $\mathbf{a}^{s+1} = c(\mathbf{a}^s)$ *with* $\mathbf{a}^0 \in \Omega$. *Then, the following properties hold:*

*(a) The sequence* $\{f(\mathbf{a}^s)\}$ *is monotonically increasing and therefore convergent as* $f$ *is bounded on* $\Omega$. *This result implies the monotonic convergence of Algorithm 1.*

*(b) If the infinite sequence* $\{f(\mathbf{a}^s)\}$ *involves a finite number of distinct terms, then the last distinct point satisfies* $c(\mathbf{a}^s) = \mathbf{a}^s$ *and therefore is a stationary point of problem 1*

*(c)* $\lim_{s\to\infty} f(\mathbf{a}^s) = f(\mathbf{a})$, *where* $\mathbf{a}$ *is a fixed point of* $c$.

*(d) The limit of any convergent subsequence of* $\{\mathbf{a}^s\}$ *is a fixed point of* $c$.

*(e) The sequence* $\{\mathbf{a}^s\}$ *is asymptotically regular:* $\lim_{s\to\infty} \sum_{j=1}^J \|\mathbf{a}_j^{s+1} - \mathbf{a}_j^s\| = 0$. *This result implies that if the threshold* $\varepsilon$ *for the stopping criterion in Algorithm 1 is made sufficiently small, the output of Algorithm 1 will be as close as wanted to a stationary point of 1.*

*(f) If the equation* $\mathbf{a} = c(\mathbf{a})$ *has a finite number of solutions, then the sequence* $\{\mathbf{a}^s\}$ *converges to one of them.*

Proposition 2.1 gathers all the convergence properties of Algorithm 1. The three first points of Proposition 2.1 concern the behavior of the sequence values $\{f(\mathbf{a}^s)\}$ of the objective function, whereas the three last points are about the behaviour of the sequence $\{\mathbf{a}^s\}$. The full proof of these properties is given Tenenhaus *et al.* (2017).

## 3. The RGCCA framework

The theoretical foundations of the Regularized Generalized Canonical Correlation Analysis (RGCCA) framework - that were previously published (Tenenhaus and Tenenhaus 2011, ; Tenenhaus and Tenenhaus 2014, ; Tenenhaus *et al.* 2015, ; Tenenhaus *et al.* 2017) - are briefly summarized.

### 3.1. Optimization problem

A random column vector $\boldsymbol{x}$ of $p$ variables is assumed to exist with finite moments of at least order two. The random vector $\boldsymbol{x}$ has zero mean and a covariance matrix $\boldsymbol{\Sigma}$. The vector $\boldsymbol{x}$ is composed of $J$ subvectors $\boldsymbol{x}_j = (x_{j1}, \dots, x_{jp_j})^\top$. The covariance matrix matrix $\boldsymbol{\Sigma}$ is composed of $J^2$ submatrices $\boldsymbol{\Sigma}_{jk} = \mathbb{E}\left[\boldsymbol{x}_j \boldsymbol{x}_k^\top\right]$. Let $\mathbf{a}_j = (a_{j1}, \dots, a_{jp_j})^\top$ be a non-random $p_j$-dimensional column vector. A composite variable $\eta_j$ is defined as the linear combination of the elements of $\boldsymbol{x}_j$: $\eta_j = \mathbf{a}_j^\top \boldsymbol{x}_j$. Therefore the covariance between two composite variables is $\mathbf{a}_j^\top \boldsymbol{\Sigma}_{jk} \mathbf{a}_k$. The RGCCA framework aims at extracting the information which is shared by the $J$ random composite variables taking into account an undirected graph of connections between them. The RGCCA framework is defined by the optimization problem (5) and consists in maximizing the sum of convex functions of the covariances between "connected" composites $\eta_j$ and $\eta_k$ subject to specific constraints on the weights $\mathbf{a}_j$s.

$$\max_{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_J} f(\mathbf{a}_1, \dots \mathbf{a}_J) = \sum_{j,k=1}^{J} c_{jk} \, \mathrm{g}\left(\mathbf{a}_j^\top \boldsymbol{\Sigma}_{jk} \mathbf{a}_k\right) \text{ s.t. } \mathbf{a}_j \in \Omega_j, j = 1, \dots, J, \tag{5}$$

where

- each $\Omega_j$ is a compact set.
- the function $g$ is any continuously differentiable convex function. Typical choices of $g$ are the identity (horst scheme, leading to maximizing the sum of covariances between block components), the absolute value[1] (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.
- The design matrix $\mathbf{C} = \{c_{jk}\}$ is a symmetric $J \times J$ matrix of non-negative elements describing the network of connections between blocks that the user wants to take into account. Usually $c_{jk} = 1$ to two connected blocks and 0 otherwise.

When the diagonal of $\mathbf{C}$ is null, the convexity and continuous differentiability of the function g imply that the objective function $f$ itself is multi-convex continuously differentiable. When at least one element of the diagonal of $\mathbf{C}$ is different from 0, additional conditions have to be imposed on g to keep the desired property on $f$. For example, when g is twice differentiable, a sufficient condition is that $\forall x \in \mathbb{R}_+, \;\; g'(x) \geq 0$. This condition guarantees that the second derivative of $g\left(\mathbf{a}_j^\top \boldsymbol{\Sigma}_{jj} \mathbf{a}_j\right)$ is positive definite:

$$\frac{\partial^2 g\left(\mathbf{a}_j^\top \boldsymbol{\Sigma}_{jj} \mathbf{a}_j\right)}{\partial \mathbf{a}_j \partial \mathbf{a}_j^\top} = 2\left[g'\left(a_j^\top \boldsymbol{\Sigma}_{jj} \mathbf{a}_j\right) \boldsymbol{\Sigma}_{jj} + 2g''\left(\mathbf{a}_j^\top \boldsymbol{\Sigma}_{jj} \mathbf{a}_j\right) \boldsymbol{\Sigma}_{jj} \mathbf{a}_j \mathbf{a}_j^\top \boldsymbol{\Sigma}_{jj}\right]. \tag{6}$$

All functions g considered in this paper satisfy this condition. Consequently, the optimization problem (5) falls under the umbrella of the general optimization framework presented in Section 1.

[1]The scheme $g(x) = |x|$ can be included in this class of functions because the case $x = 0$ never appears in practical applications.

### 3.2. Regularized Generalized Canonical Correlation Analysis (RGCCA)

Several instantiations of the RGCCA framework were proposed in (Tenenhaus and Tenenhaus 2011, Tenenhaus *et al.* (2015), Tenenhaus *et al.* (2017)) with $\Omega_j = \left\{\mathbf{a}_j \in \mathbb{R}^{p_j}; \mathbf{a}_j^\top \mathbf{M}_j \mathbf{a}_j = 1\right\}$ where $\mathbf{M}_j$ is positive definite matrix of order $p_j$. The optimization problem (5) boils down to:

$$\underset{\mathbf{a}_1,\ldots\mathbf{a}_J}{\text{maximize}}\ f(\mathbf{a}_1,\ldots\mathbf{a}_J) = \sum_{j,k=1}^{J} c_{jk} \mathrm{g}\left(\mathbf{a}_j^\top \boldsymbol{\Sigma}_{jk}\mathbf{a}_k\right) \text{ s.t. } \mathbf{a}_j^\top \mathbf{M}_j \mathbf{a}_j = 1, j = 1,\ldots,J. \tag{7}$$

Algorithm 1 can be used to solve the optimization problem (7). This is done by updating each of the parameter vectors in turn, keeping the others fixed. Hence, we want to find an update $\hat{\mathbf{a}}_j \in \Omega_j = \left\{\mathbf{a}_j \in \mathbb{R}^{p_j}; \mathbf{a}_j^\top \mathbf{M}_j \mathbf{a}_j = 1\right\}$ such that $f(\mathbf{a}) \leq f(\mathbf{a}_1,\ldots,\mathbf{a}_{j-1},\hat{\mathbf{a}}_j,\mathbf{a}_{j+1},\ldots,\mathbf{a}_J)$. the RGCCA update is obtained by considering the following optimization problem:

$$\hat{\mathbf{a}}_j = \underset{\tilde{\mathbf{a}}_j \in \Omega_j}{\operatorname{argmax}}\ \nabla_j f(\mathbf{a})^\top \tilde{\mathbf{a}}_j = \frac{\mathbf{M}_j^{-1}\nabla_j f(\mathbf{a})}{\|\mathbf{M}_j^{-1/2}\nabla_j f(\mathbf{a})\|} := r_j(\mathbf{a}), j = 1,\ldots,J. \tag{8}$$

where the partial gradient $\nabla_j f(\mathbf{a})$ of $f(\mathbf{a})$ with respect to $\mathbf{a}_j$ is a $p_j$-dimensional column vector given by:

$$\nabla_j f(\mathbf{a}) = 2\sum_{k=1}^{J} c_{jk} g'\left(\mathbf{a}_j^\top \boldsymbol{\Sigma}_{jk}\mathbf{a}_k\right)\boldsymbol{\Sigma}_{jk}\mathbf{a}_k \tag{9}$$

A sample-based optimization problem related to (7) is derived by considering a column partition $\mathbf{X} = [\mathbf{X}_1,\ldots,\mathbf{X}_j,\ldots,\mathbf{X}_J]$. In this case, each $n \times p_j$ data matrix $\mathbf{X}_j$ 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 most recent formulation of RGCCA (Tenenhaus *et al.* 2017) subsumes fifty years of multiblock component methods. It provides improvements to the initial version of RGCCA (Tenenhaus and Tenenhaus 2011) and is defined as the following optimization problem:

$$\underset{\mathbf{a}_1,\ldots,\mathbf{a}_J}{\text{maximize}} \sum_{j,k=1}^{J} c_{jk} g\left(\mathbf{a}_j^\top \widehat{\boldsymbol{\Sigma}}_{jk}\mathbf{a}_k\right) \text{ s.t. } \mathbf{a}_j^\top \widehat{\boldsymbol{\Sigma}}_{jj}\mathbf{a}_j = 1, j = 1,\ldots,J \tag{10}$$

where $\widehat{\boldsymbol{\Sigma}}_{jk} = n^{-1}\mathbf{X}_j^\top \mathbf{X}_k$ is an estimate of the inter-block covariance matrix $\boldsymbol{\Sigma}_{jk} = \mathbb{E}[\boldsymbol{x}_j \boldsymbol{x}_k^\top]$ and $\widehat{\boldsymbol{\Sigma}}_{jj}$ is an estimate of the intra-block covariance matrix $\boldsymbol{\Sigma}_{jj} = \mathbb{E}[\boldsymbol{x}_j \boldsymbol{x}_j^\top]$. In cases involving multi-collinearity within blocks or in high dimensional settings, one way of obtaining an estimate for the true covariance matrix $\boldsymbol{\Sigma}_{jj}$ is to consider the class of linear convex combinations of the identity matrix $\mathbf{I}$ and the sample covariance matrix $\mathbf{S}_{jj} = n^{-1}\mathbf{X}_j^\top \mathbf{X}_j$. We then consider a version of optimization problem (10) with $\widehat{\boldsymbol{\Sigma}}_{jj} = \tau_j \mathbf{I} + (1-\tau_j)\mathbf{S}_{jj}$ with $\tau_j \in [0,1]$ (shrinkage estimator of $\boldsymbol{\Sigma}_{jj}$). This plug-in approach leads to the RGCCA optimization problem (Tenenhaus and Tenenhaus 2011). It is worth pointing out that for each block $j$, an appropriate shrinkage parameter $\tau_j$ can be obtained using various analytical formulae (see for instance (Ledoit and Wolf 2004, Schäfer and Strimmer (2005), Chen, Wiesel, and Hero (2011))). As $\mathbf{M}_j$ must be positive definite, $\tau_j = 0$ can only be selected for a full rank data matrix $\mathbf{X}_j$.

An equivalent formulation of optimization problem (10) is given hereafter and enables a better characterization of the objective of RGCCA.

$$\underset{\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_J}{\text{maximize}} \sum_{j,k=1}^{J} c_{jk} g(\text{cov}(\mathbf{X}_j\mathbf{a}_j, \mathbf{X}_k\mathbf{a}_k)) \text{ s.t. } (1-\tau_j)\text{var}(\mathbf{X}_j\mathbf{a}_j)+\tau_j\|\mathbf{a}_j\|^2 = 1, j = 1,\ldots,J \quad (11)$$

Hence, the objective of RGCCA is to find block components $\mathbf{y}_j = \mathbf{X}_j\mathbf{a}_j, j = 1,\ldots,J$ (where $\mathbf{a}_j$ is a block weight vector of size $p_j$) summarizing the relevant information between and within the blocks. The $\tau_j$s are called shrinkage parameters ranging from 0 to 1 and interpolate smoothly between maximizing the covariance and maximizing the correlation. Setting the $\tau_j$ to 0 will force the block components to unit variance ($\text{var}(\mathbf{X}_j\mathbf{a}_j) = 1$), in which case the covariance criterion boils down to the correlation. Setting $\tau_j$ 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 $(1-\tau_j)\text{var}(\mathbf{X}_j\mathbf{a}_j)+\tau_j\|\mathbf{a}_j\|^2 = 1$. We can discuss the choice of the shrinkage parameters by providing interpretations on the properties of the resulting block components:

- $\tau_j = 1$ 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.
- $\tau_j = 0$ 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.

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), or guiding by the context of application by cross-validation or permutation.

From optimization problem (11), the term "generalized" in the acronym of RGCCA embraces at least four 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. The fourth one relates to the function $g$ that enables to consider different functions of the covariance. This generalization is embodied by a triplet of parameters: (g, $\tau_j, \mathbf{C}$) and by the fact that an arbitrary number of blocks can be handled. This triplet of parameters offers a flexibility to RGCCA and allows to encompass a large number of multiblock component methods that were published for fifty years. Table 1-3 gives the correspondences between the triplet (g, $\tau_j, \mathbf{C}$) and the multiblock component methods. For a complete overview see (Tenenhaus *et al.* 2017).

## *Special cases*

Two families of methods have come to the fore in the field of multiblock data analysis. These methods rely on correlation-based or covariance-based criteria. Canonical correlation analysis (Hotelling 1936) is the seminal paper for the first family and Tucker's inter-battery factor analysis (Tucker 1958) for the second one. These two methods have been extended to more than two blocks in many ways:

- Main contributions for generalized canonical correlation analysis (GCCA) are found in (Horst 1961, Carroll (1968a), Kettenring (1971), Wold (1982), Wold (1985), Hanafi (2007)).\
- Main contributions for extending Tucker's method to more than two blocks come from (Carroll 1968b, Chessel and Hanafi (1996), Hanafi and Kiers (2006), Hanafi, Kohler, and Qannari (2010), Hanafi, Kohler, and Qannari (2011), Hanafi and Kiers (2006), Kramer (2007), Smilde, Westerhuis, and de Jong (2003), ten Berge (1988), Van de Geer (1984), Westerhuis, Kourti, and MacGregor (1998), Wold (1982), Wold (1985)).\
- (Carroll 1968b) proposed the "mixed" correlation and covariance criterion. (Van den Wollenberg 1977) combined correlation and variance for the two-block situation (redundancy analysis). This method is extended to the multiblock situation in (Tenenhaus and Tenenhaus 2011, Tenenhaus *et al.* (2017)).

In the two block case, optimization problem (11) reduces to:

$$\underset{\mathbf{a}_1,\mathbf{a}_2}{\text{maximize}}\ \text{cov}\left(\mathbf{X}_1\mathbf{a}_1, \mathbf{X}_2\mathbf{a}_2\right)\ \text{s.t.}\ \tau_j\|\mathbf{a}_j\|^2 + (1-\tau_j)\text{var}(\mathbf{X}_j\mathbf{a}_j) = 1, j = 1,2 \tag{12}$$

This problem has been introduced under the name of Regularized Canonical Correlation Analysis (Vinod 1976, Leurgans, Moyeed, and Silverman (1993), Shawe-Taylor and Cristianini (2004)). For various extreme cases $\tau_1 = 0$ or 1 and $\tau_2 = 0$ or 1, optimization problem (12) covers a situation which goes from Tucker's interbattery factor analysis (Tucker 1958) to Canonical Correlation Analysis (Hotelling 1933) while passing through redundancy analysis (Van den Wollenberg 1977). This framework corresponds exactly to the one proposed by (Borga, Landelius, and Knutsson 1997) and (Burnham, Viveros, and MacGregor 1996) and is reported in Table 1.

Table 1: two-block component methods.

| **Methods** | $g(x)$ | $\tau_j$ | $\mathbf{C}$ |
|---|---|---|---|
| **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) or **PLS Regression** (Wold, Martens, and Wold 1983) | $x$ | $\tau_1 = \tau_2 = 1$ | $\mathbf{C}_1$ |
| **Redundancy Analysis** (Van den Wollenberg 1977) | $x$ | $\tau_1 = 1\ ;\ \tau_2 = 0$ | $\mathbf{C}_1$ |
| **Regularized Redundancy Analysis** (Takane and Hwang 2007; Bougeard, Hanafi, and Qannari 2008, ; Qannari and Hanafi 2005) | $x$ | $0 \leq \tau_1 \leq 1\ ;\ \tau_2 = 0$ | $\mathbf{C}_1$ |

| **Methods** | $g(x)$ | $\tau_j$ | **C** |
|---|---|---|---|
| **Regularized Canonical Correlation Analysis** (Vinod 1976, Leurgans *et al.* (1993), Shawe-Taylor and Cristianini (2004)) | $x$ | $0 \leq \tau_1 \leq 1$ ; $0 \leq \tau_2 \leq 1$ | $\mathbf{C}_1$ |

In the multiblock data analysis literature, all blocks $\mathbf{X}_j, j = 1, \ldots, J$ are assumed to be connected and many criteria were proposed in the literature with the objective of finding block components satisfying some kind of covariance or correlation based optimality. Most of them are special cases of optimization problem (11). These multiblock component methods are listed in Table 2. PLS path modeling is also mentioned in this table. The great flexibility of PLS path modeling lies in the possibility of taking into account certain hypotheses on connections between blocks: the researcher decides which blocks are connected and which are not.

Table 2: Multiblock component methods as special cases of RGCCA.

| **Methods** | $g(x)$ | $\tau_j$ | **C** |
|---|---|---|---|
| **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) | $x^2$ | $\tau_j = 0, j = 1, \ldots, J$ | $\mathbf{C}_2$ |
| **SABSCOR** (Hanafi 2007) | $\|x\|$ | $\tau_j = 0, j = 1, \ldots, J$ | $\mathbf{C}_2$ |
| **SUMCOV-1** (Van de Geer 1984) | $x$ | $\tau_j = 1, j = 1, \ldots, J$ | $\mathbf{C}_2$ |
| **SSQCOV-1** (Hanafi and Kiers 2006) | $x^2$ | $\tau_j = 1, j = 1, \ldots, J$ | $\mathbf{C}_2$ |
| **SABSCOV-1** (Tenenhaus and Tenenhaus 2011, ; Kramer 2007) | $\|x\|$ | $\tau_j = 1, j = 1, \ldots, J$ | $\mathbf{C}_2$ |
| **SUMCOV-2** (Van de Geer 1984) | $x$ | $\tau_j = 1, j = 1, \ldots, J$ | $\mathbf{C}_3 = \begin{pmatrix} 0 & 1 & \cdots & 1 \\ 1 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 \\ 1 & \cdots & 1 & 0 \end{pmatrix}$ |
| **SSQCOV-2** (Hanafi and Kiers 2006) | $x^2$ | $\tau_j = 1, j = 1, \ldots, J$ | $\mathbf{C}_3$ |
| **PLS path modeling - mode B** (Wold 1982) | $\|x\|$ | $\tau_j = 0, j = 1, \ldots, J$ | $c_{jk} = 1$ for two connected block and $c_{jk} = 0$ otherwise |

The goal of many multiblock component methods is to find simultaneously block components and a global component. For that purpose, we consider $J$ blocks, $\mathbf{X}_1, \dots, \mathbf{X}_J$ connected to a $(J+1)$th block defined as the concatenation of the blocks, $\mathbf{X}_{J+1} = [\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_J]$. Several criteria were introduced in the literature and many of them are listed below.

Table 3: Multiblock component methods in a situation of $J$ blocks, $\mathbf{X}_1, \dots, \mathbf{X}_J$ connected to a $(J+1)$th block defined as the concatenation of the blocks, $\mathbf{X}_{J+1} = [\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_J]$.

| **Methods** | $g(x)$ | $\tau_j$ | $\mathbf{C}$ |
|---|---|---|---|
| **Generalized CCA** (Carroll 1968a) | $x^2$ | $\tau_j = 0, j = 1, \dots, J+1$ | $\mathbf{C}_4 = \begin{pmatrix} 0 & \cdots & 0 & 1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 1 \\ 1 & \cdots & 1 & 0 \end{pmatrix}$ |
| **Generalized CCA** (Carroll 1968b) | $x^2$ | $\tau_j = 0, j = 1, \dots, J_1$ ; $\tau_j = 1, j = J_1 + 1, \dots, J$ | $\mathbf{C}_4$ |
| **Hierarchical PCA** (Wold, S. and Kettaneh, N. and Tjessem, K. 1996) | $x^4$ | $\tau_j = 1, j = 1, \dots, J$ ; $\tau_{J+1} = 0$ | $\mathbf{C}_4$ |
| **Multiple Co-Inertia Analysis** (Chessel and Hanafi 1996, Westerhuis *et al.* (1998), Smilde *et al.* (2003)) | $x^2$ | $\tau_j = 1, j = 1, \dots, J$ ; $\tau_{J+1} = 0$ | $\mathbf{C}_4$ |
| **Multiple Factor Analysis** (Escofier and Pages 1994) | $x^2$ | $\tau_j = 1, j = 1, \dots, J+1$ | $\mathbf{C}_4$ |

The list of pre-specified multiblock component methods than can be used within the RGCCA package are reported below:

```
R> RGCCA:::available_methods()
```

```
 [1] "rgcca"     "sgcca"     "pca"       "spca"      "pls"       "spls"
 [7] "cca"       "ifa"       "ra"        "gcca"      "maxvar"    "maxvar-b"
[13] "maxvar-a"  "mfa"       "mcia"      "mcoa"      "cpca-1"    "cpca-2"
[19] "cpca-4"    "hpca"      "maxbet-b"  "maxbet"    "maxdiff-b" "maxdiff"
[25] "sabscor"   "ssqcor"    "ssqcov-1"  "ssqcov-2"  "ssqcov"    "sumcor"
[31] "sumcov-1"  "sumcov-2"  "sumcov"    "sabscov-1" "sabscov-2"
```

It is quite remarkable that the single optimization problem (11) offers a framework for all the multiblock component methods referenced in Table 1-3. From these perspectives, RGCCA provides a general framework for exploratory data analysis of multiblock datasets that has immediate practical consequences for a unified statistical analysis and implementation strategy. The very simple gradient-based Algorithm 1 is monotonically convergent and hits at convergence a stationary point. Two numerically equivalent approaches for solving the RGCCA optimization problem are available. A primal formulation described in (Tenenhaus and Tenenhaus 2011, Tenenhaus *et al.* (2017)) requires the handling of matrices of dimension $p_j \times p_j$. A dual formulation described in (Tenenhaus *et al.* 2015) requires the handling of matrices of dimension $n \times n$ . Therefore, the primal formulation of the RGCCA algorithm will be preferred when $n > p_j$ and the dual form will be used when $n \leq p_j$. The `rgcca()` function of the RGCCA package implements these two formulations and selects automatically the best one.

RGCCA is a component-based approach which aims to study the relationships between several sets of variables. The quality and interpretability of the RGCCA components are likely to be affected by the usefulness and relevance of the variables in each block. Therefore, it is an important issue to identify within each block which subsets of significant variables are active in the relationships between blocks. For instance, biomedical data are known to be measurements of intrinsically parsimonious processes. In order to account for this parsimony and to improve the interpretability, RGCCA has been extended to address the issue of variable selection. Specifically, Sparse GCCA (SGCCA) is proposed to combine RGCCA with an $\ell_1$-penalty promoting sparsity.

## 3.3. Sparse Generalized Canonical Correlation Analysis

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 (Tenenhaus, Philippe, Guillemot, Lê Cao, Grill, and Frouin 2014). The SGCCA optimization problem is defined as follows:

$$\underset{\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_J}{\text{maximize}} \sum_{j,k=1}^{J} c_{jk} g(\text{cov}(\mathbf{X}_j\mathbf{a}_j, \mathbf{X}_k\mathbf{a}_k)) \text{ s.t. } \|\mathbf{a}_j\|_2 \leq 1 \text{ and } \|\mathbf{a}_j\|_1 \leq s_j, j = 1, \ldots, J \tag{13}$$

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 by cross-validation or permutation procedures. Alternatively, values of $s_j$ can simply be chosen to result in desired amounts of sparsity.

SGCCA offers a sparse counterpart for all the covariance-based methods cited above.

The optimization problem (13) falls into the RGCCA framework with $\Omega_j = \{\mathbf{a}_j \in \mathbb{R}^{p_j}; \|\mathbf{a}_j\|_2 \leq 1; \|\mathbf{a}_j\|_1 \leq s_l\}$. $\Omega_j$ is defined as the intersection between the $\ell_2$-ball of radius 1 and the $\ell_1$-ball of radius $s_l \in \mathbb{R}_+^\star$ which are two compact sets. Hence, $\Omega_j$ is a compact set. Therefore, we can consider the following update for SGCCA:

$$\hat{\mathbf{a}}_j = \underset{\|\tilde{\mathbf{a}}_j\|_2 \leq 1; \|\tilde{\mathbf{a}}_j\|_1 \leq s_j}{\text{argmax}} \nabla_j f(\mathbf{a})^\top \tilde{\mathbf{a}}_j := r_j(\mathbf{a}) \tag{14}$$

According to (Witten, Tibshirani, and Hastie 2009), solution of (14) satisfies:

$$r_j(\mathbf{a}) = \hat{\mathbf{a}}_j = \frac{\mathcal{S}(\nabla_j f(\mathbf{a}), \lambda_j)}{\|\mathcal{S}(\nabla_j f(\mathbf{a}), \lambda_j)\|_2}, \text{ where } \lambda_j = \left\{ \begin{array}{ccc} 0 \text{ if} & \frac{\|\nabla_j f(\mathbf{a})\|_1}{\|\nabla_j f(\mathbf{a})\|_2} & \leq s_j \\ \text{find } \lambda_j \text{ such that} & \|\hat{\mathbf{a}}_j\|_1 & = s_j \end{array} \right., \tag{15}$$

where function $\mathcal{S}(., \lambda)$ is the soft-thresholding operator. When applied on a vector $\mathbf{x} \in \mathbb{R}^p$, this operator is defined as:

$$\mathbf{u} = \mathcal{S}(\mathbf{x}, \lambda) \Leftrightarrow u_j = \left\{ \begin{array}{ccc} \text{sign}(x_j)(|x_j| - \lambda), & \text{if } |x_j| & > \lambda \\ 0, & \text{if } |x_j| & \leq \lambda \end{array} \right., j = 1, \ldots, p. \tag{16}$$

We made the assumption that the $\ell_2$-ball of radius $1$ is not included in the $\ell_1$-ball of radius $s_j$ and the other way round. Otherwise systematically, only one of the two constraints is active. This assumption is true when the corresponding spheres intersect. This assumption can be translated into conditions on $s_j$.

The norm equivalence between $\|.\|_1$ and $\|.\|_2$ can be formulated as the following inequality:

$$\forall \mathbf{x} \in \mathbb{R}^{p_j}, \ \|\mathbf{x}\|_2 \leq \|\mathbf{x}\|_1 \leq \sqrt{p_j}\|\mathbf{x}\|_2. \tag{17}$$

This can be converted into a condition on $s_j$: $1 \leq s_j \leq \sqrt{p_j}$. When such condition is fulfilled, the $\ell_2$-sphere of radius $1$ and the $\ell_1$-sphere of radius $s_j$ necessarily intersect. Within the RGCCA package, for consistency with the value of $\tau_j \in [0, 1]$, the level of sparsity for $\mathbf{a}_j$ is controlled with $1/s_j \in [1/\sqrt{p_j}, 1]$.

Several strategies such as Binary Search or the Projection On Convex Set algorithm (POCS), also known as alternating projection method (Boyd, Dattorro *et al.* 2003), can be used to determine the optimal $\lambda_j$ verifying the $\ell_1$-norm constraint. Here, a much faster approach described in (Gloaguen, Guillemot, and Tenenhaus 2017) is implemented within the RGCCA package.

The SGCCA algorithm is similar to the RGCCA algorithm and keeps the same convergence properties. Empirically, we note that the S/RGCCA algorithm is found to be not sensitive to the starting point and usually reaches convergence (`tol` = $10^{-16}$) within a few iterations.

## 3.4. Higher level RGCCA algorithm

In many applications, several components per block need to be identified. The traditional approach consists of incorporating the single-unit RGCCA algorithm in a deflation scheme, and computing the desired number of components sequentially. More precisely, the RGCCA optimization problem returns a set of $J$ optimal block-weight vectors. denoted here $\mathbf{a}_j^{(1)}$, $j = 1, \ldots, J$. Let $\mathbf{y}_j^{(1)} = \mathbf{X}_j\mathbf{a}_j^{(1)}$, $j = 1, \ldots, J$ be the corresponding block components. Two strategies to determine higher-level weight vectors are presented. The first one yields orthogonal block components and the second one yields orthogonal weight vectors. Deflation is the most straightforward way to add orthogonality constraints. This deflation procedure is sequential and consists in replacing within the RGCCA optimization problem the data matrix $\mathbf{X}_j$ by $\mathbf{X}_j^{(1)}$ its projection onto either:

- the orthogonal subspace of $\mathbf{y}_j^{(1)}$ for orthogonal block-components:

$$\mathbf{X}_j^{(1)} = \left(\mathbf{I} - \mathbf{P}^{\perp}_{\mathbf{y}_j^{(1)}}\right)\mathbf{X}_j = \mathbf{X}_j - \mathbf{y}_j^{(1)}\left({\mathbf{y}_j^{(1)}}^\top \mathbf{y}_j^{(1)}\right)^{-1}{\mathbf{y}_j^{(1)}}^\top \mathbf{X}_j,$$

or

- the orthogonal subspace of $\mathbf{a}_j^{(1)}$ for orthogonal block-weight vectors:

$$\mathbf{X}_j^{(1)} = \mathbf{X}_j\left(\mathbf{I} - \mathbf{P}^{\perp}_{\mathbf{a}_j^{(1)}}\right) = \mathbf{X}_j - \mathbf{X}_j\mathbf{a}_j^{(1)}\left({\mathbf{a}_j^{(1)}}^\top \mathbf{a}_j^{(1)}\right)^{-1}{\mathbf{a}_j^{(1)}}^\top.$$

The second level RGCCA optimization problem boils down to:

$$\max_{\mathbf{a}_1,\ldots,\mathbf{a}_J} \sum_{j,k=1}^{J} c_{jk} \, \mathrm{g}\left(n^{-1}\mathbf{a}_j^\top \mathbf{X}_j^{(1)^\top} \mathbf{X}_k^{(1)} \mathbf{a}_k\right) \text{ s.t. } \mathbf{a}_j \in \Omega_j. \tag{18}$$

The optimization problem (18) is solved using Algorithm 1 and returns a set of optimal block weight vectors $\mathbf{a}_j^{(2)}$ and block components $\mathbf{y}_j^{(2)} = \mathbf{X}_j^{(1)}\mathbf{a}_j^{(2)}$, for $j = 1\ldots, J$. According to the mode of deflation, either the block weight vectors or the block components will be orthogonal.

For orthogonal block weight vectors, $\mathbf{y}_j^{(2)} = \mathbf{X}_j^{(1)}\mathbf{a}_j^{(2)} = \mathbf{X}_j\left(\mathbf{I} - \mathbf{P}^\perp_{\mathbf{a}_j^{(1)}}\right)\mathbf{a}_j^{(2)} = \mathbf{X}_j\mathbf{a}_j^{(2)}$ naturally expresses as a linear combination of the original variables. For orthogonal block component, as $\mathbf{y}_j^{(1)} = \mathbf{X}_j\mathbf{a}_j^{(1)}$, the range space of $\mathbf{X}_j^{(1)}$ is included in the range space of $\mathbf{X}_j$, meaning that any block component $\mathbf{y}_j^{(2)}$ belonging to the range space of $\mathbf{X}_j^{(1)}$ can also be expressed in term of the original block $\mathbf{X}_j$: that is, it exists $\mathbf{a}_j^{(2)\star}$ such that $\mathbf{y}_j^{(2)} = \mathbf{X}_j^{(1)}\mathbf{a}_j^{(2)} = \mathbf{X}_j\mathbf{a}_j^{(2)\star}$. It implies that whatever the choice of the mode of delfation the block component can always be expressed in terms of the original variable.

This deflation procedure can be iterated in a very flexible way. For instance, 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. This might be desired in a supervised setting where we want to predict a univariate block from other blocks. In that case, the deflation procedure applies to all blocks except the one to predict.

To conclude this section, when the superblock option is used, various deflation strategies (what to deflate and how) have been proposed in the literature. We propose, as the default option, to deflate only the superblock with respect to its global components:

$$\mathbf{X}_{J+1}^{(1)} = \left(\mathbf{I} - \mathbf{P}^\perp_{\mathbf{y}_{J+1}^{(1)}}\right)\mathbf{X}_{J+1} = \left[\mathbf{X}_1^{(1)}, \ldots, \mathbf{X}_J^{(1)}\right]$$

The individual blocks $\mathbf{X}_j^{(1)}$s are then retrieved from the deflated superblock. This strategy enables recovering Multiple Factor Analysis (`ade4:::mfa()/FactoMineR::MFA`). We follow the deflation strategy described in (Chessel and Hanafi 1996) (`ade4:::mcoa()`) for Multiple Co-inertia Analysis, which is one of the most popular and established methods of the multiblock literature.

Guidelines describing R/SGCCA in practice are provided in (Garali, Adanyeguh, Ichou, Perlbarg, Seyer, Colsch, Moszer, Guillemot, Durr, Mochel, and Tenenhaus 2018). The usefulness and versatility of the RGCCA package are illustrated in the next section.

## 4. Practical session

### 4.1. RGCCA for the Russett dataset.

In this section, we propose to reproduce some of the results presented in (Tenenhaus and Tenenhaus 2011) from 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.

```
R> library(RGCCA)
```

```
R> data(Russett)
R> colnames(Russett)

 [1] "gini"     "farm"     "rent"     "gnpr"     "labo"     "inst"
 [7] "ecks"     "death"    "demostab" "demoinst" "dictator"
```

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`, `death`] measures "Political Instability":
  - `inst` = Instability of executive (45-61),
  - `ecks` = Number of violent internal war incidents (46-61),
  - `death` = Number of people killed as a result of civic group violence (50-62).
  - `demo` = Political regime: stable democracy, unstable democracy or dictatorship. Due to redundancy, the dummy variable "unstable democracy" has been left out.

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

```
R> A = list(Agric = Russett[,c("gini","farm","rent")],
+           Ind = Russett[,c("gnpr","labo")],
+           Polit = Russett[ , c("inst", "ecks",  "death", "demostab", "dictator")])
R>
R> lab = factor(apply(Russett[, 9:11], 1, which.max),
+               labels = c("demost",
+                          "demoinst",
+                          "dict"))
```

**Preprocessing.** In general, and especially for the covariance-based criterion, the data blocks might be pre-processed to ensure comparability between variables and blocks. In order to ensure comparability between variables standardization is applied (zero mean and unit variance). Such a preprocessing is reached by setting the `scale` argument to `TRUE` (default value) in the `rgcca()` function. 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 *et al.* 1998). This two-step procedure leads to $\mathrm{tr}(\mathbf{X}_j^\top \mathbf{X}_j) = n$ 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_block` argument to `TRUE` or `inertia` (default value) in the `rgcca()` function. If `scale_block = "lambda1"`, each block is divided by the square root of the highest eigenvalue of its empirical covariance matrix. If standardization is applied (`scale = TRUE`), the block scaling is applied on the result of the standardization.

**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 encoded through the design matrix **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:

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

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

**Choice of the scheme function g**. Typical choices of scheme functions are $g(x) = x, x^2$, or $|x|$. 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 *et al.* 2017).

RGCCA using the pre-defined design matrix **C**, the factorial scheme ($g(x) = x^2$), $\tau = 1$ for all blocks (full covariance criterion) and a number of (orthogonal) components equal to 2 for all blocks is obtained by specifying appropriately the arguments `connection`, `scheme`, `tau`, `ncomp`, `comp_orth` in `rgcca()`. `verbose` (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.

```
R> fit = rgcca(blocks = A, connection = C,
+              tau = 1, ncomp = 2,
+              scheme = "factorial",
+              scale = TRUE,
+              scale_block = FALSE,
+              comp_orth = TRUE,
+              verbose = FALSE)
```

the `print()` function allows summarizing the RGCCA analysis.

```
R> print(fit)

Call: method='rgcca', superblock=FALSE, scale=TRUE, scale_block=FALSE, init='svd',
bias=TRUE, tol=1e-08, NA_method='nipals', ncomp=c(2,2,2), response=NULL,
comp_orth=TRUE
There are J = 3 blocks.
The design matrix is:
      Agric Ind Polit
Agric     0   0     1
Ind       0   0     1
Polit     1   1     0
```

```
The factorial scheme is used.
Sum_{j,k} c_jk g(cov(X_j a_j, X_k a_k) = 7.9469

The regularization parameter used for Agric is: 1
The regularization parameter used for Ind is: 1
The regularization parameter used for Polit is: 1
```

The block-weight vectors solution of the optimization problem (11) are available as output of the `rgcca()` function in `fit$a` and correspond exactly to the weight vectors reported in (Tenenhaus and Tenenhaus 2011, see Figure 5). It is possible to display specific block-weight vector(s) (`type = "weight"`) block-loadings vector(s) (`type = "loadings"`) using the generic `plot()` function and specifying the arguments `block` and `component` accordingly.

```
R> plot(fit, type = "weight", block = 1:3, comp = 1,
+       display_order = FALSE, cex = 1.3)
```

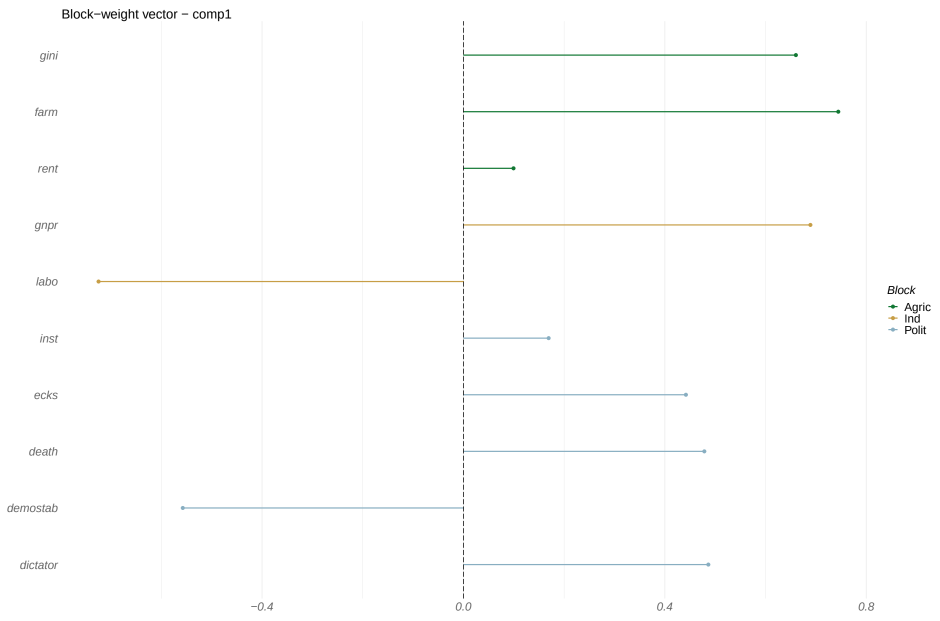


Figure 1: block weight vectors

As component-based method, the RGCCA package provides block components as output of the `rgcca()` function in `fit$Y` and graphical representations, including factor plot (`type = "sample"`), correlation circle (`type = "cor_circle"`) or biplot (`type = "biplot"`). 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.

```
R> plot(fit, type = "sample",
+       block = 1:2, comp = 1,
+       resp = lab, repel = TRUE, cex = 1.3)
```

Countries aggregate together when they share similarities. It may be noted that the lower right quadrant concentrates on dictatorships. It is difficult for a country to escape dictatorship when its industrial

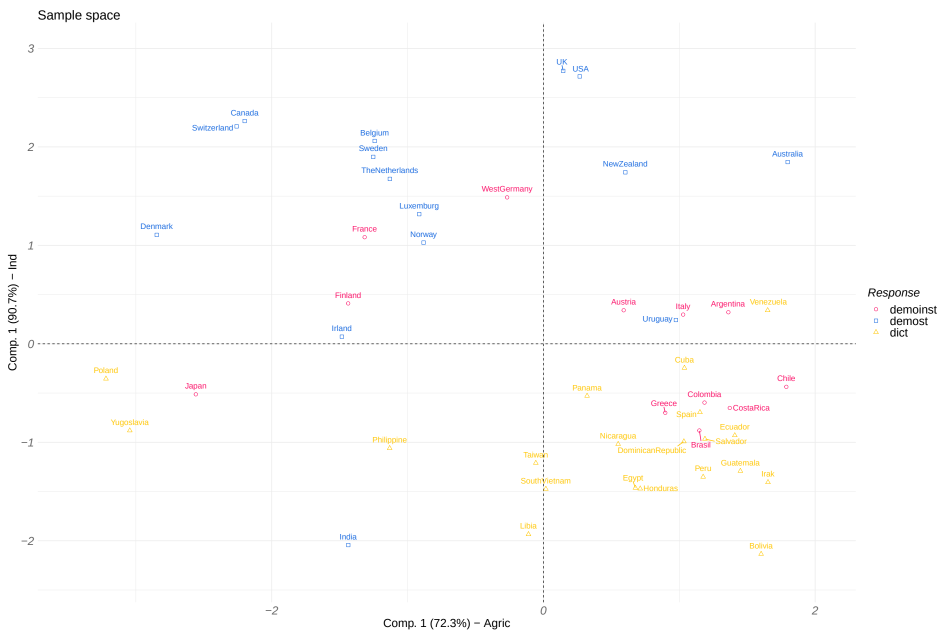


Figure 2: graphical display of the countries obtained by crossing y11 and y21 and labeled according to their political regime

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). The Average Variance Explained (AVE) defined below is also reported in the axes of the Figure. The AVE of block $\mathbf{X}_j$ for a specific block component $\mathbf{y}_j$ is defined as:

$$\text{AVE}(\mathbf{X}_j) = \frac{1}{\|\mathbf{X}_j\|^2} \sum_{h=1}^{p_j} \text{var}(\mathbf{x}_{jh}) \times \text{cor}^2(\mathbf{x}_{jh}, \mathbf{y}_j) \tag{19}$$

$\text{AVE}(\mathbf{X}_j)$ varies between 0 and 1 and reflects the proportion of variance captured by $\mathbf{y}_j$.
Additional indicators of model quality are also proposed:

- For all blocks:

$$\text{AVE}(\text{outermodel}) = \left(1/\sum_j \|\mathbf{X}_j\|^2\right) \sum_j \|\mathbf{X}_j\|^2 \text{AVE}(\mathbf{X}_j) \tag{20}$$

- For the inner model:

$$\text{AVE}(\text{innermodel}) = \left(1/\sum_{j<k} c_{jk}\right) \sum_{j<k} c_{jk} \text{cor}^2(\mathbf{y}_j, \mathbf{y}_k) \tag{21}$$

These indicators of model quality are also available as output of the `rgcca()` function in `fit$AVE`. These AVEs can be visualized using the generic `plot()` function.

```
R> plot(fit, type = "ave", cex = 1.3)
```

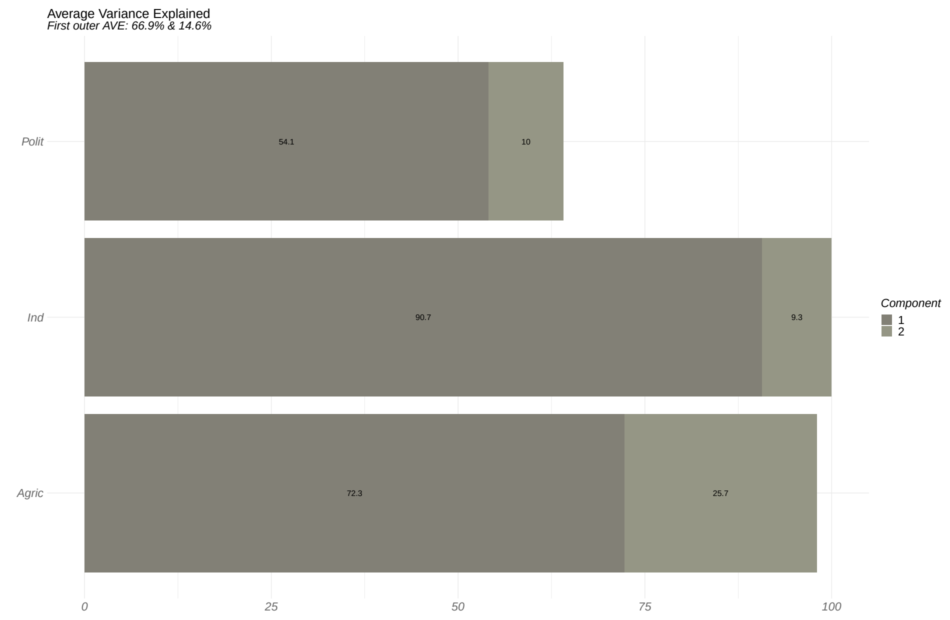


Figure 3: Average variance explained of the various blocks

The strength of the relations between each block component and each variable can be visualized using correlation circle or biplot representations.

```
R> plot(fit, type = "cor_circle", block = 1, comp = 1:2, display_blocks = 1:3)
```

By default all the variables are displayed on the correlation circle. However, it is possible to choose the block(s) to display (`display_blocks`)in the correlation_circle.

```
R> plot(fit, type = "biplot", block = 1,
+       comp = 1:2, repel = TRUE,
+       resp = lab,
+       show_arrow = TRUE)
```

As we will see in the next section, when the superblock option is considered (`superblock = TRUE` or `method` set to a method that induces the use of superblock) global components can be derived. The space spanned by the global components can be viewed as a consensus space that integrated all the modalities and facilitates the visualization of the results and their interpretation.

**Assessment of the reliability of parameter estimates.** It is possible to use a bootstrap resampling method to assess the reliability of parameter estimates (block-weight/loading vectors) obtained using RGCCA. $B =$`n_boot` 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 RGCCA estimates. We calculate the standard deviation of the estimates across the bootstrap samples, from which we derived, bootstrap confidence intervals, t-ratio (defined as the ratio of the parameter estimate to its bootstrap estimate of the standard deviation) and p-value (the p-value is computed by assuming that the ratio of the parameter estimate to its standard deviation follows the standardized normal distribution) to indicate how reliably parameters were estimated. Since several

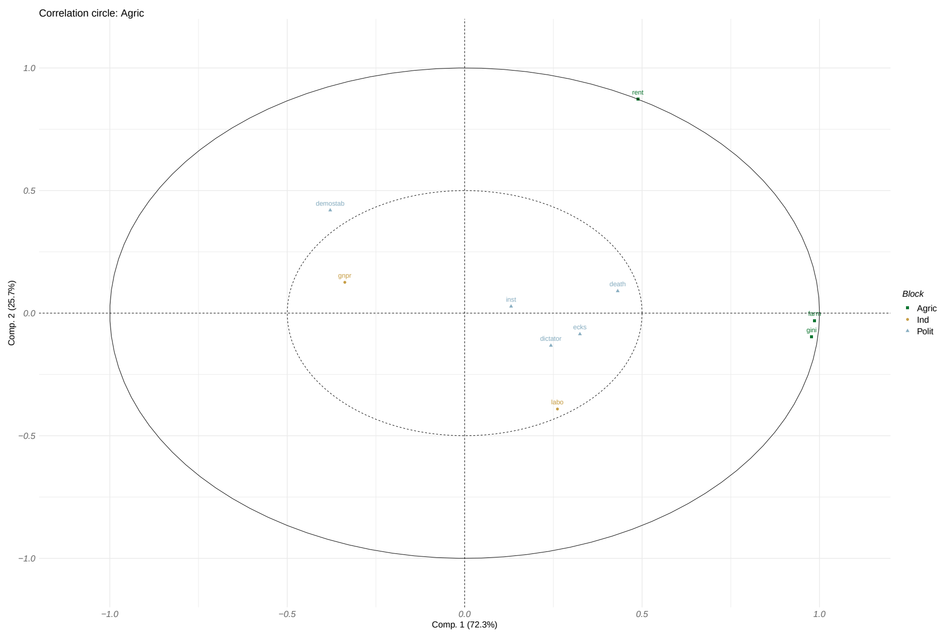


Figure 4: correlation circle associated with the two first components of Ariculture

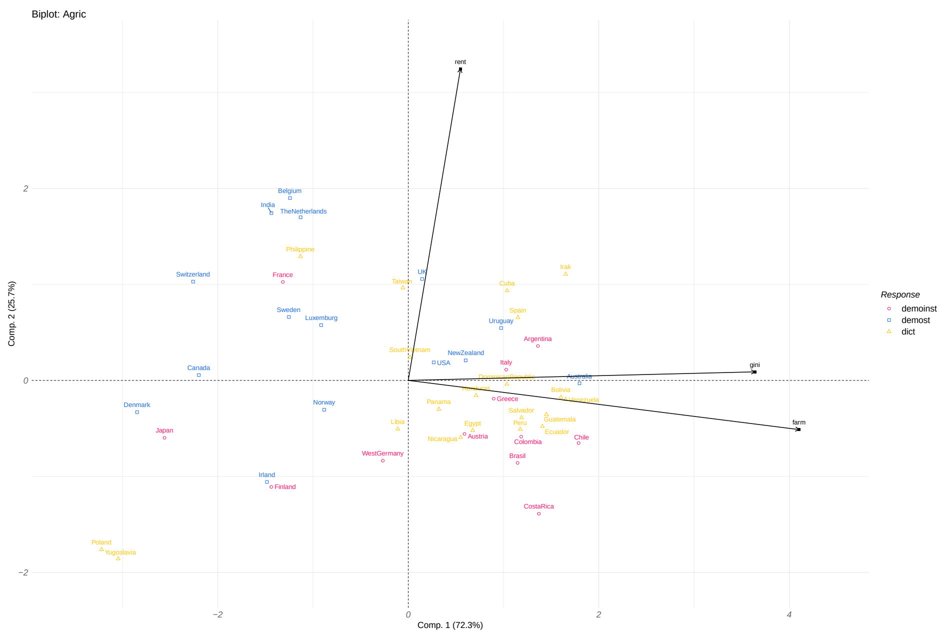


Figure 5: biplot associated with the two first components of Ariculture

p-values are constructed simultaneously, FDR correction can be applied for controlling the False Discovery Rate. This function is available using the `rgcca_bootstrap()` function of the RGCCA package.

```
R> boot_out = rgcca_bootstrap(fit, n_boot = 500, n_cores = 1)
```

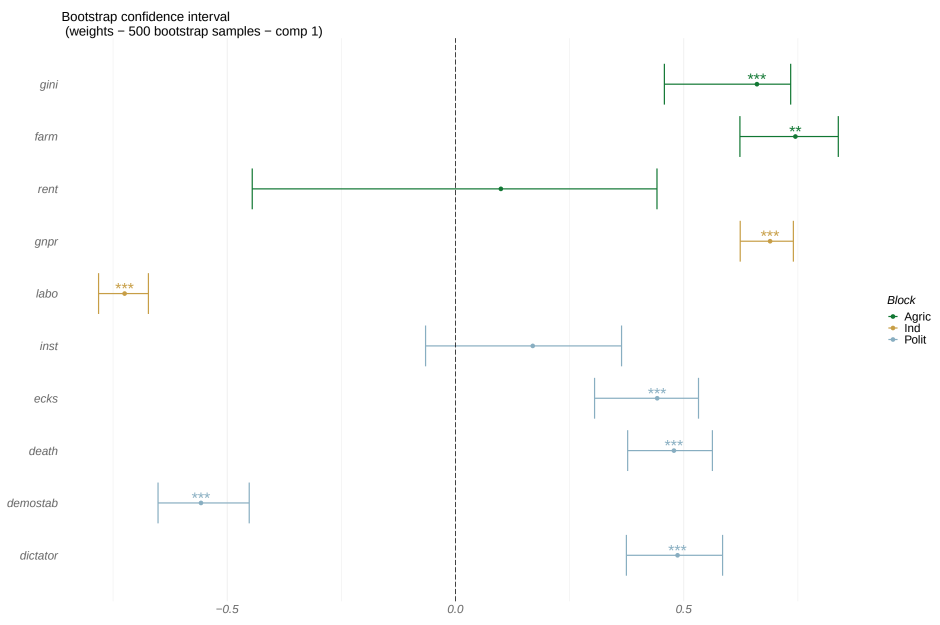
The bootstrap results are detailed using the `print()` function.

```
R> print(boot_out, block = 1:3, ncomp = 1)
```

```
Call: method='rgcca', superblock=FALSE, scale=TRUE, scale_block=FALSE, init='svd',
bias=TRUE, tol=1e-08, NA_method='nipals', ncomp=c(2,2,2), response=NULL,
comp_orth=TRUE
There are J = 3 blocks.
The design matrix is:
      Agric Ind Polit
Agric     0   0     1
Ind       0   0     1
Polit     1   1     0

The factorial scheme is used.

Extracted statistics from 500 bootstrap samples.
Block-weight vectors for component 1:
         estimate    mean     sd lower_bound upper_bound bootstrap_ratio
gini       0.6602  0.6347 0.0730      0.4575       0.734           9.050
farm       0.7445  0.7244 0.0928      0.6230       0.838           8.025
rent       0.0994  0.0788 0.2289     -0.4452       0.441           0.434
gnpr       0.6891  0.6885 0.0292      0.6236       0.740          23.610
labo      -0.7247 -0.7241 0.0271     -0.7817      -0.673         -26.768
inst       0.1692  0.1666 0.1119     -0.0655       0.364           1.512
ecks       0.4418  0.4347 0.0601      0.3046       0.532           7.356
death      0.4784  0.4708 0.0485      0.3771       0.563           9.857
demostab  -0.5574 -0.5518 0.0516     -0.6515      -0.452         -10.801
dictator   0.4864  0.4830 0.0521      0.3743       0.585           9.343
            pval adjust.pval
gini     0.00000     0.00000
farm     0.00402     0.00618
rent     0.44509     0.52363
gnpr     0.00000     0.00000
labo     0.00000     0.00000
inst     0.07066     0.10095
ecks     0.00000     0.00000
death    0.00000     0.00000
demostab 0.00000     0.00000
dictator 0.00000     0.00000
```

and displayed using the `plot()` function.

```
R> plot(boot_out, type = "weight",
+       block = 1:3, comp = 1,
+       display_order = FALSE, cex = 1.3,
+       show_stars = TRUE)
```



Each weight is shown along with its associated bootstrap confidence interval and stars (`show_stars = TRUE`) reflecting the p-value of assigning a strictly positive or negative weight to this variable.

## 4.2. RGCCA with superblock

In this section, we consider Multiple Co-Inertia Analysis (Chessel and Hanafi 1996) (MCOA, also called MCIA in (Cantini, Zakeri, Hernandez, Naldi, Thieffry, Remy, and Baudot 2021)) with 2 components per block.

See `available_methods()` for a list of pre-specified multiblock component methods.

```
R> fit.mcoa = rgcca(blocks=A, method = "mcoa", ncomp = 2)
```

the `print()` function allows summarizing the RGCCA analysis.

```
R> print(fit.mcoa)
```

```
Call: method='mcoa', superblock=TRUE, scale=TRUE, scale_block='inertia', init='svd',
bias=TRUE, tol=1e-08, NA_method='nipals', ncomp=c(2,2,2,2), response=NULL,
comp_orth=FALSE
There are J = 4 blocks.
The design matrix is:
           Agric Ind Polit superblock
Agric          0   0     0          1
Ind            0   0     0          1
Polit          0   0     0          1
superblock     1   1     1          0

The factorial scheme is used.
Sum_{j,k} c_jk g(cov(X_j a_j, X_k a_k) = 3.578

The regularization parameter used for Agric is: 1
The regularization parameter used for Ind is: 1
The regularization parameter used for Polit is: 1
The regularization parameter used for superblock is: 0
```

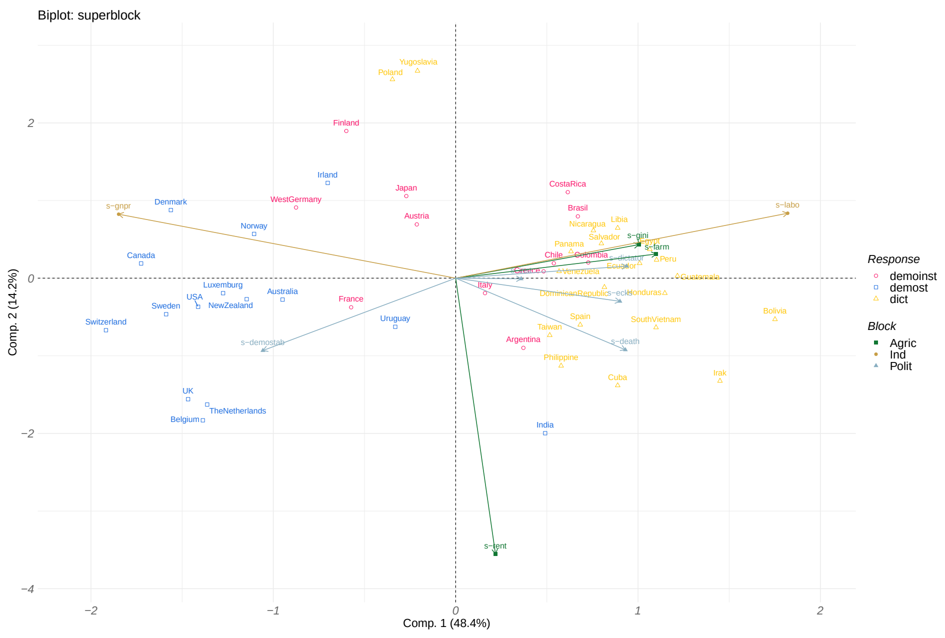


Figure 6: A biplot graphical display of the countries obtained by crossing the two first components of the superblock. Individuals are labeled according to their political regime and variables according to their block membership.

Interestingly, this output reports the arguments that has been implicitly specified to reach MCOA.

It is possible to display specific output as previously using the generic `plot()` function by specifying the argument `type` accordingly. MCOA enables the individuals to be represented in the space spanned by the two first global components. The biplot representation associated with this consensus space is given below.

```
R> plot(fit.mcoa, type = "biplot",
+       block = 4, comp = 1:2,
+       response = lab,
+       repel = TRUE, cex = 1.3)
```

As previously, this model can be easily bootstrapped using the `rgcca_bootstrap()` function and the bootstrap confidence intervals are still available using the `print()/plot()` functions.

## 4.3. Choice of the shrinkage parameters

Three fully automatic strategies are proposed to select the optimal shrinkage parameters:

**The Schafer and Strimmer analytical formula.** For each block $j$, an "optimal" shrinkage parameter $\tau_j$ can be obtained using the Schafer and Strimmer analytical formula (Schäfer and Strimmer 2005) by setting the `tau` argument of the `rgcca()` function to `"optimal"`.

```
R> fit = rgcca(blocks = A, connection=C, response=3,
+              tau = "optimal", scheme = "factorial")
```

The optimal shrinkage parameters are given by:

```
R> fit$call$tau
```

```
[1] 0.08853216 0.02703256 0.08422566
```

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.

As previously, all the fitted RGCCA object can be visualized/bootstraped using the `print`, `plot()` and `rgcca_bootstrap()` functions.

**Permutation strategy.** A permutation based strategy very similar to the one proposed in (Witten *et al.* 2009) has been also integrated within the RGCCA package through the `rgcca_permutation()` function. This function is used to select automatically the regularization parameters for R/SGCCA.

For each set of regularization parameters (generally this will be a $J$-dimensional vector), repeat the following `n_perm` times, for (`n_perm` large):

> The rows of $\mathbf{X}_1, \ldots, \mathbf{X}_J$ are randomly permuted to obtained permuted data sets $\mathbf{X}_1^*, \ldots, \mathbf{X}_J^*$.
>
> S/RGCCA is run on the permuted data set $\mathbf{X}_1^*, \ldots, \mathbf{X}_J^*$ and we record the value of the objective function, denoted $t^*$.
>
> S/RGCCA is run on the original data $\mathbf{X}_1, \ldots, \mathbf{X}_J$ and we record the value of the objective function, denoted $t$.
>
> The resulting p-value is given by the fraction of permuted $t*$ that exceed the real $t$t obtained from the non-permuted blocks.

Then choose the set of tuning parameters that yields the smallest value in step (4.3). This procedure is available though the `rgcca_permutation()` function.

```
R> set.seed(123)
R> perm_out = rgcca_permutation(blocks = A, connection=C,
+                               par_type = "tau",
+                               par_value = c(.51, .13, 0),
+                               par_length = 10,
+                               n_cores = 1,
+                               n_perms = 10)
```

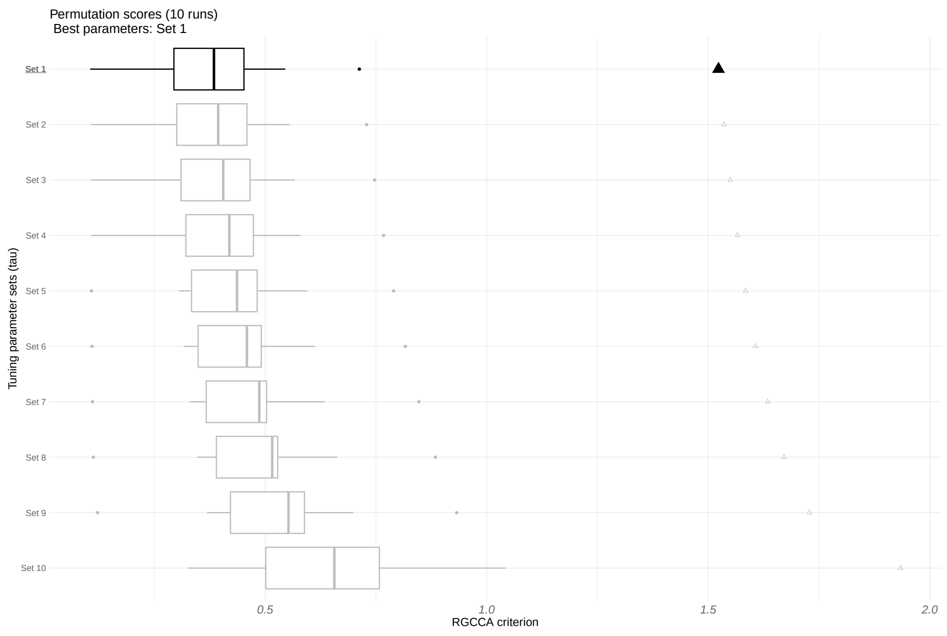
By default, the `rgcca_permutation()` function takes 10 sets of tuning parameters between min values (0 for RGCCA and $1/sqrt(ncol)$ for SGCCA) and 1. Results of the permutation procedure are summarized/displayed using the generic `print()/plot()` function

```
R> print(perm_out)

Call: method='rgcca', superblock=FALSE, scale=TRUE, scale_block=TRUE, init='svd',
bias=TRUE, tol=1e-08, NA_method='nipals', ncomp=c(1,1,1), response=NULL,
comp_orth=TRUE
There are J = 3 blocks.
The design matrix is:
      Agric Ind Polit
Agric     0   0     1
Ind       0   0     1
Polit     1   1     0

The factorial scheme is used.
```

```
Tuning parameters (tau) used:
   Agric   Ind Polit
1  0.510 0.130     0
2  0.453 0.116     0
3  0.397 0.101     0
4  0.340 0.087     0
5  0.283 0.072     0
6  0.227 0.058     0
7  0.170 0.043     0
8  0.113 0.029     0
9  0.057 0.014     0
10 0.000 0.000     0

   Tuning parameters Criterion Permuted criterion    sd zstat p-value
1              Set 1      1.52              0.392 0.165  6.87       0
2              Set 2      1.54              0.400 0.168  6.76       0
3              Set 3      1.55              0.409 0.172  6.63       0
4              Set 4      1.57              0.420 0.176  6.50       0
5              Set 5      1.58              0.433 0.181  6.36       0
6              Set 6      1.61              0.448 0.187  6.20       0
7              Set 7      1.63              0.467 0.194  6.02       0
8              Set 8      1.67              0.493 0.203  5.82       0
9              Set 9      1.73              0.531 0.214  5.61       0
10            Set 10      1.93              0.665 0.230  5.52       0
The best combination is: Set 1 for a z score of 6.87 and a p-value of 0.
```

and displayed using the `plot()` function.

```
R> plot(perm_out, cex = 1.3)
```



The fitted permutation object, `perm_out`, can be directly provided as output of `rgcca()` and visualized/bootstrapped as usual.

```
R> fit = rgcca(perm_out)
```

Of course, it is possible to define explicitly the combination of regularization parameters to be tested. In that case a matrix of dimension $K \times J$ is required. Each row of this matrix corresponds to one set of tuning parameters.

```
R> fit.perm = rgcca_permutation(A, connection = C,
+                              par_type = "tau",
+                              par_value = rbind(rep(1, 3),
+                                                seq(0, 1, l=3),
+                                                rep(0, 3),
+                                                sapply(A, RGCCA:::tau.estimate)),
+                              n_cores = 1, n_perms = 5)
```

Alternatively a numeric vector of length $J$ indicating the range of values to be tested: from the minimum values (0 for RGCCA and $1/sqrt(ncol)$ for SGCCA) to the maximum values specified by the user with `par_value`.

```
R> fit.perm = rgcca_permutation(A, connection = C,
+                              par_type = "tau",
+                              par_value = seq(0, 1, l=3),
+                              n_cores = 1, n_perms = 5)
```

**Cross-validation strategy.** The optimal tuning parameters can also be determined by cross-validating different indicators of quality, namely:

- For Classification: `Accuracy`, `Kappa`, `F1`, `Sensitivity`, `Specificity`, `Pos_Pred_Value`, `Neg_Pred_Value`, `Precision`, `Recall`, `Detection_Rate`, `Balanced_Accuracy`.
- For regression: `RMSE` and `MAE`.

This cross-validation protocol is made available through the `rgcca_cv()` functionand is used here for predicting the qualitative variable political regime from the two blocks `Agriculture inequality` and `Industrial development`.

```
R> blocks <-
+   list(agriculture = Russett[, seq(3)],
+        industry = Russett[, 4:5],
+        lab = as.matrix(factor(apply(Russett[, 9:11], 1, which.max),
+                        labels = c("Stable",
+                                   "Unstable",
+                                   "Dictator")))
+             )
R>
R> set.seed(27) #my favorite number
R> inTraining <- caret:::createDataPartition(blocks[[3]],
+                                   p = .75, list = FALSE)
R> training <- lapply(blocks,
+                    function(x) x[inTraining, , drop = FALSE])
R>
R> testing  <- lapply(blocks,
+                    function(x) x[-inTraining, , drop = FALSE])
R>
R> cv_out = rgcca_cv(blocks = training, response = 3,
+                   par_type = "tau",
+                   prediction_model = "lda",
+                   n_run = 10, k = 3,
+                    validation = "kfold",
+                    ncomp = 1, metric = "Accuracy")
```

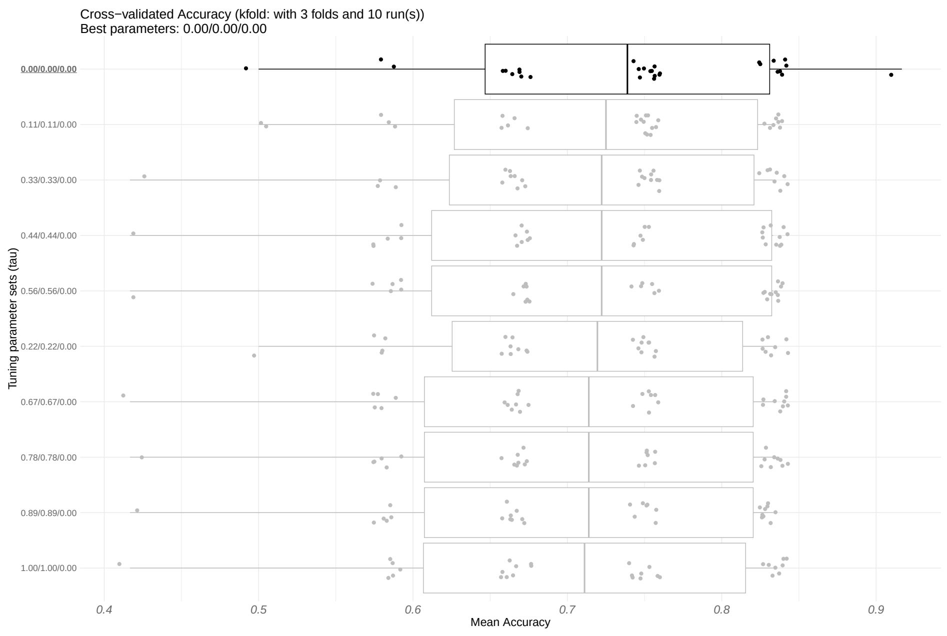
`rgcca_cv()` relies on the `caret` package. As direct consequence an astonishing large number of models are made available (see `caret::modelLookup()`). Results of the cross validation procedure are reported/displayed using the generic `print()`/`plot()` function

```
R> print(cv_out)
```

```
Call: method='rgcca', superblock=FALSE, scale=TRUE, scale_block=TRUE, init='svd',
bias=TRUE, tol=1e-08, NA_method='nipals', ncomp=c(1,1,1), response=3,
comp_orth=TRUE
There are J = 3 blocks.
The design matrix is:
            agriculture industry lab
agriculture           0        0   1
industry              0        0   1
lab                   1        1   0

The factorial scheme is used.

Tuning parameters (tau) used:
   agriculture industry lab
1        1.000    1.000   0
2        0.889    0.889   0
3        0.778    0.778   0
4        0.667    0.667   0
5        0.556    0.556   0
6        0.444    0.444   0
7        0.333    0.333   0
8        0.222    0.222   0
9        0.111    0.111   0
10       0.000    0.000   0

Validation: kfold with 3 folds and 10 run(s))
Prediction model: lda

   Tuning parameters Mean Accuracy     Sd
1     1.00/1.00/0.00         0.711 0.1043
2     0.89/0.89/0.00         0.714 0.1065
3     0.78/0.78/0.00         0.714 0.1065
4     0.67/0.67/0.00         0.714 0.1065
5     0.56/0.56/0.00         0.722 0.1101
6     0.44/0.44/0.00         0.722 0.1101
7     0.33/0.33/0.00         0.722 0.0987
8     0.22/0.22/0.00         0.719 0.0941
9     0.11/0.11/0.00         0.725 0.0982
10    0.00/0.00/0.00         0.739 0.0922

The best combination is: 0.00/0.00/0.00 for a mean Accuracy of 0.739.
```

and displayed using the `plot()` function.

```
R> plot(cv_out, cex = 1.3)
```



The fitted cval object can be provided as output of `rgcca()` and the resulting optimal model can be visualized/bootstrapped as usual.

```
R> fit = rgcca(cv_out)
```

At last, `rgcca_predict()` can be used for predicting new blocks.

```
R> perf = rgcca_predict(fit, blocks_test = testing, prediction_model = "lda")
```

and a `caret` summary of the performances be reported

```
R> perf$confusion$test

Confusion Matrix and Statistics

          Reference
Prediction Dictator Stable Unstable
  Dictator        5      0        2
  Stable          0      3        1
  Unstable        0      0        0

Overall Statistics

               Accuracy : 0.7273
                 95% CI : (0.3903, 0.9398)
    No Information Rate : 0.4545
    P-Value [Acc > NIR] : 0.06478

                  Kappa : 0.5541

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: Dictator Class: Stable Class: Unstable
```

```
Sensitivity                  1.0000        1.0000          0.0000
Specificity                  0.6667        0.8750          1.0000
Pos Pred Value               0.7143        0.7500             NaN
Neg Pred Value               1.0000        1.0000          0.7273
Prevalence                   0.4545        0.2727          0.2727
Detection Rate               0.4545        0.2727          0.0000
Detection Prevalence         0.6364        0.3636          0.0000
Balanced Accuracy            0.8333        0.9375          0.5000
```

All the functions presented previously are designed for sparse analysis. This will be illustrated in the next section.

## 5. High dimensional case study: Glioma Data

**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.

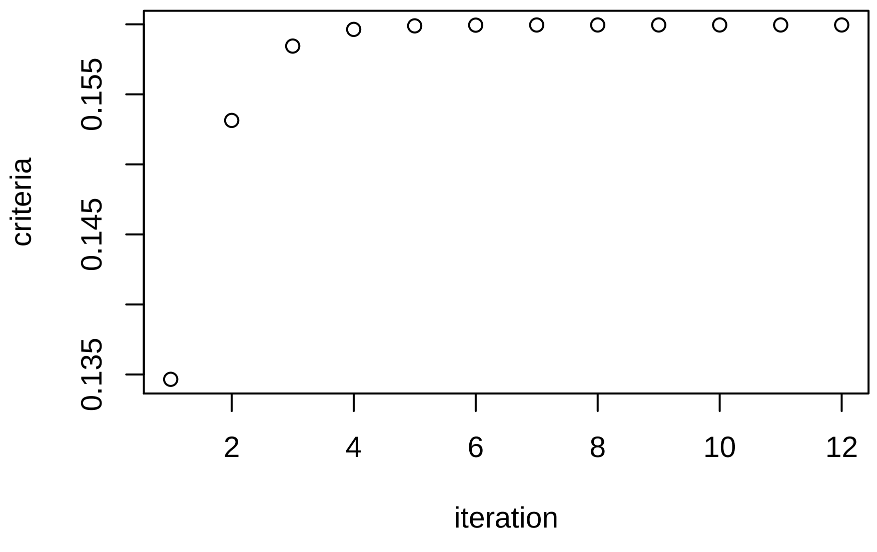
**Description of the data.** Pretreatment frozen tumor samples were obtained from 53 children with newly diagnosed pHGG from Necker Enfants Malades (Paris, France) (Puget, Philippe, Bax, Job, Varlet, Junier, Andreiuolo, Carvalho, Reis, Guerrini-Rousseau, Roujeau, Dessen, Richon, Lazar, Le Teuff, Sainte-Rose, Geoerger, Vassal, Jones, and Grill 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 the others.

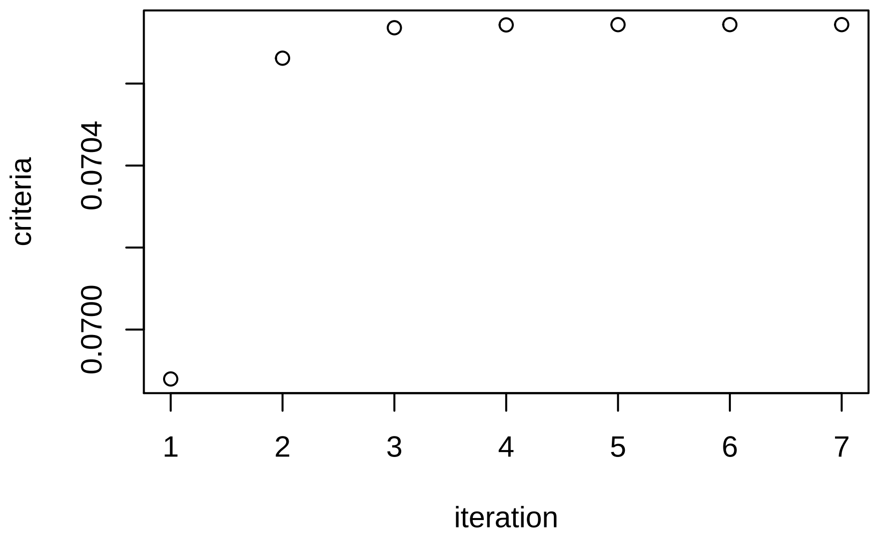
```
R> # Download the dataset's package at http://biodev.cea.fr/sgcca/.
R> # --> gliomaData_0.4.tar.gz
R>
R> require(gliomaData)

Loading required package: gliomaData

R> data(ge_cgh_locIGR)
R>
R> blocks <- ge_cgh_locIGR$multiblocks
R> Loc <- factor(ge_cgh_locIGR$y)
R> levels(Loc) <- colnames(ge_cgh_locIGR$multiblocks$y)
R> blocks[[3]] = Loc
R>
R> # check dimensions of the blocks
R> sapply(blocks, NCOL)

   GE   CGH      y
15702  1229      1
```

We impose $\mathbf{X}_1$ and $\mathbf{X}_2$ to be connected to $\mathbf{X}_3$. This design is commonly used in many applications and is oriented toward the prediction of the location. The argument `response=3` of the `rgcca()` function encodes this design.

```
R> fit.rgcca = rgcca(blocks = blocks, response = 3, ncomp = 2, verbose = TRUE)
```

```
Computation of the RGCCA block components based on the factorial scheme
Shrinkage intensity parameters are chosen manually
Computation of the RGCCA block components #1 is under progress...
 Iter:    1  Fit:  0.13465500  Dif:  0.12507270
 Iter:    2  Fit:  0.15313643  Dif:  0.01848143
 Iter:    3  Fit:  0.15844550  Dif:  0.00530907
 Iter:    4  Fit:  0.15963587  Dif:  0.00119037
 Iter:    5  Fit:  0.15988694  Dif:  0.00025107
 Iter:    6  Fit:  0.15993920  Dif:  0.00005226
 Iter:    7  Fit:  0.15995005  Dif:  0.00001085
 Iter:    8  Fit:  0.15995230  Dif:  0.00000225
 Iter:    9  Fit:  0.15995277  Dif:  0.00000047
 Iter:   10  Fit:  0.15995286  Dif:  0.00000010
 Iter:   11  Fit:  0.15995288  Dif:  0.00000002
 Iter:   12  Fit:  0.15995289  Dif:  0.00000000

The RGCCA algorithm converged to a stationary point after 11 iterations

Computation of the RGCCA block components #2 is under progress...
 Iter:    1  Fit:  0.06987951  Dif:  0.06715811
 Iter:    2  Fit:  0.07066188  Dif:  0.00078237
 Iter:    3  Fit:  0.07073615  Dif:  0.00007427
 Iter:    4  Fit:  0.07074309  Dif:  0.00000694
 Iter:    5  Fit:  0.07074374  Dif:  0.00000065
 Iter:    6  Fit:  0.07074380  Dif:  0.00000006
 Iter:    7  Fit:  0.07074380  Dif:  0.00000001

The RGCCA algorithm converged to a stationary point after 6 iterations
```





When the response variable is qualitative, two steps are implicitly performed: (i) disjunctive coding and (ii) the associated shrinkage parameter is set to 0 regardless of the value specified by the user.

```
R> fit.rgcca$call$connection

    GE CGH y
GE   0   0 1
CGH  0   0 1
y    1   1 0

R> fit.rgcca$call$tau

[1] 1 1 0
```

From the dimension of each block ($n > p$ or $n \leq p$), `rgcca()` selects automatically the dual formulation for $\mathbf{X}_1$ and $\mathbf{X}_2$ and the primal one for $\mathbf{X}_3$. The formulation used for each block is returned using the following command:

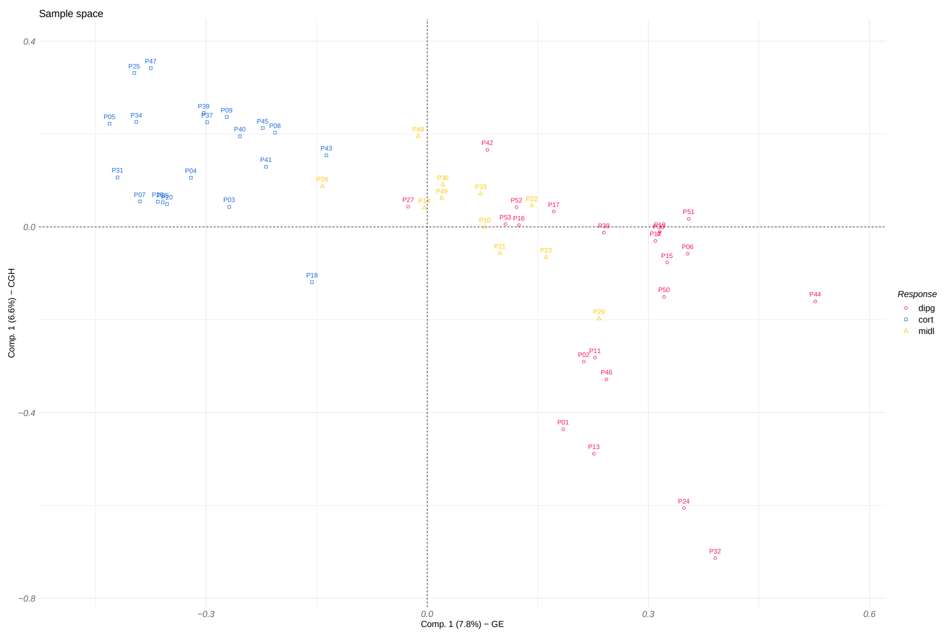
```
R> fit.rgcca$primal_dual

[1] "dual"   "dual"   "primal"
```

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

```
R> system.time(
+   rgcca(blocks = blocks, response = 3)
+ )

   user  system elapsed
   1.28    0.03    1.35
```

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.

```
R> plot(fit.rgcca, type = "sample", block=1:2,
+       comp = 1, response = Loc)
```



For easier interpretation of the results, especially in high-dimensional settings, it is often appropriate to add, within the RGCCA optimization problem, penalties promoting sparsity. For that purpose, an $\ell_1$ penalization on the weight vectors $\mathbf{a}_1, \dots, \mathbf{a}_J$ is applied. the `sparsity` argument of `rgcca()` varies between 1/sqrt(ncol) and 1 (larger values of `sparsity` correspond to less penalization) and control the amount of sparsity of the weight vectors $\mathbf{a}_1, \dots, \mathbf{a}_J$. If `sparsity` is a vector, $\ell_1$-penalties are the same for all the weights corresponding to the same block but different components:

$$\forall h, \|\mathbf{a}_j^{(h)}\|_{\ell_1} \le c_{1j}\sqrt{p_j}, \tag{22}$$

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

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

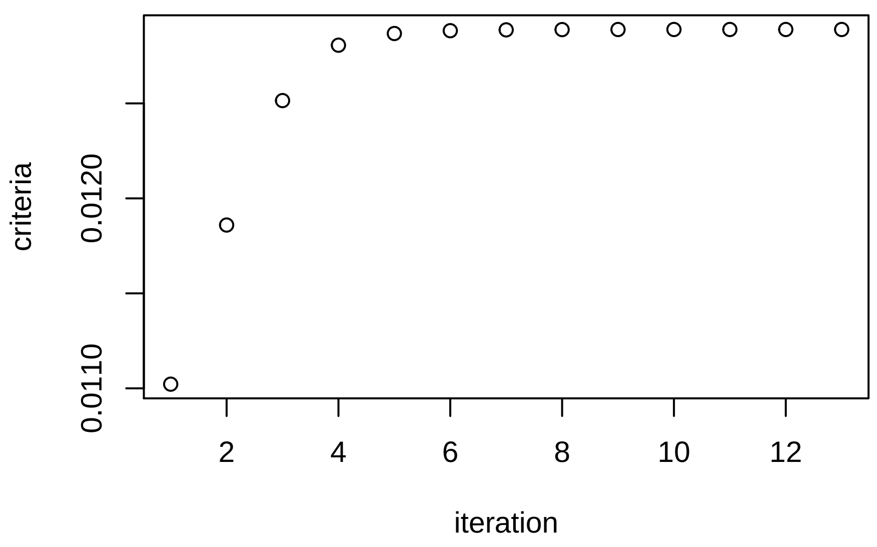
$$\forall h, \|\mathbf{a}_j^{(h)}\|_{\ell_1} \le c_1[h, j]\sqrt{p_j}. \tag{23}$$

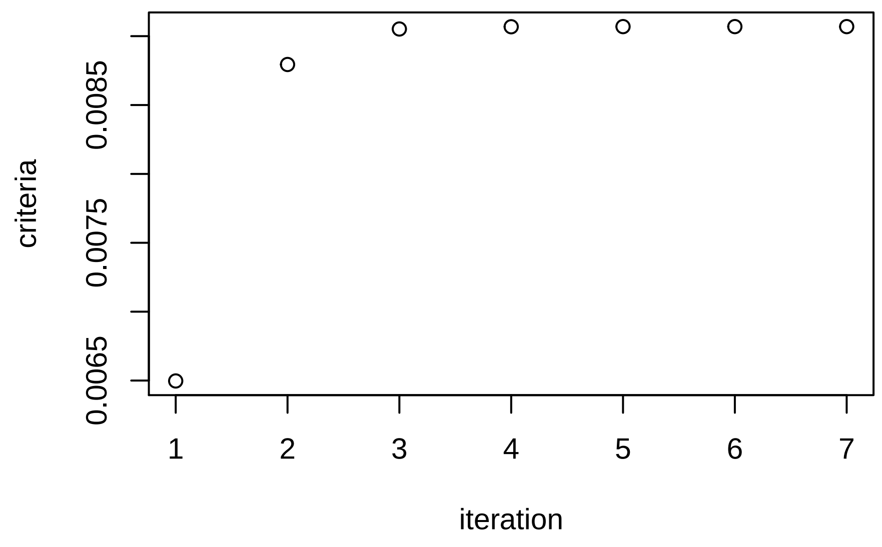
## 5.1. SGCCA for the Glioma dataset

The algorithm associated with the optimization problem (13) is available through the function `rgcca()` with the argument `method="sgcca"`.

```
R> fit.sgcca = rgcca(blocks = blocks, response = 3, ncomp = 2,
+                    sparsity = c(0.0710, 0.2000, 1),
+                    verbose = TRUE)
```

```
Computation of the SGCCA block components based on the factorial scheme
Computation of the SGCCA block components #1 is under progress...
 Iter:    1  Fit:  0.01102201  Dif:  0.01084380
 Iter:    2  Fit:  0.01185962  Dif:  0.00083761
 Iter:    3  Fit:  0.01251465  Dif:  0.00065502
 Iter:    4  Fit:  0.01280624  Dif:  0.00029159
 Iter:    5  Fit:  0.01286779  Dif:  0.00006155
 Iter:    6  Fit:  0.01288261  Dif:  0.00001483
 Iter:    7  Fit:  0.01288695  Dif:  0.00000434
 Iter:    8  Fit:  0.01288830  Dif:  0.00000135
 Iter:    9  Fit:  0.01288872  Dif:  0.00000043
 Iter:   10  Fit:  0.01288886  Dif:  0.00000014
 Iter:   11  Fit:  0.01288891  Dif:  0.00000004
 Iter:   12  Fit:  0.01288892  Dif:  0.00000001
 Iter:   13  Fit:  0.01288893  Dif:  0.00000000


The SGCCA algorithm converged to a stationary point after 12 iterations


Computation of the SGCCA block components #2 is under progress...
 Iter:    1  Fit:  0.00649703  Dif:  0.00645637
 Iter:    2  Fit:  0.00879433  Dif:  0.00229731
 Iter:    3  Fit:  0.00905191  Dif:  0.00025758
 Iter:    4  Fit:  0.00906832  Dif:  0.00001641
 Iter:    5  Fit:  0.00906923  Dif:  0.00000090
 Iter:    6  Fit:  0.00906928  Dif:  0.00000005
 Iter:    7  Fit:  0.00906928  Dif:  0.00000000


The SGCCA algorithm converged to a stationary point after 6 iterations
```





the `print()` function allows summarizing the SGCCA analysis.

```
R> print(fit.sgcca)

Call: method='sgcca', superblock=FALSE, scale=TRUE, scale_block='inertia',
init='svd', bias=TRUE, tol=1e-08, NA_method='nipals', ncomp=c(2,2,2),
response=3, comp_orth=TRUE
There are J = 3 blocks.
The design matrix is:
    GE CGH y
GE   0   0 1
CGH  0   0 1
y    1   1 0

The factorial scheme is used.
Sum_{j,k} c_jk g(cov(X_j a_j, X_k a_k) = 0.022

The sparsity parameter used for GE is: 0.071 (with 146, 145 variables selected)
The sparsity parameter used for CGH is: 0.2 (with 84, 76 variables selected)
The regularization parameter used for y is: 0
```

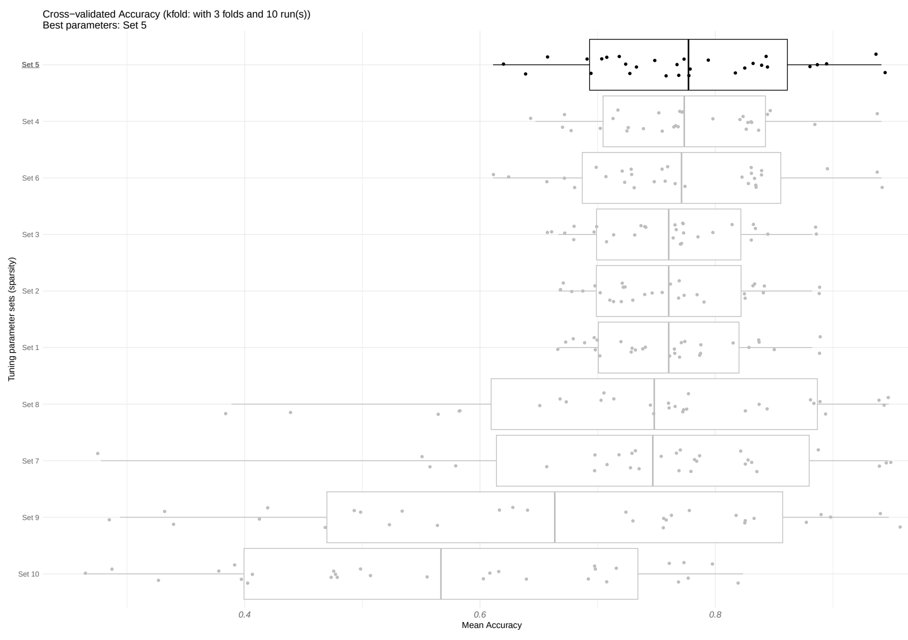
and the `plot()` returns the same graphical displays as RGCCA. We skip these representations for sake of brevity.

In this situation, the optimal sparsity parameters is usually chosen by cross-validation using the `rgcca_cv()` function. The goal is to maximize the cross-validated accuracy (`metric = Accuracy`) in a model where we try to predict the response block from all the block components with a user-defined classifier (`prediction_model`). also, we decide to upper bound the sparsity parameters for $X_1$ and $X_2$ to 0.2, to achieve an attractive amount of sparsity.

```
R> set.seed(27) #my favorite number
R> cv_out <- rgcca_cv(blocks, response = 3,
+                    par_type = "sparsity",
+                    par_value = c(.2, .2, 0),
+                    par_length = 10,
+                    prediction_model = "lda",
+                    validation = "kfold",
+                    k = 3, n_run = 10, metric = "Accuracy",
+                    n_cores = 15)
```

We can report the results of this cross-validation using the generic `print()` and `plot()` functions and build the optimal model using the fitted `cval` object as argument of `rgcca()`:

```
R> print(cv_out)

Call: method='sgcca', superblock=FALSE, scale=TRUE, scale_block=TRUE, init='svd',
bias=TRUE, tol=1e-08, NA_method='nipals', ncomp=c(1,1,1), response=3,
comp_orth=TRUE
There are J = 3 blocks.
The design matrix is:
    GE CGH y
GE   0   0 1
CGH  0   0 1
y    1   1 0

The factorial scheme is used.

Tuning parameters (sparsity) used:
      GE   CGH y
1  0.200 0.200 0
2  0.179 0.181 0
3  0.157 0.162 0
4  0.136 0.143 0
5  0.115 0.124 0
6  0.093 0.105 0
7  0.072 0.086 0
8  0.051 0.067 0
9  0.029 0.048 0
10 0.008 0.029 0

Validation: kfold with 3 folds and 10 run(s))
Prediction model: lda

   Tuning parameters Mean Accuracy     Sd
1              Set 1         0.760 0.0598
2              Set 2         0.760 0.0615
3              Set 3         0.760 0.0613
4              Set 4         0.774 0.0690
5              Set 5         0.777 0.0840
6              Set 6         0.771 0.0844
7              Set 7         0.747 0.1329
8              Set 8         0.748 0.1386
9              Set 9         0.664 0.1937
10            Set 10         0.567 0.1674

The best combination is: Set 5 for a mean Accuracy of 0.777.

R> plot(cv_out)
```



```
R> fit = rgcca(cv_out)
```

Notice that the sparsity parameter associated with $\mathbf{X}_3$ switches automatically to regularization parameter set to $\tau_3 = 0$. This choice is justified by the fact that 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.

Also it is possible to stabilize the selected variables using the `rgcca_stability()` function.

```
R> fit_stab = rgcca_stability(fit,
+                             keep = sapply(fit$a,
+                                           function(x) mean(x!=0)),
+                             n_boot = 100, verbose = TRUE, n_cores = 15)
```

and then apply the bootstrap procedure on the most stable variables.

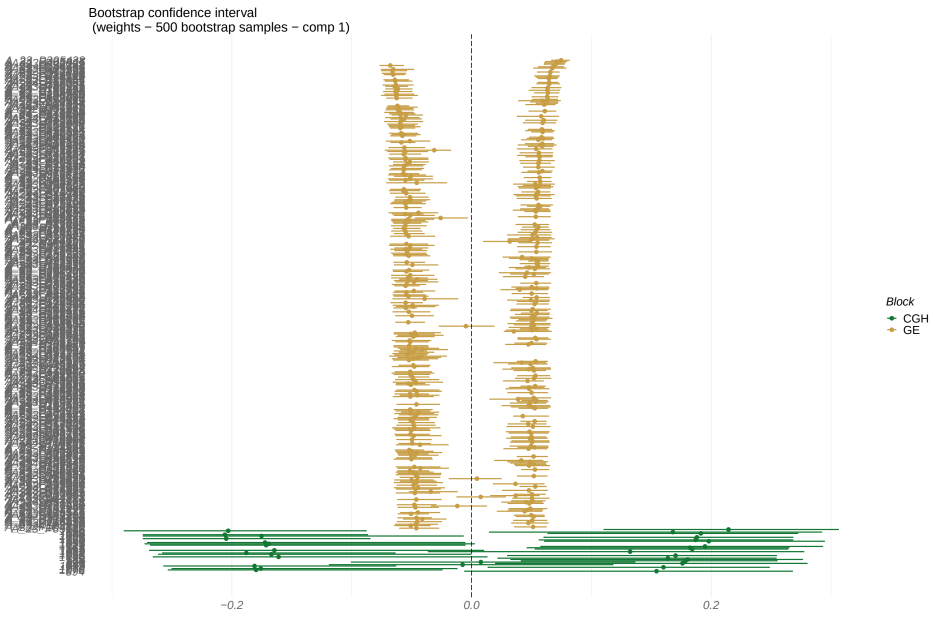
```
R> boot_out = rgcca_bootstrap(fit_stab, n_boot = 500)
```

```
All the parameters were imported from the fitted rgcca_stability object.
```

```
Bootstrap samples sanity check...OK
```

The bootstrap results can be visualized using the generic `plot()` function.

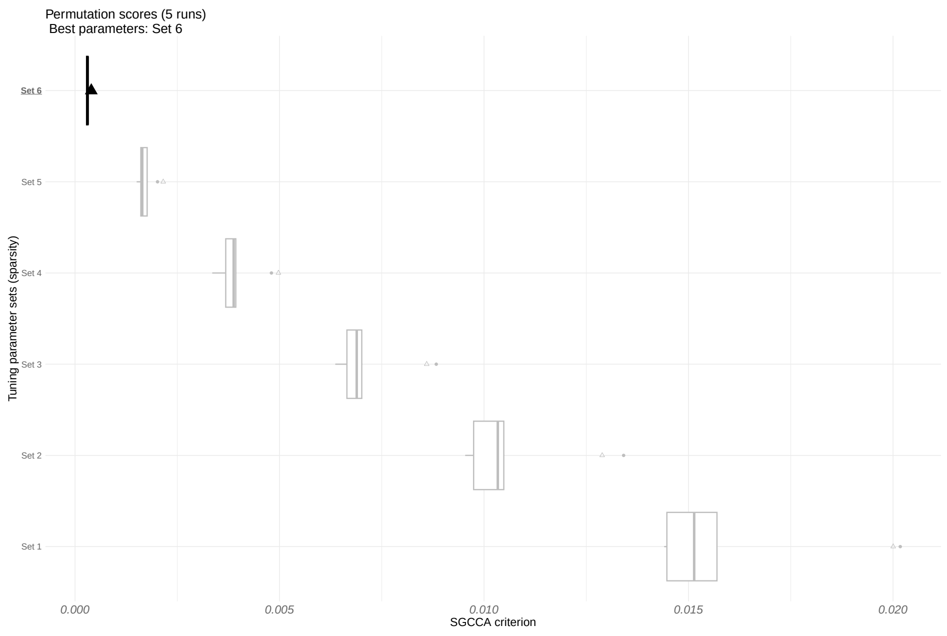
```
R> plot(boot_out, block = 1:2,
+       display_order = FALSE,
+       n_mark = 2000, cex = 1.3,
+       show_star = FALSE)
```



Of course, it is still possible to determine the optimal sparsity parameters by permutation. This is made possible by setting the `par_type` argument to `"sparsity"` (instead of `"tau"`) within the `rgcca_permutation()` function.

```
R> set.seed(123456) # -> very sparse model
R> perm_out = rgcca_permutation(blocks, connection = C, response = 3,
+                                  par_type = "sparsity",
+                                  par_value =
+                                    matrix(c(0.1, 0.25, 1,
+                                             0.0710, 0.2000, 1,
+                                             0.0552, 0.1571, 1,
+                                             0.0395, 0.1143, 1,
+                                             0.0237, 0.0714, 1,
+                                             0.0080, 0.029, 1), 6, 3,
+                                           byrow = TRUE),
+                                  n_perms = 5, n_cores = 10)
```

```
R> plot(perm_out, cex = 1.3)
```



and then directly used the optimal sparsity parameters as previously:

```
R> rgcca_opt = rgcca(perm_out)
```

```
R> fit_stab = rgcca_stability(rgcca_opt,
+                             keep = sapply(rgcca_opt$a,
+                                           function(x) mean(x!=0)),
+                             n_boot = 100, verbose = TRUE, n_cores = 15)
```

```
Bootstrap samples sanity check...OK
```

and then apply the bootstrap procedure on the most stable variables.

```
R> boot_out = rgcca_bootstrap(fit_stab, n_boot = 500)
```

```
All the parameters were imported from the fitted rgcca_stability object.
```

```
Bootstrap samples sanity check...OK
```

The bootstrap results can be visualized using the generic `plot()` function.

```
R> plot(boot_out, block = 1:2,
+       display_order = FALSE,
+       n_mark = 2000, cex = 1.3)
```

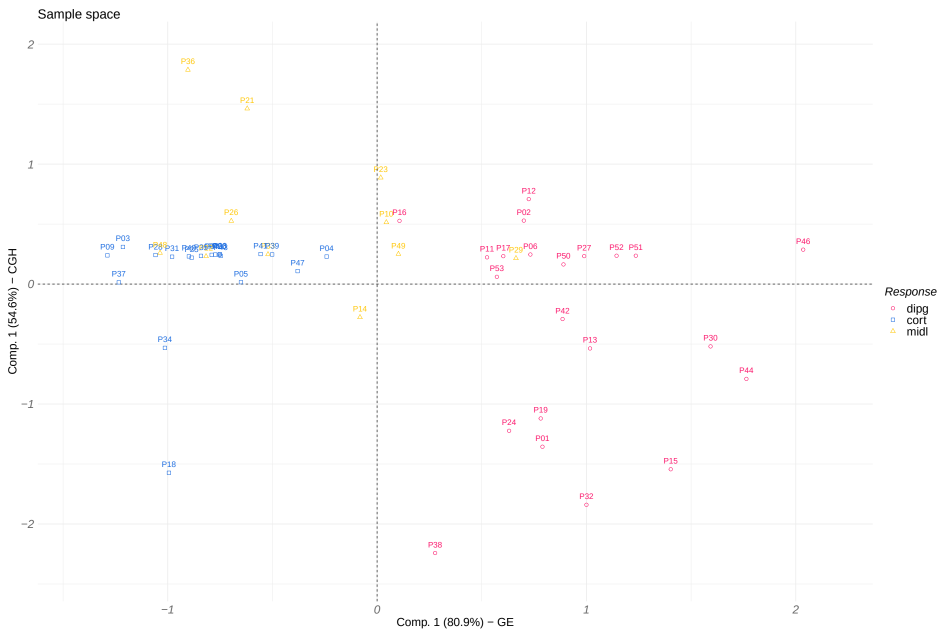
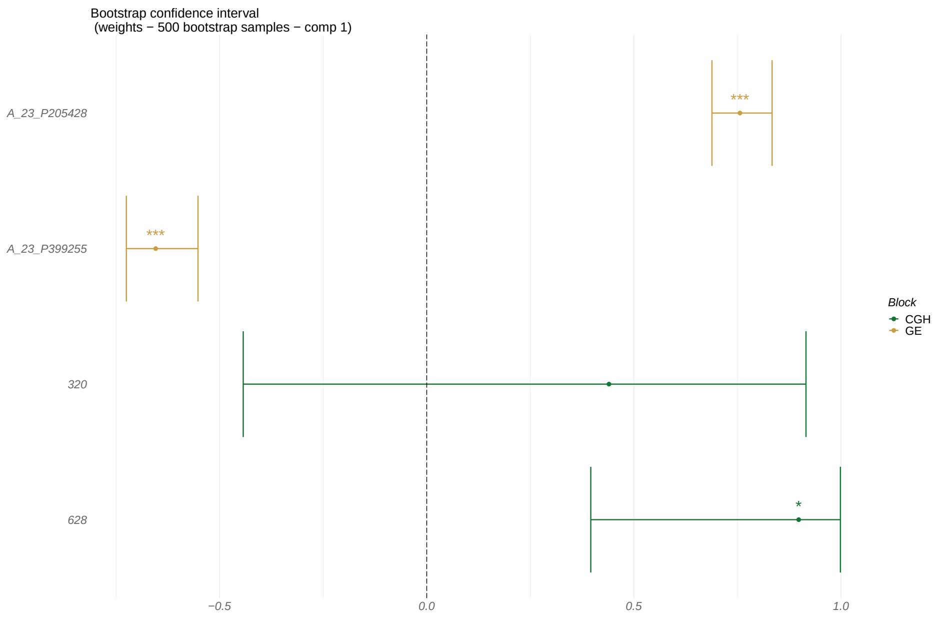


Figure 7: graphical display of the tumors obtained by crossing GE1 and CGH1 and labeled according to their location.



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.

```
R> plot(fit_stab, type = "sample", block=1:2,
+       comp=1, resp = as.character(Loc),
+       cex = 1.3
+       )
```

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

## 6. Conclusion

The RGCCA framework gathers sixty years of multiblock component methods and offers a unified implementation strategy for these methods. The RGCCA package is available on the Comprehensive R Archive Network (CRAN) and on github `https://github.com/rgcca-factory/RGCCA`. This release of the RGCCA package includes:

- Several strategies for determining automatically the shrinkage parameters/level of sparsity: Schaffer & Strimmer's analytical formulae, cross-validation or permutation strategy.
- A bootstrap resampling procedure for assessing the reliability of the parameters estimates of S/RGCCA.
- Dedicated functions for graphical displays of the output of RGCCA (sample plot, correlation circle, biplot, ...).
- 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 [@Tenenhaus2005], page 171).
- Special attention has been paid to recover the results of other R packages of the literature including ‘ade4‘ and ‘factomineR‘.

The RGCCA framework is constantly evolving and extended. Indeed, we propose RGCCA for multigroup data (Tenenhaus *et al.* 2014) and RGCCA for multiway data (Gloaguen, Philippe, Frouin, Gennari, Dehaene-Lambertz, Le Brusquet, and Tenenhaus 2020, Girka, Gloaguen, Brusquet, Zujovic, and Tenenhaus (2023)). In addition, maximizing successive criteria may be seen as suboptimal from an optimization point of view where a single global criterion might be preferred. A global version of RGCCA (Gloaguen 2020) which allows extracting simultaneously several components per block (no deflation procedure required) has been proposed. Work in progress includes the integration of this novel approaches in the next release of the package.

## References

Borga M, Landelius T, Knutsson H (1997). “A Unified Approach to PCA, PLS, MLR and CCA.”

Bougeard S, Hanafi M, Qannari E (2008). “Continuum redundancy-PLS regression: a simple continuum approach.” *Computational Statistics and Data Analysis*, **52**, 3686–3696.

Boyd S, Dattorro J, *et al.* (2003). “Alternating projections.” *EE392o, Stanford University*.

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

Cantini L, Zakeri P, Hernandez C, Naldi A, Thieffry D, Remy E, Baudot A (2021). "Benchmarking joint multi-omics dimensionality reduction approaches for the study of cancer." *Nature communications*, **12**(1), 124.

Carroll J (1968a). "A generalization of canonical correlation analysis to three or more sets of variables." In *Proceeding 76th Conv. Am. Psych. Assoc.*, pp. 227–228.

Carroll J (1968b). "Equations and tables for a generalization of canonical correlation analysis to three or more sets of variables." *Unpublished companion paper to Carroll, JD*.

Chen Y, Wiesel A, Hero A (2011). "Robust shrinkage estimation of high dimensional covariance matrices." *IEEE Transactions on Signal Processing*, **59 (9)**, 4097–4107.

Chessel D, Hanafi M (1996). "Analyse de la co-inertie de K nuages de points." *Revue de Statistique Appliquée*, **44**, 35–60.

De Leeuw J (1994). "Block relaxation algorithms in statistics." In HH Bock, L W, MM Richter (eds.), *Information Systems and Data Analysis*, pp. 308–325. Springer, Berlin.

Escofier B, Pages J (1994). "Multiple factor analysis (AFMULT package)." *Computational statistics & data analysis*, **18**(1), 121–140.

Garali I, Adanyeguh I, Ichou F, Perlbarg V, Seyer A, Colsch B, Moszer I, Guillemot V, Durr A, Mochel F, Tenenhaus A (2018). "A strategy for multimodal data integration: application to biomarkers identification in spinocerebellar ataxia." *Briefings in bioinformatics*, **19**(6), 1356–1369.

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

Girka F, Gloaguen A, Brusquet LL, Zujovic V, Tenenhaus A (2023). "Tensor Generalized Canonical Correlation Analysis." *arXiv preprint arXiv:2302.05277*.

Gloaguen A (2020). *A statistical and computational framework for multiblock and multiway data analysis*. Ph.D. thesis, Université Paris-Saclay.

Gloaguen A, Guillemot V, Tenenhaus A (2017). "An efficient algorithm to satisfy l1 and l2 constraints." In *49èmes Journées de Statistique*.

Gloaguen A, Philippe C, Frouin V, Gennari G, Dehaene-Lambertz G, Le Brusquet L, Tenenhaus A (2020). "Multiway generalized canonical correlation analysis." *Biostatistics*, **23**(1), 240–256. ISSN 1465-4644. doi:10.1093/biostatistics/kxaa010. https://academic.oup.com/biostatistics/article-pdf/23/1/240/42208904/kxaa010_supplementary_data.pdf, URL https://doi.org/10.1093/biostatistics/kxaa010.

Hanafi M (2007). "PLS Path modelling: computation of latent variables with the estimation mode B." *Computational Statistics*, **22**, 275–292.

Hanafi M, Kiers H (2006). "Analysis of K sets of data, with differential emphasis on agreement between and within sets." *Computational Statistics and Data Analysis*, **51**, 1491–1508.

Hanafi M, Kohler A, Qannari EM (2010). "Shedding new light on hierarchical principal component analysis." *Journal of Chemometrics*, **24**(11-12), 703–709.

Hanafi M, Kohler A, Qannari EM (2011). "Connections between multiple co-inertia analysis and consensus principal component analysis." *Chemometrics and intelligent laboratory systems*, **106**(1), 37–40.

Horst P (1961). "Relations among m sets of variables." *Psychometrika*, **26**, 126–149.

Hotelling H (1933). "Analysis of a complex of statistical variables into principal components." *Journal of Educational Psychology*, **24**, 417–441.

Hotelling H (1936). "Relation Between Two Sets of Variates." *Biometrika*, **28**, 321–377.

Kettenring J (1971). "Canonical analysis of several sets of variables." *Biometrika*, **58**, 433–451.

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

Ledoit O, Wolf M (2004). "A well conditioned estimator for large-dimensional covariance matrices." *Journal of Multivariate Analysis*, **88**, 365–411.

Leurgans S, Moyeed R, Silverman B (1993). "Canonical correlation analysis when the data are curves." *Journal of the Royal Statistical Society. Series B*, **55**, 725–740.

Puget S, Philippe C, Bax DA, Job B, Varlet P, Junier MP, Andreiuolo F, Carvalho D, Reis R, Guerrini-Rousseau L, Roujeau T, Dessen P, Richon C, Lazar V, Le Teuff G, Sainte-Rose C, Geoerger B, Vassal G, Jones C, Grill J (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, Hanafi M (2005). "A simple continuum regression approach." *Journal of Chemometrics*, **19**, 387–392.

Russett B (1964). "Inequality and Instability: The Relation of Land Tenure to Politics." *World Politics*, **16:3**, 442–454.

Schäfer J, Strimmer K (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, Cristianini N (2004). *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA.

Smilde AK, Westerhuis JA, de Jong S (2003). "A framework for sequential multiblock component methods." *Journal of chemometrics*, **17**(6), 323–337.

Takane Y, Hwang H (2007). "Regularized linear and kernel redundancy analysis." *Computational Statistics and Data Analysis*, **52**, 394–405.

ten Berge JM (1988). "Generalized approaches to the MAXBET problem and the MAXDIFF problem, with applications to canonical correlations." *Psychometrika*, **53**(4), 487–494.

Tenenhaus A, Philippe C, Frouin V (2015). "Kernel Generalized Canonical Correlation Analysis." *Computational Statistics & Data Analysis*, **90**, 114–131.

Tenenhaus A, Philippe C, Guillemot V, Lê Cao KA, Grill J, Frouin V (2014). "Variable Selection for Generalized Canonical Correlation Analysis." *Biostatistics*, **15(3)**, 569–583.

Tenenhaus A, Tenenhaus M (2011). "Regularized Generalized Canonical Correlation Analysis." *Psychometrika*, **76**, 257–284.

Tenenhaus A, Tenenhaus M (2014). "Regularized Generalized Canonical Correlation Analysis for multiblock or multigroup data analysis." *European Journal of Operational Research*, **238**, 391–403.

Tenenhaus M, Tenenhaus A, Groenen P (2017). "Regularized generalized canonical correlation analysis: A framework for sequential multiblock component methods." *Psychometrika*, **82**(3), 737–777.

Tucker L (1958). "An inter-battery method of factor analysis." *Psychometrika*, **23**, 111–136.

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

Van den Wollenberg A (1977). "Redudancy analysis: an alternative for canonical correlation analysis." *Psychometrika*, **42**, 207–219.

Vinod H (1976). "Canonical ridge and econometrics of joint production." *Journal of Econometrics*, **4**, 147–166.

Westerhuis J, Kourti T, MacGregor J (1998). "Analysis of multiblock and hierarchical PCA and PLS models." *Journal of Chemometrics*, **12**, 301–321.

Witten D, Tibshirani R, Hastie T (2009). "A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis." *Biostatistics*, **10**(3), 515–534.

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*, pp. 1–54.

Wold H (1985). "Partial Least Squares." In *Encyclopedia of Statistical Sciences, vol. 6, Kotz, S and Johnson, N.L. (Eds), John Wiley and Sons, New York*, pp. 581–591.

Wold S, Martens H, Wold H (1983). "The multivariate calibration problem in chemistry solved by the PLS method." In *In Proc. Conf. Matrix Pencils, Ruhe A. and Kastrom B. (Eds), March 1982, Lecture Notes in Mathematics, Springer Verlag, Heidelberg*, pp. 286–293.

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**, 463–482.

**Affiliation:**

FirstName LastName
University/Company
First line
Second line
E-mail: name@company.com
URL: http://rstudio.com

Third Author
Universitat Autònoma de Barcelona
Department of Statistics and Mathematics,
Faculty of Biosciences,
Universitat Autònoma de Barcelona

*Journal of Statistical Software* http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd