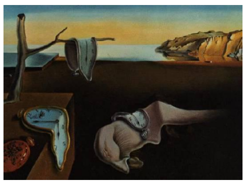


Journal of Statistical Software

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.

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 versatile statistical framework for data integration.

1.1. Optimization background

The RGCCA package relies on a master algorithm for maximizing a continuously differentiable multi-convex function $f(\mathbf{a}_1, \ldots, \mathbf{a}_J) : \mathbb{R}^{J_1} \times \ldots \times \mathbb{R}^{J_J} \rightarrow \mathbb{R}$ (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 $\Omega_j \subset \mathbb{R}^{J_j}$. 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 $J = \sum_{j=1}^J J_j$. Moreover, for the vertical concatenation of column vectors, the notation $\mathbf{a} = (\mathbf{a}_1;\ldots;\mathbf{a}_J)$ is preferred for the sake of simplification. This last formulation is also used to define a vertical concatenation of matrices. These notations are used all along this manuscript.

1.2. Algorithm

A simple, monotonically and globally convergent algorithm is presented for maximizing (73). An algorithm is globally convergent if, regardless of its initialization, it converges towards a stationary point. For an unconstrained optimization problem with a continuously differentiable objective function, a stationary point is a point where the derivative of the objective function is null. For a constrained optimization problem, a stationary point is a point where the derivative of the Lagrangian function associated with the problem is null. For such a point, the derivative of the objective function lies in the subspace defined by the derivative of each constraint. This condition is called the Karush-Kuhn-Tucker (KKT) condition.

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 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:

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

On the right-hand side of the inequality (74), 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 $\ell_j(\tilde{\mathbf{a}}_j,\mathbf{a})$ 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 2.

To study the convergence properties of Algorithm 2, we introduce some notations: $\Omega = \Omega_1\times\ldots\times\Omega_J$, $\mathbf{a} = (\mathbf{a}_1;\ldots;\mathbf{a}_L) \in \Omega$, $c_l\ :\ \Omega \mapsto \Omega$ is an operator defined as $c_l(\mathbf{a}) = (\mathbf{a}_1;\ldots;\mathbf{a}_{j-1};r_j(\mathbf{a});\mathbf{a}_{j+1};\ldots;\mathbf{a}_J)$ with $r_l(\mathbf{a})$ introduced in equation (75) and $c\ :\ \Omega \mapsto \Omega$ is defined as $c = c_L \circ c_{L-1} \circ ... \circ c_1$, where $\circ$ stands for the function composition operator. We consider the sequence $\{\mathbf{a}^s = (\mathbf{a}_1^s;\ldots;\mathbf{a}_J^s)\}$ generated by Algorithm 2. Using the operator c, the «for loop» inside Algorithm 2 can be replaced by

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

```
1: Result: $\mathbf{a}_1^s, \ldots, \mathbf{a}_J^s$ (approximate solution of (??) subject to (73))
2: Initialization: choose random vector $\mathbf{a}_j^0 \in \Omega_j, j = 1, \ldots, J, \epsilon$;
3: $s = 0$ ;
4: repeat
5:   for $j = 1$ to $J$ do
6:       $\mathbf{a}_j^{s+1} = r_j \left(\mathbf{a}_1^{s+1}, \ldots, \mathbf{a}_{j-1}^{s+1}, \mathbf{a}_j^s, \ldots, \mathbf{a}_J^s\right).$   (4)
7:   end for
8:   $s = s + 1$ ;
9: until $f(\mathbf{a}_1^{s+1}, \ldots, \mathbf{a}_J^{s+1}) - f(\mathbf{a}_1^s, \ldots, \mathbf{a}_J^s) < \varepsilon$
```

the following recurrence relation: $\mathbf{a}^{s+1} = c(\mathbf{a}^s)$. The convergence properties of Algorithm 2 are summarized in the following proposition:

Proposition 1.1. *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 Ω. This result implies the monotonic convergence of Algorithm 2.

(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 (??).

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

(d) $\lim\limits_{s \to \infty} f(\mathbf{a}^s) = f(\mathbf{v}^\star)$, where $\mathbf{a}^\star$ is a fixed point of c.

(e) The sequence $\{\mathbf{a}^s = (\mathbf{a}_1^s; \ldots; \mathbf{a}_J^s)\}$, $l = 1, \ldots, L$, is asymptotically regular: $\lim\limits_{s \to \infty} \sum_{l=1}^{L} \|\mathbf{a}_j^{s+1} - \mathbf{a}_j^s\| = 0$. This result implies that if the threshold ε for the stopping criterion in Algorithm 2 is made sufficiently small, the output of Algorithm 2 will be as close as wanted to a stationary point of (??).

(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.

The goal is to demonstrate Proposition 1.5 that gathers all the convergence properties of Algorithm 2. For this purpose, the results given in the following lemma are useful.

Lemma 1.2. *Consider the set Ω, the function $f : \Omega \mapsto \mathbb{R}$ and the operator $c : \Omega \mapsto \Omega$ defined above. Then, the following properties hold:*

(i) Ω is a compact set;

(ii) c is a continuous operator;

(iii) $f(\mathbf{a}) \leq f(c(\mathbf{a}))$ for any $\mathbf{a} \in \Omega$;

(iv) If $f(\mathbf{a}) = f(c(\mathbf{a}))$, then $c(\mathbf{a}) = \mathbf{a}$.

Proof of Lemma 1.6. Point *(i)* In this section, $\forall l$, Ω_j are assumed to be compact. As the Cartesian product of L compact sets is compact, $\Omega = \Omega_1 \times \ldots \times \Omega_J$ is compact.

Point *(ii)*

We assume that $r_l(\mathbf{a})$ defined in equation (75) exists and is unique. As Ω_j is a compact set and l_l defined in equation (74) is a real-valued continuous function, Berge's maximum theorem applies and guarantees that the maximizer $r_l(\mathbf{a})$ of $l_l(\tilde{\mathbf{a}}_l, \mathbf{a})$ is continuous on Ω_j (**?**). This implies that $c_l : \Omega \to \Omega$ is a continuous operator and that $c = c_L \circ c_{L-1} \circ \ldots \circ c_1$ is also continuous as composition of L continuous operators.

Point *(iii)* According to equation (74) based on multi-convexity of f and equation (75) that sets the definition of $r_l : \Omega \mapsto \Omega_j$, we know that:

$$f(\mathbf{a}) = \ell_l(\mathbf{a}_j, \mathbf{a}) \le \ell_l(r_l(\mathbf{a}), \mathbf{a}) \le f(\mathbf{a}_1, \ldots, \mathbf{a}_{l-1}, r_l(\mathbf{a}), \mathbf{a}_{l+1}, \ldots, \mathbf{a}_J) = f(c_l(\mathbf{a})). \tag{5}$$

This implies that updating $\mathbf{a}_j$ by $\hat{\mathbf{a}}_l = r_l(\mathbf{a})$ increases $f(\mathbf{a})$, or $f(\mathbf{a})$ stays the same. Moreover, the following inequality is deduced from (77) for each $l = 2, \ldots, L$:

$$f(c_{l-1} \circ \ldots \circ c_1(\mathbf{a})) \le f(c_l \circ c_{l-1} \circ \ldots \circ c_1(\mathbf{a})). \tag{6}$$

This yields the desired inequalities for any $\mathbf{a} \in \Omega$:

$$f(\mathbf{a}) \le f(c_1(\mathbf{a})) \le f(c_2 \circ c_1(\mathbf{a})) \le \ldots \le f(c_L \circ \ldots \circ c_1(\mathbf{a})) = f(c(\mathbf{a})). \tag{7}$$

Point *(iv)* If $f(\mathbf{a}) = f(c(\mathbf{a}))$ for $\mathbf{a} \in \Omega$ then equation (79) implies

$$f(\mathbf{a}) = f(c_1(\mathbf{a})) = f(c_2 \circ c_1(\mathbf{a})) = \ldots = f(c_L \circ \ldots \circ c_1(\mathbf{a})) = f(c(\mathbf{a})). \tag{8}$$

Using equation (77), the equality $f(\mathbf{a}) = f(c_1(\mathbf{a}))$ implies $\ell_1(\mathbf{a}_1, \mathbf{a}) = \ell_1(r_1(\mathbf{a}), \mathbf{a})$ and therefore $\mathbf{a}_1 = r_1(\mathbf{a})$ as $r_1(\mathbf{a})$ is the unique maximizer of $\ell_1(r_1(\tilde{\mathbf{a}}_1), \mathbf{a})$ with respect to $\tilde{\mathbf{a}}_1 \in \Omega_1$. From this result, we deduce $\mathbf{a} = (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_J) = (r_1(\mathbf{a}), \mathbf{a}_2, \ldots, \mathbf{a}_J) = c_1(\mathbf{a})$ and then, by transitivity,

$$\mathbf{a} = c_1(\mathbf{a}) = c_2 \circ c_1(\mathbf{a}) = \ldots = c_L \circ \ldots \circ c_1(\mathbf{a}) = c(\mathbf{a}). \tag{9}$$

□

Proof of Proposition 1.5. Point *(a)* Point (iii) of Lemma 1.6 implies that the sequence $f(\mathbf{a}^s)$ is monotonically increasing, and therefore, convergent as the continuous function f is bounded on the compact set Ω.

Point *(b)* If the infinite sequence $f(\mathbf{a}^s)$ has a finite number of distinct terms, it cannot be a strictly increasing sequence and consequently there exists some integer M such that $f(\mathbf{a}^0) < f(\mathbf{a}^1) < \ldots < f(\mathbf{a}^M) = (\mathbf{a}^{M+1})$. Then, Point (iv) of Lemma 1.6 implies that $\mathbf{a}^M$ is a fixed point of c.

Point *(c) to (f)* They are deduced from a direct application of Meyer's monotone convergence theorem (Theorem 3.1 in (Meyer 1976)). This theorem gives quite general conditions under which a sequence $(\mathbf{a}^s)$ produced by an algorithm that monotonically increases a continuous objective function will converge. Meyer considered the case of a point-to-set operator $c : \Omega \mapsto \mathcal{P}(\Omega)$, where $\mathcal{P}(\Omega)$ is the set of all nonempty subsets of Ω. In this manuscript, c is a point-to-point operator and the conditions of Meyer's theorem reduce to the three following conditions (see (**?**)): (1) c is a continuous operator; (2) c is strictly monotone (increasing) with respect to f ; and (3) c is uniformly compact on Ω. Condition (2) means that points (iii) and (iv) of Lemma 1.6 are verified. Condition (3) means that there exists a compact set $\mathcal{K}$ such that $c(\mathbf{a}) \in \mathcal{K}$ for all $\mathbf{a} \in \Omega$. According to Lemma 1.6, these three conditions are satisfied for Algorithm 2 and therefore, Meyer's theorem can be applied to any sequence $\mathbf{a}^s$ produced by the recurrence equation $\mathbf{a}^{s+1} = c(\mathbf{a}^s)$ with $\mathbf{a}^0 \in \Omega$.

□

A specific instantiation of this quite general optimization problem has been first introduced in Tenenhaus, Tenenhaus, and Groenen (2017).

Let $\mathbf{X}_1, \ldots, \mathbf{X}_l, \ldots, \mathbf{X}_L$ be a collection of L data matrices. Each $I \times J_l$ data matrix $\mathbf{X}_l = [\mathbf{x}_{l1}, \ldots, \mathbf{x}_{lJ_l}]$ is a block and represents a set of J_l variables observed on I 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 the RGCCA optimization problem (Tenenhaus *et al.* 2017) is:

$$\begin{gathered}\max_{\mathbf{w}_1,\ldots,\mathbf{w}_L} \sum_{k,l=1}^{L} c_{kl}\ \mathrm{g}\left(I^{-1}\mathbf{w}_k^\top \mathbf{X}_k^\top \mathbf{X}_l \mathbf{w}_l\right) \\ \text{s.t. } \mathbf{w}_l^\top \mathbf{M}_l \mathbf{w}_l = 1,\ l = 1, \ldots, L\end{gathered} \tag{10}$$

where g, $\mathbf{C} \in \mathbb{R}^{L\times L}$ and $\mathbf{M}_l \in \mathbb{R}^{J_l \times J_l}$, $l = 1, \ldots L$ are defined in Chapter **??**, section **??**. The optimization problem (82) can be simplified by considering the two following transforms $\mathbf{P}_l = I^{-1/2}\mathbf{X}_l\mathbf{M}_l^{-1/2}$ and $\mathbf{a}_j = \mathbf{M}_l^{1/2}\mathbf{w}_l$, which leads to:

$$\max_{\mathbf{a}_1,\ldots,\mathbf{a}_J} f(\mathbf{a}_1, \ldots, \mathbf{a}_J) = \sum_{k,l=1}^{L} c_{lk}\ \mathrm{g}\left(\mathbf{a}_k^\top \mathbf{P}_k^\top \mathbf{P}_l \mathbf{a}_j\right) \tag{11}$$

$$\text{s.t. } \mathbf{a}_j^\top \mathbf{a}_j = 1, l = 1, \ldots, L. \tag{12}$$

We consider J matrices $\mathbf{Q}_1, \ldots, \mathbf{Q}_J$. Each matrix $\mathbf{Q}_j$ is of dimension $m \times p_j$. We also associate to each matrix $\mathbf{Q}_j$ a weight column vector $\mathbf{w}_j$ of dimension p_j and a symmetric definite positive matrix $\mathbf{M}_j$ of dimensions $p_j \times p_j$. Moreover, let $\mathbf{C}$ be a $J \times J$ symmetric design matrix of non-negative entries that indicate the strength of the relations between the pairs of matrices. The core optimization problem considered in this work is defined by

$$\underset{\mathbf{w}_1,\ldots,\mathbf{w}_J}{\text{maximize}} \sum_{j,k=1}^{J} c_{jk} g(\mathbf{w}_j^\top \mathbf{Q}_j^\top \mathbf{Q}_k \mathbf{w}_k) \text{ s.t. } \mathbf{w}_j^\top \mathbf{M}_j \mathbf{w}_j = 1, j = 1, \ldots, J \tag{13}$$

with the function $g(x)$ a convex function of the scalar $x \in \mathbb{R}$.

By setting $\mathbf{a}_j = \mathbf{M}_j^{1/2}\mathbf{w}_j$ and $\mathbf{P}_j = \mathbf{Q}_j\mathbf{M}_j^{-1/2}$, the optimization problem (13) becomes

$$\begin{gathered}\underset{\mathbf{a}\in\Omega}{\text{maximize}}\ f(\mathbf{a}) = \underset{\mathbf{a}_1,\ldots,\mathbf{a}_J}{\text{maximize}} f(\mathbf{a}_1, \ldots, \mathbf{a}_J) = \sum_{j,k=1}^{J} c_{jk} g(\mathbf{a}_j^\top \mathbf{P}_j^\top \mathbf{P}_k \mathbf{a}_k) \\ \text{subject to } \mathbf{a}_j \in \Omega_j \text{ for all } j = 1, \ldots, J\end{gathered} \tag{14}$$

with $\Omega_j = \{\mathbf{a}_j \in \mathbb{R}^{p_j}\ ;\ \mathbf{a}_j^\top \mathbf{a}_j = 1\}$, Ω equals the Cartesian product of the sets Ω_j ($\Omega = \Omega_1 \otimes \Omega_2 \otimes \ldots \otimes \Omega_J$) and $\mathbf{a} = (\mathbf{a}_1^\top, \mathbf{a}_2^\top, \ldots, \mathbf{a}_J^\top)^\top$.

An efficient algorithm is proposed for solving the optimization problem (14). This algorithm is presented in detail in (Groenen and Tenenhaus 2015).

Cancelling the partial subgradients of the Lagrangian function associated with optimization problem (14) with respect to $\mathbf{a}_j$ yields the following stationary equations:

$$\mathbf{a}_j \propto \sum_{k=1}^{J} c_{jk} g'\left(\mathbf{a}_j^\top \mathbf{P}_j^\top \mathbf{P}_k \mathbf{a}_k\right) \mathbf{P}_j^\top \mathbf{P}_k \mathbf{a}_k, j = 1, \ldots, J \tag{15}$$

where $g'\left(\mathbf{a}_j^\top \mathbf{P}_j^\top \mathbf{P}_k \mathbf{a}_k\right)$ denotes a subgradient of $g(x)$ at $x = \mathbf{a}_j^\top \mathbf{P}_j^\top \mathbf{P}_k \mathbf{a}_k$ and $\propto$ means that the left term is the normalized right term. These stationnary equations can be solved using a strategy to be described in the next section.

Two general optimization principles constitute the internal mechanism for the maximization of the optimization problem (14), that is, block relaxation (De Leeuw 1994) and maximization by minorization (MM) (Lange 2010; Hunter and Lange 2004). The former maximizes the function over different parameter vectors (i.e., $\mathbf{a}_1, \ldots, \mathbf{a}_J$), and this is done by updating each of the parameter vectors in turn, keeping the others fixed (Gauss-Seidel update rule). Maximizing (14) only over $\mathbf{a}_j$ amounts to maximizing $f(\mathbf{a}_1^{s+1}, \ldots, \mathbf{a}_{j-1}^{s+1}, \mathbf{a}_j, \mathbf{a}_{j+1}^s, \ldots, \mathbf{a}_J^s)$, or equivalently maximizing

$$f_j(\mathbf{a}_j, \mathbf{a}_{-j}^s) = \sum_{k=1}^{j-1} c_{jk} g(\mathbf{a}_j^\top \mathbf{P}_j^\top \mathbf{P}_k \mathbf{a}_k^{s+1}) + \sum_{k=j}^{J} c_{jk} g(\mathbf{a}_j^\top \mathbf{P}_j^\top \mathbf{P}_k \mathbf{a}_k^s) \tag{16}$$

where $\mathbf{a}_{-j}^s = (\mathbf{a}_1^{s+1}, \ldots, \mathbf{a}_{j-1}^{s+1}, \mathbf{a}_{j+1}^s, \ldots, \mathbf{a}_J^s)$ and $\mathbf{a}_k^s$ denotes a known weights of $\mathbf{a}_k$ obtained at iteration s. Since f_j is convex (as sum of convex functions), the heart of the algorithm is to maximize the convex function f_j over the compact set Ω_j. It is not necessary to assume that f_j is differentiable and we will denote ${f_j}'(\mathbf{a}_j, \mathbf{a}_{-j}^s)$ any subgradient of function f_j at $\mathbf{a}_j$. Then, if the update $\mathbf{a}_j^{s+1}$ improves the function value, so that $f_j(\mathbf{a}_j^{s+1}, \mathbf{a}_{-j}^s) \geq f_j(\mathbf{a}_j^s, \mathbf{a}_{-j}^s)$, the objective function in (14) and consequently in (13) will be improved over the complete set of parameter vectors. This principle is called block relaxation by (De Leeuw 1994). Algorithm 1 describes the internal mechanism of a classical block relaxation algorithm.

The critical step in Algorithm **??** is updating $f_j(\mathbf{a}_j, \mathbf{a}_{-j}^s)$ and relies on the principle underlying MM-algorithms (minimization by majorization or maximization by minorization). Note that this method has been available in the literature for some time (Voss and Eckhardt 1980) and rediscovered independently under the name majorization by (De Leeuw 1977). Let $f_j'(\mathbf{a}_j^s, \mathbf{a}_{-j}^s)$ denote any subgradient of f_j at $\mathbf{a}_j^s$, that is,

$$f_j'(\mathbf{a}_j^s, \mathbf{a}_{-j}^s) = \sum_{k=1}^{j-1} c_{jk} g'(\mathbf{a}_j^{s\top} \mathbf{P}_j^\top \mathbf{P}_k \mathbf{a}_k^{s+1}) \mathbf{P}_j^\top \mathbf{P}_k \mathbf{a}_k^{s+1} + \sum_{k=j}^{J} c_{jk} g'(\mathbf{a}_j^{s\top} \mathbf{P}_j^\top \mathbf{P}_k \mathbf{a}_k^s) \mathbf{P}_j^\top \mathbf{P}_k \mathbf{a}_k^s \tag{17}$$

%where $\boldsymbol{\nabla}_{jk}^s = g'(\mathbf{a}_j^{s\top} \mathbf{P}_j^\top \mathbf{P}_k \mathbf{a}_k^s) \mathbf{P}_j^\top \mathbf{P}_k \mathbf{a}_k^s$. From the subgradient inequality defined for any convex function, $f_j(\mathbf{a}_j, \mathbf{a}_{-j}^s)$ can be minorized through the minorizing inequality

$$f_j(\mathbf{a}_j, \mathbf{a}_{-j}^s) \geq f_j(\mathbf{a}_j^s, \mathbf{a}_{-j}^s) + (\mathbf{a}_j - \mathbf{a}_j^s)^\top f_j'(\mathbf{a}_j^s, \mathbf{a}_{-j}^s) := \tilde{f}_j(\mathbf{a}_j, \mathbf{a}_{-j}^s) \tag{18}$$

where the right-hand term of (18) is the so-called linear minorizing function. Maximizing this minorizing function over $\mathbf{a}_j$ under the length constraint $\mathbf{a}_j^\top \mathbf{a}_j = 1$ is obtained using the Cauchy-Schwartz inequality. The resulting update is defined as

$$\mathbf{a}_j^{s+1} = \frac{f_j'(\mathbf{a}_j^s, \mathbf{a}_{-j}^s)}{\|f_j'(\mathbf{a}_j^s, \mathbf{a}_{-j}^s)\|} = \mathbf{r}_j(\mathbf{a}_j^{s+1}, \ldots, \mathbf{a}_{j-1}^{s+1}, \mathbf{a}_j^s, \ldots, \mathbf{a}_J^s) = \mathbf{r}_j(\mathbf{a}_j^s, \mathbf{a}_{-j}^s). \tag{19}$$

where $\mathbf{r}_j$ denotes the mapping from Ω to Ω_j. Figure **??** illustrates the MM mechanisms underlying the update.

By construction, the so-called sandwich inequality

$$f_j(\mathbf{a}_j^{s+1}, \mathbf{a}_{-j}^s) \geq f_j(\mathbf{a}_j^s, \mathbf{a}_{-j}^s) + (\mathbf{a}_j^{s+1} - \mathbf{a}_j^s)^\top f_j'(\mathbf{a}_j^s, \mathbf{a}_{-j}^s) \geq f_j(\mathbf{a}_j^s, \mathbf{a}_{-j}^s) \tag{20}$$

shows that the update $\mathbf{a}_j^{s+1}$ increases f_j (or f_j stays the same). Therefore, the update (19) is the one used in Algorithm **??**. It is worth noticing that this update consitutes also a single iteration of the gradient based algorithm proposed by (Journée, Nesterov, Richtárik, and Sepulchre 2010) for maximizing the convex function $f_j(\mathbf{a}_j, \mathbf{a}_{-j}^s)$ over $\mathbf{a}_j \in \Omega_j$.

Using (61) and doing the loop over j in Algorithm **??** implies that

$$\begin{aligned} f(\mathbf{a}^{s+1}) &= f_J(\mathbf{a}_J^{s+1}, \mathbf{a}_{-J}^s) \geq f_{J-1}(\mathbf{a}_{J-1}^{s+1}, \mathbf{a}_{-(J-1)}^s) \\ &\geq \ldots \geq f_j(\mathbf{a}_j^{s+1}, \mathbf{a}_{-j}^s) \geq \ldots \geq f_2(\mathbf{a}_2^{s+1}, \mathbf{a}_{-2}^s) \geq f_1(\mathbf{a}_1^{s+1}, \mathbf{a}_{-1}^s) \geq f(\mathbf{a}^s) \end{aligned} \tag{21}$$

must hold and, therefore, the algorithm guarantees a monotonically increasing series of function values $f(\mathbf{a}^s)$. As the function f is bounded on Ω, the monotone sequence $\{f(\mathbf{a}^s)\}$ is actually converging.

In this section, we study the property of the convergence of the series of $\mathbf{a}^s$ obtained with Algorithm **??**. Here, we have to assume that $g(x)$ is continuously differentiable anywhere. In particular, we study the property of global convergence, that is, for any chosen initial point the sequence generated by the algorithm converges to a point for which a necessary condition of optimality holds (here the first order optimality condition). To study the global convergence of Algorithm **??**, we rely on the seminal work of (Zangwill 1969) and related results due to (Meyer 1976). Indeed, this algorithm can be placed in that framework by introducing the mapping $\mathbf{c}$ defined as follows.

Let $\mathbf{c}_j : \Omega \to \Omega$ a function defined as $\mathbf{c}_j(\mathbf{a}_1, \ldots, \mathbf{a}_J) = (\mathbf{a}_1, \ldots, \mathbf{a}_{j-1}, \mathbf{r}_j(\mathbf{a}), \mathbf{a}_{j+1}, \ldots, \mathbf{a}_J)$ and $\mathbf{c} : \Omega \to \Omega$ the recurrence equation defined as $\mathbf{c} = \mathbf{c}_J \circ \ldots \circ \mathbf{c}_1$. By construction, we have $\mathbf{a}^{s+1} = \mathbf{c}(\mathbf{a}^s)$ and the following inequalities

$$f(\mathbf{c}(\mathbf{a})) \geq \ldots \geq f(\mathbf{c}_2(\mathbf{c}_1(\mathbf{a})) \geq f(\mathbf{c}_1(\mathbf{a}) \geq f(\mathbf{a}) \tag{22}$$

holds. Let $\{\mathbf{a}^s\}_{s=0}^\infty$ be a sequence of iterates generated by Algorithm~**??** using update (19). Then, the following properties hold.

1. The limit of any convergent subsequence of $\{\mathbf{a}^s\}_{s=0}^\infty$ is a fixed point of $\mathbf{c}$ and therefore a solution of the stationary equation (15).
2. $\lim_{s\to\infty} f(\mathbf{a}^s) = f(\mathbf{a}^*)$, with $\mathbf{a}^* \in \Gamma$ where $\Gamma = \{\mathbf{a} \in \Omega : \mathbf{a} = \mathbf{c}(\mathbf{a})\}$ is the set of fixed points of $\mathbf{c}$.
3. The sequence $\{\mathbf{a}^s\}$ is asymptotically regular: $\lim_{s\to\infty} \|\mathbf{a}^{s+1} - \mathbf{a}^s\| = 0$.
4. Either $\{\mathbf{a}^s\}$ converges or its limit points form a continuum.

Proof. These properties are a direct application of Theorem 3.1 of (Meyer 1976, p. 110). This theorem assumes that two conditions are fulfilled: (i) the function $\mathbf{c} : \Omega \to \Omega$ is continuous and uniformly compact over the closed set Ω and (ii) $\mathbf{c}$ is strictly monotone with respect to the continuous function f.

(i) From the continuity of $\mathbf{r}_j$, $\mathbf{c}_j$ is continuous for $j = 1, \ldots, J$ and therefore $\mathbf{c} = \mathbf{c}_J \circ \ldots \circ \mathbf{c}_1$ is continuous because $\mathbf{c}$ is a composition of J continuous functions. Since Ω is compact, $\mathbf{c}$ is uniformly compact on Ω.

(ii) It remains to be shown that the function $\mathbf{c} : \Omega \to \Omega$ is strictly monotonically increasing with respect to f. This means that $\mathbf{c}$ must satisfy $f(\mathbf{a}) \leq f(\mathbf{c}(\mathbf{a}))$ with equality only when $\mathbf{a} \in \Gamma$. Monotonicity comes from equation (22). The property of strict monotonicity is shown as follows. Suppose $f(\mathbf{a}) = f(\mathbf{c}(\mathbf{a}))$. As $\mathbf{r}_j(\mathbf{a})$ is the unique maximizer of $f_j(\mathbf{a}) + (\tilde{\mathbf{a}}_j - \mathbf{a}_j)^\top f_j{}'(\mathbf{a})$ over the set of unit norm vector $\tilde{\mathbf{a}}_j$, we must have $\mathbf{a}_j = \mathbf{r}_j(\mathbf{a})$ for all j and therefore $\mathbf{a} = \mathbf{c}(\mathbf{a})$. From that, we deduce that if $\mathbf{a}$ is not a fixed point of $\mathbf{c}$, $f(\mathbf{a}) < f(\mathbf{c}(\mathbf{a}))$.

This concludes the proof. $\square$

To conclude this section, we have shown that the proposed algorithm is guaranteed to improve its function value in each iteration until a solution of the stationary equations related to optimization problem (14) is reached. In addition, under the assumptions that the function g is continuously differentiable and Γ is finite, we proved that the series of $\{\mathbf{a}^s\}$ produced by Algorithm **??** converges to a stationary point.

A more general formulation of Generalized Canonical Correlation Analysis to Reproducing Kernel Hilbert Space (KGCCA) has been proposed in (**?**).

In the materials described for GCCA, we are looking for linear components $\boldsymbol{\xi}_j^t \boldsymbol{x}_j, j = 1, \ldots, J$ which maximize optimization problem (98). It is possible to introduce a more general class of problem by considering the following optimization problem: %

$$\underset{f_1,\ldots,f_J}{\text{maximize}} \sum_{j,k=1}^{J} c_{jk}\, \mathrm{g}\left(\mathrm{cov}(f_j(\boldsymbol{x}_j), f_k(\boldsymbol{x}_k))\right) \text{ s.t. } \mathrm{var}(f_j(\boldsymbol{x}_j)) = 1, j = 1, \ldots, J. \tag{23}$$

For easier mathematical treatments, we assume that the f_js' belong to Reproducing Kernel Hilbert Space (RKHS) and we use cross-covariance operators on reproducing kernel Hilbert spaces (Baker 1973) to reformulate optimization problem (23) as suggested by (e.g. (Fukumizu, Bach, and Jordan 2004; Fukumizu, Bach, and Gretton 2007)). Cross-covariance operator is a natural RKHS extension of the covariance matrix in the original space. For two random vectors $\boldsymbol{x}_j$ and $\boldsymbol{x}_k$ endowed with Hilbert space $\mathcal{H}_j$ (resp. $\mathcal{H}_k$) with positive definite kernel function $k_j(\cdot, \cdot)$ (resp. $k_k(\cdot, \cdot)$). The cross-covariance operator Σ_{jk} from $\mathcal{H}_k$ to $\mathcal{H}_j$ is defined for all $f_j \in \mathcal{H}_j$ and $f_k \in \mathcal{H}_k$ by

$$\langle f_j, \Sigma_{jk} f_k \rangle_{\mathcal{H}_j} = \mathbb{E}\left[f_j(\boldsymbol{x}_j) f_k(\boldsymbol{x}_k)\right] - \mathbb{E}\left[f_j(\boldsymbol{x}_j)\right] \mathbb{E}\left[f_k(\boldsymbol{x}_k)\right] (= \mathrm{cov}\left(f_j(\boldsymbol{x}_j), f_k(\boldsymbol{x}_k)\right)).$$

The cross-covariance operator contains all the information regarding the dependance between $\boldsymbol{x}_j$ and $\boldsymbol{x}_k$ expressible by functions in the RKHS.

Optimization problem (23) can then be formulated in terms of cross-covariance operators and gives rise to the Population Kernel Generalized Canonical Correlation Analysis (Population KGCCA) optimization problem.

$$\underset{f_1,\ldots,f_J \in \mathcal{H}_1 \times \ldots \times \mathcal{H}_J}{\text{maximize}} \sum_{j,k=1}^{J} c_{jk} g\left(\langle f_j, \Sigma_{jk} f_k \rangle_{\mathcal{H}_j}\right) \text{ s.t. } \langle f_j, \Sigma_{jj} f_j \rangle_{\mathcal{H}_j} = 1, j = 1, \ldots, J \tag{24}$$

A sample-based optimization problem related to (24) can be derived by considering J blocks of centered variables $\mathbf{X}_1, \dots, \mathbf{X}_J$ measured on a set of n individuals (a row of $\mathbf{X}_j$ represents a realization of the row-random vector $\boldsymbol{x}_j^\top$). Let us note by x_j^i the ith row of block $\mathbf{X}_j$. $\mathbf{C} = (c_{jk})$ is now a design matrix describing a network of relationships between blocks. The empirical counterpart of optimization problem (24) is defined as:

$$\begin{aligned}&\underset{f_1,\dots,f_J \in \mathcal{H}_1 \times \dots \times \mathcal{H}_J}{\text{maximize}} \sum_{j,k=1}^{J} c_{jk}\, \text{g}\left(\langle f_j, \widehat{\Sigma}_{jk} f_k \rangle_{\mathcal{H}_j}\right) \\ &\text{s.t. } \langle f_j, ((1-\tau_j)\widehat{\Sigma}_{jj} + \tau_j I) f_j \rangle_{\mathcal{H}_j} = 1, \quad j = 1, \dots, J\end{aligned} \tag{25}$$

where the empirical cross-covariance is defined by

$$\langle f_j, \widehat{\Sigma}_{jk} f_k \rangle_{\mathcal{H}_j} = \frac{1}{n}\sum_{i=1}^{n}\left(f_j(x_j^i) - \frac{1}{n}\sum_{r=1}^{n} f_j(x_j^r)\right)\left(f_k(x_k^i) - \frac{1}{n}\sum_{s=1}^{n} f_k(x_k^s)\right) \tag{26}$$

and where $\tau_j \in [0, 1]$ are shrinkage parameters which enforce smoothness of f_j and enable operator inversion.

The resolution of optimization problem (25) is facilitated by supposing that the $f_j s'$ belong to a Reproducing Kernel Hilbert space (RKHS) with associated real-valued positive definite kernel $k_j(\cdot, \cdot)$. Indeed, from the reproducing property, equation (26) can be formulated as follows:

$$\langle f_j, \widehat{\Sigma}_{jk} f_k \rangle = \frac{1}{n}\sum_{i=1}^{n}\left[\langle f_j, \tilde{k}_j(\cdot, x_j^i)\rangle_{\mathcal{H}_j} \langle f_k, \tilde{k}_k(\cdot, x_k^i)\rangle_{\mathcal{H}_k}\right] \tag{27}$$

where

$$\tilde{k}_j(\cdot, x_j^i) = k_j(\cdot, x_j^i) - \frac{1}{n}\sum_{s=1}^{n} k_j(\cdot, x_j^s) \tag{28}$$

By using the standard orthogonality arguments (Wahba 1990; Bach and Jordan 2002), the solution of (25) can be written as follows:

$$f_j = \sum_{i=1}^{n} \alpha_j^i \tilde{k}_j(\cdot, x_j^i) = \sum_{i=1}^{n} \alpha_j^i \left[k_j(\cdot, x_j^i) - \frac{1}{n}\sum_{s=1}^{n} k_j(\cdot, x_j^s)\right] \tag{29}$$

Combining equations (27) and (29) leads to:

$$\begin{aligned}\langle f_j, \widehat{\Sigma}_{jk} f_k \rangle_{\mathcal{H}_j} &= \frac{1}{n}\sum_{i=1}^{n}\left[\langle \sum_{l=1}^{n} \alpha_j^l \tilde{k}_j(x_j^l, \cdot), \tilde{k}_j(\cdot, x_j^i)\rangle_{\mathcal{H}_j} \langle \sum_{m=1}^{n} \alpha_k^m \tilde{k}_k(x_k^m, \cdot), \tilde{k}_k(\cdot, x_k^i)\rangle_{\mathcal{H}_k}\right] \\ &= \frac{1}{n}\sum_{i=1}^{n}\sum_{l=1}^{n}\sum_{m=1}^{n} \alpha_j^l \tilde{k}_j(x_j^l, x_j^i) \tilde{k}_k(x_k^m, x_k^i) \alpha_k^m \\ &= n^{-1} \boldsymbol{\alpha}_j^\top \tilde{\mathbf{K}}_j \tilde{\mathbf{K}}_k \boldsymbol{\alpha}_k\end{aligned} \tag{30}$$

where $\tilde{\mathbf{K}}_j$ is the centered Gram matrix (Schölkopf, Smola, and Müller 1998) defined by:

$$\tilde{\mathbf{K}}_j = (\mathbf{I}_n - n^{-1}\mathbf{1}_n\mathbf{1}_n^\top)\mathbf{K}_j(\mathbf{I}_n - n^{-1}\mathbf{1}_n\mathbf{1}_n^\top) \tag{31}$$

with $(\mathbf{K}_j)_{st} = k_j(x_j^s, x_j^t)$ and $\mathbf{1}_n$ is an $n \times 1$ vector of ones.

We also obtain:

$$\langle f_j, \widehat{\Sigma}_{jj} f_j \rangle_{\mathcal{H}_j} = \frac{1}{n}\sum_{i=1}^{n}\left(f_j(x_j^i) - \frac{1}{n}\sum_{s=1}^{n} f_j(x_j^s)\right)^2 = n^{-1}\boldsymbol{\alpha}_j^\top \tilde{\mathbf{K}}_j^2 \boldsymbol{\alpha}_j \tag{32}$$

and

$$\|f_j\|_{\mathcal{H}_j}^2 = \langle f_j, f_j \rangle_{\mathcal{H}_j} = \boldsymbol{\alpha}_j^\top \tilde{\mathbf{K}}_j \boldsymbol{\alpha}_j. \tag{33}$$

Thus,

$$(1-\tau_j)\langle f_j, \widehat{\Sigma}_{jj} f_j \rangle_{\mathcal{H}_j} + \tau_j \|f_j\|_{\mathcal{H}_j}^2 = \boldsymbol{\alpha}_j^\top \left[\tilde{\mathbf{K}}_j \left((1-\tau_j)\frac{1}{n}\tilde{\mathbf{K}}_j + \tau_j \mathbf{I}\right)\right]\boldsymbol{\alpha}_j \tag{34}$$

Optimization problem (25) becomes

$$\begin{aligned} &\underset{\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_J}{\text{maximize}} \sum_{j,k=1}^{J} c_{jk}\, \mathrm{g}\left(n^{-1}\boldsymbol{\alpha}_j^\top \tilde{\mathbf{K}}_j \tilde{\mathbf{K}}_k \boldsymbol{\alpha}_k\right) \\ &\text{s.t. } \boldsymbol{\alpha}_j^\top \left[\tilde{\mathbf{K}}_j \left((1-\tau_j)n^{-1}\tilde{\mathbf{K}}_j + \tau_j \mathbf{I}_n\right)\right]\boldsymbol{\alpha}_j = 1, \quad j = 1,\ldots,J \end{aligned} \tag{35}$$

and fits optimization problem (13) by setting $\mathbf{Q}_j = n^{-1/2}\tilde{\mathbf{K}}_j$ and $\mathbf{M}_j = \tilde{\mathbf{K}}_j\left((1-\tau_j)n^{-1}\tilde{\mathbf{K}}_j + \tau_j \mathbf{I}_n\right)$, $0 \le \tau_j \le 1$, with m and p_j both equal to the number n of individuals. Therefore, Algorithm **??** can also be used for the resolution of optimization problem (35) with $\mathbf{P}_j = \mathbf{Q}_j \mathbf{M}_j^{-1/2}$ and $\mathbf{a}_j = \mathbf{M}_j^{1/2}\boldsymbol{\alpha}_j$. Therefore, the KGCCA algorithm reduces to Algorithm **??** described below.

An implementation of the KGCCA algorithm is freely available on CRAN as part of the RGCCA package.

We note that the choice of the kernel function for each block is not discussed here but we stress that KGCCA is a versatile tool for multiblock data analysis that allows recovering nonlinear relationships between blocks (non linear kernel function) and handling any type of blocks (e.g. observations characterized by histograms, intervals, strings, ...) as long as relevant kernel function can be defined for each block (Vert and Kanehisa 2003; Yamanishi, Vert, and Kanehisa 2004). To conclude, RGCCA is recovered with KGCCA associated with linear KGCCA. It turns out that KGCCA provides a "kernel" counterpart of all the multiblock methods that are covered by RGCCA. At last, the KGCCA algorithm is much more attractive than the RGCCA algorithm in high dimensional block setting.

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 ℓ_1-penalty promoting sparsity. Numerous ℓ_1 and/or ℓ_2 regularized extensions of CCA have been proposed when the number of variables p_j exceeds the number of observations n for any jth block (e.g. (Vinod 1976; Waaijenborg, Verselewel de Witt Hamer, and Zwinderman 2008; Parkhomenko, Tritchler, and Beyene 2009; Le Cao, Martin, Robert-Granie, and Besse 2009; Witten, Tibshirani, and Hastie 2009; Lykou and Whittaker 2010; Hardoon and Shawe-Taylor 2011)). SGCCA extends these approaches to the more than two block case.

The SGCCA optimization problem and the associated algorithm are presented in this section. From now on, all τ_j equal 1, which means that the constraints are applied on the length of the $\mathbf{a}_j$. Adding an ℓ_1 penalty on $\mathbf{a}_1, \dots, \mathbf{a}_J$, SGCCA is defined as the following optimization problem:

$$\underset{\mathbf{a}_1,\mathbf{a}_2,\dots,\mathbf{a}_J}{\text{maximize}} f(\mathbf{a}_1,\dots,\mathbf{a}_J) = \sum_{j,k=1}^{J} c_{jk} g(n^{-1}\mathbf{a}_j^\top \mathbf{X}_j^\top \mathbf{X}_k \mathbf{a}_k) \tag{36}$$

$$\text{s. t. } \|\mathbf{a}_j\|_2 = 1 \text{ and } \|\mathbf{a}_j\|_1 \leq s_j, j = 1,\dots,J$$

where s_j is a positive constant that determines the amount of sparsity for $\mathbf{a}_j$, $j = 1,\dots,J$. The smaller s_j, the larger the degree of sparsity for $\mathbf{a}_j$.

First, we re-write the criterion (102) using Lagrange multipliers

$$\mathcal{L} = \sum_{j,k=1}^{J} c_{jk} g\Big(n^{-1}\mathbf{a}_j^\top \mathbf{X}_j^\top \mathbf{X}_k \mathbf{a}_k\Big) - \Bigg[\sum_{j=1}^{J} \frac{\lambda_{2j}}{2}\Big(\|\mathbf{a}_j\|_2^2 - 1\Big) + \sum_{j=1}^{J} \lambda_{1j}\Big(\|\mathbf{a}_j\|_1 - s_j\Big)\Bigg] \tag{37}$$

where $\lambda_{11},\dots,\lambda_{1J},\lambda_{21},\dots,\lambda_{2J}$ are the Lagrange multipliers. Considering the partial subgradients of the Lagrangian function with respect to $\mathbf{a}_j$ yields the following J equations for SGCCA:

$$\partial_{\mathbf{a}_j}\mathcal{L} = \sum_{k=1}^{J} c_{jk} g'\Big(\mathbf{a}_j^\top \mathbf{X}_j^\top \mathbf{X}_k \mathbf{a}_k\Big)\mathbf{X}_j^\top \mathbf{X}_k \mathbf{a}_k - \Big[\lambda_{2j}\mathbf{a}_j + \lambda_{1j}\boldsymbol{\gamma}_j\Big], \ j = 1,\dots,J \tag{38}$$

where γ_{jk}, the k^{th} element of $\boldsymbol{\gamma}_j$, is the subgradient of $\sum_{k=1}^{p_j} |\mathbf{a}_{jk}|$ with respect to $\mathbf{a}_{jk}$, and is defined by: $\gamma_{jk} = \text{sign}(a_{jk})$ if $a_{jk} \neq 0$; and $\gamma_{jk} \in [-1,+1]$ if $a_{jk} = 0$.

Let us introduce the inner components $\mathbf{z}_j$defined as

$$\mathbf{z}_j = \sum_{k=1}^{J} c_{jk} g'\Big(n^{-1}\mathbf{a}_j^\top \mathbf{X}_j^\top \mathbf{X}_k \mathbf{a}_k\Big)\mathbf{X}_k \mathbf{a}_k \tag{39}$$

The inner component plays a central role in the SGCCA algorithm to be described and enable us to simplify equations (38) as follows:

$$\partial_{\mathbf{a}_j}\mathcal{L} = n^{-1}\mathbf{X}_j^\top \mathbf{z}_j - \lambda_{2j}\mathbf{a}_j - \lambda_{1j}\boldsymbol{\gamma}_j, \ \ j = 1,\dots,J \tag{40}$$

From the definition of the subgradient and from equation (40),

$$\partial_{a_{jk}}\mathcal{L} = \begin{cases} n^{-1}\mathbf{x}_{jk}^\top \mathbf{z}_j - \lambda_{2j} a_{jk} - \lambda_{1j}\text{sign}(a_{jk}) & \text{if} \quad a_{jk} \neq 0 \\ \Big[n^{-1}\mathbf{x}_{jk}^\top \mathbf{z}_j - \lambda_{1j}, n^{-1}\mathbf{x}_{jk}^\top \mathbf{z}_j + \lambda_{1j}\Big] & \text{if} \quad a_{jk} = 0 \end{cases} \tag{41}$$

where $\mathbf{x}_{jk}$ represents the k^{th} column $\mathbf{X}_j$. At the optimum, we must have $0 \in \partial_{\mathbf{a}_j}\mathcal{L}$ and we get:

$$\begin{cases} n^{-1}\mathbf{x}_{jk}^\top \mathbf{z}_j - \lambda_{2j} a_{jk} - \lambda_{1j}\text{sign}(a_{jk}) = 0 & \text{if} \quad a_{jk} \neq 0 \\ |n^{-1}\mathbf{x}_{jk}^\top \mathbf{z}_j| < \lambda_{1j} & \text{if} \quad a_{jk} = 0 \end{cases} \tag{42}$$

From equation (42), the following equality holds: $\text{sign}(n^{-1}\mathbf{x}_{jk}^\top \mathbf{z}_j) = \text{sign}(a_{jk})$, which yields:

$$a_{jk} = \begin{cases} \frac{1}{\lambda_{2j}}\Big(n^{-1}\mathbf{x}_{jk}^\top \mathbf{z}_j - \lambda_{1j}\text{sign}(n^{-1}\mathbf{x}_{jk}^\top \mathbf{z}_j)\Big) \\ 0 \quad \text{if} \quad |n^{-1}\mathbf{x}_{jk}^\top \mathbf{z}_j| < \lambda_{1j} \end{cases} = \begin{cases} \frac{1}{\lambda_{2j}}\text{sign}(n^{-1}\mathbf{x}_{jk}^\top \mathbf{z}_j)\Big(|n^{-1}\mathbf{x}_{jk}^\top \mathbf{z}_j| - \lambda_{1j}\Big) \\ 0 \quad \text{if} \quad |n^{-1}\mathbf{x}_{jk}^\top \mathbf{z}_j| < \lambda_{1j} \end{cases} \tag{43}$$

or in compact form as follows:

$$a_{jk} = \frac{1}{\lambda_{2j}}\text{sign}(n^{-1}\mathbf{x}_{jk}^\top \mathbf{z}_j)\text{max}(0, |n^{-1}\mathbf{x}_{jk}^\top \mathbf{z}_j| - \lambda_{1j}) \tag{44}$$

$\mathbf{a}_j$ can be written in matrix notation:
$\mathbf{a}_j = \frac{1}{\lambda_{2j}} S(n^{-1}\mathbf{X}_j^\top \mathbf{z}_j, \lambda_{1j})$ where S is the soft-thresholding operator defined by $S(a,\lambda) = \text{sign}(a)\text{max}(0, |a| - \lambda)$, λ_{2j} is chosen such that $\|\mathbf{a}_j\|_2 = 1$ and λ_{1j} chosen such that $\|\mathbf{a}_j\|_1 \leq s_j$. This yields the following final expression for $\mathbf{a}_j$:

$$\mathbf{a}_j = \frac{S(n^{-1}\mathbf{X}_j^\top \mathbf{z}_j, \lambda_{1j})}{\|S(n^{-1}\mathbf{X}_j^\top \mathbf{z}_j, \lambda_{1j})\|_2}, \quad j = 1, \ldots, J \tag{45}$$

From equation (44), we conclude that each a_{jk} is cancelled out and does not contribute to the construction of the block component $\mathbf{X}_j\mathbf{a}_j$, if the covariance between $\mathbf{x}_{jk}$ and $\mathbf{z}_j$ is below a given threshold λ_{1j} (where λ_{1j} is defined such that $\|\mathbf{a}_j\|_1 \leq s_j$).

Note that equation (45) is also obtained as the unique solution of the following convex optimization problem.

$$\underset{\mathbf{a}_j}{\text{argmax}}\ \text{cov}(\mathbf{X}_j\mathbf{a}_j, \mathbf{z}_j) \ \text{ s.t. } \ \|\mathbf{a}_j\|_2 \leq 1 \ \text{ and } \ \|\mathbf{a}_j\|_1 \leq s_j. \tag{46}$$

Again, the maximization of the optimization problem (102) requires combining block relaxation with MM principle and the SGCCA algorithm reduces to Algorithm **??**.

This algorithm is avalaible on CRAN as part of the RGCCA package. From the unicity of the update, the arguments that were used for Algorithm **??** to prove the various convergence properties can be extended to Algorithm **??**. In addition, Algorithm **??** is found to be very stable and usually reaches a convergence tolerance within a few iterations. Moreover, it is worth mentioning that Algorithm **??** handle missing data simply by skipping the missing elements in the computation of inner products.

Discussion and conclusion. From the viewpoint of the optimization problem (99), the regularization parameters $\tau_j \in [0,1]$, $j = 1, \ldots, J$ enable a smooth interpolation between the maximization of the covariance (all $\tau_j s' = 1$) and the maximization of the correlation (all $\tau_j s' = 0$). The covariance based model ($\tau_j = 1$) that underlies SGCCA first tends to find components with large variance that explain well their own block (PCA criteria), while taking into account the correlations with neighboring components. This unbalanced compromise between variance and correlation is mandatory for stable variable selection. Moreover, by setting all τ_j equal to 1, we implicitly assume that for each block, the true covariance matrix is estimated by the identity. To some extent, this also justifies the stability of the variable selection in SGCCA (see Table **??**). This also highlights the fact that when $\tau_j = 1$, neither matrix inversion nor diagonalization is required.

In general, and especially for the covariance-based criterion, the data blocks might be pre-processed to ensure comparability between variables and blocks. To make variables comparable, standardization is applied (zero mean and unit variance). To make blocks comparable, a strategy is to divide each block by the square root of its number of variables. This two-step procedure leads to Trace$(I^{-1}\mathbf{X}_l^\top \mathbf{X}_l) = 1$ for each block (i.e. the sum of the eigenvalues of the correlation matrix of $\mathbf{X}_l$ is equal to 1 whatever the block).

We present in this paper a global method for multiblock component analysis called Global RGCCA. This approach will be compared to MAXBET and MAXDIFF presented in **?**, **?** and in Hanafi and Kiers (2006).

Regularized Generalized Canonical Correlation Analysis (RGCCA) is defined in this paper at the population level.

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 $\mathbf{\Sigma}$. The vector $\boldsymbol{x}$ is composed of J subvectors $\boldsymbol{x}_j = (x_{j1}, \ldots, x_{jp_j})^\top$. The covariance matrix matrix $\mathbf{\Sigma}$ is composed of J^2 submatrices $\mathbf{\Sigma}_{jk} = \mathbb{E}\left[\boldsymbol{x}_j \boldsymbol{x}_k^\top\right]$. Let $\mathbf{a}_j = (a_{j1}, \ldots, a_{jp_j})^\top$ be a non-random p_j-dimensional column vector. A composite variables y_j is defined as the linear combination of the elements of $\boldsymbol{x}_j$: $y_j = \sum_h a_{jh} x_{jh} = \mathbf{a}_j^\top \boldsymbol{x}_j$.

RGCCA at population level is defined as the following optimization problem:

$$\underset{\mathbf{a}_1,\ldots\mathbf{a}_J}{\text{maximize}} \sum_{j,k=1}^{J} c_{jk} \mathrm{g}\left(\mathrm{cov}\left(\mathbf{a}_j^\top \boldsymbol{x}_j, \mathbf{a}_k^\top \boldsymbol{x}_k\right)\right) \text{ s.t. } \mathbf{a}_j^\top \mathbf{M}_j \mathbf{a}_j = 1, j = 1, \ldots, J. \tag{47}$$

where

- The function g is any continuously differentiable convex function. Its derivative is noted g'. If $c_{jj} \neq 0$ for some j the constraint $g'(x) \geq 0$ for $x \geq 0$ must be added. The scheme $g(x) = |x|$ can be included in this class of functions because the case $x = 0$ never appears in practical applications.

- 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.

- Each block metric matrix $\mathbf{M}_j$ is positive definite matrix.

Optimization problem (47) can be expressed as

$$\underset{\mathbf{a}_1,\ldots\mathbf{a}_J}{\text{maximize}} f(\mathbf{a}_1, \ldots \mathbf{a}_J) = \sum_{j,k=1}^{J} c_{jk} \mathrm{g}\left(\mathbf{a}_j^\top \mathbf{\Sigma}_{jk} \mathbf{a}_k\right) \text{ s.t. } \mathbf{a}_j^\top \mathbf{M}_j \mathbf{a}_j = 1, j = 1, \ldots, J. \tag{48}$$

The partial gradient $\nabla_j f(\mathbf{a}_1, \ldots \mathbf{a}_J)$ of $f(\mathbf{a}_1, \ldots \mathbf{a}_J)$ with respect to $\mathbf{a}_j$ is a p_j-dimensional column vector given by:

$$\nabla_j f(\mathbf{a}_1, \ldots \mathbf{a}_J) = 2 \sum_{k=1}^{J} c_{jk} g'\left(\mathbf{a}_j^\top \mathbf{\Sigma}_{jk} \mathbf{a}_k\right) \mathbf{\Sigma}_{jk} \mathbf{a}_k \tag{49}$$

The stationary points of (48) (i.e. the points cancelling the partial derivatives of the Lagrange function associated with (48)) satisfy the following stationary equations

$$\mathbf{a}_j = \frac{\mathbf{M}_j^{-1}\nabla_j f(\mathbf{a}_1, \ldots \mathbf{a}_J)}{\|\mathbf{M}_j^{-1/2}\nabla_j f(\mathbf{a}_1, \ldots \mathbf{a}_J)\|}, j = 1, \ldots, J. \tag{50}$$

In some situations, the stationary equations (50) are easy to solve (immediate solution or eigenvector equation). In others situations, the iterative alogrithm described in the next section has to be used to find a solution of (50).

Optimization problem (48) can be simplified by considering the transforms $\mathbf{v}_j = \mathbf{M}_j^{1/2}\mathbf{a}_j$ and $\mathbf{Q}_{jk} = \mathbf{M}_j^{-1/2}\mathbf{\Sigma}_{jk}\mathbf{M}_k^{-1/2}$. Expressing (48) in terms of $\mathbf{v}_j$ and $\mathbf{Q}_{jk}$ yields

$$\underset{\mathbf{v}_1,\ldots\mathbf{v}_J}{\text{maximize}} f(\mathbf{v}_1, \ldots \mathbf{v}_J) = \sum_{j,k,=1}^{J} c_{jk}\text{g}\left(\mathbf{v}_j^\top \mathbf{Q}_{jk}\mathbf{v}_k\right) \text{ s.t. } \mathbf{v}_j^\top\mathbf{v}_j = 1, j = 1, \ldots, J. \tag{51}$$

The partial gradient of $f(\mathbf{v}_1, \ldots \mathbf{v}_J)$ with respect to $\mathbf{v}_j$ is equal to

$$\nabla_j f(\mathbf{v}_1, \ldots \mathbf{v}_J) = 2\sum_{k=1}^{J} c_{jk}g'\left(\mathbf{v}_j^\top \mathbf{Q}_{jk}\mathbf{v}_k\right)\mathbf{Q}_{jk}\mathbf{v}_k. \tag{52}$$

Cancelling the partial gradients of the Lagrange function of (51) with respect to $\mathbf{v}_j$ and taking into account the normalization constraints yields the following stationnary equations:

$$\mathbf{v}_j = \frac{\nabla_j f(\mathbf{v}_1, \ldots \mathbf{v}_J)}{\|\nabla_j f(\mathbf{v}_1, \ldots \mathbf{v}_J)\|}, j = 1, \ldots, J. \tag{53}$$

It is useful to introduce some additional notations: $\Omega_j = \{\mathbf{v}_j \in \mathbb{R}^{p_j}; \|\mathbf{v}_j\| = 1\}$, $\Omega = \Omega_1 \times \ldots \times \Omega_J$ and $\mathbf{v} = (\mathbf{v}_1, \ldots, \mathbf{v}_J)$. We denote by $\mathbf{v}^* = (\mathbf{v}_1^*, \ldots, \mathbf{v}_J^*) \in \Omega$ a solution of (51). $\mathbf{v}^*$ is solution of the stationary equation (53). The iterative algorithm described in the next section can be used to find a solutin of (53).

Note that, when all $c_{jj} = 0$ in (48), convexity and continous differentiability of the scheme function g imply that the objective functin $f(\mathbf{v}_1, \ldots \mathbf{v}_J)$ is continuously differentiable multi-convex function (i.e. for each j, $f(\mathbf{v}_1, \ldots \mathbf{v}_J)$ is convex function of $\mathbf{v}_j$ while other $\mathbf{v}_k$ are fixed). When some $c_{jj} \neq 0$ in (48), the condition $g'(x) \geq 0$ for $x \geq 0$ is a sufficient condition to guaranty this property for $f(\mathbf{v}_1, \ldots \mathbf{v}_J)$. This condition guarantees that the second derivative of $g\left(\mathbf{v}_k^\top \mathbf{Q}_{kk}\mathbf{v}_k\right)$ is positive definite:

$$\frac{\partial^2 g\left(\mathbf{v}_k^\top \mathbf{Q}_{kk}\mathbf{v}_k\right)}{\partial\mathbf{v}_k\partial\mathbf{v}_k^\top} = 2\left[g'\left(\mathbf{v}_k^\top \mathbf{Q}_{kk}\mathbf{v}_k\right)\mathbf{Q}_{kk} + 2g''\left(\mathbf{v}_k^\top \mathbf{Q}_{kk}\mathbf{v}_k\right)\mathbf{Q}_{kk}\mathbf{v}_k\mathbf{v}_k^\top\mathbf{Q}_{kk}\right]. \tag{54}$$

Therefore (51) is a special case of the general framework for maximizing a multi-convex coninuously differentiable function described in the next section.

In this section, $f(\mathbf{v}_1, \ldots, \mathbf{v}_J) : \mathbb{R}^{p_1} \times \ldots \times \mathbb{R}^{p_J} \to \mathbb{R}$ is any multiconvex continuously differentiable function. We consider the following optimization problem:

$$\text{maximize} f(\mathbf{v}_1, \ldots, \mathbf{v}_J) \text{ s.t. } \mathbf{v}_j^\top\mathbf{v}_j = 1, j = 1, \ldots, J. \tag{55}$$

We assume $\nabla_j f(\mathbf{v}_1, \ldots, \mathbf{v}_J) \neq \mathbf{0}$. This assumption is not too binding as $\nabla_j f(\mathbf{v}_1, \ldots, \mathbf{v}_J) = \mathbf{0}$ characterizes the global minimum of $f(\mathbf{v}_1, \ldots, \mathbf{v}_J)$ with respect to $\mathbf{v}_j$ when all other vector $\mathbf{v}_k$ are fixed. Therefore, we can introduce the unit norm partial gradient

$$r_j(\mathbf{v}_1, \ldots, \mathbf{v}_J)) = \frac{\nabla_j f(\mathbf{v}_1, \ldots, \mathbf{v}_J)}{\|\nabla_j f(\mathbf{v}_1, \ldots, \mathbf{v}_J)\|}. \tag{56}$$

Cancelling the partial gradients of the Lagrange function of (55) with respect to $\mathbf{v}_j$ and taking into account the normalization constraints yields the following stationary equations of (55):

$$\mathbf{v}_j = r_j(\mathbf{v}_1, \ldots, \mathbf{v}_J), j = 1, \ldots, J. \tag{57}$$

Solutions of (57) are the stationary points of (55). We propose Algorithm **??** described below to solve (57).

In the case of a single block ($J = 1$), Algorithm **??** is similar to the gradient based algorithm proposed by (Journée *et al.* (2010)) for maximizing a convex function of several variables with spherical constraints (see Problem 27, p. 529). For studying the convergence properties of Algorithm **??**, it is useful to introduce some additional notations: $c_j : \Omega \mapsto \Omega$ is an operator defined as $c_j(\mathbf{v}) = (\mathbf{v}_1, \ldots, \mathbf{v}_{j-1}, r_j(\mathbf{v}), \mathbf{v}_{j+1}, \ldots, \mathbf{v}_J)$ 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 function composition.

We consider the sequence $\{\mathbf{v}^s = (\mathbf{v}_1^s, \ldots, \mathbf{v}_J^s)\}$ generated by Algorithm **??**. Using the operator c, the "for loop" and the equations (**??**) inside Algorithm **??** can be replaced by the following recurrence relation:

$$\mathbf{v}^{s+1} = c(\mathbf{v}^s). \tag{58}$$

Note that the set of stationary points of optimization problem (55) is equal to the set of fixed points of c. To study the convergence properties of Algorithm **??**, we will consider the infinite sequence $\{\mathbf{v}^s\}_{s=0}^{\infty}$ generated by (58). The convergence properties of Algorithm (55) are summarized in the next proposition.

Proposition 1.3. *Let $\{\mathbf{v}^s\}_{s=0}^{\infty}$ be any sequence generated by the recurrence relation $\mathbf{v}^{s+1} = c(\mathbf{v}^s)$ with $\mathbf{v}^0 \in \Omega$. Then, the following properties hold:*

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

b. If the infinite sequence $\{f(\mathbf{v}^s)\}$ involves a finite number of distinct terms, then the last distinct point satisfies $c(\mathbf{v}^s) = \mathbf{v}^s$ and therefore is a stationary point of problem (55).

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

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

e. The sequence $\{\mathbf{v}^s\}$ is asymptotically regular, that is $\lim_{s \to \infty} \|\mathbf{v}^{s+1} - \mathbf{v}^s\|^2 = 0$. This result implies that if the threshold ε for the stopping criterion in Algorithm **??** *is made sufficiently small, the output of Algorithm* **??** *will be as close as wanted to a stationary point of (55).*

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

The three first points a. to c. of Proposition 1.3 concern the behavior of the sequence values $\{f(\mathbf{v}^s)\}$ of the objective function, whereas the three last points d. to f. are about the behaviour of the sequence $\{\mathbf{v}^s\}$. The results given in the following Lemma are useful for proving Proposition 1.3.

Lemma 1.4. *Consider the set* Ω*, the function* $f : \Omega \mapsto \mathbb{R}$ *and the operator* $c : \Omega \mapsto \Omega$ *defined above. Then, the following properties hold:*

1. Ω *is a compact set;*
2. c *is a continuous operator;*
3. $f(\mathbf{v}) \leq f(c(\mathbf{v}))$ *for any* $\mathbf{v} \in \Omega$*;*
4. *If* $f(\mathbf{v}) = f(c(\mathbf{v}))$*, then* $c(\mathbf{v}) = \mathbf{v}$.

Proof of Lemma 1.4. **Point 1**: Ω is compact as the Cartesian product of J compact sets.

Point 2: f being a continuously differentiable convex function, r_j is continuous. This implies that $c_j : \Omega \to \Omega$ is a continuous operator. The operator $c = c_J \circ c_{J-1} \circ \ldots \circ c_1$ is also continuous as composition of J continuous operators.

Point 3: Let $\mathbf{v} = (\mathbf{v}_1, \ldots, \mathbf{v}_j, \ldots, \mathbf{v}_J) \in \Omega$. First, we want to find an update $\hat{\mathbf{v}}_j \in \Omega_j$ of $\mathbf{v}_j$ such that $f(\mathbf{v}) \leq f(\mathbf{v}_1, \ldots, \mathbf{v}_{j-1}, \hat{\mathbf{v}}_j, \mathbf{v}_{j+1}, \ldots, \mathbf{v}_J)$. For that purpose, we use the following inequality which state that a continosly differentiable convex function lies above its linear approximation at $\mathbf{v}_j$ for any $\tilde{\mathbf{v}}_j \in \Omega_j$:

$$f(\mathbf{v}_1, \ldots, \mathbf{v}_{j-1}, \tilde{\mathbf{v}}_j, \mathbf{v}_{j+1}, \ldots, \mathbf{v}_J) \geq f(\mathbf{v}) + \left(\nabla_j f(\mathbf{v})\right)^\top (\tilde{\mathbf{v}}_j - \mathbf{v}_j) = \ell_j(\tilde{\mathbf{v}}_j, \mathbf{v}) \tag{59}$$

Using the Cauchy-Scwartz inequality, we obtain the unique maximizer $\hat{\mathbf{v}}_j \in \Omega_j$ of $\ell_j(\tilde{\mathbf{v}}_j, \mathbf{v})$ with respect to $\tilde{\mathbf{v}}_j \in \Omega_j$:

$$\hat{\mathbf{v}}_j = \underset{\tilde{\mathbf{v}}_j \in \Omega_j}{\text{argmax}} \frac{\nabla_j f(\mathbf{v})}{\|\nabla_j f(\mathbf{v})\|}. \tag{60}$$

We deduce from (59) the following inequalites for each $j = 1, \ldots, J$:

$$f(\mathbf{v}) = \ell_j(\mathbf{v}_j, \mathbf{v}) \leq \ell_l(r_j(\mathbf{v}), \mathbf{v}) \leq f(\mathbf{v}_1, ..., \mathbf{v}_{j-1}, r_j(\mathbf{v}), \mathbf{v}_{j+1}, ..., \mathbf{v}_J) = f(c_j(\mathbf{v})). \tag{61}$$

This implies that updating $\mathbf{v}_j$ by $\hat{\mathbf{v}}_j = r_j(\mathbf{v})$ increases $f(\mathbf{v})$ (or $f(\mathbf{v})$ stays the same). Moreover, the following inequality is deduced from (61) for each $j = 2, ..., J$:

$$f(c_{j-1} \circ \ldots \circ c_1(\mathbf{v})) \leq f(c_j \circ c_{j-1} \circ \ldots \circ c_1(\mathbf{v})). \tag{62}$$

This yields the desired inequalities for any $\mathbf{v} \in \Omega$:

$$f(\mathbf{v}) \leq f(c_1(\mathbf{v})) \leq f(c_2 \circ c_1(\mathbf{v})) \leq \ldots \leq f(c_J \circ \ldots \circ c_1(\mathbf{v})) = f(c(\mathbf{v})). \tag{63}$$

Point 4: If $f(\mathbf{v}) = f(c(\mathbf{v}))$ for $\mathbf{v} \in \Omega$ then equation (63) implies

$$f(\mathbf{v}) = f(c_1(\mathbf{v})) = f(c_2 \circ c_1(\mathbf{v})) = \ldots = f(c_J \circ \ldots \circ c_1(\mathbf{v})) = f(c(\mathbf{v})). \tag{64}$$

Using equation (61), the equality $f(\mathbf{v}) = f(c_1(\mathbf{v}))$ implies $\ell_1(r_1(\mathbf{v}), \mathbf{v}) = \ell_1(\mathbf{v}_1, \mathbf{v})$ and therefore $\mathbf{v}_1 = r_1(\mathbf{v})$ as $r_1(\mathbf{v})$ is the unique maximizer of $\ell_1(r_1(\tilde{\mathbf{v}}_1), \mathbf{v})$ with respect to $\tilde{\mathbf{v}}_1 \in \Omega_1$. From this result, we deduce $\mathbf{v} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_L) = (r_1(\mathbf{v}), \mathbf{v}_2, \ldots, \mathbf{v}_L) = c_1(\mathbf{v})$ and then, by induction,

$$\mathbf{v} = c_1(\mathbf{v}) = c_2 \circ c_1(\mathbf{v}) = \ldots = c_J \circ \ldots \circ c_1(\mathbf{v}) = c(\mathbf{v}). \tag{65}$$

□

Proof of Proposition 1.3. **Point a.**. Point 3 of Lemma 1.4 implies that the sequence $f(\mathbf{v}^s)$ is monotonically increasing, and therefore, convergent as the continuous function f is bounded on the compact set Ω.

Point b.. If the infinite sequence $f(\mathbf{v}^s)$ has a finite number of distinct terms, it cannot be a strictly increasing sequence and consequently there exists some integer M such that $f(\mathbf{v}^0) < f(\mathbf{v}^1) < \ldots < f(\mathbf{v}^M) = (\mathbf{v}^{M+1})$. Then, Point 4 of Lemma 1.4 implies that $\mathbf{v}^M$ is a fixed point of c.

Point c. to f. They are deduced from a direct application of Meyer's monotone convergence theorem (Theorem 3.1 in Meyer (1976)). This theorem gives quite general conditions under which a sequence $(\mathbf{v}^s)$ produced by an algorithm that monotonically increases a continuous objective function will converge. Meyer considered the case of a point-to-set operator $c : \Omega \mapsto \mathcal{P}(\Omega)$, where $\mathcal{P}(\Omega)$ is the set of all nonempty subsets of Ω. In this article, c is a point-to-point operator and the conditions of Meyer's theorem reduce to the three following conditions (see **?**): (1) c is a continuous operator; (2) c is strictly monotone (increasing) with respect to f ; and (3) c is uniformly compact on Ω. Condition (2) means that points (3) and (4) of Lemma 1.4 are verified. Condition (3) means that there exists a compact set $\mathcal{K}$ such that $c(\mathbf{v}) \in \mathcal{K}$ for all $\mathbf{v} \in \Omega$. According to Lemma 1.4, these three conditions are satisfied for Algorithm **??** and therefore, Meyer's theorem applies to any sequence $\mathbf{v}^s$ produced by the recurrence equation $\mathbf{v}^{s+1} = c(\mathbf{v}^s)$ with $\mathbf{v}^0 \in \Omega$. □

First we need to solve optimization problem (51). The optimization problem (51) is a special case of (55). The partial gradient $\nabla_j f(\mathbf{v}_1, \ldots, \mathbf{v}_J)$ of the objective function $f(\mathbf{v}_1, \ldots, \mathbf{v}_J)$ in (51) with respect to $\mathbf{v}_j$ is given in equation (49). The following Algorithm **??** for solving (51) is deduced from Algorithm **??**.

The original weight vectors $\mathbf{a}_j, j = 1, \ldots, J$ are recovered using the following transforms: $\mathbf{a}_j = \mathbf{M}_j^{-1/2}\mathbf{v}_j$, $j = 1, \ldots, J$. However, we can write Algorithm **??** in terms of the original data and we obtain Algorithm **??** given below. The partial gradient $\nabla_j f(\mathbf{a}_1, \ldots, \mathbf{a}_J)$ of the objective function $f(\mathbf{a}_1, \ldots, \mathbf{a}_J)$ in (48) with respect to $\mathbf{a}_j$ is given in equation (49). The following Algorithm **??** for solving (48) is deduced from Algorithm **??** by expressing update (**??**) in terms of the original data.

We consider several data matrices $\mathbf{X}_1, \ldots, \mathbf{X}_J$. Each $n \times p_j$ data matrix $\mathbf{X}_j = \left[\mathbf{x}_{j1}, \ldots, \mathbf{x}_{jp_j}\right]$ is called a block and represents a set of p_j variables observed on n individuals. The number and the nature of the variables usually differ from one block to another, but the individuals must be the same across blocks. We assume that all the variables are centered. $\mathbf{S}_{jk} = n^{-1}\mathbf{X}_j^\top \mathbf{X}_k$ is the sample cross-covariance matrix between blocks j and k. The objective of multiblock component methods is to find block components $\mathbf{y}_j = \mathbf{X}_j\mathbf{a}_j, j = 1, \ldots, J$ summarizing relevant information between and within blocks. The sample version of optimization problem (47) is:

$$\underset{\mathbf{a}_1,\dots\mathbf{a}_J}{\text{maximize}} \sum_{j,k,=1}^{J} c_{jk}\text{g}\left(\text{cov}\left(\mathbf{X}_j\mathbf{a}_j, \mathbf{X}_k\mathbf{a}_k\right)\right) \text{ s.t. } \mathbf{a}_j^\top\mathbf{M}_j\mathbf{a}_j = 1, j = 1,\dots,J. \tag{66}$$

Or in terms of the cross-covariance matrices

$$\underset{\mathbf{a}_1,\dots\mathbf{a}_J}{\text{maximize}} \sum_{j,k,=1}^{J} c_{jk}\text{g}\left(\mathbf{a}_j^\top\mathbf{S}_{jk}\mathbf{a}_k\right) \text{ s.t. } \mathbf{a}_j^\top\mathbf{M}_j\mathbf{a}_j = 1, j = 1,\dots,J. \tag{67}$$

Algorithm **??** can be used to solve the optimization problem (67) by replacing the population covariance matrix $\mathbf{\Sigma}_{jk}$ by its empirical counterpart $\mathbf{S}_{jk}$.

We consider in this section sequential RGCCA where the first stage block composites $y_j^{(1)} = \mathbf{a}_j^{(1)\top}\boldsymbol{x}_j, j = 1,\dots,J$ are solution of optimization problem (47). Seeking second stage block composites $y_j^{(2)} = \mathbf{a}_j^{(2)\top}\boldsymbol{x}_j, j = 1,\dots,J$ implies that some constraints must be added to the optimization problem. For example, we may formulate RGCCA at the second stage as the following optimization problem:

$$\underset{\mathbf{a}_1,\dots\mathbf{a}_J}{\text{maximize}} \sum_{j,k=1}^{J} c_{jk}\text{g}\left(\text{cov}\left(\mathbf{a}_j^\top\boldsymbol{x}_j, \mathbf{a}_k^\top\boldsymbol{x}_k\right)\right) \text{ s.t. } \mathbf{a}_j^\top\mathbf{M}_j\mathbf{a}_j = 1 \text{ and } \text{cov}\left(\mathbf{a}_j^\top\boldsymbol{x}_j, y_j^{(1)}\right) = 0, j = 1,\dots,J. \tag{68}$$

The second stage block composites $y_j^{(2)}$ are uncorrelated with the corresponding first stage block composites $y_j^{(1)} = \mathbf{a}_j^{(1)\top}\boldsymbol{x}_j$.

In this section, let $\mathbf{y}_j^{(1)} = \mathbf{X}_j\mathbf{a}_j^{(1)},\ j = 1,\dots,J$ be the first-stage block components solution of optimization problem (66). Seeking the second-stage block components $\mathbf{y}_j^{(2)} = \mathbf{X}_j\mathbf{a}_j^{(2)},\ j = 1,\dots,J$, implies that some constraints must be added to the optimization problem. For example, orthogonality constraints can be considered, leading to the following formulation of the RGCCA optimization problem at the second stage:

$$\begin{gathered}\max_{\mathbf{a}_1,\dots,\mathbf{a}_L} \sum_{j,k=1}^{J} c_{kj}\ \text{g}\left(n^{-1}\mathbf{a}_j^\top\mathbf{X}_j^\top\mathbf{X}_k\mathbf{a}_k\right) \\ \text{s.t. } \mathbf{a}_j^\top\mathbf{M}_j\mathbf{a}_j = 1, \text{ and } \mathbf{a}_j^\top\mathbf{X}_j^\top\mathbf{y}_j^{(1)} = 0,\ j = 1,\dots,J\end{gathered} \tag{69}$$

For each block, the resulting second-stage block component $\mathbf{y}_j^{(2)}$ is orthogonal with the first-stage block component $\mathbf{y}_j^{(1)}$.

Optimization Problem (69) is easy to solve with the first-stage RGCCA algorithm by using a deflation procedure. This procedure consists in replacing a block $\mathbf{X}_j$ by the residual $\mathbf{X}_j^{(1)} = \mathbf{X}_j - \mathbf{y}_j^{(1)}\left(\mathbf{y}_j^{(1)\top}\mathbf{y}_j^{(1)}\right)^{-1}\mathbf{y}_j^{(1)\top}\mathbf{X}_j$ related to the regression of $\mathbf{X}_j$ on the first-stage block component $\mathbf{y}_j^{(1)}$. Moreover, 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 $\mathbf{y}_j$ belonging to the range space of $\mathbf{X}_j^{(1)}$ can also be expressed in term of the original block $\mathbf{X}_j$:

$$\mathbf{y}_j = \mathbf{X}_j^{(1)} \tilde{\mathbf{a}}_j = \mathbf{X}_j \mathbf{a}_j. \tag{70}$$

Furthermore, by assuming that each $\mathbf{X}_j$ is of full-rank, then $\mathbf{a}_j$ can be expressed in terms of $\tilde{\mathbf{a}}_j$: $\mathbf{a}_j = \left(\mathbf{X}_j^\top \mathbf{X}_j\right)^{-1} \mathbf{X}_j^\top \mathbf{X}_j^{(1)} \tilde{\mathbf{a}}_j$. Thus, the constraint $\mathbf{a}_j^\top \mathbf{M}_j \mathbf{a}_j = 1$ can be rewritten in terms of $\tilde{\mathbf{a}}_j$:

$$1 = \mathbf{a}_j^\top \mathbf{M}_j \mathbf{a}_j = \tilde{\mathbf{a}}_j^\top \mathbf{X}_j^{(1)^\top} \mathbf{X}_j \left(\mathbf{X}_j^\top \mathbf{X}_j\right)^{-1} \mathbf{M}_j \left(\mathbf{X}_j^\top \mathbf{X}_j\right)^{-1} \mathbf{X}_j^\top \mathbf{X}_j^{(1)} \tilde{\mathbf{a}}_j \tag{71}$$

Setting $\mathbf{M}_j^{(1)} = \mathbf{X}_j^{(1)^\top} \mathbf{X}_j \left(\mathbf{X}_j^\top \mathbf{X}_j\right)^{-1} \mathbf{M}_j \left(\mathbf{X}_j^\top \mathbf{X}_j\right)^{-1} \mathbf{X}_j^\top \mathbf{X}_j^{(1)}$, optimization problem (69) becomes equivalent to:

$$\begin{aligned} &\max_{\tilde{\mathbf{a}}_1, \ldots, \tilde{\mathbf{a}}_J} \sum_{k,j=1}^{J} c_{jk} \, \mathrm{g}\left(n^{-1} \tilde{\mathbf{a}}_k^\top \mathbf{X}_k^{(1)\top} \mathbf{X}_j^{(1)} \tilde{\mathbf{a}}_j\right) \\ &\text{s.t. } \tilde{\mathbf{a}}_j^\top \mathbf{M}_j^{(1)} \tilde{\mathbf{a}}_j = 1, \; j = 1, \ldots, J \end{aligned} \tag{72}$$

So the first-stage RGCCA algorithm can be used to solve (72) and leads to the block weight vector $\mathbf{a}_j^{(2)}$ and the second-stage block component $\mathbf{y}_j^{(2)} = \mathbf{X}_j^{(1)} \mathbf{a}_j^{(2)}$.

However, maximizing successive criteria may be seen as suboptimal from an optimization point of view where a single global criterion might be preferred.

The goal of this section is to present the optimization framework under which most of the algorithms derived in this document were designed. This framework has already been presented in (Tenenhaus *et al.* 2017) under the example of a spherical constraint. It is recalled here for a broader class of constraints.

1.3. Optimization Problem

The RGCCA package relies on a master algorithm for maximizing a continuously differentiable multi-convex function
$f(\mathbf{a}_1, \ldots, \mathbf{a}_J) : \mathbb{R}^{J_1} \times \ldots \times \mathbb{R}^{J_J} \to \mathbb{R}$ (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 $\Omega_j \subset \mathbb{R}^{J_j}$. This general optimization problem can be formulated as follows:

$$\max_{\mathbf{a}_1, \ldots, \mathbf{a}_J} f(\mathbf{a}_1, \ldots, \mathbf{a}_J) \text{ s.t. } \mathbf{a}_j \in \Omega_j, \; j = 1, \ldots, J. \tag{73}$$

Remark on notations. 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 $J = \sum_{j=1}^{J} J_j$. Moreover, for the vertical concatenation of column vectors, the notation $\mathbf{a} = (\mathbf{a}_1; \ldots; \mathbf{a}_J)$ is preferred for the sake of simplification. This last formulation is also used to define a vertical concatenation of matrices. These notations are used all along this manuscript.

1.4. Algorithm

A simple, monotonically and globally convergent algorithm is presented for maximizing (**??**) subject to (73). An algorithm is globally convergent if, regardless of its initialization, it converges towards a stationary point. For an unconstrained optimization problem with a continuously differentiable objective function, a stationary point is a point where the derivative of the objective function is null. For a constrained optimization problem, a stationary point is a point where the derivative of the Lagrangian function associated with the problem is null. For such a point, the derivative of the objective function lies in the subspace defined by the derivative of each constraint. This condition is called the Karush-Kuhn-Tucker (KKT) condition.

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 cyclic Block Coordinate Ascent (BCA).

In order to do so, 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:

$$f(\mathbf{a}_1, ..., \mathbf{a}_{j-1}, \tilde{\mathbf{a}}_j, \mathbf{a}_{j+1}, \ldots, \mathbf{a}_J) \geq f(\mathbf{a}) + \nabla_j f(\mathbf{a})^\top (\tilde{\mathbf{a}}_j - \mathbf{a}_j) := \ell_j(\tilde{\mathbf{a}}_j, \mathbf{a}) \tag{74}$$

On the right-hand side of the inequality (74), 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 $\ell_j(\tilde{\mathbf{a}}_j, \mathbf{a})$ over $\tilde{\mathbf{a}}_j \in \Omega_j$ is obtained by considering the following optimization problem:

$$\hat{\mathbf{a}}_j = \underset{\tilde{\mathbf{a}}_j \in \Omega_j}{\operatorname{argmax}} \nabla_j f(\mathbf{a})^\top \tilde{\mathbf{a}}_j := r_j(\mathbf{a}). \tag{75}$$

The entire algorithm is subsumed in Algorithm 2.

Algorithm 2 Algorithm for the maximization of a continuously differentiable multi-convex function

```
1: Result: $\mathbf{a}_1^s, \ldots, \mathbf{a}_J^s$ (approximate solution of (??) subject to (73))
2: Initialization: choose random vector $\mathbf{a}_j^0 \in \Omega_j, j = 1, \ldots, J, \epsilon$;
3: $s = 0$ ;
4: repeat
5:   for $j = 1$ to $J$ do
6:       $$\mathbf{a}_j^{s+1} = r_j\left(\mathbf{a}_1^{s+1}, \ldots, \mathbf{a}_{j-1}^{s+1}, \mathbf{a}_j^s, \ldots, \mathbf{a}_J^s\right). \tag{76}$$
7:   end for
8:   $s = s + 1$ ;
9: until $f(\mathbf{a}_1^{s+1}, \ldots, \mathbf{a}_J^{s+1}) - f(\mathbf{a}_1^s, \ldots, \mathbf{a}_J^s) < \varepsilon$
```

1.5. Convergence Properties

To study the convergence properties of Algorithm
2, we introduce some notations: $\Omega = \Omega_1 \times \ldots \times \Omega_J$, $\mathbf{a} = (\mathbf{a}_1; \ldots; \mathbf{a}_L) \in \Omega$, $c_l \ : \ \Omega \mapsto \Omega$ is

an operator defined as $c_l(\mathbf{a}) = (\mathbf{a}_1; \ldots; \mathbf{a}_{j-1}; r_j(\mathbf{a}); \mathbf{a}_{j+1}; \ldots; \mathbf{a}_J)$ with $r_l(\mathbf{a})$ introduced in equation (75) and $c : \Omega \mapsto \Omega$ is defined as $c = c_L \circ c_{L-1} \circ ... \circ c_1$, where $\circ$ stands for the function composition operator. We consider the sequence $\{\mathbf{a}^s = (\mathbf{a}_1^s; \ldots; \mathbf{a}_J^s)\}$ generated by Algorithm 2. Using the operator c, the «for loop» inside Algorithm 2 can be replaced by the following recurrence relation: $\mathbf{a}^{s+1} = c(\mathbf{a}^s)$. The convergence properties of Algorithm 2 are summarized in the following proposition:

Proposition 1.5. *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 Ω. This result implies the monotonic convergence of Algorithm 2.

(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 (**??**).

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

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

(e) The sequence $\{\mathbf{a}^s = (\mathbf{a}_1^s; \ldots; \mathbf{a}_J^s)\}$, $l = 1, \ldots, L$, is asymptotically regular: $\lim_{s \to \infty} \sum_{l=1}^{L} \|\mathbf{a}_j^{s+1} - \mathbf{a}_j^s\| = 0$. This result implies that if the threshold ε for the stopping criterion in Algorithm 2 is made sufficiently small, the output of Algorithm 2 will be as close as wanted to a stationary point of (**??**).

(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.

The goal is to demonstrate Proposition 1.5 that gathers all the convergence properties of Algorithm 2. For this purpose, the results given in the following lemma are useful.

Lemma 1.6. *Consider the set Ω, the function $f : \Omega \mapsto \mathbb{R}$ and the operator $c : \Omega \mapsto \Omega$ defined above. Then, the following properties hold:*

(i) Ω is a compact set;

(ii) c is a continuous operator;

(iii) $f(\mathbf{a}) \leq f(c(\mathbf{a}))$ for any $\mathbf{a} \in \Omega$;

(iv) If $f(\mathbf{a}) = f(c(\mathbf{a}))$, then $c(\mathbf{a}) = \mathbf{a}$.

Proof of Lemma 1.6. Point *(i)* In this section, $\forall l$, Ω_j are assumed to be compact. As the Cartesian product of L compact sets is compact, $\Omega = \Omega_1 \times \ldots \times \Omega_J$ is compact.

Point *(ii)*

We assume that $r_l(\mathbf{a})$ defined in equation (75) exists and is unique. As Ω_j is a compact set and l_l defined in equation (74) is a real-valued continuous function, Berge's maximum theorem applies and guarantees that the maximizer $r_l(\mathbf{a})$ of $l_l(\tilde{\mathbf{a}}_l, \mathbf{a})$ is continuous on Ω_j (**?**). This implies that $c_l : \Omega \to \Omega$ is a continuous operator and that $c = c_L \circ c_{L-1} \circ \ldots \circ c_1$ is also continuous as composition of L continuous operators.

Point *(iii)* According to equation (74) based on multi-convexity of f and equation (75) that sets the definition of $r_l : \Omega \mapsto \Omega_j$, we know that:

$$f(\mathbf{a}) = \ell_l(\mathbf{a}_j, \mathbf{a}) \leq \ell_l(r_l(\mathbf{a}), \mathbf{a}) \leq f(\mathbf{a}_1, \ldots, \mathbf{a}_{l-1}, r_l(\mathbf{a}), \mathbf{a}_{l+1}, \ldots, \mathbf{a}_J) = f(c_l(\mathbf{a})). \quad (77)$$

This implies that updating $\mathbf{a}_j$ by $\hat{\mathbf{a}}_l = r_l(\mathbf{a})$ increases $f(\mathbf{a})$, or $f(\mathbf{a})$ stays the same. Moreover, the following inequality is deduced from (77) for each $l = 2, \dots, L$:

$$f(c_{l-1} \circ \ldots \circ c_1(\mathbf{a})) \leq f(c_l \circ c_{l-1} \circ \ldots \circ c_1(\mathbf{a})). \tag{78}$$

This yields the desired inequalities for any $\mathbf{a} \in \Omega$:

$$f(\mathbf{a}) \leq f(c_1(\mathbf{a})) \leq f(c_2 \circ c_1(\mathbf{a})) \leq \ldots \leq f(c_L \circ \ldots \circ c_1(\mathbf{a})) = f(c(\mathbf{a})). \tag{79}$$

Point *(iv)* If $f(\mathbf{a}) = f(c(\mathbf{a}))$ for $\mathbf{a} \in \Omega$ then equation (79) implies

$$f(\mathbf{a}) = f(c_1(\mathbf{a})) = f(c_2 \circ c_1(\mathbf{a})) = \ldots = f(c_L \circ \ldots \circ c_1(\mathbf{a})) = f(c(\mathbf{a})). \tag{80}$$

Using equation (77), the equality $f(\mathbf{a}) = f(c_1(\mathbf{a}))$ implies $\ell_1(\mathbf{a}_1, \mathbf{a}) = \ell_1(r_1(\mathbf{a}), \mathbf{a})$ and therefore $\mathbf{a}_1 = r_1(\mathbf{a})$ as $r_1(\mathbf{a})$ is the unique maximizer of $\ell_1(r_1(\tilde{\mathbf{a}}_1), \mathbf{a})$ with respect to $\tilde{\mathbf{a}}_1 \in \Omega_1$. From this result, we deduce $\mathbf{a} = (\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_J) = (r_1(\mathbf{a}), \mathbf{a}_2, \dots, \mathbf{a}_J) = c_1(\mathbf{a})$ and then, by transitivity,

$$\mathbf{a} = c_1(\mathbf{a}) = c_2 \circ c_1(\mathbf{a}) = \ldots = c_L \circ \ldots \circ c_1(\mathbf{a}) = c(\mathbf{a}). \tag{81}$$

□

Proof of Proposition 1.5. Point *(a)* Point (iii) of Lemma 1.6 implies that the sequence $f(\mathbf{a}^s)$ is monotonically increasing, and therefore, convergent as the continuous function f is bounded on the compact set Ω.

Point *(b)* If the infinite sequence $f(\mathbf{a}^s)$ has a finite number of distinct terms, it cannot be a strictly increasing sequence and consequently there exists some integer M such that $f(\mathbf{a}^0) < f(\mathbf{a}^1) < \ldots < f(\mathbf{a}^M) = (\mathbf{a}^{M+1})$. Then, Point (iv) of Lemma 1.6 implies that $\mathbf{a}^M$ is a fixed point of c.

Point *(c) to (f)* They are deduced from a direct application of Meyer's monotone convergence theorem (Theorem 3.1 in (Meyer 1976)). This theorem gives quite general conditions under which a sequence $(\mathbf{a}^s)$ produced by an algorithm that monotonically increases a continuous objective function will converge. Meyer considered the case of a point-to-set operator $c : \Omega \mapsto \mathcal{P}(\Omega)$, where $\mathcal{P}(\Omega)$ is the set of all nonempty subsets of Ω. In this manuscript, c is a point-to-point operator and the conditions of Meyer's theorem reduce to the three following conditions (see (**?**)): (1) c is a continuous operator; (2) c is strictly monotone (increasing) with respect to f ; and (3) c is uniformly compact on Ω. Condition (2) means that points (iii) and (iv) of Lemma 1.6 are verified. Condition (3) means that there exists a compact set $\mathcal{K}$ such that $c(\mathbf{a}) \in \mathcal{K}$ for all $\mathbf{a} \in \Omega$. According to Lemma 1.6, these three conditions are satisfied for Algorithm 2 and therefore, Meyer's theorem can be applied to any sequence $\mathbf{a}^s$ produced by the recurrence equation $\mathbf{a}^{s+1} = c(\mathbf{a}^s)$ with $\mathbf{a}^0 \in \Omega$.

□

A specific instantiation of this quite general optimization problem has been first introduced in Tenenhaus *et al.* (2017).

Let $\mathbf{X}_1, \dots, \mathbf{X}_l, \dots, \mathbf{X}_L$ be a collection of L data matrices. Each $I \times J_l$ data matrix $\mathbf{X}_l = [\mathbf{x}_{l1}, \dots, \mathbf{x}_{lJ_l}]$ is a block and represents a set of J_l variables observed on I 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 the RGCCA optimization problem (Tenenhaus *et al.* 2017) is:

$$\begin{aligned} &\max_{\mathbf{w}_1,\dots,\mathbf{w}_L} \sum_{k,l=1}^{L} c_{kl}\ \mathrm{g}\left(I^{-1}\mathbf{w}_k^\top \mathbf{X}_k^\top \mathbf{X}_l \mathbf{w}_l\right) \\ &\text{s.t. } \mathbf{w}_l^\top \mathbf{M}_l \mathbf{w}_l = 1,\ l = 1,\dots,L \end{aligned} \tag{82}$$

where g, $\mathbf{C} \in \mathbb{R}^{L\times L}$ and $\mathbf{M}_l \in \mathbb{R}^{J_l\times J_l}$, $l = 1,\dots L$ are defined in Chapter **??**, section **??**. The optimization problem (82) can be simplified by considering the two following transforms $\mathbf{P}_l = I^{-1/2}\mathbf{X}_l\mathbf{M}_l^{-1/2}$ and $\mathbf{a}_j = \mathbf{M}_l^{1/2}\mathbf{w}_l$, which leads to:

$$\max_{\mathbf{a}_1,\dots,\mathbf{a}_J} f(\mathbf{a}_1,\dots,\mathbf{a}_J) = \sum_{k,l=1}^{L} c_{lk}\ \mathrm{g}\left(\mathbf{a}_k^\top \mathbf{P}_k^\top \mathbf{P}_l \mathbf{a}_j\right) \tag{83}$$

$$\text{s.t. } \mathbf{a}_j^\top \mathbf{a}_j = 1, l = 1,\dots,L. \tag{84}$$

1.6. The RGCCA algorithm

The convexity and continuous differentiability of the scheme function g imply that the objective function f defined in (83) is a continuously differentiable multi-convex[1] function. Consequently, the maximization of (83) subject to (84) can be cast under the general optimization framework presented in section **??**. Under this framework, the function f, defined in equation (83) over different parameter vectors $(\mathbf{a}_1,\dots,\mathbf{a}_J)$, is maximized by updating each of the parameter vectors in turn, keeping the others fixed. Hence, we want to find an update $\hat{\mathbf{a}}_l \in \Omega_j = \left\{\mathbf{a}_j \in \mathbb{R}^{J_l}; \|\mathbf{a}_j\|_2 = 1\right\}$ such that $f(\mathbf{a}) \le f(\mathbf{a}_1,\dots,\mathbf{a}_{l-1},\hat{\mathbf{a}}_l,\mathbf{a}_{l+1},\dots,\mathbf{a}_J)$, where $\mathbf{a} = (\mathbf{a}_1;\dots;\mathbf{a}_J)$. Following section **??**, this update is obtained by considering the following optimization problem:

$$\hat{\mathbf{a}}_l = \underset{\tilde{\mathbf{a}}_l\in\Omega_j}{\operatorname{argmax}} \nabla_j f(\mathbf{a})^\top \tilde{\mathbf{a}}_l = \frac{\nabla_j f(\mathbf{a})}{\|\nabla_j f(\mathbf{a})\|_2} := r_l(\mathbf{a}), \tag{85}$$

where $\nabla_j f(\mathbf{a})$ is the partial gradient of $f(\mathbf{a})$ with respect to $\mathbf{a}_j$:

$$\nabla_j f(\mathbf{a}) = 2\sum_{k=1}^{L} c_{lk}\ \mathrm{g}'(\mathbf{a}_j{}^\top \mathbf{P}_l^\top \mathbf{P}_k \mathbf{a}_k)\mathbf{P}_l^\top \mathbf{P}_k \mathbf{a}_k = \mathbf{P}_l^\top \mathbf{z}_l \tag{86}$$

where $\mathbf{z}_l$, called the inner component, is defined as $\mathbf{z}_l = 2\sum_{k=1}^{L} c_{lk}\ \mathrm{g}'(\mathbf{a}_j{}^\top \mathbf{P}_l^\top \mathbf{P}_k \mathbf{a}_k)\mathbf{P}_k\mathbf{a}_k$.

The entire RGCCA algorithm is subsumed in Algorithm **??**.

At the end of the Algorithm **??**, the original weight vectors $\mathbf{w}_l$ are recovered by $\mathbf{w}_l = (\mathbf{M}_l)^{-1/2}\mathbf{a}_j$.

In the case of a single block ($L = 1$), Algorithm **??** is similar to the gradient-based algorithm proposed by (Journée *et al.* 2010) for maximizing a convex function of several variables with spherical constraints (see Problem 27, p. 529).

[1]When one element of the diagonal of the design matrix $\mathbf{C}$ is equal to 1, additional conditions have to be imposed on the scheme function g in order for f to still be multi-convex. For example, when g is twice differentiable, a sufficient condition is that $\forall x \in \mathbb{R}_+,\ g'(x) \ge 0$. All scheme functions g considered in this document respect this condition and the case where one element of the diagonal of the design matrix $\mathbf{C}$ is equal to 1 is never considered in our examples.

1.7. Convergence properties of the RGCCA algorithm

The convergence properties subsumed in Proposition 1.5 are satisfied for Algorithm **??**. In order to show that Proposition 1.5 holds for the RGCCA algorithm, point (i-iv) of Lemma 1.6 are demonstrated below.

Proof of Lemma 1.6 for the RGCCA Algorithm. Point *(i)* $\Omega_j = \left\{\mathbf{a}_j \in \mathbb{R}^{J_l}; \|\mathbf{a}_j\|_2 = 1\right\}$ is the ℓ_2-sphere of radius 1 and is a compact set. As $\Omega = \Omega_1 \times \ldots \times \Omega_J$ is the Cartesian product of L compact sets, it is compact.

Point *(ii)*

$r_l(\mathbf{a})$ defined in equation (85) is the orthogonal projection of $\nabla_j f(\mathbf{a})$ onto the ℓ_2-sphere of radius 1. Under the assumption made in Chapter **??** section **??** paragraph 3, $r_l(\mathbf{a})$ exists and is unique.

Point *(iii)* The demonstration presented in Chapter **??** for point *(iii)* of Lemma 1.6 still holds here. The proof is based on the multi-convexity of the objective function and on the fact that the update function $r_l(\mathbf{a})$ defined in equation (85) increases the value of the criterion. These two points are satisfied for RGCCA.

Point *(iv)* The proof is based on the uniqueness of $r_j(\mathbf{a})$ defined in equation (85).

□

Therefore, whatever the starting point, Algorithm **??** converges towards a stationary point of the RGCCA optimization problem.

1.8. Higher-stage RGCCA block component

The optimization problem (82) is associated with the first component of RGCCA. A deflation procedure was proposed in order to extract more than one component.

In this section, let $\mathbf{y}_l^{(1)} = \mathbf{X}_l\mathbf{w}_l^{(1)},\ l = 1, \ldots, L$ be the first-stage block components solution of optimization problem (82). Seeking the second-stage block components $\mathbf{y}_l^{(2)} = \mathbf{X}_l\mathbf{w}_l^{(2)},\ l = 1, \ldots, L$, implies that some constraints must be added to the optimization problem. For example, orthogonality constraints can be considered, leading to the following formulation of the RGCCA optimization problem at the second stage:

$$\begin{aligned} &\max_{\mathbf{w}_1,\ldots,\mathbf{w}_L} \sum_{k,l=1}^{L} c_{kl}\, \mathrm{g}\left(I^{-1}\mathbf{w}_k^\top \mathbf{X}_k^\top \mathbf{X}_l \mathbf{w}_l\right) \\ &\text{s.t. } \mathbf{w}_l^\top \mathbf{M}_l \mathbf{w}_l = 1, \text{ and } \mathbf{w}_l^\top \mathbf{X}_l^\top \mathbf{y}_l^{(1)} = 0,\ l = 1, \ldots, L \end{aligned} \tag{87}$$

For each block, the resulting second-stage block component $\mathbf{y}_l^{(2)}$ is uncorrelated with the first-stage block component $\mathbf{y}_l^{(1)}$.

Problem (87) is easy to solve with the first-stage RGCCA algorithm by using a deflation procedure. This procedure consists in replacing a block $\mathbf{X}_l$ by the residual $\mathbf{X}_l^{(1)} = \mathbf{X}_l - \mathbf{y}_l^{(1)}\left(\mathbf{y}_l^{(1)\top}\mathbf{y}_l^{(1)}\right)^{-1}\mathbf{y}_l^{(1)\top}\mathbf{X}_l$ related to the regression of $\mathbf{X}_l$ on the first-stage block component $\mathbf{y}_l^{(1)}$. Moreover, as $\mathbf{y}_l^{(1)} = \mathbf{X}_l\mathbf{w}_l^{(1)}$, the range space of $\mathbf{X}_l^{(1)}$ is included in the range space of $\mathbf{X}_l$, meaning that any block component $\mathbf{y}_l$ belonging to the range space of $\mathbf{X}_l^{(1)}$ can also be expressed in term of the original block $\mathbf{X}_l$:

$$\mathbf{y}_l = \mathbf{X}_l^{(1)} \tilde{\mathbf{w}}_l = \mathbf{X}_l \mathbf{w}_l. \tag{88}$$

Furthermore, by assuming that each $\mathbf{X}_l$ is of full-rank, then $\mathbf{w}_l$ can be expressed in terms of $\tilde{\mathbf{w}}_l$: $\mathbf{w}_l = \left(\mathbf{X}_l^\top \mathbf{X}_l\right)^{-1} \mathbf{X}_l^\top \mathbf{X}_l^{(1)} \tilde{\mathbf{w}}_l$. Thus, the constraint $\mathbf{w}_l^\top \mathbf{M}_l \mathbf{w}_l = 1$ can be rewritten in terms of $\tilde{\mathbf{w}}_j$:

$$1 = \mathbf{w}_l^\top \mathbf{M}_l \mathbf{w}_l = \tilde{\mathbf{w}}_l^\top \mathbf{X}_l^{(1)^\top} \mathbf{X}_l \left(\mathbf{X}_l^\top \mathbf{X}_l\right)^{-1} \mathbf{M}_l \left(\mathbf{X}_l^\top \mathbf{X}_l\right)^{-1} \mathbf{X}_l^\top \mathbf{X}_l^{(1)} \tilde{\mathbf{w}}_l \tag{89}$$

Setting $\mathbf{M}_l^{(1)} = \mathbf{X}_l^{(1)^\top} \mathbf{X}_l \left(\mathbf{X}_l^\top \mathbf{X}_l\right)^{-1} \mathbf{M}_l \left(\mathbf{X}_l^\top \mathbf{X}_l\right)^{-1} \mathbf{X}_l^\top \mathbf{X}_l^{(1)}$, optimization problem (87) becomes equivalent to:

$$\begin{aligned} &\max_{\tilde{\mathbf{w}}_1,\ldots,\tilde{\mathbf{w}}_L} \sum_{k,l=1}^{L} c_{kl}\, \mathrm{g}\left(I^{-1} \tilde{\mathbf{w}}_k^\top \mathbf{X}_k^{(1)^\top} \mathbf{X}_l^{(1)} \tilde{\mathbf{w}}_l\right) \\ &\text{s.t. } \tilde{\mathbf{w}}_l^\top \mathbf{M}_l^{(1)} \tilde{\mathbf{w}}_l = 1,\ l = 1, \ldots, L \end{aligned} \tag{90}$$

So the first-stage RGCCA algorithm can be used to solve (90) and leads to the block weight vector $\tilde{\mathbf{w}}_l^{(2)}$ and the second-stage block component $\mathbf{y}_l^{(2)} = \mathbf{X}_l^{(1)} \tilde{\mathbf{w}}_l^{(2)}$. This deflation procedure can be iterated in a very flexible way. For example, in a supervised situation where we want to predict a block based on other blocks, it might be interesting to apply this deflation procedure to all blocks except the one to predict.

However, maximizing successive criteria may be seen as suboptimal from an optimization point of view where a single global criterion might be preferred. Secondly, with this sequential procedure, if the first components are poorly estimated, this is going to affect the estimation of the following components, which is a major drawback. Thirdly, as seen previously, we have to assume that each block matrix $\mathbf{X}_l$ is of full-rank in order to properly define the constraint matrix $\mathbf{M}_l^{(1)}$. Nonetheless, this is not always true.

To improve the interpretability of the RGCCA model, an important task is to identify subsets of variables within each block that are active in the relationships among blocks. This variable selection step can be achieved by adding within the RGCCA optimization process different kinds of penalty promoting sparsity (ℓ_1) or structured sparsity (like group LASSO, sparse group, fused or elitist LASSO penalty). In general, one might think to penalize the ℓ_0-pseudo-norm of the block weight vector to enforce variable selection. However, the resulting optimization problem becomes really hard to solve due to the combinatorial properties of the ℓ_0-pseudo-norm and its non-convexity. A relaxation of this problem was proposed by replacing the ℓ_0-pseudo-norm by its tightest convex envelop (**?**), the ℓ_1-norm: $\|\mathbf{x}\|_1 = \sum_{j=1}^{J} |x_j|$. An ℓ_1-norm was added to the RGCCA optimization problem, this algorithm is called Sparse Generalized Canonical Correlation Analysis (SGCCA) (**?**). At the heart of the SGCCA algorithm lies an optimization problem under both an ℓ_1 and an ℓ_2-norm constraint. Section **??**, presents a new procedure for solving this problem. The convergence properties of SGCCA are also studied.

2. Sparse Generalized Canonical Correlation Analysis (SGCCA)

A collection of L data matrices $\mathbf{X}_1, \ldots, \mathbf{X}_l, \ldots, \mathbf{X}_L$ is introduced. Each $I \times J_l$ data matrix $\mathbf{X}_l = [\mathbf{x}_{l1}, \ldots, \mathbf{x}_{lJ_l}]$ is a set of J_l variables observed on I 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.

A sparse version of RGCCA called SGCCA (**?**) was proposed to add an ℓ_1-norm constraint to the weights in order to perform variable selection. The optimization criterion of SGCCA can be written as:

$$\begin{aligned} \max_{\mathbf{w}_1,\ldots,\mathbf{w}_L} f(\mathbf{w}_1,\ldots,\mathbf{w}_L) &= \sum_{k,l=1}^{L} c_{kl}\, \mathrm{g}\left(I^{-1}\mathbf{w}_k^\top \mathbf{X}_k^\top \mathbf{X}_l \mathbf{w}_l\right) \\ \text{s.t. } \mathbf{w}_l &\in \Omega_j,\ l = 1,\ldots,L, \end{aligned} \tag{91}$$

where function g, and the design matrix $\mathbf{C} \in \mathbb{R}^{L\times L}$ are defined in chapter **??** section **??**.

In the original presentation of SGCCA , $\Omega_j = \{\mathbf{w}_l \in \mathbb{R}^{I\times J_l}; \|\mathbf{w}_l\|_2 = 1; \|\mathbf{w}_l\|_1 \leq s_l\}$, with $s_l \in \mathbb{R}_+^\star$. Here, in order to ease the convergence study of SGCCA, a slighty different set is considered:

$$\Omega_j = \{\mathbf{w}_l \in \mathbb{R}^{I\times J_l}; \|\mathbf{w}_l\|_2 \leq 1; \|\mathbf{w}_l\|_1 \leq s_l\}. \tag{92}$$

This set is defined as the intersection between the ℓ_2-ball of radius 1 and the ℓ_1-ball of radius $s_l \in \mathbb{R}_+^\star$ which are two convex sets. Hence, Ω_j is a convex set.

In comparison to RGCCA, $\mathbf{M}_l,\ l = 1,\ldots,L$, are all set to the identity in SGCCA optimization problem.

In the rest of this chapter, the assumption is made that the ℓ_2-ball of radius 1 is not included in the ℓ_1-ball of radius s_l and the other way round. Otherwise systematically, only one of the two constraints is active. This assumption is true when the corresponding spheres intersect. When $J_l = 2$, the two borderline cases are shown on Figure **??**. This assumption can be translated into conditions on s_l.

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

$$\forall \mathbf{x} \in \mathbb{R}^{J_l},\ \|\mathbf{x}\|_2 \leq \|\mathbf{x}\|_1 \leq \sqrt{J_l}\|\mathbf{x}\|_2. \tag{93}$$

This can be converted into a condition on s_l: $1 \leq s_l \leq \sqrt{J_l}$. When such condition is fulfilled, the ℓ_2-sphere of radius 1 and the ℓ_1-sphere of radius s_l intersect. This is depicted in figure **??**.

2.1. The SGCCA Algorithm

SGCCA and RGCCA have the same objective function so the general optimization framework presented in section **??** applies for SGCCA. Hence, under this framework, the update defined in equation (85) for RGCCA can be used again here but with Ω_j defined in (92). This update tries to find $\hat{\mathbf{w}}_l \in \Omega_j$ obtained by considering the optimization problem below:

$$\hat{\mathbf{w}}_l = \underset{\substack{\|\tilde{\mathbf{w}}_l\|_2 \leq 1 \\ \|\tilde{\mathbf{w}}_l\|_1 \leq s_l}}{\operatorname{argmax}} \nabla_j f(\mathbf{w})^\top \tilde{\mathbf{w}}_l := r_j(\mathbf{w}) \tag{94}$$

where $\mathbf{w} = (\mathbf{w}_1;\ldots;\mathbf{w}_L)$ and $\nabla_j f(\mathbf{w})$ is the partial gradient of f with respect to $\mathbf{w}_l$ that can be found in equation (86).

According to (**?**), solution of (94) satisfies:

$$r_j(\mathbf{w}) = \hat{\mathbf{w}}_l = \frac{\mathcal{S}(\nabla_j f(\mathbf{w}), \lambda_j)}{\|\mathcal{S}(\nabla_j f(\mathbf{w}), \lambda_j)\|_2}, \text{ where } \lambda_j = \left\{ \begin{array}{cll} 0 \text{ if} & \frac{\|\nabla_j f(\mathbf{w})\|_1}{\|\nabla_j f(\mathbf{w})\|_2} & \leq s_l \\ \text{find } \lambda_j \text{ such that} & \|\hat{\mathbf{w}}_l\|_1 & = s_l \end{array} \right. , \quad (95)$$

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

$$\mathbf{u} = \mathcal{S}(\mathbf{x}, \lambda) \Leftrightarrow u_j = \left\{ \begin{array}{cll} \text{sign}(x_j)(|x_j| - \lambda), & \text{if } |x_j| & > \lambda \\ 0, & \text{if } |x_j| & \leq \lambda \end{array} \right. , j = 1, \dots, J. \quad (96)$$

In the case of a scalar $a \in \mathbb{R}$, Figure **??** depicts the function $\mathcal{S}(., \lambda)$ with $\lambda = 1$.

The entire SGCCA Algorithm is presented in Algorithm **??**.

The next section proposes an algorithm to solve problem (95) at the heart of SGCCA.

2.2. The update function of SGCCA

This section focuses on the update of the SGCCA algorithm presented in equation (94). This problem can be stated as follows:

$$\underset{\mathbf{x} \in \Omega}{\text{argmax}} \, \mathbf{a}^\top \mathbf{x}, \quad (97)$$

where $\mathbf{a} \in \mathbb{R}^J$ and $\Omega = \left\{ \mathbf{x} \in \mathbb{R}^J \mid \|\mathbf{x}\|_2 \leq 1 \text{ and } \|\mathbf{x}\|_1 \leq s \right\}$ with $s \in \mathbb{R}_+^\star$. As mentioned above, the solution of (97) satisfies $\mathbf{u} = \mathcal{S}(\mathbf{a}, \lambda)/\|\mathcal{S}(\mathbf{a}, \lambda)\|_2$, where $\lambda = 0$ if $\|\mathbf{a}\|_1/\|\mathbf{a}\|_2 \leq s$ and λ is chosen such that $\|\mathbf{u}\|_1 = s$ otherwise.

Several strategies such as Binary Search or the Projection On Convex Set algorithm (POCS), also known as alternating projection method (**?**), can be used to determine λ verifying the ℓ_1-norm constraint. Here, an alternative approach inspired by (**?**) is proposed. In the entire section we assume that the maximum value of the vector $|\mathbf{a}|$ is reached for only one element. Appendix **??**, section **??** justifies the importance of this assumption.

3. Multiblock data analysis with the RGCCA package

For the sake of comprehension of the use of the RGCCA package, the theoretical foundations of RGCCA and variations - that were previously published (Tenenhaus and Tenenhaus 2011, ; Tenenhaus and Tenenhaus 2014, ; Tenenhaus, Philippe, and Frouin 2015, ; Tenenhaus *et al.* 2017) - are briefly summarized.

Let us consider J random p_j-dimensional centered column vectors $\boldsymbol{x}_j$ and J non random p_j-dimensional column vectors $\boldsymbol{\xi}_j$. Moreover, let $\mathbf{C} = (c_{jk})$ be a $J \times J$ symmetric design matrix of nonnegative entries that indicate the strength of the relations between $\boldsymbol{x}_j$ and $\boldsymbol{x}_k$. Usually, $c_{jk} = 1$ for two connected blocks and 0 otherwise. Now consider two linear combinations $\eta_j = \boldsymbol{\xi}_j^\top \boldsymbol{x}_j$ and $\eta_k = \boldsymbol{\xi}_k^\top \boldsymbol{x}_k$. Generalized Canonical Correlation Analysis is defined as the following optimization problem:

$$\underset{\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \dots, \boldsymbol{\xi}_J}{\text{maximize}} \sum_{j,k=1}^{J} c_{jk} \, \text{g} \left(\text{cov}(\boldsymbol{\xi}_j^\top \boldsymbol{x}_j, \boldsymbol{\xi}_k^\top \boldsymbol{x}_k) \right) \text{ s.t. } \text{var}(\boldsymbol{\xi}_j^\top \boldsymbol{x}_j) = 1, j = 1, \dots, J \quad (98)$$

Generalized Canonical Correlation Analysis aims at extracting the information which is shared by the J random variables taking into account an undirected graph of connections between these random variables. A sample-based optimization problem related to (98) 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 (A row of $\mathbf{X}_j$ represents a realization of the row-random vector $\boldsymbol{x}_j^\top$). The main aim is to investigate the relationships between blocks.

Regularized Generalized Canonical Correlation Analysis (RGCCA) introduced in (Tenenhaus and Tenenhaus 2011) is defined as the empirical counterpart of optimization problem (98) and is defined as follows:

$$\underset{\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_J}{\text{maximize}} \sum_{j,k=1}^{J} c_{jk} g\left(\mathbf{w}_j^\top \widehat{\boldsymbol{\Sigma}}_{jk} \mathbf{w}_k\right) \text{ s.t. } \mathbf{w}_j^\top \widehat{\boldsymbol{\Sigma}}_{jj} \mathbf{w}_j = 1, j = 1, \ldots, J \qquad (99)$$

where $\widehat{\boldsymbol{\Sigma}}_{jj}$ is an estimate of the intra-block covariance matrix $\boldsymbol{\Sigma}_{jj} = \mathbb{E}[\boldsymbol{x}_j \boldsymbol{x}_j^\top]$ and $\widehat{\boldsymbol{\Sigma}}_{jk} = n^{-1}\mathbf{X}_j^\top \mathbf{X}_k$ is an estimate of the inter-block covariance matrix $\boldsymbol{\Sigma}_{jk} = \mathbb{E}[\boldsymbol{x}_j \boldsymbol{x}_k^\top]$. In cases involving multi-collinearity within blocks or 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 (99) 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 τ_j can be obtained using various analytical formulae (see for instance Ledoit and Wolf (2004); Schäfer and Strimmer (2005); Chen, Wiesel, and Hero (2011)).

We consider J data matrices $\mathbf{X}_1, \ldots, \mathbf{X}_J$. Each $n \times p_j$ data matrix $\mathbf{X}_j = [\mathbf{x}_{j1}, \ldots, \mathbf{x}_{jp_j}]$ is called a block and represents a set of p_j variables observed on n individuals. The number and the nature of the variables may differ from one block to another, but the individuals must be the same across blocks. We assume that all variables are centered. The objective of RGCCA is to find, for each block, a weighted composite of variables (called block component) $\mathbf{y}_j = \mathbf{X}_j \mathbf{a}_j, j = 1, \ldots, J$ (where $\mathbf{a}_j$ is a column-vector with p_j elements) summarizing the relevant information between and within the blocks. The block components are obtained such that (i) block components explain well their own block and/or (ii) block components that are assumed to be connected are highly correlated.

3.1. Regularized Generalized Canonical Correlation Analysis

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:

$$\underset{\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_J}{\text{maximize}} \sum_{j,k=1}^{J} c_{jk} g(\text{cov}(\mathbf{X}_j \mathbf{a}_j, \mathbf{X}_k \mathbf{a}_k)) \text{ s.t. } (1-\tau_j)\text{var}(\mathbf{X}_j \mathbf{a}_j) + \tau_j \|\mathbf{a}_j\|^2 = 1, j = 1, \ldots, J \quad (100)$$

where:

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

- The design matrix $\mathbf{C}$ is a symmetric $J \times J$ matrix of nonnegative elements describing the network of connections between blocks that the user wants to take into account. Usually, $c_{jk} = 1$ for two connected blocks and 0 otherwise.

- The τ_j are called shrinkage parameters ranging from 0 to 1 and interpolate smoothly between maximizing the covariance and maximizing the correlation. Setting the τ_j to 0 will force the block components to unit variance ($\mathrm{var}(\mathbf{X}_j\mathbf{a}_j = 1)$), in which case the covariance criterion boils down to the correlation. The correlation criterion is better in explaining the correlated structure across datasets, thus discarding the variance within each individual dataset. Setting τ_j to 1 will normalize the block weight vectors ($\mathbf{a}_j^\top\mathbf{a}_j = 1$), which applies the covariance criterion. A value between 0 and 1 will lead to a compromise between the two first options and correspond to the following constraint $(1 - \tau_j)\mathrm{var}(\mathbf{X}_j\mathbf{a}_j) + \tau_j\|\mathbf{a}_j\|^2 = 1$ in (100). The choices $\tau_j = 1$, $\tau_j = 0$ and $0 < \tau_j < 1$ are respectively referred as Modes A, B and Ridge. In the RGCCA package, for each block, the determination of the shrinkage parameter can be made fully automatic by using the analytical formula proposed by (Schäfer and Strimmer 2005). Also, depending on the context, the shrinkage parameters should also be determined based on cross-validation or permutation. We can define the choice of the shrinkage parameters by providing interpretations on the properties of the resulting block components:

  - $\tau_j = 1$ yields the maximization of a covariance-based criterion. It is recommended when the user wants a stable component (large variance) while simultaneously taking into account the correlations between blocks. The user must, however, be aware that variance dominates over correlation.
  - $\tau_j = 0$ yields the maximization of a correlation-based criterion. It is recommended when the user wants to maximize correlations between connected components. This option can yield unstable solutions in case of multi-collinearity and cannot be used when a data block is rank deficient (e.g. $n < p_j$).
  - $0 < \tau_j < 1$ is a good compromise between variance and correlation: the block components are simultaneously stable and as well correlated as possible with their connected block components. This setting can be used when the data block is rank deficient.

From optimization problem (100), 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 gives the correspondences between the triplet $g, \tau_j, \mathbf{C}$) and the mutliblock component methods. For a complete overview see (Tenenhaus *et al.* 2017).

Table 1: Multiblock component methods as special cases of RGCCA. When τ_{J+1} is introduced, it is implicitly assumed that $\mathbf{X}_1, \ldots, \mathbf{X}_J$ are connected to the superblock $\mathbf{X}_{J+1} = \left[\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_J\right]$ and that τ_{J+1} corresponds to the shrinkage parameter associated with $\mathbf{X}_{J+1}$.

Methods	$g(x)$	τ_j	$\mathbf{C}$
Canonical Correlation Analysis (Hotelling 1936)	x	$\tau_1 = \tau_2 = 0$	$\mathbf{C}_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
Interbattery Factor Analysis (Tucker 1958)	x	$\tau_1 = \tau_2 = 1$	$\mathbf{C}_1$
Redundancy Analysis (Van den Wollenberg 1977)	x	$\tau_1 = 1\,;\, \tau_2 = 0$	$\mathbf{C}_1$
SUMCOR (Horst 1961b)	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	$\tau_j = 1, j = 1, \ldots, J$	$\mathbf{C}_3$
Generalized CCA (Carroll 1968a)	x^2	$\tau_j = 0, j = 1, \ldots, J+1$	$\mathbf{C}_4 = \begin{pmatrix} 0 & \cdots & 0 & 1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 1 \\ 1 & \cdots & 1 & 0 \end{pmatrix}$
Generalized CCA (Carroll 1968b)	x^2	$\tau_j = 0, j = 1, \ldots, J_1\,;$ $\tau_j = 1, j = J_1, \ldots, J$	$\mathbf{C}_4$
Hierarchical PCA (Wold, S. and Kettaneh, N. and Tjessem, K. 1996)	x^4	$\tau_j = 1, j = 1, \ldots, J\,;$ $\tau_{J+1} = 0$	$\mathbf{C}_4$
Multiple Co-Inertia Analysis (Chessel and Hanafi 1996)	x^2	$\tau_j = 1, j = 1, \ldots, J\,;$ $\tau_{J+1} = 0$	$\mathbf{C}_4$

Methods	$g(x)$	τ_j	**C**
PLS path modeling - mode B (Wold 1982)	$\lvert x\rvert$	$\tau_j = 0, j = 1, \dots, J$	$c_{jk} = 1$ for two connected block and $c_{jk} = 0$ otherwise

For all the methods reported in Table 1, a single very simple monotonically and globally convergent algorithm is proposed for solving the optimization problem (100) - i.e. the bounded criterion to be maximized increases at each step of the iterative procedure -, which hits at convergence a stationary point of (100). Two numerically equivalent approaches for solving the RGCCA optimization problem are available. A primal formulation described in (Tenenhaus *et al.* 2017, ; Tenenhaus and Tenenhaus 2011) 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 prefered when $n > p_j$ and the dual form will be used when $n \leq p_j$. From these perspectives, RGCCA provide a general framework for exploratory data analysis of multiblock datasets that has immediate practical consequences for a unified statistical analysis and implementation strategy.

From the viewpoint of optimization problem (99), it appears that the term "generalized" embraces at least two notions. The first one relies on the generalization of Canonical Correlation Analysis (Hotelling 1936) to three or more sets of variables. The second one relies on the possibility of taking into account certain hypotheses on connections between blocks: the researcher decides which blocks are connected ($c_{jk} \neq 0$) and which are not ($c_{jk} = 0$). For instance, in the imaging-genetics context, the path between genetic and behavioural is mediated by the neuroimaging (no direct relationship between genetic and behavioural) and RGCCA allows taking into account this network of connections between blocks during the optimization process.

Note that $n^{-1}\mathbf{w}_j\mathbf{X}_j^\top\mathbf{X}_k\mathbf{w}_k$ is equal to the covariance between the block components $\mathbf{X}_j\mathbf{w}_j$ and $\mathbf{X}_k\mathbf{w}_k$ so that $g(n^{-1}\mathbf{w}_j\mathbf{X}_j^\top\mathbf{X}_k\mathbf{w}_k)$ can be interpreted as a convex function of the covariance between the block components, $\mathbf{X}_j\mathbf{w}_j$ and $\mathbf{X}_k\mathbf{w}_k$ of sets j and k. Consequently, maximizing (99) is the same as maximizing a sum of functions of the covariances between connected block components. If $\tau_j = \tau_k = 0$, then the covariance can be interpreted as a correlation. When using covariance based criteria, the data might be preprocessed in order to ensure comparability between variables and blocks ; for that purpose different strategies are possible. For instance, the usual practice is to standardize all the variables and then to divide each block by the square root of its number of variables $\sqrt{p_j}$ (Wold, Hellberg, Lundstedt, Sjöström, and Wold 1987) (yielding Trace$(\mathbf{X}_j^\top\mathbf{X}_j) = n$ for each block) or by the square root of the first eigenvalue of its covariance matrix (Escofier and Pages 1994) (yielding $\lambda_1(\mathbf{S}_{jj}) = 1$).

By setting $\mathbf{Q}_j = n^{-1/2}\mathbf{X}_j$ and $\mathbf{M}_j = \widehat{\mathbf{\Sigma}}_{jj} = \tau_j\mathbf{I}_{p_j} + (1-\tau_j)n^{-1}\mathbf{X}_j^\top\mathbf{X}_j$, $0 \leq \tau_j \leq 1$, optimization problem (99) is special case of optimization problem (13) with m equals to the number n of individuals. Therefore, the algorithm associated with optimization problem (99) is exactly the Algorithm **??** with $\mathbf{a}_j = \mathbf{M}_j^{1/2}\mathbf{w}_j$ and $\mathbf{P}_j = \mathbf{Q}_j\mathbf{M}_j^{-1/2}$. Therefore, the RGCCA algorithm for multiblock analysis reduces to:

In the multiblock and PLS path modeling literature (Tenenhaus, Esposito Vinzi, Chatelin, and Lauro 2005), typical choices of scheme functions are $g(x) = x, x^2$, or $|x|$.
Algorithm~**??** reconstructs exactly the RGCCA algorithm described in (Tenenhaus and Tenenhaus 2011, 2014) for these three scheme functions. An implementation of this algorithm is freely available on CRAN as part of the RGCCA package (Tenenhaus and Guillemot 2013). Empirically, we note that Algorithm **??** is found to be not very sensitive to the starting point and usually reaches convergence (`tol` $= 10^{-16}$) within a few iterations.

Many multiblock methods are special cases of problem (99). We refer the interested reader to (Tenenhaus and Tenenhaus 2011, 2014; **?**; Tenenhaus *et al.* 2017) for details; but an overview of methods that constitutes the RGCCA framework is given in the next section.

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:

(i) Main contributions for generalized canonical correlation analysis (GCCA) are found in (Horst 1961b; Carroll 1968a; Kettenring 1971; Wold 1982, 1985; Hanafi 2007).

(ii) 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, 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, 1985).

(iii) (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).

Regularized Generalized Canonical Correlation Analysis (RGCCA) is about maximizing (99) subject to specific constraints on the weight (choice of the shrinkage parameters $\tau_j s'$), the choice of the scheme function g and the design matrix $\mathbf{C}$. By appropriatly fixing these constraints, RGCCA encompasses many of these references as particular cases as outlined in Table 2-4.

In the two block case, optimization problem (99) becomes

$$\underset{\mathbf{w}_1,\mathbf{w}_2}{\text{maximize}}\ \text{cov}\left(\mathbf{X}_1\mathbf{w}_1,\mathbf{X}_2\mathbf{w}_2\right)\ \text{s.t.}\ \tau_j\|\mathbf{w}_j\|^2+(1-\tau_j)\text{var}(\mathbf{X}_j\mathbf{w}_j)=1, j=1,2 \tag{101}$$

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 (101) 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 2.

The special situation where $0 \leq \tau_1 \leq 1$ and $\tau_2 = 0$ which corresponds to a regularized version of redundancy analysis has been studied by (Takane and Hwang 2007) and by (Bougeard, Hanafi, and Qannari 2008) under the name "Continuum redundancy-PLS regression". When one block is reduced to only one variable, optimization problem (101) is equivalent to the simple continuum regression approach proposed by (Qannari and Hanafi 2005).

In the **multiblock data analysis** framework, 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 (99). These multiblock methods are listed in Table 3.

The SUMCOR criterion was first proposed by (Horst 1961b,a, 1965) under the name "maximum correlation method". Kettenring (1971) named this method SUMCOR and proposed a very simple algorithm for its resolution. The SSQCOR criterion has been first introduced by (Kettenring 1971).

Methods	Optimization problem
Interbattery Factor Analysis (Tucker 1958) (or **PLS Regression** (Wold, Martens, and Wold 1983))	$\underset{\substack{\mathbf{w}_1,\mathbf{w}_2 \\ \|\mathbf{w}_1\|=\|\mathbf{w}_2\|=1}}{\text{maximize}}\ \text{cov}\left(\mathbf{X}_1\mathbf{w}_1, \mathbf{X}_2\mathbf{w}_2\right)$
Canonical Correlation Analysis (Hotelling 1936)	$\underset{\mathbf{w}_1,\mathbf{w}_2}{\text{maximize}}\ \text{cor}\left(\mathbf{X}_1\mathbf{w}_1, \mathbf{X}_2\mathbf{w}_2\right)$
Redundancy analysis of $\mathbf{X}_1$ with respect to $\mathbf{X}_2$ (Van den Wollenberg 1977)	$\underset{\substack{\mathbf{w}_1,\mathbf{w}_2 \\ \|\mathbf{w}_1\|=1}}{\text{maximize}}\ \text{cor}\left(\mathbf{X}_1\mathbf{w}_1, \mathbf{X}_2\mathbf{w}_2\right) \text{var}(\mathbf{X}_1\mathbf{w}_1)^{1/2}$

Table 2: two-block component methods.

Methods	**Optimization problem**
SUMCOR (Horst 1961b)	$\underset{\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_J}{\text{maximize}} \sum_{j,k=1}^{J} \text{cor}(\mathbf{X}_j\mathbf{w}_j, \mathbf{X}_k\mathbf{w}_k)$
SSQCOR (Kettenring 1971)	$\underset{\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_J}{\text{maximize}} \sum_{j,k=1}^{J} \text{cor}^2(\mathbf{X}_j\mathbf{w}_j, \mathbf{X}_k\mathbf{w}_k)$
SABSCOR (Hanafi 2007)	$\underset{\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_J}{\text{maximize}} \sum_{j,k=1}^{J} \lvert\text{cor}(\mathbf{X}_j\mathbf{w}_j, \mathbf{X}_k\mathbf{w}_k)\rvert$
SUMCOV (= one-component MAXBET) (Van de Geer 1984)	$\underset{\substack{\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_J \\ \|\mathbf{w}_j\|=1, j=1,\ldots,J}}{\text{maximize}} \sum_{j,k=1}^{J} \text{cov}(\mathbf{X}_j\mathbf{w}_j, \mathbf{X}_k\mathbf{w}_k)$
MAXDIFF (Van de Geer 1984)	$\underset{\substack{\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_J \\ \|\mathbf{w}_j\|=1, j=1,\ldots,J}}{\text{maximize}} \sum_{j,k=1; j\neq k}^{J} \text{cov}(\mathbf{X}_j\mathbf{w}_j, \mathbf{X}_k\mathbf{w}_k)$
SSQCOV (one-component MAXBET B) (Hanafi and Kiers 2006)	$\underset{\substack{\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_J \\ \|\mathbf{w}_j\|=1, j=1,\ldots,J}}{\text{maximize}} \sum_{j,k=1}^{J} \text{cov}^2(\mathbf{X}_j\mathbf{w}_j, \mathbf{X}_k\mathbf{w}_k)$
MAXDIFF B (Hanafi and Kiers 2006)	$\underset{\substack{\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_J \\ \|\mathbf{w}_j\|=1, j=1,\ldots,J}}{\text{maximize}} \sum_{j,k=1; j\neq k}^{J} \text{cov}^2(\mathbf{X}_j\mathbf{w}_j, \mathbf{X}_k\mathbf{w}_k)$
SABSCOV (Tenenhaus and Tenenhaus 2011; Kramer 2007)	$\underset{\substack{\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_J \\ \|\mathbf{w}_j\|=1, j=1,\ldots,J}}{\text{maximize}} \sum_{j,k=1}^{J} \lvert\text{cov}(\mathbf{X}_j\mathbf{w}_j, \mathbf{X}_k\mathbf{w}_k)\rvert$

Table 3: Multiblock component methods.

The SABSCOR criterion has been studied by (Hanafi 2007). The SUMCOV criterion is actually a "one component per block" version of the MAXBET criterion proposed by (Van de Geer 1984). The SSQCOV criterion is a "one component per block" version of the MAXBET B criterion proposed by (Hanafi and Kiers 2006). The SABSCOV criterion has been introduced by (Kramer 2007).

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

Methods	Optimization problem
SUMCOR (Horst 1961b)	$\underset{\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_{J+1}}{\text{maximize}} \sum_{j=1}^{J} \text{cor}(\mathbf{X}_j\mathbf{w}_j, \mathbf{X}_{J+1}\mathbf{w}_{J+1})$
Carroll's GCCA (Carroll 1968a)	$\underset{\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_{J+1}}{\text{maximize}} \sum_{j=1}^{J} \text{cor}^2(\mathbf{X}_j\mathbf{w}_j, \mathbf{X}_{J+1}\mathbf{w}_{J+1})$
MCOA (Chessel and Hanafi 1996), **CPCA-W** (Smilde *et al.* 2003) and **Consensus PCA** (Westerhuis *et al.* 1998)	$\underset{\substack{\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_{J+1} \\ \Vert\mathbf{w}_j\Vert=1, j=1,\ldots,J+1}}{\text{maximize}} \sum_{j=1}^{J} \text{cov}^2(\mathbf{X}_j\mathbf{w}_j, \mathbf{X}_{J+1}\mathbf{w}_{J+1})$
Carroll's GCCA (Carroll 1968b)	$\underset{\substack{\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_{J+1} \\ \Vert\mathbf{w}_j\Vert=1, j=J_1,\ldots,J+1}}{\text{maximize}} \sum_{j=1}^{J_1} \text{cor}^2(\mathbf{X}_j\mathbf{w}_j, \mathbf{X}_{J+1}\mathbf{w}_{J+1}) + \sum_{j=J_1}^{J} \text{cov}^2(\mathbf{X}_j\mathbf{w}_j, \mathbf{X}_{J+1}\mathbf{w}_{J+1})$
Hierarchical PCA (Smilde *et al.* 2003)	$\underset{\substack{\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_{J+1} \\ \Vert\mathbf{w}_j\Vert=1, j=1,\ldots,J+1}}{\text{maximize}} \sum_{j=1}^{J} \text{cov}^4(\mathbf{X}_j\mathbf{w}_j, \mathbf{X}_{J+1}\mathbf{w}_{J+1})$
Hierarchical inter-battery factor analysis (Tenenhaus and Tenenhaus 2011)	$\underset{\substack{\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_{J+1} \\ \Vert\mathbf{w}_j\Vert=1, j=1,\ldots,J+1}}{\text{maximize}} \sum_{j=1}^{J} g\left(\text{cov}(\mathbf{X}_j\mathbf{w}_j, \mathbf{X}_{J+1}\mathbf{w}_{J+1})\right)$
Hierarchical canonical correlation analysis (Tenenhaus and Tenenhaus 2011)	$\underset{\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_{J+1}}{\text{maximize}} \sum_{j=1}^{J} g\left(\text{cor}(\mathbf{X}_j\mathbf{w}_j, \mathbf{X}_{J+1}\mathbf{w}_{J+1})\right)$
Hierarchical redundancy analysis of $\mathbf{X}_{J+1}$ with respect to the $\mathbf{X}_j$'s (Tenenhaus and Tenenhaus 2011)	$\underset{\substack{\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_{J+1} \\ \Vert\mathbf{w}_{J+1}\Vert=1}}{\text{maximize}} \sum_{j=1}^{J} g\left(\text{cor}(\mathbf{X}_j\mathbf{w}_j, \mathbf{X}_{J+1}\mathbf{w}_{J+1})\text{var}(\mathbf{X}_{J+1}\mathbf{w}_{J+1})^{1/2}\right)$

Table 4: Multiblock component methods in a situation of J blocks, $\mathbf{X}_1, \ldots, \mathbf{X}_J$ connected to a $(J+1)$th block defined as the concatenation of the blocks, $\mathbf{X}_{J+1} = \left[\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_J\right]$. g is any convex function.

It is quite remarkable that the single optimization problem (99) offers a framework for all the multi-block component methods outlined in Table 2-4. This has immediate practical consequences for unified statistical analysis and unified implementation strategies. The very simple gradient-based Algorithm **??** is monotonically convergent and gives at convergence a solution of the stationary equations related to the optimization problem.

A list of pre-specified methods than can be used with the rgcca() function 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 possible to obtain more than one block-component per block. Higher stage block components is obtained using a deflation strategy. This strategy forces all the block components within a block to be uncorrelated. This deflation procedure can be iterated in a very flexible way. It is not necessary to keep all the blocks in the procedure at all stages: the number of components summarizing a block can vary from one block to another (see (Tenenhaus *et al.* 2017) for details). The deflation procedure must be adapted when superblock is used. Various deflation strategies (what to deflate and how) have been proposed in the literature and we propose, as default option, the deflation strategy that yields to recover the most popular methods of the multiblock literature: Multiple factor analysis ((`ade4:::mfa()` / `FactoMineR::MFA`)) and Multiple Co-inertia Analysis (`ade4:::mcoa()`).

Two numerically equivalent approaches for solving the RGCCA optimization problem are available. A primal formulation described in (Tenenhaus *et al.* 2017, ; Tenenhaus and Tenenhaus 2011) 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 used when $n > p_j$ and the dual form will be preferred when $n \le p_j$. The `rgcca()` function of the RGCCA package implements these two formulations and selects automatically the best one.

3.2. Variable selection in RGCCA: 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. Specifically, RGCCA with all $\tau_j = 1$ equal to 1 is combined with an ℓ_1-penalty that gives rise to SGCCA (Tenenhaus, Philippe, Guillemot, Lê Cao, Grill, and Frouin 2014). The SGCCA optimization problem is defined as follows:

$$\underset{\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_J}{\text{maximize}} \sum_{j,k=1}^{J} c_{jk} g(\text{cov}(\mathbf{X}_j\mathbf{a}_j, \mathbf{X}_k\mathbf{a}_k)) \text{ s.t. } \|\mathbf{a}_j\|_2 = 1 \text{ and } \|\mathbf{a}_j\|_1 \le s_j, j = 1, \ldots, J \quad (102)$$

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.

The SGCCA algorithm is similar to the RGCCA algorithm and keeps the same convergence properties (Tenenhaus *et al.* 2014). The algorithm associated with the optimization problem (2) is available through the function `rgcca()` with the argument `method="sgcca`.

Guidelines describing how to use R/SGCCA in practice are provided in (Garali, Adanyeguh, Ichou, Perlbarg, Seyer, Colsch, Moszer, Guillemot, Durr, Mochel, and Tenenhaus 2018).

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 $\mathbf{X}_1, \dots, \mathbf{X}_J$ 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 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 $\mathbf{C}$; usually $c_{jk} = 1$ for two connected blocks and 0 otherwise. Therefore, we have decided to connect Agricultural Inequality to Political Instability ($c_{13} = 1$), Industrial Development to Political Instability ($c_{23} = 1$) and to not connect Agricultural Inequality to Industrial Development ($c_{12} = 0$). The resulting design matrix $\mathbf{C}$ 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
```

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

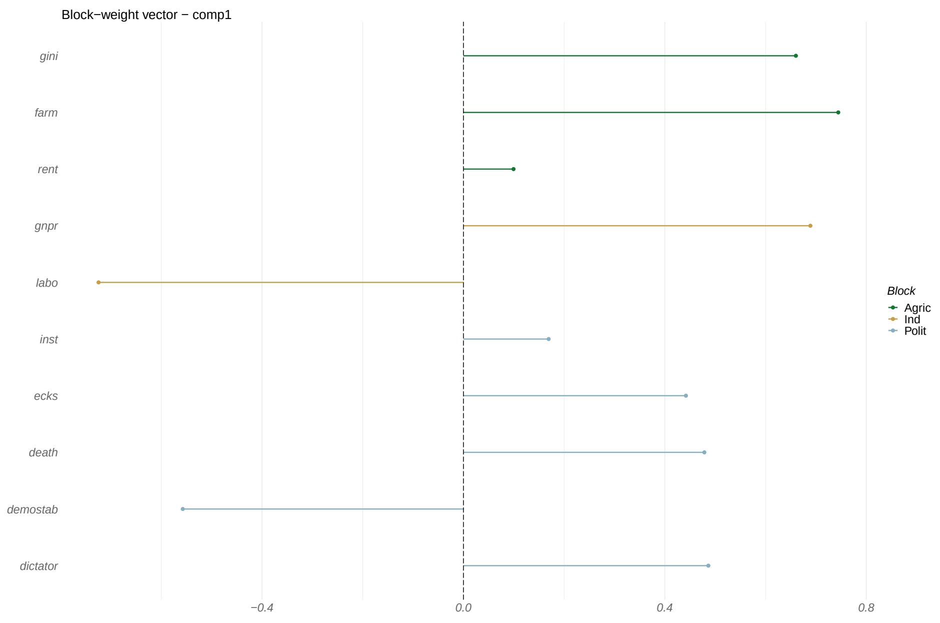
```
R> fit = rgcca(blocks = A, connection = C,
+             tau = 1, ncomp = 2,
+             scheme = "factorial",
+             scale = TRUE, scale_block = FALSE,
+             verbose = FALSE)
```

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 (100) 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)
```



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 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)
```

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

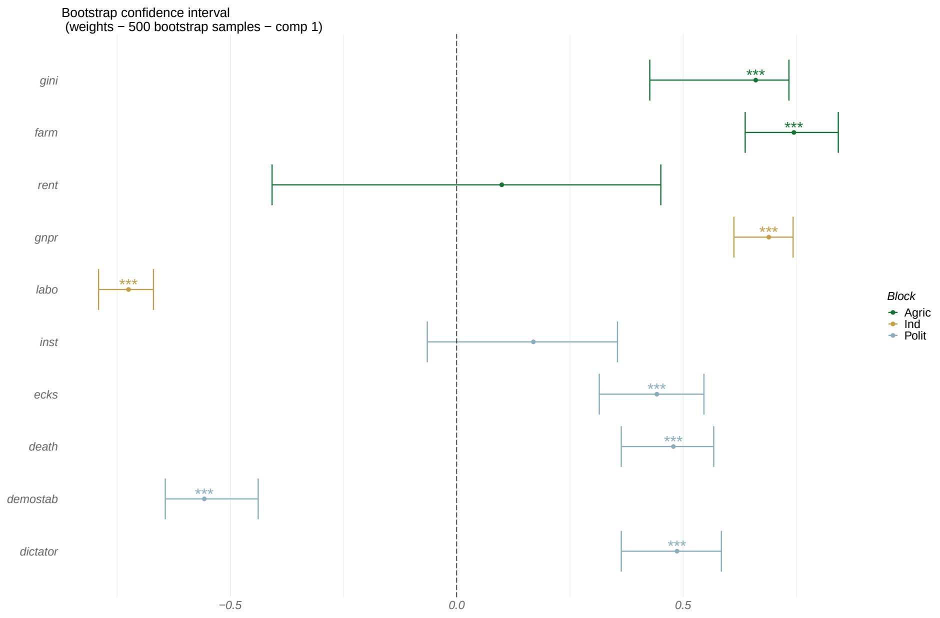
The bootstrap results are detailed using the `print()` function.

```
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   pval
gini       0.6602  0.6360 0.0773       0.426       0.734           8.540 0.0000
farm       0.7445  0.7318 0.0522       0.637       0.842          14.261 0.0000
rent       0.0994  0.0783 0.2128      -0.408       0.451           0.467 0.4006
gnpr       0.6891  0.6886 0.0325       0.612       0.743          21.222 0.0000
labo      -0.7247 -0.7237 0.0301      -0.791      -0.670         -24.108 0.0000
inst       0.1692  0.1681 0.1136      -0.065       0.355           1.489 0.0893
ecks       0.4418  0.4352 0.0621       0.315       0.546           7.120 0.0000
death      0.4784  0.4705 0.0515       0.363       0.567           9.296 0.0000
demostab  -0.5574 -0.5505 0.0503      -0.644      -0.438         -11.086 0.0000
dictator   0.4864  0.4828 0.0538       0.363       0.584           9.045 0.0000
         adjust.pval
gini           0.000
farm           0.000
rent           0.471
gnpr           0.000
labo           0.000
inst           0.128
ecks           0.000
death          0.000
demostab       0.000
dictator       0.000
```

and displayed using the `plot()` function.

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



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

```
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 ight quadrant concentrates on dictatorships. It is difficult for a country to escape dictatorship when its industrial development is below-average and its agricultural inequality is above average. It is worth pointing out that some unstable democracies located in this quadrant (or close to it) became dictatorships for a period of time after 1960: Greece (1967-1974), Brazil (1964-1985), Chili (1973-1990), and Argentina (1966-1973). The Average Variance Explained (AVE) defined below is also reported in the axis of the Figure. The AVE of block $\mathbf{X}_j$ for a specific block component $\mathbf{y}_j$ is defined as:

$$\mathrm{AVE}(\mathbf{X}_j) = \frac{1}{\|\mathbf{X}_j\|^2} \sum_{h=1}^{p_j} var(\mathbf{x}_{jh}) \times cor^2(\mathbf{x}_{jh}, \mathbf{y}_j) \tag{103}$$

$\mathrm{AVE}(\mathbf{X}_j)$ 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:

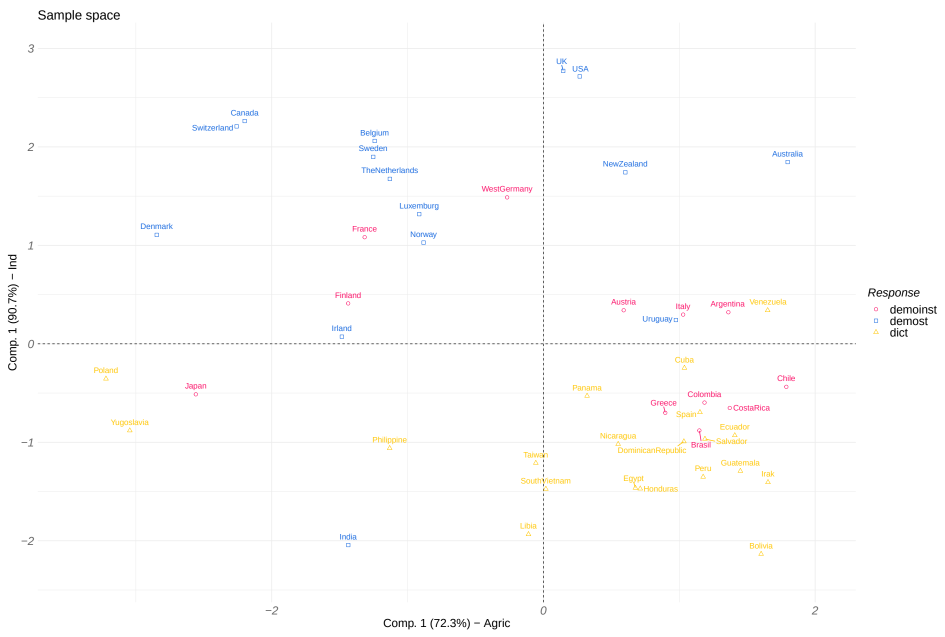


Figure 1: graphical display of the countries obtained by crossing y11 and y21 and labeled according to their political regime

$$\mathrm{AVE}(\mathrm{outermodel}) = \left(1/\sum_j p_j\right)\sum_j p_j \mathrm{AVE}(\mathbf{X}_j) \tag{104}$$

- For the inner model:

$$\mathrm{AVE}(\mathrm{innermodel}) = \left(1/\sum_{j<k} c_{jk}\right)\sum_{j<k} c_{jk}\mathrm{cor}^2(\mathbf{y}_j, \mathbf{y}_k) \tag{105}$$

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)
```

4.2. RGCCA with superblock

When the superblock option is considered 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. In this section, we consider Multiple Co-Inertia Analysis Hanafi and Kiers (2006) 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)
```

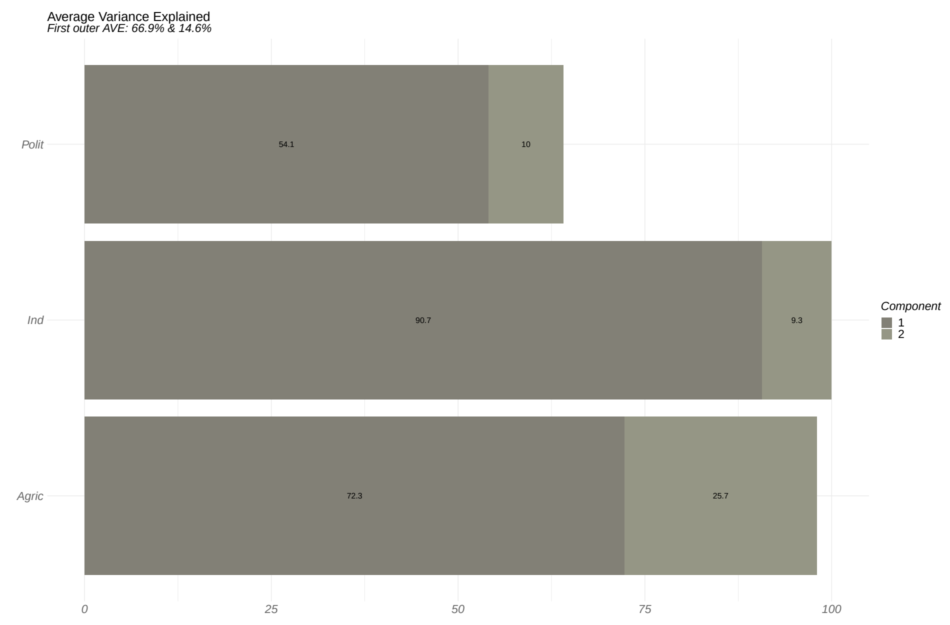


Figure 2: Average variance explained of the various blocks

MCOA enables the countries to be represented in the space spanned by the two first global components. Despite some overlap, the first global component exhibits a separation/continuum among regimes.

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

Moreover, the correlation circle highlights the contribution of each variable to the construction of the global components.

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

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

It is often useful to visualize jointly the sample space and the correlation circle. This is make possible by using the `type = "both"` argument of the plot function.

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

The block-weight vectors solution of MCOA are displayed below:

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

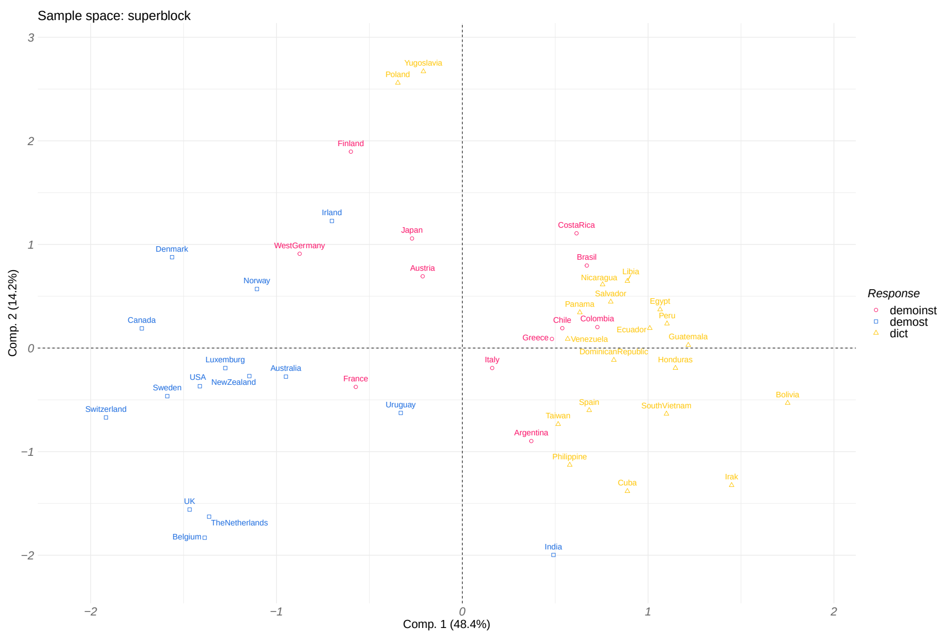


Figure 3: graphical display of the countries obtained by crossing the two first components of the superblock and labeled according to their political regime

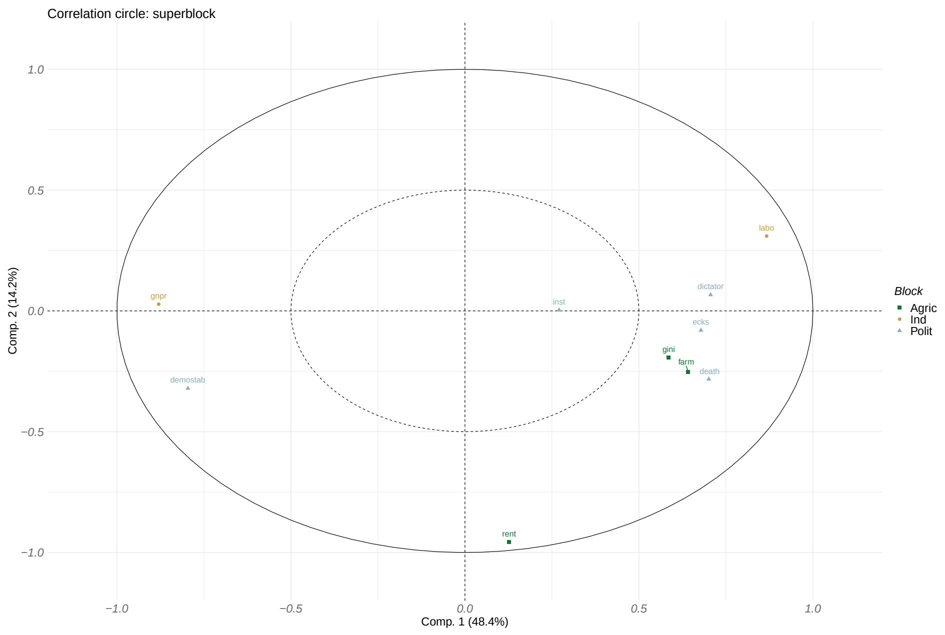


Figure 4: graphical display of the correlation cicle associated with the two first components of the superblock. Each variable are colored according to their block membership

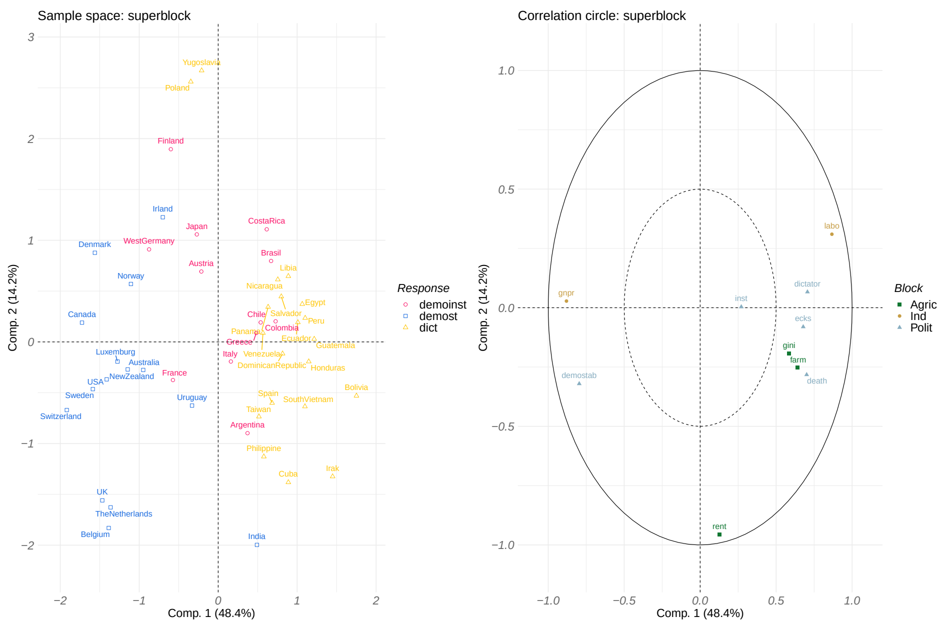
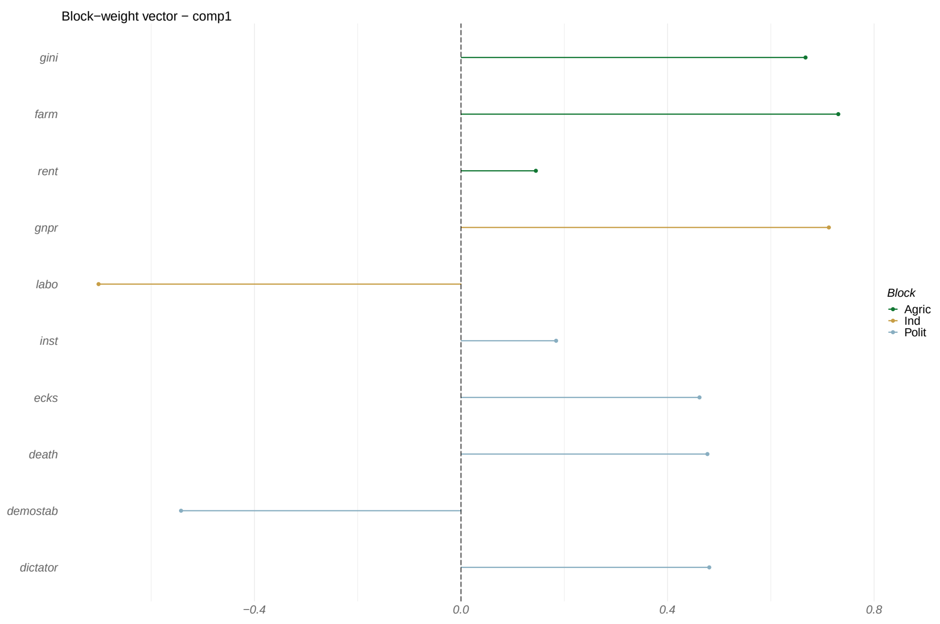


Figure 5: sample space and correlation cirle



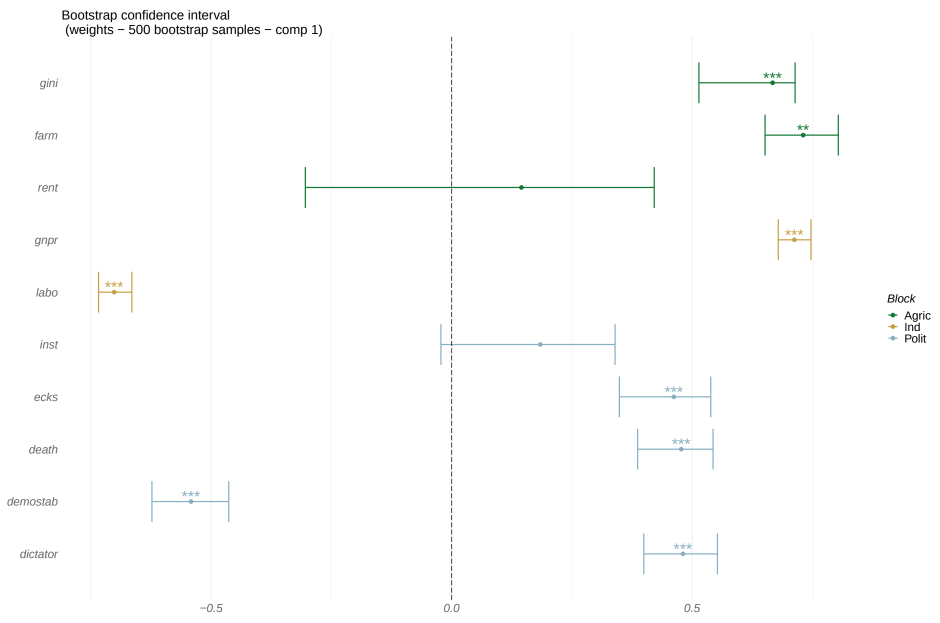
This model can be easily bootstrapped as follows:

```
R> boot_out = rgcca_bootstrap(fit.mcoa, n_boot = 500)
```

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

The bootstrap confidence intervals are available using the print()/plot() function

```
R> plot(boot_out, comp = 1,
+       display_order = FALSE, cex = 1.3)
```



4.3. Choice of the shrinkage parameter

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

Choice of the shrinkage parameter using the Schafer and Strimmer analytical formula (Schäfer and Strimmer 2005) For each block j, an "optimal" shrinkage parameter τ_j can be obtained from the Schafer and Strimmer analytical formula (Schäfer and Strimmer 2005) by setting the `tau` argument of the 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.

Choice of the shrinkage parameter by 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 samples in $\mathbf{X}_1, \ldots, \mathbf{X}_J$ are randomly permuted to obtained data sets $\mathbf{X}_1^*, \ldots, \mathbf{X}_J^*$.
>
> S/RGCCA is run on the permuted data set $\mathbf{X}_1^*, \ldots, \mathbf{X}_J^*$ to get block weight vectors $\mathbf{w}_1^*, \ldots, \mathbf{w}_J^*$.
>
> Record $t^* = \sum_{j,k} c_{jk} g(\text{cov}(\mathbf{X}_j^* \mathbf{w}_j^*, \mathbf{X}_k^* \mathbf{w}_k^*))$.
>
> S/RGCCA is run on the original data $\mathbf{X}_1, \ldots, \mathbf{X}_J$ to obtain the block weight vectors $\mathbf{w}_1, \ldots, \mathbf{w}_J$.
>
> Record $t = \sum_{j,k} c_{jk} g(\text{cov}(\mathbf{X}_j \mathbf{w}_j, \mathbf{X}_k \mathbf{w}_k))$.
>
> The resulting p-value is given by the fraction of permuted $t*$ that exceed the real tt obtained from the real data.

Then choose the set of tuning parameters that gives 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)
```

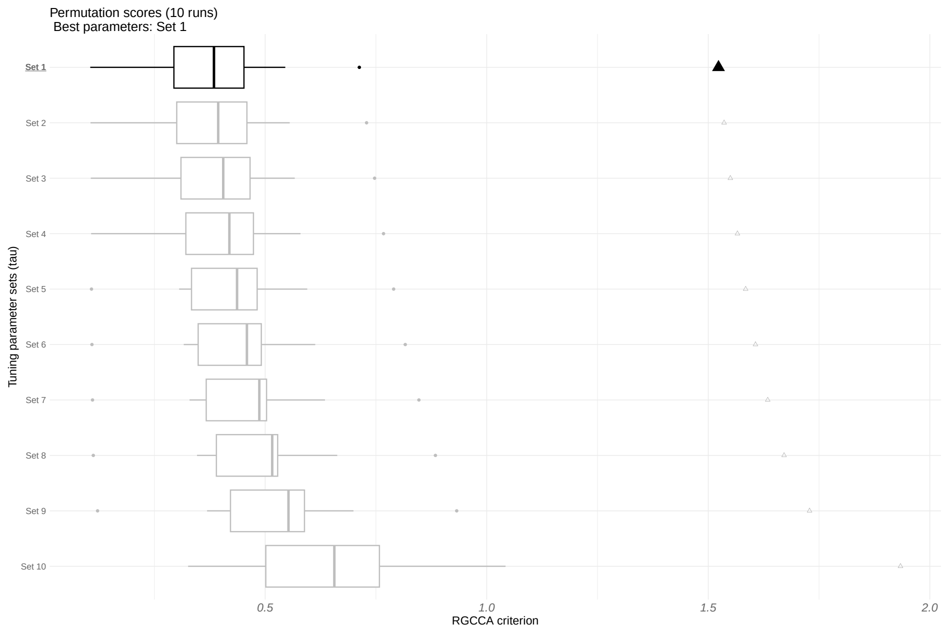
By default, the `rgcca_permutation()` function takes 10 sets of tuning parameters between min values (0 for RGCCA and $1/sqrt(ncol)$ for SGCCA) and 1. Results of the permutation procedure are summarized/displayed using the generic `print()/plot()` function

```
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)
```

Choice of the shrinkage parameter by 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 avalaible through the `rgcca_cv()` function. and is used for predicting the qualitative variable political regime from 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>
```

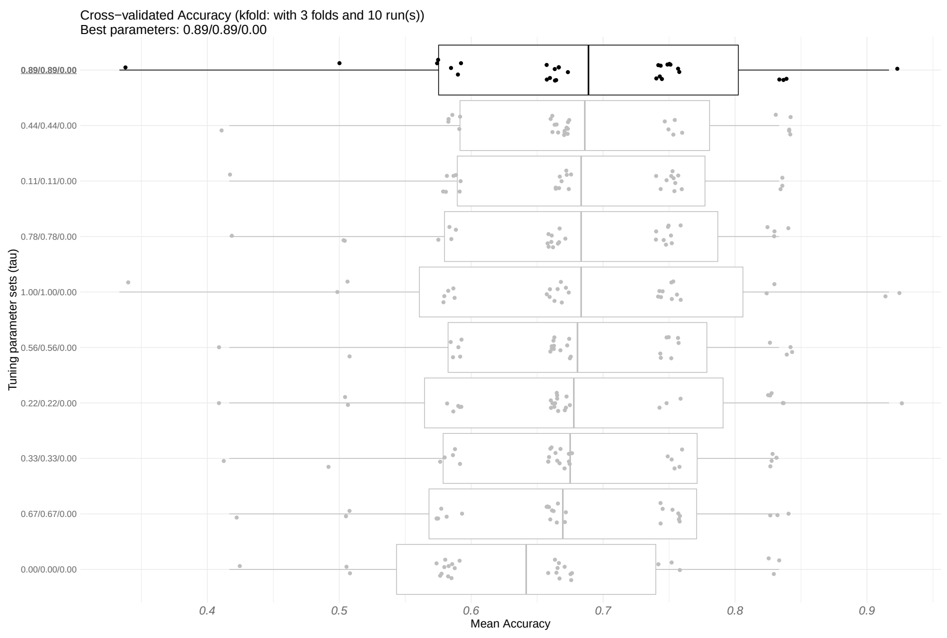
```
R> cv_out = rgcca_cv(blocks = training, response = 3,
+                    par_type = "tau",
+                    prediction_model = "rf",
+                    n_run = 10, k = 3,
+                    validation = "kfold",
+                    ncomp = 1, metric = "Accuracy")
```

`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: rf

   Tuning parameters Mean Accuracy     Sd
1     1.00/1.00/0.00         0.683 0.1226
2     0.89/0.89/0.00         0.689 0.1136
3     0.78/0.78/0.00         0.683 0.1036
4     0.67/0.67/0.00         0.669 0.1014
5     0.56/0.56/0.00         0.681 0.0981
6     0.44/0.44/0.00         0.686 0.0946
7     0.33/0.33/0.00         0.675 0.0963
8     0.22/0.22/0.00         0.678 0.1132
9     0.11/0.11/0.00         0.683 0.0939
10    0.00/0.00/0.00         0.642 0.0982

The best combination is: 0.89/0.89/0.00 for a mean Accuracy of 0.689.
```

and displayed using the `plot()` function.

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



Of course, the fitted cval object can be provided as output of `rgcca()` and 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$results$lab$confusion$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 has been adapted to 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 he 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 = 1)
```

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
```

At last, 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"
```

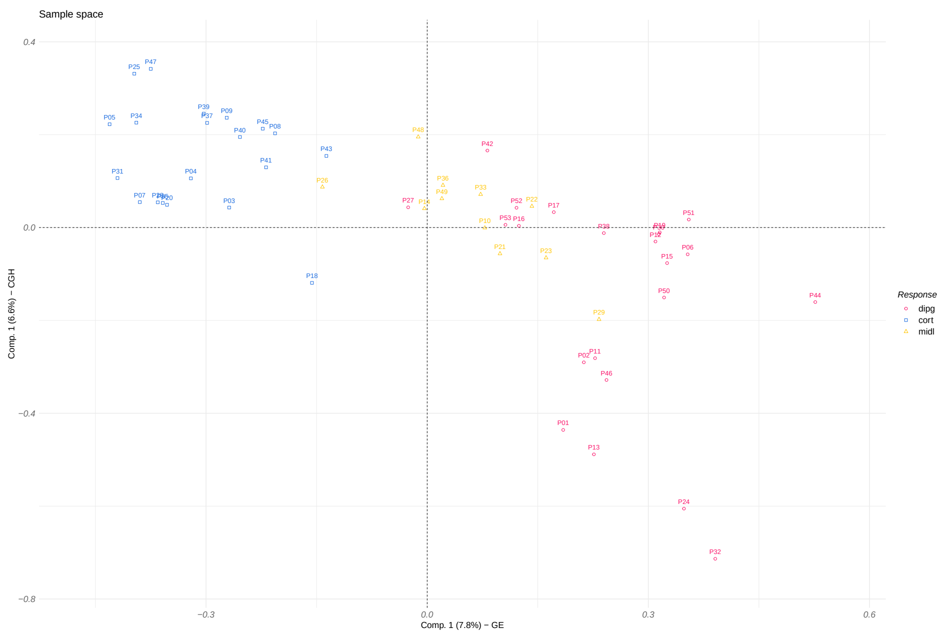
The dual formulation make the RGCCA algorithm highly efficient even in a high dimensional setting.

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

```
   user  system elapsed
   3.13    0.06    3.39
```

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 ℓ_1 penalization on the weight vectors $\mathbf{a}_1, \ldots, \mathbf{a}_J$ 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 $\mathbf{a}_1, \ldots, \mathbf{a}_J$. If `sparsity` is a vector, ℓ_1-penalties are the same for all the weights corresponding to the same block but different components:

$$\forall h, \|\mathbf{a}_j^{(h)}\|_{\ell_1} \leq c_{1j}\sqrt{p_j}, \tag{106}$$

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:

$$\forall h, \|\mathbf{a}_j^{(h)}\|_{\ell_1} \leq c_1[h, j]\sqrt{p_j}. \tag{107}$$

5.1. SGCCA for the Glioma dataset

In this situation, the optimal sparsity parameters is usually chosen to minimizing the cross-validated error rate. We decide to upper bound the sparsity paamerters for X1 and X3 to .2, to achieved an attactive amount of sparsity.

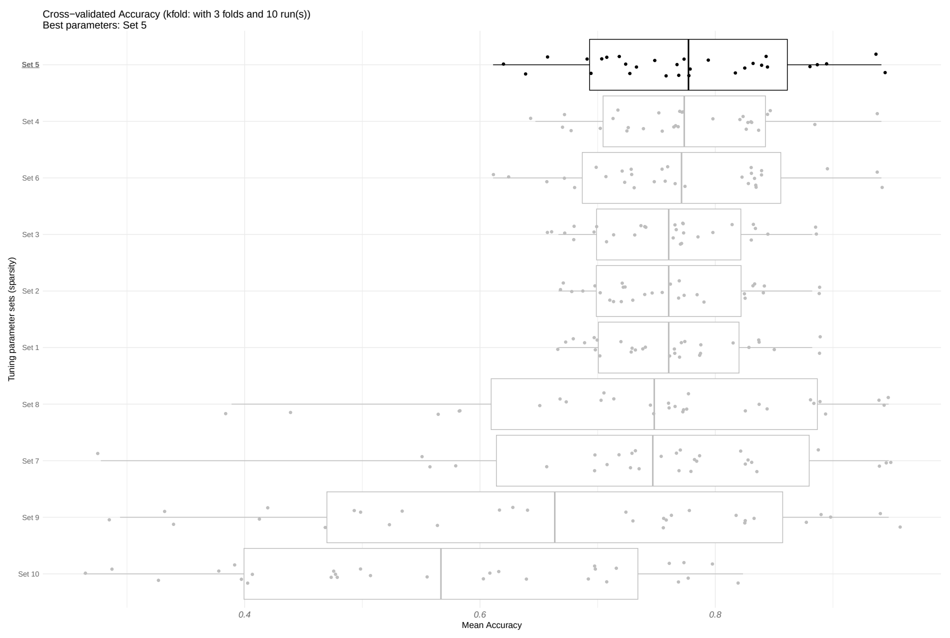
```
R> set.seed(27) #my favorite number
R> cv_out <- rgcca_cv(blocks, response = 3,
+                     par_type = "sparsity",
+                     par_value = c(.2, .2, 0),
+                     par_length = 10,
+                     prediction_model = "lda",
+                         validation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

As usual, 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)
```



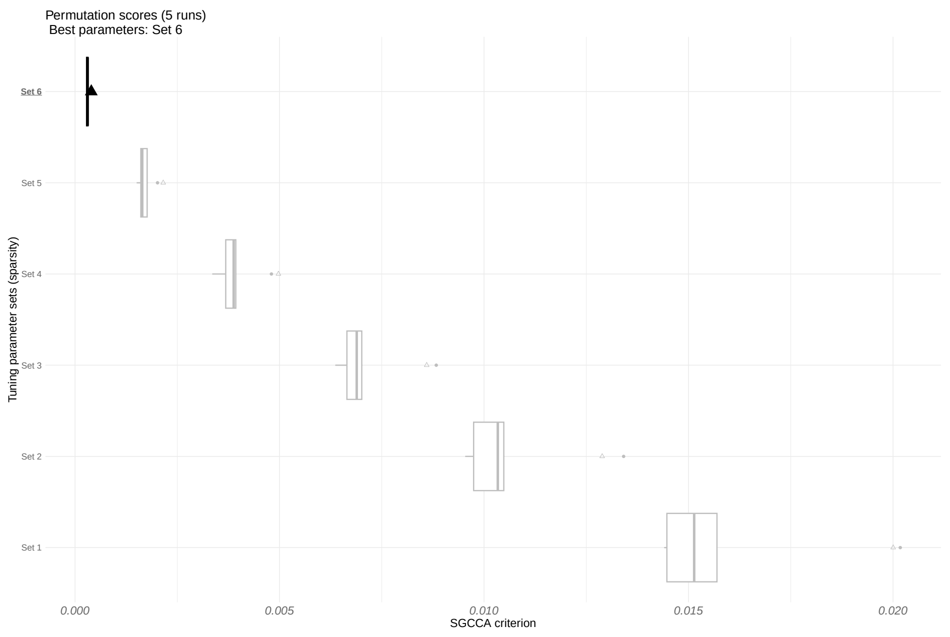
```
R> fit = rgcca(cv_out)
```

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.

First, it possible to determine the optimal sparsity parameters by permutation. This is made possible using 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)
```

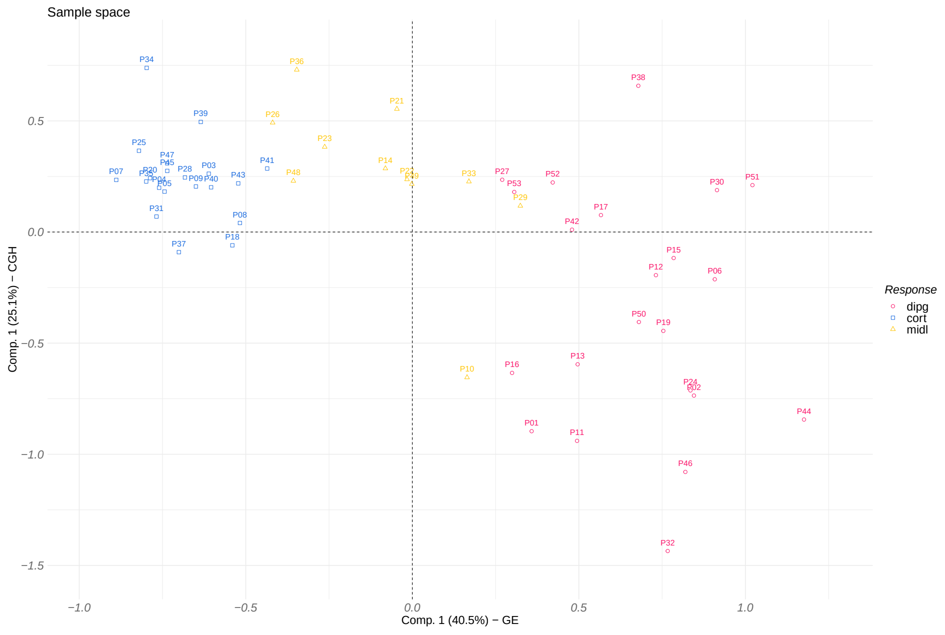
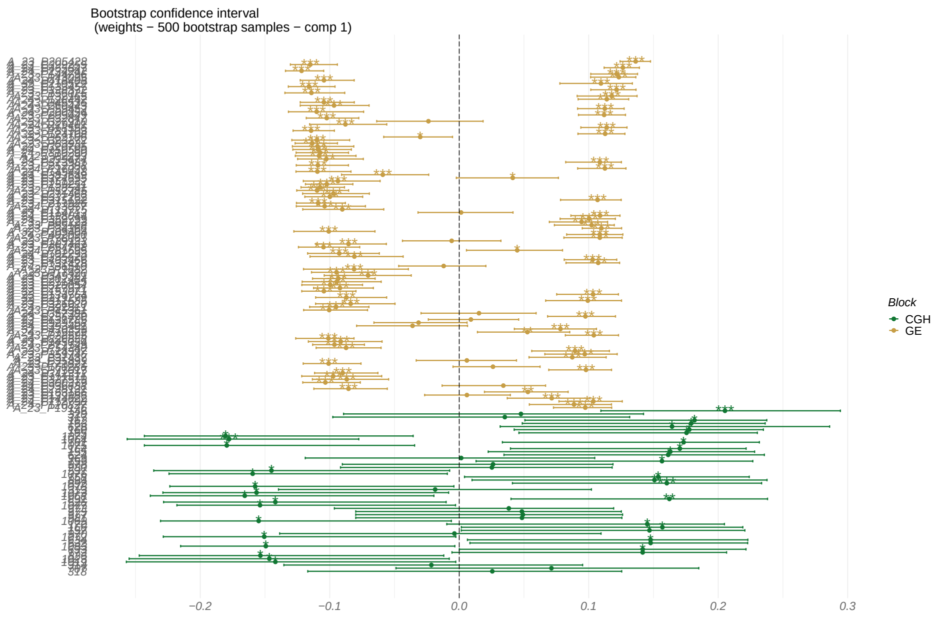


Figure 6: graphical display of the tumors obtained by crossing GE1 and CGH1 and labeled according to their location.



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$rgcca_res, 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

This package gathers 60 years of multiblock component methods and offers a unified implementation strategy for these methods. This release of the RGCCA package includes:

- Special attention has been paid to recover the results of other R packages of the literature including `ade4` and `factomineR` and `mixOmics`.
- K-fold cross-validation and permutation based strategies for optimal choice of the shrinkage parameters/level of sparsity.
- a bootstrap resampling procedure for assessing the reliability of the parameters estimates of S/RGCCA.
- Dedicated functions for graphical representations of the ouptut of RGCCA (sample plot, correlation circle, etc. . .).
- multiblock data faces two types of missing data structure: (i) if an observation i has missing values on a whole block j and (ii) if an observation i has some missing values on a block j (but not all). For these two situations, it is possible to exploit the algorithmic solution proposed for PLS path modeling to deal with missing data (see (Tenenhaus *et al.* 2005), page 171).

At last, RGCCA for multigroup data (Tenenhaus *et al.* 2014) and for RGCCA for multiway data (Gloaguen, Philippe, Frouin, Gennari, Dehaene-Lambertz, Le Brusquet, and Tenenhaus 2020) has been proposed but not yet integrated in the RGGCA package. In addition, global RGCCA has been recently developed and enables extracting simultaneously several components per block (no deflation procedure required). Work in progress includes the integration of this novel procedures in the next release of the package.

References

Bach F, Jordan M (2002). "Kernel independent component analysis." *Journal of Machine Learning Research*, **3**, 1–48.

Baker C (1973). "Joint Measures and cross-covariance operators." *Transactions of the American Mathematical Society*, **183**, 273–289.

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.

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

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 (1977). "Applications of convex analysis to multidimensional scaling." In JR Barra, F Brodeau, G Romier, B van Cutsem (eds.), *Recent developments in statistics*, pp. 133–145. North-Holland, Amsterdam, The Netherlands.

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.

Fukumizu K, Bach F, Gretton A (2007). "Statistical Consistency of Kernel Canonical Correlation Analysis." *Journal of Machine Learning Research*, **8**, 361–383.

Fukumizu K, Bach F, Jordan M (2004). "Dimensionality Reduction for Supervised Learning with Reproducing Kernel Hilbert Spaces." *Journal of Machine Learning Research*, **5**, 73–99.

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.

Gloaguen A, Philippe C, Frouin V, Gennari G, Dehaene-Lambertz G, Le Brusquet L, Tenenhaus A (2020). "Multiway generalized canonical correlation analysis." *Biostatistics*. ISSN 1465-4644. doi:10.1093/biostatistics/kxaa010. Kxaa010, https://academic.oup.com/biostatistics/advance-article-pdf/doi/10.1093/biostatistics/kxaa010/33294277/kxaa010.pdf, URL https://doi.org/10.1093/biostatistics/kxaa010.

Groenen PJ, Tenenhaus A (2015). "Regularized Generalized Canonical Correlation Analysis as an MM-algorithm." *arXiv preprint arXiv:XXXX*.

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.

Hardoon D, Shawe-Taylor J (2011). "Sparse Canonical Correlation Analysis." *Machine Learning*, **83**, 331–353.

Horst P (1961a). "Generalized canonical correlations and their applications to experimental data." *J. Clin. Psychol.*, **14**, 331–347.

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

Horst P (1965). *Factor Analysis of Data Matrices*. Holt, Rinehart and Winston, New York.

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.

Hunter DR, Lange K (2004). "A tutorial on MM algorithms." *The American Statistician*, **58**(1), 30–37.

Journée M, Nesterov Y, Richtárik P, Sepulchre R (2010). "Generalized power method for sparse principal component analysis." *The Journal of Machine Learning Research*, **11**, 517–553.

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.*

Lange K (2010). *Numerical analysis for statisticians*. Springer Science & Business Media.

Le Cao KA, Martin P, Robert-Granie C, Besse P (2009). "Sparse canonical methods for biological data integration: application to a cross-platform study." *BMC Bioinformatics*, **10 (34)**, 1–17.

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.

Lykou A, Whittaker J (2010). "Sparse CCA using a Lasso with positivity constraints." *Computational Statistics and Data Analysis*, **54 (12)**, 3144–3157.

Meyer RR (1976). "Sufficient conditions for the convergence of monotonic mathematical programming algorithms." *Journal of Computer and System Sciences*, **12**(1), 108–121.

Parkhomenko E, Tritchler D, Beyene J (2009). "Sparse canonical correlation analysis with application to genomic data integration." *Statistical Applications in Genetics and Molecular Biology*, **8**, 1–34.

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.

Schölkopf B, Smola A, Müller K (1998). "Nonlinear component analysis as a kernel eigenvalue problem." *Neural Computation*, **10**, 1299–1319.

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, Guillemot V (2013). *RGCCA: RGCCA and Sparse GCCA for multi-block data analysis*. R package version 2.0, URL `http://CRAN.R-project.org/package=RGCCA`.

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, Esposito Vinzi V, Chatelin Y, Lauro C (2005). "PLS path modeling." *Computational Statistics and Data Analysis*, **48**, 159–205.

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.

Vert J, Kanehisa M (2003). "Graph-driven features extraction from microarray data using diffusion kernels and kernel CCA." In *Neural Information Processing Systems (NIPS), MIT Press*.

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

Voss H, Eckhardt U (1980). “Linear Convergence of Generalized Weiszfeld’s Method.” *Computing*, **25**(3), 243–251.

Waaijenborg S, Verselewel de Witt Hamer P, Zwinderman A (2008). “Quantifying the association between gene expressions and DNA-markers by penalized canonical correlation analysis.” *Statistical Applications in Genetics and Molecular Biology*, **7**(1), Article 3.

Wahba G (1990). “Spline Models for Observational Data.” In *in CBMS-NSF Regional Conference Series in Applied Mathematics*, volume 59. SIAM.

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, Hellberg S, Lundstedt T, Sjöström M, Wold H (1987). “PLS modelling with latent variables in two or more dimensions.” In *In Proceedings of the symposium on PLS model building: Theory and application, Germany: Frankfurt am Main* , pp. 1–21.

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.

Yamanishi Y, Vert J, Kanehisa M (2004). “Heterogeneous data comparison and gene selection with kernel canonical correlation analysis.” In B Schölkopf, K Tsuda, J Vert (eds.), *Kernel Methods in Computational Biology*, pp. 209–234. MIT Press.

Zangwill WI (1969). *Nonlinear programming: a unified approach*. Prentice-Hall Englewood Cliffs, NJ.

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