Sparse PCA for Multi-Block Data

Proefschrift ter verkrijging van de graad van doctor aan Tilburg University
op gezag van de rector magnificus, prof. dr. W.B.H.J. van de Donk,
in het openbaar te verdedigen ten overstaan van een
door het college voor promoties aangewezen commissie in de
Aula van de Universiteit op vrijdag 21 mei 2021 om 10:00 uur

door

Niek Cornelis de Schipper,

geboren te Eindhoven

Colophon

Printing was financially supported by Tilburg University.

Printed by:	printenbind
Cover photo by:	Shifaaz Shamoon on Unsplash
Licence:	CC-BY 4.0

Table of Contents

Chapter 1

Introduction

1.1 Background

Researchers are sometimes faced with a situation where they can supplement their data with other data types for the same individuals. For example, besides having questionnaire data, researchers might also have say experience sampling data, online behavior data, or genetic data on the same subjects. We refer to each of the different data types as a data block. Linking multiple data blocks together holds promising prospects as it allows studying relationships as the result of the concerted action of multiple determinants. For example, having both questionnaire data on eating and health behavior and data on genetic variants for the same subjects holds the key to finding how genes and environment act together in the emergence of eating disorders. Indeed, for most psycho-pathologies and many other behavioral outcomes, it holds that these are the result of a genetic susceptibility in combination with a risk provoking environment (Halldorsdottir and Binder, 2017). Thus, analyzing multiple data blocks together could provide us with crucial insights into the complex interplay between the multiple factors that determine human behavior.

A powerful way of gaining insight into such data sets consisting of multiple blocks is by means of latent variable modelling techniques. One of such techniques is simultaneous component analysis, which is the main approach discussed in this thesis. But, let us first consider the single block version, principal component analysis (PCA; Jolliffe, 1986), for a data set consisting of I rows or individuals and J columns or variables. A PCA with Q components decomposes the $I \times J$ data block $\mathbf{X}$ as follows:

$$\begin{aligned} \mathbf{X} &= \mathbf{XWP}^T + \mathbf{E} \\ &= \mathbf{TP}^T + \mathbf{E}. \end{aligned} \tag{1.1}$$

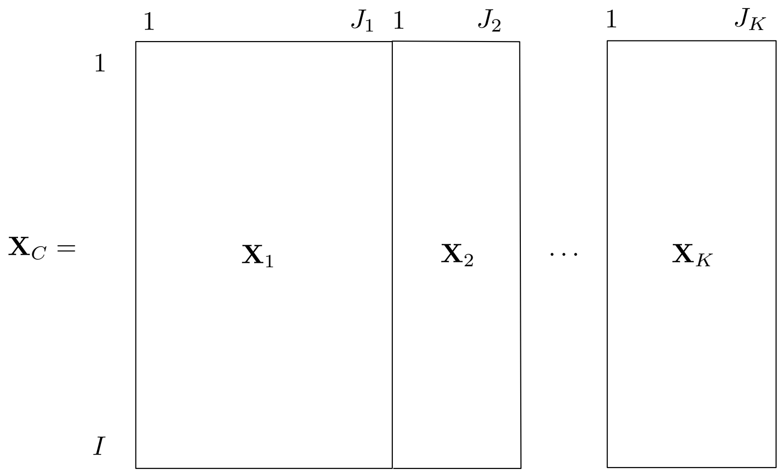


Figure 1.1. Example of linked data set: K data blocks concatenated together where each data block $\mathbf{X}_k$ contains J_k variables for the same I subjects

Here, $\mathbf{T}$ is the $I \times Q$ matrix with component scores, $\mathbf{W}$ the $J \times Q$ (with $J \gg Q$) component weight matrix, $\mathbf{P}$ the $J \times Q$ loading matrix, and $\mathbf{E}$ the $I \times J$ matrix with residuals. PCA is usually defined with $\mathbf{P}^T\mathbf{P} = \mathbf{I}$ as identification constraint, which is, however, not sufficient to uniquely define $\mathbf{P}$ because there is still rotational freedom. Note that in the above formulation of PCA, the component scores $\mathbf{T}$ are written explicitly as a linear combination of the variables. Let t_{iq} be the component score of subject i on a component q, then $t_{iq} = \sum_{j=1}^{J} x_{ij}w_{jq}$, which clearly shows that the component scores are a linear combination of the variables scores. Insight into the relationships within data set concerned can be gained by deriving meaning to the components; that is, an interpretation for $\mathbf{t}_q$ can be directly inferred from inspecting the weights $\mathbf{w}_q$. For example, if only variables related to depression symptoms have substantial weights, then $\mathbf{t}_q$ can be interpreted to be a depression related component.

The PCA decomposition can be extended to the case where the data set of interest consists of linked blocks. Assume we have K data blocks $\mathbf{X}_k$ for $k = 1 \ldots K$ with in each block J_k variables for the same I subjects. The resulting linked data sets with in total $\sum_k J_k$ variables which depicted in Figure 1.1 is called a multi-block data set (Tenenhaus and Tenenhaus, 2011). The PCA decomposition presented above can also be applied to all data blocks $\mathbf{X}_k$ jointly by treating the multi-block data set as one big matrix with $\sum_k J_k$ columns (variables). That is,

$$\left[\mathbf{X}_1 \ldots \mathbf{X}_K\right] = \left[\mathbf{X}_1 \ldots \mathbf{X}_K\right]\left[\mathbf{W}_1^T \ldots \mathbf{W}_K^T\right]^T\left[\mathbf{P}_1^T \ldots \mathbf{P}_K^T\right] + \left[\mathbf{E}_1 \ldots \mathbf{E}_K\right] \tag{1.2}$$

or in shorthand notation,

$$\begin{aligned} \mathbf{X}_C &= \mathbf{X}_C \mathbf{W}_C \mathbf{P}_C^T + \mathbf{E}_C \\ &= \mathbf{T}\mathbf{P}_C^T + \mathbf{E}_C. \end{aligned} \tag{1.3}$$

This model is referred to as the simultaneous component (SC) model (Kiers and ten Berge, 1989). In the SC model the component score t_{iq} of subject i on component q is $t_{iq} = \sum_{k=1}^{K} \sum_{j_k=1}^{J_k} x_{ij_k} w_{j_k q}$ which is a linear combination of the variables scores of *all* data blocks. Valuable insight in the relationships between multiple blocks can be gained by inspecting in what way the variables from the different blocks are weighted together to form the component scores.

1.2 Aim and outline of the thesis

There are two problems associated with the SC model:

1. Because the component scores in Equation (1.3) are a linear combination of variables of all blocks, all blocks contribute to all components. This is not particularly insightful as it obscures components that are not shared by all data blocks. In order to alleviate this problem, the common sources of variation need to be separated from the distinctive sources of variation. This serves two purposes: first, it increases efficiency of the estimation of the common components (Acar et al., 2014; Lock et al., 2013; Trygg and Wold, 2002), and second, it may be instructive (substantively) to detect such unique sources of variation (Alter et al., 2003; Van Deun et al., 2012). Our strategy will be to model the unique sources of variation by a weight vector containing zero(s) for all blocks except the block for which the component is unique. This imposes absence of the component in all blocks, except for the one for which it is unique. This approach has been shown to have a clear interpretational advantage compared to methods that fail to control such absence (Schouteden et al., 2013; Van Deun et al., 2013).

2. Besides not accounting for the multi-block structure, the formulation in Equation (1.3) does not imply sparsity into the weights; that is, there is no guarantee that a large portions of the weights gets values (close to) zero. Sparse weights are important for the interpretation of the results, which is especially helpful with data sets consisting of a (very) large number of variables, where a solution with many non-zero weights would be very difficult to interpret. This issue has been addressed in the context of single-block data by penalty based approaches such as the elastic net penalty (Zou et al.,

2006), but the problem has not yet been tackled in conjunction with identifying common and distinctive source of variation in the SC model.

This thesis aims at providing solutions to the above two problems. Chapter 2 explores an exhaustive approach which combines constraints imposed on the weights in a block-wise manner to force common and distinctive sources of variation with a sparseness penalty. Chapter 3 explores a more general penalty-based approach for finding common and distinctive sources of variation and, moreover, examines various model selection techniques needed to decide on the parameters of these models. The findings obtained in this chapter apply to both SCA and PCA. Chapter 4 focusses on the analysis of single block data with PCA, meaning that it deviates somewhat from the main theme of the thesis. In this chapter, we present an alternative way of obtaining sparse weights, i.e., by means of cardinality constraints instead of penalties. In Chapter 5, we present the software created for applying the methods developed in the other chapters, which is a freely available package created using the `R` programming language.

Below, we introduce the content of the different chapters in greater detail. It should be noted that the thesis chapters are written in the form of separate journal articles. This led to some overlap, repetition and possibly also inconsistencies in notation across the chapters.

Chapter 2 In Chapter 2, we take a closer look at multi-block analysis with SCA. We argue that the current practice of analyzing multi-block data by merging all data and apply methods developed for a single block of data, is an inappropriate approach that does not guarantee the discovery of the shared processes (shared variation) between blocks. Each block is dominated by specific information that is typical for the kind of processes it measures (e.g., behavioral processes and response tendencies in questionnaire data, biological processes in the genetic data) resulting in higher associations between the variables within a block than between blocks. Hence, an analysis that does not account for the multi-block data structure is highly unlikely to find the linked variables underlying the subtle joint mechanisms at play. In order to tackle this problem, we propose a simultaneous component approach (Kiers, 2000; Van Deun et al., 2009) that introduces both proper constraints and regularization terms, including the LASSO, to account for the presence of dominant block-specific sources of variation and to force variable selection. The usage of this novel approach is illustrate by applying it to publicly available data from the 500 Family Study (Schneider and Waite). Furthermore, we show its potential by comparing it to sparse PCA (Zou et al., 2006), which is the single block alternative that does not account for the presence of dominant block-specific sources of variation.

Chapter 3 In order to get sparse weights in either PCA or SCA models, values for the hyper-parameters of the penalty terms need to be selected, which is a delicate process. Choose a value that is too small, and too many coefficients will be selected making the interpretation of the models difficult. Choose a value that is too high, and you might miss important relationships within and between data blocks. In Chapter 3, we compare various model selection procedures with respect to their ability of finding the hyper-parameter values yielding the correct structure of the data; i.e., selecting the right set of variables both in the single block setting and in the multi-block setting with common and distinct variation. The model selection procedures investigated are cross-validation with the Eigenvector method (Bro et al., 2008), BIC (Guo et al., 2010; Croux et al., 2013), Convex Hull (Wilderjans et al., 2012), and the Index of Sparseness (Gajjar et al., 2017; Trendafilov et al., 2017), which are readily available methods from the existing literature on the estimation of meta-parameters for the weight-based PCA model. For sparse PCA and sparse SCA, we examine these model selection procedures in a simulation study with a single block of data and multi-block data, respectively. In the multi-block case, we assess whether the model selection procedures produce a final model that correctly identifies the joint and individual structure of the components. In order to inform the analysis about the block structure of the variables, we implemented the group LASSO penalty in a block-wise fashion, aiming at either selecting or canceling out data blocks in an automated way.

Chapter 4 In Chapter 4, we present a sparse PCA method relying on cardinality constraints instead of penalties. A well-documented disadvantage of using penalties for introducing sparsity into the coefficients is that these penalties are not intended to find the best subset of variables. That is, these penalties introduce bias in the estimates while reducing their variance. The resulting variable selection process increases the efficiency of the estimators, but it is not designed to recover the true underlying set of variables. To overcome this problem, we present a cardinality constrained alternative to PCA. Instead of penalizing the coefficient in the model, we solve the problem of finding the optimal subset given a number of non-zero coefficients using a surrogate function. For this purpose, we use cardinality constrained regression, which has the sole aim of identifying the true underlying subset of variables. In this chapter, we compare this cardinality constrained PCA to sparse PCA (Zou et al., 2006) estimated with the LARS algorithm.

Chapter 5 In Chapter 5, we introduce an `R` package to perform regularized SCA and PCA with sparsity on the component weights. This package also includes model selection procedures. The procedures developed are based on the work presented in the Chapter 2, 3, and 4. The main parts of the computational code has been written in `C++` to provide maximal efficiency of the underlying numerical computations. The chapter is written as a tutorial with the aim of making the methods developed in this thesis accessible to potential users. It starts with a short introduction to PCA and its multi-block extension SCA, followed by a substantiation of the models that the procedures in this package estimate. After that the `R` implementation of the package is discussed, followed by detailed examples of data analysis and model selection.

Chapter 2

Revealing the joint mechanisms in traditional data linked with Big Data

Abstract

Recent technological advances have made it possible to study human behavior by linking novel types of data to more traditional types of psychological data, for example, linking psychological questionnaire data with genetic risk scores. Revealing the variables that are linked throughout these traditional and novel types of data gives crucial insight into the complex interplay between the multiple factors that determine human behavior, for example, the concerted action of genes and environment in the emergence of depression. Little or no theory is available on the link between such traditional and novel types of data, the latter usually consisting of a huge number of variables. The challenge is to select – in an automated way – those variables that are linked throughout the different blocks, and this eludes currently available methods for data analysis. To fill the methodological gap, we here present a novel data integration method.

Keywords: Linked Data, Variable Selection, Component Analysis, Big Data

2.1 Introduction

In this era of big data, psychological researchers are faced with a situation where they can supplement the data they are accustomed to with novel kinds of data. For example, besides having questionnaire data also other types of data like experience sampling data, online behavior data, GPS coordinates, or genetic data may be available on the same subjects. Linking such additional blocks of information to the more traditional data holds promising prospects as it allows to study human behavior as the result of the concerted action of multiple influences. For example, having both questionnaire data on eating and health behavior together with data on genetic variants for the same subjects holds the key to finding how genes and environment act together in the emergence of eating disorders. Indeed, for most psycho-pathologies and many other behavioral outcomes, it holds that these are the result of a genetic susceptibility in combination with a risk provoking environment (Halldorsdottir and Binder, 2017). Thus, analyzing these traditional data together with novel types of data could provide us with crucial insights into the complex interplay between the multiple factors that determine human behavior.

Revealing the joint mechanisms in these integrated or linked data, such as the interplay between genes and environments, is challenging from a data analysis point of view because of the complex structure of the data. First, there is the novel kind of data that are very different from the traditional data we are used to work with: Instead of consisting of a limited number of targeted measurements, they consist of a huge amount of variables that have been collected without a specific focus. A typical example is so-called genome wide or omics data consisting of several thousands up to several millions of variables, but it is also the case with naturally occurring data like tweets, web page visits, or GPS signals (Paxton and Griffiths, 2017). As there is very little theoretical knowledge about the link between traditional and novel types of data, one is faced with a variable selection problem meaning that a data analysis method is needed that can reveal the relevant variables in an automated way. Such variable selection methods have been a very active research topic in statistics during the last years and led to approaches like lasso regression (Tibshirani, 1996) and sparse component analysis (Zou et al., 2006). Second, the data consist of multiple blocks of data, and interest is in finding shared or joined mechanisms; this means revealing the sets of variables that are linked *throughout the blocks*. Current practice is to merge all data and apply methods developed for a single block of data, for example, state-of-the-art variable selection techniques such as lasso regression and sparse principal component analysis (PCA). This is an inappropriate approach that does not guarantee that variables from each of the blocks will be selected in case of joined mechanisms. First, usually the variables in the novel types of data outnumber those in the traditional data by far. Second, the blocks are dominated by specific information that is typical for the kind of processes they measure (e.g., behavioral processes and response tendencies in questionnaire data, biological processes in the genetic data) resulting in higher associations between the variables within blocks than between blocks. Hence, analyses that do not account for the multi-block structure of the data are highly unlikely to find the linked variables underlying the subtle joint mechanisms at play.

This paper proposes a novel data integration method that tries to overcome both of these challenges. It presents a significant extension of sparse PCA to the case of linked data, also called multi-block data. A simultaneous component approach (Kiers, 2000; Van Deun et al., 2009) is taken, and proper constraints and regularization terms, including the lasso, are introduced to account for the presence of dominant block-specific sources of variation and to force variable selection.

The remainder of this paper is structured as follows: First, we will present the method as an extension of PCA to the multi-block case, and we will introduce an estimation procedure that is scalable to the setting of (very) large data. Second, using empirical data with three blocks of data on parentchild interactions, the substantive added value of singling out block-specific from common sources of variation and of sparse representations will be illustrated. Third, as a proof of concept, we will evaluate the performance of the method in a simulation study and compare it to the current practice of applying sparse PCA. We conclude with a discussion.

2.2 Methods

In this section, first, the notation and data will be introduced; then the model, its estimation, model selection, and some related methods will be discussed.

2.2.1 Notation and description of linked data

In this paper, we will make use of the standardized notation proposed by Kiers (2000): Bold lower- and uppercases will denote vectors and matrices, respectively, superscript ^{T} denotes the transpose of a vector or matrix, and a running index will range from 1 to its uppercase letter (e.g., there is a total of I subjects where i runs from $i = 1 \ldots I$).

The data of interest are linked data, where for the same group of subjects, several blocks of data are analyzed together. A block of data is defined here as a group of variables that are homogeneous in the kind of information they measure (e.g., a set of items, a set of time points, a set of genes). Formally, we have K blocks of data $\mathbf{X}_k$ for $k = 1 \ldots K$ with in each block scores of the same I subjects on the J_k variables making up the linked data set (see Figure 2.1). Such data are called multi-block data (Tenenhaus and Tenenhaus, 2011) and are to be distinguished from multi-set data where scores are obtained on the same set of J variables but for different groups of subjects. Note that this paper is about multi-block data and does not apply to multi-set data. Furthermore, it is assumed that all data blocks consist of continues variables.

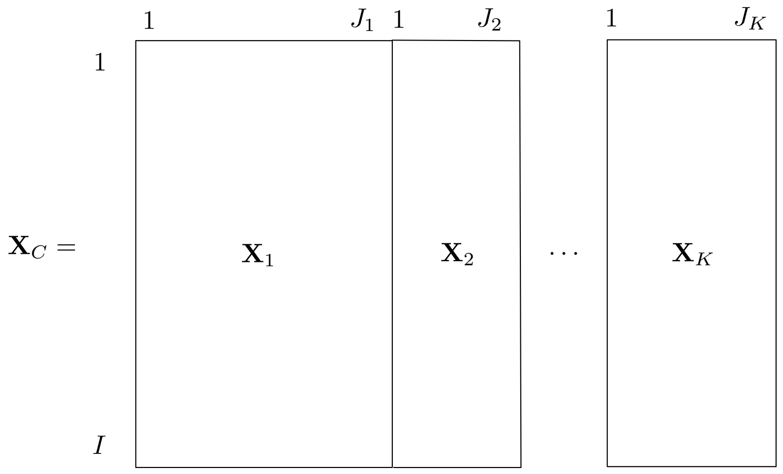


Figure 2.1. Example of linked data set: K data blocks concatenated together where each data block $\mathbf{X}_k$ contains J_k variables for the same I subjects

2.2.2 Model description of PCA and SCA

A powerful method for finding the sources of structural variation is principal component analysis (PCA; Jolliffe, 1986). Applied to a single block of data, PCA decomposes the data of an $I \times J_k$ data block $\mathbf{X}_k$ into,

$$\begin{aligned} \mathbf{X}_k &= \mathbf{X}_k \mathbf{W}_k \mathbf{P}_k^T + \mathbf{E}_k \\ &= \mathbf{T}_k \mathbf{P}_k^T + \mathbf{E}_k, \end{aligned} \tag{2.1}$$

where $\mathbf{W}_k$ denotes the $J_k \times Q$ component weight matrix and $\mathbf{P}_k$ denotes the $J_k \times Q$ loading matrix and $\mathbf{E}_k$ denotes the error matrix. PCA is usually defined with $\mathbf{P}_k^T \mathbf{P}_k = \mathbf{I}$ as identification constraint. In this formulation of PCA the component scores are written explicitly as a linear combination of the variables. Let t_{iq} be the component score of subject i on a component q, then $t_{iq} = \sum_{j_k=1}^{J_k} x_{ij_k} w_{j_k q}$ which clearly shows that the component scores are a linear combination (weighted sum) of the variables scores. The PCA decomposition can also be applied to all $\mathbf{X}_k$ jointly by treating the multi-block data as one big matrix of $\sum_k J_k$ variables,

$$\left[\mathbf{X}_1 \ldots \mathbf{X}_K\right] = \left[\mathbf{X}_1 \ldots \mathbf{X}_K\right] \left[\mathbf{W}_1^T \ldots \mathbf{W}_K^T\right]^T \left[\mathbf{P}_1^T \ldots \mathbf{P}_K^T\right] + \left[\mathbf{E}_1 \ldots \mathbf{E}_K\right] \tag{2.2}$$

or in shorthand notation,

$$\begin{aligned}\mathbf{X}_C &= \mathbf{X}_C \mathbf{W}_C \mathbf{P}_C^T + \mathbf{E}_C \\ &= \mathbf{T}\mathbf{P}_C^T + \mathbf{E}_C.\end{aligned} \tag{2.3}$$

This model is the simultaneous component (SC) model (Kiers and ten Berge, 1989). An important property of SC models is that the same set of component scores underlies each of the data blocks: $\mathbf{X}_k = \mathbf{T}\mathbf{P}_k + \mathbf{E}_k$ for all k. Note that these component scores are a linear combination of *all* the variables contained in the different blocks. Simultaneous components analysis (SCA) as defined in (2.3) does not account for block-specific components nor does it imply variable selection. Therefore, we further extend it.

To account for the presence of block-specific components and to induce variable selection, we introduce particular constraints on the component weights $\mathbf{W}_C$ in the SC model; see model equation (2.3). First, we will discuss the constraints to control for the presence of strong block-specific variation in the linked data, then we will discuss the sparseness constraints.

2.2.3 Common and distinctive components

Consider the following example with two data blocks and three components with imposed blocks of zeroes,

$$\mathbf{T} = \left[\mathbf{X}_1 \quad \mathbf{X}_2\right] \begin{bmatrix} \mathbf{W}_1 \\ \\ \mathbf{W}_2 \end{bmatrix} = \left[\mathbf{X}_1 \quad \mathbf{X}_2\right] \begin{bmatrix} 0 & w_{1_1 2} & w_{1_1 3} \\ \vdots & \vdots & \vdots \\ 0 & w_{J_1 2} & w_{J_1 3} \\ & & \\ w_{1_2 1} & 0 & w_{1_2 3} \\ \vdots & \vdots & \vdots \\ w_{J_2 1} & 0 & w_{J_2 3} \end{bmatrix}. \tag{2.4}$$

(Note that the variable subscripts in (2.4) have their own subscript to denote the block they belong to; for example $w_{1_1 1}$ is the weight of the first variable in the *first block* on the first component while $w_{1_2 1}$ is the weight of the first variable in the *second block* on the first component). Due to the zero constraints, the scores on the first component only depend on the variables in the second block:

$t_{i1} = \sum_{j_1=1}^{J_1} x_{ij_1} w_{j_1 1} + \sum_{j_2=1}^{J_2} x_{ij_2} w_{j_2 1} = \sum_{j_2=1}^{J_2} x_{ij_2} w_{j_2 1}$. Likewise, the scores on the second component only depend on the variables in the first block. Because these components only incorporate the information of one particular type of data, we call them distinctive components as they reflect sources of variation that are particular for a block. These are examples of distinctive components that are formed by a linear combination of variables from *one* particular data block only. The third component $\mathbf{t}_3$ is a linear combination of the variables from both data blocks $\mathbf{X}_1$ and $\mathbf{X}_2$. Hence it reflects sources of variation that play in both data blocks. We call these components common components. If there are more than three blocks the distinction between common and distinctive components can get blurred, for a detailed discussion see Smilde et al. (2017).

Usually the most suitable common and distinctive structure for $\mathbf{W}_C$ given the data is not known. Further on, in Section 2.2.6, we will discuss a strategy that can be used to find the most suitable common and distinctive weight structure for the data at hand.

2.2.4 Sparse common and distinctive components

The component weight matrix in (2.4) has non-zero coefficients for all weights related to the common component and also for the non-zero blocks of the distinctive components. For the common component, for example, this implies that it is determined by all variables; no variable selection takes place. To accomplish variable selection we impose sparseness constraints on the component weight matrix $\mathbf{W}_C$, in addition to the constraints that impose distinctiveness in (2.4), for example,

$$\mathbf{T} = \begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 \end{bmatrix} \begin{bmatrix} 0 & w_{1_1 2} & 0 \\ 0 & 0 & w_{2_1 3} \\ \vdots & \vdots & \vdots \\ 0 & 0 & w_{j_1 3} \\ \vdots & \vdots & \vdots \\ 0 & w_{J_1 2} & 0 \\ \\ 0 & 0 & w_{1_2 3} \\ w_{2_2 1} & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & w_{j_2 3} \\ \vdots & \vdots & \vdots \\ w_{J_2 1} & 0 & 0 \end{bmatrix}. \tag{2.5}$$

In this example model, the common component is a linear combination of some instead of all variables; the same holds for the distinctive components. The number and position of the zeroes are assumed to be unknown. Next, we will introduce a statistical criterion that implies automated selection of the position of the zeroes. How to determine the number of zeroes, or the degree of sparsity, will be discussed in the section on model selection (Section 2.2.6).

2.2.5 Finding sparse common and distinctive components

To find the desired model structure with sparse common and distinctive components, the following optimization criterion is introduced:

$$\begin{gathered}\underset{\mathbf{W}_C,\mathbf{P}_C}{\arg\min} L(\mathbf{W}_C,\mathbf{P}_C) = \|\mathbf{X}_C - \mathbf{X}_C\mathbf{W}_C\mathbf{P}_C^T\|_2^2 + \lambda_1\|\mathbf{W}_C\|_1 + \lambda_2\|\mathbf{W}_C\|_2^2 \\ \text{s.t. } \mathbf{P}_C^T\mathbf{P}_C = \mathbf{I}, \lambda_2, \lambda_1 \geq 0 \text{ and zero block constraints on } \mathbf{W}_C,\end{gathered} \tag{2.6}$$

with the notation $\|.\|_2^2$ denoting the squared Frobenius norm, this is the sum of squared matrix elements, e.g., $\|\mathbf{X}\|_2^2 = \sum_{i,j} x_{ij}^2$ and $\|.\|_1$ denoting the sum of the absolute values of the matrix elements, e.g., $\|\mathbf{X}\|_1 = \sum_{i,j} |x_{ij}|$. The first term in the optimization criterion is the usual PCA least-squares optimization criterion and implies a solution for $\mathbf{W}_C$ and $\mathbf{P}_C$ with minimal squared reconstruction error of the data by the components. The second and the third term are, respectively, the lasso and ridge penalty imposed on the component weight matrix $\mathbf{W}_C$. Both penalties encourage solutions with small weights, this is shrinkage towards zero (to minimize (2.6) not only a good fit is needed, but also weights that are as small as possible). The lasso has the additional property of setting weights exactly to zero (Tibshirani, 1996), introducing variable selection. The ridge penalty is needed in addition to the lasso penalty, because it leads to stabler estimates for $\mathbf{W}_C$ and eases the restriction that only I coefficients can be selected, which is the case when only the lasso penalty is used (Zou and Hastie, 2005). The tuning parameters λ_1 and λ_2 are the costs associated with the penalties, a larger value for the tuning parameter means that having large weights is more expensive, and thus imply more shrinkage of the weights or — in case of the lasso — also more zero component weights. The ridge and lasso regularization together with the common and distinctive component weight constraints, can lead to the desired component weight estimates as outlined in (2.5). Note that the function in (2.6) also includes the special cases of PCA (when $\lambda_1 = 0$ and $\lambda_2 = 0$ and there are no constraints on $\mathbf{W}_C$) and of sparse PCA as presented by Zou et al. (2006) (when there are no constraints $\mathbf{W}_C$).

We call this novel approach of finding sparse common and distinctive components by minimizing (2.6), SCaDS, short for: sparse common and distinctive

SCA. In order to find the estimates $\mathbf{W}_C$ and $\mathbf{P}_C$ of SCaDS given a fixed number of components, values for λ_1, λ_2, and zero block constraints for $\mathbf{W}_C$, we make use of a numerical procedure that alternates between the estimation of $\mathbf{W}_C$ and $\mathbf{P}_C$ until the conditions for stopping have been met. Conditional on fixed values for $\mathbf{W}_C$ there is an analytic solution for $\mathbf{P}_C$, see for example ten Berge (1993) and Zou et al. (2006); for the conditional update of $\mathbf{W}_C$ given fixed values for $\mathbf{P}_C$ we use a coordinate descent procedure (see for example Friedman et al. (2010)). Our choice for coordinate descent is motivated by computational efficiency, meaning that it can be implemented in a way that it is a very fast procedure and scalable to the setting of thousands or even millions of variables without having to rely on specialized computing infrastructure. Another advantage is that constraints on the weights can be accommodated in a straightforward way because of the fact that each weight is updated in turn, conditional upon fixed values for the other weights; hence, weights that are constrained to have a set value are not updated. The derivation of the estimates for the component loadings and weights is detailed in Appendix 2.6.2.

The alternating procedure results in a non-increasing sequence of loss values and converges[1] to a fixed point, usually a local minimum. Multiple random starts can be used. The full SCaDS algorithm is presented in Appendix 2.6.2 and its implementation in the statistical software R (R Core Team, 2020) is available from `https://www.github.com/trbKnl`.

2.2.6 Model selection

SCaDS runs with fixed values for the number of components, their status (whether they are common or distinctive), and the value of the lasso and ridge tuning parameters. Often these are unknown and model selection procedures are needed to guide users of the method in the selection of proper values.

In the component and regression analysis literature, several model selection tools have been proposed. The scree plot, for example, is a popular tool to decide upon the number of components (Jolliffe, 1986) but also cross-validation has been proposed (Smilde et al., 2004). Given a known number of components, Schouteden et al. (2013) proposed an exhaustive strategy that relies upon an ad hoc criterion to decide upon the status (common or distinctive) of the components. Finally, tuning of the lasso and ridge penalties is usually based on cross-validation (Hastie et al., 2009a).

Here, we propose to use the following sequential strategy. First, the number of components is decided upon using cross-validation, more specifically the

[1]Under mild conditions that hold in practice. An example where there is no convergence is starting from $\mathbf{W}_C = \mathbf{0}$.

Eigenvector method. In a comparison of several cross-validation methods for determining the number of components, this method came out as the best choice in terms of accuracy and low computational cost; see Bro et al. (2008). Briefly, this method leaves out one or several samples and predicts the scores for each variable in turn based on a model that was obtained from the retained samples: for one up to a large number of components the mean predicted residual sum of squares (MPRESS) is calculated and the model with the lowest MPRESS is retained. Second, a suitable common and distinctive structure for $\mathbf{W}_C$ is found using cross-validation: in this case, the MPRESS is calculated for all possible common and distinctive structures. Also in this case we propose to use the Eigenvector method detailed in Bro et al. (2008). In a third and final step, the lasso and ridge parameters λ_1 and λ_2 are tuned using the Eigenvector cross-validation method on a grid of values, chosen such that overly sparse and non-sparse solutions are avoided.

An alternative to the sequential strategy proposed here, is to use an exhaustive strategy in which all combinations of possible values for the components, their status, and λ_1 and λ_2 are assessed using cross-validation and retaining the solution with lowest MPRESS. However, there are known cases where sequential strategies outperform exhaustive strategies (Vervloet et al., 2016) and, furthermore, sequential strategies have a computational advantage as the number of models that needs to be compared is much larger in the exhaustive setting. This number is already large in the sequential setting because all possible common and distinctive structures are inspected, these are in total $\binom{(2^K-1)+Q-1}{Q}$ possible model structures[2]. For example, with $K = 2$ data blocks and $Q = 3$ components there are $\binom{(2^2-1)+3-1}{3} = 10$ possible common and distinctive structures to examine.

2.2.7 Related methods

The method introduced here, builds further on extensions of principal component analysis. These include sparse PCA (Zou et al., 2006), simultaneous components with rotation to common and distinctive components (Schouteden et al., 2013), and sparse simultaneous component analysis (Gu and Van Deun, 2016; Van Deun et al., 2011).

[2]The number of possible common and distinctive structures for a single component weight vector is equal to $2^K - 1$, because each of the K data block segments can either be constrained to be equal to zero or can be unconstrained; the case where the component is constrained to be equal to zero everywhere is not considered, hence the minus 1. For each of the Q components, one of these structures can be picked (with replacement meaning that two components can be of the same type, e.g., two common components) where the order of the components is of no importance.

Sparse PCA In practice, multi-block data are analyzed by treating them as a single block of variables. The problem of selecting the linked variables may then be addressed by using a sparse PCA technique. Zou et al. (2006) proposed a PCA method with a lasso and ridge penalty on the component weights. As previously discussed, this is a special case of the method we propose here (see equation (2.6)). The drawback of this approach is that it does not allow to control for dominant sources of variation.

SCA with rotation to common and distinctive components Schouteden et al. (2013) proposed a rotation technique for multi-block data that rotates the components resulting from the simultaneous component analysis toward common and distinctive components: A target matrix is defined for the loading matrix that contains blocks of zeros for the distinctive components (similar to the model structure in Equation 2.4 and remains undefined for the remaining parts). In general, the rotated loadings will not be exactly equal to zero and may even be large. To decide whether the components are indeed common or distinctive after rotation, Schouteden et al. (2013) propose to inspect the proportion of variance accounted for (%VAF) by the components in each of the blocks: A component is considered distinctive when the %VAF is considerably higher in the block(s) underlying the component than in the other blocks; it is considered common when the %VAF is approximately the same in all blocks. This introduces some vagueness in defining the common and distinctive components. Furthermore, no variable selection is performed. An often used strategy in the interpretation of the loadings is to neglect small loadings. This corresponds to treating them as zeros and performing variable selection. As shown by Cadima and Jolliffe (1995), this is a suboptimal selection strategy in the sense that they account for less variation than optimally selected variables. At this point, we would also like to point out that the definition in terms of %VAF is not useful when the zero constraints are imposed on the component weights as the %VAF by a distinctive component can still be considerable for the block that does not make up the component. This is because the %VAF is determined by the component scores and loadings with zero weights not implying (near) zero loadings.

Sparse SCA An extension of sparse PCA to the multi-block case was proposed by Van Deun et al. (2011). This approach allows for sparse estimation of the component weights using penalties that do not account for the multi-block structure like the ridge and lasso penalty but also using penalties that are structured at the level of the blocks like the group and elitist lasso (Yuan and Lin, 2006; Kowalski and Torrésani, 2009). The group lasso operates like the lasso at the block level, meaning that it sets whole blocks of coefficients equal to zero. The elitist lasso performs selection within each of the blocks, setting many but not all coefficients within each block equal to zero. Although sparse SCA allows for block-specific sparsity patterns, no distinction can be made between common and distinctive components because the penalties are defined at the level of the blocks (i.e., the same penalty for all components). Furthermore, the proposed algorithmic approach is not scalable to the setting of a (very) large number of variables: The procedure becomes slow and requires too much memory with a large number of variables.

SCA with penalized loadings Recently Gu and Van Deun (2016) developed an extension to sparse SCA by penalizing the loading matrix in a componentwise fashion, hence allowing for both common and distinctive components. The main distinguishing characteristic of this paper is that it penalizes the component weights and not the loadings. This raises the question whether this is very different, and if so, when to use penalized loadings and when to use penalized weights.

In regular unrotated PCA, loadings and weights are proportional or even exactly the same in approaches — such as the one taken here and by Zou et al. (2006) — that impose orthogonality on the matrix of weights or loadings (Smilde et al., 2004, p. 54). In case of penalties and sparsity constraints, however, loadings and weights take very different values and careful consideration should be given to their interpretation. Let us first consider the component weights. These are the regression weights in the calculation of the component scores and make the component scores directly observable. Sparseness of the component weights implies that the component scores are based on a selection of variables. An example where such a weight based approach may be most useful, is in the calculation of polygenic risk scores (Vassos et al., 2017). The loadings, on the other hand, measure the strength of association or correlation between the component and variable scores and give a more indirect or latent meaning to the components.

From an interpretational standpoint there is also an important difference between the component weights and the component loadings. As ten Berge (1986) and references therein point out, the component weights convey how the components depend on the variables, whereas the component loading matrix conveys the relationship between the component and the variables. The component loadings can only be interpreted if the meaning of the components are more or less understood (if the components are not understood you are inspecting the correlation between an observed item and something unknown, which is not insightful), in order to discover the meaning of the components, it necessary to inspect the component weights first. To conclude, when the aim is to automatically detect the linked variables throughout different data blocks in order to reveal common mechanisms at play (e.g., a risk score based on genetic as well as environmental risk), in a situation where the components are not yet understood, sparseness of the weights is warranted.

Besides these differences in interpretation, there are also other differences between a sparse loading and a sparse weight approach. These include differences in reconstruction error, with the reconstruction error of a sparse loading approach being much larger, and differences in the algorithmic approach with algorithms for sparse weights being computationally more intensive and less stable than algorithms for sparse loadings.

2.3 Empirical data examples

We will now provide two empirical data examples illustrating SCaDS. The purpose of these examples is twofold: one, to show how the analysis of linked data would go in practice when using SCaDS and two, to showcase the interpretational gain of common and distinctive components for multi-block data and of sparseness in general.

2.3.1 500 Family Study

For the first data example, we will make use of the 500 Family Study (Schneider and Waite). This study contains questionnaire data from family members of families in the United States and aims to explore how work affects the lives and well-being of the members of a family. From this study, we will use combined scores of different items from questionnaires collected for the father, mother, and child of a family. These scores are about the mutual relations between parents, between parents and their child, and items about how the child perceives itself; see Table 2.3 for an overview of the variable labels. In this example, the units of observation are the families, and the three data blocks are formed by the variables collected from the father, the mother and the child. The father and the mother block both contain eight variables while the child block contains seven variables. There are 195 families in this selection of the data.

In this section we will discuss the key steps in the analysis of linked data with SCaDS: pre-processing of the data, selecting the number of components, identifying the common and distinctive structure, the tuning of the ridge and lasso parameters, and the interpretation of the component weights.

Pre-processing of the data In this example, the linked data blocks have been scaled and centered, meaning that all variables have a variance of one and a mean of zero. This is common practice in PCA and SCA and has been done to give all variables equal weight in the analysis. The blocks have not been individually weighted because they contain (almost) exactly the same number of variables.

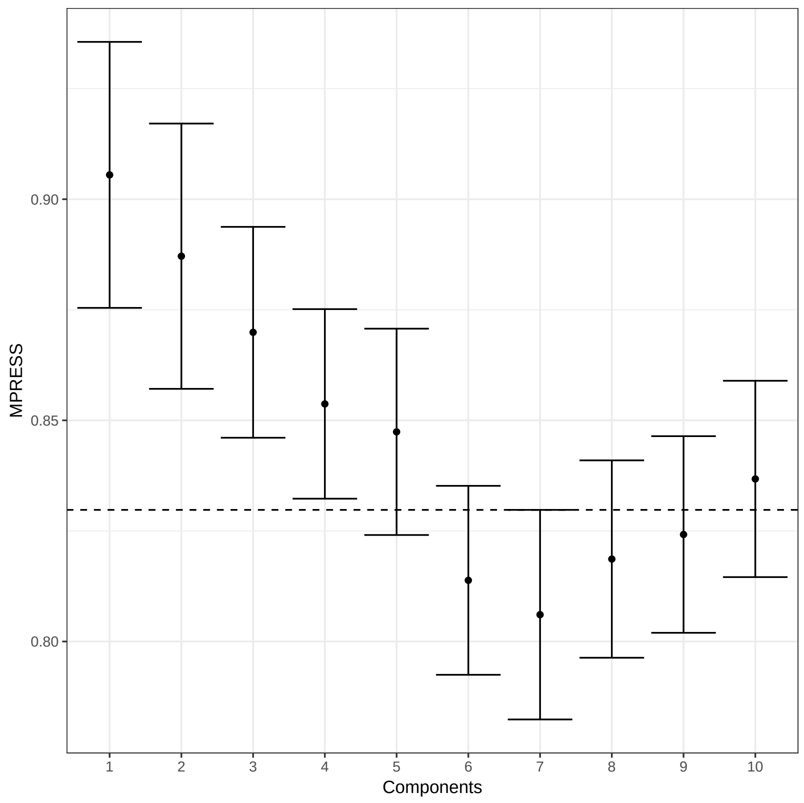


Figure 2.2. The MPRESS and standard error of the models with estimated with different number of components. The model estimated with seven components was the model with the lowest MPRESS, the model with six components was chosen for the final analysis

Selecting the number of components To find the number of components to retain, we made use of 10-fold cross-validation with the Eigenvector method. Figure 2.2 shows the MPRESS and the standard error of the MPRESS of the SC models with one up to ten components. The seven component solution is the solution with the lowest MPRESS; however, the solution with six components is within one standard error of the seven components solution. Relying on the one standard error rule, we will retain six components as this strikes a better balance between model fit and model complexity (Hastie et al., 2009a).

Identifying the common and distinctive structure To find the common and distinctive structure of the component weights that fits best to the data, we performed 10-fold cross-validation with the Eigenvector method. Hence, we have six components and three data blocks, so there are a total of $\binom{(2^3-1)+6-1}{6} = 924$ possible component weight structures to evaluate; the model with the lowest MPRESS was retained for further analysis; see Table 2.1. This is a model with one father-specific component (i.e., a component which is a linear combination of items from the father block only), one mother-specific component, one child-specific component, two parent (mother and father) components, and a common family component (a linear combination of items from all three blocks).

Table 2.1

The common and distinctive structure that resulted in the model with the lowest MPRESS out of the 924 possible models.

	$\mathbf{w}_1$	$\mathbf{w}_2$	$\mathbf{w}_3$	$\mathbf{w}_4$	$\mathbf{w}_5$	$\mathbf{w}_6$
F: Relationship with partners	1	0	1	1	0	1
F: Argue with partners	1	0	1	1	0	1
F: Childs bright future	1	0	1	1	0	1
F: Activities with children	1	0	1	1	0	1
F: Feeling about parenting	1	0	1	1	0	1
F: Communication with children	1	0	1	1	0	1
F: Argue with children	1	0	1	1	0	1
F: Confidence about oneself	1	0	1	1	0	1
M: Relationship with partners	0	1	1	1	0	1
M: Argue with partners	0	1	1	1	0	1
M: Childs bright future	0	1	1	1	0	1
M: Activities with children	0	1	1	1	0	1
M: Feeling about parenting	0	1	1	1	0	1
M: Communication with children	0	1	1	1	0	1
M: Argue with children	0	1	1	1	0	1
M: Confidence about oneself	0	1	1	1	0	1
C: Self confidence/esteem	0	0	0	0	1	1
C: Academic performance	0	0	0	0	1	1
C: Social life and extracurricular	0	0	0	0	1	1
C: Importance of friendship	0	0	0	0	1	1
C: Self Image	0	0	0	0	1	1
C: Happiness	0	0	0	0	1	1
C: Confidence about the future	0	0	0	0	1	1

Note. The items starting with an D, M or C belong to the father mother or child block. Zero indicates a constraint component weight constraint to zero and one indicates a non zero (free) component weight. The first component is a father component, the second component is a mother component, the third and the fourth are mother and father components, the fifth is a child component and the sixth is a common component

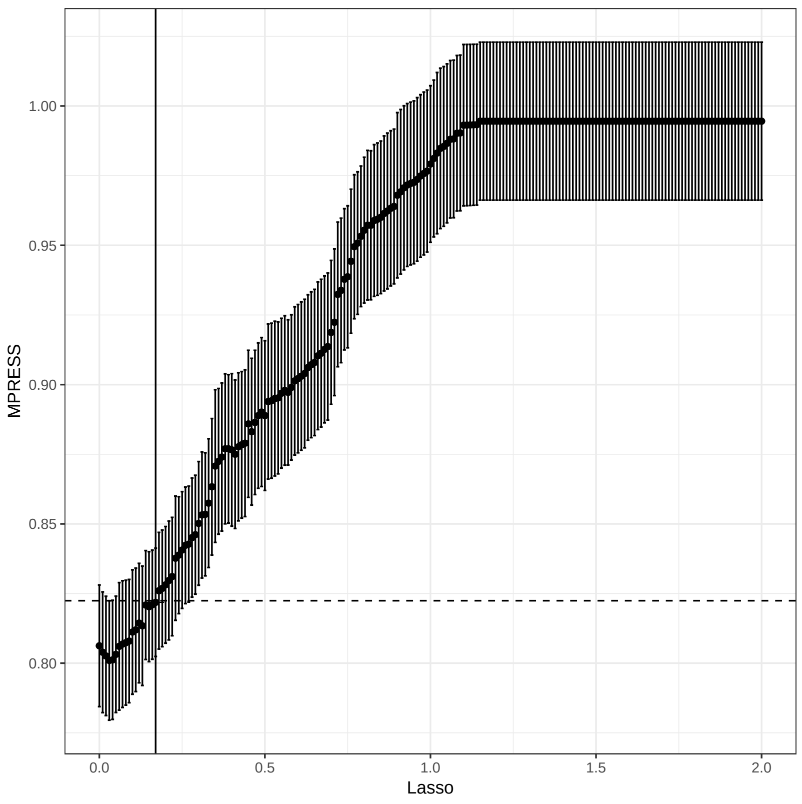


Figure 2.3. The MPRESS and standard error of the models for different values of the lasso parameter estimated with six components. The models below the dashed line are all models within one standard error of the model with the lowest MPRESS. The solid line indicates the lasso value that was picked for further analysis

Tuning of the ridge and lasso parameters To further increase the interpretability of the components, we will estimate the component weights with the common and distinctive component weight structure resulting from the previous step but including sparseness constraints on the weights. This requires choosing values for the ridge and lasso tuning parameters λ_1 and λ_2. In this example, the solution is identified because we have more variables than cases; therefore we do not need the ridge penalty term; thus the ridge penalty is set to 0. The optimal value for λ_1 was picked by performing 10-fold cross-validation by the Eigenvector method for a sequence of λ_1 values that results in going from no sparsity at all to very high sparsity in $\mathbf{W}_C$. The MPRESS and the standard error of the MPRESS of the models with the different values for the lasso parameter λ_1 can be seen in Figure 2.3; the one standard error rule was used to select the value for λ_1.

Interpretation of the component weights We subjected the data to a SCaDS analysis with six components, with zero constraints as in Table 2.1 and $\lambda_1 = 0.17$. The estimated component weights are displayed in Table 2.2. For comparison, we also included component weights resulting from SCA followed by Varimax rotation in Table 2.3, and SCA followed by thresholding of the weights after rotation to the common and distinctive structure in Table 2.1. We will discuss the component weights from SCaDS first, after which we will compare these results to the alternative methods.

The six columns in Table 2.2 show the component weights obtained with SCaDS. In total, these components account for 50.3% of the variance. As imposed, the first component is father-specific, the second mother-specific, the third is a parent component, the fifth is child-specific, and the sixth component is a common family component. The fourth component was constrained to be a parent component but, as a result of the lasso penalty, became a second mother-specific component with nonzero loadings only from variables belonging to the mother block. Interestingly, the shared parent component is formed by the variables “activities with children”, "communication with children" of the father block, and “activities with children” of the mother block. The variable descriptions tell us that this component could be a parentchild involvement indicator. Large component weights for the common component are: “child’s bright future” in the mother and father block, and “self-confidence/esteem” and “academic performance” in the child block. This component indicates that a child’s self-confidence and academic performance is associated with both parents believing in a bright future for their child.

For comparison we included in Table 2.3 the component weights of the six components obtained using SCA with Varimax rotation, this is an unconstrained analysis with maximal VAF. In total, the six components explain 55.2% of the variance in the data; this is a bit more than the 50.3% obtained with SCaDS. Even this example with rather few variables is not straightforward to interpret because all variables contribute to each of the component. In this case, a more fair comparison is to rotate the component weights resulting from the SCA to the common and distinctive structure displayed in Table 2.1 and to threshold the small (in absolute value) coefficients as is often done in practice. We thresholded such that the same number of zero coefficients was obtained for each component as for SCaDS. The results of this analysis can be seen in Table 2.4. The first thing that strikes is that the variance accounted for drops to 41.9%. This confirms the observation made by (Cadima and Jolliffe, 1995) that the practice of thresholding is a flawed way to perform variable selection when the aim is to maximize the VAF. Also the meaning of the components changed, although the main patterns found in SCaDS can still be observed.

Concluding, these results illustrate well that identifying the common and distinctive structure in multi-block data eases the interpretation substantially, while

Table 2.2

Component weights for the family data as obtained with SCaDS

	$\mathbf{w}_1$	$\mathbf{w}_2$	$\mathbf{w}_3$	$\mathbf{w}_4$	$\mathbf{w}_5$	$\mathbf{w}_6$
F: Relationship with partners	0	0	0	0	0	0
F: Argue with partners	-0.57	0	0	0	0	0
F: Childs bright future	0	0	0	0	0	0.56
F: Activities with children	0	0	0.61	0	0	0
F: Feeling about parenting	-0.12	0	0	0	0	0
F: Communication with children	0	0	0.39	0	0	0
F: Argue with children	-0.45	0	0	0	0	0
F: Confidence about oneself	-0.45	0	0	0	0	0
M: Relationship with partners	0	1.00	0	0	0	0
M: Argue with partners	0	0	0	-0.31	0	0
M: Childs bright future	0	0	0	0	0	0.53
M: Activities with children	0	0	0.42	0	0	0
M: Feeling about parenting	0	0	0	-0.26	0	0.04
M: Communication with children	0	0	0	-0.44	0	0
M: Argue with children	0	0	0	-0.61	0	0
M: Confidence about oneself	0	0.26	0	-0.18	0	0
C: Self confidence/esteem	0	0	0	0	-0.27	0.13
C: Academic performance	0	0	0	0	0	0.36
C: Social life and extracurricular	0	0	0	0	0	0.00
C: Importance of friendship	0	0	0	0	-0.41	0
C: Self Image	0	0	0	0	-0.56	0
C: Happiness	0	0	0	0	-0.45	0
C: Confidence about the future	0	0	0	0	-0.15	0.06
%VAF: per component	0.08	0.07	0.07	0.09	0.10	0.09
%VAF: total			50.3			

Note. The items starting with an F, M or C belong to the father mother or child block

still retaining a high variance accounted for.

An advantage of interpreting the component weights directly is that the researcher exactly knows the composition of the component. In some cases, the components themselves are used in subsequent analysis for example as predictors in a regression model. For the interpretation of that model, it is certainly useful to have a good grasp on what the predictors represent. Another advantage is that if the weights already have been estimated, then computing new component scores for new units of observation is straightforward. Because these component weights are sparse, only the items with nonzero component weights have to be measured to predict the component score of a new observed unit. This could greatly reduce the costs of predicting component scores for newly observed units.

Table 2.3

Component weights for the family data resulting from SCA with Varimax rotation

	$\mathbf{w}_1$	$\mathbf{w}_2$	$\mathbf{w}_3$	$\mathbf{w}_4$	$\mathbf{w}_5$	$\mathbf{w}_6$
F: Relationship with partners	0.05	0.57	-0.02	0.03	-0.03	-0.09
F: Argue with partners	0.04	0.15	-0.03	-0.06	0.05	-0.47
F: Childs bright future	-0.06	-0.08	0.15	0.47	0.01	-0.20
F: Activities with children	0.10	-0.03	0.04	-0.08	-0.63	-0.08
F: Feeling about parenting	-0.06	-0.15	0.06	0.06	-0.12	-0.40
F: Communication with children	-0.01	-0.01	-0.08	0.05	-0.49	-0.07
F: Argue with children	-0.11	-0.11	-0.06	-0.04	0.04	-0.53
F: Confidence about oneself	0.15	0.22	0.03	0.07	-0.08	-0.43
M: Relationship with partners	-0.07	0.60	0.06	0.01	0.06	0.03
M: Argue with partners	-0.27	0.16	-0.04	-0.26	0.06	-0.14
M: Childs bright future	-0.38	-0.02	0.18	0.37	0.06	0.03
M: Activities with children	-0.27	-0.01	0.09	-0.10	-0.44	0.13
M: Feeling about parenting	-0.37	0.06	0.03	0.10	-0.01	-0.03
M: Communication with children	-0.42	-0.05	-0.03	-0.02	-0.16	0.05
M: Argue with children	-0.39	-0.14	-0.07	-0.15	0.17	-0.14
M: Confidence about oneself	-0.35	0.31	-0.07	-0.08	0.01	0.12
C: Self confidence/esteem	-0.18	-0.10	-0.31	0.23	0.01	-0.01
C: Academic performance	-0.02	-0.03	-0.12	0.42	0.11	-0.04
C: Social life and extracurricular	0.08	0.12	0.01	0.37	-0.03	0.09
C: Importance of friendship	0.11	0.06	-0.37	0.23	-0.05	0.07
C: Self Image	-0.04	-0.02	-0.56	-0.07	0.01	-0.01
C: Happiness	0.02	-0.01	-0.55	-0.11	0.01	-0.04
C: Confidence about the future	-0.01	0.13	-0.19	0.27	-0.24	0.07
Variance Accounted For (%)			55.2			

Note. The items starting with an F, M or C belong to the father mother or child block

2.3.2 Alzheimer study

For the second data example we will use the Alzheimer's Disease Neuroimaging Initiative (ADNI) data[3]. The purpose of the ADNI study is "to validate biomarkers for use in Alzheimers disease clinical treatment trials" (Alzheimers Disease Neuroimaging Initiative, 2017).

The ADNI data is a collection of datasets from which we selected a dataset with items measuring neuropsychological constructs, and a dataset with gene expression data for genes related to Alzheimers disease. The neuropsychological data block consists of 12 variables containing items from a clinical dementia scale assessed by a professional and from a self-assessment scale relating to everydays cognition. The gene data block contains 388 genes. For a group of 175 participants, complete data for both the genetic and the neuropsychological variables is available. This is an example of a high-dimensional dataset where the number of variables exceeds the number of cases.

In this specific case, it would be interesting to see whether there is an association between particular Alzheimer-related genes and items from the clinical scales or whether the two types of data measure different sources of variation.

Pre-processing of the data As in the previous example, the linked data blocks have been scaled and centered. Furthermore, as one block is much larger than the other, the blocks have been scaled to equal sum of squares by dividing each block by the square root of the number of variables in that block. In this way, the larger block does not dominate the analyses (see Van Deun et al. (2009), for a discussion of different weighting strategies).

Selecting the number of components The number of components has been selected making use of 10-fold cross-validation with the Eigenvector method. This resulted in a four-component solution (see Figure 2.4).

Tuning of the ridge parameter This linked dataset contains more variables than cases, therefore we included a ridge penalty (this is $\lambda_2 \neq 0$) to make the solution stable. To tune the value of the ridge paramter, we performed 10-fold cross-validation with the Eigenvector method on a sequence of values. The resulting

[3]The ADNI was launched in 2003 as a public-private partnership, led by Principal Investigator Michael W. Weiner, MD. The primary goal of ADNI has been to test whether serial magnetic resonance imaging (MRI), positron emission tomography (PET), other biological markers, and clinical and neuropsychologi- cal assessment can be combined to measure the progression of mild cognitive impairment (MCI) and early Alzheimers disease (AD). For up-to-date information, see www.adni-info.org.

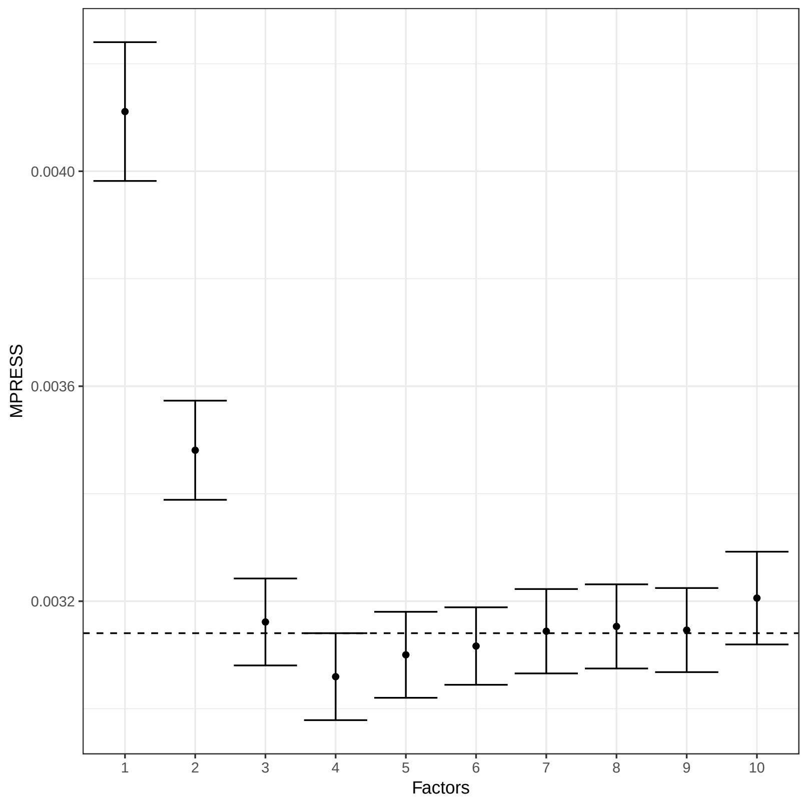


Figure 2.4. The MPRESS and standard error of the models with estimated with different number of components. The model estimated with four components was the model with the lowest MPRESS

MPRESS statistics and standard errors thereof are shown in Figure 2.5. The value within one standard error of the lowest MPRESS was retained for further analyses.

Identifying the common and distinctive structure To find the common and distinctive structure of the component weights which fits best to the data, we performed 10-fold cross-validation with the EigenVector method on all possible structures. In this example we have four components and two data blocks, so there are a total of $\binom{(2^2-1)+4-1}{4} = 15$ possible component weight structures to evaluate. After cross-validation we found the model with the lowest MPRESS to be a model with four distinctive components: two for each block; see Figure 2.6 for the MPRESS and standard error of the MPRESS of all the 15 models.

Tuning of the lasso parameters A final step in selecting a suitable model for the ADNI data, is the tuning of the lasso parameter to obtain sparsity in the component weights beyond the zeroes resulting from the imposed common and distinctive structure. The value of the lasso parameter was determined with 10-fold cross validation (EigenVector method). The MPRESS of the models for different values of the lasso parameter can be seen in Figure 2.7; the largest value of λ_1 within one

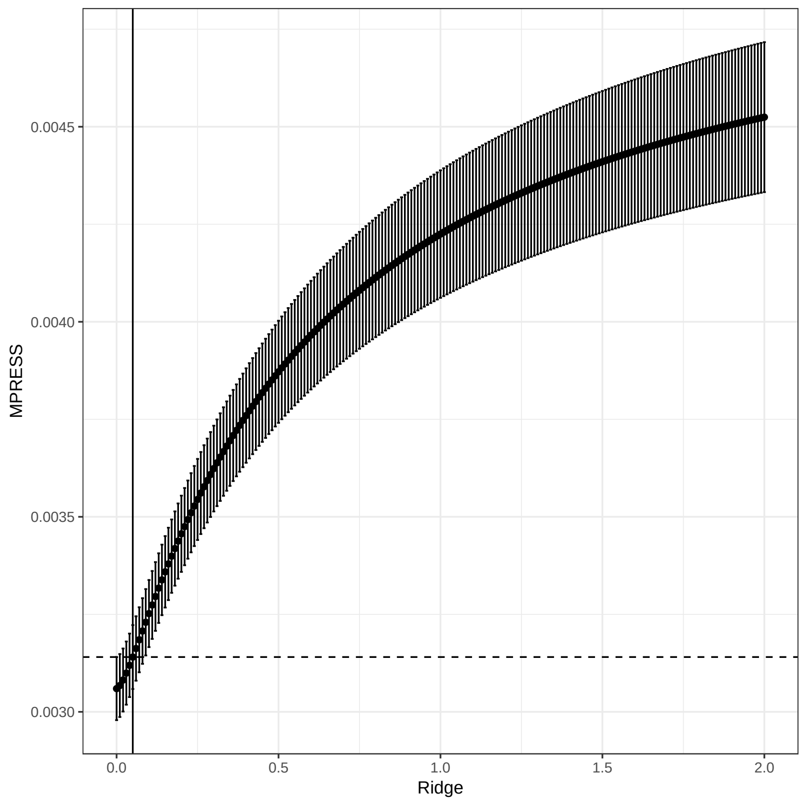


Figure 2.5. The MPRESS and standard error of the models estimated with four components for different values of the ridge parameter. The models below the dashed line are all models within one standard error of the model with the lowest MPRESS. The solid line indicates the ridge value that was picked for further analysis

standard error of the lowest MPRESS was retained for the final SCaDS analysis.

Interpretation of the component weights The component weights of the final analysis with the chosen meta-parameters are summarized in a heat plot in Figure 2.8. The first two components contain only items from the gene expression block, and the third and the fourth component only contain items from the neuropsychological data block. Notably, the third component mainly contains items of the self-assessment scale while the fourth component mainly contains items of the dementia scale assessed by the clinician.

Concluding, this particular example shows that SCaDS can also be applied in the setting of (many) more variables than observation units. Whether the obtained results also make sense from a neuropsychological perspective needs further investigation.

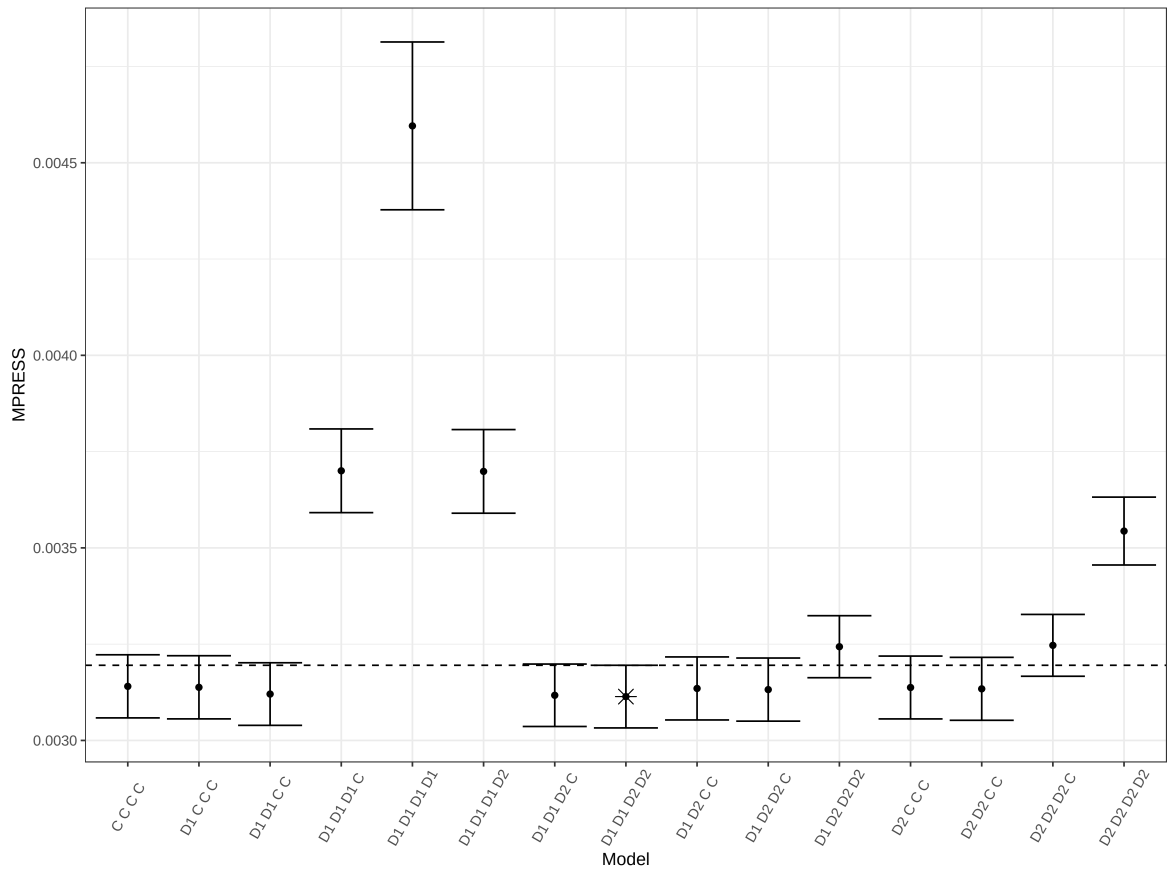


Figure 2.6. The MPRESS and standard error of all 15 models with different common and distinctive structures of the linked data set from the ADNI study. Model "D1 D1 D2 D2" is the model with the lowest MPRESS. D1 denotes a distinctive component for the first block, D2 denotes a distinctive component for the second block, and C denotes a common component

2.4 Simulation studies

We tested the performance of SCaDS in finding back a sparse common and distinctive model structure in a controlled setting using simulated data. First of all, we were interested to see whether accounting for the presence of block-specific components in $\mathbf{W}_C$ would result in improved estimates compared to a sparse PCA analysis of the concatenated data. If there is no improvement of the estimated weights by SCaDS over sparse PCA, sparse PCA can be used for the analysis of multiblock data and there is no need for SCaDS. Second, we tested the performance of the cross-validation method in finding back the right common-distinctive structure given the correct number of components.

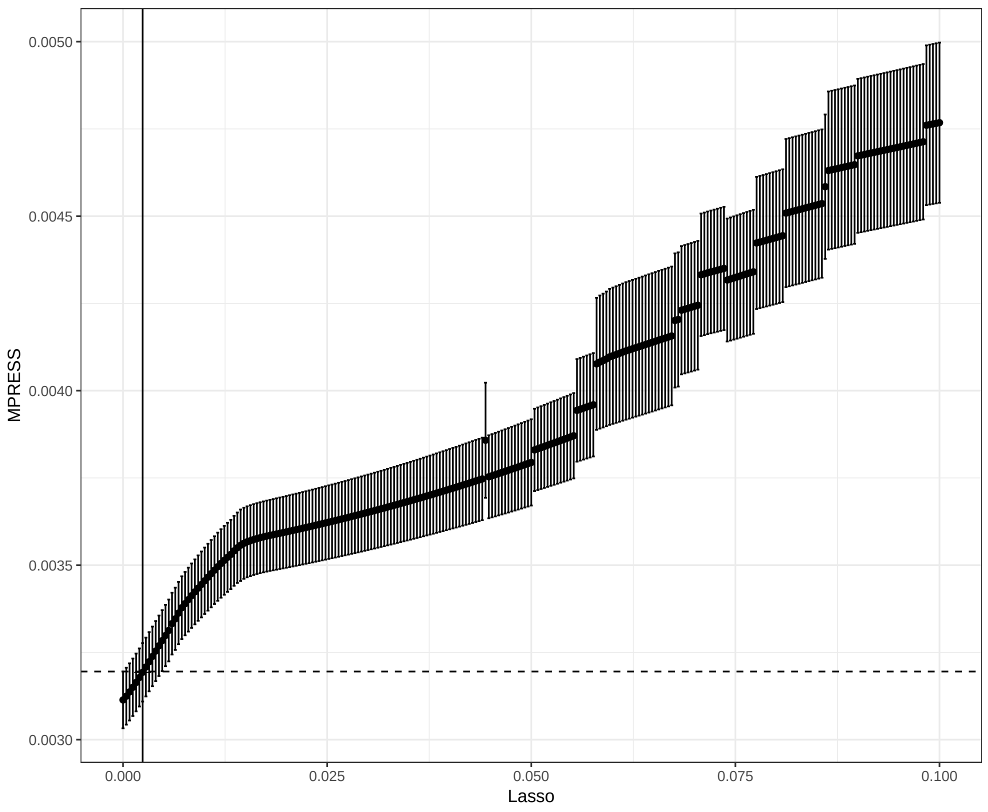


Figure 2.7. The MPRESS and standard error of the models for different values of the lasso parameter, with four components and the picked ridge tuning parameter. The models below the dashed line are all models within one standard error of the model with the lowest MPRESS. The solid line indicates the lasso value that was picked for further analysis

2.4.1 Recovery of the model parameters under the correct model

The data in the first simulation study were generated under a sparse SCA model with two data blocks and three components, of which one component is common and two are distinctive (one distinctive for each data block; see Equation (2.5) for such a model structure). The size of the two data blocks was fixed to 100 rows (subjects) and 250 columns (variables) per block.

We generated data under six conditions, resulting from a fully crossed design determined by two factors. A first factor was the amount of noise in the generated data with three levels: 5%, 25%, and 50% of the total variation. The second factor was the amount of sparsity in $\mathbf{W}_C$ with two levels: a high amount of sparsity (60 % in all three components) and almost no sparsity (2 % in the common component and 52 % in the distinctive components) in the component weight matrix $\mathbf{W}_C$. In each condition 20 data sets were generated. We refer the reader to Appendix 2.6.1 for the details on the procedure we used to generate data with the desired model

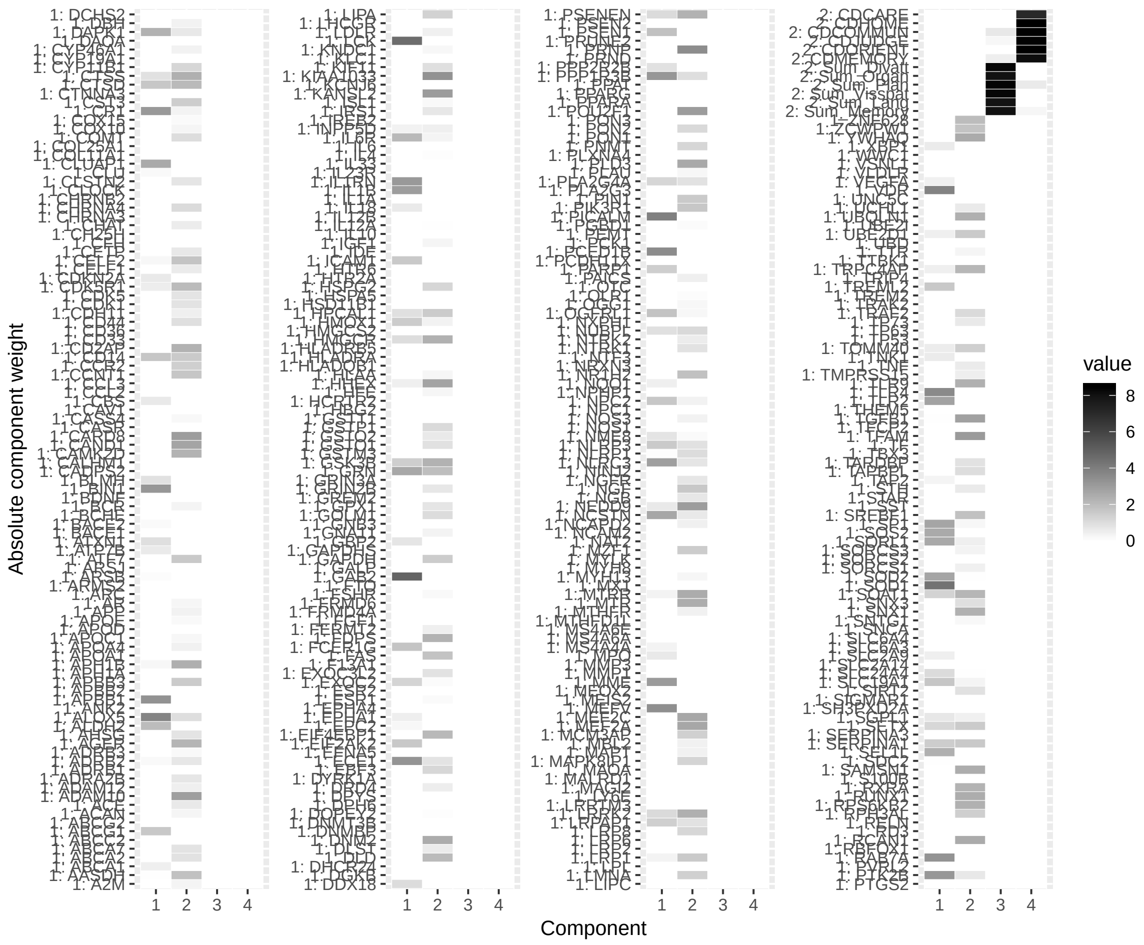


Figure 2.8. A heat plot of the absolute values of the component weights table of the final analysis for the ADNI data example. The variable names with prefix 1 denotes variables belonging to gene expression block, names with prefix 2 denotes variables belonging to the neuropsychological block. The table has been broken row wise into four pieces to fit the page.

structure.

All data sets were analyzed using both the SCaDS method introduced here and the sparse PCA analysis introduced by Zou et al. (2006) and implemented in the elasticnet R package (Zou and Hastie, 2018). SCaDS was applied with correct input for the zero-block constraints on the component weight matrix, this is with input of the common and distinctive structure that underlies the data. Sparse PCA was applied with input of the correct number of zero component weights in $\mathbf{W}_C$ and this for each component (sparse PCA can be tuned to yield exactly a given number of zero coefficients because it relies on a LARS estimation procedure (Tibshirani et al., 2004)). Using sparse PCA with the correct number of zero component weights is equal to supplying the analysis with a perfectly tuned lasso parameter. In order to achieve a perfectly tuned lasso parameter for SCaDS, we used an iterative scheme based on the bisection method for root finding. The method boils down to estimating the model with a certain lasso value, after which depending on the number of non zero weights in $\widehat{\mathbf{W}}_C$ compared the the number of non zero weights in $\mathbf{W}_C$, the lasso is increased or decreased. This process is repeated until the number of non zero component weights in $\widehat{\mathbf{W}}_C$ is within 0.01% of the number of non zero component weights in $\mathbf{W}_C$. The ridge parameter λ_2 was tuned for one particular data set in each of the six conditions with cross validation and picked according to the one standard error rule. (The ridge was not tuned for each individual data set because of computational constraints.)

In order to quantify how well the component weight matrix $\mathbf{W}_C$ can be recovered by SCaDS and sparse PCA of the concatenated data, we calculated Tucker's coefficient of congruence between the model structure $\mathbf{W}_C$ and its estimate $\widehat{\mathbf{W}}_C$ as resulting from SCaDS and sparse PCA. Tucker's coefficient of congruence (Lorenzo-Seva and ten Berge, 2006) is a standardized measure of proportionality between two vectors, calculated as the cosine of the angle between two vectors. Note that $\mathbf{W}_C$ and $\widehat{\mathbf{W}}_C$ are vectorized first before they are compared. A Tucker congruence coefficient in the range from 0.85 to 0.95 corresponds to fair similarity between vectors, while a Tucker congruence coefficient of > 0.95 correspond to near equal vectors (Lorenzo-Seva and ten Berge, 2006). Furthermore, we also calculated the percentage of correctly as (non-)zero classified component weights.

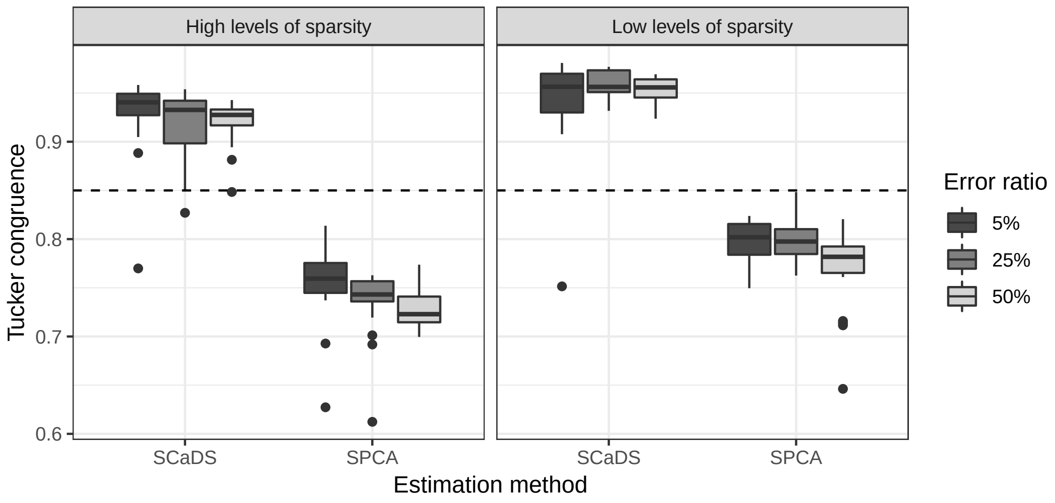


Figure 2.9. Tucker congruence coefficients between $\mathbf{W}_C$ and $\widehat{\mathbf{W}}_C$, where $I = 100$ and $J = 500$. Each condition is based on 20 replications, the dashed line indicates a Tucker congruence coefficient of 0.85

Box plots of Tucker's coefficient of congruence between $\mathbf{W}_C$ and $\widehat{\mathbf{W}}_C$ are shown in Figure 2.9, both for the estimates obtained with our SCaDS method and with sparse PCA. The two panels correspond to the two levels of sparseness; within panels the box plots differ with respect to the method used to estimate the weight matrix and the noise level. In all conditions, SCaDS has on average higher congruence than sparse PCA. This indicates that controlling for block specific sources of variation results in a better recovery of the model coefficients (given the correct model). Furthermore, the bulk of Tucker congruence coefficients obtained when using SCaDS are above the threshold value of 0.85 thus indicating fair similarity of the estimated component weights to the model component weights. Sparse PCA, on the other hand, has almost all solutions below the 0.85 threshold. The manipulated noise and sparseness factors had some influence on the size of Tucker's congruence. First, as one may expect, congruence decreased with an increasing level of noise. Second, comparing the left panel (high level of sparsity) to the right panel (low level of sparsity), Tucker congruence was higher for the low level of sparsity.

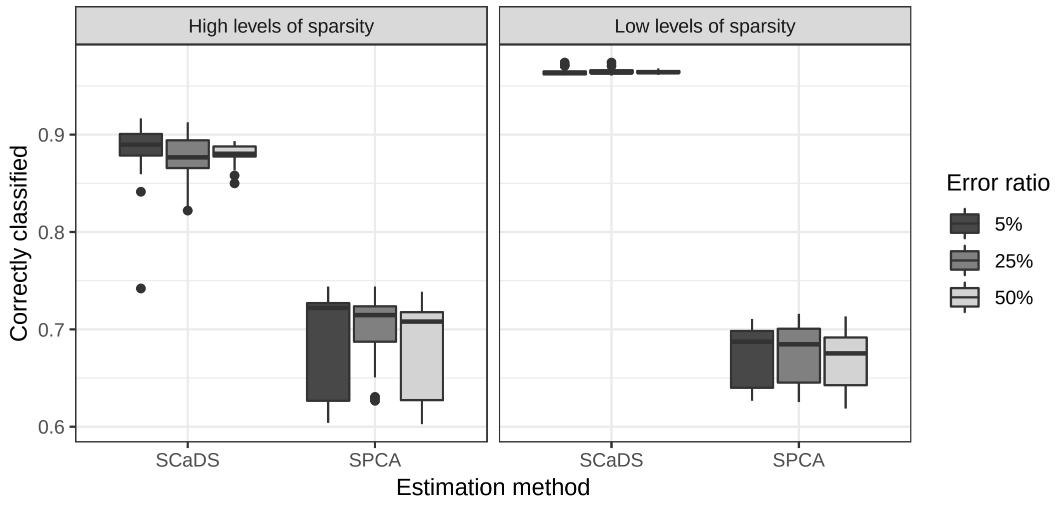


Figure 2.10. Percentage of correctly classified zero and non zero weights between $\mathbf{W}_C$ and $\widehat{\mathbf{W}}_C$, where $I = 100$ and $J = 500$. Each condition is based on 20 replications.

The box plots in Figure 2.10 show the percentage of correctly classified component weights for both estimation procedures in each of the six conditions. An estimated component weight is counted as correctly classified if it has non zero status in $\mathbf{W}_C$ as well as in $\widehat{\mathbf{W}}_C$ or if it has zero status in $\mathbf{W}_C$ as well as in $\widehat{\mathbf{W}}_C$. Not surprisingly, SCaDS does far better compared to sparse PCA, this because SCaDS makes use of true underlying structure of the data. More importantly, these results show that if the data do actually contain an underlying multi-block structure, sparse PCA is not able to find this structure by default, too much weights are incorrectly classified. For good recovery of the component weights it necessary to take the correct block structure into account.

Concluding, this simulation study shows that a multi-block structure is not picked up by sparse PCA by default. Furthermore, the simulation results shows that to have satisfactory component weights estimates the correct multi-block structure needs to be taken into account. In practice, the underlying multi-block structure of the data is unknown. Hence, model selection tools that can recover the correct model are needed.

2.4.2 Finding the underlying common and distinctive structure of the data

In the previous section, we concluded that in order to have good estimation, the correct underlying multi-block structure needs to be known. In this section, we will explore to what extent 10-fold cross-validation with the Eigenvector method can be used to identify the correct underlying block structure of the data, assuming the number of components is known. We will consider both a high- and a low-dimensional setting.

In the high-dimensional setting, data were generated under the same conditions as the previous simulation study but analyzed without input of the correct common-distinctive model structure. Instead, for each of the generated data sets, we calculated the MPRESS and its standard error for all possible combinations of common and distinctive components, this is 10 possible models for each generated dataset (2 data blocks 3 combinations). The models are estimated without a lasso penalty (this is $\lambda_1 = 0$), and with the same value for the ridge parameter as in the previous simulation study.

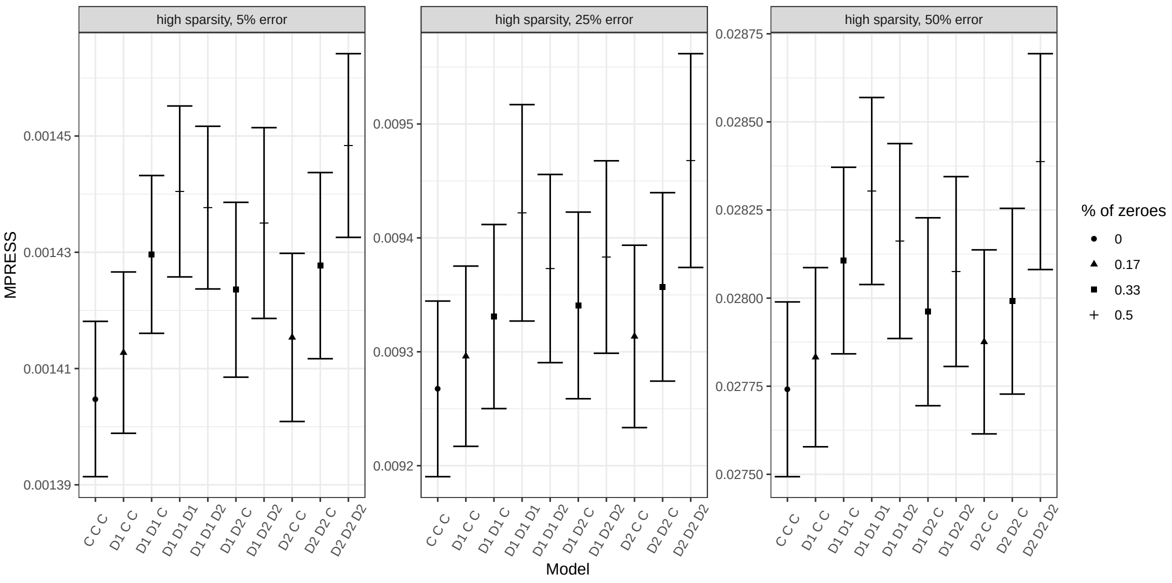


Figure 2.11. The MPRESS and standard error of the MPRESS of all common and distinctive structures in three conditions. % of zeroes refers to the amount of zeroes in the common and distinctive structure. "D1 D2 C" is the data generating structure. D1 denotes a distinctive component for the first block, D2 denotes a distinctive component for the second block, and C denotes a common component

We illustrate the results obtained for the first three generated data sets in the high sparsity condition in Figure 2.11. The correct model used to generate the data is the model labeled “D1 D2 C” (representing a model with one distinctive component for each block and one common component). The plots show that the most complex model (this is the unconstrained “C C C” model) always has the lowest MPRESS. Furthermore, the plots show that model fit decreases for models with more imposed zeroes. This means that 10-fold cross-validation with the Eigenvector method favors models that are overfitted (i.e. models with too many non-zero coefficients). To remedy this situation, the one standard error rule has been proposed (Hastie et al., 2009a). Here, this means that the model with the lowest complexity (or, the highest number of zeroes) is chosen that still falls within one standard error of the model with the lowest MPRESS; if this is more than one model, the model with lowest MPRESS is chosen. The results in 2.11 suggest that this may lead to the correct model in a number of cases (the two panels at the right).

The results of the full simulation study are summarized in Table 2.5. The column labeled “Best model” shows the proportion of cases where the true model was selected based on choosing the model with lowest MPRESS. This strategy never results in selecting the correct model. Upon closer inspection of the results (e.g., Figure 2.11) the model with lowest MPRESS often was the unconstrained model. Whether the correct model would be selected when applying the one standard error rule (this is, choosing the model with the highest MPRESS but within one standard error of the model with the lowest MPRESS) can be seen in the column labeled “One Std Error rule”. Unfortunately this does not seem to be the case very often, in only about 10% of the cases the correct model was chosen based on this heuristic. Clearly, cross validation as a method for selecting the true common-distinctive model structure does not work in the high-dimensional setting.

We also included results of a second simulation study to see how 10-fold cross-validation would perform in the low dimensional case: data were generated as previously but with 195 cases and 20 variables. Figure 2.12 includes results of three specific generated data sets. Still cross validation based on selecting the model with the lowest MPRESS is biased towards more complex models with fewer zero constraints. However, using the one standard error rule, often the correct model is selected. A full summary of the results can be seen in Table 2.6.

Concluding, 10-fold cross validation with the Eigenvector method and using the one standard error rule does seem to work for selecting the correct common-distinctive model structure in the low-dimensional setting. However, in the high-dimensional setting overly complex models are chosen even when using the one

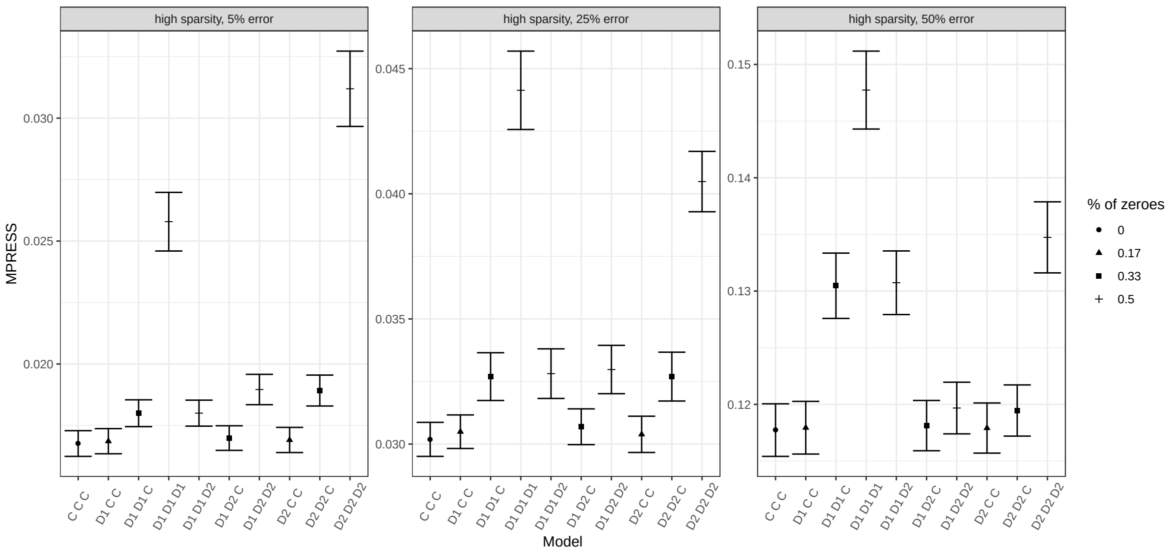


Figure 2.12. The MPRESS and standard error of the MPRESS of all common and distinctive structures in three conditions. % of zeroes refers to the amount of zeroes in the common and distinctive structure. "D1 D2 C" is the data generating structure. D1 denotes a distinctive component for the first block, D2 denotes a distinctive component for the second block, and C denotes a common component

standard error rule. Clearly other model selection tools have to be tested, also taking into account that here only one model selection step was isolated. Although we suggested a sequential strategy to reduce the computational burden, simultaneous strategies may be needed in order to find the correct model.

2.5 Discussion

In this era of big data, researchers in psychology often have novel types of data available to supplement the more traditional types of data they are accustomed to. This opens the avenue to a more informed understanding of the human behavior system; the different types of data usually probe different components of the behavioral system and by integrating them a more complete view is obtained. To get this deeper understanding that goes beyond a fragmented view, it is crucial that the following questions can be answered: How do the components of the human behavior system interact and what do they contribute independently from the other components? As we argued in this paper, this means disentangling joint sources of variation from specific sources of variation present in the data. A further complicating factor in the analysis of linked traditional and novel data resides in the often untargeted collection of the novel data: This not only leads to a very large number of variables but also to the collection of variables that may or may not be relevant for the problem under study. On the side of data analysis, this requires methods that are computationally efficient and capable of automated variable selection. To address these issues, we introduced SCaDS, a novel variable selection method that is suitable to detect the common and specific mechanisms at play. In this paper, we proposed the SCaDS model, a procedure to estimate the model parameters and an implementation of the algorithm in the freely available statistical software R. Importantly, the proposed implementation of SCaDS can handle a large number of variables including cases where the total number of variables exceeds the number of observations.

We illustrated how to use SCaDS using publicly available data from the 500 Family Study (Schneider and Waite) using a block of data for father, for mother, and for their child. The interpretational advantage of using a sparse common and distinctive model structure was clearly shown. Furthermore, support for the superior performance of SCaDS compared to sparse PCA of the concatenated data in estimating back the model parameters was convincingly shown in a simulation study. Especially in situations where the number of variables was large compared to the number of observation units, SCaDS outperformed the approach of applying sparse methods for a single block of data while ignoring the multi-block structure.

In this paper we used cross-validation as a tool to determine the meta-parameters of the SCaDS method, namely the number of common and distinctive components and the level of sparsity. For data generated in the low-dimensional setting satisfactory results were obtained; yet, in the high-dimensional setting we observed a bias towards overly complex models. More research needs to be done — including the use of simulation studies — to investigate if cross-validation indeed recovers the correct number of common and distinctive components and the degree of sparseness. Other alternatives that have been proposed in the literature need to be explored as well, including the convex hull method (Wilderjans et al., 2012; Timmerman et al., 2016). Of particular interest are model selection tools that are less computationally intensive than cross-validation like the index of sparseness (Trendafilov, 2013).

To conclude, SCaDS is a promising method for the analysis of multi-block data that yields insightful representations of linked data: Intricate relations between very different sources of information on human behavior are revealed, even in presence of irrelevant variables. Here, the methodology was introduced and show-cased on data with a modest number of variables. The implementation proposed here is scalable to the high-dimensional case of very large sets of variables but more work is needed to study the performance of SCaDS in such settings using both simulated and empirical data.

2.6 Appendix

2.6.1 Specifics of the simulation study

As a starting point for the generation of data in the simulation study, initial data matrices were generated according to $\mathbf{x}_i^{(1)} \sim \mathcal{MVN}(\mathbf{0}, 3\mathbf{I})$, where $\mathbf{x}_i^{(1)} \in \mathbb{R}^{500}$ for $i = 1 \ldots 100$ subjects. The variables in the resulting dataset $\mathbf{X}^{(1)}$ were standardized to have zero mean and unit variance. Let $\mathbf{UDV}^T$ be the singular value decomposition of the standardized matrix $\mathbf{X}^{(2)}$. Then, the standard PCA decomposition can be obtained by setting the loading matrix $\mathbf{P}^{(1)}$ equal to the first three columns of $\mathbf{V}$ and setting $\mathbf{W}^{(1)}$ also equal to the first three columns of $\mathbf{V}$ (note that the basis PCA model with orthogonality of the loadings indeed has equal weights and loadings). As a next step, we imposed the common and distinctive sparse structure on $\mathbf{W}^{(1)}$ as follows: First, a non-sparse common and distinctive structure was obtained by setting — for the distinctive components — all those component weights that correspond to the variables of a particular block equal to zero (see Equation (2.4)); next, sparseness of the common component and of the non-zero parts of the distinctive component was imposed by setting all coefficients with an absolute value lower than some threshold to zero. The threshold was taken such that the level of sparseness defined by the condition was attained. The resulting matrix of component weights $\mathbf{W}$ is the model structure. Subsequently, the model structure for $\mathbf{P}$ was obtained by setting the loadings equal to solution of the least squares problem $\arg\min_{\mathbf{P}} \|\mathbf{X}^{(2)} - \mathbf{X}^{(2)}\mathbf{W}\mathbf{P}^{(1)}\|^2$ conditional on $\mathbf{W}$, see Appendix 2.6.2. Finally, the data $\mathbf{X}^*$ that were used as the input to the SCaDS and sparse PCA analyses were obtained by adding noise,

$$\mathbf{X}^* = \mathbf{X}^{(2)} + c\mathbf{E}, \tag{2.7}$$

where $\mathbf{E}$ is a random error matrix where the rows are generated from $\mathcal{MVN}(\mathbf{0}, \mathbf{I})$, and where c is a scalar that controls the signal to noise ratio (SNR) in $\mathbf{X}^*$. The SNR is calculated as follows.

$$\text{SNR} = 1 - \frac{\widehat{\text{Var}}(\mathbf{X}^{(2)})}{\widehat{\text{Var}}(\mathbf{X}^*)}. \tag{2.8}$$

To obtain the scalar c to get the desired SNR, substitute the SNR (in the simulation study these 0.05, 0.25, 0.5) into Equation 2.8 and solve for c. No multiple starts were used in the simulation study, the algorithm was started with a "warm" start by initializing $\mathbf{W}$ with first three columns of $\mathbf{V}$ from $\mathbf{X}^*= \mathbf{UDV}^T$.

2.6.2 Description of algorithm

In order to obtain a solution for $\mathbf{W}_C$ with SCaDS the following objective function needs to be minimized given selected values for the tuning parameters: λ_2, λ_1, the number of components and a common distinctive component weight structure,

$$\begin{aligned} \underset{\mathbf{W}_C, \mathbf{P}_C}{\arg\min} L(\mathbf{W}_C, \mathbf{P}_C) &= \|\mathbf{X}_C - \mathbf{X}_C \mathbf{W}_C \mathbf{P}_C^T\|_2^2 + \lambda_1 \|\mathbf{W}_C\|_1 + \lambda_2 \|\mathbf{W}_C\|_2^2, \\ \text{s.t. } \mathbf{P}_C^T \mathbf{P}_C &= \mathbf{I}, \lambda_2, \lambda_1 \geq 0 \text{ zero block constraints on } \mathbf{W}_C, \end{aligned} \tag{2.9}$$

where $\mathbf{X}_C \in \mathbb{R}^{I \times \sum_k J_k}$, $\mathbf{W}_C \in \mathbb{R}^{\sum_k J_k \times Q}$, $\mathbf{P}_C \in \mathbb{R}^{\sum_k J_k \times Q}$ and I, J_k and Q denote the number of cases, variables in block k and components respectively. For ease of notation we will drop the subscript C and let $\sum_k J_k = L$ where $l = 1 \dots L$.

A solution to (2.9) can be obtained using an alternating least squares approach. Meaning, estimates for $\mathbf{P}$ can be obtained by minimizing (2.9) conditional on $\mathbf{W}$ and vice versa. The alternating least squares minimization is repeated until the convergence criterion has been reached. First, we will discuss the optimizing equation (2.9) with respect to $\mathbf{P}$ conditional on $\mathbf{W}$, then we will discuss the optimization of $\mathbf{W}$ conditional on $\mathbf{P}$.

The loading matrix $\mathbf{P}$ which minimizes equation (2.9) conditional on $\mathbf{W}$ has an analytic solution given $\mathbf{P}^T\mathbf{P} = \mathbf{I}$. The solution is given by $\mathbf{P} = \mathbf{U}\mathbf{V}^T$, where U and V are the left and right singular vectors of $\mathbf{X}^T\mathbf{X}\mathbf{W}$ (ten Berge, 1993; Zou et al., 2006).

In order to obtain the component weight matrix $\mathbf{W}$ that minimizes equation (2.9) conditional on $\mathbf{P}$, we implemented a coordinate descent optimization procedure. In order to use coordinate descent to get estimates for $\mathbf{W}$ we rewrite equation (2.9) as follows,

$$\begin{aligned} \underset{\mathbf{W}}{\arg\min}\, L(\mathbf{W}) &= \|\mathbf{X} - \mathbf{X}\mathbf{W}\mathbf{P}^T\|_2^2 + \lambda_1 \|\mathbf{W}\|_1 + \lambda_2 \|\mathbf{W}\|_2^2 \\ &= \|\text{vec}(\mathbf{X}) - (\mathbf{P} \otimes \mathbf{X})\text{vec}(\mathbf{W})\|_2^2 + \lambda_1 \|\text{vec}(\mathbf{W})\|_1 \\ &+ \lambda_2 \|\text{vec}(\mathbf{W})\|_2^2, \end{aligned} \tag{2.10}$$

where $(\mathbf{P} \otimes \mathbf{X}) \in \mathbb{R}^{IL \times QL}$ denotes the Kronecker product between the factor loading matrix $\mathbf{P}$ and the data $\mathbf{X}$ and where $\text{vec}(\mathbf{X})$ denotes the column vector representation of $\mathbf{X}$. In the Kronecker product each entry of $\mathbf{P}$ is multiplied with $\mathbf{X}$ and put together in one big matrix. An example of $(\mathbf{P} \otimes \mathbf{X})$ and $\text{vec}(\mathbf{X})$ is given by:

$$(\mathbf{P} \otimes \mathbf{X}) = \begin{bmatrix} p_{11}\mathbf{X} & \dots & p_{1Q}\mathbf{X} \\ \vdots & \ddots & \vdots \\ p_{J_K 1}\mathbf{X} & \dots & p_{J_K Q}\mathbf{X} \end{bmatrix}, \quad \text{vec}(\mathbf{X}) = \begin{bmatrix} x_{11} \\ \vdots \\ x_{I1} \\ x_{12} \\ \vdots \\ x_{I2} \\ x_{1J} \\ \vdots \\ x_{IJ} \end{bmatrix}.$$

This optimization problem can be recognized as an elastic net regression problem (Zou and Hastie, 2005), to demonstrate this let $\text{vec}(\mathbf{X})$ be the outcome variable $\mathbf{y}$, let $(\mathbf{P} \otimes \mathbf{X})$ be the matrix $\mathbf{X}^*$ and let $\text{vec}(\mathbf{W})$ be the coefficient vector $\boldsymbol{\beta}$. For clarity we index the rows and columns of $\mathbf{X}^*$ by m and n respectively. To find estimates for $\boldsymbol{\beta}$ we will use a coordinate descent procedure (Friedman et al., 2010). This is an iterative procedure for which the solution to each successive approximation of β_n is given by,

$$\beta_n := \frac{S(\frac{1}{M}\sum_m x^*_{mn}(r_m + x^*_{mn}\widetilde{\beta}_n), \lambda_1)}{\frac{1}{M}\sum_m x^*_{mn}x^*_{mn} + \lambda_2}, \tag{2.11}$$

where $S(z, \gamma)$ denotes the soft-thresholding operator: $\text{sign}(z)(|z| - \gamma)_+$ and $(\alpha)_+$ denotes the positive part function, $\widetilde{\beta}_n$ denotes the current estimate of β_n and r_m denotes the current residual $y_m - \sum_n x^*_{mn}\widetilde{\beta}_n$. The coefficients get updated until a convergence criterion has been reached. The component weights constraint by the common and distinctive structure are skipped by (2.11) and therefore stay zero. A minimum of (2.6) can be found by successively alternating between the estimation of $\mathbf{P}$ and $\mathbf{W}$ until the convergence criterion has been reached. Note that this optimization problem is not convex, meaning that a minimum of (2.6) does not have to be the global minimum. Note that $\mathbf{P}$ conditional on $\mathbf{W}$ and vice versa, have global optima because they are convex optimization problems.

The coordinate descent procedure in (2.11) in its current form is infeasible to work with and has to be rewritten in order for it the be useable in practice. This has two reasons, first, the procedure relies on dot products with $\mathbf{X}^*$, this Kronecker product can get very large which may result into memory problems and makes the procedure slow. Second, $\mathbf{r}$ needs to be calculated every time β_n has been updated, this is costly as $\mathbf{r} = \mathbf{y} - \mathbf{X}^*\boldsymbol{\beta}$. In order to make the coordinate descent procedure efficient, the kronecker product needs to be avoided. This can be done by noting some properties of $\sum_m x^*_{mn} x^*_{mn}$. First, let us focus on the dot product of two columns of $\mathbf{X}^*$ where each entry in those columns is multiplied by $p_{\bullet q}$ carrying the same index q, (index numbers with a $\bullet$ follow from the context) then because of the orthonormality of $\mathbf{P}$ recognize that $\sum_m x^*_{m\bullet} x^*_{m\bullet}$ can be rewritten as follows,

$$\begin{aligned}\sum_m^M x^*_{m\bullet} x^*_{m\bullet} &= \mathbf{x}^{*T}_\bullet \mathbf{x}^*_\bullet = \sum_l (p_{lq}\mathbf{x}_\bullet)^T (p_{lq}\mathbf{x}_\bullet) \\ &= \sum_l p^2_{lq} \mathbf{x}^T_\bullet \mathbf{x}_\bullet = \mathbf{x}^T_\bullet \mathbf{x}_\bullet. \end{aligned} \tag{2.12}$$

This means that the dot product of two columns of $\mathbf{X}^*$ multiplied by the same $p_{\bullet q}$, is the same as the dot product of the corresponding columns from the original data. The inner product of two columns from $\mathbf{X}^*$ multiplied by $p_{\bullet q}$ and $p_{\bullet z}$ where $q \neq z$ results in $\sum_l p_{lq} p_{lz} \mathbf{x}^T_\bullet \mathbf{x}_\bullet = 0$. Secondly, recognize that in (2.11),

$$\begin{aligned}\sum_m^M x^*_{mn} r_m &= \mathbf{x}^{*T}_n \mathbf{r} = \mathbf{x}^{*T}_n (\mathbf{y} - \mathbf{X}^* \widetilde{\boldsymbol{\beta}}) \\ &= \sum_l (p_{l\bullet}\mathbf{x}_\bullet)^T \mathbf{x}_l - \sum_l (p_{l\bullet}\mathbf{x}_\bullet)^T \mathbf{X}^* \widetilde{\boldsymbol{\beta}} \\ &= \sum_l (p_{l\bullet}\mathbf{x}_\bullet)^T \mathbf{x}_l - \sum_n \sum_l (p_{l\bullet}\mathbf{x}_\bullet)^T (p_{l\bullet}\mathbf{x}_n) \widetilde{\beta}_n \\ &= \sum_l (p_{l\bullet}\mathbf{x}_\bullet)^T \mathbf{x}_l - \sum_l \mathbf{x}^T_\bullet \mathbf{x}_l \widetilde{w}_{l\bullet} \\ &= \sum_l \mathbf{x}^T_\bullet \mathbf{x}_l (p_{l\bullet} - \widetilde{w}_{l\bullet}) \\ &= \mathbf{x}^T_\bullet \mathbf{X}(\mathbf{p}_\bullet - \widetilde{\mathbf{w}}_\bullet). \end{aligned} \tag{2.13}$$

Making use of (2.12) and (2.13) the updating equation (2.11) can be rewritten as,

$$w_{l\bullet} := \frac{S(\frac{1}{M}(\mathbf{x}^T_l \mathbf{X}(\mathbf{p}_\bullet - \widetilde{\mathbf{w}}_\bullet) + \mathbf{x}^T_l \mathbf{x}_l \widetilde{w}_{l\bullet}), \lambda_1)}{\frac{1}{M}\mathbf{x}^T_l \mathbf{x}_l + \lambda_2}, \tag{2.14}$$

which does not rely on the kronecker product. Note that each time a coefficient gets updated, the vector $\mathbf{X}(\mathbf{p}_\bullet - \widetilde{\mathbf{w}}_\bullet)$ needs to be calculated again, this matrix vector product can be partially avoided by calculating $\mathbf{X}(\mathbf{p}_\bullet - \widetilde{\mathbf{w}}_\bullet)$ before the updating of the weights in $\mathbf{w}_q$ starts. If during the updating w_{lq} is put to zero, the vector $\mathbf{x}_j \widetilde{w}_{lq}$ gets added back to $\mathbf{X}(\mathbf{p}_\bullet - \widetilde{\mathbf{w}}_\bullet)$, and if w_{lq} gets updated to a new value, the difference $\mathbf{x}_j(\widetilde{w}_{lq} - w_{lq})$ is added back to $\mathbf{X}(\mathbf{p}_\bullet - \widetilde{\mathbf{w}}_\bullet)$. The coordinate descent algorithm is given by Algorithm (1) and the full SCaDS algorithm is given by Algorithm (2).

Algorithm 1: Coordinate descent algorithm for the component weights

```
1  CoorDes ;
   Input  : $\mathbf{X}, \mathbf{W}, \mathbf{P}, \lambda_2, \lambda_1, \epsilon,$ fixed structure for $\mathbf{W}$
   Output: $\widehat{\mathbf{W}}$
2  $\mathbf{c}$ = empty array of length $I$
3  while convergence criterion $\epsilon$ is not satisfied do
4      for $q = 1$ to $Q$ do
5          $\mathbf{c} = \mathbf{X}(\mathbf{p}_q - \widetilde{\mathbf{w}}_q)$
6          for $l = 1$ to $L$ do
7              if $w_{lq}$ is not constraint $0$ then
8                  $\widetilde{w}_{lq} = w_{lq}$
9                  $a = \frac{1}{I}(\mathbf{x}_l^T\mathbf{c} + \widetilde{w}_{jq}\mathbf{x}_l^T\mathbf{x}_l)$
10                 $b = \text{sign}(a)(|a| - \lambda_1)_+$
11                 $w_{lq} = \frac{b}{\frac{1}{I}\mathbf{x}_l^T\mathbf{x} + \lambda_2}$
12                 if $|a| < \lambda_1$ then
13                     $w_{jq} = 0$
14                     $\mathbf{c} = \mathbf{c} + \widetilde{w}_{lq}\mathbf{x}_l$
15                 else
16                     $\mathbf{c} = \mathbf{c} + \mathbf{x}_l(\widetilde{w}_{jq} - w_{jq})$
17             end
18         end
19     end
20 end
21 return $\widehat{\mathbf{W}}$;
```

Algorithm 2: Algorithm for SCaDS

```
1  SCaDS ;
   Input  : $\mathbf{X}, Q, \lambda_2, \lambda_1, \epsilon_1, \epsilon_2,$ fixed structure for $\mathbf{W}$
   Output: $\widehat{\mathbf{W}}, \widehat{\mathbf{P}}$
2  initialize $\mathbf{W} \in \mathbb{R}^{J \times Q}$
3  while convergence criterion $\epsilon_1$ is not satisfied do
4      store $\mathbf{U}, \mathbf{V}^T$ from SVD($\mathbf{X}^T\mathbf{X}\mathbf{W}$)
5      $\mathbf{P} = \mathbf{U}\mathbf{V}^T$
6      $\mathbf{W}$ = CoorDes($\mathbf{X}, \mathbf{W}, \mathbf{P}, \lambda_2, \lambda_1, \epsilon_2,$ fixed structure for $\mathbf{W}$)
7  end
8  return $\widehat{\mathbf{W}}, \widehat{\mathbf{P}}$;
```

Table 2.4

Component weights for the family data resulting from thresholded SCA with rotation to the common and distinctive structure

	$\mathbf{w}_1$	$\mathbf{w}_2$	$\mathbf{w}_3$	$\mathbf{w}_4$	$\mathbf{w}_5$	$\mathbf{w}_6$
F: Relationship with partners	0	0	-0.41	0.36	0	0
F: Argue with partners	-0.43	0	0	0	0	0
F: Childs bright future	0	0	0	0	0	0.43
F: Activities with children	0	0	0.42	0.4	0	0
F: Feeling about parenting	-0.4	0	0	0	0	0
F: Communication with children	0	0	0	0	0	0
F: Argue with children	-0.45	0	0	-0.28	0	0
F: Confidence about oneself	-0.47	0	0	0	0	0
M: Relationship with partners	0	0	-0.46	0.33	0	0
M: Argue with partners	0	0	0	0	0	-0.26
M: Childs bright future	0	0	0	0	0	0.38
M: Activities with children	0	0	0	0	0	0
M: Feeling about parenting	0	0	0	0	0	0
M: Communication with children	0	0.42	0	0	0	0
M: Argue with children	0	0	0	-0.31	0	0
M: Confidence about oneself	0	0.41	0	0	0	0
C: Self confidence/esteem	0	0	0	0	-0.37	0
C: Academic performance	0	0	0	0	0	0.32
C: Social life and extracurricular	0	0	0	0	0	0.38
C: Importance of friendship	0	0	0	0	-0.43	0
C: Self Image	0	0	0	0	-0.50	-0.26
C: Happiness	0	0	0	0	-0.48	-0.29
C: Confidence about the future	0	0	0	0	-0.29	0
%VAF: per component	0.08	0.06	0.06	0.05	0.11	0.06
%VAF: total			41.9			

Note. The items starting with an F, M or C belong to the father mother or child block. Small absolute components weights have been set to zero in order to get just as much sparsity in the component weights as in the SCaDS solution in Table 2.2.

Table 2.5

Results of the simulation study for finding the underlying common and distinctive structure with 10-fold cross validation in the high dimensional setting

Sparsity	Noise	Best model[a]	One Std Error rule[b]
high	5%	0	0.05
high	25%	0	0.35
high	50%	0	0.15
low	5%	0	0.20
low	25%	0	0.05
low	50%	0	0.05

Note. [a]The proportion of cases where the model with the true structure is the model with the lowest MPRESS. [b]The proportion of cases the model with the true structure is selected based on the one standard error rule. The results are based on 20 replications in each condition.

Table 2.6

Results of the simulation study for finding the underlying common and distinctive structure with 10-fold cross validation in the low dimensional setting

Sparsity	Noise	Best model[a]	One Std Error rule[b]
high	5%	0	0.60
high	25%	0	0.95
high	50%	0	0.85
low	5%	0	0.65
low	25%	0	1.00
low	50%	0	0.85

Note. [a]: The proportion of cases where the model with the true structure is the model with the lowest MPRESS. [b]: The proportion of cases the model with the true structure is selected based on the one standard error rule. The results are based on 20 replications in each condition.

Chapter 3

Model selection techniques for sparse weight based PCA

Abstract

Many studies make use of multiple types of data that are collected for the same set of samples, resulting in so-called multi-block data (for example multi-omics studies). A popular analysis framework is sparse principal component analysis (PCA) of the concatenated data. The sparseness in the component weights of these models is usually induced by penalties. A crucial factor in the use of such penalized methods, is a proper tuning of the regularization parameters used to give more or less weight to the penalties. In this paper we examine several model selection procedures to tune these regularization parameters for sparse PCA. The model selection procedures include cross-validation, BIC, Index of sparseness, and the convex Hull procedure. Furthermore, to account for the multi-block structure, we present a sparse PCA algorithm with a group LASSO penalty added to it, to either select or cancel out blocks of data in an automated way. Also the tuning of the group LASSO parameter is studied for the proposed model selection procedures. We conclude that when the component weights are to be interpreted, cross-validation with the one standard error rule is preferred; alternatively, if the interest lies in obtaining component scores using a very limited set of variables, the convex Hull, BIC, and index of sparseness are all suitable.

Keywords: Sparse PCA, Model Selection, Multi-block Data

3.1 Introduction

Many studies make use of multiple types of data that are collected for the same set of samples, resulting in so-called multi-block data Tenenhaus and Tenenhaus (2011). Examples include multi-omics studies in which the same set of samples are profiled using different molecular assays such as mRNA expression, DNA methylation, DNA copy number, and somatic mutation data; see Wang et al. (2014) for a multi-omics study of breast cancer and Reinke et al. (2018) for a joint analysis of six different data blocks collected from 22 individuals from an asthma cohort. Another example are multi-modal studies that use different MRI modalities (e.g., anatomical, diffusion, and resting state functional magnetic resonance), e.g., to study the same group of Alzheimer patients (Schouten et al., 2016). Each of the data blocks gives a partial view of the complex system under study. A full understanding of how the system works, requires to understand both the drivers of the system that operate independently and those that operate only by concerted action. At the level of the data, this means that insight is needed in the relations between variables both within and between the data blocks: Components of the system that work independently will show up as variation that is determined by the variables of a single block only while those components that work by concerted action will show up as variation that is determined jointly by variables linked throughout the blocks. A particular challenge in studying the jointly and individually determined variation is the need to automatically select variables that are of interest; not only to ease interpretation but also because data are often collected in an untargeted way and one of the primary aims of the data analysis is to hint at variables that may be key players in the process under study (Rasmussen and Bro, 2012). This is of particular relevance when using high-throughput approaches resulting in thousands of measured variables.

Following the strong rise of multi-block data in many disciplines, several integrative methods for exploratory data analysis have been put forward including clustering and dimension reduction techniques and combinations thereof, see for example the review by Meng et al. (2016). Among the dimension reduction techniques, a number of methods that model joint and individual variation, also called common and distinctive latent variables or components, have been proposed (Smilde et al., 2017; Shu et al., 2019). Some of these methods perform variable selection (Lock et al., 2013; Gu and Van Deun, 2016; de Schipper and Van Deun, 2018) by adding a least absolute shrinkage and selection operator (LASSO) penalty to the objective function (Tibshirani, 1996). This penalty has the property to shrink the estimates to zero, some exactly with the implication that that variable does not contribute (e.g., a zero regression weight means that the predictor does not contribute to the prediction and a zero component weight does mean that the variable does not contribute to the component). The use of such penalties that introduce zeros in the estimates is the current state-of-the-art in variable selection. The main reasons for the popularity of penalties over subset selection methods such as best subset selection, are better stability of the penalized regression model (Tibshirani, 1996) and their computational efficiency (e.g., compared to calculating the solutions for all possible subsets of variables (Bertsimas et al., 2016)). A popular framework for the analysis of multi-block data is sparse principal component analysis (PCA) (in the multi-block case also known as sparse simultaneous component analysis, SCA; see Van Deun et al. (2011)) this framework will be the focus of the current paper.

A crucial factor in the use of penalized methods, is the tuning of the regularization parameters used to give more or less weight to the penalties. In practice, the amount of sparseness in the data and the number of common and distinct components are not known beforehand. Hence, to make good use of penalized PCA approaches, model selection tools are needed to determine the strength of the LASSO and group LASSO penalties. In the context of sparse PCA a few methods have been put forward to address this issue: these include popular solutions such as cross-validation (e.g.,Bro et al. (2008)) and the bayesian information criterion (BIC) (Guo et al., 2010; Croux et al., 2013) but also less known alternatives such as the Index of Sparseness (Gajjar et al., 2017; Trendafilov et al., 2017) and the Convex Hull (CHull) procedure (Timmerman et al., 2016; Wilderjans et al., 2012). A comparison of these methods in the context of sparse PCA misses.

In this paper we will discuss and evaluate several existing model selection procedures to select proper values of the tuning parameters used in sparse PCA. Furthermore we will extend sparse PCA with a group LASSO penalty (Yuan and Lin, 2006) to model the common and distinct variation, by selecting at the level of the data blocks. The main focus of the paper will be on comparing several model selection procedures with respect to finding those values of the tuning parameters that yield the correct structure of the data; this is, selecting the right set of variables both in the single block setting and in the multi-block setting with common and distinct variation. The following model selection procedures, and adaptations thereof, will be discussed: Cross validation with the Eigenvector method (Bro et al., 2008), BIC (Guo et al., 2010; Croux et al., 2013), Convex Hull (Wilderjans et al., 2012) and the Index of Sparseness (Gajjar et al., 2017; Trendafilov et al., 2017). We will examine these model selection procedures because they are readily available from the existing literature and can be used to estimate meta-parameters for the weight based PCA model with little to no modification of the original propositions. For sparse PCA we will examine these model selection procedures in a simulation study with a single block of data (the most common case where all variables are assumed to represent one unit of interest). For sparse SCA we will examine the procedures making use of multi-block data (several sets of variables are available for the same cases with variables within one set representing a unit of interest). In the multi-block case we will assess whether the model selection procedures produce a final model that correctly identifies the joint and individual structure of the components. In order to inform the analysis of the block structure of the variables, we implemented the group LASSO penality in a blockwise fashion, to either select or cancel out blocks of data in an automated way.

The remainder of the paper is structured as follows: First, we will introduce sparse PCA with the LASSO penalty and its extension to the multi-block setting including a group LASSO penalty. Second, we will discuss several existing or adapted model selection procedures for tuning the LASSO and group LASSO penalty in sparse PCA. Third, we will examine these model selection procedures in case of single and multi-block data in a simulation study. Lastly, we conclude with a discussion.

3.2 Sparse PCA for single and multi-block data

In this section we will introduce the notation and give a brief introduction to sparse PCA. Then we will discuss the extension to the multi-block setting and introduce the group LASSO penalty to account for common and distinct variation.

We will make use of the standardized notation proposed by Kiers (2000): Bold lower- and uppercases will denote vectors and matrices, respectively; the superscript "T" denotes the transpose of a vector or matrix, and a running index will range from 1 to its uppercase letter (e.g., there is a total of I cases where i runs from $i = 1, \ldots, I$).

Given is a data matrix $\mathbf{X}$ that contains the scores for I observations on J variables; we follow the convention to present the J variable scores of observation i in row i and thus $\mathbf{X}$ has size $I \times J$. PCA decomposes the data into Q components as follows,

$$\begin{aligned} &\mathbf{X} = \mathbf{X}\mathbf{W}\mathbf{P}^T + \mathbf{E} \\ &\text{subject to } \mathbf{P}^T\mathbf{P} = \mathbf{I}, \end{aligned} \tag{3.1}$$

where $\mathbf{W}$ is a $J \times Q$ component weight matrix, $\mathbf{P}$ is a $J \times Q$ loading matrix, and $\mathbf{E}$ is a $I \times J$ residual matrix. Often the model is presented using the notation $\mathbf{T}$ for the component score matrix that results from the linear combinations shown explicitly in $\mathbf{XW}$. In this type of representation of the PCA model, interpretation is usually based on the loadings. Yet, an attractive property of the PCA formulation in (3.1) is that it explicitly shows how the variables contribute to the construction of the components: the meaning of the components scores $t_{iq} = \sum_j x_{ij} w_{jq}$ can be derived by inspecting what variables are weighted together to form the components; see de Schipper and Van Deun (2018) for a further discussion of weights versus loadings. Automatic selection of variables that contribute to the component scores can be obtained by penalizing $\mathbf{W}$ in the least squares problem that is typically solved to obtain suitable estimates for the component weights and loadings. This leads to the following penalized least squares problem: minimize with respect to $\mathbf{W}$ and $\mathbf{P}$

$$\begin{aligned} &L(\mathbf{W}, \mathbf{P}) = \|\mathbf{X} - \mathbf{X}\mathbf{W}\mathbf{P}^T\|_2^2 + \lambda_L \|\mathbf{W}\|_1 + \lambda_R \|\mathbf{W}\|_2^2 \\ &\text{subject to } \mathbf{P}^T\mathbf{P} = \mathbf{I} \end{aligned} \tag{3.2}$$

with $\|\mathbf{W}\|_1 = \sum_{j,r} |w_{jq}|$ the LASSO penalty - tuned by $\lambda_L \geq 0$ - and $\|\mathbf{W}\|_2^2 = \sum_{j,r} w_{jq}^2$ the ridge penalty, also known as Tikhonov regularization - tuned by $\lambda_R \geq 0$. The objective function in Equation (3.2) has been popularized by Zou et al. (2006). As pointed out there, the inclusion of a ridge penalty is needed in the high dimensional setting, this is having $J > I$; the combination of LASSO and ridge is known as the elastic net.

The decomposition in (3.1) can be extended to the case of multi-block data by taking $\mathbf{X} = [\mathbf{X}_1 \dots \mathbf{X}_K]$; this is concatenating the K data blocks composed of different sets of variables for the same observation units. The decomposition of $\mathbf{X}$ has the same block structured decomposition with $\mathbf{W} = [\mathbf{W}_1^T \dots \mathbf{W}_K^T]^T$ and $\mathbf{P} = [\mathbf{P}_1^T \dots \mathbf{P}_K^T]^T$. This multi-block formulation of PCA is known as simultaneous component analysis (Van Deun et al., 2009). Also in the multi-bock case $\mathbf{W}$ can be penalized to obtain sparse weights, we will call this variant sparse SCA. When analyzing multi-block data with sparse SCA, we can search for blockwise structures in the component weights that tell us whether a component is uniquely determined by variables from one single data block (distinctive component), or whether it is a component that is determined by variables from multiple data blocks (common component). In other words, a distinctive component is a linear combination of variables of a particular data block only, whereas a common component is a linear combination of variables of multiple data blocks. An example of common and distinctive components in the situation with two data blocks is given below. The first two components are distinctive components, the third component is a common component,

$$\mathbf{T} = \begin{bmatrix}\mathbf{X}_1 & \mathbf{X}_2\end{bmatrix}\begin{bmatrix}\mathbf{W}_1 \\ \mathbf{W}_2\end{bmatrix} = \begin{bmatrix}\mathbf{X}_1 & \mathbf{X}_2\end{bmatrix}\begin{bmatrix} 0 & w_{1_1 2} & w_{1_1 3} \\ 0 & w_{2_1 2} & w_{2_1 3} \\ 0 & w_{3_1 2} & w_{3_1 3} \\ w_{1_2 1} & 0 & w_{1_2 3} \\ w_{2_2 1} & 0 & w_{2_2 3} \\ w_{3_2 1} & 0 & w_{2_2 3} \end{bmatrix}.$$

In total there are $\binom{(2^K-1)+Q-1}{Q}$ possible combinations of common and distinctive components. There are $2^K - 1$ states for each component (minus one to exclude components with only zero weights) and each of these specific states can be assigned to each of the components where the ordering does not matter. Therefore counting all possible common and distinct configurations for Q components takes on the form of unordered sampling with replacement.

In the work of de Schipper and Van Deun (2018) the challenge of finding the right sparse block structure for the component weight matrix was handled by an exhaustive approach, examining all possible common and distinctive structures. If the number of components and blocks is not too large, calculating all possible models is feasible. However, if the number of blocks and components is large it is not and can be expected to yield highly variable results (as is the case with the best subset selection method for variable selection). Another option to perform selection at the level of the blocks, is to add a group LASSO penalty to the PCA objective; see Jenatton et al. (2010), Van Deun et al. (2011), and Erichson et al. (2020), for similar proposals. Let $\mathbf{w}_q^{(k)}$ denote the component weights of the variables of block k in component q. To have selection both at the level of the blocks as well as within blocks, the following penalized least squares criterion can be used:

$$L(\mathbf{W}, \mathbf{P}) = \|\mathbf{X} - \mathbf{XWP}^T\|_2^2 + \lambda_L\|\mathbf{W}\|_1 + \lambda_R\|\mathbf{W}\|_2^2 + \lambda_G \sum_{q,k}(\sqrt{J_k}\|\mathbf{w}_q^{(k)}\|_2) \tag{3.3}$$

subject to $\mathbf{P}^T\mathbf{P} = \mathbf{I}$.

Hence the group LASSO is tuned by $\lambda_G \geq 0$ with sufficiently large values resulting in components that are based on a linear combination of variables of just one or a few data blocks. To find estimates that minimize Equation (3.3) under the constraint of orthonormal component loading vectors, we rely on an alternating procedure that yields a non-increasing sequence of loss values thus converging — in practice — to a fixed point. The details of this numerical routine are discussed in the Appendix (3.7.1). Importantly, the numerical procedure only optimizes with respect to the component weights and loadings and thus needs fixed values for the number of components and the tuning parameters of the penalties. How to obtain suitable values for λ_L, λ_R, and λ_G is the main topic of this paper.

3.3 Model selection procedures for sparse PCA

In this paper we will discuss several model selection techniques for the selection of the penalty tuning parameters. These methods are Cross validation with the Eigenvector method (Bro et al., 2008), the BIC (Guo et al., 2010; Croux et al., 2013), Convex Hull (Wilderjans et al., 2012) and the Index of Sparseness (Gajjar et al., 2017; Trendafilov et al., 2017). These model selection techniques have been previously proposed in the context of PCA, some also in the context of sparse PCA as defined here, this is with penalties on the weights. Application of these methods to sparse SCA with a group LASSO penalty is novel. A thorough comparison of these methods -both for sparse PCA as well as SCA - lacks.

Cross validation with the Eigenvector method In the context of PCA, cross-validation can be applied in several ways; a discussion and comparison with respect to selecting the number of components for the $\mathbf{X} = \mathbf{T}\mathbf{P}^T$ model can be found in Bro et al. (2008). In that comparison, the best performing method was cross-validation (CV) with the Eigenvector method; de Schipper and Van Deun (2018) discussed the method in the context of sparse SCA to determine the value of the LASSO and ridge tuning parameters. Let $^{(-j)}$ denote that (the coefficients of) variable j are removed. Following Bro et al. and de Schipper and Van Deun, given a number of components Q, to determine the value of a tuning parameter λ the method then works as follows[1]:

1. Divide the sample into K folds each of size I_k
2. Leave out the kth fold and calculate $\widehat{\mathbf{W}}$ and $\widehat{\mathbf{P}}$ on the remainder given a set of tuning parameters λ
3. For the left-out samples in the kth fold $i = 1 \ldots I_k$, for variables $j = 1 \ldots J$
   a) Estimate the score as $\mathbf{t}_i^{(-j)} = \mathbf{x}_i^{(-j)T}\widehat{\mathbf{W}}^{(-j)}$
   b) Estimate x_{ij} as $\widehat{x}_{ij} = \mathbf{t}_i^{(-j)}\widehat{\mathbf{p}}_j^T$, where $\widehat{\mathbf{p}}_j$ is the jth row of $\widehat{\mathbf{P}}$
   c) Find the prediction error of the element x_{ij} by taking $e_{ij} = x_{ij} - \widehat{x}_{ij}$
4. Calculate the mean squared error of the kth fold, $\widehat{\mathrm{MSE}}(\lambda)_k = \frac{1}{I_k J}\sum_i^{I_k}\sum_j^J e_{ij}^2$

[1]Note that here K is used to denote the number of folds used in the cross-validation procedure and *not* the number of data blocks

5. Repeat 2 and 3 for each fold and calculate the overall mean squared error,

$$\widehat{\text{MSE}}(\lambda) = \frac{1}{\sum I_k} \sum_{k=1}^{K} I_k \widehat{\text{MSE}}(\lambda)_k. \tag{3.4}$$

The standard error of Equation (3.4) is obtained by taking the sample standard deviation of $\widehat{\text{MSE}}(\lambda)_1$, . . . , $\widehat{\text{MSE}}(\lambda)_K$ divided by $\sqrt{K}$ (see for example Gordon et al., 1984). Typically the data is split into $K = 10$ folds of (approximately) equal size, which we will also do in the current paper. The attractive features of CV with the Eigenvector method are that it is relatively fast to perform and that the estimated data $\widehat{x}_{ij}$ are obtained independent of the data used to construct the model. For more detailed information we refer the reader to Bro et al. (2008).

The model with the lowest MSE, is chosen as the best model. Cross validation tends to select models that are too complex, therefore the one standard error rule was developed (Hastie et al., 2009b). The one standard error rule selects the set of tuning parameters that lead to the least complex model, still within one standard error of the best model. In this paper we will examine the models chosen according to the best (this is having lowest MSE) and the one standard error rule.

The BIC criterion Let RV be the residual variance resulting from the PCA decomposition with Q components,

$$RV = \|\mathbf{X} - \mathbf{X}\widehat{\mathbf{W}}\widehat{\mathbf{P}}^T\|_2^2. \tag{3.5}$$

Likewise let $\widetilde{\text{RV}}$ denote the residual variance for a given a model with a specific λ and Q. Following Guo et al. (2010) and Croux et al. (2013) the BIC for a set of tuning parameters λ and given the number of components Q is then given by,

$$\text{BIC}(\lambda) = \frac{\widetilde{\text{RV}}}{\text{RV}} + \text{df}(\lambda)\frac{\log(I)}{I}. \tag{3.6}$$

with $\text{df}(\lambda)$ the number of non-zero weights in $\widehat{\mathbf{W}}$. The optimal set of λ values is then defined as the set of λ's that results in the model with the lowest BIC.

CHull: a convex hull based model selection method CHull (Wilderjans et al., 2012), also known as L-Curve (see for example, Hansen and OLeary (1993)), is a generic model selection procedure that aims at striking an optimal balance between the goodness of fit/misfit and model complexity. As stated by the authors: "The CHull procedure consists of (1) determining the convex hull of the fit-measure-by-complexity-measure plot of the models under consideration and (2) identifying the model on the boundary of the convex hull for which it is true that increasing the complexity (i.e., adding more parameters) has only a small effect on the fit measure, whereas lowering complexity (e.g., dropping parameters from the model) changes the goodness of fit (or, respectively, the misfit) substantially" (Wilderjans et al., 2012, p. 2). In this application of CHull we will use the variance accounted for (VAF) as a goodness-of-fit-measure:

$$VAF_\lambda = \frac{\|\mathbf{X}\widehat{\mathbf{W}}\widehat{\mathbf{P}}^T\|_2^2}{\|\mathbf{X}\|_2^2}. \tag{3.7}$$

This is the goodness-of-fit-measure the authors originally used in their application of CHull as a method to determine the number of components in PCA. In our example we will also make use of the $\widehat{\mathrm{MSE}}(\lambda)$ as a goodness-of-fit-measure, that is the MSE values we obtain from the cross-validation procedure as described before, see Equation (3.4). Our motivation is that VAF_λ is subject to overfitting and gives a downward biased estimate of the error. For the complexity measure we use the number of non-zero weights in $\widehat{\mathbf{W}}$. The models are selected using the multichull package (Vervloet et al., 2017).

Index of Sparseness According to Gajjar et al. (2017) and Trendafilov et al. (2017), the index of sparseness (IS) given by

$$\mathrm{IS}(\lambda) = VAF_{pca} \times VAF_\lambda \times \frac{\mathrm{df}(\lambda)}{JQ}, \tag{3.8}$$

where $\mathrm{df}(\lambda)$ is defined as previously, the VAF_{pca} is given by Equation (3.7) with $\widehat{\mathbf{W}}$ and $\widehat{\mathbf{P}}$ resulting from the PCA decomposition with Q components and all $\lambda = 0$, and VAF_λ is also given by Equation (3.7) but with $\widehat{\mathbf{W}}$ and $\widehat{\mathbf{P}}$ resulting from PCA with Q components and a set of regularization parameters $\lambda \geq 0$. The IS increases with goodness-of-fit and the sparseness of the solution. The (combination of) value(s) of the tuning parameter(s) λ that result(s) in the model with the largest IS is picked as the optimal value(s).

3.4 Simulation studies

The model selection techniques are assessed under different conditions by means of a simulation study. First we will discuss the case of a single block of data with an unstructured sparsity pattern, then we will discuss the case of multi-block data with structured sparsity resulting in common and distinct variation.

3.4.1 Single block data

In the simulation study, we kept the number of variables fixed to $J = 50$ and the number of components to $Q = 3$. The study included the following design factors:

- The number of observation units I: $25, 50$ and 100.
- The level of sparseness (percentage of the -in total $JQ = 150$ weights - that are equal to zero): 30% and 80%.
- The noise level: 5% and 20%.

The design is fully crossed, resulting in $3 \times 2 \times 2 = 12$ design cells. For each design cell, 50 data sets were simulated. The generation of the data is detailed in the appendix, see Section (3.7). The resulting data was analyzed using an implementation of Algorithm (3) (see the appendix) in the R software for statistical computing (R Core Team, 2020). Algorithm (3) is freely available in R (R Core Team, 2020) and downloadable from github.com/trbKnl. Each data set was analyzed using a 50×10 grid of LASSO and ridge penalty tuning parameters respectively. For the ridge, a sequence of ten values equally spaced on the interval $\ln 0$ to $\ln 500$ was used and for the LASSO 50 equally spaced values on the same interval. Note that the values were back-transformed to the range $0 - 500$. For each obtained (sparse) PCA model, the model selection statistics were calculated and a best model was obtained for each of the six model selection methods. The chosen models according to the model selection criterion was then evaluated by looking at the following performance measures:

- The similarity between the true model matrix $\mathbf{W}$ and the estimated $\widehat{\mathbf{W}}$. We use Tucker congruence between the vectorized version of $\mathbf{W}$ and $\widehat{\mathbf{W}}$ to measure the similarity. The Tucker congruence (also known as cosine similarity) is defined as the cosine of the angle between two vectors. If the two vectors share no similarity, they are orthogonal and the Tucker congruence will be 0. If the vectors are linearly dependent, i.e. perfect similarity, the angle between these two vectors is 0 and the Tucker congruence will be 1.
- The percentage of correctly identified zero weights, calculated as the percentage of zero weights in the true matrix that are recovered as a zero weight in the estimated matrix.
- The percentage of correctly identified non-zero weights, calculated as the percentage of non-zero weights in the true matrix that are recovered as a non-zero weight in the estimated matrix..

Results The results of the simulation study for the single block data are summarized in Figures 3.1 and 3.2. Figure 3.1)shows the Tucker congruence coefficient for the different model selection methods. Usually a threshold of $.85$ is recommended (Lorenzo-Seva and ten Berge, 2006). In the condition where the sparsity is 80%, only 10-fold CV, 10-fold CV with the 1 standard error rule, and CHull with the MSE, often attain Tucker congruence values above the threshold-value of acceptable similarity. Interestingly, CHull with MSE performs well while this is not the case for the CHull badness-of-fit measure previously used in the literature. In the conditions were the sparsity is 30%, only 10-fold CV and 10-fold CV with the one standard error rule, attain Tucker Congruence values above 0.85. This means that the BIC, IS, and the CHull with VAF procedures result in models where the estimated component weights are too dissimilar from the true component weights. When the true underlying models are very sparse (the conditions with 80% of sparsity), the procedures in general perform better.

Because the Tucker congruence coefficient is relatively insensitive to whether the correct status of the weights (i.e., zero or non-zero status) is estimated back, we also inspect whether the model selection procedures result in models that select the right subset of variables. The results are summarized in Figure 3.2. Three patterns can be discerned. First, cross-validation finds almost 100% of the non-zero weights yet recovers very few of the zero weights; this confirms that cross-validation is known to yield too complex models. Second, the index of sparseness, BIC and CHull with VAF, show the opposite behavior and favor very sparse models which results in good recovery of the zero weights at the expense of recovering very few of the non-zero weights. Third, cross-validation with the one standard error rule yields a high percentage of recovery both for the zero and non-zero weights.

It may seem surprising that most of these model selection techniques perform badly while having showed good performance in the literature with sparse loadings (e.g., Gu et al., 2019). This can be explained by the fact that — for the reconstruction of the data — the component scores and the loadings matter while the component weights play an indirect role. The component weights enter in the construction of the scores: $\widehat{\mathbf{T}} = \mathbf{X}\widehat{\mathbf{W}}$. As long as the scores are recovered well, the data are reconstructed well. This is the case for the data here: Tucker congruence between $\widehat{\mathbf{T}}$ and $\mathbf{T}$ is larger than 0.85 for the bulk of the selected models with each of the model selection procedures, see Figure 3.3. This in fact means that the component scores themselves can be retrieved rather well without the need of having to estimate that many non-zero weights. Hence model selection procedures that balance fit with the number of non-zero coefficients result in very sparse models. This implies that few weights actually need to be estimated in order for the model

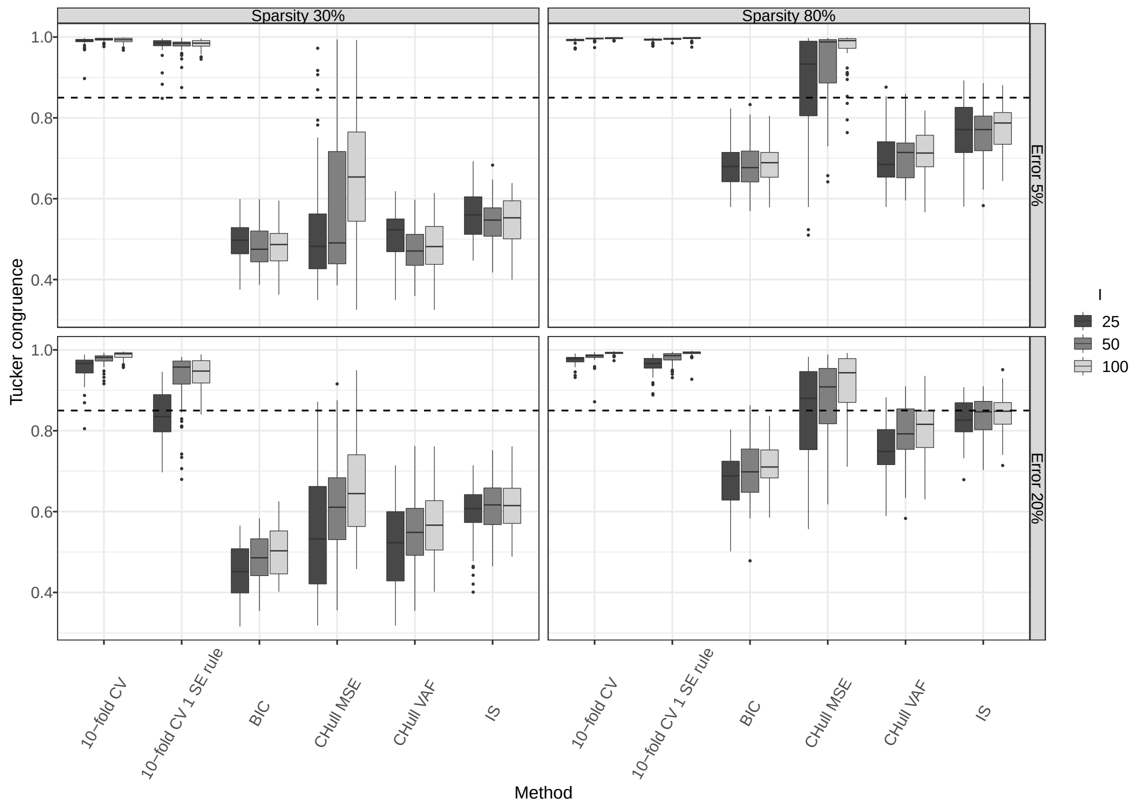


Figure 3.1. The Tucker congruence coefficient between $\mathbf{W}$ and $\widehat{\mathbf{W}}$ for the various model selection procedures. The dashed line indicates a threshold value of 0.85 used as a cut-off for fair similarity. In each condition 50 replicate data sets were used. The boxplots display the median and upper and lower quartiles.

to attain a good fit.

3.4.2 Multi-block data

In this simulation study we assess the performance of the model selection criteria for the case of multi-block data that have structured sparsity, this is we assume the component weights to have a common and distinctive structure. Here, particular interest will be in evaluating whether the model selection methods recover the common and distinctive structure.

Simulation study design The data that will be analyzed in this simulation study consist of 2 data blocks ($\mathbf{X} = [\mathbf{X}_1 \quad \mathbf{X}_2]$) each with 25 variables. The structure imposed on the data is $Q = 3$ components with 2 distinctive components and 1 common component. The study includes the following design factors:

- The number of samples I: 25 and 100.

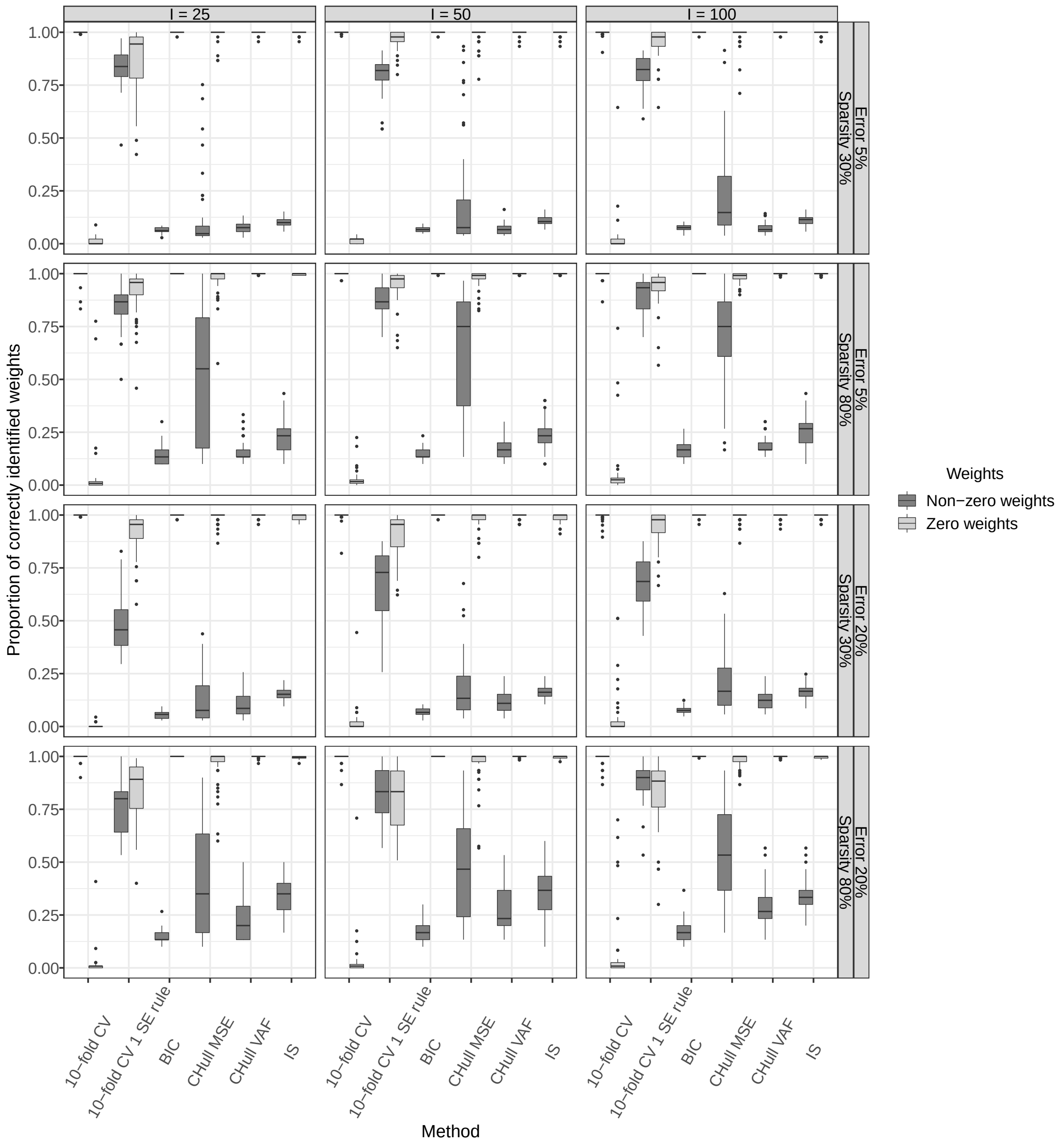


Figure 3.2. The percentage of correctly classified weights in $\widehat{\mathbf{W}}$, for the various model selection procedures. For good recovery, both the percentage of correctly classified non-zero and zero-weights should be high. The boxplots display the median and upper and lower quartiles.

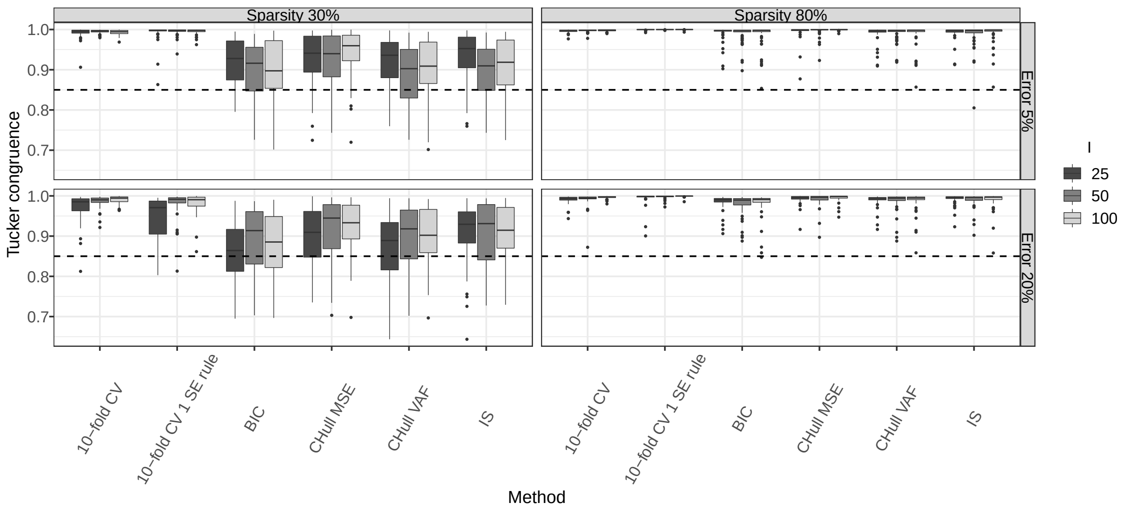


Figure 3.3. The Tucker congruence coefficient between $\mathbf{T}$ and $\widehat{\mathbf{T}}$ for the various model selection procedures. The dashed line indicates a threshold value of 0.85 used as a cut-off for fair similarity. In each condition 50 replicate data sets were used. The boxplots display the median and upper and lower quartiles.

- The level of sparseness in the non-zero blocks in the columns of $\mathbf{W}$: 30% and 80%.
- The noise level: 5% and 20%.

The design was fully crossed, resulting in $2 \times 2 \times 2 = 8$ design cells. For each design cell 50 datasets were generated. The details of the data generation scheme used can be found in Appendix Section 3.7. The data were analyzed with Algorithm 3 for a grid of LASSO, group LASSO, and ridge tuning parameters. The sequences of the ridge, LASSO and group LASSO parameters are given by a sequence from 0 to 500 of length 10 on the natural log scale for each of the three tuning parameters. The chosen model according to the model selection criteria is then evaluated by looking at the following performance measures:

- Tucker congruence coefficient.
- Whether the 2 distinctive components are estimated back (i.e. all weights in the zero segments are estimated as zero),
- Whether the common component is estimated back (i.e. at least one non-zero weights in each data block).

Results The results of the multi-block simulation study are summarized in Figure 3.4, and Tables 3.1 and 3.2. In Figure 3.4 the Tucker congruence coefficients are displayed; these mainly show low congruence, this is below the threshold of

0.85, except for the two cross-validation procedures. Also here, as was the case in the single block simulation, the low Tucker congruence coefficients are caused by most model selection procedures having put too many weights to zero, compared to the actual number of zero weights. Compared to the first simulation, Tucker congruence is a bit higher because the distinctive components induce higher levels of overall sparsity, meaning that the true model is more sparse and thus supportive of selection methods that favor higher levels of sparsity.

We now turn to the question whether the model selection methods recover the common and distinct components. Table 3.1 summarizes whether the common component is identified for the different model selection procedures, this is whether at least one non-zero component weight within each block was retained. Table 3.2 summarizes whether the distinctive components are identified by the different model selection procedures (this is, whether all weights of the block not making up the component are set to zero). Together, these tables show the same patterns previously observed for the single block simulation study: cross-validation favors complex models which results in defining most components as common and not finding the distinctive components; the BIC, CHull-VAF, CHull-MSE, and the IS estimate models that are too sparse and hence declare most components to be distinctive at the expense of the common components; again, only 10-Fold CV with the one standard error rule accurately estimates the sparsity, both within and between blocks.

To decide on which method is best based on combining the identification rates for the common and distinct components, we used sum of ranking differences (SRD) scores and summarized these in a plot. SRD scores is a consensus decision making tool for situations with multiple optimality criteria (Lourenco and Lebensztajn, 2018) (for further reading also see Héberger (2010)). Here, the scores are based on rankings of the model selection procedures on the basis of the identification rates for common (see Table 3.1) and distinctive (see Table 3.2) components. For further details on how to obtain the SRD score we refer to Lourenco and Lebensztajn (2018). The SRD scores are summarized in Figure 3.5 with lower scores indicating better overall performance of the method. The grey solid curve denotes the cumulative distribution of SRD scores based on a random ranking of the methods on the different optimality criteria (we relied on an approximate distribution). In the plot, the score corresponding to the 0.05 smallest SRD scores for the randomly ranked methods is indicated: this is our chosen cut off for significance with methods having higher scores being considered to not perform consistently better on each of the optimality criteria than based on a random ranking. It can be observed that only 10-Fold CV with the one standard error rule falls (just) below the cut-off indicating that it is all-round better than the other methods. The other model selection procedures do not consistently perform relatively better.

For the interested reader we will provide an example of the analysis of multi-block data making use of Equation (3.3) in the next section.

Table 3.1

Common components identified in percentages

	Error 5%				Error 20%			
	I = 25		I = 100		I = 25		I = 100	
	Sparsity 30%	Sparsity 80%	Sparsity 30 %	Sparsity 80%	Sparsity 30%	Sparsity 80%	Sparsity 30%	Sparsity 80%
10-Fold CV	100	98	100	96	100	100	100	100
10-Fold CV 1std error	88	78	100	82	96	82	88	82
BIC	44	12	82	36	58	30	48	20
CHull-MSE	46	86	88	92	66	62	94	70
CHull-VAF	66	44	82	38	82	62	96	68
IS	76	52	82	56	96	76	96	68

Note. The percentages of times the common component was identified (this is at least one non-zero weight in each data block). The percentages are based upon 50 replicate data sets.

Table 3.2

Distinctive components identified in percentages

	Error 5%				Error 20%			
	I = 25		I = 100		I = 25		I = 100	
	Sparsity 30%	Sparsity 80%	Sparsity 30 %	Sparsity 80%	Sparsity 30%	Sparsity 80%	Sparsity 30%	Sparsity 80%
10-Fold CV	8	2	8	4	0	2	6	6
10-Fold CV 1std error	82	88	98	92	94	70	90	72
BIC	98	100	96	100	98	100	98	100
CHull-MSE	92	70	78	84	84	76	80	82
CHull-VAF	82	94	94	100	68	68	90	100
IS	78	92	92	100	46	48	92	100

Note. The percentages of times the two distinctive components were found (this is no non-zero weights estimated in the zero data block). The percentages are based upon 50 replicate data sets.

3.5 Empirical Example: Herring data

We will now provide an illustrative example were we analyse a dataset on salted herring samples using sparse weight based SCA. The dataset on salted herring consists of two blocks, each containing a specific set of variables on 21 herring samples with the samples corresponding to different ripening conditions; see (Bro et al., 2002) and (Nielsen et al., 1999) for more information about the data. The first block contains chemical and physical measurements whereas the second block consists of sensory variables. For an overview of the variables names see Table 3.3.

The analysis of multi-block data follow three steps:

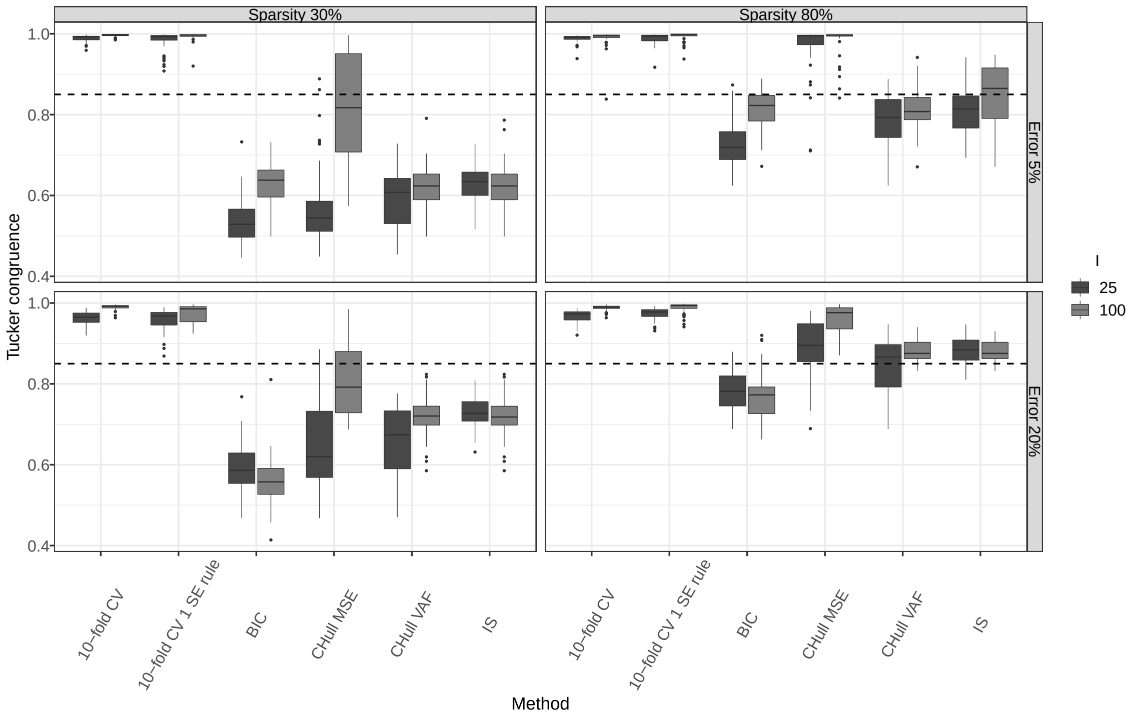


Figure 3.4. The Tucker congruence coefficient between $\mathbf{W}$ and $\widehat{\mathbf{W}}$ for the various model selection procedures in case of multi-block data. The dashed line indicates a threshold value of 0.85 indication fair similarity. In each condition 50 replicated data sets were used. The boxplots display the median and upper and lower quartiles.

- Pre-processing of the data
- Tuning the model; selecting the meta-parameters
- Analysing the final model; interpreting the component weights

We will discuss each of these steps here below.

Pre-processing of the data Pre-processing has a large impact on the final results of the analysis, and should be done according to the needs of the researchers; see Van Deun et al. (2009) for an overview. For the herring data here, we first centered and scaled (to unit variance) the variables as we are not interested in scale differences. As the two blocks have the same number of variables, no further block scaling is needed.

Tuning the model Multiple meta-parameters need to be tuned in order to get to a satisfactory final model. For the sparse PCA method that we use here these are the number of components and the regularization parameters λ_L, λ_G and λ_R.

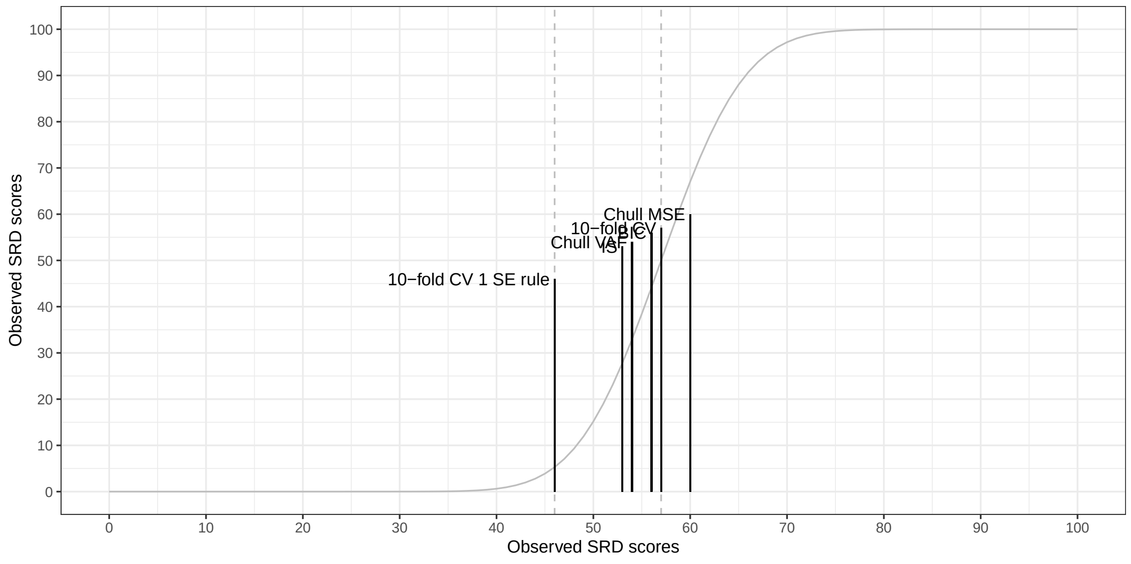


Figure 3.5. The sum of ranking differences scores of the various model selection procedures. The solid grey line indicates an approximation of the cumulative distribution function of random SRD-scores. The vertical dashed lines indicate the 0.05 and 0.95 cut off points. Model selection procedures having an SRD score to the right of the 0.05 cut are not statistically different, at the .05 level of significance, from random rankings.

Also for the selection of the number of components, cross-validation has been recommended (Bro et al., 2008). Hence, two strategies can be considered, namely tuning all parameters together or following a sequential strategy. Because of the computational burden of the simultaneous strategy we opt for the sequential approach: First, we select the number of components and then, given the selected number of components, we tune the LASSO and group LASSO parameters. To determine the number of components we used 10-fold cross-validation with the 1 standard error rule on each block. This resulted twice in three components hence we analyzed the concatenated data with the maximum number of components possible, this is six distinctive components.

Also the regularization parameters were tuned using 10-fold cross-validation with the 1 standard error rule. More specifically, we tuned the LASSO, ridge and group LASSO regularization parameters on a three dimensional grid with 25, equally spaced values between 0 to 500 on the log-scale; for the data here this covers solutions ranging from no sparseness at all to all coefficients being zero. We chose a log-scale because it tends to do well in practice and has been recommended elsewhere, see Friedman et al. (2010, p. 10). Note that the upper bound depends on the scale of the data.

Analysis of the final mode and interpretation of the results The component weights resulting from the final analysis (this is, using six components and with the values of the regularization parameters set at those selected under the cross-validation scheme) are summarized in Table 3.3. Note that in this case there are 2 distinctive components (components 2 and 5), and 4 common components (components 1, 3, 4, and 6). The component weights directly relate the components to the observed variable asă$t_{iq} = \sum_j w_{jq}x_{ij}$. For comparison we included results from a non-spare PCA analysis of the concatenated data in Table 3.4 where the weights/loadings [2] are estimated using the singular value decomposition and subsequently rotated to a simple structure using varimax rotation (Kaiser, 1958). Strikingly, there is no structuring of the components into common and distinctive components. Furthermore, components in PCA are a linear combination of all variables and interpreting these is much more difficult compared to sparse SCA. Take for example the fifth component, in case of PCA with varimax rotation, the component is a linear combination of mainly the variable Spice, but other variables are weighted as well with non-negligible loadings such as TCAIndexB, Malt, Stockfish smell et cetera. In the case of sparse SCA, the fifth component is just the variable Salty, making it the unequivocal Salty component. Importantly, the gain in interpretation obtained by imposing sparseness comes at barely any cost in terms of the variation accounted for.

As for the meaning of the components, we examine the first and most important component in terms of explained variance (42.4% variance explained) from the sparse SCA analysis. We observe that ProteinB, TCAIndexM, TCAM, TCAB, from the first block, together with Ripened, Malt, Sweetness, Softness, Toughness and Watery from the second block make up the first component. In Nielsen et al. (1999) the authors note that softness (the most important quality indicator used in the herring industry) correlates with TCAM/B TCAIndexM and ProteinB, which to them makes sense because: “A correlation between these parameters and softness may be expected as muscle proteins are broken down during the ripening thus explaining the increase in low molecular nitrogenous compounds and at the same time softening of the tissue is encountered. ProteinB is mainly salt soluble muscle protein diffusing into brine from the muscle. The solubilisation of muscle proteins will therefore also probably affect the texture.” (Nielsen et al., 1999, p. 23). Furthermore, they note that Softness, Toughness and Watery measure the same characteristics, and that TCAIndexM, TCAM and TCAB measure the same characteristics. This corresponds to the reported weights for the first component, except for the small weight of ProteinM. We could view component one as a “quality of herring” component. The first component obtained with PCA followed by varimax

[2]Note that the loadings and the weights are the same in PCA when $\mathbf{P}^T\mathbf{P} = \mathbf{I}$

rotation is also the most important component (31.9% variance explained) and we may expect this to also represent “quality of herring”. The weights for this component in Table 3.4 show a somewhat similar pattern, yet there are some deviations and the interpretation is much harder because all variables make up the first component. Furthermore this component explains less variance compared to the first component of sparse SCA.

Table 3.3

MM sparse SCA: Estimated component weights for the herring data

Components	1	2	3	4	5	6
pHB	0	-0.148	-0.481	0	0	0.162
ProteinM	-0.081	0.228	0	0	0	0
ProteinB	0.148	0	0	0.176	0	0
Water	0	-0.790	0	0	0	0
AshM	0	-0.126	0.595	-0.030	0	0
Fat	0	0.394	0	0	0	0
TCAIndexM	0.320	0	0	0	0	0.484
TCAIndexB	0	0.549	0	0	0	0
TCAM	0.102	0	0	0	0	0.502
TCAB	0.464	0	0	0	0	0
Ripened	0.275	0	0	0	0	0
Rawness	0	0	-0.126	-0.491	0	-0.196
Malt	0.402	0	0	-0.001	0	0
Stockfish smell	0	0.002	0.674	0	0	0
Sweetness	0.289	0	0	0	0	-0.530
Salty	0	0	0	0.855	0	-0.109
Spice	0	0	0	0	1.000	0
Softness	0.355	0	0	0	0	0
Toughness	0.349	0	0	0	0	0
Watery	0.360	0	0	0	0	0
%VAF: per component	42.4	20.0	11.4	9.1	5.5	5.2
%VAF: total			93.9			

Note. The first block, corresponding to the first 10 variables (rows), consist of physical and chemical analyses of the herring samples measured either in brine (B) or fish muscle (M). The second block contains sensory data on the herring samples. These are the results are obtained using Algorithm 3 with the chosen tuning parameters.

Table 3.4

Varimax: Estimated component weights for the herring data

Components	1	2	3	4	5	6
pHB	-0.107	-0.214	-0.470	0.053	-0.180	0.372
ProteinM	-0.116	0.341	-0.113	-0.036	-0.084	-0.107
ProteinB	0.191	-0.075	0.029	0.264	-0.008	0.097
Water	-0.009	-0.483	-0.076	-0.076	0.040	0.079
AshM	-0.054	-0.327	0.597	-0.138	0.105	-0.024
Fat	0.025	0.486	0.047	0.019	0.036	0.062
TCAIndexM	0.082	-0.046	-0.000	0.019	0.019	0.481
TCAIndexB	0.032	0.448	-0.027	-0.245	0.115	0.181
TCAM	0.071	0.042	-0.011	0.032	0.009	0.504
TCAB	0.222	0.120	0.080	0.180	0.002	0.160
Ripened	0.281	0.056	0.244	0.064	-0.135	0.070
Rawness	0.105	0.024	-0.110	-0.510	0.018	-0.266
Malt	0.354	0.027	0.006	-0.098	-0.117	0.108
Stockfish smell	-0.055	0.106	0.526	0.157	-0.161	0.079
Sweetness	0.502	-0.083	-0.176	0.060	-0.095	-0.353
Salty	-0.069	0.053	-0.105	0.709	0.074	-0.254
Spice	0.039	-0.012	-0.044	0.039	0.922	0.046
Softness	0.372	0.008	-0.018	0.004	0.076	0.029
Toughness	0.366	0.036	0.032	0.024	0.060	0.021
Watery	0.353	-0.101	-0.018	-0.023	0.016	0.011
%VAF: per component	31.9	19.9	11.8	12.5	5.0	12.8
%VAF: total			94			

3.6 Conclusion

The current paper examined several model selection procedures to select the penalty tuning parameters of sparse weight based PCA for the unstructured case of a single block of data and of sparse weights SCA for the multi-block case having structured sparsity. Most model selection procedures that have been proposed in the sparse PCA literature did not perform well in terms of finding back the correct component weights. When analyzing single block data, the procedures led to either too complex or too sparse models. When analyzing multi-block data it led to either identifying most components as common components and not as distinctive, or not identifying common components as such. The only model selection procedure that seems to strike a good balance between model complexity and goodness of fit in both the single and multi-block case was 10-fold cross-validation with the Eigenvector method employing the one standard error rule. It has to be noted that we did not tune the number of components together with the tuning parameters, this could be addressed in further research.

As discussed in the paper, although the weights are recovered badly, this barely affects the recovery of the component scores nor the reconstruction of the data and hints at the fact that the estimation of the weights is an ill-conditioned problem. Importantly, this means that if the goal is to obtain good estimates of the component scores, loadings, or data yet with no interest in the estimates of the component weights, proper tuning of the penalties on the weights is not needed. In this situation, an economical decision may be to select a very sparse model (e.g., as resulting from the IS, BIC, or CHull) as good estimates of the component scores can be obtained with few variables. Yet, when insight in the processes at play in the data is needed, our advice is to use cross-validation with the one standard error rule.

It has to be noted that a good solution for the component weights is in the eyes of the beholder, a situation where a very sparse solution might be desirable is when the component scores themselves are of interest and when observing new data is expensive. For newly observed cases, only the variables with non-zero component weights have to be observed to compute component scores.

3.7 Appendix

3.7.1 Description of algorithm

In order to obtain the component weights we need to optimize the following objective function with respect to $\mathbf{W}_c$ and $\mathbf{P}_c$,

$$\begin{aligned} L(\mathbf{W}_c, \mathbf{P}_c) =& \|\mathbf{X}_c - \mathbf{X}_c\mathbf{W}_c\mathbf{P}_c^T\|_2^2 + \lambda_L\|\mathbf{W}_c\|_1 + \lambda_R\|\mathbf{W}_c\|_2^2 \\ &+ \sum_{q,k}(\lambda_G\sqrt{J_k}\|\mathbf{w}_q^{(k)}\|_2 + \lambda_E\|\mathbf{w}_q^{(k)}\|_{1,2}) \end{aligned} \tag{3.9}$$

where $\mathbf{W}_c = [(\mathbf{W}^{(1)})^T \dots (\mathbf{W}^{(K)})^T]^T$, and $\mathbf{w}_q^{(k)}$ denotes the qth column from the submatrix $\mathbf{W}^{(k)}$. In order to get a minimum for (3.9) we alternate between the estimation of $\mathbf{W}_c$ and $\mathbf{P}_c$. Given $\mathbf{W}_c$ we can estimate $\mathbf{P}_c$ by using procrustes rotation (ten Berge, 1993; Zou et al., 2006). Given $\mathbf{P}_c$ we can find estimates for $\mathbf{W}_c$ by using the majorization minimization algorithm; for a short review see Hunter and Lange (2004). To majorize (3.9) we can majorize all individual terms seperately. First we majorize $\|\mathbf{W}_c\|_1$. For the ease of simplicity let $j = 1 \dots \sum_k J_k$

index the rows of $\mathbf{W}_c$ and let $q = 1 \dots Q$ index the columns of $\mathbf{W}_c$, then we can majorize $\|\mathbf{W}_c\|_1$ as follows,

$$\begin{aligned}\lambda_L \|\mathbf{W}_c\|_1 = \lambda_L \sum_{j,q} |w_{jq}| &\leq \lambda_L \sum_{q,j} \Big(\frac{1}{2} \frac{w_{jq}^2}{|\widetilde{w}_{jq}|} + \frac{1}{2} |\widetilde{w}_{jq}| \Big) \\ &= \frac{\lambda_L}{2} \mathrm{vec}(\mathbf{W}_c)^T \mathbf{D}_1 \mathrm{vec}(\mathbf{W}_c) + c,\end{aligned} \tag{3.10}$$

where $\widetilde{w}_{jq}$ is the current estimate of w_{jq}, vec() denotes the vectorized version of a matrix, $\mathbf{D}_1$ a diagonal matrix of $|w_{jq}^{-1}|$, and c contains the terms that do not depend on w_{jq} and thus can be neglected in solving the optimization problem with respect to the elements of $\mathbf{W}$. Next we consider a majorizing function for the QK group LASSO terms,

$$\begin{aligned}\lambda_G \sum_{k,q} \sqrt{J_k} \|\mathbf{w}_q^{(k)}\|_2 = \lambda_G \sum_{k,q} \sqrt{J_k} \Big(\sum_{j=1}^{J_k} (w_{jq}^{(k)})^2 \Big)^{1/2} \\ \leq \frac{\lambda_G}{2} \sum_{k,q} \frac{\sqrt{J_k}}{2} \Big(\sum_{j=1}^{J_k} (\widetilde{w}_{jq}^{(k)})^2 \Big)^{1/2} \\ + \frac{\lambda_G}{2} \sum_{k,q} \frac{\sqrt{J_k}}{2} \Big(\sum_{j=1}^{J_k} (\widetilde{w}_{jq}^{(k)})^2 \Big)^{-1/2} \sum_{j=1}^{J_k} (w_{jq}^{(k)})^2 \\ = \frac{\lambda_G}{2} \sum_{k,q} (\mathbf{w}_q^{(k)})^T \mathbf{D}_2^{(k,q)} \mathbf{w}_q^{(k)} + c,\end{aligned} \tag{3.11}$$

with $\mathbf{D}_2^{(k,q)}$ being a diagonal matrix containing $\frac{\sqrt{J_k}}{2} (\sum_{j=1}^{J_k} (\widetilde{w}_{jq}^{(k)})^2)^{-1/2}$ on its diagonal for a given k and q. The sum of quadratic forms in the marjorizing function in (3.11) can be rewritten into one quadratic form by arranging the terms,

$$\frac{\lambda_G}{2} \sum_{k,q} (\mathbf{w}_q^{(k)})^T \mathbf{D}_2^{(k,q)} \mathbf{w}_q^{(k)} = \frac{\lambda_G}{2} \mathrm{vec}(\mathbf{W}_c)^T \mathbf{D}_2 \mathrm{vec}(\mathbf{W}_c) + c. \tag{3.12}$$

Lastly, we will majorize the QK elitist LASSO penalty terms,

$$\begin{aligned}\lambda_E \sum_{k,q} \|\mathbf{w}_q^{(k)}\|_{1,2} = \lambda_E \sum_{k,q} \Big(\sum_{j=1}^{J_k} |w_{jq}^{(k)}| \Big)^2 \\ \leq \lambda_E \sum_{q,k} \left(\Big(\sum_{j=1}^{J_k} |\widetilde{w}_{jq}^{(k)}| \Big) \sum_{j=1}^{J_k} \frac{(w_{jq}^{(k)})^2}{|\widetilde{w}_{jq}^{(k)}|} \right) \\ = \lambda_E \sum_{k,q} (\mathbf{w}_q^{(k)})^T \mathbf{D}_3^{(k,q)} \mathbf{w}_q^{(k)},\end{aligned} \tag{3.13}$$

with $\mathbf{D}_3^{(k,q)}$ being a diagonal matrix containing on its $\big(\sum_{j=1}^{J_k}|\widetilde{w}_{jq}^{(k)}|\big)\big(|\widetilde{w}_{jq}^{(k)}|\big)^{-1}$ diagonal for a given k and q. Equation (3.13) can be rewritten (the same was as Equation (3.12)) into $\lambda_E \text{vec}(\mathbf{W}_c)^T\mathbf{D}_2\text{vec}(\mathbf{W}_c)$ by arranging the terms correctly. Combining the above results we can majorize Equation (3.9) as follows,

$$\begin{aligned} L(\mathbf{W}_c,\mathbf{P}_c) =& \|\mathbf{X}_c - \mathbf{X}_c\mathbf{W}_c\mathbf{P}_c^T\|_2^2 + \lambda_L\|\mathbf{W}_c\|_1 + \lambda_R\|\mathbf{W}_c\|_2^2 \\ &+ \sum_{q,k}(\lambda_G\sqrt{J_k}\|\mathbf{w}_q^{(k)}\|_2 + \lambda_E\|\mathbf{w}_q^{(k)}\|_{1,2}) \\ &\leq \|\text{vec}(\mathbf{X}_c) - (\mathbf{P}_c \otimes \mathbf{X}_c)\text{vec}(\mathbf{W}_c)\|_2^2 \\ &+ \text{vec}(\mathbf{W}_c)^T\mathbf{D}_{sup}\text{vec}(\mathbf{W}_c) + c \\ &= Q(\mathbf{W}_c,\mathbf{P}_c), \end{aligned} \tag{3.14}$$

with $\mathbf{D}_{sup} = \frac{\lambda_L}{2}\mathbf{D}_1 + \frac{\lambda_G}{2}\mathbf{D}_2 + \lambda_E\mathbf{D}_3 + \lambda_R\mathbf{I}$. Because $Q(\mathbf{W}_c,\mathbf{P}_c)$ is a quadratic function which can be easily minimized by taking the partial derivatives with respect to the elements to $\text{vec}(\mathbf{W}_c)$ and setting them to zero, also see Van Deun et al. (2011), doing this gives us the following estimates for $\text{vec}(\mathbf{W}_c)$,

$$\text{vec}(\widehat{\mathbf{W}}_c) = (\mathbf{D}_{sup} + \mathbf{I}\otimes\mathbf{X}_c^T\mathbf{X}_c)^{-1}\text{vec}(\mathbf{X}_c^T\mathbf{X}_c\mathbf{P}_c), \tag{3.15}$$

with $\mathbf{I}$ being a $Q\times Q$ identity matrix. Estimates for $\text{vec}(\widehat{\mathbf{W}}_c)$ can be found relatively efficiently by making use of the block diagonality of $(\mathbf{D}_{sup} + \mathbf{I}\otimes\mathbf{X}_c^T\mathbf{X}_c)$, meaning that the weights can be estimated per component separately,

$$\widehat{\mathbf{w}}_q = (\mathbf{D}_{sup}^{(q)} + \mathbf{X}_c^T\mathbf{X}_c)^{-1}\mathbf{a}_q, \tag{3.16}$$

with $\widehat{\mathbf{w}}_q$ denoting the estimates of the qth component, $\mathbf{D}_{sup}^{(q)}$ denoting the part of $\mathbf{D}_{sup}$ corresponding to the qth component and $\mathbf{a}_q$ denoting the qth column of $\mathbf{A} = \mathbf{X}_c^T\mathbf{X}_c\mathbf{P}_c$. The computation of the inverse in Equation (3.16) can be costly if $\sum J_k$ is large. To make the algorithm well suited to handle a large number of variables, we can implement a coordinate descent procedure to solve for $\mathbf{W}_c$ in $Q(\mathbf{W}_c,\mathbf{P}_c)$. For the ease of notation we will drop subscript c and let $j = 1\ldots Q\sum J_k$. Then, the update for an element of $\text{vec}(\mathbf{W}_c)$ is given by,

$$\text{vec}(\widehat{\mathbf{W}})_j := \frac{(\mathbf{P}\otimes\mathbf{X})_j^T\text{vec}(\mathbf{X}) - (\mathbf{P}\otimes\mathbf{X})_j^T(\mathbf{P}\otimes\mathbf{X})_{-j}\text{vec}(\mathbf{W})_{-j}}{(\mathbf{P}\otimes\mathbf{X})_j^T(\mathbf{P}\otimes\mathbf{X})_j + D_{jj}}, \tag{3.17}$$

where subscript j denotes the jth element of a vector or the jth column of a matrix and $-j$ denotes the object minus the jth element or column. Making use of the orthogonality of $\mathbf{P}$ and with $j = 1 \dots \sum J_k$ this simplifies to,

$$\widehat{w}_{jq} := \frac{\mathbf{p}_q^T \mathbf{X}^T \mathbf{x}_j - \mathbf{x}_j^T \mathbf{X}_{-j} \mathbf{w}_{-jq}}{\mathbf{x}_j^T \mathbf{x}_j + D_{jj}^{(q)}}. \tag{3.18}$$

With these derivations the estimation of $\mathbf{W}_c$ can be summarized in Algorithm 3.

Although different regularizers are implemented in algorithm 3, it is not advised to combine them all together. For example, it is not advised to combine the group LASSO and the elitist LASSO as they have opposing goals. A use case for the elitist LASSO is when common components have to be extracted; this is imposing zeros on each block in such a way that for each block segment also non-zero component weights remain.

Algorithm 3: MM algorithm for sparse SCA

```
1  sparse SCA ($\mathbf{X}_c, Q, \lambda$);
   Input  : $\mathbf{X}_c, Q$, initialize $\widehat{\mathbf{W}}_c$ with the right singular vectors of $\mathbf{X}_c$,
            or a random initialization
   Output: $\widehat{\mathbf{W}}_c$
2  while $\Delta$lossfunction value $> \epsilon$ do
3      $\widehat{\mathbf{P}}_c \leftarrow$ procruste rotation($\mathbf{X}_c, \widehat{\mathbf{W}}_c$)
4      for $q \leftarrow 1$ to $Q$ do
5          if $\sum J_k >> N$ then
6              for $j \leftarrow 1$ to $\sum J_k$ do
7                  $\widehat{w}_{jq} \leftarrow$ Eq. (3.18)
8              end
9          else
10             $\widehat{\mathbf{w}}_q \leftarrow$ Eq. (3.16)
11     end
12 end
13 return $\widehat{\mathbf{W}}_c$;
```

3.7.2 Data generation

Single block The data for the simulation study was generated from the following model,

$$\mathbf{X} = \mathbf{X} \mathbf{W} \mathbf{P}^T, \tag{3.19}$$

where $\mathbf{W}$ is $J \times J$, $\mathbf{W}^T\mathbf{W} = \mathbf{I}$ and $\mathbf{W} = \mathbf{P}$. $\mathbf{W}$ is manipulated such that it contains a given level of sparsity. To achieve this, we make use of an iterative procedure that proceeds as follows. First, a random $\mathbf{W}$ matrix is generated with zero weights in the desired places. After this step orthogonality of the columns is attempted by applying the gramm-schmidt orthogonalization procedure *only* on the intersection of the non-zero weights between two columns of $\mathbf{W}$. When $\mathbf{W}$ only has sets of columns that contain non-overlapping sparsity patterns, this immediately results into orthogonal columns, but when the columns in $\mathbf{W}$ have overlapping sparsity patterns the procedure will not always lead to $\mathbf{W}^T\mathbf{W} = \mathbf{I}$ on the first pass. In such cases multiple passes are needed in order to achieve orthogonality (additional coefficients might need to be put to zero). Some sparsity patterns are not possible, for example, an initialization where $\mathbf{W}$ does not have full column rank, or an initial set that degenerates to a linearly dependent set after multiple passes. In those cases the algorithm fails to converge.

After a suitable $\mathbf{W}$ has been obtained, $\boldsymbol{\Sigma}$ can be constructed by taking $\boldsymbol{\Sigma} = \mathbf{W}\boldsymbol{\Lambda}\mathbf{W}^T$. Here, $\boldsymbol{\Lambda}$ is a diagonal matrix with eigenvalues of the J components underlying the full decomposition. We specify these eigenvalues such that the first Q components account for a set amount of structural variance and the remaining eigenvalues for a set amount of noise variance. The data matrices $\mathbf{X}$ having a desired underlying sparse structure and noise level can then be obtained by sampling from the multivariate normal distribution using $\boldsymbol{\Sigma}$ and a zero mean vector.

Multi-block The data generation for the multi-block simulation study is the same as the data generation in the single set simulation study. Except that the data have been generated with two distinctive components and one common component. We define a distinctive component as being a linear combination of variables from a particular data block and a common component as a linear combination of all data blocks. In order to achieve the desired common and distinctive structure, full block segments of zeros are inserted in the $\mathbf{W}$ matrix.

Chapter 4

Cardinality constrained weight based PCA

Abstract

Principal component analysis (PCA) (Jolliffe, 1986) is a widely used analysis technique for dimension reduction. The components resulting from a PCA are linear combinations of all variables. This can make their interpretation difficult, especially when the number of variables in a data set is large. To that end, so-called regularized sparse PCA methods have been developed that aim at reducing the number of non-zero coefficients by relying on shrinkage penalties implying that only a subset of the variables makes up the components. The problem that shrinkage methods solve is not that of finding the best subset of variables optimizing the PCA criterion. In this paper we present cardinality constraint PCA (CCPCA) to solve the best subset PCA problem by fixing the number of non-zero coefficients through a cardinality constraint approach. For this purpose, we propose using the cardinality constraints regression algorithm from Adachi and Kiers (2017) and Bertsimas et al. (2016). Consistent with results obtained for regression analysis, we found that CCPCA outperforms sparse PCA based on shrinkage penalties (Zou et al., 2006) when noise levels are low but not when noise levels are high.

Keywords: Sparse PCA, Cardinality Constraint Regression, Best Subset Selection

Niek C. de Schipper, Anya Tonne & Katrijn Van Deun. *To be submitted.*

4.1 Introduction

Principal component analysis (PCA) (Jolliffe, 1986) is a widely used analysis technique for dimension reduction. The components resulting from PCA are linear combinations of all variables. This can make their interpretation difficult, especially when the number of variables in a data set is large. To that end so-called sparse PCA methods have been developed that aim at reducing the number of non-zero coefficients in the model, implying that only a subset of the variables make up the components. These include rotation techniques (such as varimax rotation (Kaiser, 1958)) and sparse PCA approaches that rely on shrinkage penalties (e.g., the least absolute shrinkage and selection operator or LASSO penalty; Tibshirani (1994)). Such shrinkage penalties steer the PCA coefficients towards (exact) zero. Penalized PCA approaches with sparsity inducing penalties have been extensively developed during the last decades (see for example; Zou et al. (2006); Shen and Huang (2008); Gu and Van Deun (2016); Van Deun et al. (2009) and are a highly popular tool for the analysis of multivariate data.

The problem that shrinkage methods solve is not that of finding the best subset of variables optimizing the PCA criterion. As pointed out in the discussion of Tibshirani (2011) solutions based on the LASSO will, unless under very stringent conditions, not return the best subset. Why then, one may wonder, isn't the best subset problem solved? A first reason for relying on shrinkage methods is of a computational nature: The best subset problem quickly becomes intractable as it implies trying out all possible subsets; approximate methods, e.g., based on convex relaxations such as the LASSO penalty, then offer a computationally attractive alternative (Tibshirani, 2011). A second reason, already put forward in the context of the nonnegative garrotte penalty (Breiman, 1995), is of a statistical nature: shrinkage of the non-zero coefficients avoids inflation and hence a better bias-variance trade-off of the estimators.

Notwithstanding the computational and statistical advantages of shrinkage penalties, recently significant progress has been made with respect to solving — to (near) optimality — the best subset regression problem for a large numbers of variables (in the order of a few thousands; Bertsimas et al. (2016)). The resulting algorithmic procedure providing near-optimal solutions has been also presented in Adachi and Kiers (2017). This opens the venue to develop a best subset sparse PCA method and to subsequently study the properties of the penalized and constrained approaches in terms of correctness of selection and the bias-variance trade-off.

In this paper, we propose to solve the best subset PCA problem by fixing the number of non-zero coefficients through a cardinality constraint approach. Furthermore we will compare this cardinality constrained PCA to sparse PCA (Zou et al., 2006) making use of shrinkage penalties.

The remainder of this paper is structured as follows: First, we discuss the notation, PCA and penalized sparse PCA, and introduce cardinality constrained PCA. Second, we assess the performance of cardinality constrained PCA in a simulation study and compare it with the performance of penalized sparse PCA. We end with a conclusion.

4.2 Methods

Given is a data matrix $\mathbf{X}$ that contains the scores for $i = 1 \ldots I$ observations on $j = 1 \ldots J$ variables; we follow the convention to present the J variable scores of observation i in row i and thus $\mathbf{X}$ has size $I \times J$. PCA decomposes the data into Q components as follows,

$$\begin{aligned} &\mathbf{X} = \mathbf{XWP}^T + \mathbf{E} \\ &\text{subject to } \mathbf{P}^T\mathbf{P} = \mathbf{I}, \end{aligned} \tag{4.1}$$

where $\mathbf{W}$ is a $J \times Q$ component weights matrix, $\mathbf{P}$ is a $J \times Q$ loadings matrix and $\mathbf{E}$ is the $I \times J$ matrix of residuals. The component weights matrix $\mathbf{W}$ will be the focus of this paper, where it should be noted that the linear combinations $\mathbf{XW}$ represent the component scores. In PCA the matrices $\mathbf{W}$ and $\mathbf{P}$ — given the number of components Q — can be obtained by applying the singular value decomposition (SVD) to $\mathbf{X}$ (Hastie et al., 2009b, p. 535). The resulting solution is optimal both in the least squares sense, i.e. minimizing

$$\|\mathbf{X} - \mathbf{XWP}^T\|_2^2 = \sum_{i,j} e_{ij}^2, \tag{4.2}$$

as well as in terms of maximizing the variance of the components. In general, the estimated component weights $\widehat{\mathbf{W}}$ resulting from the SVD will contain $J \times Q$ non-zero component weights. A disadvantage of PCA is that all estimates are non-zero, making the interpretation of the components difficult when J is large; there are simply too many weights to inspect.

To get interpretable component weights, numerous sparse PCA methods have been developed. In these methods components are estimated that are least-squares optimal or have maximal variance yet subject to penalties or constraints that put weights to zero, greatly increasing the interpretability of the components. Finding the best subset of weights that should be non-zero is a NP-hard problem (Natarajan, 1995) and is considered to be computationally intractable when JQ is large. To circumvent the NP-hard problem the usual approach taken in the literature is to relax the problem by relying on (convex) shrinkage penalties with variable selection properties (such as the LASSO penalty). Instead, here we will propose a sparse PCA problem that solves the cardinality constrained problem to near-optimality and with great computational efficiency.

In this section, we first introduce the standard approach to sparse PCA, this is solving the penalized PCA problem. Next, we introduce a procedure to solve the cardinality constrained PCA problem as this has not been proposed previously.

4.2.1 Sparse PCA with the elastic net penalty by Zou et al., 2006

A well known sparse PCA method that relies on the LASSO and ridge penalties has been developed by Zou et al. (2006). It solves the following objective function:

$$\begin{aligned} &\underset{\mathbf{W},\mathbf{P}}{\arg\min}\, L(\mathbf{W},\mathbf{P}) = \|\mathbf{X} - \mathbf{X}\mathbf{W}\mathbf{P}^T\|_2^2 + \lambda_L\|\mathbf{W}\|_1 + \lambda_R\|\mathbf{W}\|_2^2 \\ &\text{subject to } \mathbf{P}^T\mathbf{P} = \mathbf{I}, \end{aligned} \tag{4.3}$$

with $\|\mathbf{W}\|_1 = \sum_{j,q}|w_{jq}|$ the LASSO penalty (tuned by $\lambda_L \geq 0$) and $\|\mathbf{W}\|_2^2 = \sum_{j,q} w_{jq}^2$ the ridge penalty (tuned by $\lambda_R \geq 0$). The combination of the LASSO and the ridge is called the elastic net. The LASSO not only shrinks the component weights to zero but sets some of them exactly to zero. The ridge only shrinks the coefficients and is included to regularize the problem in the high-dimensional setting ($J > I$); e.g., to allow for more non-zero coefficients than variables, see also Zou et al. (2006).

The problem formulated in (4.3) is not that of finding the subset of variables with smallest loss as defined in (4.2). As discussed in the literature on penalized regression, the correct subset is only recovered under stringent conditions (see Bertsimas et al. (2016) and references therein). On the other hand, shrinkage of all coefficients reduces the variance of the estimated coefficients; hence the coefficients estimated under a penalized regime may be more accurate than those obtained with best subset selection Hastie et al. (2017).

Another drawback associated to the penalized sparse PCA approach is the limited control over the tuning parameter λ_L. In practice, one would like to tune the parameter such that it results in a given number of zero coefficients or, when model selection is of interest, such that a sequence of solutions is obtained with a controlled range and step size for the number of non-zero coefficients. The effect of the tuning parameter on the number of non-zero coefficients is not clear, hence the difficulty in tuning. An exception is formed when $\mathbf{W}$ is estimated with the least-angle regression (LARS) algorithm (Tibshirani et al., 2004). The LARS algorithm provides a piecewise linear solution path, which makes it straightforward to choose a model containing exactly the number of desired non-zero coefficients. Sparse PCA estimated with the LARS algorithm is available in the elasticnet package (Zou and Hastie, 2018) in R (R Core Team, 2020).

4.2.2 Sparse PCA with cardinality constraints

We propose to directly solve the best subset selection method for PCA and therefore solve the following cardinality constrained problem:

$$\begin{aligned} &\underset{\mathbf{W},\mathbf{P}}{\arg\min} \|\mathbf{X} - \mathbf{X}\mathbf{W}\mathbf{P}^T\|_2^2 \\ &\text{subject to } \mathbf{P}^T\mathbf{P} = \mathbf{I} \text{ and } \|\mathbf{W}\|_0 = K, \end{aligned} \tag{4.4}$$

with $\|\mathbf{W}\|_0$ counting the number of non-zero coefficients in $\mathbf{W}$. Hence zeros are introduced into the solution without the additional shrinkage of the non-zero coefficients. A practical advantage of the Cardinality Constrained PCA (CCPCA) problem in (4.4) is that there is direct control over K and thus the exact level of sparsity in $\mathbf{W}$. The cardinality constraints make it straightforward to pick a range of integers from which a suitable K can be selected.

Algorithm It is not our aim to find the global optimum of the CCPCA problem as this is an intractable problem for large J. Instead, we aim to reach a solution that is computationally feasible, also for large J and relying on procedures with proven good quality. The basic idea is to formulate the PCA problem as a regression problem, in line with the approach taken by Zou et al. (2006), and to rely on state-of-the-art optimization tools that have been proposed in the context of the best subset regression problem.

The parameters of Equation (4.4) are estimated using an alternating iterative procedure, where $\mathbf{W}$ is updated conditional upon the current estimate for $\mathbf{P}$ and vice versa. The problem of solving for $\mathbf{P}$ given $\mathbf{W}$ is a standard problem, known as the reduced rank Procrustes rotation problem, which has a closed form expression; we refer the interested reader to the appendix for the details. The conditional problem of solving for the component weight matrix $\mathbf{W}$ takes the form of a cardinality constrained regression problem (Adachi and Kiers (2017), see also Algorithm (1) in Bertsimas et al. (2016)):

$$\begin{aligned} &\underset{\mathbf{W}}{\arg\min} \|\text{vec}(\mathbf{X}) - (\mathbf{P} \otimes \mathbf{X})\text{vec}(\mathbf{W})\|_2^2 \\ &\text{subject to } \|\text{vec}(\mathbf{W})\|_0 = K. \end{aligned} \tag{4.5}$$

A numerical procedure that solves this problem was proposed by Adachi and Kiers (2017) as a special case of a majorize-minimize (Hunter and Lange, 2004) or iterative majorization (Kiers, 2002) procedure. The same the procedure has been proposed as a projected gradient descent algorithm Bertsimas et al. (2016). Cardinality constrained regression iteratively finds estimates for $\mathbf{W}$ by minimizing a majorizing function for Equation (4.4), which enables the inclusion of cardinality constraints. Given a fixed $\mathbf{P}$ consecutive updates for $\mathbf{W}$ are given by:

$$\begin{aligned}&\text{vec}(\mathbf{W}) := \text{vec}(\widetilde{\mathbf{W}}) - \alpha^{-1}((\mathbf{I}_Q \otimes \mathbf{X}^T\mathbf{X})\text{vec}(\widetilde{\mathbf{W}}) - \text{vec}(\mathbf{X}^T\mathbf{X}\mathbf{P}))\\ &\text{with the smallest } K \text{ absolute values set to zero.}\end{aligned} \tag{4.6}$$

with $\widetilde{\mathbf{W}}$ being the current estimates of $\mathbf{W}$ and α is the maximum eigen value of $\mathbf{X}^T\mathbf{X}$. Further details can be found in Section (4.5.1), also on how to fix the cardinality per component instead of over all Q components. Hence an approximate solution to the best subset selection problem is found relying on a procedure where the main complexity is to sort a $J \times Q$ matrix. In Bertsimas et al. (2016) the cardinality constrained regression problem is solved by first estimating the regression coefficients using (4.6) (see Algorithm (1) in Bertsimas et al. (2016)) and then using these regression coefficients as warm starts in a mixed integer optimization (MIO) program. This further improves the solution. Note that we do not perform the MIO step.

We call the full alternating procedure CCPCA; the details are outlined in Section (4.5.1). CCPCA converges to a stationary point yet is subject to local optima. We propose to initialize $\mathbf{W}$ with the estimates $\widehat{\mathbf{W}}$ from PCA (this is the Q right singular vectors associated to the Q largest singular values). Another option is using multiple starts: The procedure is started multiple times with different initializations of $\mathbf{W}$ and the estimates from the analysis with the smallest loss function value are retained. Multiple starts are more costly, which especially adds up when K and Q still need to be determined using computationally intensive model selection procedures such as cross-validation.

Summary We presented two methods for obtaining sparse PCA solutions, one based on penalizing the PCA problem and a novel one based on solving the cardinality constraint or best subset PCA problem to near optimality. Penalized approaches have been put forward in the statistical literature for reasons of computational and statistical efficiency. A main drawback is that they in general do not result in the best subset. Recently, significant progress has been made in optimizing the best subset selection problem such that solving the problem has become feasible for a large number of variables. Bertsimas et al. (2016) found cardinality constrained regression to be superior to LASSO regression not only in terms of recovering the correct subset of variables but also in terms of predictive performance, which is contrary to expectations based on the bias-variance trade-off. As a response Hastie et al. (2017) extended the simulations of (Bertsimas et al., 2016) focussing on prediction accuracy and found that cardinality constrained regression only outperforms the LASSO when there is high signal to noise ratio. Here we will study the properties of the penalized and cardinality constrained approaches in the context of sparse PCA. In the next section we will assess and compare the performance of CCPCA and sparse PCA with LARS in a simulation study; we will focus both on the recovery of the true subset of variables and the efficiency of the weights.

4.3 Simulation Study

To assess the performance of CCPCA we performed a simulation study, in this simulation study the performance of CCPCA is compared with the performance of the existing alternative: Penalized sparse PCA estimated with the LARS algorithm. Two types of performance are of interest: is the correct subset of variables recovered and are the estimates efficient (i.e. do they have lower mean squared error of estimation)? This will be addressed in two separate simulation studies.

In the simulation studies, we kept the number of variables fixed to $J = 50$ and the number of components to $Q = 3$. The study included the following design factors:

- The number of observation units I: 25, 50 and 100.
- The level of sparseness (percentage of the, in total $JQ = 150$ weights that are equal to zero): 30% and 80%.
- The noise level: 5% and 20%. These noise levels are created by manipulating the eigenvalues in the covariance matrix $\mathbf{\Sigma}$, in such a way that the Q signal components account for 95% and 80% of the total variance.

The design is fully crossed, resulting in $3 \times 2 \times 2 = 12$ design cells. For each design cell, 50 models were generated from which we simulated data. The generation of the data is detailed in Section (4.5.2). The resulting data were analyzed using CCPCA with Algorithm (4) programmed in the `R` software for statistical computing (R Core Team, 2020) and sparse PCA with LARS using the elasticnet package (Zou and Hastie, 2018). A single rational start, based on the SVD decomposition of the data, is used. To estimate $\mathbf{W}$ from the generated data sets with CCPCA and sparse PCA we used oracle information: We supplied the analysis with the true number components $q = 3$ and of non-zero weights per column of $\mathbf{W}$. The ridge penalty for sparse PCA was left at the default value of 10^{-6}. As the CCPCA and sparse PCA solutions are indeterminate with respect to the sign and order of the component weight vectors $\mathbf{w}_q$, we match $\widehat{\mathbf{w}}_q$ to the correct $\mathbf{w}_q$ based on the highest cosine similarity taking the sign into account.

In the first simulation study, our interest is mainly in the selection of the correct subset of variables. For this performance criterion, we expect CCPCA to outperform the penalized sparse PCA approach as finding the best subset is the objective of CCPCA but not of penalized sparse PCA. We will also assess the overall quality of estimation based on the Tucker congruence coefficient (Lorenzo-Seva and ten Berge, 2006) between $\widehat{\mathbf{W}}$ and $\mathbf{W}$, which equals the cosine of the angle between two vectors. Typically, a Tucker congruence coefficient above 0.85 is used as an indication of fair vector similarity.

In the second simulation study we will inspect the bias, variance, and mean squared error (MSE) of the estimators from both procedures. In this simulation study we randomly generated one model per condition, from which we created 5000 replicated data sets. We know sparse PCA introduces bias into its weights because of the LASSO and ridge penalties. We expect that the estimates from CCPCA will be less biased compared to sparse PCA because there is no shrinkage of the weights. On the other hand, for noisier data, we expect more variance of the estimates obtained with CCPCA. Taken together, a better bias-variance trade-off may be observed for penalized sparse PCA in the (very) noisy conditions.

The estimators of bias, variance and MSE we will use in the simulation study are slight modifications of the "one parameter" versions because we want to summarize the results of all JQ weights in $\mathbf{W}$ together. To assess the bias we will use the mean absolute bias (MAB) (Willmott and Matsuura, 2006) which is given by,

$$\text{MAB} = \frac{1}{JQ}\sum_j\sum_q |\overline{\widehat{w}}_{jq} - w_{jq}|, \tag{4.7}$$

where $\overline{\widehat{w}}_{jq}$ is the average estimated component weight $\frac{1}{R}\sum_r \widehat{w}_{jq}^{(r)}$ for the $r = 1 \ldots R$ replicated data sets. To quantify the variance of the estimators, we will use the mean variance of all estimates, given by,

$$\text{mean VAR} = \frac{1}{JQ}\sum_j\sum_q \frac{1}{R}\sum_r (\overline{\widehat{w}}_{jq} - \widehat{w}_{jq}^{(r)})^2. \tag{4.8}$$

To quantify the MSE, we will use the mean MSE of all estimates,

$$\text{mean MSE} = \frac{1}{JQ}\sum_j\sum_q \frac{1}{R}\sum_r (w_{jq} - \widehat{w}_{jq}^{(r)})^2. \tag{4.9}$$

4.3.1 Overall quality of the estimation of the weights

The Tucker congruence coefficients of CCPCA and sparse PCA are given in Figure (4.1). Both and CCPCA and sparse PCA tend not to do so well when the sparsity is low (30%), irrespective of the number of objects and the level of noise: Most Tucker congruence coefficients are below the acceptable threshold of 0.85. With high sparsity (80%) CCPCA shows acceptable performance, while sparse PCA only shows acceptable performance in the condition with 20% noise, and shows poor performance in the condition with 5% noise. In general CCPCA is showing slightly higher Tucker congruence coefficients for all conditions indicating a slightly better estimation overall. Concluding, if the data generating $\mathbf{W}$ is sparse, component weights can be estimated back fairly well by both methods, where the edge is given to CCPCA.

Whether CCPCA and sparse PCA identified the correct state of the weights (non-zero vs zero weights) is summarized in Figure (4.2). In correspondence to our expectations, Figure (4.2) shows that in all conditions CCPCA has a slightly higher proportion of correctly identified weights compared to sparse PCA. In all conditions the identification of the correct state of the weights is substantially higher compared to random performance (this is the expected number of hits given that the number of non-zero weights is known beforehand). In case the underlying model is sparse and the sample size is large enough, the correct state of the weights can be identified rather well (above 80% percent approximately). The proportion of correctly identified weights drops as the model gets less sparse and the sample size decreases. The results of both procedures are both in line with the “bet on sparsity principal” (Hastie et al., 2009b), which states that you can better assume the truth is sparse and use a method that works well in that context, because if not, no method will be able to recover the underlying model without a large number of observations per parameter.

4.3.2 Mean absolute bias, mean variance & mean MSE of the weights

In Table (4.1) the MAB, mean variance and mean MSE of the estimators from CCPCA and sparse PCA are reported. Not surprisingly, the MAB of sparse PCA is higher than CCPCA (although by a small margin only): approximately 0.010. To give an indication of this difference, the average absolute weight from the simulation study is 0.112. The mean variance of the CCPCA weights compared to the mean variance of the sparse penalized PCA weights is approximately equal when there is little noise on the data (5%). In case of a larger level of noise (20%) the variance of penalized sparse PCA is lower than that of CCPCA; this is because

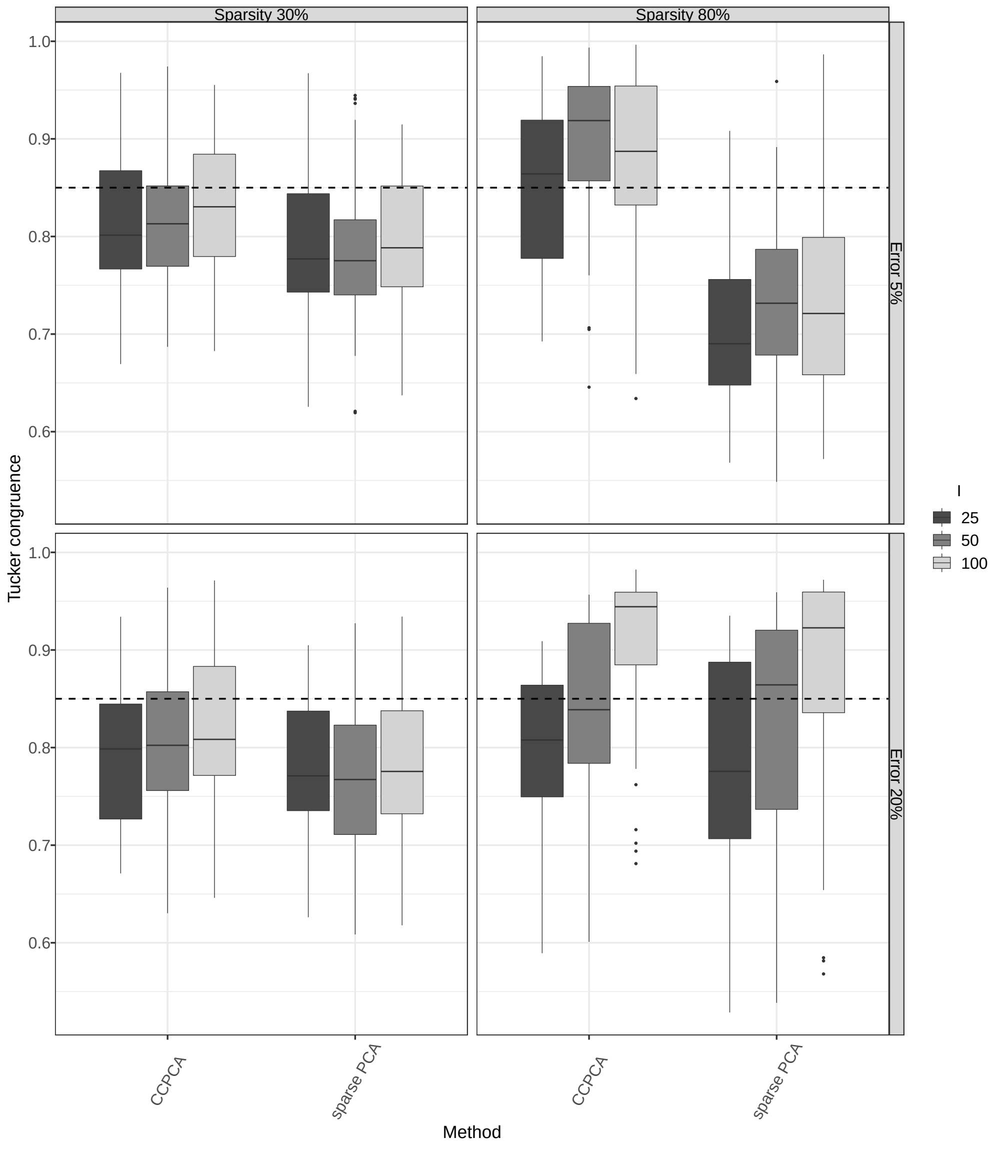


Figure 4.1. The Tucker congruence coefficient between $\mathbf{W}$ and $\widehat{\mathbf{W}}$ for the various model selection procedures. The dashed line indicates a threshold value of 0.85 indication fair similarity. In each condition 50 replicated data sets were used.

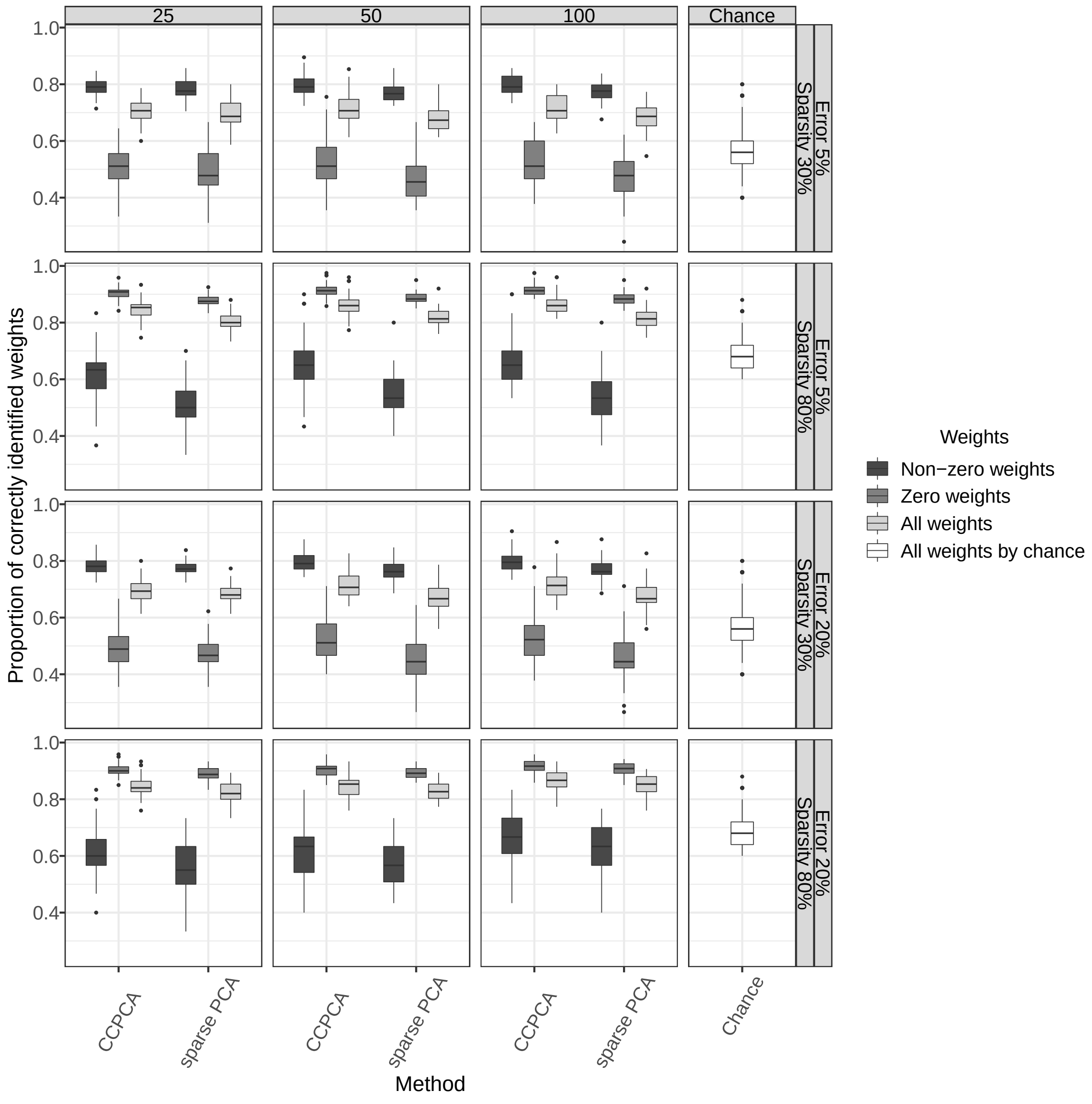


Figure 4.2. The proportion of correctly identified weights between **W** and $\widehat{\mathbf{W}}$. In each condition 50 replicated data sets were used. In the chance condition very low probablities for a certain amount of proportion correct are rounded to zero

of the shrinkage effect of the penalties in sparse PCA (bias variance trade-off). The mean MSE in case of small noise is slightly lower for CCPCA compared to sparse PCA, while the mean MSE is lower for sparse PCA in case of 20 percent noise and 80 percent sparsity. Concluding, overall CCPCA seems to do a little better in most of the simulation study, but we also see lower mean MSE for penalized sparse PCA in the condition with the highest noise level thanks to the bias variance trade-off. In general the differences between the two procedures are small.

Table 4.1

MAB, mean variance & mean MSE of the estimators from CCPCA and sparse PCA

		Error 5%				Error 20%			
		I = 25		I = 100		I = 25		I = 100	
		Sparsity 30%	Sparsity 80%	Sparsity 30%	Sparsity 80%	Sparsity 30%	Sparsity 80%	Sparsity 30%	Sparsity 80%
MAB	CCPCA	0.0172	0.0115	0.0177	0.0113	0.0186	0.0115	0.0181	0.0127
	Spca	0.0272	0.0221	0.0287	0.0242	0.0275	0.0219	0.0323	0.0179
mean Var	CCPCA	0.0066	0.0070	0.0065	0.0047	0.0072	0.0102	0.0071	0.0057
	Spca	0.0069	0.0084	0.0071	0.0072	0.0076	0.0062	0.0076	0.0039
mean MSE	CCPCA	0.0072	0.0076	0.0071	0.0053	0.0079	0.0108	0.0077	0.0064
	Spca	0.0081	0.0109	0.0084	0.0099	0.0090	0.0088	0.0092	0.0056

Note. The estimates are based on 5000 replicated data sets. For each condition a random model is chosen from which to estimate the bias, variance and MSE. The average absolute true component weight is 0.112.

4.4 Conclusion

In this paper we presented PCA with cardinality constraints on the component weights as an alternative to PCA with shrinkage penalties. We compared it to the alternative which is sparse PCA estimated with the LARS algorithm. In the first simulation study we found that CCPCA marginally outperforms sparse PCA with LARS when comparing the Tucker congruence coefficients between $\mathbf{W}$ and $\widehat{\mathbf{W}}$, and also in terms of the correct subset found of non-zero weights.

In the second simulation study we found that the MAB of CCPCA was lower compared to sparse PCA in all conditions, which is favourable when the variance of the weights is also low. This was the case in the condition with 5% noise. When the noise level is greater (20%) we observe that in some cases the MSE of the sparse PCA estimator is lower. As in Hastie et al. (2017), we expect that this difference will be greater when the noise levels increase.

Concluding, CCPCA and sparse PCA perform relatively similar, where CCPCA has the edge when the noise levels are low. When noise levels are large, the bias-variance trade-off can lead to better estimates for sparse PCA, but for our conditions these differences were marginal. It is important to mention that sparse

PCA with LARS is computationally more efficient than CCPCA. When $\mathbf{X}$ contains a large number of variables, sparse PCA might be preferred.

4.5 Appendix

4.5.1 Description of algorithm

In order to obtain the component weights, we need to optimize the following objective function with respect to $\mathbf{W}$ and $\mathbf{P}$,

$$\begin{aligned} &\underset{\mathbf{W},\mathbf{P}}{\arg\min}\, \|\mathbf{X} - \mathbf{X}\mathbf{W}\mathbf{P}^T\|_2^2 \\ &\text{subject to } \mathbf{P}^T\mathbf{P} = \mathbf{I} \text{ and } \|\mathbf{W}\|_0 = K, \end{aligned} \tag{4.10}$$

In order to get a minimum for (4.10), we can alternate between the estimation of $\mathbf{W}$ and $\mathbf{P}$. Given $\mathbf{W}$ we can estimate $\mathbf{P}$ by using Procruste rotation (ten Berge, 1993; Zou et al., 2006), $\mathbf{P} = \mathbf{U}\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are the left and right singular vectors of $\mathbf{X}^T\mathbf{X}\mathbf{W}$. Given $\mathbf{P}$ we can find estimates for $\mathbf{W}$ given the cardinality constraints using the cardinality constraint regression algorithm (Adachi and Kiers, 2017) which uses a majorization minimization approach, for a short overview see Hunter and Lange (2004). We can rewrite Equation (4.10) as a regression problem as follows,

$$\begin{aligned} &L(\mathbf{W}) = \|\text{vec}(\mathbf{X}) - (\mathbf{P} \otimes \mathbf{X})\text{vec}(\mathbf{W})\|_2^2 \\ &\text{subject to } \|\text{vec}(\mathbf{W})\|_0 = K, \end{aligned} \tag{4.11}$$

Following Kiers (2002) we can majorize Equation (4.11) as follows,

$$L(\mathbf{W}) \leq m(\mathbf{W}) = c + \alpha\|\mathbf{b} - \text{vec}(\mathbf{W})\|_2^2, \tag{4.12}$$

where c is a constant with respect to $\mathbf{W}$, α is the maximum eigen value of $\mathbf{X}^T\mathbf{X}$ and $\mathbf{b}$ is given by $\mathbf{b} = \text{vec}(\widetilde{\mathbf{W}}) - \alpha^{-1}(\mathbf{I}_Q \otimes \mathbf{X}^T\mathbf{X}\text{vec}(\widetilde{\mathbf{W}}) - \text{vec}(\mathbf{X}^T\mathbf{X}\mathbf{P}))$ with $\widetilde{\mathbf{W}}$ being the current estimates of $\mathbf{W}$. Equation (4.12) given the cardinality constraints of (4.10) is minimized when vec($\mathbf{W}$) is set to $\mathbf{b}$ with the smallest K absolute values set to zero. Equation (4.12) can be optimized repeatedly until convergence i.e. the value of $L(\mathbf{W})$ does not go down anymore. Note that the ridge penalty is not really needed here, since the problem of estimating $\mathbf{W}$ is only high-dimensional when $(\mathbf{P} \otimes \mathbf{X})$ is wide and not long. This is only the case when $Q > I$, which we argue is a very rare case.

It can be more useful to specify the cardinality constraints per column of $\mathbf{W}$ as opposed to the full matrix, which leads to more control over the sparsity level in the weights pertaining to specific components. This can be done as follows by imposing separate cardinality constraints for $\mathbf{w}_q$:

$$\begin{aligned} &L(\mathbf{W}) = \|\text{vec}(\mathbf{X}) - (\mathbf{P} \otimes \mathbf{X})\text{vec}(\mathbf{W})\|_2^2 \\ &\text{subject to } \|\mathbf{w}_q\|_0 = K_q \text{ for } q = 1 \dots Q, \end{aligned} \tag{4.13}$$

where K_q denotes the number of non-zero component weights per component. The updating formula per component becomes,

$$\begin{aligned} &\mathbf{w}_q := \widetilde{\mathbf{w}}_q - \alpha^{-1}(\mathbf{X}^T\mathbf{X}\widetilde{\mathbf{w}}_q - \mathbf{X}^T\mathbf{X}\mathbf{p}_q) \\ &\text{with the smallest } K_q \text{ absolute values set to zero.} \end{aligned} \tag{4.14}$$

Note that other constraints can also be freely applied. This leads to the full CCPCA Algorithm in (4).

Algorithm 4: CCPCA algorithm for sparse PCA

```
1 CCPCA PCA ($\mathbf{X}, Q, K_q \forall q$);
  Input  : $\mathbf{X}, Q, K_q \forall q$, initialize $\widehat{\mathbf{W}}$
  Output: $\widehat{\mathbf{W}}$
2 while $\Delta$ lossfunction value $> \epsilon$ do
3     $\widehat{\mathbf{P}} \leftarrow$ Procruste rotation($\mathbf{X}, \widehat{\mathbf{W}}$)
4     for $q \leftarrow 1$ to $Q$ do
5         $\mathbf{w}_q := \widetilde{\mathbf{w}}_q - \alpha^{-1}(\mathbf{X}^T\mathbf{X}\widetilde{\mathbf{w}}_q - \mathbf{X}^T\mathbf{X}\widehat{\mathbf{p}}_q)$
6         with the smallest $K_q$ absolute values set to zero.
7     end
8 end
9 return $\widehat{\mathbf{W}}$;
```

Algorithm (4) is freely available in R (R Core Team, 2020) and downloadable from github.com/trbKnl. This algorithm is susceptible to local minima as it is a non-convex optimization problem. A way of handling this problem is by initializing $\mathbf{W}$ with the estimates $\widehat{\mathbf{W}}$ from PCA. This "warm" start will nudge the analysis in the right direction, minimizing the risk that the algorithm will end up in a greater local minimum. Another safeguard against local minima is by using multiple starts. The procedure is started multiple times with different initializations of $\mathbf{W}$, and the estimates from the analysis with the smallest loss function value are retained. Multiple starts are more costly, which especially adds up when K and Q still need to be determined using model selection.

4.5.2 Data generation

The data for the simulation study was generated from the following model,

$$\mathbf{X} = \mathbf{X}\mathbf{W}\mathbf{P}^T, \tag{4.15}$$

where $\mathbf{W}$ is $J \times J$, $\mathbf{W}^T\mathbf{W} = \mathbf{I}$ and $\mathbf{W} = \mathbf{P}$. $\mathbf{W}$ is manipulated such that it contains a given level of sparsity. To achieve this, we make use of an iterative procedure that proceeds as follows. First, a random $\mathbf{W}$ matrix is generated with zero weights in the desired places. After this step orthogonality of the columns is attempted by applying the gramm-schmidt orthogonalization procedure *only* on the intersection of the non-zero weights between two columns of $\mathbf{W}$. When $\mathbf{W}$ only has sets of columns that contain non-overlapping sparsity patterns, this immediately results into orthogonal columns, but when the columns in $\mathbf{W}$ have overlapping sparsity patterns the procedure will not always lead to $\mathbf{W}^T\mathbf{W} = \mathbf{I}$ on the first pass. In such cases multiple passes are needed in order to achieve orthogonality (additional coefficients might need to be put to zero). Some sparsity patterns are not possible, for example, an initialization where $\mathbf{W}$ does not have full column rank, or an initial set that degenerates to a linearly dependent set after multiple passes. In those cases the algorithm fails to converge.

After a suitable $\mathbf{W}$ has been obtained, $\boldsymbol{\Sigma}$ can be constructed by taking $\boldsymbol{\Sigma} = \mathbf{W}\boldsymbol{\Lambda}\mathbf{W}^T$. Here, $\boldsymbol{\Lambda}$ is a diagonal matrix with eigenvalues of the J components underlying the full decomposition. We specify these eigenvalues such that the first Q components account for a set amount of structural variance and the remaining eigenvalues for a set amount of noise variance. The data matrices $\mathbf{X}$ having a desired underlying sparse structure and noise level can then be obtained by sampling from the multivariate normal distribution using $\boldsymbol{\Sigma}$ and a zero mean vector.

Chapter 5

sparseWeightBasedPCA: An R package for regularized weight based SCA and PCA

Abstract

In this chapter we introduce an R package to perform regularized simultaneous component analysis (SCA) and principal component analysis (PCA) with sparsity on the component weights. This package also includes model selection procedures. The procedures developed are based on recent work by de Schipper and Van Deun, described in Chapters 2, 3, 4. The main procedures of the package have been written in C++ using Rcpp (Eddelbuettel and François, 2011) and RcppArmadillo (Eddelbuettel and Sanderson, 2014) to provide maximal efficiency of the underlying numerical computations. In this chapter we introduce the reader to PCA and its multi-block extension SCA, followed by a description of the models that the procedures in this package estimate. Thereafter, the R implementation of the package is discussed followed by detailed examples of data analysis and model selection.

Keywords: SCA, PCA, R Package, Multi-Block Data

Niek C. de Schipper & Katrijn Van Deun. *To be submitted.*

5.1 Introduction

In this chapter we introduce an `R` package to perform regularized SCA and PCA with sparsity on the component weights. This package also includes model selection procedures. The procedures developed are based on recent work by de Schipper and Van Deun. The main procedures of the package have been written in `C++` using Rcpp (Eddelbuettel and François, 2011) and RcppArmadillo (Eddelbuettel and Sanderson, 2014) to provide maximal efficiency of the underlying numerical computations. In this chapter we will introduce the reader to PCA and its multi-block extension SCA, followed by a description of the models that the procedures in this package estimate. Thereafter, the `R` implementation of the package is discussed followed by detailed examples of data analysis and model selection.

5.2 Theoretical background

5.2.1 Principal Component Analysis

Principal component analysis (PCA) (Jolliffe, 1986) is a widely used analysis technique for data reduction. It can give crucial insights in the underlying structure of the data when used as a latent variable model.

Let $\mathbf{X}$ be a data matrix that contains the scores for $i = 1...I$ objects on $j = 1...J$ variables, where we follow the convention to present the J variable scores of observation i in row i, thus $\mathbf{X}$ has size $I \times J$. PCA decomposes the data into Q components as follows,

$$\begin{aligned} &\mathbf{X} = \mathbf{XWP}^T + \mathbf{E} \\ &\text{subject to } \mathbf{P}^T\mathbf{P} = \mathbf{I}, \end{aligned} \tag{5.1}$$

where $\mathbf{W}$ is a $J \times Q$ component weight matrix, $\mathbf{P}$ is a $J \times Q$ loading matrix and $\mathbf{E}$ is a $I \times J$ residual matrix. The component weight matrix $\mathbf{W}$ will be of main interest in this package, where it should be noted that $\mathbf{T} = \mathbf{XW}$ represent the component scores.

The advantage of inspecting the component weights instead of the loadings is that one can directly give meaning to $\mathbf{T}$, this because they express how the components are a weighted combination of the observed variables. The score on the qth component for object i is given by $t_{iq} = \sum_j w_{jq}x_{ij}$. Note that in Equation 5.1 the loadings are equal to the weights. This is not the case anymore when either the weights or loadings are not orthogonal anymore, for example, because penalties have been applied to them.

5.2.2 Simultaneous Component Analysis

The decomposition in (5.1) can be extended to the case of multi-block data by taking $\mathbf{X}_c = [\mathbf{X}_1 \ldots \mathbf{X}_K]$; this is concatenating the K data blocks composed of different sets of variables of size J_k for the same units of observation. The decomposition of $\mathbf{X}_c$ has the same block structured decomposition as in (5.1) with $\mathbf{W}_c = [\mathbf{W}_1^T \ldots \mathbf{W}_K^T]^T$ and $\mathbf{P}_c = [\mathbf{P}_1^T \ldots \mathbf{P}_K^T]^T$. This multi-block formulation of PCA is known as the simultaneous component model:

$$\begin{aligned}&[\mathbf{X}_1 \ldots \mathbf{X}_K] = [\mathbf{X}_1 \ldots \mathbf{X}_K][\mathbf{W}_1^T \ldots \mathbf{W}_K^T]^T[\mathbf{P}_1^T \ldots \mathbf{P}_K^T] + \mathbf{E}\\ &\text{subject to } [\mathbf{P}_1^T \ldots \mathbf{P}_K^T][\mathbf{P}_1^T \ldots \mathbf{P}_K^T]^T = \mathbf{I}.\end{aligned} \tag{5.2}$$

When analyzing multi-block data with simultaneous component analysis (SCA), identifying meaningful relations between data blocks is of prime interest. In order to gain insight in how multiple data blocks relate to each other, we can search for block-wise structures in the component weights that tell us whether a component is uniquely determined by variables from one single data block (distinctive component), or whether it is a component that is determined by variables from multiple data blocks (common component). In other words, a distinctive component is a linear combination of variables of a particular data block only, whereas a common component is a linear combination of variables of multiple data blocks. An example of common and distinctive components with two data blocks is given below. The first two components are distinctive components, while the third component is a common component, i.e.,

$$\mathbf{T} = \begin{bmatrix}\mathbf{X}_1 & \mathbf{X}_2\end{bmatrix}\begin{bmatrix}\mathbf{W}_1\\ \mathbf{W}_2\end{bmatrix} = \begin{bmatrix}\mathbf{X}_1 & \mathbf{X}_2\end{bmatrix}\begin{bmatrix}0 & w_{1_1 2} & w_{1_1 3}\\ 0 & w_{2_1 2} & w_{2_1 3}\\ 0 & w_{3_1 2} & w_{3_1 3}\\ w_{1_2 1} & 0 & w_{1_2 3}\\ w_{2_2 1} & 0 & w_{2_2 3}\\ w_{3_2 1} & 0 & w_{2_2 3}\end{bmatrix}.$$

5.2.3 Content of the `sparseWeightBasedPCA` package

The `sparseWeightBasedPCA` package provides functions that perform regularized PCA and SCA with regularization on the component weights. Furthermore the package will provide model selection procedures for the selection of the hyper parameters of the models. The core procedures of this package consist of the following functions:

1. `scads` Regularized SCA with common and distinctive component weights using constraints
2. `mmsca` Regularized SCA with common and distinctive component weights using the group LASSO
3. `ccpca` PCA with sparse component weights using cardinality constraints

Packages that provide similar functionality to the `sparseWeightBasedPCA` are: the elasticnet package (Zou and Hastie, 2018) which provides sparse PCA for the component weights with ridge and LASSO regularization, the regularized SCA package (Gu and Van Deun, 2018) which provides procedures for SCA/PCA with regularization on the loadings instead of the weights, sparse PCA using variable projection (Erichson et al., 2018) which also provides sparse PCA with regularization on the component weights with ridge and LASSO regularization including robust sparse PCA, the regularized PCA (Wang and Huang, 2017) for sparse pca with spatial data, the mixOmics package (Rohart et al., 2017) that provides a sparse PCA for the loadings with deflation based on work from Shen and Huang (2008) and the penalized matrix decomposition package (Witten et al., 2009) that provides a penalized matrix decomposition with an application for sparse PCA. Note that this list is not exhaustive.

Our package is different from the aforementioned packages in that it provides functionality for SCA in order to analyze multi-block data for high-dimensional data with regularization on the component weights, such that common and distinctive components can be revealed. It also provides procedures for model selection that can be applied to any sparse SCA/PCA procedure with regularization on the component weights.

5.3 Models of the `sparseWeightBasedPCA` package

5.3.1 Regularized SCA with sparse component weights using constraints

Here we present an approach of performing regularized SCA, with ridge and LASSO regularization and block wise constraints on $\mathbf{W}_c$ by solving,

$$\begin{aligned} L(\mathbf{W}_c, \mathbf{P}_c) =& \|\mathbf{X}_c - \mathbf{X}_c\mathbf{W}_c\mathbf{P}_c^T\|_2^2 + \lambda_L\|\mathbf{W}_c\|_1 + \lambda_R\|\mathbf{W}_c\|_2^2 \\ & \text{subject to } \mathbf{P}_c\mathbf{P}_c^T = \mathbf{I}, \\ & \text{and } \lambda_L, \lambda_R \geq 0 \text{ and zero block constraints on } \mathbf{W}_c. \end{aligned} \tag{5.3}$$

In order to get a minimum for (5.3) we alternate between the estimation of $\mathbf{W}_c$ and $\mathbf{P}_c$. Given $\mathbf{W}_c$ we can estimate $\mathbf{P}_c$ by using Procruste rotation (ten Berge, 1993; Zou et al., 2006), $\widehat{\mathbf{P}_c} = \mathbf{U}\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are the left and right singular vectors of $\mathbf{X}_c^T\mathbf{X}_c\widehat{\mathbf{W}}_c$. Given $\mathbf{P}_c$ we find estimates for $\mathbf{W}_c$ by using a coordinate descent algorithm that works by soft-thresholding the weights and allows to impose constraints (here: the block-wise zero constraints). For the specifics we refer the reader to the appendix of Chapter 2. This iterative procedure stops when an optimum has been found (i.e. the loss function value is not decreasing anymore beyond a pre-specified tolerance level). The optimization problem in (5.3) is non-convex meaning that there are local minima. In order to deal with local minima, multiple random starts can be used with different initializations of $\mathbf{W}_c$, the solution leading to the lowest evaluation of (5.3) is retained. Typically starting the algorithm with the SVD of the concatenated data (e.g. the first Q right singular vectors of $\mathbf{X}_c$) will lead to the smallest loss function value among the multiple starts.

The main advantage of analyzing multi-block data by using this procedure is that it is fast and scalable to large data sets thanks to the coordinate descent implementation. The inclusion of the block-wise constraints on $\mathbf{W}_c$ make sure common and distinctive components are found. The ridge and LASSO regularizers are optional and facilitate extra sparsity within the component weights. A disadvantage of the method is that the common and distinctive structure for $\mathbf{W}_c$ is not known beforehand, thus it needs to be selected using model selection. This can be computationally demanding depending on the total number of common and distinctive structures that need to be assessed.

The procedure that finds minimizers to (5.3) has been implemented in the `scads` function. This function will be discussed in detail in the next section and examples will be given outlining the analysis including model selection.

5.3.2 Regularized SCA with sparse component weights using the group LASSO

Here we present a very flexible approach of performing regularized SCA using, ridge, LASSO, group LASSO and elitist LASSO regularization by solving:

$$\begin{aligned} L(\mathbf{W}_c, \mathbf{P}_c) =& \|\mathbf{X}_c - \mathbf{X}_c\mathbf{W}_c\mathbf{P}_c^T\|_2^2 + \lambda_L\|\mathbf{W}_c\|_1 + \lambda_R\|\mathbf{W}_c\|_2^2 \\ &+ \sum_{q,k}(\lambda_G\sqrt{J_k}\|\mathbf{w}_q^{(k)}\|_2 + \lambda_E\|\mathbf{w}_q^{(k)}\|_{1,2}) \\ &\text{subject to } \mathbf{P}_c\mathbf{P}_c^T = \mathbf{I} \text{ and } \lambda_L, \lambda_R, \lambda_G, \lambda_E \geq 0 \end{aligned} \tag{5.4}$$

where $\mathbf{W}_c = [(\mathbf{W}^{(1)})^T \dots (\mathbf{W}^{(K)})^T]^T$, and $\mathbf{w}_q^{(k)}$ denotes the qth column from the submatrix $\mathbf{W}^{(k)}$. In order to get a minimum for (5.4) we alternate between the estimation of $\mathbf{W}_c$ and $\mathbf{P}_c$. Given $\mathbf{W}_c$ we can estimate $\mathbf{P}_c$ by using using Procruste rotation (ten Berge, 1993; Zou et al., 2006), $\widehat{\mathbf{P}_c} = \mathbf{U}\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are the left and right singular vectors of $\mathbf{X}_c^T\mathbf{X}_c\widehat{\mathbf{W}}_c$. Given $\mathbf{P}_c$ we can find estimates for $\mathbf{W}_c$ by using the majorization minimization (MM) algorithm. For the specific we refer the reader to the appendix of Chapter 3. This iterative procedure stops when an optimum has been found (i.e. the loss function value is not decreasing anymore beyond a pre-specified tolerance level). The optimization problem in (5.4) is non-convex and meaning there are local minima. In order to deal with that multiple random starts can be used with different initializations of $\mathbf{W}_c$, the start leading to the lowest evaluation of (5.4) is retained. Typically starting the algorithm with the SVD of the concatenated data (e.g. the first Q right singular vectors of $\mathbf{X}_c$) will lead to the smallest solution among the multiple starts.

The main advantage of analyzing multi-block data by using this procedure is that it can automatically look for common and distinctive components by taking advantage of the properties of the group LASSO. Because the group LASSO is specified on the colored segments (see below), it will either include a segment or put all the weights in a segment to zero, uncovering common and distinctive components. This is especially useful if the number of blocks and the number of components are substantial and an exhaustive comparison of all possible combinations of common and distinct components is computationally too demanding.

$$\mathbf{T} = \begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 \end{bmatrix} \begin{bmatrix} \mathbf{W}_1 \\ \mathbf{W}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 \end{bmatrix} \begin{bmatrix} w_{1_1 1} & w_{1_1 2} & w_{1_1 3} \\ w_{2_1 1} & w_{2_1 2} & w_{2_1 3} \\ w_{3_1 1} & w_{3_1 2} & w_{3_1 3} \\ w_{1_2 1} & w_{1_2 2} & w_{1_2 3} \\ w_{2_2 1} & w_{2_2 2} & w_{2_2 3} \\ w_{3_2 1} & w_{3_2 2} & w_{2_2 3} \end{bmatrix}$$

The inclusion of the LASSO and ridge penalties are optional and facilitate extra sparsity within the colored segments. The elitist LASSO has a very special use case, that is, it will include all colored segments and will put weights within each segment to zero. The elitist lasso can be used to force components to be common. It is not advised to use the group LASSO and the elitist LASSO together as they have opposing goals. A disadvantage of using this procedure is that is potentially slow, this because its implemented using a MM-algorithm which tends to be slow in convergence.

The MM-procedure has been implemented in the `mmsca` function. This function will be discussed in detail in the next section and examples will be given outlining the analysis including model selection.

5.3.3 PCA with sparse component weights using cardinality constraints

Here we present an approach of solving PCA by applying cardinality constraints to the component weights by solving:

$$\begin{aligned} &L(\mathbf{W},\mathbf{P}) = \|\mathbf{X} - \mathbf{X}\mathbf{W}\mathbf{P}^T\|_2^2 \\ &\text{subject to } \mathbf{W} \text{ including } K \text{ zeros}. \end{aligned} \tag{5.5}$$

In order to get a minimum for (5.5) we will alternate between the estimation of $\mathbf{W}$ and $\mathbf{P}$. Given $\mathbf{W}$ we can estimate $\mathbf{P}$ by using Procruste rotation (ten Berge, 1993; Zou et al., 2006), $\mathbf{P} = \mathbf{U}\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are the left and right singular vectors of $\mathbf{X}^T\mathbf{X}\widehat{\mathbf{W}}$. Given $\mathbf{P}$ we can find estimates for $\mathbf{W}$ given the cardinality constraints using the cardinality constraint regression algorithm; for details see the appendix of Chapter 4. The optimization problem in (5.5) is non-convex meaning there are local minima. In order to deal with that, multiple random starts can be used with different initializations of $\mathbf{W}$, the solution leading to the lowest evaluation of (5.5) is retained. Typically starting the algorithm with the SVD of the concatenated data (e.g. the first Q right singular vectors of $\mathbf{X}$) will lead to the smallest solution among the multiple starts.

The main advantage of solving (5.5) is that this model tries to directly tackle the problem of finding the underlying subset of weights, in contrast to the usage of a penalty that shrinks the weights and also induces sparsity such as the LASSO. This approach can lead to a better selection of the relevant variables and more precise estimates of the weights compared to LASSO (see Chapter 4). Another advantage is that one can directly impose cardinality constraints on $\mathbf{W}$. This gives the user control over the number of zero coefficients. This can be desirable if there is already an idea about the level of sparsity in the final model. A disadvantage of using this procedure is that is potentially slow, this because the cardinality constraint algorithm is an MM-algorithm which tends to be slow in convergence. Another potential downside could be the absence of penalties, since these tend to shrink the variance of the estimators, leading to more efficiency. In noisy situations other procedures, such as such as PCA penalized with the lasso might outperform this procedure.

This model has been implemented in the `ccpca` function. This function will be discussed in detail in the next section.

5.4 The implementation in R of the sparseWeightBasedPCA package

The sparseWeightBasedPCA package provides functions for the aforementioned models. Model selection procedures are also provided in order to tune the hyper parameters. The main functions, scads, mmsca and ccpca are implemented in C++ using the packages Rcpp (Eddelbuettel and François, 2011) and RcppArmadillo (Eddelbuettel and Sanderson, 2014). The model selection procedures are implemented in R. The functions of the package will be discussed after which detailed examples will be given.

5.4.1 Core estimation procedures

The main functions: scads, mmsca and ccpca are all implemented in a similar way. At the core they perform SCA/PCA with variations depending on the goals of the user. Here below we give a minimal working example of a typical scads analysis (note that analyses with mmsca and ccpca are performed in a very similar way). First we generate some data:

```r
J <- 30
I <- 100
X <- matrix(rnorm(I*J), I, J)
```

With these data scads can be run with supplied values for the minimal required arguments. In this example we perform PCA without constraints but with ridge and LASSO regularization

```r
ncomp <- 3 # The number of components to estimate
constraints <- matrix(1, J, ncomp) # No constraints
res <- scads(X,
             ncomp = ncomp,
             ridge = 10e-8,
             lasso = rep(1, ncomp),
             constraints = constraints,
             Wstart = matrix(1, J, ncomp),
             itr = 10e5)
```

scads, mmsca and ccpca return a list with the following elements:

- W A matrix containing the component weights
- P A matrix containing the loadings

- `loss` A numeric variable containing the minimum loss function value of all the `nStarts` starts
- `converged` A boolean containing `TRUE` if the algorithm converged, `FALSE` if the algorithm did not converge

Detailed examples of data analysis will follow in the next section or can be obtained within `R` by looking at the documentation (Execute `?scads` within `R`). The main functions `scads`, `mmsca` and `ccpca` have a slightly different set of arguments. See the following lists for the specifics.

Overview of the `scads` arguments

- `X` A data matrix of class `matrix`
- `ncomp` The number of components to estimate (an integer)
- `ridge` A numeric value containing the ridge parameter for ridge regularization on the component weight matrix `W`
- `lasso` A vector containing a lasso parameter for each column of `W` separately. To set the same lasso penalty for the component weights `W` specify: lasso = `rep(value, ncomp)`
- `constraints` A matrix of the same dimensions as the component weights matrix W (`ncol(X) x ncomp`). Entries in `W` can only be constrained to zero. This by entering a zero in the `constraints` matrix at the position where the zero is desired; non-zero entries are treated as unconstrained values and will be estimated during the `scads` procedure.
- `itr` The maximum number of iterations (an integer)
- `Wstart` A matrix of `ncomp` columns and `nrow(X)` rows with starting values for the component weight matrix `W`; if `Wstart` only contains zeros, a warm start is used: the first `ncomp` right singular vectors of `X`
- `tol` The convergence is determined by comparing the loss function value after each iteration, if the difference is smaller than `tol`, the analysis has converged. The default value is `10e-8`.
- `nStarts` The number of random starts the analysis should perform. The first start will be performed with the values given by `Wstart`. The consecutive starts will be `Wstart` plus a matrix with values generated from a uniform distribution times the current start number (the first start has index zero).
- `printLoss` A boolean: `TRUE` will print the loss function value after each 10th iteration.

Overview of the `mmsca` arguments

- `X` A data matrix of class `matrix`
- `ncomp` The number of components to estimate (an integer)

- `ridge` A vector containing a ridge parameter for each column of `W` separately. To set the same ridge penalty for the component weights W, specify: ridge = `rep(value, ncomp)`, value is a non-negative double
- `lasso` A vector containing a ridge parameter for each column of `W` separately. To set the same lasso penalty for the component weights W, specify: lasso = `rep(value, ncomp)`, value is a non-negative double
- `grouplasso` A vector containing a grouplasso parameter for each column of `W` separately. To set the same grouplasso penalty for the component weights W, specify: grouplasso = `rep(value, ncomp)`, value is a non-negative double
- `elitistlasso` A vector containing a elitistlasso parameter for each column of `W` separately. To set the same elitistlasso penalty for the component weights `W`, specify: elitistlasso = `rep(value, ncomp)`, value is a non-negative double
- `groups` A vector specifying the sizes of the blocks that make up `X`. Example: `c(10, 100, 1000)`. The first 10 variables belong to the first block, the 100 variables after belong to the second block, etc.
- `constraints` A matrix of the same dimensions as the component weights matrix W (`ncol(X) x ncomp`). Entries in `W` can only be constrained to zero. This by entering a zero in the `constraints` matrix at the position where the zero is desired; non-zero entries are treated as unconstrained values and will be estimated during the `scads` procedure.
- `itr` The maximum number of iterations (a positive integer)
- `Wstart` A matrix of `ncomp` columns and `nrow(X)` rows with starting values for the component weight matrix W, if `Wstart` only contains zeros, a warm start is used: the first `ncomp` right singular vectors of `X`
- `tol` The convergence is determined by comparing the loss function value after each iteration, if the difference is smaller than `tol`, the analysis has converged. Default value is `10e-8`
- `nStarts` The number of random starts the analysis should perform. The first start will be performed with the values given by `Wstart`. The consecutive starts will be `Wstart` plus a matrix with values generated from a uniform distribution times the current start number (the first start has index zero).
- `printLoss` A boolean: `TRUE` will print the loss function value after each 10th iteration.
- `coorDes` A boolean with the default `FALSE`. If coorDes is `FALSE` the estimation of the component weights `W` conditional on the loadings `P` will be estimated using a matrix inverse, which can be slow for large datasets. If set to `TRUE` the weights will be estimated using coordinate descent, in some cases coordinate descent will be faster
- `coorDesItr` An integer specifying the maximum number of iterations for the coordinate descent algorithm, the default is set to 1. You do not have to run

this algorithm until convergence before turning back to the estimation of the loadings. The tolerance for this algorithm is hard coded and set to 10^8.

Overview of the ccpca arguments

- X A data matrix of class matrix
- ncomp The number of components to estimate (an integer)
- nzeros A vector of length ncomp containing the number of desired zeros in the columns of the component weight matrix W
- itr The maximum number of iterations (an integer)
- Wstart A matrix of ncomp columns and nrow(X) rows with starting values for the component weight matrix W, if Wstart only contains zeros, a warm start is used: the first ncomp right singular vectors of X
- nStarts The number of random starts the analysis should perform. The first start will be performed with the values given by Wstart. The consecutive starts will be Wstart plus a matrix with values generated from a uniform distribution times the current start number (the first start has index zero).
- tol The convergence is determined by comparing the loss function value after each iteration, if the difference is smaller than tol the analysis has converged. The default value is 10e-8
- printLoss A boolean: TRUE will print the loss function value after each 10th iteration.

5.4.2 Model selection procedures

To execute scads, mmsca and ccpca, values for the hyper parameters need to selected. For example the number of components or values for the lasso argument. The sparseWeightBasedPCA package provides three procedures for selecting the hyper-parameters of the model: Cross-validation using the EigenVector method, the Index of Sparseness (IS) and the Bayesian Information Criterion (BIC); for details see Chapter 3. The procedures are implemented in the functions: CVforPCAwithSparseWeights, ISforPCAwithSparseWeights and BICforPCAwith-SparseWeights. They are parametrized as follows: First the arguments specific to the model selection function are entered, followed by a pointer to the function (In R that is the function name with no brackets) that does the analysis, followed by arguments to that function. An example is given here:

```r
J <- 30
I <- 100
X <- matrix(rnorm(I*J), I, J)
```

With this data, CVforPCAwithSparseWeights using scads works as follows,

```r
ncomp <- 3
res <- CVforPCAwithSparseWeights(X = X,
                          nrFolds = 10,
                          FUN = scads, # Pointer to scads()
                          ncomp = ncomp, # Followed by the arguments
                          ridge = 0,
                          lasso = rep(0.1, ncomp),
                          constraints = matrix(1, J, ncomp),
                          Wstart = matrix(0, J, ncomp),
                          itr = 10e5,
                          printLoss = FALSE)
```

This function returns a list with the following elements:

- `MSPE` The mean squared prediction error given the tuning parameters
- `MSPEstdError` The standard error of the `MSPE`
- `nNonZeroCoef` The number of non-zero coefficients in the model

Model selection of the hyper-parameters can be based on the model that results in the lowest mean squared prediction error, or a model can be picked with the least number of non-zero coefficients still within one standard error of the model with the lowest mean squared prediction error. The other functions work similar, see `?BICforPCAwithSparseWeights` and `?ISforPCAwithSparseWeights` for more detailed information.

5.4.3 Additional tuning functions for `mmsca`

The `sparseWeightBasedPCA` package provides more elaborate model selection functions for `mmsca`. Because of the flexibility of `mmsca` it can be overwhelming to use the aforementioned tuning functions. Therefore two additional functions for `mmsca` are provided: `mmscaModelSelection` and `mmscaHyperCubeSelection`. `mmscaModelSelection` uses a fixed grid of all combinations of the hyper-parameters to pick the best combination from, whereas `mmscaHyperCubeSelection` uses an adaptive grid that zooms in on a good combination of hyper-parameters until it converges on a certain combination. Note that `mmscaHyperCubeSelection` is experimental, it could potentially speed up the process of tuning enormously but it has not been scrutinized using a simulation study. A basic example of both is given here:

To perform model selection with `mmsca` using an exhaustive grid of all combinations of the tuning parameters the following can be done:

```
J <- 30
I <- 100
X <- matrix(rnorm(I*J), I, J)

out <- mmscaModelSelection(X,
        ridgeSeq = seq(0, 1, by = 0.1),
        lassoSeq = 0:100,
        grouplassoSeq = 0, # No group lasso and no elitist lasso
        elitistlassoSeq = 0,
        ncompSeq = 1:3,
        tuningMethod = "CV", # Indicate the tuning method
        nrFolds = 10,
        groups = ncol(X), # Arguments for mmsca()
        itr = 100000,
        nStart = 1,
        coorDes = FALSE,
        coorDesItr = 100,
        printProgress = TRUE)
```

This function returns a list with two elements:

- `results` A list with `ncomp` elements each containing the following elements
  - `"BIC, IS or MSPE"` The index chosen in tuningMethod for all combinations of ridge, lasso, grouplasso and elististlasso
  - `"bestBIC, bestIS, bestMSPE or bestMSPE1stdErrorRule"` The best index according to the chosen tuning method
  - `"nNonZeroCoef"` The number of nonzero weights in the best model
  - `"ridge"` The value of the ridge penalty corresponding to the best model
  - `"lasso"` The value of the lasso penalty corresponding to the best model
  - `"grouplasso"` The value of the group lasso penalty corresponding to the best model
  - `"elististlasso"` The value of the elitist lasso penalty corresponding to the best model
  - `"ncomp"` The number of component that was used for these items
  - `"ridge1stdErrorRule"` In case tuningMethod == "CV", the value of the ridge penalty according to the 1 standard error rule: the most sparse model within one standard error of the model with the lowest MSPE
  - `"lasso1stdErrorRule"` In case tuningMethod == "CV", the value of the lasso penalty according to the 1 standard error rule: the most sparse model within one standard error of the model with the lowest MSPE
  - `"grouplasso1stdErrorRule"` In case tuningMethod == "CV", the value of the group lasso penalty according to the 1 standard error rule: the

most sparse model within one standard error of the model with the lowest MSPE
  - "elitistlasso1stdErrorRule" In case tuningMethod == "CV", the value of the elitist lasso penalty according to the 1 standard error rule: the most sparse model within one standard error of the model with the lowest MSPE
  - "ridge1stdErrorRule" In case tuningMethod == "CV", the value of the ridge according to the 1 standard error rule: the most sparse model within one standard error of the model with the lowest MSPE
- `bestNcomp` The number of components with the best value for the chosen tuning index

We refer to the package documentation for more details (see `?mmscaModelSelection`). This procedure can be slow because the number of combinations for the tuning parameters can be large and it takes a lot of time to evaluate them all in order to pick the best one. To that end we also provide an alternative way of tuning using `mmscaHyperCubeSelection`. This function tunes a grid of the tuning parameters determined by the min and max of their corresponding sequences and a step size provided by the `stepsize` argument. It picks the best combination of that grid, and zooms in on that combination, by making a new, smaller grid around the previous best combination. This process continues until the average range of the sequences is less than `stopWhenRange` (a pre-specified stopping criterion). The new sequences are determined by taking the minimum value to be: best value - range, and maximum value by: best value + range, and a pre-specified step size in `stepsize`. In order for this procedure to work well, the grid needs to include an optimal combination of tuning parameters, and it needs a reasonable step size (at least 3, 5 is better, 2 is too small). This approach assumes that a local optimum of tuning parameters is good enough to get interpretable results. Note that this function is experimental and has not been scrutinized using a simulation study.

This procedure can be performed as follows,

```
out <- mmscaHyperCubeSelection(X,
            ncomp = 3, # Only a fixed number of components
            ridgeSeq = 0:3,
            lassoSeq = 0:10,
            grouplassoSeq = 0,
            elitistlassoSeq = 0,
            stepsize = 5, # Step size of the sequences
            logscale = FALSE, # Sequences not on the log-scale
            stopWhenRange = 0.01, # stop when average range is < 0.01
            method = "CV1stdError", # Tuning method
```

```
              groups = ncol(X), # Arguments for mmsca()
              nStart = 1,
              itr = 100000,
              printProgress = TRUE,
              coorDes = FALSE,
              coorDesItr = 1,
              tol = 10e-5,
              nrFolds = 10)
```

This function returns a list with the following elements:

- `ridge` A vector with `ncomp` elements all equal to the chosen ridge value
- `lasso` A vector with `ncomp` elements all equal to the chosen lasso value
- `grouplasso` A vector with `ncomp` elements all equal to the chosen group lasso value
- `elitistlasso` A vector with `ncomp` elements all equal to the chosen elitist lasso value

For more details see the package documentation.

5.5 Detailed examples of SCA and PCA with the `sparseWeightBasedPCA` package

In this section we will give some detailed examples where we analyze multi- and single block data with SCA and PCA including the model selection process. We give examples of `scads`, `mmsca` and `ccpca` and the model selection functions. We will demonstrate these procedures using simulated data, for that purpose the package includes data generating functions that simulate data according to the following structure:

$$\mathbf{X} = \mathbf{X}\mathbf{W}\mathbf{P}^T, \tag{5.6}$$

where $\mathbf{W}$ is $J \times J$, $\mathbf{W}^T\mathbf{W} = \mathbf{I}$ and $\mathbf{W} = \mathbf{P}$. $\mathbf{W}$ is manipulated such that it contains a specified level of sparsity in the first Q columns. The covariance matrix of $\mathbf{X}$ can be constructed by taking $\boldsymbol{\Sigma} = \mathbf{W}\boldsymbol{\Lambda}\mathbf{W}^T$, the eigenvalues in $\boldsymbol{\Lambda}$ can be manipulated to control the variance of the signal components versus the variance of the noise components. Using the covariance matrix data can be sampled from the multivariate normal distribution using `mvrnorm` from the `MASS` package (Venables and Ripley, 2002). Functions for data generation are provided by:

- `sparsify` to put sparsity in $\mathbf{W}$
- `makeVariance` to manipulate the eigenvalues in $\boldsymbol{\Lambda}$
- `makeDat` to simulate the data.

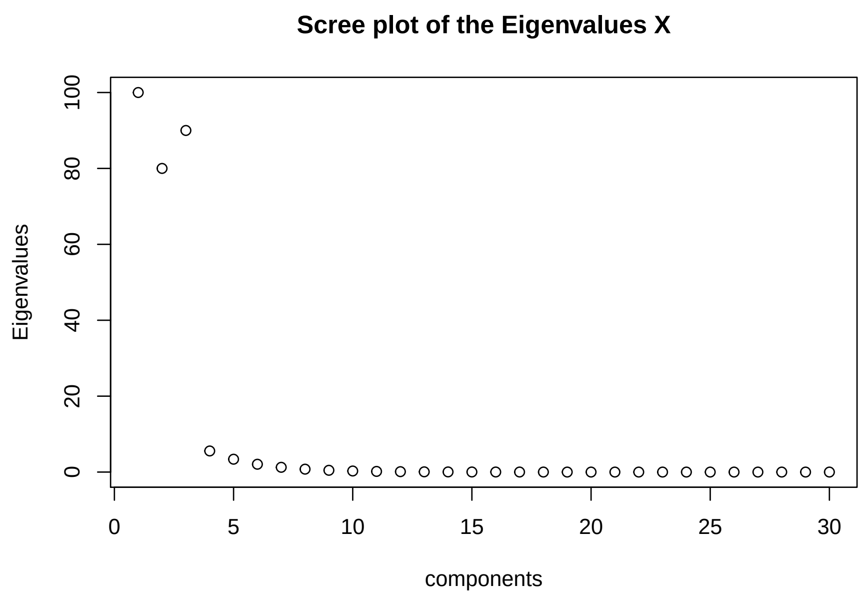
Check the package documentation for more details. In case own data are used, preprocessing may be needed: usually this is centering and scaling of each variable followed by a block scaling. The `RegularizedSCA` package (Gu and Van Deun, 2018) provides functionality for this with `pre_process`, see their documentation for more details.

5.5.1 Example of SCA with `scads`

Here we will demonstrate data analysis using `scads`. In this example we will have 2 data blocks each with 15 variables and 3 components. First we create a common and distinctive structure for the component weights to generate data from, we will use a structure with 2 distinctive components and 1 common component.

```
set.seed(1)
ncomp <- 3
J <- 30
comdis <- matrix(1, J, ncomp) # Component weight structure

# The first component is distinctive for the first block
comdis[1:15, 1] <- 0

# The second component is distinctive for the second block
comdis[16:30, 2] <- 0

# Set 20 percent of the 1's to zero
comdis <- sparsify(comdis, 0.2)

# Inspect the component weight structure. Weights indicated with a 1
# will in the data generating model
comdis
```

```
##       [,1] [,2] [,3]
##  [1,]    0    0    0
##  [2,]    0    0    1
##  [3,]    0    1    1
##  [4,]    0    1    1
##  [5,]    0    1    1
##  [6,]    0    1    1
##  [7,]    0    1    1
##  [8,]    0    1    1
##  [9,]    0    1    1
## [10,]    0    1    1
## [11,]    0    1    0
## [12,]    0    1    1
```



```
## [13,]    0    0    1
## [14,]    0    1    0
## [15,]    0    1    1
## [16,]    1    0    1
## [17,]    1    0    1
## [18,]    1    0    0
## [19,]    0    0    0
## [20,]    1    0    1
## [21,]    1    0    1
## [22,]    0    0    1
## [23,]    1    0    0
## [24,]    0    0    1
## [25,]    1    0    1
## [26,]    1    0    1
## [27,]    1    0    1
## [28,]    1    0    1
## [29,]    1    0    1
## [30,]    1    0    1
```

Now given this component weight structure we can simulate data as follows,

```
variances <- makeVariance(varianceOfComps = c(100, 80, 90),
                          J = J, error = 0.05)
plot(variances, xlab ="components",
     ylab ="Eigenvalues", main = "Scree plot of the Eigenvalues X")
```

```
dat <- makeDat(n = 100, comdis = comdis, variances = variances)
X <- dat$X
round(dat$P[, 1:ncomp], 3) # The data generating component weight structure
```

```
##         [,1]   [,2]   [,3]
##  [1,]  0.000  0.000  0.000
##  [2,]  0.000  0.000  0.072
##  [3,]  0.000 -0.082 -0.004
##  [4,]  0.000  0.517  0.149
##  [5,]  0.000  0.165  0.121
##  [6,]  0.000 -0.133 -0.226
##  [7,]  0.000 -0.245 -0.103
##  [8,]  0.000 -0.343  0.098
##  [9,]  0.000 -0.313  0.233
## [10,]  0.000 -0.460 -0.086
## [11,]  0.000  0.340  0.000
## [12,]  0.000  0.245 -0.197
## [13,]  0.000  0.000 -0.418
## [14,]  0.000  0.078  0.000
## [15,]  0.000 -0.111  0.334
## [16,]  0.054  0.000  0.208
## [17,]  0.326  0.000 -0.162
## [18,] -0.023  0.000  0.000
## [19,]  0.000  0.000  0.000
## [20,]  0.441  0.000  0.158
## [21,] -0.280  0.000  0.004
## [22,]  0.000  0.000  0.365
## [23,]  0.019  0.000  0.000
## [24,]  0.000  0.000  0.378
## [25,] -0.220  0.000 -0.188
## [26,] -0.176  0.000  0.055
## [27,] -0.264  0.000 -0.177
## [28,]  0.295  0.000  0.085
## [29,]  0.468  0.000 -0.105
## [30,]  0.403  0.000 -0.203
```

In the scree plot you can see the eigenvalues of the data generating covariance matrix and `round(dat$P[, 1:ncomp], 3)` prints the data generating component weights for the signal components.

Given the generated data set we can perform `scads` by first looking for the common and distinctive structure by trying out all structures and finding the best structure according to a model selection criterion, in this case we will use cross-validation. To generate all common and distinctive structures we implemented a function called `allCommonDistinctive` for more details see the documentation.

```
# Generate all possible common and distinctive structures
allstructures <- allCommonDistinctive(
                      vars = c(15, 15),
                      ncomp = 3,
                      allPermutations = TRUE,
                      filterZeroSegments = TRUE)

#Use cross-validation to look for the data generating structure
index <- rep(NA, length(allstructures))
```

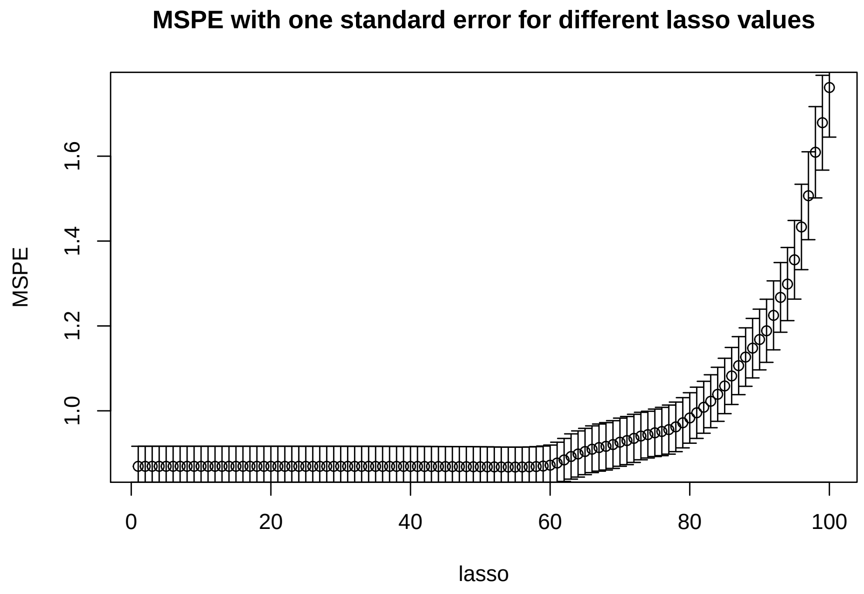
```
for (i in 1:length(allstructures)) {
    index[i] <- CVforPCAwithSparseWeights(X = X,
                                      nrFolds = 10,
                                      FUN = scads, # All parameters SCaDS needs
                                      ncomp, ridge = 0, lasso = rep(0, ncomp),
                                      constraints = allstructures[[i]],
                                      Wstart = matrix(0, J, ncomp),
                                      itr = 100000, nStarts = 1,
                                      printLoss = FALSE, tol = 10^-5)$MSPE
}

winningStructure <- allstructures[[which.min(index)]]

# Print the best common and distinctive structure
allstructures[[which.min(index)]]
```

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

Given this common and distinctive structure we can tune the lasso parameter in order to get some sparsity inside the weights, for this we will use cross-validation with the one standard error rule. This entails cross validating the model for different values of the `lasso` parameter, from which we will select the model with the most weights at zero still within one standard error of the model with the lowest mean squared prediction error (MSPE). If there are no models within one standard error of the best model the step size of the LASSO sequence should be decreased.

```r
# Generate candidate lasso values on the log-scale
lasso <- exp(seq(log(0.0000001), log(1), length.out = 100))
MSPE <- rep(NA, length(lasso))
MSPEstdError <- rep(NA, length(lasso))
nNonZeroCoef <- rep(NA, length(lasso))

for (i in 1:length(lasso)) {
    res <- CVforPCAwithSparseWeights(X = X,
                      nrFolds = 10,
                      FUN = scads,
                      ncomp, ridge = 0, lasso = rep(lasso[i], ncomp),
                      constraints = winningStructure,
                      Wstart = matrix(0, J, ncomp),
                      itr = 100000, nStarts = 1,
                      printLoss = FALSE, tol = 10^-5)

    MSPE[i] <- res$MSPE # Mean squared prediction error
    MSPEstdError[i] <- res$MSPEstdError # Standard error of the MSPE
    nNonZeroCoef[i] <- res$nNonZeroCoef # The number of non-zero weights
}

x <- 1:length(lasso)
plot(x , MSPE, xlab = "lasso", ylab = "MSPE",
     main = "MSPE with one standard error for different lasso values")

# Add error bars to the plot
arrows(x, MSPE - MSPEstdError, x,
        MSPE + MSPEstdError, length = 0.05, angle = 90, code = 3)
```

```r
# Select all models within one standard error of the best model
eligibleModels <- MSPE < MSPE[which.min(MSPE)] +
                    MSPEstdError[which.min(MSPE)]

# Selected from those models the models with the
# lowest number of non-zero weights
```



```
best <- which.min(nNonZeroCoef[eligibleModels])

# Do the analysis with the "winning" structure and best lasso
results <- scads(X = X, ncomp = ncomp,
                 ridge = 0, lasso = rep(lasso[best], ncomp),
                 constraints = allstructures[[which.min(index)]],
                 Wstart = matrix(0, J, ncomp),
                 itr = 100000, nStarts = 1,
                 printLoss = FALSE , tol = 10^-5)

# Compare results from the analysis with the data generating model
compare <- cbind(dat$P[, 1:ncomp], results$W)
colnames(compare) <- c(paste("True W_", 1:3, sep = ""),
                        paste("Est W_", 1:3, sep = ""))
rownames(compare) <- paste("Var", 1:30)
round(compare, 3)
```

```
##        True W_1 True W_2 True W_3 Est W_1 Est W_2 Est W_3
## Var 1     0.000    0.000    0.000   0.000   0.000   0.005
## Var 2     0.000    0.000    0.072   0.000   0.000   0.000
## Var 3     0.000   -0.082   -0.004   0.000   0.000   0.000
## Var 4     0.000    0.517    0.149   0.000  -0.021  -0.529
## Var 5     0.000    0.165    0.121   0.000  -0.138  -0.057
## Var 6     0.000   -0.133   -0.226   0.000   0.351   0.032
## Var 7     0.000   -0.245   -0.103   0.000   0.046   0.211
## Var 8     0.000   -0.343    0.098   0.000  -0.315   0.452
## Var 9     0.000   -0.313    0.233   0.000  -0.194   0.382
## Var 10    0.000   -0.460   -0.086   0.000   0.261   0.498
## Var 11    0.000    0.340    0.000   0.000   0.000  -0.328
## Var 12    0.000    0.245   -0.197   0.000   0.000  -0.178
## Var 13    0.000    0.000   -0.418   0.000   0.604   0.002
## Var 14    0.000    0.078    0.000   0.000   0.000   0.000
## Var 15    0.000   -0.111    0.334   0.000  -0.276   0.000
## Var 16    0.054    0.000    0.208   0.000  -0.279   0.000
## Var 17    0.326    0.000   -0.162   0.357   0.000   0.000
## Var 18   -0.023    0.000    0.000   0.000   0.000   0.000
## Var 19    0.000    0.000    0.000   0.000   0.000   0.000
## Var 20    0.441    0.000    0.158   0.446  -0.190   0.000
## Var 21   -0.280    0.000    0.004  -0.234  -0.042   0.000
## Var 22    0.000    0.000    0.365  -0.157  -0.536   0.000
## Var 23    0.019    0.000    0.000   0.000   0.000   0.000
## Var 24    0.000    0.000    0.378  -0.189  -0.232   0.000
## Var 25   -0.220    0.000   -0.188  -0.237   0.191   0.000
## Var 26   -0.176    0.000    0.055  -0.354   0.000   0.000
## Var 27   -0.264    0.000   -0.177  -0.164   0.137   0.000
## Var 28    0.295    0.000    0.085   0.177  -0.050   0.000
## Var 29    0.468    0.000   -0.105   0.479   0.000   0.000
## Var 30    0.403    0.000   -0.203   0.423  -0.005   0.000
```

This concludes the analysis of multi-block data with scads. This procedure offers lots of flexibility to the user at the cost of a little more complexity (i.e. the user has to know for-loops and some R basics). Note that the model selection procedures can be easily sped up making use of parallel versions of the for-loops.

5.5.2 Example of SCA with mmsca

We will now demonstrate data analysis of multi-block data using mmsca. In this example we will use the same data as in the scads example. We will demonstrate the use of mmscaHyperCubeSelection as already for this toy-example tuning an exhaustive grid with mmscaModelSelection takes too long. We use cross-validation with the one standard error rule.

```
# Tune the hyper parameters by looking at an adaptive grid
# that zooms in around a plausible combination of values
out <- mmscaHyperCubeSelection(X,
              ncomp = 3,
              ridgeSeq = 0, # No ridge
              lassoSeq = 0:10, # Lasso from 0 to 10
              grouplassoSeq = 0:10,  # Lasso from 0 to 10
              elitistlassoSeq = 0, # No elitist lasso
              stepsize = 3,
              logscale = FALSE,
              stopWhenRange = 0.01, # Stop when average range is < 0.01
              groups = c(15, 15),
              nStart = 1,
              itr = 100000,
              printProgress = FALSE,
              coorDes = FALSE,
              coorDesItr = 1,
              method = "CV1stdError",
              tol = 10e-5,
              nrFolds = 10)

# Inspect the chosen hyper parameters
out
```

```
## $ridge
## [1] 0 0 0
##
## $lasso
## [1] 6.210938 6.210938 6.210938
##
## $grouplasso
## [1] 7.460938 7.460938 7.460938
##
## $elitistlasso
## [1] 0 0 0
```

```
# Run the analysis with the chosen hyper parameters
results <- mmsca(X = X,
             ncomp = ncomp,
             ridge = out$ridge,
             lasso = out$lasso,
             grouplasso = out$grouplasso,
             elitistlasso = out$elitistlasso,
             groups = c(15, 15),
             constraints = matrix(1, J, ncomp),
             itr = 1000000,
             Wstart = matrix(0, J, ncomp),
             nStarts = 1,
             printLoss = FALSE)

# Compare results from the analysis with the data generating model
compare <- cbind(dat$P[, 1:ncomp], results$W)
colnames(compare) <- c(paste("True W_", 1:3, sep = ""),
                        paste("Est W_", 1:3, sep = ""))
rownames(compare) <- paste("Var", 1:30)
round(compare, 3)
```

```
##         True W_1 True W_2 True W_3 Est W_1 Est W_2 Est W_3
## Var 1      0.000    0.000    0.000   0.000   0.000   0.000
## Var 2      0.000    0.000    0.072   0.000   0.000   0.000
## Var 3      0.000   -0.082   -0.004   0.000   0.000   0.000
## Var 4      0.000    0.517    0.149   0.000  -0.091  -0.583
## Var 5      0.000    0.165    0.121   0.000  -0.023  -0.086
## Var 6      0.000   -0.133   -0.226   0.000   0.161   0.059
## Var 7      0.000   -0.245   -0.103   0.000   0.000   0.207
## Var 8      0.000   -0.343    0.098   0.000   0.000   0.403
## Var 9      0.000   -0.313    0.233   0.000  -0.157   0.345
## Var 10     0.000   -0.460   -0.086   0.000   0.000   0.492
## Var 11     0.000    0.340    0.000   0.000   0.000  -0.319
## Var 12     0.000    0.245   -0.197   0.000   0.010  -0.165
## Var 13     0.000    0.000   -0.418   0.000   0.527   0.000
## Var 14     0.000    0.078    0.000   0.000   0.000   0.000
## Var 15     0.000   -0.111    0.334   0.000  -0.369   0.000
## Var 16     0.054    0.000    0.208   0.000  -0.286   0.000
## Var 17     0.326    0.000   -0.162   0.313   0.032   0.000
## Var 18    -0.023    0.000    0.000   0.000   0.000   0.000
## Var 19     0.000    0.000    0.000   0.000   0.000   0.000
## Var 20     0.441    0.000    0.158   0.419  -0.129   0.000
## Var 21    -0.280    0.000    0.004  -0.223   0.000   0.000
## Var 22     0.000    0.000    0.365   0.000  -0.498   0.000
## Var 23     0.019    0.000    0.000   0.000   0.000   0.000
## Var 24     0.000    0.000    0.378   0.000  -0.533   0.000
## Var 25    -0.220    0.000   -0.188  -0.153   0.119   0.000
## Var 26    -0.176    0.000    0.055  -0.046   0.000   0.000
## Var 27    -0.264    0.000   -0.177  -0.233   0.178   0.000
## Var 28     0.295    0.000    0.085   0.328   0.000   0.000
## Var 29     0.468    0.000   -0.105   0.544   0.050   0.000
## Var 30     0.403    0.000   -0.203   0.483   0.168   0.000
```

We now demonstrate another interesting use case for `mmsca` where a researcher wants to identify common and distinctive components and where an exhaustive approach might fail because there are too many data blocks and components. We use multi-block data with 5 blocks and 5 variables per block and 5 components.

```r
# Generate the data
set.seed(1)
ncomp <- 5
J <- 30
comdis <- matrix(0, J, ncomp) # Component weight structure
comdis[1:5, 1] <- 1
comdis[6:10, 2] <- 1
comdis[25:30, 3] <- 1
comdis[11:15, 4] <- 1
comdis[0:10, 5] <- 1
comdis[16:30, 5] <- 1

# check the generated component weight structure
comdis
```

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

```
# Generate data according to that structure
variances <- makeVariance(varianceOfComps = c(100, 80, 90, 50, 60),
                          J = J, error = 0.05)

dat <- makeDat(n = 100, comdis = comdis, variances = variances)
X <- dat$X

# Tune the group lasso parameter
out <- mmscaHyperCubeSelection(X,
              ncomp = ncomp,
              ridgeSeq = 0, # No ridge
              lassoSeq = 0, # No lasso
              grouplassoSeq = 0:10,  # Lasso from 0 to 10
              elitistlassoSeq = 0, # No elitist lasso
              stepsize = 3,
              logscale = FALSE,
              stopWhenRange = 0.01, # Stop when average range is < 0.01
              groups = c(5, 5, 5, 5, 5, 5),
              nStart = 1,
              itr = 100000,
              printProgress = FALSE,
              coorDes = FALSE,
              coorDesItr = 1,
              method = "CV1stdError",
              tol = 10e-5,
              nrFolds = 10)

# Inspect the chosen group lasso
out$grouplasso
```

```
## [1] 13.125 13.125 13.125 13.125 13.125
```

```
# Run the analysis
results <- mmsca(X = X,
                ncomp = ncomp,
                ridge = out$ridge,
                lasso = out$lasso,
                grouplasso = out$grouplasso,
                elitistlasso = out$elitistlasso,
                groups = c(5, 5, 5, 5, 5, 5),
                constraints = matrix(1, J, ncomp),
                itr = 1000000,
                Wstart = matrix(0, J, ncomp),
                nStarts = 1,
                printLoss = FALSE)



# Check results from the analysis
colnames(results$W) <- paste("Est W_", 1:5, sep = "")
rownames(results$W) <- paste("Var", 1:30)
round(results$W, 3)
```

```
##         Est W_1 Est W_2 Est W_3 Est W_4 Est W_5
## Var 1     0.228   0.000   0.000  -0.035   0.000
## Var 2    -0.065   0.000   0.000   0.042   0.000
## Var 3     0.346   0.000   0.000  -0.022   0.000
## Var 4    -0.893   0.000   0.000  -0.002   0.000
## Var 5    -0.209   0.000   0.000   0.001   0.000
## Var 6     0.000   0.000  -0.278   0.000   0.000
## Var 7     0.001   0.000  -0.233   0.000   0.000
## Var 8     0.000   0.000  -0.103   0.000   0.000
## Var 9    0.000   0.000   0.741   0.000   0.000
## Var 10   0.000   0.000   0.558   0.000   0.000
## Var 11   0.000   0.000   0.000   0.000   0.516
## Var 12   0.000   0.000   0.000   0.000   0.015
## Var 13   0.000   0.000   0.000   0.000   0.631
## Var 14   0.000   0.000   0.000   0.000  -0.222
## Var 15   0.000   0.000   0.000   0.000   0.547
## Var 16   0.000   0.000   0.000   0.000  -0.014
## Var 17   0.000   0.000   0.000   0.000   0.008
## Var 18   0.000   0.000   0.000   0.000   0.009
## Var 19   0.000   0.000   0.000   0.000   0.007
## Var 20   0.000   0.000   0.000   0.000  -0.001
## Var 21   0.012  -0.150   0.000  -0.663   0.000
## Var 22   0.005   0.085   0.000   0.363   0.000
## Var 23   0.007  -0.034   0.000  -0.464   0.000
## Var 24  -0.004  -0.049   0.000  -0.178   0.000
## Var 25  -0.014   0.299   0.000  -0.448   0.000
## Var 26   0.000   0.231   0.000  -0.051   0.003
## Var 27   0.000   0.869   0.000   0.185  -0.003
## Var 28   0.000  -0.286   0.000   0.112  -0.002
## Var 29   0.000   0.150   0.000  -0.308   0.000
## Var 30   0.000   0.031   0.000  -0.432   0.000
```

```r
# You can compare them to the data generating structure
round(dat$P[, 1:ncomp], 3)
```

```
##          [,1]   [,2]   [,3]   [,4]   [,5]
##  [1,] -0.322  0.000  0.000  0.000 -0.089
##  [2,]  0.094  0.000  0.000  0.000  0.270
##  [3,] -0.430  0.000  0.000  0.000 -0.025
##  [4,]  0.821  0.000  0.000  0.000 -0.073
##  [5,]  0.170  0.000  0.000  0.000 -0.028
##  [6,]  0.000 -0.285  0.000  0.000  0.095
##  [7,]  0.000 -0.271  0.000  0.000 -0.063
##  [8,]  0.000 -0.041  0.000  0.000 -0.015
##  [9,]  0.000  0.755  0.000  0.000 -0.006
## [10,]  0.000  0.524  0.000  0.000  0.026
## [11,]  0.000  0.000  0.000 -0.480  0.000
## [12,]  0.000  0.000  0.000  0.033  0.000
## [13,]  0.000  0.000  0.000 -0.705  0.000
## [14,]  0.000  0.000  0.000  0.122  0.000
## [15,]  0.000  0.000  0.000 -0.507  0.000
## [16,]  0.000  0.000  0.000  0.000 -0.313
## [17,]  0.000  0.000  0.000  0.000 -0.061
## [18,]  0.000  0.000  0.000  0.000 -0.108
## [19,]  0.000  0.000  0.000  0.000 -0.133
## [20,]  0.000  0.000  0.000  0.000 -0.012
## [21,]  0.000  0.000  0.000  0.000 -0.390
## [22,]  0.000  0.000  0.000  0.000  0.240
## [23,]  0.000  0.000  0.000  0.000 -0.339
## [24,]  0.000  0.000  0.000  0.000 -0.094
## [25,]  0.000  0.000  0.432  0.000 -0.252
## [26,]  0.000  0.000  0.242  0.000 -0.167
## [27,]  0.000  0.000  0.772  0.000  0.381
## [28,]  0.000  0.000 -0.221  0.000  0.016
## [29,]  0.000  0.000  0.269  0.000 -0.278
## [30,]  0.000  0.000  0.194  0.000 -0.346
```

The data generating structure has been estimated back fairly well. Note not all segments have been recovered. If a segment does not really contribute or only contributes marginally to the signal variance, it can be put to zero by the group lasso. One can also notice some small coefficients still being estimated, if the `grouplasso` is increased, these small coefficients will be put to zero.

5.5.3 Example of PCA with `ccpca`

Here we will demonstrate data analysis using `ccpca`. In this example we will have just 1 data block with 30 variables and 3 components. Here we demonstrate the use case for `ccpca` where the underlying model is rather sparse, and we already assume the model is sparse and we ballpark the sparsity.

```
set.seed(1)
ncomp <- 3
J <- 30
comdis <- matrix(1, J, ncomp) # Component weight structure
comdis <- sparsify(comdis, 0.8) # Set 80 percent of the 1's to zero
comdis
```

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

```
variances <- makeVariance(varianceOfComps = c(100, 80, 90),
                          J = J, error = 0.05)
dat <- makeDat(n = 100, comdis = comdis, variances = variances)
X <- dat$X
round(dat$P[, 1:ncomp], 3) # The data generating component weight structure
```

```
##         [,1]   [,2]   [,3]
##  [1,]  0.000  0.000  0.000
##  [2,]  0.000  0.000  0.000
##  [3,]  0.766  0.000  0.000
##  [4,]  0.000 -0.469  0.000
##  [5,]  0.000 -0.740  0.000
##  [6,]  0.176  0.000  0.000
##  [7,]  0.000  0.000  0.000
##  [8,]  0.000  0.000  0.000
##  [9,]  0.000  0.000  0.016
## [10,]  0.000  0.000  0.000
## [11,]  0.000 -0.216  0.837
## [12,]  0.000  0.000  0.000
## [13,]  0.000 -0.142  0.000
## [14,]  0.000  0.000  0.000
## [15,]  0.000  0.000  0.000
## [16,]  0.238  0.000  0.000
## [17,]  0.000  0.000  0.000
## [18,]  0.000  0.000  0.000
## [19,]  0.000  0.000  0.000
## [20,] -0.540  0.000  0.000
## [21,]  0.000  0.062  0.000
## [22,]  0.000  0.000  0.000
## [23,]  0.000  0.000  0.152
## [24,]  0.165  0.000  0.000
## [25,]  0.000  0.000  0.000
## [26,]  0.000  0.402  0.448
## [27,]  0.000  0.000 -0.275
## [28,]  0.000  0.000  0.000
## [29,] -0.079  0.000  0.000
## [30,]  0.000  0.000  0.000
```

```
# We can ballpark the number of zeros and can get pretty good results
results <- ccpca(X = X, ncomp = ncomp,
            nzeros = c(20, 20, 20), itr = 10000000,
            Wstart = matrix(0, J, ncomp),
            nStarts = 1, tol = 10^-8,
            printLoss = FALSE)
```

```r
# Compare the results
compare <- cbind(dat$P[, 1:ncomp], results$W)
colnames(compare) <- c(paste("True W_", 1:3, sep = ""),
                       paste("Est W_", 1:3, sep = ""))
rownames(compare) <- paste("Var", 1:30)
round(compare, 3)
```

```
##        True W_1 True W_2 True W_3 Est W_1 Est W_2 Est W_3
## Var 1     0.000    0.000    0.000   0.000   0.000   0.000
## Var 2     0.000    0.000    0.000   0.000   0.000   0.000
## Var 3     0.766    0.000    0.000  -0.749   0.013  -0.014
## Var 4     0.000   -0.469    0.000   0.000   0.000   0.541
## Var 5     0.000   -0.740    0.000  -0.041   0.000   0.708
## Var 6     0.176    0.000    0.000  -0.198   0.000   0.000
## Var 7     0.000    0.000    0.000   0.000   0.000   0.000
## Var 8     0.000    0.000    0.000   0.000   0.000   0.000
## Var 9     0.000    0.000    0.016   0.000   0.000   0.000
## Var 10    0.000    0.000    0.000   0.000   0.000   0.000
## Var 11    0.000   -0.216    0.837  -0.023  -0.821   0.212
## Var 12    0.000    0.000    0.000   0.000   0.000   0.000
## Var 13    0.000   -0.142    0.000   0.000   0.000   0.061
## Var 14    0.000    0.000    0.000   0.000   0.000   0.000
## Var 15    0.000    0.000    0.000   0.000   0.000   0.000
## Var 16    0.238    0.000    0.000  -0.264   0.000   0.000
## Var 17    0.000    0.000    0.000   0.000   0.000   0.000
## Var 18    0.000    0.000    0.000   0.000   0.000   0.000
## Var 19    0.000    0.000    0.000   0.000  -0.029   0.000
## Var 20   -0.540    0.000    0.000   0.537  -0.010   0.027
## Var 21    0.000    0.062    0.000   0.000   0.000  -0.191
## Var 22    0.000    0.000    0.000   0.000   0.000   0.000
## Var 23    0.000    0.000    0.152   0.000  -0.141   0.000
## Var 24    0.165    0.000    0.000  -0.159   0.000   0.000
## Var 25    0.000    0.000    0.000   0.000   0.000   0.000
## Var 26    0.000    0.402    0.448  -0.009  -0.473  -0.382
## Var 27    0.000    0.000   -0.275   0.000   0.283  -0.012
## Var 28    0.000    0.000    0.000   0.000  -0.026   0.000
## Var 29   -0.079    0.000    0.000   0.139  -0.080   0.066
## Var 30    0.000    0.000    0.000   0.000  -0.012   0.000
```

The estimation results are similar to the data generating model, this shows you do not have to know the exact number of zero weights beforehand in order to get interpretable results. That being said, it should preferably in the ballpark of what it should really be. In order to examine that in more detail we will now provide an example of model selection with `ccpca` also in this example we use cross-validation with the one standard error rule.

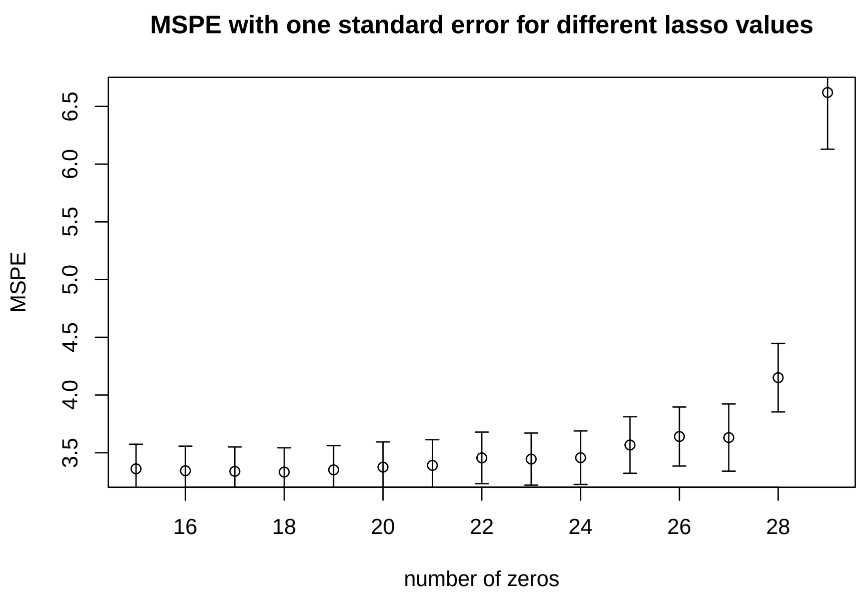
```r
# Cross validate the number of non-zero weights per component weight vector
# Note: we assume the sparsity within the columns of W to be the same
nzeros <- rep(15:29, each = ncomp)
nzeros <- split(nzeros, ceiling(seq_along(nzeros) / ncomp))
```

```r
MSPE <- rep(NA, length(nzeros))
MSPEstdError <- rep(NA, length(nzeros))
nNonZeroCoef <- rep(NA, length(nzeros))

for (i in 1:length(nzeros)) {
    res <- CVforPCAwithSparseWeights(X = X, nrFolds = 10, FUN = ccpca,
                  ncomp = ncomp,  nzeros = nzeros[[i]], itr = 10000000,
                  Wstart = matrix(0, J, ncomp), nStarts = 1,
                  tol = 10^-5, printLoss = FALSE)

    # Store MSPE for each lasso value
    MSPE[i] <- res$MSPE
    # Store the standard error of the MSPE
    MSPEstdError[i] <- res$MSPEstdError
    # Store the number of non-zero weights
    nNonZeroCoef[i] <- res$nNonZeroCoef
}


x <- 15:29
plot(x , MSPE, xlab = "number of zeros", ylab = "MSPE",
     main = "MSPE with one standard error for different lasso values")

# Add error bars to the plot
arrows(x, MSPE - MSPEstdError, x , MSPE + MSPEstdError,
        length = 0.05, angle = 90, code = 3)
```

```r
# Select all models within one standard error of the best model
eligibleModels <- MSPE < MSPE[which.min(MSPE)] +
                    MSPEstdError[which.min(MSPE)]
eligibleModels
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE
## [13] FALSE FALSE FALSE
```

```r
# Select from those models the models with
# the lowest number of non-zero weights
best <- which.min(nNonZeroCoef[eligibleModels])

# The number of zero weights per component that was "best"
nzeros[[best]]
```



```
## [1] 24 24 24
```

```
results <- ccpca(X = X,
                 ncomp = ncomp,
                 nzeros = nzeros[[best]],
                 itr = 10000000,
                 Wstart = matrix(0, J, ncomp),
                 nStarts = 1,
                 tol = 10^-8,
                 printLoss = FALSE)

compare <- cbind(dat$P[, 1:ncomp], results$W)
colnames(compare) <- c(paste("True W_", 1:3, sep = ""),
        paste("Est W_", 1:3, sep = ""))
rownames(compare) <- paste("Var", 1:30)
round(compare, 3)
```

```
##        True W_1 True W_2 True W_3 Est W_1 Est W_2 Est W_3
## Var 1     0.000    0.000    0.000   0.000   0.000   0.000
## Var 2     0.000    0.000    0.000   0.000   0.000   0.000
## Var 3     0.766    0.000    0.000  -0.810   0.106  -0.028
## Var 4     0.000   -0.469    0.000   0.000   0.000   0.507
## Var 5     0.000   -0.740    0.000   0.000   0.000   0.755
## Var 6     0.176    0.000    0.000  -0.188   0.000   0.000
## Var 7     0.000    0.000    0.000   0.000   0.000   0.000
## Var 8     0.000    0.000    0.000   0.000   0.000   0.000
## Var 9     0.000    0.000    0.016   0.000   0.000   0.000
## Var 10    0.000    0.000    0.000   0.000   0.000   0.000
## Var 11    0.000   -0.216    0.837  -0.098  -0.802   0.225
## Var 12    0.000    0.000    0.000   0.000   0.000   0.000
## Var 13    0.000   -0.142    0.000   0.000   0.000   0.000
## Var 14    0.000    0.000    0.000   0.000   0.000   0.000
## Var 15    0.000    0.000    0.000   0.000   0.000   0.000
## Var 16    0.238    0.000    0.000  -0.152   0.000   0.000
## Var 17    0.000    0.000    0.000   0.000   0.000   0.000
## Var 18    0.000    0.000    0.000   0.000   0.000   0.000
## Var 19    0.000    0.000    0.000   0.000   0.000   0.066
## Var 20   -0.540    0.000    0.000   0.491   0.000   0.000
## Var 21    0.000    0.062    0.000   0.000   0.000   0.000
## Var 22    0.000    0.000    0.000   0.000   0.000   0.000
## Var 23    0.000    0.000    0.152   0.000  -0.178   0.000
## Var 24    0.165    0.000    0.000  -0.248   0.000   0.000
## Var 25    0.000    0.000    0.000   0.000   0.000   0.000
## Var 26    0.000    0.402    0.448   0.000  -0.480  -0.384
## Var 27    0.000    0.000   -0.275   0.000   0.298   0.000
## Var 28    0.000    0.000    0.000   0.000  -0.074   0.000
## Var 29   -0.079    0.000    0.000   0.000   0.000   0.000
## Var 30    0.000    0.000    0.000   0.000   0.000   0.000
```

Running the model selection procedure gives the user a pretty good idea what number of zero weights is plausible. The results of the estimation using cross-validation with the one standard error rule are very similar to the one where we ballparked the number of zero weights.

5.6 Conclusion

The sparseWeightBasedPCA package provides flexible analysis tools to perform SCA and PCA on multi- or single block data with sparsity in the component weights. Model selection tools are provided to select the hyper-parameters of these models. Future research should focus on selecting the regularization parameters together with the number of components. Although doable with this package it is still too computationally demanding for most practical use cases. For the moment we recommend to use a sequential strategy where first the number of components is determined and, next, the common and distinctive structure and/or the regularization parameters are tuned. The selection of the components can be based for example on scree-plots or using CVforPCAwithSparseWeights. Furthermore this package includes mmscaHyperCubeSelection, a promising model selection procedure because of its speed and ability to examine an enormous grid of potential candidate values for the hyper parameter. However, it still needs to be properly examined using a simulation study in further research.

Chapter 6

Epilogue

This final chapter discusses some closing thoughts on the research reported previous chapters. I will reflect upon the results presented and give some pointers for directions that future research could go in.

This thesis presented sparse weight based PCA and SCA procedures for the analysis of multi-block data. By taking into account the multi-block structure, these procedures aim at facilitating a better recovery of the underlying processes than procedures that do not. In Chapter 2, we presented an exhaustive approach for identifying common and distinctive components[1]. Compared to sparse PCA of the concatenated data and thus ignoring the multi-block structure, recovery of the underlying model turned out to be substantially better when accounting for the multi-block data structure. This clearly shows the need for specific analysis tools for multi-block structure. As a follow up, we answered the question whether a multi-block structure can be discovered when it is not known beforehand. In Chapter 2, we examined 10-fold cross-validation as a means to automatically detect the multi-block structure of the data set concerned, and concluded that cross-validation is not well suited as a method for automatically identifying the underlying common and distinctive structure of the data. This lead to Chapter 3 in search of model selection techniques that might be more successful in discovering the common and distinctive structure.

In Chapter 3, we tested a variety of readily available model selection techniques in order to examine their applicability in the context of sparse weight based SCA/PCA. We found 10-fold cross-validation with the one standard error rule to be the best tool for this purpose. This seems bad news, as we concluded previously, in Chapter 2, that cross-validation (including the one standard error rule) did not perform well. However, in Chapter 3 we concluded it is still the best overall. In

[1]The term common and distinctive is not unambiguous, but in our case we mean structured sparsity in the weights rather than in the component scores (Schouteden et al., 2013)

the next section, I will put these seemingly contradicting results into a broader perspective.

6.1 A note on model selection

One of the reasons cross-validation (as other model selection techniques) performed badly is associated with the use of the rather strict definition of common and distinctive introduced in Chapters 2 and 3. In this definition, for a common component there should be at least one non-zero weight in each of the blocks, and for a distinctive component all weights of the distinctive blocks should be zero. Indeed, according to this definition, it is valid to conclude that, for the most part, one can not automatically estimate back common and distinctive components using the investigated model selection rules. However, if one relaxes classification according to definition by allowing for a certain amount of violation in the estimates in order to meet the definition (i.e., some very small coefficients in distinctive blocks are allowed), it can be seen that common and distinctive components can be found back rather well, albeit not automatically. This observation implies a researcher still has to decide upon an inspection of the weights whether some weights are so small that they probably are not relevant and therefore conclude that a certain component might be a distinctive one. This can lead to the urge of setting small coefficients to zero — pretending they are not there — but it would better to resist this urge and instead increase the value of the tuning parameter in order to get the preferred result. Concluding, common and distinctive components cannot be found automatically based on model selection tools investigated here, but with a little additional effort from the researcher, they *can* be found.

Another point I want to clarify is the poor performance of the other model selection techniques, such as the BIC, the IS, and cross-validation without the one standard error rule. Examining their complexity versus misfit (or goodness-of-fit) curves shows these procedures yield very similar curves, indicating they contain similar information about the underlying complexity of the model. However, if you select a model using the “pick the best index” rule, they differ quite substantially. For example, it is well known that cross-validation tends to pick overly complex models, while BIC and IS pick very sparse models. This does not mean that the indices themselves are useless, but that there is no single best rule for model selection. Cross-validation seemingly has a useful rule to select models with, namely the one standard error rule, which performed overall the best in our simulation studies. However, this does not mean it works best in any situation, implying users should not rely on it blindly. The main reason for its good performance in our simulation study is its tendency to pick a model that is right on the curve where complexity is getting lower while misfit is also still low. But this is not always the case, since the standard error of the mean squared prediction error can vary quite substantially as a function of the data and the number of folds, which in turn influences the model that is picked as the most suitable one, for better or worse. In order to select a model that strikes the right balance between complexity and misfit, the C-Hull (or L-curve) method was developed. In Chapter 3, however, it seemed to perform badly, indicating it is perhaps too difficult for this method to pick the right point on the curve that strikes an optimal balance between complexity and misfit, or it is not well suited for dealing with these specific kinds of misfit versus complexity curves. Speculations aside, I argue this might be a case of the “traveling salesman problem” a problem, which is notoriously hard for an algorithm to solve, but easy for a human. A human can easily select a solution that is among the best of the possible solutions. Therefore, I advise the end users of these models not to hold on too tightly on strict rules for model selection. In case the results from the automatically chosen models are not satisfactory, one can always inspect the complexity versus misfit curve, and visually infer a reasonable value for the tuning parameter.

6.2 Computational feasibility

One of the topics dealt with in this thesis was the tuning of the regularization parameters. A natural extension of the work presented here would be to study the tuning of regularization parameters in conjunction with the selection of the number of components. However, for this research to be even relevant and applicable in the real world, it should be computationally feasible to do so. What is computationally feasible depends on the computing power at your disposal and your patience, which, in most cases (as in my case) is some and none, respectively. In my opinion, estimation should be a matter of seconds for a reasonably sized data set (say 1000 variables), which is currently not the case for the methods presented here. It is hard to say what the actual average computational time is as it depends on a large number of external factors. But, what is clear is that it is (much) more than a few seconds, implying there is still room for improvements.

A possible way to speed up the algorithms presented in this thesis is to make use of the fact that the columns of component weight matrix $\mathbf{W}$ can be updated independently from each other. The latter implies the estimation of the columns in $\mathbf{W}$ can performed in parallel, potentially decreasing the computational time drastically. The algorithms in this thesis have been programmed in `C++`, allowing for straightforward integration with OpenMP (Dagum and Menon, 1998), which supports multi-platform shared-memory multiprocessing programming. Theoretically, this will speed up estimation by a factor Q (the number of components) when implemented in `sparseWeightBasedPCA` package. Another improvement that could readily be implemented is to solve the loss function with group lasso penalty from Chapter 3 with coordinate descent (Yuan and Lin, 2006) instead of an MM-algorithm, which has slow convergence properties.

Although model selection can still take a long time, the procedure itself can already be completely parallelized. When computing clusters get more accessible — and I am certain they will — applied researchers can capitalize on them.

6.3 Sparse weights versus sparse loadings models

The last topic that I would like to address in this chapter is the difference between sparse weights based SCA/PCA models and sparse loadings SCA/PCA models. I will discuss them both briefly to shed some light on their differences, as well as on why a researcher might want to opt for one or the other. In classical PCA estimated with the singular value decomposition (SVD), the loading matrix $\mathbf{P}$ equals the weight matrix $\mathbf{W}$, entailing the interpretation of the loadings and the weights can be interchanged, which is convenient.

In classical PCA estimated with the singular value decomposition (SVD), the loading matrix $\mathbf{P}$ equals the weight matrix $\mathbf{W}$, entailing the interpretation of the loadings and the weights can be interchanged, which is convenient. However, this equality does no longer hold when either $\mathbf{P}$ or $\mathbf{W}$ is restricted to be sparse, implying the interpretation of the weights and the loadings is not interchangeable anymore. This is a rather common cause of confusion and begs the question why to pick one approach over the other. In my opinion, the origin of the two approaches is purely driven by algorithmic convenience, and little thought has been given so far to the theoretical differences between the two approaches. Therefore, let us briefly discuss them here.

In the weight based PCA approach, the following loss function is optimized,

$$\begin{aligned} L(\mathbf{W},\mathbf{P}) &= \|\mathbf{X} - \mathbf{X}\mathbf{W}\mathbf{P}^T\|_2^2 + p(\mathbf{W}) \\ &= \|\text{vec}(\mathbf{X}) - (\mathbf{P} \otimes \mathbf{X})\text{vec}(\mathbf{W})\|_2^2 + p(\mathbf{W}) \qquad (6.1) \\ &\text{subject to } \mathbf{P}^T\mathbf{P} = \mathbf{I}, \end{aligned}$$

where $p()$ can be any penalty function. Given $\mathbf{P}$, the problem of finding $\mathbf{W}$ can be recognized as a regression problem and can thus be solved by any (sparse) regression procedure. Since $\mathbf{T} = \mathbf{X}\mathbf{W}$, it is straightforward to understand the meaning of $\mathbf{T}$, namely a linear combination of known entities (the columns of $\mathbf{X}$). However, when using this approach, $\mathbf{P}$ is a non-sparse matrix, implying that relating the components back to the original items might be complicated. This is most likely not going to be an issue, as the goal — understanding the components — is already attained by inspecting the weights.

The sparse $\mathbf{P}$ model is given by,

$$\begin{aligned} L(\mathbf{T}, \mathbf{P}) &= \|\mathbf{X} - \mathbf{T}\mathbf{P}^T\|_2^2 + p(\mathbf{P}) \\ &= \|\text{vec}(\mathbf{X}) - (\mathbf{I}_J \otimes \mathbf{T})\text{vec}(\mathbf{P}^T)\|_2^2 + p(\mathbf{P}) \\ &\text{subject to } \mathbf{T}^T\mathbf{T} = \mathbf{I}, \end{aligned} \tag{6.2}$$

where the identification constraints are on the component scores $\mathbf{T}$. Also in this instance, given $\mathbf{T}$, a solution to $\mathbf{P}$ can be found by using any (sparse) regression technique. A nice consequence of using this model is that is the predictor matrix $\mathbf{I}_J \otimes \mathbf{T}$ is blockwise diagonal and also orthogonal. This means that finding estimates for p_{jq} equals a simple univariate regression problem in which $\mathbf{x}_j$ is regressed on $\mathbf{t}_q$. That is, since there is no collinearity, the estimates of p_{jq} do not depend on the other $J-1$ variables included in the model, which is a good property to have. Such a feature is certainly not present in the sparse $\mathbf{W}$ model, where only its columns can be estimated independently from each other.

Given $\mathbf{P}$, the component scores are estimated by taking $\mathbf{T}$ equal to $\mathbf{U}\mathbf{V}^T$ (Gu and Van Deun, 2018) obtained with a singular value decomposition of $\mathbf{XP}$. This shows the interpretation from traditional PCA (where we can take $\mathbf{T} = \mathbf{XP}$) does not hold anymore. In this instance, $\mathbf{T}$ is given by an orthogonal basis for the column space of $\mathbf{XP}$ times $\mathbf{V}^T$, which is not immediately interpretable as $\mathbf{T}$ being a linear combination of unknown entities. An interpretation of the component scores would have to go indirectly through $\mathbf{P}$, which I believe is a suboptimal way of interpreting the components.

Theoretical differences aside, future research will have to determine whether there really is a practical difference between sparse $\mathbf{P}$ and $\mathbf{W}$ models. For example, do these models reveal the same patterns in the data? This research has to assume a model that applied researchers actually believe is underlying their data, that is in my opinion the PCA model with $\mathbf{X} = \mathbf{X}\mathbf{P}\mathbf{P}^T$ with $\mathbf{P}^T\mathbf{P} = \mathbf{I}$ and where $\mathbf{P}$ is sparse. The `sparseWeightBasedPCA` package provides functionality for generating data according to this model. This comparison should also take the algorithmic differences between the two implementations into consideration. As sparse $\mathbf{P}$ models are easier to deal with (because $\mathbf{T}^T\mathbf{T} = \mathbf{I}$), this could outweigh any benefits sparse $\mathbf{W}$ models might have.

The next logical step in this line of research would be to develop a sparse PCA method that estimates sparse $\mathbf{P}$ with $\mathbf{P}^T\mathbf{P} = \mathbf{I}$. Such a problem similar to the sparse singular value decomposition by Yang et al. (2014) or the sparse eigenvalue decomposition by Benidis et al. (2016). Naturally, extensions for multi-block data are needed and should be compared to the current sparse $\mathbf{W}$ and $\mathbf{P}$ approaches.

References

Acar, E., Papalexakis, E. E., Gürdeniz, G., Rasmussen, M. A., Lawaetz, A. J., Nilsson, M., and Bro, R. (2014). Structure-revealing data fusion. *BMC Bioinformatics*, 15(1).

Adachi, K. and Kiers, H. A. L. (2017). Sparse regression without using a penalty function.

Alter, O., Brown, P. O., and Botstein, D. (2003). Generalized singular value decomposition for comparative analysis of genome-scale expression data sets of two different organisms. *Proceedings of the National Academy of Sciences*, 100(6):33513356.

Alzheimers Disease Neuroimaging Initiative (2017). Adni study design. `http://adni.loni.usc.edu/study-design/#background-container`. [Online; accessed 22-August-2018].

Benidis, K., Sun, Y., Babu, P., and Palomar, D. P. (2016). Orthogonal sparse pca and covariance estimation via procrustes reformulation. *IEEE Transactions on Signal Processing*, 64(23):62116226.

Bertsimas, D., King, A., and Mazumder, R. (2016). Best subset selection via a modern optimization lens. *The Annals of Statistics*, 44(2):813852.

Breiman, L. (1995). Better subset regression using the nonnegative garrote. *Technometrics*, 37(4):373384.

Bro, R., Kjeldahl, K., Smilde, A. K., and Kiers, H. A. L. (2008). Cross-validation of component models: A critical look at current methods. *Analytical and Bioanalytical Chemistry*, 390(5):12411251.

Bro, R., Nielsen, H. H., Stefánsson, G., and Skåra, T. (2002). A phenomenological study of ripening of salted herring. assessing homogeneity of data from different countries and laboratories. *Journal of Chemometrics*, 16(2):8188.

Cadima, J. and Jolliffe, I. T. (1995). Loading and correlations in the interpretation of principle compenents. *Journal of Applied Statistics*, 22(2):203–214.

Croux, C., Filzmoser, P., and Fritz, H. (2013). Robust sparse principal component analysis. *Technometrics*, 55(2):202–214.

Dagum, L. and Menon, R. (1998). Openmp: an industry standard api for shared-memory programming. *Computational Science & Engineering, IEEE*, 5(1):46–55.

de Schipper, N. C. and Van Deun, K. (2018). Revealing the joint mechanisms in traditional data linked with big data. *Zeitschrift für Psychologie*, 226(4):212231.

Eddelbuettel, D. and François, R. (2011). Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18.

Eddelbuettel, D. and Sanderson, C. (2014). Rcpparmadillo: Accelerating r with high-performance c++ linear algebra. *Computational Statistics and Data Analysis*, 71:1054–1063.

Erichson, N. B., Zheng, P., Manohar, K., Brunton, S. L., Kutz, J. N., and Aravkin, A. Y. (2018). Sparse principal component analysis via variable projection.

Erichson, N. B., Zheng, P., Manohar, K., Brunton, S. L., Kutz, J. N., and Aravkin, A. Y. (2020). Sparse principal component analysis via variable projection. *SIAM Journal on Applied Mathematics*, 80(2):977–1002.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1).

Gajjar, S., Kulahci, M., and Palazoglu, A. (2017). Selection of non-zero loadings in sparse principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 162:160171.

Gordon, A. D., Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). Classification and regression trees. *Biometrics*, 40(3):874.

Gu, Z., de Schipper, N. C., and Deun, K. V. (2019). Variable selection in the regularized simultaneous component analysis method for multi-source data integration. *Scientific Reports*, 9(1):18608.

Gu, Z. and Van Deun, K. (2016). A variable selection method for simultaneous component based data integration. *Chemometrics and Intelligent Laboratory Systems*, 158:187–199.

Gu, Z. and Van Deun, K. (2018). Regularizedsca: Regularized simultaneous component analysis of multiblock data in r. *Behavior Research Methods*, 51(5):22682289.

Gu, Z. and Van Deun, K. (2018). *RegularizedSCA: Regularized Simultaneous Component Based Data Integration*. R package version 0.5.4.

Guo, J., James, G., Levina, E., Michailidis, G., and Zhu, J. (2010). Principal component analysis with sparse fused loadings. *Journal of Computational and Graphical Statistics*, 19(4):930946.

Halldorsdottir, T. and Binder, E. B. (2017). Gene Œ environment interactions: From molecular mechanisms to behavior. *Annual Review of Psychology*, 68(1):215–241.

Hansen, P. C. and OLeary, D. P. (1993). The use of the l-curve in the regularization of discrete ill-posed problems. *SIAM Journal on Scientific Computing*, 14(6):14871503.

Hastie, T., Tibshirani, R., and Friedman, J. (2009a). *The Elements of Statistical Learning*. Springer New York.

Hastie, T., Tibshirani, R., and Friedman, J. (2009b). The elements of statistical learning. *Springer Series in Statistics*.

Hastie, T., Tibshirani, R., and Tibshirani, R. J. (2017). Extended comparisons of best subset selection, forward stepwise selection, and the lasso.

Hunter, D. R. and Lange, K. (2004). A tutorial on mm algorithms. *The American Statistician*, 58(1):3037.

Héberger, K. (2010). Sum of ranking differences compares methods or models fairly. *TrAC Trends in Analytical Chemistry*, 29(1):101109.

Jenatton, R., Obozinski, G., and Bach, F. (2010). Structured sparse principal component analysis. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 9:366–373.

Jolliffe, I. T. (1986). *Principal Component Analysis*. Springer New York.

Kaiser, H. F. (1958). The varimax criterion for analytic rotation in factor analysis. *Psychometrika*, 23(3):187–200.

Kiers, H. A. (2002). Setting up alternating least squares and iterative majorization algorithms for solving various matrix optimization problems. *Computational Statistics & Data Analysis*, 41(1):157170.

Kiers, H. A. L. (2000). Towards a standardized notation and terminology in multiway analysis. *Journal of Chemometrics*, 14(3):105–122.

Kiers, H. A. L. and ten Berge, J. M. F. (1989). Alternating least squares algorithms for simultaneous components analysis with equal component weight matrices in two or more populations. *Psychometrika*, 54(3):467473.

Kowalski, M. and Torrésani, B. (2009). Sparsity and persistence: mixed norms provide simple signal models with dependent coefficients. *Signal, Image and Video Processing*, 3(3):251–264.

Lock, E. F., Hoadley, K. A., Marron, J. S., and Nobel, A. B. (2013). Joint and individual variation explained (jive) for integrated analysis of multiple data types. *The Annals of Applied Statistics*, 7(1):523542.

Lorenzo-Seva, U. and ten Berge, J. M. F. (2006). Tuckers congruence coefficient as a meaningful index of factor similarity. *Methodology*, 2(2):57–64.

Lourenco, J. M. and Lebensztajn, L. (2018). Post-pareto optimality analysis with sum of ranking differences. *IEEE Transactions on Magnetics*, 54(8):110.

Meng, C., Zeleznik, O. A., Thallinger, G. G., Kuster, B., Gholami, A. M., and Culhane, A. C. (2016). Dimension reduction techniques for the integrative analysis of multi-omics data. *Briefings in Bioinformatics*, 17(4):628641.

Natarajan, B. K. (1995). Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24(2):227234.

Nielsen, H., Bro, R., Stefansson, G., and Skåra, T. (1999). Salting and ripening of herring: collection and analysis of research results and industrial experience within the nordic countries. *TemaNord*, 578.

Paxton, A. and Griffiths, T. L. (2017). Finding the traces of behavioral and cognitive processes in big data and naturally occurring datasets. *Behavior Research Methods*, 49(5):1630–1638.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rasmussen, M. A. and Bro, R. (2012). A tutorial on the lasso approach to sparse modeling. *Chemometrics and Intelligent Laboratory Systems*, 119:2131.

Reinke, S. N., Galindo-Prieto, B., Skotare, T., Broadhurst, D. I., Singhania, A., Horowitz, D., Djukanovi, R., Hinks, T. S., Geladi, P., Trygg, J., and et al. (2018). Onpls-based multi-block data integration: A multivariate approach to interrogating biological interactions in asthma. *Analytical Chemistry*, 90(22):1340013408.

Rohart, F., Gautier, B., Singh, A., and Lê Cao, K.-A. (2017). mixomics: An r package for omics feature selection and multiple data integration. *PLOS Computational Biology*, 13(11):e1005752.

Schneider, B. and Waite, L. J. (n.d.). The 500 family study [1998-2000: United states].

Schouteden, M., Van Deun, K., Pattyn, S., and Van Mechelen, I. (2013). Sca with rotation to distinguish common and distinctive information in linked data. *Behavior Research Methods*, 45(3):822–833.

Schouten, T. M., Koini, M., de Vos, F., Seiler, S., van der Grond, J., Lechner, A., Hafkemeijer, A., Möller, C., Schmidt, R., de Rooij, M., and et al. (2016). Combining anatomical, diffusion, and resting state functional magnetic resonance imaging for individual classification of mild and moderate alzheimers disease. *NeuroImage: Clinical*, 11:4651.

Shen, H. and Huang, J. Z. (2008). Sparse principal component analysis via regularized low rank matrix approximation. *Journal of Multivariate Analysis*, 99(6):10151034.

Shu, H., Wang, X., and Zhu, H. (2019). D-CCA: A decomposition-based canonical correlation analysis for high-dimensional datasets. *Journal of the American Statistical Association*, 115(529):292–306.

Smilde, A., Bro, R., and Geladi, P. (2004). *Multi-Way Analysis with Applications in the Chemical Sciences*. John Wiley & Sons, Ltd.

Smilde, A. K., Måge, I., Naes, T., Hankemeier, T., Lips, M. A., Kiers, H. A. L., Acar, E., and Bro, R. (2017). Common and distinct components in data fusion. *Journal of Chemometrics*, 31(7):e2900.

ten Berge, J. M. (1993). *Least squares optimization in multivariate analysis*. DSWO Press, Leiden University Leiden, The Netherlands.

ten Berge, J. M. F. (1986). Some relationships between descriptive comparisons of components from different studies. *Multivariate Behavioral Research*, 21(1):29–40.

Tenenhaus, A. and Tenenhaus, M. (2011). Regularized generalized canonical correlation analysis. *Psychometrika*, 76(2):257–284.

Tibshirani, R. (1994). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288.

Tibshirani, R. (2011). Regression shrinkage and selection via the lasso: a retrospective. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(3):273–282.

Tibshirani, R., Johnstone, I., Hastie, T., and Efron, B. (2004). Least angle regression. *The Annals of Statistics*, 32(2):407499.

Timmerman, M. E., Kiers, H. A., and Ceulemans, E. (2016). Searching components with simple structure in simultaneous component analysis: Blockwise simplimax rotation. *Chemometrics and Intelligent Laboratory Systems*, 156:260–272.

Trendafilov, N. T. (2013). From simple structure to sparse components: a review. *Computational Statistics*, 29(3-4):431–454.

Trendafilov, N. T., Fontanella, S., and Adachi, K. (2017). Sparse exploratory factor analysis. *Psychometrika*, 82(3):778794.

Trygg, J. and Wold, S. (2002). Orthogonal projections to latent structures (o-pls). *Journal of Chemometrics*, 16(3):119128.

Van Deun, K., Smilde, A., Thorrez, L., Kiers, H., and Van Mechelen, I. (2013). Identifying common and distinctive processes underlying multiset data. *Chemometrics and Intelligent Laboratory Systems*, 129:4051.

Van Deun, K., Smilde, A. K., van der Werf, M. J., Kiers, H. A., and Van Mechelen, I. (2009). A structured overview of simultaneous component based data integration. *BMC Bioinformatics*, 10(1):246.

Van Deun, K., Van Mechelen, I., Thorrez, L., Schouteden, M., De Moor, B., van der Werf, M. J., De Lathauwer, L., Smilde, A. K., and Kiers, H. A. L. (2012). Disco-sca and properly applied gsvd as swinging methods to find common and distinctive processes. *PLoS ONE*, 7(5):e37840.

Van Deun, K., Wilderjans, T. F., van den Berg, R. A., Antoniadis, A., and Van Mechelen, I. (2011). A flexible framework for sparse simultaneous component based data integration. *BMC Bioinformatics*, 12(1):448.

Vassos, E., Di Forti, M., Coleman, J., Iyegbe, C., Prata, D., Euesden, J., OReilly, P., Curtis, C., Kolliakou, A., Patel, H., and et al. (2017). An examination of polygenic score risk prediction in individuals with first-episode psychosis. *Biological Psychiatry*, 81(6):470–477.

Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S*. Springer, New York, fourth edition. ISBN 0-387-95457-0.

Vervloet, M., Deun, K. V., den Noortgate, W. V., and Ceulemans, E. (2016). Model selection in principal covariates regression. *Chemometrics and Intelligent Laboratory Systems*, 151:26–33.

Vervloet, M., Wilderjans, T., Durieux, J., and Ceulemans, E. (2017). *multichull: A Generic Convex-Hull-Based Model Selection Method*. R package version 1.0.0.

Wang, L., Xiao, Y., Ping, Y., Li, J., Zhao, H., Li, F., Hu, J., Zhang, H., Deng, Y., Tian, J., and et al. (2014). Integrating multi-omics for uncovering the architecture of cross-talking pathways in breast cancer. *PLoS ONE*, 9(8):e104282.

Wang, W.-T. and Huang, H.-C. (2017). Regularized principal component analysis for spatial data. *Journal of Computational and Graphical Statistics*, 26(1):1425.

Wilderjans, T. F., Ceulemans, E., and Meers, K. (2012). Chull: A generic convex-hull-based model selection method. *Behavior Research Methods*, 45(1):115.

Willmott, C. J. and Matsuura, K. (2006). On the use of dimensioned measures of error to evaluate the performance of spatial interpolators. *International Journal of Geographical Information Science*, 20(1):89102.

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

Yang, D., Ma, Z., and Buja, A. (2014). A sparse singular value decomposition method for high-dimensional data. *Journal of Computational and Graphical Statistics*, 23(4):923942.

Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320.

Zou, H. and Hastie, T. (2018). *elasticnet: Elastic-Net for Sparse Estimation and Sparse PCA*. R package version 1.1.1.

Zou, H., Hastie, T., and Tibshirani, R. (2006). Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286.

Summary

Researchers are sometimes faced with a situation where they can supplement their data with other data types for the same individuals. For example, besides having questionnaire data, researchers might also have experience sampling data, online behavior data, or genetic data on the same subjects. We refer to each of the different data types as a data block. Linking multiple data blocks together holds promising prospects as it allows studying relationships as the result of the concerted action of multiple determinants. For example, having both questionnaire data on eating and health behavior and data on genetic variants for the same subjects holds the key to finding how genes and environment act together in the emergence of eating disorders. Indeed, for most psycho-pathologies and many other behavioral outcomes, it holds that these are the result of a genetic susceptibility in combination with a risk provoking environment (Halldorsdottir and Binder, 2017). Thus, analyzing multiple data blocks together could provide us with crucial insights into the complex interplay between the multiple factors that determine human behavior.

A powerful way of gaining insight into such data sets consisting of multiple blocks is by means of latent variable modelling techniques. One of such techniques is simultaneous component (SC) analysis, which is the main approach discussed in this thesis. There are two problems associated with the SC model:

1. Because the component scores of the SC model are a linear combination of variables of all blocks, all blocks contribute to all components. This is not particularly insightful as it obscures components that are not shared by all data blocks. In order to alleviate this problem, the common sources of variation need to be separated from the distinctive sources of variation. This serves two purposes: first, it increases efficiency of the estimation of the common components (Acar et al., 2014; Lock et al., 2013; Trygg and Wold, 2002), and second, it may be instructive (substantively) to detect such unique sources of variation (Alter et al., 2003; Van Deun et al., 2012). Our strategy will be to model the unique sources of variation by a weight vector containing zero(s) for all blocks except the block for which the component is unique. This imposes absence of the component in all blocks, except for

the one for which it is unique. This approach has been shown to have a clear interpretational advantage compared to methods that fail to control such absence (Schouteden et al., 2013; Van Deun et al., 2013).

2. Besides not accounting for the multi-block structure, the SC model does not imply sparsity into the weights; that is, there is no guarantee that a large portions of the weights gets values (close to) zero. Sparse weights are important for the interpretation of the results, which is especially helpful with data sets consisting of a (very) large number of variables, where a solution with many non-zero weights would be very difficult to interpret. This issue has been addressed in the context of single-block data by penalty based approaches such as the elastic net penalty (Zou et al., 2006), but the problem has not yet been tackled in conjunction with identifying common and distinctive source of variation in the SC model.

This thesis aimed at providing solutions to the above two problems.

In Chapter 2, we took a closer look at multi-block analysis with simultaneous component analysis (SCA). We argued that the current practice of analyzing multi-block data by merging all data and applying methods developed for a single block of data, is an inappropriate approach that does not guarantee the discovery of the joint relationships between blocks. Each block is dominated by specific information that is typical for the kind of processes it measures (e.g., behavioral processes and response tendencies in questionnaire data, biological processes in the genetic data) resulting in higher associations between the variables within a block than between blocks. Hence, an analysis that does not account for the multi-block data structure is highly unlikely to find the linked variables underlying the subtle joint mechanisms at play. In order to tackle this problem, we proposed a simultaneous component approach (Kiers, 2000; Van Deun et al., 2009) called SCaDS that introduces both proper constraints and regularization terms, including the LASSO, to account for the presence of dominant block-specific sources of variation and to force variable selection. We illustrated how to use SCaDS using publicly available data from the 500 Family Study (Schneider and Waite). The interpretational advantage of SCaDS was clearly shown. Furthermore, support for the superior performance of SCaDS compared to sparse PCA of the concatenated data in estimating back the model parameters was convincingly shown in a simulation study. Especially in situations where the number of variables was large compared to the number of observation units, SCaDS outperformed the approach of applying sparse methods for a single block of data while ignoring the multi-block structure. We used cross-validation as a tool to determine the status of the components (common or distinctive) and the strength of the LASSO penalty of the SCaDS method. For data generated in the low-dimensional setting satisfactory results were obtained; yet, in the high-dimensional setting we observed a bias towards overly complex models.

In Chapter 3 we examined various model selection procedures that can be used to select the hyper-parameters in sparse PCA and SCA. In order to get sparse weights in either PCA or SCA models, values for the hyper-parameters of the penalty terms need to be selected, which is a delicate process. If one chooses a value that is too small, too many coefficients will be selected making the interpretation of the models difficult, while choosing a value that is too high might result in missing the important relationships within and between data blocks. In Chapter 3, we compared various model selection procedures with respect to their ability of finding the hyper-parameter values yielding the correct structure of the data; i.e., selecting the right set of variables both in the single block setting and in the multi-block setting with common and distinct variation. The model selection procedures investigated are cross-validation with the Eigenvector method (Bro et al., 2008), BIC (Guo et al., 2010; Croux et al., 2013), Convex Hull (Wilderjans et al., 2012), and the Index of Sparseness (Gajjar et al., 2017; Trendafilov et al., 2017). In order to inform the analysis about the block structure of the variables, we implemented the group LASSO penalty in a block-wise fashion, aiming at either selecting or canceling out data blocks in an automated way. We concluded that when the component weights are to be interpreted, cross-validation with the one standard error rule is preferred; alternatively, if the interest lies in obtaining component scores using a very limited set of variables, the convex Hull, BIC, and index of sparseness are all suitable.

In Chapter 4, we presented a sparse PCA method relying on cardinality constraints instead of penalties. A well-documented disadvantage of using penalties for introducing sparsity into the coefficients is that these penalties are not intended to find the best subset of variables. That is, these penalties introduce bias in the estimates while reducing their variance. The resulting variable selection process increases the efficiency of the estimators, but it is not designed to recover the true underlying set of variables. To overcome this problem, we presented a cardinality constrained alternative to PCA. Instead of penalizing the coefficient in the model, we solved the problem of finding the optimal subset given a number of non-zero coefficients using a surrogate function. For this purpose, we used cardinality constrained regression, which has the sole aim of identifying the best subset of variables. We compared this cardinality constrained PCA (CCPCA) to sparse PCA with the LASSO penalty (Zou et al., 2006) estimated with the LARS algorithm. We found that CCPCA and sparse PCA performed relatively similar, where CCPCA has the edge when the noise levels are low. When noise levels are large, the bias-variance trade-off can lead to better coefficients for sparse PCA, but for the conditions in this chapter these differences were marginal. It is important to mention that sparse PCA with LARS is computationally more efficient than CCPCA, when the data contains a large number of variables, sparse PCA might be preferred.

In Chapter 5, we introduced an `R` package to perform regularized SCA and PCA with sparsity on the component weights. This package also includes model selection procedures. The procedures developed are based on the work presented in the previous chapters. The main parts of the computational code have been written in `C++` to provide maximal efficiency of the underlying numerical computations. The chapter is written as a tutorial with the aim of making the methods developed in this thesis accessible to potential users. It starts with a short introduction to PCA and its multi-block extension SCA, followed by a substantiation of the models that the procedures in this package estimate. After that the `R` implementation of the package is discussed, followed by detailed examples of data analysis and model selection.

Concluding, in Chapter 6 we presented some closing thoughts on the previous chapters, and presented directions future research can go in. Most notably, we shed some light on the difference between sparse weight based PCA/SCA compared to sparse loadings PCA/SCA, the former being the main interest of this thesis.

Dankwoord

Dit proefschrift is niet zomaar ineens tot stand gekomen, en ik maak graag van deze mogelijkheid om een aantal mensen te bedanken.

Allereerst wil ik Katrijn bedanken, zonder jou was dit proefschrift er niet gekomen. We hebben altijd fijn samengewerkt, je liet me vrij om mijn eigen ideeën uit te voeren. Het maakte niet uit op wat voor tempo ik dat deed, ik heb alle ruimte gekregen om me op diverse vlakken in de breedte kunnen ontwikkelen. Ik werd volledig vertrouwd en dat werkte prettig. Ik kon mijn eigen plan trekken, maar je was er daarna ook altijd om me te helpen, bijvoorbeeld bij het schrijven van een introductie. Ik heb daar altijd een hekel aan gehad, ik kreeg nauwelijks meer dan een enkel kantje op papier, en jij wist er toch altijd een prachtig verhaal van te maken (ook al vond jij het ook het minst leuke onderdeel van het schrijven). Daarnaast hebben we de afgelopen vier jaar fijn samen onderwijs gegeven. Je hebt in die korte tijd mooie cursussen in elkaar gezet, waar je trots op kan zijn!

Daarnaast wil ik Jeroen bedanken. Ik heb me door jou altijd gesteund gevoeld, ondanks dat we niet veel hebben samengewerkt. Ik weet dat ik op je kon rekenen mocht ik het nodig hebben gehad, en dat is een prettig gevoel.

Ik wil ook graag de commissie bedanken voor het lezen van mijn proefschrift. Bedankt dat jullie de tijd hebben genomen om je door die tekst heen te worstelen en dat jullie al dan niet in persoon bij mijn verdediging aanwezig zijn.

Herbert Hoijtink wil ik ook graag bedanken, jij hebt me overgehaald om in Utrecht statistiek te gaan studeren, en daar heb ik nooit spijt van gehad. Ik wil ook Dave Hessen bedanken voor het inspirerende statistiekonderwijs dat je geeft en hopelijk nog steeds geeft. Ook wil ik mijn studiegenoten, in het bijzonder Pieter en Wouter, bedanken, jullie waren en zijn een inspiratie voor mij.

Daarnaast wil ik mijn oud-collega's bedanken. Jules, Eva, Laura en Inga, bedankt voor alle gezellige wandelingen, we hebben samen plezier en steun aan elkaar gehad. Bedankt Chris en Robbie, voor de grote hoeveelheid plezier die we samen hebben gehad, en bedankt voor jullie kundige en inspirerende adviezen op het gebied van R, Linux en open source software in zijn algemeenheid. Daarnaast wil ik mijn oud-kamergenoten, Paul, Soogeun en IJsbrand bedanken, voor alle mooie leuke momenten in kantoor, maar vooral ook buiten kantoor. Ik wil ook graag de andere Collega's bij MTO bedanken: Michèle, Paulette, Erwin, Mattis, Ghislaine, Elise, Leonie, Hilde, Joris, Mihai, Amir, Olmo, Esther, Damiano en Felix en alle andere (Ex)-PhD's die ik nu vergeet en uiteraard ook de senior staff.

Last but not least! Wil ik onze lab groep de cool people club, bedanken: Pia, Davide, Zhengguo, Soogeun, Shuai, Rosember and Edoardo, voor alle mooie momenten die wesamen hebben gehad. Dat jullie nog maar veel mooi onderzoek mogen doen.

Ook wil ik mijn vrienden bedanken, Bas Bas en Wouter, voor het geven van het goede voorbeeld, zonder jullie zat ik nu huilend koffie te drinken op het gemeentehuis. Jan, Job, Tycho, Tito, Kiki, Elles, Raphael, Ouren, Arjun, Michel, Jip, Kevin en Sean, bedankt voor alle gezelligheid en steun. Ook mijn O.G. vrienden uit Eindhoven en Nijmegen bedankt, dat we nog veel mooie dingen met elkaar mogen beleven.

Lieve familie, jullie hebben me altijd gesteund. Bedankt voor alle liefde, dankzij jullie ben ik uiteindelijk zindelijk geworden. Dat is vrij praktisch gebleken, in de trein, maar ook op kantoor. Ook mijn schoonfamilie bedankt, jullie hebben me met open armen ontvangen en altijd gesteund.

Als laatste wil ik de belangrijkste persoon in mijn leven bedanken. Lieve Roos, dankjewel voor alle fijne jaren samen en alle jaren die nog komen gaan.