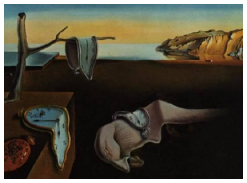

Journal of Statistical Software

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

Multiblock data analysis with the RGCCA package

Fabien Girka
Laboratoire des Signaux et Systèmes

Etienne Camenen
INSERM

Caroline Peltier
INRAE

Arnaud Gloaguen
CEA

Vincent Guillemot
Institut Pasteur

Laurent Le Brusquet
Laboratoire des Signaux et Systèmes

Arthur Tenenhaus
Laboratoire des Signaux et Systèmes

Abstract

Multiblock component methods play a central role in exploring complex relationships between multiple sets of variables. Regularized generalized canonical correlation analysis (RGCCA) is a unified and versatile framework that consolidates over six decades of multiblock component methods. The **RGCCA** package offers an implementation for this framework. Beyond implementation, the **RGCCA** package proposes graphical outputs and statistics to assess the robustness/significance of the analysis. The usefulness of the **RGCCA** package is illustrated in this paper on two real datasets. The **RGCCA** package is freely available on the Comprehensive R Archive Network (CRAN) `http://www.r-project.org/` and GitHub `https://github.com/rgcca-factory/RGCCA`.

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. This paradigm is referred to by several names, including "learning from multimodal data", "data integration", "multiview data", "multisource data", "data fusion", or "multiblock data analysis". Despite the availability of various statistical methods and dedicated software for multiblock data analysis, handling multiple highly multivariate datasets remains a critical issue for effective analysis and knowledge discovery.

Regularized generalized canonical correlation analysis (RGCCA, Tenenhaus and Tenenhaus 2011, 2014; Tenenhaus, Philippe, and Frouin 2015; Tenenhaus, Tenenhaus, and Groenen 2017) is a unified statistical framework that gathers six decades of multiblock component methods. While RGCCA encompasses a large number of methods, it is based on a single optimization problem that bears immediate practical implications, facilitating statistical analyses and implementation strategies. In this paper, we present the R package called **RGCCA**, which implements the RGCCA framework.

For the statistical computing environment R (R Core Team 2022), various R packages have come to the fore for carrying out multiblock data analysis. The bioinformatician community mostly uses **mixOmics** (Rohart, Gautier, Singh, and Lê Cao 2017). It wraps the main functions of the **RGCCA** package for performing multiblock analyses.

The **ade4** package (Dray and Dufour 2007) covers a wide range of multivariate methods. **ade4** mainly relies on three approaches for performing multiblock analysis: multiple co-inertia analysis (MCOA, Chessel and Hanafi 1996), multiple factor analysis (MFA, Escofier and Pages 1994) and Statis (Lavit, Escoufier, Sabatier, and Traissac 1994).

FactoMineR (Lê, Josse, and Husson 2008) is also one widely used package for multiblock methods mainly due to the implementation of MFA and, to a somewhat lesser extent, generalized Procrustes analysis (GPA, Gower 1975).

The **multiblock** package (Liland 2022) covers a wide range of multiblock methods for data fusion but intensively relies on a wrapper strategy for performing multiblock analyses. Consequently, this results in strong dependencies between the **multiblock** package and other packages including (i) **FactoMineR** (for MFA and GPA), **RGCCA** (for hierarchical PCA Wold, S. and Kettaneh, N. and Tjessem, K. 1996; Hanafi, Kohler, and Qannari 2010, MCOA, inter-battery factor analysis Tucker 1958 and Carroll's GCCA Carroll 1968a), **ade4** (multiblock redundancy analysis Bougeard, Qannari, and Rose 2011, and Statis), **r.jive** (for JIVE Lock, Hoadley, Marron, and Nobel 2013), and **RegularizedSCA** (for DISCO Schouteden, Van Deun, Wilderjans, and Van Mechelen 2014).

To the sake of completeness, we also mention the **multiview** R package that can be used for predicting a univariate response (member of the exponential family of distributions) from several blocks of variables (Ding, Li, Narasimhan, and Tibshirani 2022).

Each package uses its own way of specifying multiblock models and storing the results.

For Python users, **mvlearn** (Perry, Mischler, Guo, Lee, Chang, Koul, Franz, Richard, Carmichael, Ablin *et al.* 2021) seems to be the most advanced Python module for multiview data. **mvlearn** offers a suite of algorithms for learning latent space embeddings and joint representations of views, including a limited version of the RGCCA framework, a kernel version of GCCA (Hardoon, Szedmak, and Shawe-Taylor 2004; Bach and Jordan 2002), and deep CCA (Andrew, Arora, Bilmes, and Livescu 2013). Several other methods for dimensionality reduction and joint subspace learning are also included, such as multiview multidimensional scaling (Trendafilov 2010).

From this perspective, it appears that the **RGCCA** package already plays a central role within the R community and beyond for conducting multiblock analyses. Also, special attention has been paid to ensuring that **RGCCA**'s implementation of various multiblock component methods, including MCOA and MFA, aligns with results obtained from other packages such as **ade4** or **FactoMineR**.

Within the **RGCCA** package, all implemented methods from the literature are made available through the single `rgcca()` function and thus share the same function interface and a clear class structure. In addition to the main statistical functionalities, the **RGCCA** package provides several utility functions for data preprocessing and offers various advanced **ggplot2** (Wickham 2016) visualization tools to help users interpret and extract meaningful information from integrated data. It also includes metrics

for assessing the robustness and significance of the analysis. The package also includes several built-in datasets and examples to help users get started quickly. Package **RGCCA** is available from the Comprehensive R Archive Network (CRAN), at https://CRAN.R-project.org/package=RGCCA and can be installed from the R console with the following command:

```
R> install.packages("RGCCA")
```

This paper presents an overview of the RGCCA framework's theoretical foundations, summarizes the optimization problem under which all the algorithms were designed, and provides code examples to illustrate the package's usefulness and versatility. Our package offers a valuable contribution to the field of multiblock data analysis and will enable researchers to conduct more effective analyses and gain new insights into complex datasets.

The paper's remaining sections are organized as follows: Section 2 presents the RGCCA framework, how it generalizes existing methods from the literature, and the master algorithm underlying the RGCCA framework. Section 3 presents the structure of the package and provides code examples to illustrate the package's capabilities. Finally, we conclude the paper in Section 4.

2. The RGCCA framework

We first introduce the optimization problem defining the RGCCA framework, previously published in Tenenhaus and Tenenhaus (2011, 2014); Tenenhaus *et al.* (2015, 2017). We then show how it includes many existing methods.

2.1. Formulation

Let $\boldsymbol{x}$ be a random column vector of p variables such that $\boldsymbol{x}$ is the concatenation of J subvectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_J$, with $\boldsymbol{x}_j = (x_{j1}, \ldots, x_{jp_j})^\top$ for $j \in \{1, \ldots, J\}$. We assume that $\boldsymbol{x}$ has zero mean and a finite second-order moment. Its covariance matrix $\boldsymbol{\Sigma}$ is then composed of J^2 submatrices: $\boldsymbol{\Sigma}_{jk} = \mathbb{E}\left[\boldsymbol{x}_j \boldsymbol{x}_k^\top\right]$ for $(j,k) \in \{1, \ldots, J\}^2$. Let $\mathbf{a}_j = (a_{j1}, \ldots, a_{jp_j})^\top$ be a non-random p_j-dimensional column vector. A composite variable y_j is defined as the linear combination of the elements of $\boldsymbol{x}_j$: $y_j = \mathbf{a}_j^\top \boldsymbol{x}_j$. Therefore the covariance between two composite variables is $\mathbf{a}_j^\top \boldsymbol{\Sigma}_{jk} \mathbf{a}_k$.

The RGCCA framework aims to extract the information shared by the J random composite variables, taking into account an undirected graph of connections between them. It consists in maximizing the sum of the covariances between "connected" composites y_j and y_k subject to specific constraints on the weights $\mathbf{a}_j$ for $j \in \{1, \ldots, J\}$. The following optimization problem thus defines the RGCCA framework:

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

where

- the set Ω_j is compact.
- the function g is any continuously differentiable convex function. Typical choices of g are the identity (horst scheme, leading to maximizing the sum of covariances between block compo-

nents), the absolute value[1] (centroid scheme, yielding maximization of the sum of the absolute values of the covariances), the square function (factorial scheme, thereby maximizing the sum of squared covariances), or, more generally, for any even integer m, $g(x) = x^m$ (m-scheme, maximizing the power of m of the sum of covariances). The horst scheme penalizes negative structural correlation between block components, while the centroid scheme and the m-scheme enable two components to be negatively correlated.

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

2.2. Sample-based RGCCA

A sample-based optimization problem related to (1) can be defined by considering n observations of the vector $\boldsymbol{x}$. It yields a column-partition data matrix $\mathbf{X} = [\mathbf{X}_1, \ldots, \mathbf{X}_j, \ldots, \mathbf{X}_J] \in \mathbb{R}^{n \times p}$. Each $n \times p_j$ data matrix $\mathbf{X}_j$ is called a block and represents a set of p_j variables observed on the same set of n individuals. The variables' number and nature may differ from one block to another, but the individuals must be the same across blocks. We assume that all variables are centered.

The **RGCCA** package proposes a first implementation of the RGCCA framework, focusing on the following sample-based optimization problem:

$$\max_{\mathbf{a}_1,\ldots,\mathbf{a}_J} \sum_{j,k=1}^{J} c_{jk}\, \mathrm{g}\left(\mathbf{a}_j^\top \widehat{\boldsymbol{\Sigma}}_{jk} \mathbf{a}_k\right) \text{ s.t. } \mathbf{a}_j^\top \widehat{\boldsymbol{\Sigma}}_{jj} \mathbf{a}_j = 1, j = 1, \ldots, J. \tag{2}$$

where $\widehat{\boldsymbol{\Sigma}}_{jk} = n^{-1}\mathbf{X}_j^\top \mathbf{X}_k$ is an estimate of the interblock covariance matrix $\boldsymbol{\Sigma}_{jk} = \mathbb{E}[\boldsymbol{x}_j \boldsymbol{x}_k^\top]$ and $\widehat{\boldsymbol{\Sigma}}_{jj}$ is an estimate of the intra-block covariance matrix $\boldsymbol{\Sigma}_{jj} = \mathbb{E}[\boldsymbol{x}_j \boldsymbol{x}_j^\top]$. As we will see in the next section, an important family of methods is recovered by choosing $\Omega_j = \{\mathbf{a}_j \in \mathbb{R}^{p_j}; \mathbf{a}_j^\top \widehat{\boldsymbol{\Sigma}}_{jj} \mathbf{a}_j = 1\}$. In cases involving multi-collinearity within blocks or in high dimensional settings, one way of obtaining an estimate for the true covariance matrix $\boldsymbol{\Sigma}_{jj}$ is to consider the class of linear convex combinations of the sample covariance matrix $\mathbf{S}_{jj} = n^{-1}\mathbf{X}_j^\top \mathbf{X}_j$ and the identity matrix $\mathbf{I}_{p_j}$. Therefore, $\widehat{\boldsymbol{\Sigma}}_{jj} = (1 - \tau_j)\mathbf{S}_{jj} + \tau_j \mathbf{I}_{p_j}$ with $\tau_j \in [0, 1]$ (shrinkage estimator of $\boldsymbol{\Sigma}_{jj}$).

From this viewpoint, an equivalent formulation of optimization problem (2) is given hereafter and enables a better description of the objective of RGCCA.

$$\max_{\mathbf{a}_1,\ldots,\mathbf{a}_J} \sum_{j,k=1}^{J} c_{jk} \mathrm{g}\left(\widehat{\mathrm{cov}}\left(\mathbf{X}_j \mathbf{a}_j, \mathbf{X}_k \mathbf{a}_k\right)\right) \text{ s.t. } (1 - \tau_j)\widehat{\mathrm{var}}(\mathbf{X}_j \mathbf{a}_j) + \tau_j \|\mathbf{a}_j\|^2 = 1, j = 1, \ldots, J. \tag{3}$$

The objective of RGCCA is thus to find block components $\mathbf{y}_j = \mathbf{X}_j \mathbf{a}_j$ for $j \in \{1, \ldots, J\}$, summarizing the relevant information between and within the blocks. The τ_js are called shrinkage parameters ranging from 0 to 1 and interpolate smoothly between maximizing the covariance or the correlation. Setting τ_j to 0 will force the block components to unit variance, in which case the covariance criterion boils down to the correlation. Setting τ_j to 1 will normalize the block weight vectors, which

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

applies the covariance criterion. A value between 0 and 1 will lead to a compromise between the two first options. We can discuss the choice of shrinkage parameters by providing interpretations of the properties of the resulting block components:

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

It is worth pointing out that for each block j, an appropriate shrinkage parameter τ_j can be obtained using various analytical formulae (see Ledoit and Wolf 2004; Schäfer and Strimmer 2005; Chen, Wiesel, and Hero 2011, for instance). As $\widehat{\mathbf{\Sigma}}_{jj}$ must be positive definite, $\tau_j = 0$ can only be selected for a full rank data matrix $\mathbf{X}_j$. In the **RGCCA** package, for each block, the determination of the shrinkage parameter can be made fully automatic by using the analytical formula proposed by Schäfer and Strimmer (2005) or guided by the context of an application by cross-validation or permutation.

From the optimization problem (3), 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), inter-battery 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 to take 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 the capture of both correlation or covariance-based criteria. The fourth one relates to the function g that enables considering different functions of the covariance. A triplet of parameters embodies this generalization: (g, τ_j, $\mathbf{C}$) and by the fact that an arbitrary number of blocks can be handled. This triplet of parameters offers flexibility and allows RGCCA to encompass a large number of multiblock component methods that have been published for sixty years. Tables 1-3 give the correspondences between the triplet (g, τ_j, $\mathbf{C}$) and the multiblock component methods. For a complete overview, see Tenenhaus *et al.* (2017).

2.3. Special cases

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

- Main contributions for generalized canonical correlation analysis (GCCA) are found in Horst (1961); Carroll (1968a); Kettenring (1971); Wold (1982, 1985); Hanafi (2007).
- Main contributions for extending Tucker's method to more than two blocks come from Carroll (1968b); Chessel and Hanafi (1996); Hanafi and Kiers (2006); Hanafi *et al.* (2010); Hanafi,

Kohler, and Qannari (2011); Hanafi and Kiers (2006); Kramer (2007); Smilde, Westerhuis, and de Jong (2003); ten Berge (1988); Van de Geer (1984); Westerhuis, Kourti, and MacGregor (1998); Wold (1982, 1985).

- Carroll (1968b) proposed the "mixed" correlation and covariance criterion. Van den Wollenberg (1977) combined correlation and variance for the two-block situation (redundancy analysis). This method is extended to the multiblock situation in Tenenhaus and Tenenhaus (2011); Tenenhaus *et al.* (2017).

In the two block case, the optimization problem (3) reduces to:

$$\underset{\mathbf{a}_1,\mathbf{a}_2}{\text{maximize}}\ \widehat{\text{cov}}\left(\mathbf{X}_1\mathbf{a}_1, \mathbf{X}_2\mathbf{a}_2\right)\ \text{s.t.}\ (1-\tau_j)\widehat{\text{var}}(\mathbf{X}_j\mathbf{a}_j) + \tau_j\|\mathbf{a}_j\|^2 = 1, j = 1, 2. \tag{4}$$

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 (4) covers a situation which goes from canonical correlation analysis (Hotelling 1933) to Tucker's inter-battery factor analysis (Tucker 1958), while passing through redundancy analysis (Van den Wollenberg 1977). This framework corresponds exactly to the one proposed by Borga, Landelius, and Knutsson (1997) and Burnham, Viveros, and MacGregor (1996) and is reported in Table 1.

Table 1: Two-block component methods.

Methods	$g(x)$	τ_j	$\mathbf{C}$
Canonical correlation analysis (Hotelling 1936)	x	$\tau_1 = \tau_2 = 0$	$\mathbf{C}_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
Inter-battery factor analysis (Tucker 1958) or **PLS regression** (Wold, Martens, and Wold 1983)	x	$\tau_1 = \tau_2 = 1$	$\mathbf{C}_1$
Redundancy analysis (Van den Wollenberg 1977)	x	$\tau_1 = 1\ ;\ \tau_2 = 0$	$\mathbf{C}_1$
Regularized redundancy analysis (Takane and Hwang 2007; Bougeard, Hanafi, and Qannari 2008; Qannari and Hanafi 2005)	x	$0 \leq \tau_1 \leq 1\ ;\ \tau_2 = 0$	$\mathbf{C}_1$
Regularized canonical correlation analysis (Vinod 1976; Leurgans *et al.* 1993; Shawe-Taylor and Cristianini 2004)	x	$0 \leq \tau_1 \leq 1\ ;$ $0 \leq \tau_2 \leq 1$	$\mathbf{C}_1$

In the multiblock data analysis literature, all blocks $\mathbf{X}_j, j = 1, \ldots, J$ are assumed to be connected, and many criteria were proposed to find block components satisfying some covariance or correlation-based optimality. Most of them are special cases of the optimization problem (3). These multiblock component methods are listed in Table 2. PLS path modeling is also mentioned in this table. The

great flexibility of PLS path modeling lies in the possibility of taking into account certain hypotheses on connections between blocks: the researcher decides which blocks are connected and which are not.

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

Methods	$g(x)$	τ_j	$\mathbf{C}$
SUMCOR (Horst 1961)	x	$\tau_j = 0, j = 1, \ldots, J$	$\mathbf{C}_2 = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 \\ 1 & \cdots & 1 & 1 \end{pmatrix}$
SSQCOR (Kettenring 1971)	x^2	$\tau_j = 0, j = 1, \ldots, J$	$\mathbf{C}_2$
SABSCOR (Hanafi 2007)	$\lvert x \rvert$	$\tau_j = 0, j = 1, \ldots, J$	$\mathbf{C}_2$
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)	$\lvert x \rvert$	$\tau_j = 1, j = 1, \ldots, J$	$\mathbf{C}_2$
SUMCOV-2 (Van de Geer 1984)	x	$\tau_j = 1, j = 1, \ldots, J$	$\mathbf{C}_3 = \begin{pmatrix} 0 & 1 & \cdots & 1 \\ 1 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 \\ 1 & \cdots & 1 & 0 \end{pmatrix}$
SSQCOV-2 (Hanafi and Kiers 2006)	x^2	$\tau_j = 1, j = 1, \ldots, J$	$\mathbf{C}_3$
PLS path modeling - mode B (Wold 1982; Tenenhaus, Vinzi, Chatelin, and Lauro 2005)	$\lvert x \rvert$	$\tau_j = 0, j = 1, \ldots, J$	$c_{jk} = 1$ for two connected block and $c_{jk} = 0$ otherwise

Many multiblock component methods aim to simultaneously find 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, $\mathbf{X}_{J+1} = [\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_J]$. Several criteria were introduced in the literature, and many are listed below.

Table 3: 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: $\mathbf{X}_{J+1} = [\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_J]$.

Methods	$g(x)$	τ_j	$\mathbf{C}$
Generalized CCA (Carroll 1968a)	x^2	$\tau_j = 0, j = 1, \ldots, J+1$	$\mathbf{C}_4 = \begin{pmatrix} 0 & \cdots & 0 & 1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 1 \\ 1 & \cdots & 1 & 0 \end{pmatrix}$

Methods	$g(x)$	τ_j	$\mathbf{C}$
Generalized CCA (Carroll 1968b)	x^2	$\tau_j = 0, j = 1, \ldots, J_1$; $\tau_j = 1, j = J_1 + 1, \ldots, J$	$\mathbf{C}_4$
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; Westerhuis *et al.* 1998; Smilde *et al.* 2003)	x^2	$\tau_j = 1, j = 1, \ldots, J$; $\tau_{J+1} = 0$	$\mathbf{C}_4$
Multiple factor analysis (Escofier and Pages 1994)	x^2	$\tau_j = 1, j = 1, \ldots, J + 1$	$\mathbf{C}_4$

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

2.4. Sparse generalized canonical correlation analysis

RGCCA is a component-based approach that aims to study the relationships between several sets of variables. The quality and interpretability of the RGCCA block components $\mathbf{y}_j = \mathbf{X}_j\mathbf{a}_j, j = 1, \ldots, J$ are likely to be affected by the usefulness and relevance of the variables in each block. Therefore, it is important 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. Sparse generalized canonical correlation analysis (SGCCA) extends RGCCA to address this issue of variable selection (Tenenhaus, Philippe, Guillemot, Lê Cao, Grill, and Frouin 2014) and enhances the RGCCA framework. The SGCCA optimization problem is obtained by considering another set Ω_j as follows:

$$\underset{\mathbf{a}_1,\ldots,\mathbf{a}_J}{\text{maximize}} \sum_{j,k=1}^{J} c_{jk} \mathrm{g}\left(\widehat{\mathrm{cov}}\left(\mathbf{X}_j\mathbf{a}_j, \mathbf{X}_k\mathbf{a}_k\right)\right) \text{ s.t. } \|\mathbf{a}_j\|_2 \leq 1 \text{ and } \|\mathbf{a}_j\|_1 \leq s_j, j = 1, \ldots, J. \quad (5)$$

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 be chosen to result in desired amounts of sparsity.

SGCCA is also implemented in the **RGCCA** package and offers a sparse counterpart for all the covariance-based methods cited above.

2.5. A master algorithm for the RGCCA framework

The general optimization problem behind the RGCCA framework can be formulated as follows:

$$\max_{\mathbf{a}_1,\ldots,\mathbf{a}_J} f(\mathbf{a}_1,\ldots,\mathbf{a}_J) \text{ s.t. } \mathbf{a}_j \in \Omega_j,\ j=1,\ldots,J, \tag{6}$$

where $f(\mathbf{a}_1,\ldots,\mathbf{a}_J) : \mathbb{R}^{p_1} \times \ldots \times \mathbb{R}^{p_J} \to \mathbb{R}$ is a continuously differentiable multi-convex function and each $\mathbf{a}_j$ belongs to a compact set $\Omega_j \subset \mathbb{R}^{p_j}$. For such a function defined over a set of parameter vectors $(\mathbf{a}_1,\ldots,\mathbf{a}_J)$, we make no difference between the notations $f(\mathbf{a}_1,\ldots,\mathbf{a}_J)$ and $f(\mathbf{a})$, where $\mathbf{a}$ is the column vector $\mathbf{a} = \left(\mathbf{a}_1^\top,\ldots,\mathbf{a}_J^\top\right)^\top$ of size $p = \sum_{j=1}^J p_j$. Moreover, the vertical concatenation of a column vector is denoted $\mathbf{a} = (\mathbf{a}_1;\ldots;\mathbf{a}_J)$ for the sake of simplification of notation.

A simple, monotonically, and globally convergent algorithm is presented for solving the optimization problem (6). The maximization of the function f defined over different parameter vectors $(\mathbf{a}_1,\ldots,\mathbf{a}_J)$ is approached by updating each of parameter vector in turn, keeping the others fixed. This update rule was recommended in De Leeuw (1994) and is called Block Relaxation or cyclic Block Coordinate Ascent (BCA).

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

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

$$f(\mathbf{a}_1,...,\mathbf{a}_{j-1},\tilde{\mathbf{a}}_j,\mathbf{a}_{j+1},\ldots,\mathbf{a}_J) \geq f(\mathbf{a}) + \nabla_j f(\mathbf{a})^\top(\tilde{\mathbf{a}}_j - \mathbf{a}_j). \tag{7}$$

On the right-hand side of the inequality (7), only the term $\nabla_j f(\mathbf{a})^\top\tilde{\mathbf{a}}_j$ is relevant to $\tilde{\mathbf{a}}_j$ and the solution that maximizes the minorizing function over $\tilde{\mathbf{a}}_j \in \Omega_j$ is obtained by considering the following optimization problem:

$$\hat{\mathbf{a}}_j = \underset{\tilde{\mathbf{a}}_j\in\Omega_j}{\operatorname{argmax}}\ \nabla_j f(\mathbf{a})^\top\tilde{\mathbf{a}}_j := r_j(\mathbf{a}). \tag{8}$$

The entire algorithm is subsumed in Algorithm 1.

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

Result: $\mathbf{a}_1^s,\ldots,\mathbf{a}_J^s$ (approximate solution of (6))
Initialization: choose random vector $\mathbf{a}_j^0 \in \Omega_j, j=1,\ldots,J$, ε;
$s = 0$;
repeat
  for $j = 1$ **to** J **do**

$$\mathbf{a}_j^{s+1} = r_j\left(\mathbf{a}_1^{s+1},\ldots,\mathbf{a}_{j-1}^{s+1},\mathbf{a}_j^s,\ldots,\mathbf{a}_J^s\right). \tag{9}$$

  end for
  $s = s + 1$;
until $f(\mathbf{a}_1^{s+1},\ldots,\mathbf{a}_J^{s+1}) - f(\mathbf{a}_1^s,\ldots,\mathbf{a}_J^s) < \varepsilon$

We need to introduce some extra notations to present the convergence properties of Algorithm 1: $\Omega = \Omega_1 \times \ldots \times \Omega_J$, $\mathbf{a} = (\mathbf{a}_1; \ldots; \mathbf{a}_J) \in \Omega$, $c_j : \Omega \mapsto \Omega$ is an operator defined as $c_j(\mathbf{a}) = (\mathbf{a}_1; \ldots; \mathbf{a}_{j-1}; r_j(\mathbf{a}); \mathbf{a}_{j+1}; \ldots; \mathbf{a}_J)$ with $r_j(\mathbf{a})$ introduced in Equation~8 and $c : \Omega \mapsto \Omega$ is defined as $c = c_J \circ c_{J-1} \circ ... \circ c_1$, where $\circ$ stands for the composition operator. Using the operator c, the «for loop» inside Algorithm 1 can be replaced by the following recurrence relation: $\mathbf{a}^{s+1} = c(\mathbf{a}^s)$. The convergence properties of Algorithm 1 are summarized in the following proposition:

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

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

(b) If the infinite sequence $\{f(\mathbf{a}^s)\}$ involves a finite number of distinct terms, then the last distinct point satisfies $c(\mathbf{a}^s) = \mathbf{a}^s$ and therefore is a stationary point of the problem (6).

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

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

(e) The sequence $\{\mathbf{a}^s\}$ is asymptotically regular: $\lim_{s \to \infty} \sum_{j=1}^{J} \|\mathbf{a}_j^{s+1} - \mathbf{a}_j^s\| = 0$. This result implies that if the threshold ε for the stopping criterion in Algorithm 1 is made sufficiently small, the output of Algorithm 1 will be as close as wanted to a stationary point of (6).

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

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

The optimization problem (1) defining the RGCCA framework is a particular case of (6). Indeed, we assume the Ω_js' to be compact. In addition, when the diagonal of $\mathbf{C}$ is null, the convexity and the continuous differentiability of the function g imply that the objective function of (1) itself is multi-convex continuously differentiable. When at least one element of the diagonal of $\mathbf{C}$ is different from 0, additional conditions have to be imposed on g to keep the objective function multi-convex. For example, when g is twice differentiable, a sufficient condition is that $\forall x \in \mathbb{R}_+$, $g'(x) \geq 0$. This condition guarantees that the second derivative of $g\left(\mathbf{a}_j^\top \boldsymbol{\Sigma}_{jj} \mathbf{a}_j\right)$ is positive-definite:

$$\frac{\partial^2 g\left(\mathbf{a}_j^\top \boldsymbol{\Sigma}_{jj} \mathbf{a}_j\right)}{\partial \mathbf{a}_j \partial \mathbf{a}_j^\top} = 2\left[g'\left(\mathbf{a}_j^\top \boldsymbol{\Sigma}_{jj} \mathbf{a}_j\right) \boldsymbol{\Sigma}_{jj} + 2g''\left(\mathbf{a}_j^\top \boldsymbol{\Sigma}_{jj} \mathbf{a}_j\right) \boldsymbol{\Sigma}_{jj} \mathbf{a}_j \mathbf{a}_j^\top \boldsymbol{\Sigma}_{jj}\right]. \tag{10}$$

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

2.6. The specific case of RGCCA

The optimization problem (2) defining sample-based RGCCA is a particular case of (6). Indeed, $\Omega_j = \{\mathbf{a}_j \in \mathbb{R}^{p_j}; \mathbf{a}_j^\top \widehat{\boldsymbol{\Sigma}}_{jj} \mathbf{a}_j = 1\}$ defines a compact set. Therefore, Algorithm 1 can be used to solve the optimization problem (2). This is done by updating each parameter vector, in turn, keeping the others fixed. Hence, we want to find an update $\hat{\mathbf{a}}_j \in \Omega_j = \left\{\mathbf{a}_j \in \mathbb{R}^{p_j}; \mathbf{a}_j^\top \widehat{\boldsymbol{\Sigma}}_{jj} \mathbf{a}_j = 1\right\}$ such

that $f(\mathbf{a}) \leq f(\mathbf{a}_1, \ldots, \mathbf{a}_{j-1}, \hat{\mathbf{a}}_j, \mathbf{a}_{j+1}, \ldots, \mathbf{a}_J)$. The RGCCA update is obtained by considering the following optimization problem:

$$\hat{\mathbf{a}}_j = \underset{\tilde{\mathbf{a}}_j \in \Omega_j}{\operatorname{argmax}} \nabla_j f(\mathbf{a})^\top \tilde{\mathbf{a}}_j = \frac{\widehat{\boldsymbol{\Sigma}}_{jj}^{-1} \nabla_j f(\mathbf{a})}{\|\widehat{\boldsymbol{\Sigma}}_{jj}^{-1/2} \nabla_j f(\mathbf{a})\|} := r_j(\mathbf{a}), j = 1, \ldots, J, \tag{11}$$

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

$$\nabla_j f(\mathbf{a}) = 2 \sum_{k=1}^{J} c_{jk} g' \left(\mathbf{a}_j^\top \widehat{\boldsymbol{\Sigma}}_{jk} \mathbf{a}_k \right) \widehat{\boldsymbol{\Sigma}}_{jk} \mathbf{a}_k. \tag{12}$$

2.7. The specific case of SGCCA

The optimization problem (5) falls into the RGCCA framework with $\Omega_j = \{\mathbf{a}_j \in \mathbb{R}^{p_j}; \|\mathbf{a}_j\|_2 \leq 1; \|\mathbf{a}_j\|_1 \leq s_l\}$. Ω_j 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 compact sets. Hence, Ω_j is a compact set. Therefore, we can consider the following update for SGCCA:

$$\hat{\mathbf{a}}_j = \underset{\|\tilde{\mathbf{a}}_j\|_2 \leq 1; \|\tilde{\mathbf{a}}_j\|_1 \leq s_j}{\operatorname{argmax}} \nabla_j f(\mathbf{a})^\top \tilde{\mathbf{a}}_j := r_j(\mathbf{a}). \tag{13}$$

According to Witten, Tibshirani, and Hastie (2009), the solution of (13) satisfies:

$$r_j(\mathbf{a}) = \hat{\mathbf{a}}_j = \frac{\mathcal{S}(\nabla_j f(\mathbf{a}), \lambda_j)}{\|\mathcal{S}(\nabla_j f(\mathbf{a}), \lambda_j)\|_2}, \text{ where } \lambda_j = \begin{cases} 0 \text{ if} & \frac{\|\nabla_j f(\mathbf{a})\|_1}{\|\nabla_j f(\mathbf{a})\|_2} & \leq s_j \\ \text{find } \lambda_j \text{ such that} & \|\hat{\mathbf{a}}_j\|_1 & = s_j \end{cases}, \tag{14}$$

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

$$\mathbf{u} = \mathcal{S}(\mathbf{x}, \lambda) \Leftrightarrow u_j = \begin{cases} \operatorname{sign}(x_j)(|x_j| - \lambda), & \text{if } |x_j| > \lambda \\ 0, & \text{if } |x_j| \leq \lambda \end{cases}, j = 1, \ldots, p. \tag{15}$$

We made the assumption that the ℓ_2-ball of radius 1 is not included in the ℓ_1-ball of radius s_j and the other way round. Otherwise, systematically, only one of the two constraints is active. This assumption is true when the corresponding spheres intersect. This assumption can be translated into conditions on s_j.

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

$$\forall \mathbf{x} \in \mathbb{R}^{p_j}, \ \|\mathbf{x}\|_2 \leq \|\mathbf{x}\|_1 \leq \sqrt{p_j} \|\mathbf{x}\|_2. \tag{16}$$

This can be converted into a condition on s_j: $1 \leq s_j \leq \sqrt{p_j}$. When such a condition is fulfilled, the ℓ_2-sphere of radius 1 and the ℓ_1-sphere of radius s_j necessarily intersect. Within the **RGCCA** package, for consistency with the value of $\tau_j \in [0, 1]$, the level of sparsity for $\mathbf{a}_j$ is controlled with $s_j/p_j \in [1/\sqrt{p_j}, 1]$.

Several strategies, such as binary search or the projection on convex sets algorithm (POCS), also known as the alternating projection method (Boyd, Dattorro *et al.* 2003), can be used to determine the optimal λ_j verifying the ℓ_1-norm constraint. Here, a much faster approach described in Gloaguen, Guillemot, and Tenenhaus (2017) is implemented within the **RGCCA** package.

The SGCCA algorithm is similar to the RGCCA algorithm and keeps the same convergence properties. Empirically, we note that the S/RGCCA algorithm is found to be not sensitive to the starting point and usually reaches convergence (`tol = 1e-16`) within a few iterations.

2.8. Higher level RGCCA algorithm

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

The second level RGCCA optimization problem boils down to:

$$\max_{\mathbf{a}_1, \ldots, \mathbf{a}_J} \sum_{j,k=1}^{J} c_{jk}\, \mathrm{g} \left(n^{-1} \mathbf{a}_j^\top {\mathbf{X}_j^{(1)}}^\top \mathbf{X}_k^{(1)} \mathbf{a}_k \right) \text{ s.t. } \mathbf{a}_j \in \Omega_j. \tag{17}$$

The optimization problem (17) is solved using Algorithm 1 and returns a set of optimal block-weight vectors $\mathbf{a}_j^{(2)}$ and block components $\mathbf{y}_j^{(2)} = \mathbf{X}_j^{(1)} \mathbf{a}_j^{(2)}$, for $j = 1 \ldots, J$.

For orthogonal block weight vectors, $\mathbf{y}_j^{(2)} = \mathbf{X}_j^{(1)} \mathbf{a}_j^{(2)} = \mathbf{X}_j \mathbf{a}_j^{(2)}$ naturally expresses as a linear combination of the original variables. For orthogonal block component, as $\mathbf{y}_j^{(1)} = \mathbf{X}_j \mathbf{a}_j^{(1)}$, the range space of $\mathbf{X}_j^{(1)}$ is included in the range space of $\mathbf{X}_j$. Therefore any block component $\mathbf{y}_j^{(2)}$ belonging to the range space of $\mathbf{X}_j^{(1)}$ can also be expressed in terms of the original block $\mathbf{X}_j$: that is, it exists $\mathbf{a}_j^{(2)^\star}$ such that $\mathbf{y}_j^{(2)} = \mathbf{X}_j^{(1)} \mathbf{a}_j^{(2)} = \mathbf{X}_j \mathbf{a}_j^{(2)^\star}$. It implies that the block components can always be expressed in terms of the original data matrix, whatever the deflation mode.

This deflation procedure can be iterated in a very flexible way. For instance, it is not necessary to keep all the blocks in the procedure at all stages: the number of components per block can vary from one block to another. This might be interesting in a supervised setting where we want to predict a univariate block from other blocks. In that case, the deflation procedure applies to all blocks except the one to predict.

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

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

The individual blocks $\mathbf{X}_j^{(1)}$s are then retrieved from the deflated superblock. This strategy enables recovering multiple factor analysis (`ade4::mfa()` and `FactoMineR::MFA()`). Note that, in this case, block components do not belong to their block space and are correlated. On the contrary, we follow the deflation strategy described in Chessel and Hanafi (1996) (`ade4::mcoa()`) for multiple co-inertia analysis, which is one of the most popular and established methods of the multiblock literature.

2.9. Average variance explained

In this section, using the idea of average variance explained (AVE), the following indicators of model quality are defined:

- The AVE for a given block component $\mathbf{y}_j$ can be computed using the following formula:

$$\text{AVE}(\mathbf{X}_j) = \frac{1}{\|\mathbf{X}_j\|^2} \sum_{h=1}^{p_j} \text{var}(\mathbf{x}_{jh}) \times \text{cor}^2(\mathbf{x}_{jh}, \mathbf{y}_j). \tag{18}$$

- A global indicator of model quality can be obtained by considering a weighted sum of these individual AVEs. This outer AVE is defined as:

$$\text{AVE(outermodel)} = \left(1/\sum_j \|\mathbf{X}_j\|^2\right) \sum_j \|\mathbf{X}_j\|^2 \text{AVE}(\mathbf{X}_j). \tag{19}$$

However, the previous quantities do not take into account the correlations between blocks. Therefore, another indicator of model quality is the inner AVE, defined as follows:

$$\text{AVE(innermodel)} = \left(1/\sum_{j<k} c_{jk}\right) \sum_{j<k} c_{jk} \text{cor}^2(\mathbf{y}_j, \mathbf{y}_k). \tag{20}$$

All these quantities vary between 0 and 1 and reflect important properties of the model.

Equation~18 is defined for a specific block component. The cumulative AVE is obtained by summing these individual AVEs over the different components. However, this sum applies only to orthogonal block components. For correlated components, we follow the QR-orthogonalization procedure described in Zou, Hastie, and Tibshirani (2006) to consider only the increase of AVE due to adding the new components.

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

3. The RGCCA package

This section describes how the **RGCCA** package is designed and how it can be used on two real datasets.

3.1. Software design

The main user-accessible functions are listed in Table 4.

Table 4: Functions implemented in the **RGCCA** package.

Function	Description
`rgcca`	Main entry point of the package, this function allows fitting a R/SGCCA model on a multiblock dataset.
`rgcca_cv`	Find the best set of parameters for a R/SGCCA model using cross-validation.
`rgcca_predict`	Train a caret model on the block components of a fitted R/SGCCA model and predict values for unseen individuals.
`rgcca_transform`	Use a fitted R/SGCCA model to compute the block components of unseen individuals.
`rgcca_permutation`	Find the best set of parameters for a R/SGCCA model using a permutation strategy.
`rgcca_bootstrap`	Evaluate the significance of the block-weight vectors produced by a R/SGCCA model using bootstrap.
`rgcca_stability`	Select the most stable variables of a R/SGCCA model using their VIPs.
`summary/print/plot`	Summary, print and plot methods for outputs of functions `rgcca`, `rgcca_cv`, `rgcca_permutation`, `rgcca_bootstrap`, and `rgcca_stability`.

These functions are detailed hereafter.

- `rgcca()`

The cornerstone of the RGCCA package is the `rgcca()` function. This function enables the construction of a flexible pipeline encompassing all model-building steps. This master function automatically checks the proper formatting of the blocks and performs preprocessing steps based on the `scale_block` and `scale` arguments. Several user-specified strategies for handling missing data are available through the `NA_method` argument. The final step of the pipeline explicitly focuses on the choice of the multiblock component methods. The list of pre-specified multiblock component methods that can be used within the **RGCCA** package is 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"
```

These method can be retrieved by setting the `method` argument of the `rgcca()` function. Alternatively, several arguments (`tau`, `sparsity`, `scheme`, `connection`, `comp_orth`, `superblock`) allow users to manually recover the various methods listed in Section 2.3.

- `rgcca_cv`

The optimal tuning parameters can be determined by cross-validating different indicators of quality, namely:

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

This cross-validation protocol is made available through the `rgcca_cv()` function. This function can be used in the presence of a response block specified by the `response` argument. The primary objective of this function is to automatically find the set of tuning parameters (shrinkage parameters, amount of sparsity, number of components) that yields the highest cross-validated scores. The prediction model aims to predict the response block from all available block components. `rgcca_cv()` harnesses the power of the `caret` package. As a direct consequence, an astonishingly large number of prediction models becomes accessible (for an exhaustive list, refer to `caret::modelLookup()`). `rgcca_cv()` calls `rgcca_transform()` to obtain the block components associated with the test set and `rgcca_predict()` for making predictions.

- `rgcca_permutation()`

When blocks play a symmetric role (i.e., without a specific response block), a permutation-based strategy, very similar to the one proposed in Witten *et al.* (2009) has also been integrated within the **RGCCA** package through the `rgcca_permutation()` function.

For each set of candidate parameters, the following steps are performed:

1. S/RGCCA is run on the original data $\mathbf{X}_1, \ldots, \mathbf{X}_J$, and we record the value of the objective function, denoted as t.
2. `n_perm` times, the rows of $\mathbf{X}_1, \ldots, \mathbf{X}_J$ are randomly permuted to obtained permuted data sets $\mathbf{X}_1^*, \ldots, \mathbf{X}_J^*$. S/RGCCA is then run on these permuted data sets, and we record the value of the objective function, denoted as t^*.
3. The resulting p-value is determined by calculting the fraction of permuted t^* values that exceeds the non-permuted t value.
4. The resulting zstat is defined as $\frac{t - \text{mean}(t^*)}{\text{sd}(t^*)}$.

The best set of tuning parameters is then the set that yields the highest zstat. This procedure is available through the `rgcca_permutation()` function.

- `rgcca_bootstrap()`

Once a fitted model has been obtained, the `rgcca_bootstrap()` function can be used to assess the reliability of parameter estimates (block-weight/loading vectors). Bootstrap samples of the same size as the original data are 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 derive 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 the reliability of parameter estimates.

Since multiple p-values are constructed simultaneously, correction for multiple testing applies and, adjusted p-values are returned.

- `rgcca_stability()`

We also implemented a procedure to stabilize the selection of variables in SGCCA. Tenenhaus (1998) defines the variable importance in projection (VIP) score for the PLS method. This score can be used for variable selection: the higher the score, the more important the variable. We use this idea to propose a procedure for evaluating the stability of the variable selection procedure of SGCCA. This procedure relies on the following score:

$$\text{VIP}(\mathbf{x}_{jh}) = \frac{1}{K}\sum_{k=1}^{K}\left(\mathbf{a}_{jh}^{(k)2}\text{AVE}\left(\mathbf{X}_j^{(k)}\right)\right). \tag{21}$$

SGCCA is applied several times using a bootstrap resampling procedure. For each fitted model, the VIPs are computed for each variable. The higher VIPs averaged over the different models are kept. This procedure is available through the `rgcca_stability()` function.

The outputs of most of the aforementioned functions can be described and visualized using the generic `summary()`, `print()` and `plot()` functions.

The following two sections provide examples illustrating the practical applications of these functions.

3.2. Case study: the Russett dataset

In this section, we 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 dataset (Russett 1964) is 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 $\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3$ are arranged in the list format.

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

Preprocessing. In general, and especially for the covariance-based criterion, the data blocks might be preprocessed to ensure comparability between variables and blocks. In order to ensure comparability between variables, standardization is applied (zero mean and unit variance). Such preprocessing is reached by setting the `scale` argument to `TRUE` (default value) in the `rgcca()` function. A possible strategy to make blocks comparable is to standardize the variables and 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 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 agricultural inequality is above average and industrial development is 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 $\mathbf{X}_1$ to $\mathbf{X}_3$ ($c_{13} = 1$), $\mathbf{X}_2$ to $\mathbf{X}_3$ ($c_{23} = 1$) and to not connect $\mathbf{X}_1$ to $\mathbf{X}_2$ ($c_{12} = 0$). The resulting design matrix $\mathbf{C}$ is:

```
R> C <- matrix(c(0, 0, 1,
+                0, 0, 1,
```

```
+                1, 1, 0), 3, 3)
R>
R> C

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

Choice of the scheme function g. Typical choices of scheme functions are $g(x) = x, x^2$, or $|x|$. According to Van de Geer (1984), a fair model is a model where all blocks contribute equally to the solution in opposition to a model dominated by only a few of the J sets. If fairness is a major objective, the user must choose $m = 1$. $m > 1$ is preferable if the user wants to discriminate between blocks. In practice, m is equal to 1, 2 or 4. The higher the value of m, the more the method acts as block selector (Tenenhaus *et al.* 2017).

RGCCA using the pre-defined design matrix $\mathbf{C}$, the factorial scheme ($g(x) = x^2$), $\tau = 1$ for all blocks (full covariance criterion) and a number of (orthogonal) components equal to 2 for all blocks is obtained by specifying appropriately the arguments `connection`, `scheme`, `tau`, `ncomp`, `comp_orth` in `rgcca()`. `verbose` (default value = `TRUE`) indicates that the progress will be reported while computing and that a plot illustrating the convergence of the algorithm will be displayed.

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

The summary() and plot() functions. The `summary()` function allows summarizing the RGCCA analysis.

```
R> summary(fit)

Call: method='rgcca', superblock=FALSE, scale=TRUE, scale_block=FALSE, init='svd',
bias=TRUE, tol=1e-08, NA_method='na.ignore', ncomp=c(2,2,2), response=NULL,
comp_orth=TRUE
There are J = 3 blocks.
The design matrix is:
            Agriculture Industrial Politic
Agriculture           0          0       1
Industrial            0          0       1
Politic               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 Agriculture is: 1
The regularization parameter used for Industrial is: 1
The regularization parameter used for Politic is: 1
```

The block-weight vectors solution of the optimization problem (1) are available as an output of the `rgcca()` function in `fit$a` and correspond exactly to the weight vectors reported in Figure 5 of Tenenhaus and Tenenhaus (2011). 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 comp accordingly. The $\mathbf{a}_j^{(k)\star}$, mentioned in Section Higher level RGCCA algorithm, are available in fit$astar.

```
R> plot(fit, type = "weight", block = 1:3, comp = 1,
+       display_order = FALSE, cex = 2)
```

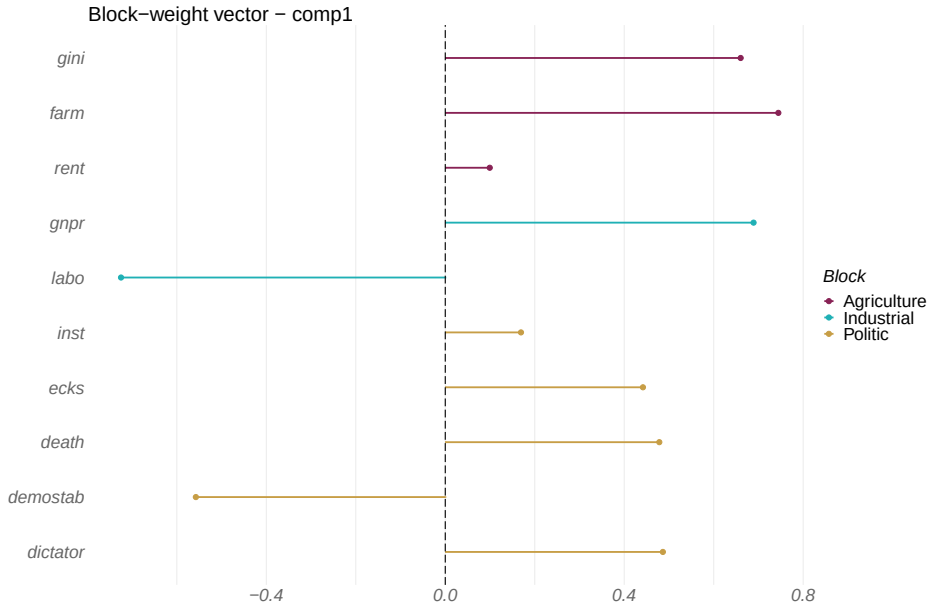

Figure 1: Block-weight vectors of a fitted RGCCA model.

As a component-based method, the **RGCCA** package provides block components as an output of the rgcca() function in fit$Y. Various graphical representations of the block components are available using the plot() function on a fitted RGCCA object by setting the type argument. They include factor plot ("sample"), correlation circle ("cor_circle"), or biplot ("biplot"). This graphical display allows visualization of the sources of variability within blocks, the relationships between variables within and between blocks, and the correlation between blocks. The factor plot of the countries obtained by crossing the block components associated with the agricultural inequality and industrial development blocks, marked by their political regime in 1960, is shown below.

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

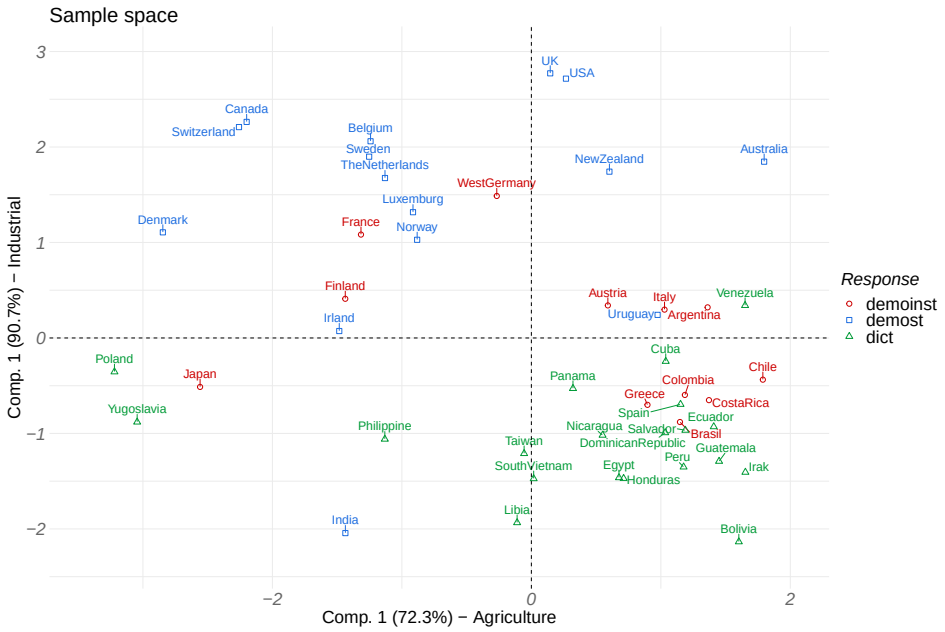

Figure 2: Graphical display of the countries by drawing the block component of the first block against the block component of the second block, colored according to their political regime.

Countries aggregate together when they share similarities. It may be noted that the lower right quadrant concentrates on dictatorships. It is difficult for a country to escape dictatorship when its industrial 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 AVEs of the different blocks are reported in the axes of Figure 2. All AVEs (defined in (18)-(20)) are available as output of the `rgcca()` function in `fit$AVE`. These indicators of model quality can also be visualized using the generic `plot()` function.

```
R> plot(fit, type = "ave", cex = 2)
```

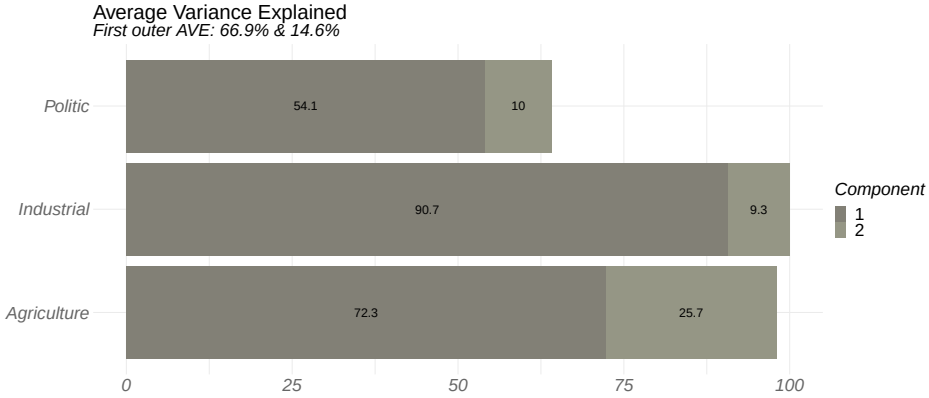

Figure 3: Average variance explained of the different blocks.

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

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

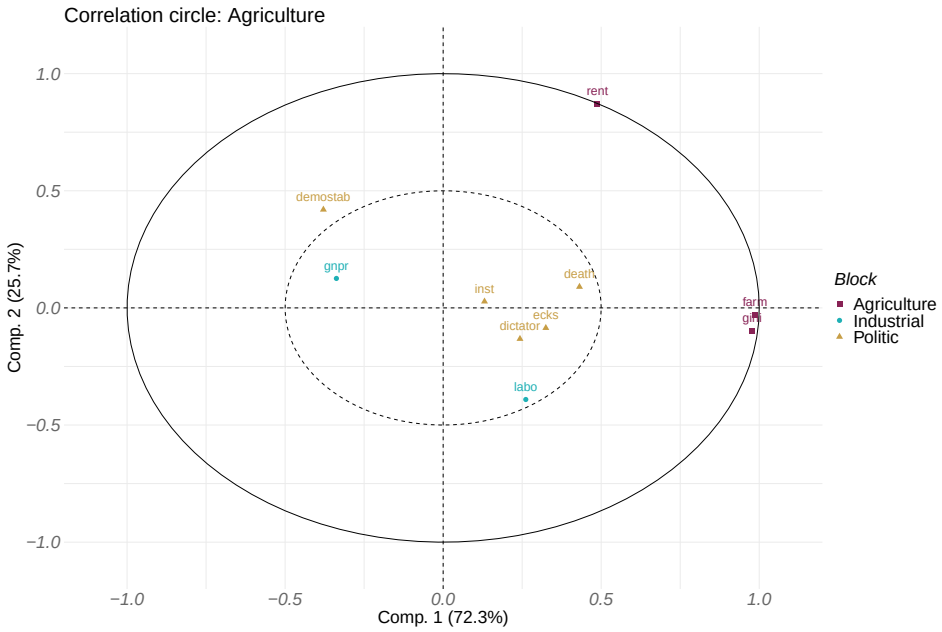

Figure 4: Correlation circle associated with the first two components of the first block.

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

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

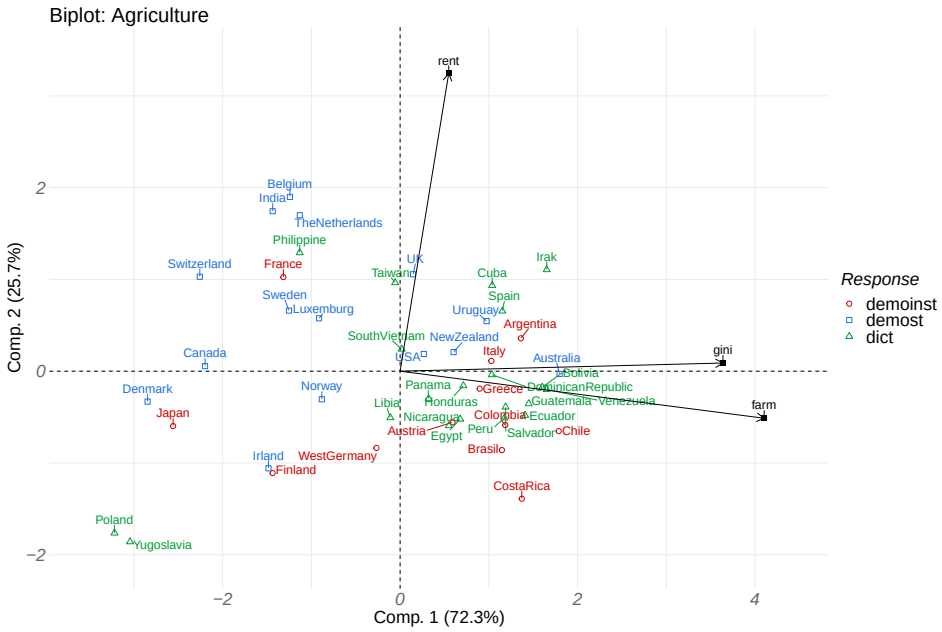

Figure 5: Biplot associated with the first two components of the first block.

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

Bootstrap confidence intervals. We illustrate the use of the `rgcca_bootstrap()` function. It is possible to set the number of bootstrap samples using the `n_boot` argument:

```
R> set.seed(0)
R> boot_out <- rgcca_bootstrap(fit, n_boot = 500, n_cores = 1)
```

The bootstrap results are detailed using the `summary()` function,

```
R> summary(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='na.ignore', ncomp=c(2,2,2), response=NULL,
comp_orth=TRUE
There are J = 3 blocks.
The design matrix is:
            Agriculture Industrial Politic
Agriculture           0          0       1
Industrial            0          0       1
Politic               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 0.0696      0.4744       0.725           9.490 0.0000
farm       0.7445  0.7304 0.0555      0.6443       0.828          13.423 0.0000
```

```
rent        0.0994  0.0762 0.2203     -0.3943       0.454          0.451 0.4663
gnpr        0.6891  0.6894 0.0298      0.6252       0.739         23.140 0.0000
labo       -0.7247 -0.7232 0.0278     -0.7804      -0.673        -26.087 0.0000
inst        0.1692  0.1672 0.1174     -0.0801       0.357          1.442 0.0917
ecks        0.4418  0.4340 0.0592      0.3224       0.545          7.458 0.0000
death       0.4784  0.4699 0.0483      0.3769       0.560          9.914 0.0000
demostab   -0.5574 -0.5520 0.0509     -0.6400      -0.432        -10.942 0.0000
dictator    0.4864  0.4831 0.0524      0.3846       0.586          9.285 0.0000
          adjust.pval
gini            0.000
farm            0.000
rent            0.523
gnpr            0.000
labo            0.000
inst            0.131
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 = 2,
+       show_stars = TRUE)
```

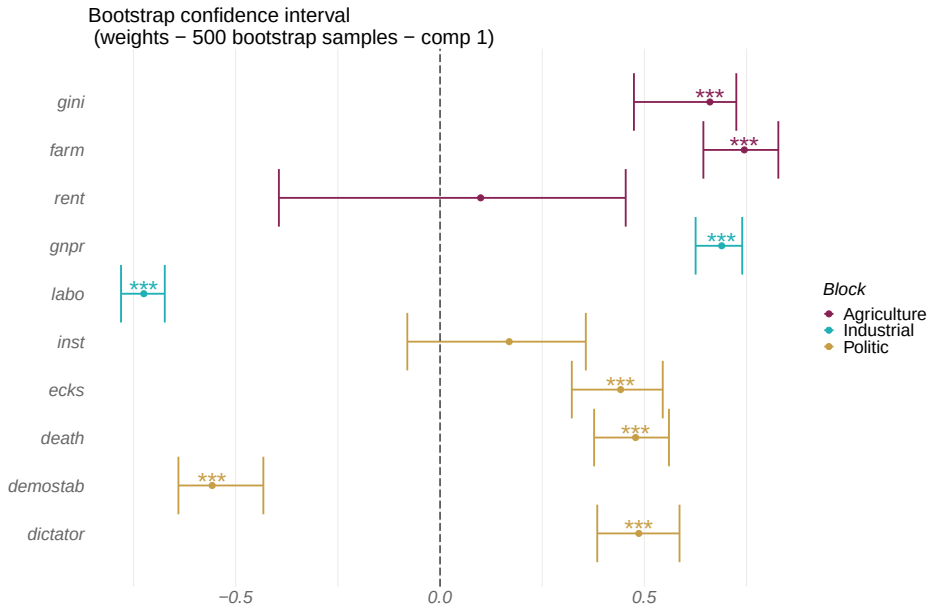

Figure 6: Bootstrap confidence intervals for the block-weight vectors.

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

Choice of the shrinkage parameters. In the previous section, the shrinkage parameters were manually set. However, the **RGCCA** package introduces three fully automated strategies for selecting the optimal shrinkage parameters.

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

```
R> fit <- rgcca(blocks = A, connection = C,
+               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 before, all the fitted RGCCA objects can be visualized/bootstrapped using the `summary()`, `plot()` and `rgcca_bootstrap()` functions.

Permutation strategy. We illustrate the use of the `rgcca_permutation()` function here. The number of permutations can be set using the `n_perms` argument:

```
R> set.seed(0)
R> perm_out <- rgcca_permutation(blocks = A, connection = C,
+                                par_type = "tau",
+                                par_length = 10,
+                                n_cores = 1,
+                                n_perms = 10)
```

By default, the `rgcca_permutation` function generates 10 sets of tuning parameters uniformly between some minimal values (0 for RGCCA and 1/sqrt(ncol) for SGCCA) and 1. Results of the permutation procedure are summarized using the generic `summary()` function,

```
R> summary(perm_out)
```

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

The factorial scheme is used.

Tuning parameters (tau) used:
   Agriculture Industrial Politic
1        1.000      1.000   1.000
2        0.889      0.889   0.889
3        0.778      0.778   0.778
```

```
4         0.667       0.667    0.667
5         0.556       0.556    0.556
6         0.444       0.444    0.444
7         0.333       0.333    0.333
8         0.222       0.222    0.222
9         0.111       0.111    0.111
10        0.000       0.000    0.000

   Tuning parameters Criterion Permuted criterion     sd zstat p-value
1     1.00/1.00/1.00     0.708              0.0909 0.0479 12.88       0
2     0.89/0.89/0.89     0.758              0.0993 0.0512 12.86       0
3     0.78/0.78/0.78     0.814              0.1093 0.0549 12.83       0
4     0.67/0.67/0.67     0.878              0.1214 0.0592 12.80       0
5     0.56/0.56/0.56     0.953              0.1365 0.0640 12.76       0
6     0.44/0.44/0.44     1.040              0.1561 0.0696 12.70       0
7     0.33/0.33/0.33     1.144              0.1829 0.0764 12.58       0
8     0.22/0.22/0.22     1.273              0.2233 0.0854 12.29       0
9     0.11/0.11/0.11     1.449              0.2974 0.1007 11.44       0
10    0.00/0.00/0.00     1.934              0.6049 0.1419  9.37       0
The best combination is: 1.00/1.00/1.00 for a z score of 12.9 and a p-value of 0
```

and displayed using the generic `plot()` function.

```
R> plot(perm_out, cex = 2)
```

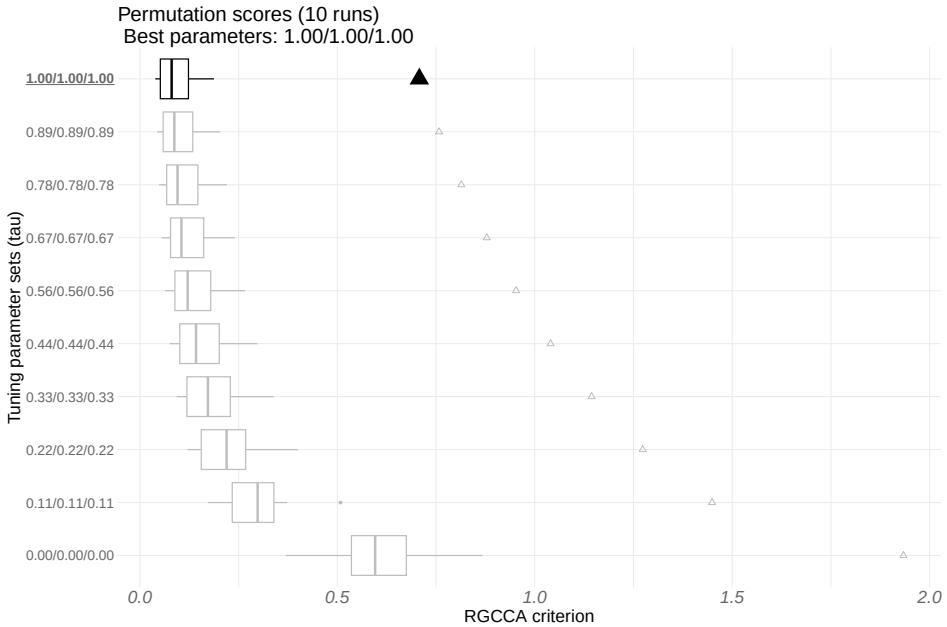

Figure 7: Values of the objective function of RGCCA against the sets of tuning parameters, triangles correspond to evaluations on non-permuted datasets.

The fitted permutation object, `perm_out`, can be directly provided as the 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. Alternatively, a numeric vector of length J indicating the maximum range values to be tested can be given. The set of parameters is then uniformly generated between the minimum values (0 for RGCCA and $1/\text{sqrt(ncol)}$ for SGCCA) and the maximum values specified by the user with `par_value`.

Cross-validation strategy. The optimal tuning parameters can also be obtained by cross-validation. In the forthcoming section, we will illustrate this strategy in the second case study.

RGCCA with a superblock. To conclude this section on the analysis of the Russett dataset, we consider multiple co-inertia analysis (Chessel and Hanafi 1996) (MCOA, also called MCIA in Cantini, Zakeri, Hernandez, Naldi, Thieffry, Remy, and Baudot 2021) with 2 components per block.

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

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

Interestingly, the `summary()` function reports the arguments implicitly specified to perform MCOA.

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

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

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

The regularization parameter used for Agriculture is: 1
The regularization parameter used for Industrial is: 1
The regularization parameter used for Politic is: 1
The regularization parameter used for superblock is: 0
```

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

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

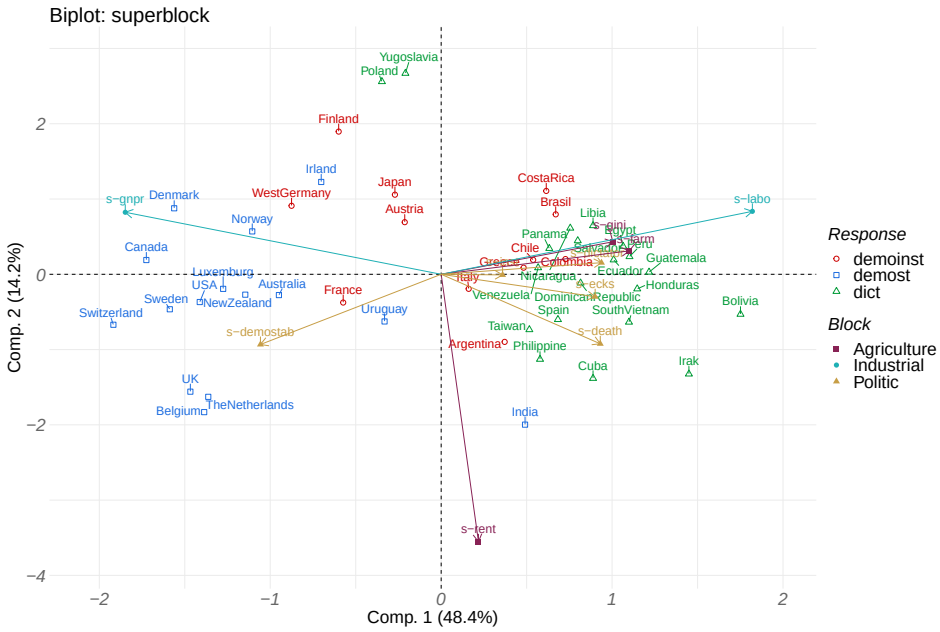

Figure 8: Biplot of the countries obtained by crossing the two first components of the superblock. Individuals are colored according to their political regime and variables according to their block membership.

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

3.3. High dimensional case study: the glioma study

Biological problem. Brain tumors are children's most common solid tumors and have the highest mortality rate of all pediatric cancers. Despite advances in multimodality therapy, children with pediatric high-grade gliomas (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 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 into 3 blocks of variables defined for the 53 tumors: the first block $\mathbf{X}_1$ provides the expression of 15702 genes (GE). The second block $\mathbf{X}_2$ contains the imbalances of 1229 segments (CGH) of chromosomes. $\mathbf{X}_3$ is a block of dummy variables describing the categorical variable location. One dummy variable has been left out because of redundancy with the others.

The next lines of code can be run to download the dataset from `http://biodev.cea.fr/sgcca/`:

```
R> if (!("gliomaData" %in% rownames(installed.packages()))) {
+   destfile <- tempfile()
+   download.file("http://biodev.cea.fr/sgcca/gliomaData_0.4.tar.gz", destfile)
+   install.packages(destfile, repos = NULL, type = "source")
+ }

R> data("ge_cgh_locIGR", package = "gliomaData")
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> vapply(blocks, NCOL, FUN.VALUE = 1L)
```

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 predicting the location. The argument `response = 3` of the `rgcca()` function encodes this design.

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

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

The primal/dual RGCCA algorithm. 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
```

The dual formulation makes the RGCCA algorithm highly efficient, even in a high-dimensional setting.

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

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

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

For easier interpretation of the results, especially in high-dimensional settings, adding penalties promoting sparsity within the RGCCA optimization problem is often appropriate. 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 controls 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:

$$\forall h, \|\mathbf{a}_j^{(h)}\|_{\ell_1} \leq \texttt{sparsity}_j \sqrt{p_j}, \tag{22}$$

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

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

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

SGCCA for the Glioma dataset. The algorithm associated with the optimization problem (5) is available through the `rgcca()` function with the argument `method = "sgcca"` or by specifying directly the `sparsity` argument as below.

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

The `summary()` function allows summarizing the SGCCA analysis,

```
R> summary(fit.sgcca)
```

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

Determining the sparsity parameters by cross-validation. Of course, it is still possible to determine the optimal sparsity parameters by permutation. This is made possible by setting the `par_type` argument to `"sparsity"` (instead of `"tau"`) within the `rgcca_permutation()` function. However, in this section, we will adopt a different approach. The optimal tuning parameters can be determined by cross-validating using different indicators of quality for classification and regression.

This cross-validation protocol is made available through the `rgcca_cv` function and is used here to predict the tumors' location. In this situation, the goal is to maximize the cross-validated balanced accuracy (`metric = "Balanced_Accuracy"`) in a model where we try to predict the response block from all the block components with a user-defined classifier (`prediction_model = "lda"`). The cross-validation protocol is set by specifying the arguments like the number of folds (`k`) and the number of cross-validations to be run (`n_run`). Also, we decide to upper bound the sparsity parameters for $\mathbf{X}_1$ and $\mathbf{X}_2$ to 0.2 to achieve an attractive amount of sparsity.

```
R> set.seed(0)
R> in_train <- caret::createDataPartition(
+   blocks[[3]], p = .75, list = FALSE
+ )
R> training <- lapply(blocks, function(x) as.matrix(x)[in_train, , drop = FALSE])
R> testing <- lapply(blocks, function(x) as.matrix(x)[-in_train, , drop = FALSE])
R>
R> cv_out <- rgcca_cv(blocks = training, response = 3,
+                     par_type = "sparsity",
+                     par_value = c(.2, .2, 0),
+                     par_length = 10,
+                     prediction_model = "lda",
+                     validation = "kfold",
+                     k = 7, n_run = 3, metric = "Balanced_Accuracy",
+                     n_cores = 1)
```

`rgcca_cv()` relies on the **caret** package. As a direct consequence, an astonishingly large number of models are made available (see `caret::modelLookup()`). Results of the cross-validation procedure are reported using the generic `summary()` function,

```
R> summary(cv_out)
```

and displayed using the generic `plot()` function.

```
R> plot(cv_out, cex = 2)
```

As before, the optimal sparsity parameters can be used to fit a new model, and the resulting optimal model can be visualized/bootstrapped.

```
R> fit <- rgcca(cv_out)
R> summary(fit)
```

Note that the sparsity parameter associated with $\mathbf{X}_3$ switches automatically 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 is correlated with its neighboring components.

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

```
R> pred <- rgcca_predict(fit, blocks_test = testing, prediction_model = "lda")
```

and a **caret** summary of the performances can be reported.

```
R> pred$confusion$test
```

The `rgcca_transform()` function can be used if, for a specific reason, only the block components are wanted for the test set.

```
R> projection <- rgcca_transform(fit, blocks_test = testing)
```

Stability selection. We finally illustrate the use of the `rgcca_stability()` function:

```
R> set.seed(0)
R> fit_stab <- rgcca_stability(fit,
+                              keep = vapply(
+                                fit$a, function(x) mean(x != 0),
+                                FUN.VALUE = 1.0
+                              ),
+                              n_boot = 100, verbose = TRUE, n_cores = 1)
```

Once the most stable variables have been found, a new model using these variables is automatically fitted. This last model can be visualized using the usual `summary()` and `plot()` functions. We can finally apply the bootstrap procedure on the most stable variables.

```
R> set.seed(0)
R> boot_out <- rgcca_bootstrap(fit_stab, n_boot = 500)
```

The bootstrap results can be visualized using the generic `plot()` function. We use the `n_mark` parameter to display the top 50 variables of GE.

```
R> plot(boot_out, block = 1,
+       display_order = FALSE,
+       n_mark = 50, cex = 1.5, cex_sub = 17,
+       show_star = TRUE)
```

4. Conclusion

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

- Several strategies for determining the shrinkage parameters/level of sparsity automatically: Schaffer & Strimmer's analytical formulae, cross-validation, or permutation strategy.
- A bootstrap resampling procedure for assessing the reliability of the parameter estimates of S/RGCCA.
- Dedicated functions for graphical displays of the output of RGCCA (sample plot, correlation circle, biplot, ...).
- Various deflation strategies for obtaining orthogonal block-components or orthogonal block-weight vectors.
- Strategies for handling missing data. Specifically, 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, we exploit the algorithmic solution proposed for PLS path modeling to deal with missing data (see Tenenhaus *et al.* 2005).
- Special attention has been paid to providing a bunch of "mathematical" unit tests which, in a sense, guarantee the implementation quality. Also, when appropriate, a particular focus was given to recovering the results of other R packages of the literature, including **ade4** and **FactoMineR**.

The **RGCCA** package will be a valuable resource for researchers and practitioners who are interested in multiblock data analysis to gain new insights and improve decision-making.

The RGCCA framework is constantly evolving and extending. Indeed, we proposed RGCCA for multigroup data (Tenenhaus *et al.* 2014), RGCCA for multiway data (Gloaguen, Philippe, Frouin, Gennari, Dehaene-Lambertz, Le Brusquet, and Tenenhaus 2020; Girka, Gloaguen, Brusquet, Zujovic, and Tenenhaus 2023), and RGCCA for (sparse and irregular) functional data (Sort, Tenenhaus, and Le Brusquet 2023). In addition, maximizing successive criteria may be sub-optimal from an optimization point of view, where a single global criterion is preferred. A global version of RGCCA (Gloaguen 2020), which allows simultaneously extracting several components per block (no deflation procedure required), has been proposed. Also, it is possible to use RGCCA in structural equation modeling with latent and emergent variables for obtaining consistent and asymptotically normal estimators of the parameters (Tenenhaus, Tenenhaus, and Dijkstra 2023). At last, several alternatives for handling missing values are discussed in Peltier, Le Brusquet, Lejeune, Moszer, and Tenenhaus (2022). Work in progress includes the integration of all these novel approaches in the next release of the **RGCCA** package. The modular design of the `rgcca()` function will greatly simplify the integration of these extensions into the package.

Acknowledgments

This project has received funding from UDOPIA - ANR-20-THIA-0013, the European Union's Horizon 2020 research and innovation program under grant agreement No 874583 and the AP-HP Foundation, within the framework of the AIRACLES Chair.

References

Andrew G, Arora R, Bilmes J, Livescu K (2013). "Deep canonical correlation analysis." In *International conference on machine learning*, pp. 1247–1255. PMLR.

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

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.

Bougeard S, Qannari EM, Rose N (2011). "Multiblock redundancy analysis: interpretation tools and application in epidemiology." *Journal of Chemometrics*, **25**(9), 467–475.

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

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

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

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

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

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

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

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

Ding DY, Li S, Narasimhan B, Tibshirani R (2022). "Cooperative learning for multiview analysis." *Proceedings of the National Academy of Sciences*, **119**(38), e2202113119.

Dray S, Dufour AB (2007). "The ade4 package: implementing the duality diagram for ecologists." *Journal of statistical software*, **22**, 1–20.

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

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

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

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

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

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

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

Gower JC (1975). "Generalized procrustes analysis." *Psychometrika*, **40**, 33–51.

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, Szedmak S, Shawe-Taylor J (2004). "Canonical Correlation Analysis: An Overview with Application to Learning Methods." *Neural Computation*, **16 (12)**, 2639–2664.

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

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

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

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

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

Lavit C, Escoufier Y, Sabatier R, Traissac P (1994). "The act (statis method)." *Computational Statistics & Data Analysis*, **18**(1), 97–119.

Lê S, Josse J, Husson F (2008). "FactoMineR: A Package for Multivariate Analysis." *Journal of Statistical Software*, **25**(1), 1–18. doi:10.18637/jss.v025.i01.

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 B*, **55**, 725–740.

Liland KH (2022). *multiblock: Multiblock Data Fusion in Statistics and Machine Learning*. R package version 0.8.1, URL https://CRAN.R-project.org/package=multiblock.

Lock EF, Hoadley KA, Marron JS, Nobel AB (2013). "Joint and individual variation explained (JIVE) for integrated analysis of multiple data types." *The annals of applied statistics*, **7**(1), 523.

Peltier C, Le Brusquet L, Lejeune FX, Moszer I, Tenenhaus A (2022). "Missing Values in RGCCA: Algorithms and Comparisons." In *8th International Conference on Partial Least Squares Structural Equation Modeling (PLS'22)*.

Perry R, Mischler G, Guo R, Lee T, Chang A, Koul A, Franz C, Richard H, Carmichael I, Ablin P, *et al.* (2021). "mvlearn: Multiview machine learning in python." *The Journal of Machine Learning Research*, **22**(1), 4938–4944.

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.

R Core Team (2022). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Rohart F, Gautier B, Singh A, Lê Cao KA (2017). "mixOmics: An R package for 'omics feature selection and multiple data integration." *PLOS Computational Biology*, **13**(11), 1–19. doi:10.1371/journal.pcbi.1005752. URL https://doi.org/10.1371/journal.pcbi.1005752.

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.

Schouteden M, Van Deun K, Wilderjans TF, Van Mechelen I (2014). "Performing DISCO-SCA to search for distinctive and common information in linked data." *Behavior research methods*, **46**, 576–587.

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.

Sort L, Tenenhaus A, Le Brusquet L (2023). "Functional Generalized Canonical Correlation Analysis for studying jointly several longitudinal responses." *submitted to Biostatistics*.

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

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

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

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

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

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

Tenenhaus A, Tenenhaus M, Dijkstra T (2023). "Structural Equation Modeling with Latent/Emergent Variables: RGCCAc." *Submitted to Psychometrika*.

Tenenhaus M (1998). *La régression PLS: théorie et pratique*. Editions technip.

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

Tenenhaus M, Vinzi VE, Chatelin YM, Lauro C (2005). "PLS path modeling." *Computational statistics & data analysis*, **48**(1), 159–205.

Trendafilov NT (2010). "Stepwise estimation of common principal components." *Computational Statistics & Data Analysis*, **54**(12), 3446–3457.

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

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

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

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

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

Wickham H (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. ISBN 978-3-319-24277-4. URL https://ggplot2.tidyverse.org.

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 & Sons, New York*, pp. 581–591.

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

Wold, S and Kettaneh, N and Tjessem, K (1996). "Hierarchical multiblock PLS and PC models for easier model interpretation and as an alternative to variable selection." *Journal of Chemometrics*, **10**, 463–482.

Zou H, Hastie T, Tibshirani R (2006). "Sparse principal component analysis." *Journal of Computational and Graphical Statistics*, **15**, 265–286.

Affiliation:

Fabien Girka
Laboratoire des Signaux et Systèmes
Université Paris-Saclay, CNRS, CentraleSupélec,
Gif-sur-Yvette, 91190, France
Institut du Cerveau de Paris (ICM), Sorbonne Université
Assistance Publique-Hôpitaux de Paris (AP-HP)
E-mail: fabien.girka@centralesupelec.fr

Etienne Camenen
INSERM
Inserm U1127, CNRS UMR 7225,
Hôpital Universitaire Pitié-Salpêtrière
75013 Paris, France
Institut du Cerveau de Paris (ICM), Sorbonne Université
Assistance Publique-Hôpitaux de Paris (AP-HP)

Caroline Peltier
INRAE
Centre des Sciences du Goût et de l'Alimentation
CNRS, INRAE, Institut Agro
University of Bourgogne F-21000 Dijon
CNRS, INRAE, PROBE research infrastructure
ChemoSens facility, F-21000 Dijon, France
Institut du Cerveau de Paris (ICM), Sorbonne Université
Assistance Publique-Hôpitaux de Paris (AP-HP)

Arnaud Gloaguen
CEA
Centre National de Recherche en Génomique Humaine
Institut de Biologie François Jacob
CEA, Université Paris-Saclay, Évry, France

Vincent Guillemot
Institut Pasteur
Université Paris Cité
Bioinformatics and Biostatistics Hub
F-75015 Paris, France

Laurent Le Brusquet
Laboratoire des Signaux et Systèmes
Université Paris-Saclay, CNRS, CentraleSupélec
Gif-sur-Yvette, 91190, France

Arthur Tenenhaus
Laboratoire des Signaux et Systèmes
Université Paris-Saclay, CNRS, CentraleSupélec
Gif-sur-Yvette, 91190, France
Institut du Cerveau de Paris (ICM), Sorbonne Université
Assistance Publique-Hôpitaux de Paris (AP-HP)
E-mail: arthur.tenenhaus@centralesupelec.fr

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