Chemometrics and Intelligent Laboratory Systems 156 (2016) 260–272

Contents lists available at ScienceDirect

Chemometrics and Intelligent Laboratory Systems

journal homepage: www.elsevier.com/locate/chemolab

Searching components with simple structure in simultaneous component analysis: Blockwise Simplimax rotation☆

Marieke E. Timmerman [a,*], Henk A.L. Kiers [a], Eva Ceulemans [b]

[a] University of Groningen, The Netherlands
[b] KU Leuven, Belgium

ARTICLE INFO

Article history:
Received 27 January 2016
Received in revised form 31 March 2016
Accepted 8 May 2016
Available online 13 May 2016

Keywords:
Sparse group
Simultaneous component analysis
Multiset data
Sensory profiling data

ABSTRACT

Simultaneous component analysis (SCA) is a fruitful approach to disclose the structure underlying data stemming from multiple sources on the same objects. This kind of data can be organized in blocks. To identify which component relates to all, and which to some sources, the block structure in the data should be taken into account. In this paper, we propose a new rotation criterion, Blockwise Simplimax, that aims at block simplicity of the loadings, implying that for some components all variables in a block have a zero loading. We also present an associated model selection criterion, to aid in selecting the required degree of simplicity for the data at hand. An extensive simulation study is conducted to evaluate the performance of Blockwise Simplimax and the associated model selection criterion, and to compare it with a sparse competitor, namely Sparse group SCA. In the conditions considered Blockwise Simplimax performed reasonably well, and either performed equally well as, or clearly outperformed Sparse group SCA. The model selection criterion performed well in simple conditions. The usefulness of Blockwise Simplimax and Sparse group SCA is illustrated using sensory profiling data regarding different cheeses.

© 2016 Elsevier B.V. All rights reserved.

1. Introduction

Combining multiple sources of data on the same objects is a fruitful approach to achieve an in-depth insight into the phenomena under study [1]. Boosted by technological advances, multiple data sources are available easier than ever, in many different fields of research [2]. For example, in metabolomics, the abundance of large numbers of biomolecules in samples of biological material, like plasma, may be measured using different analytical techniques as fluorescence spectroscopy and nuclear magnetic resonance spectroscopy [3]. In sensory profiling, food samples may be characterized based on instrumental measurements and various sensory attributes, as visual appearance, smell and taste [4]; also multiple panelists may rate the food samples [5]. The resulting data consist of multiple data blocks that are linked by the objects (e.g., samples of biological material, food samples), and are denoted as multiset data.

The core interest is often to achieve an insight into the differences between the objects, where some of these differences come to expression in all data blocks, whereas others are only revealed by specific sources. If each data block pertains to a myriad of variables, these differences can be summarized in an exploratory way using one of the available simultaneous dimension reduction methods. Those methods reduce the variables to a smaller number of components that represent the original data as well as possible [6,7]. The components are supposed to reflect the object differences that are present in all the different blocks, while some of the methods also identify components referring to block specific differences. For an in-depth insight into the differences between the objects and how they come about, it appears wise to use a method that identifies both shared and block specific components. Among the available approaches, the one taken in simultaneous component analysis (SCA) [8,9] is versatile and useful, as has been shown in a wide range of applications [10,11].

Different SCA variants have been developed. They share the key element that some of the model parameters are restricted to be identical across the blocks. The SCA variants can be categorized in various ways, according to the type of link between the data blocks (e.g., objectwise or variablewise linked), the specific model imposed [8,9,12], the scaling of the data blocks before the actual SCA analysis [6,13], the loss function used [14] and the rotation applied to identify the solution [15].

The interpretation of an SCA solution is often based on the loadings. In particular, in case of standardized orthogonal components, the loadings are covariances or correlations between the components and the variables. Since loadings equal to zero indicate the absence of a relationship, a solution with many loadings (close to) zero is favored in terms of

☆ The research leading to the results reported in this paper was sponsored in part by a research grant from the Fund for Scientific Research-Flanders (FWO, Project no. G.0582.14 awarded to Eva Ceulemans, Peter Kuppens and Francis Tuerlinckx), by the Belgian Federal Science Policy within the framework of the Interuniversity Attraction Poles program (IAP/P7/06), and by the Research Council of KU Leuven (GOA/15/003).

* Corresponding author at: University of Groningen, Grote Kruisstraat 2/1, 9712TS Groningen, The Netherlands.
E-mail address: m.e.timmerman@rug.nl (M.E. Timmerman).

http://dx.doi.org/10.1016/j.chemolab.2016.05.001
0169-7439/© 2016 Elsevier B.V. All rights reserved.

interpretability. To ease the interpretation, one may use a sparse SCA variant [14], in which parameters are shrunken toward zero, by adding a penalty to the default loss function. Alternatively, one may exploit the rotational freedom that many SCA variants have. That is, without loss of fit, the loadings of those SCA variants can be rotated toward a solution that is easier to interpret. Unlike in chemometrics, rotation is a default step in psychometrics. In psychometrics, many different rotation criteria have been proposed [16], which aim at finding a rotated solution with a simple structure. A simple structure is characterized by five rules for the ideal positioning of zero loadings [17]. Because in practice rotated solutions typically do not comply with all rules, different criteria may arrive at different rotated solutions. Since from a mathematical point of view all rotated solutions are equivalent, the interpretability is the key factor in selecting a rotated solution.

The standard rotation criteria operate upon the individual variables. Thus, standard criteria do not take into account the block structure present in the variables, and therefore may be less suitable for rotating SCA loadings (or even PCA loadings for which it is known that variables have a block structure), in such a way that block-specific object differences are optimally revealed. To overcome this limitation, we present a new rotation criterion that aims at block simplicity of the loadings, implying that for some components all variables in a block have a zero loading. We also propose an associated model selection criterion, to aid in selecting the required degree of simplicity for the data at hand. Specifically, the new rotation method can be seen as a blockwise version of the Simplimax rotation method [18], and will therefore be denoted as Blockwise Simplimax. Using the (Blockwise) Simplimax rotation method, a binary matrix **W** specifies which loadings are 'aimed to be small' and which are 'free' after rotation. With Blockwise Simplimax, this binary matrix thus gives an explicit expression of which object differences may be present in all the blocks and which object differences show up in a single or a few blocks only. Because the user does not have to inspect all loadings, but can see at a single glance which blocks of variables are associated with which component, this facilitates the interpretation tremendously. Further, if there are deviations from a perfect block structure, in that a (few) variable(s) within a block do not have a near zero loading whereas the rest does, then this can be identified, because this shows up in the rotated loadings.

First, we present the SCA model, including issues important for modeling as preprocessing the data and estimation. Then, we introduce the Blockwise Simplimax rotation method, including the rotation criterion and the model selection criterion. Then, we outline the main differences with Sparse group SCA [14], which is an alternative to SCA followed by Blockwise Simplimax rotation, based on a sparseness penalty. The performance of the Blockwise Simplimax algorithm is evaluated in a simulation study in terms of optimization and recovery, and its results applied to SCA loadings are compared to those from Sparse group SCA. The model selection procedure is tested in a second simulation study. The use of Blockwise Simplimax and Sparse group SCA is illustrated using empirical data from a sensory profiling study.

2. Theory

2.1. Simultaneous component model

We consider K data blocks $\mathbf{X}_k$ ($k = 1, \ldots, K$) containing the score of I objects on J_k variables ($j_k = 1, \ldots, J_k$), with $J = \sum_{k=1}^{K} J_k$ the total number of variables across the K blocks. Hence, the blocks are linked objectwise. The simultaneous component decomposition is given as

$$\mathbf{X}_k = \mathbf{F}\mathbf{P}_k^{\mathrm{T}} + \mathbf{E}_k, \tag{1}$$

for all k, with $\mathbf{F}$ ($I \times R$) containing the component scores of the I objects on the R components, $\mathbf{P}_k$ ($J_k \times R$) containing the loadings, and $\mathbf{E}_k$ ($I \times J_k$) the matrix with residuals. Because we focus on interpreting the loadings, the component score matrix $\mathbf{F}$ is here required to be columnwise orthonormal, i.e., $\mathbf{F}^{\mathrm{T}}\mathbf{F} = \mathbf{I}$ [14]. This way, the loadings are covariances (and correlations if the observed data $\mathbf{X}_k$ ($k = 1, \ldots, K$) is normalized per variable prior to analysis) between components and variables, provided that the observed data are columnwise centered, as usual. Further, the plots of loadings actually pertain to projections of variables on a subspace of R^I. The simultaneous component decomposition can also be written as

$$\mathbf{X} = \mathbf{F}\mathbf{P}^{\mathbf{T}} + \mathbf{E}, \tag{2}$$

with $\mathbf{X} = [\mathbf{X}_1 | \ldots | \mathbf{X}_K]$, $\mathbf{P}^{\mathrm{T}} = [\mathbf{P}_1^{\mathrm{T}} | \ldots | \mathbf{P}_K^{\mathrm{T}}]$ and $\mathbf{E} = [\mathbf{E}_1 | \ldots | \mathbf{E}_K]$. Without imposing further restrictions, the model is not identified. That is, without loss of fit, one may rotate each loading matrix $\mathbf{P}_k$ by postmultiplying it with an orthonormal matrix $\mathbf{T}$, provided that this rotation is compensated in the component score matrix $\mathbf{F}$, as

$$\mathbf{X}_k = \mathbf{F}\mathbf{T}\mathbf{T}^{\mathrm{T}}\mathbf{P}_k^{\mathrm{T}} + \mathbf{E}_k = \tilde{\mathbf{F}}\tilde{\mathbf{P}}_k^{\mathrm{T}} + \mathbf{E}_k, \text{ for } k = 1, \ldots, K \tag{3}$$

with $\mathbf{T}$ the rotation matrix, $\mathbf{T}\mathbf{T}^{\mathrm{T}} = \mathbf{I}$, $\tilde{\mathbf{P}}_k = \mathbf{P}_k\mathbf{T}$ the rotated loading matrix of block k, and $\tilde{\mathbf{F}} = \mathbf{F}\mathbf{T}$ the rotated component score matrix. To identify the model, one can position the axes in principal axes orientation [19]. Alternatively, one may select a set of rotated K loading matrices ($\tilde{\mathbf{P}}_k$, $k = 1, \ldots, K$, or equivalently, $\tilde{\mathbf{P}} = \mathbf{P}\mathbf{T}$) that are well-interpretable. Note that the orthogonality of $\mathbf{F}$ is not necessary, and thus oblique components can be allowed for. Then, the matrix $\mathbf{T}$ should meet the constraint that $\mathrm{diag}(\mathbf{T}^{-1}\mathbf{T}^{-1\mathrm{T}}) = \mathbf{I}$, to ensure that the components remain of length one after transformation.

The model is estimated by minimizing the least squares criterion

$$f(\mathbf{F}, \mathbf{P}_k) = \sum_{k=1}^{K} \left\| \mathbf{X}_k - \mathbf{F}\mathbf{P}_k^{\mathrm{T}} \right\|^2 = \tilde{f}(\mathbf{F}, \mathbf{P}) = \left\| \mathbf{X} - \mathbf{F}\mathbf{P}^{\mathrm{T}} \right\|^2 \tag{4}$$

subject to $\mathbf{F}^{\mathrm{T}}\mathbf{F} = \mathbf{I}$. The solution can be found using the singular value decomposition of $\mathbf{X}$, $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}}$, with $\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}$, $\mathbf{V}^{\mathrm{T}}\mathbf{V} = \mathbf{V}\mathbf{V}^{\mathrm{T}} = \mathbf{I}$, and $\mathbf{S}$ a diagonal matrix with the singular values in descending order on its diagonal, and taking $\mathbf{F} = \mathbf{U}_R$, and $\mathbf{P} = \mathbf{V}_R\mathbf{S}_R$, where the subscript R indicates that only the first R singular values or vectors are considered. The matrices $\mathbf{P}_1, \ldots, \mathbf{P}_K$ can be obtained by selecting the proper rows from $\mathbf{P}$.

As can be derived from the loss function in Eq. (4), the solution is influenced by the relative sum-of-squares of both the variables and the data blocks. That is, variables and data blocks with relatively large sum-of-squares influence the model estimates to a relatively large extent. For a variable, the sum-of-squares is determined by the size of the mean and variance, while for a data block it is also determined by its relative size, i.e., the number of variables involved. To equalize the weight of the variables and/or data blocks, it is common to apply centering, possibly combined with scaling to the observed data before analyzing the data blocks. The centering equalizes the means across variables. The scaling can be applied variable-wise, or blockwise, such that the variables or blocks are scaled to sum of squares one. For discussions about scaling in SCA we refer to [13] on variable-wise scaling and [6] on blockwise scaling. In what follows, we presume that the variables are mean-centered and not scaled, unless indicated otherwise.

2.2. Rotation in SCA

When considering rotation in SCA, the core question is what would make the loading matrices for the K blocks easy to interpret. The standard rotation criteria operate upon the individual variables. This may yield a rotated solution in which the clearly non-zero loadings of all components are scattered over the various blocks, implying that all components show up in all blocks. Such a solution fails to reveal systematic differences between blocks. For instance, it may remain unclear that one of the components does not play a role in a particular data block. To

Table 1
Simulated loadings, loadings after normalized Varimax rotation, Orthogonal Simplimax rotation and Blockwise Simplimax rotation. Values in absolute sense larger than .125 are printed in bold face; loadings that are rotated to a small value (i.e., associated with a value of 0 in W) are printed in a gray cell; Var nr is a variable number.

Var nr	Simulated Loadings				Normalized Varimax Loadings				Orthogonal Simplimax Loadings				Blockwise Simplimax Loadings			
1	**.39**	**.40**	.00	.00	**.24**	**.38**	.11	.08	**.29**	**.31**	-.01	**.20**	**.39**	**.27**	.02	-.00
2	**.73**	**.72**	.00	.00	.03	**.99**	.05	**.16**	**.36**	**.93**	.02	.04	**.42**	**.91**	-.07	.05
3	**.64**	**.39**	.00	.00	**.24**	**.39**	**.34**	**.20**	**.40**	**.30**	**.22**	**.26**	**.47**	**.28**	**.23**	.11
4	**.74**	.00	**.29**	.00	**.62**	**.14**	**.36**	.01	**.42**	.00	.03	**.59**	**.68**	-.09	**.22**	-.12
5	**.35**	.00	**.41**	.00	.12	.11	**.29**	.08	**.18**	.07	**.21**	**.19**	**.23**	.07	**.24**	.05
6	**.32**	.00	**.71**	.00	.00	.00	**.65**	-.04	.01	.01	**.54**	**.36**	**.14**	.02	**.63**	-.02
7	**.70**	.00	.00	**.44**	**.47**	**.13**	.02	**.37**	**.60**	-.02	-.09	**.13**	**.55**	-.05	-.11	**.25**
8	**.28**	.00	.00	**.63**	**.14**	-.00	**.23**	**.49**	**.48**	-.09	**.27**	-.06	**.28**	-.04	**.15**	**.46**
9	**.29**	.00	.00	**.68**	.01	-.01	-.07	**.52**	**.40**	-.07	.08	**-.32**	**.10**	-.00	-.12	**.50**
10	**.52**	.00	.00	**.62**	**.41**	**.18**	.08	**.46**	**.64**	.03	.01	.07	**.55**	.02	-.05	**.35**
11	**.29**	.00	.00	**.45**	.10	.09	-.07	**.31**	**.31**	.03	-.03	**-.13**	**.17**	.04	**-.13**	**.26**
12	**.44**	.00	.00	**.46**	**.31**	.12	.07	**.42**	**.54**	-.00	.04	.02	**.42**	.00	-.04	**.33**
13	**.58**	.00	.00	**.53**	**.34**	**.27**	-.04	**.43**	**.59**	**.14**	-.08	-.02	**.48**	.12	**-.17**	**.32**
14	**.46**	.00	.00	**.48**	**.31**	.06	.05	**.38**	**.50**	-.05	.01	.03	**.39**	-.05	-.05	**.30**
15	**.27**	.00	.00	**.68**	**.14**	**.13**	.04	**.38**	**.41**	.05	.07	-.09	**.26**	.08	-.04	**.33**
16	**.35**	.00	.00	**.66**	-.09	.07	**.20**	**.69**	**.52**	.01	**.39**	**-.33**	**.14**	.12	**.14**	**.69**
17	**.47**	.00	.00	**.29**	**.16**	**.14**	**.17**	**.33**	**.39**	.07	**.16**	.02	**.30**	.08	.09	**.28**
18	**.28**	.00	.00	**.41**	**.14**	.12	.01	**.26**	**.32**	.05	.02	-.03	**.23**	.06	-.05	**.22**

solve this problem, we propose to pursue block simplicity in the rotation, implying that we try to find a solution with as many near zero loading *vectors* as possible, where the vectors pertain to blocks of variables. For example, suppose we have $K = 4$ blocks, each consisting of J_k variables, and $R = 4$ components. A block simplicity rotated loading matrix $\tilde{\mathbf{P}}$ would contain values similar to, for instance,

$$\mathbf{G} = \begin{bmatrix} \times & \times & 0 & 0 \\ \times & 0 & \times & 0 \\ \times & 0 & 0 & \times \\ \times & 0 & 0 & \times \end{bmatrix}, \tag{5}$$

where **0** indicates a vector of zero loadings for a block, and × a vector of free (in practice clearly non-zero) loadings for a block. It can be seen that then component 1 plays a substantial role for all blocks involved, whereas component 2 is only important for block 1, component 3 only for block 2, and component 4 only for blocks 3 and 4. Note that it is also possible that more than a single component relates freely to one or more blocks, and zero to the other blocks; for example one might have a structure as in **G** (see Eq. (5)), but with the second column repeated once. In those cases, there is rotational freedom within the block.

Now, the question is how block simplicity, as represented in Eq. (5), can be achieved. Standard rotation criteria could be applied to the concatenated loading matrix $\mathbf{P} = [\mathbf{P}_1^T | \ldots | \mathbf{P}_K^T]^T$. If the block-simple structure is perfect (e.g., a pattern as in **G**, with loadings exactly equal to zero), a standard rotation criterion may reveal the block structure. However, in the presence of noise, a block structure can be masked and hence not be found by methods that do not use the presence of a block structure in the variables. To illustrate this, we searched for a data set that highlights such masking. Specifically, we used a simulated data set with an underlying structure with 8 vectors of zero loadings according to **G** (see Eq. (5)). The blocks represented $J_1 = 3, J_2 = 3, J_3 = 6$ and $J_4 = 6$ variables. The free loadings (i.e., × vectors) were sampled uniformly from the interval [025, 0.75]. The constructed loading matrix **A** is given in the first panel of Table 1. Next a random sample of 100×4 component scores (in **F**) and of 100x18 noise values (in **E**) was drawn from the standard normal distribution, and a data matrix was constructed as $\mathbf{X} = \mathbf{F}\mathbf{A}^T + \mathbf{E}$, while scaling **E** such that the ratio of the expected variance of **E** to the expected variance of **X** equals 0.5.[1] These data were analyzed with SCA, using 4 components, and the resulting loadings were rotated by the simple structure rotation method Normalized Varimax [20]. The results are given in the second panel of Table 1; the explanation of the contents of the third and fourth panel follow at the end of Sections 2.2.1 and 2.2.2, respectively.

As can be seen in Table 1, clearly, the blockwise structure is not fully recovered by Normalized Varimax. This can be seen most easily when looking at the small values, which are printed in regular font (large values are set in bold). We chose as threshold "in absolute sense smaller than 0.125"; this choice is somewhat arbitrary, but was taken because it nicely highlights differences between the results in the different panels, as will become clear later. Inspecting the values in the Normalized Varimax solution, we see that it only once recovers a zero vector completely, and in 12 cases, loadings that should be small, actually were not (i.e., exceeded the threshold and thus are printed in bold). Also, it can be seen that the solution is far from similar to the constructed loading matrix.

[1] The data construction procedure is the same as the one used in the simulation study in Section 3, and we inspected several such data sets in order to arrive at a data set highlighting the masking.

In the following paragraphs, we propose our new rotation criterion that explicitly aims at block simplicity, and a model selection method that aims to help identifying the suitable degree of required simplicity for the empirical data at hand. We start by recapitulating the Simplimax criterion [18], since our new rotation criterion builds on Simplimax.

2.2.1. Simplimax

Simplimax aims at rotating a matrix such that it optimally resembles a matrix with exactly p zero elements, where the number p is specified by the researcher, and the positions of the p zeros are optimized by the procedure. Simplimax was proposed for oblique rotation, but it can also handle the constrained version of orthogonal rotation. Here, we prefer to use orthogonal rotation, because using orthogonal components **F**, we can interpret **P** as covariances or correlations between the components and the variables, and contributions of components to the total fit are unequivocally defined. In particular, Simplimax for orthogonal rotation minimizes

$$g(\mathbf{T},\mathbf{G}) = \|\mathbf{PT}-\mathbf{G}\|^2, \tag{6}$$

over **T** and **G** ($J \times R$), where **T** is subject to $\mathbf{T}^T\mathbf{T} = \mathbf{I}$, and **G** has p zero elements and ($JR - p$) arbitrary elements. Thus, the rotation aims at finding the target matrix **G** with p zero elements, which can be approximated best by the rotated loading matrix **PT**.

Making use of a binary matrix **W** of the same size as **G**, with $w_{jr} = 0$ if $g_{jr} = 0$, and $w_{jr} = 1$ if g_{jr} is arbitrary, we optimize over **W** and **T**. That is, minimizing Eq. (6) over the arbitrary elements of **G** obviously comes down to setting these arbitrary elements equal to the associated values of **PT**, implying that their contribution to the loss equals 0. Thus, it remains to minimize the sum of squares of the values of **PT** for $w_{jr} = 0$. Writing $\mathbf{A} = \mathbf{PT}$ we thus have to minimize [18]

$$\tilde{g}(\mathbf{W},\mathbf{T}) = \sum_{j=1}^{J}\sum_{r=1}^{R}(1-w_{jr})a_{jr}^2, \tag{7}$$

where a_{jr} are the entries of the matrix $\mathbf{A} = \mathbf{PT}$.

The **T** matrix that minimizes the function in Eq. (7) can be found using the general procedure described by Browne [21]. Alternatingly updating **W** (by setting those values in **W** that correspond to the p smallest squared values of **A** equal to zero, and setting all other values equal to 1) and updating **T** by Browne's procedure, we obtain the algorithm presented in [18]. The algorithm can fairly easily hit a local minimum, so typically many runs are used with different starting values, to increase the chance of finding the globally optimal solution. In terms of Eq. (7), Simplimax rotation can be seen as the rotation that finds those components for which the sum of squares of the p loadings that are required to be small, is smallest. Thus, the rotation aims to 'make' p loadings as small as possible (in the sense of their sum of squares).

Using Simplimax in empirical practice requires the choice of p, the number of zeros in **W**. Kiers [18] recommended to consider the sequence of function values associated with a range of increasing values for p, and select the value for p for which it holds that the function value of the $(p + 1)$-th solution is considerably larger than that of the p-th solution. The range of values to consider is not prescribed. However, the smallest value for p that is useful in case of an orthogonal rotation equals $p_{\min} = .5(R - 1)R + 1$. This is so because an orthogonal rotation can always be performed such that one has at least $.5(R - 1)R$ zeros, thus yielding a function value of zero. Having more than $(J - 1)R$ zeros is not useful either, as one would then have minimally one column with zero loadings, making the component concerned superfluous.

We have applied Orthogonal Simplimax to the example data from Section 2.2.1. We chose $p = 36$, because the eight vectors with zeros in the underlying loading matrix had 3, 3, 3, 3, 6, 6, 6, and 6 values, respectively, hence totaling 36 zeros. The results of Orthogonal Simplimax are given in the third panel of Table 1. The gray cells indicate which 36 loadings are rotated toward to zero. These loadings do not correspond well with the loadings that should be zero (first panel).

As can be seen in Table 1, even though Simplimax works better than Normalized Varimax, it still does not fully recover the blockwise structure in the loadings. Specifically, it recovers two of the zero vectors completely, but still finds 10 loadings above the threshold, while they actually should be small. The Simplimax solution comes closer to the true loading matrix than Normalized Varimax does, but for the fourth component still deviates dramatically.[2]

2.2.2. Blockwise Simplimax

To rotate toward block simplicity, we propose to use Eq. (7) again, but now with the restriction that variables belonging to the same block are all treated in the same way. Specifically, we propose to minimize

$$\tilde{g}^b(\mathbf{W},\mathbf{T}) = \sum_{k=1}^{K}\sum_{r=1}^{R}J_k^{-1}(1-w_{kr})\|\mathbf{a}_{kr}\|^2, \tag{8}$$

with **W** ($K \times R$) a binary weight matrix, and where the main differences between Eqs. (7) and (8) are that variables (indicated by j) are replaced by blocks of variables (indicated by k) and the squared elements $a_{jr}{}^2$ are replaced by $\|\mathbf{a}_{kr}\|^2$, which denotes the sum of values $a_{jr}{}^2$ for variables j within block k. The only additional modification is the "normalizing" division by J_k, the number of variables in block k, in order to avoid a tendency of the method to select blocks with few variables, since it is easier to make sums of squares of such small blocks small. This normalizing division makes sense when the SCA is performed on the observed data itself, or on data after variable-wise scaling. In the case of blockwise scaling we recommend to refrain from the division, because this type of scaling already accounts for differences in block size in estimating the SCA solution. Thus, Blockwise Simplimax can be seen as the rotation that finds those components for which the sum of the mean of squares of the p loading vectors that should be small, is smallest. Thus, the rotation aims to 'make' p block-specific vectors of loadings as small as possible (in terms of their mean of squares).

To minimize Eq. (8) we employ an alternating least squares algorithm in which alternately **T** is updated keeping **W** fixed, and **W** is updated keeping **T** fixed. The former problem can be seen as a special case of Browne [21], where it should be noted that first the blockwise version Eq. (8) should be rewritten in the variablewise version Eq. (7). Using Browne's method implies replacing the loadings by $J_k^{-\frac{1}{2}}p_{jr}$, and deriving from the matrix **W** ($K \times R$) containing information on the blocks, a different and larger binary matrix of size ($J \times R$) with elements for *each variable* specified as 1 if it should be rotated to a small value, and 0 otherwise. (Note that zeros in *this* binary matrix used by Browne, have the opposite meaning as those in our matrix **W**, which may be confusing while reading the computer program).

The latter problem, of minimizing Eq. (8) over **W**, can be handled just as in Simplimax, as follows. The matrix **W** that minimizes Eq. (8) subject to the constraint that the number of zeroes in **W** is p is the one in which those p elements of w_{kr} are set equal to zero that correspond to (i.e., have the same indices as) the p smallest values of $J_k^{-1}\|\boldsymbol{a}_{kr}\|^2$.

After each update of **T** and **W**, the loss function value Eq. (8) is computed. If the difference in loss is smaller than the convergence criterion (e.g., 10^{-10}) multiplied by the current function value, or the maximal number of iterations is reached (e.g., 500) the algorithm stops. The block rotation algorithm ensures that the loss is a non-increasing function of the iterations.

The ALS algorithm has to be initialized by providing initial positions for the p zero loading vectors. To reduce the probability of ending up in a local minimum, a multistart procedure is applied, with different random

[2] To be sure, in this case for Simplimax 1000 random starts were used, 57 of which found the same lowest function value, so there is hardly any reason to believe that we deal with a local optimum here.

and rational initializations. The solution with the lowest loss value is retained. As a rational start, we use a Varimax rotation based initialization. This is found by first applying a normalized Varimax rotation on the loading matrix **P**, and then finding **W** such that the loss function Eq. (8) is minimized.

To illustrate that Blockwise Simplimax can make quite a difference, we return to the simulated example from Section 2.2.1 (see Table 1). From the fourth panel, which contains the Blockwise Simplimax rotated loadings, it can be concluded that the block structure was now recovered almost perfectly. The gray cells correspond completely with the zero values in the first panel. In total, only 5 loadings that 'should be' small are above threshold, which is clearly much better than when using Normalized Varimax and Orthogonal Simplimax (12 and 10, respectively, above threshold).

2.2.3. CHull: selecting the suitable degree of simplicity

A crucial step in applying the Blockwise Simplimax criterion is to select the number p, i.e., the number of vectors for which the loadings are as small as possible, thereby indicating the required degree of simplicity. To this end, we identify the model that optimally balances the goodness of fit and the complexity of the model. We do so using CHull, a numerical convex hull based procedure that is generally applicable for model selection [22,23]. To apply CHull, one fits a series of models to incorporate in the comparison, and computes for each model a complexity measure and a goodness of fit measure. Then, the CHull procedure consists of (1) determining the convex hull of the plot of the complexity by goodness of fit measures and (2) identify the model(s) for which increasing the complexity increases the fit little and decreasing the complexity reduces the fit considerably. CHull results in a rank ordering of the models considered. In this way, CHull can be used to indicate 'the optimal model', as well as a series of promising models.

We implemented the steps of the CHull procedure as described in Wilderjans et al. [23] using the following specific choices. As a complexity measure we use the number of free (i.e., not-targeted) loading vectors $c = KR - p$. Note that the number of zero target vectors p is inversely related to the model complexity. As a goodness of fit measure, we use the percentage of fit as $100 \times (1-\tilde{g}^b/\|\mathbf{A}\|^2)$, where $\tilde{g}^b$ denotes the loss function value of Blockwise Simplimax (Eq. (8). With respect to the series of models to submit to the CHull procedure, we considered all models in the range c_{min},$c_{min}+1$, …, c_{max}. As the value for c_{min}, we used $c_{min} = KR - p_{max}$, with p_{max} the smallest number of zero target vectors for the data at hand that is associated with at least one column of zero vectors. Such a solution is deemed uninteresting, because making entire vectors of loadings as small as possible would reduce the number of components considered. To select a reasonable maximal complexity, one should take into account that "… in most cases there is a degree of complexity, which often exceeds the complexity of the optimal model, after which adding more complexity does not result in large differences in fit." [23]. To avoid this, we discarded models with a large complexity c when their fit is less than 1% higher than the model with complexity $c-1$. That is, we selected c_{max} by sequentially fitting the models with $c = KR$,$(KR-1)$, …, and setting c_{max} as the lowest value for c after which the increase in fit for all models with higher complexity is 1% or lower.

The CHull procedure is meant as a heuristic to identify one or more solutions with a sharp decrease in fit, which may help the user in selecting p. The ultimate choice of p should not be based only on this, but also on interpretability of solutions and possibly other model desiderata. MATLAB code implementing the block rotation and the CHull model selection algorithms can be obtained from the first author upon request.

2.3. Sparse SCA using the lasso

A framework for sparse SCA of objectwise linked data blocks has been proposed [14]. Adopting this framework, one can impose sparseness on the loadings, using different kinds of penalties, possibly in combination with each other. Depending on the specific combination, this results in sparse SCA with the known sparse approaches elitist, lasso, ridge and group lasso. The effect of imposing a lasso or a ridge penalty is a shrinkage of the parameter estimates. Unlike the ridge penalty, the lasso penalty results in zero coefficients, as the penalty parameter becomes large enough. Specifically, while the penalty parameter increases, increasingly more values become exactly zero. Because we aim at achieving simple structure loadings, which involve zero elements in the loading matrix, we consider the lasso based penalties only. Sparse SCA considering only the lasso based penalties on the loadings boils down to:

$$h(\mathbf{F},\mathbf{P}_k) = \sum_{k=1}^{K}\left(\left\|\mathbf{X}_k - \mathbf{F}\mathbf{P}_k^T\right\|^2 + \lambda_L\|\mathbf{P}_k\|_1 + \lambda_G\sqrt{J_k}\|\mathbf{P}_k\|_2 + \lambda_E\|\mathbf{P}_k\|_{1,2}\right), \quad (9)$$

with λ_L the lasso penalty parameter, $\|\mathbf{P}_k\|_1 = \sum_{j_k=1}^{J_k}\sum_{r=1}^{R}|p_{j_k r}|$, λ_G the group lasso parameter,[3] $\|\mathbf{P}_k\|_2 = \sum_{r=1}^{R}\left(\sum_{j_k=1}^{J_k}p_{j_k r}^2\right)^{0.5}$, λ_E the elitist parameter and $\|\mathbf{P}_k\|_{1,2} = \sum_{r=1}^{R}\left(\sum_{j_k=1}^{J_k}|p_{j_k r}|\right)^2$. The lasso parameter λ_L operates upon the individual variables, and hence does not take into account the block structure. In contrast, the group lasso parameter λ_G operates upon the blocks directly. At the block level, it behaves as the lasso. This implies that, as the parameter becomes large enough, all loadings on a certain component of a particular block become equal to zero. Within the blocks, the group lasso behaves as the ridge, implying that individual loadings within a block are only shrunken toward zero, and not become zero (unless all loadings on a component become zero, because that would be the effect of the lasso operating at the block level). The elitist parameter has an effect opposite to the group lasso parameter: At the block level, the elitist behaves as the ridge, and within the blocks as the lasso. This implies that it steers certain individual loadings within a block toward zero, and yields only shrinkage across blocks.

The application of sparse SCA requires the selection of the different penalty parameter values. In practice, this can be done by estimating a range of sparse SCA models, across certain ranges of penalty parameters, and selecting a sparse SCA model with good fit and interpretability. However, even with the guidelines given by Van Deun et al. [14], one may end up with a series of solutions to interpret, and it can be a tedious task to select a final model. To reduce the number of estimated sparse SCA models to consider, the range of penalty parameters used should be selected carefully. First, because the group lasso and the elitist have opposite effects, it does not seem useful to apply both penalties together, and either one should be kept at zero. Second, prior knowledge could guide which penalty parameters could be reasonably set at zero. Because we are interested in achieving a simple block structure, it seems warranted to only include *only* the group lasso penalty, and thus set the elitist and lasso penalty parameters equal to zero. We will denote this variant as Sparse group SCA.

3. Simulation studies

In this section we first present a simulation study to evaluate the performance of the Blockwise Simplimax algorithm when the number p (i.e. the number of loading vectors for which the loadings are required to be as small as possible) is known. We also compare the results of SCA followed by Blockwise Simplimax to those obtained with Sparse group SCA. Next, we present a simulation study to evaluate the performance

[3] In fact, Van Deun et al. [14] present the formulas per component (see p. 3 just above Eq. (9)), and from their paper it does not become clear whether summation over r is in or outside the parentheses for the elitist and the group lasso penalties. Van Deun (personal communication, 15-01-2016) and the program we obtained, confirmed that summation has to be done outside the parentheses, leading to the above expressions.

of the CHull procedure for Blockwise Simplimax model selection (i.e., selecting the number p).

3.1. Simulation study 1

3.1.1. Problem

The first simulation study focuses on optimization and recovery. With regard to optimization, we will examine how sensitive the algorithms are to local minima. With regard to recovery, we will determine to what extent the algorithms succeed in disclosing the position of the ones and zeros in the binary matrix **W**, and in disclosing the true (simple) loading matrix. Moreover, we will investigate how the performance is influenced by four factors. For factor (1), the equality of block sizes, we expect that the performance with equal blocks would be better than or similar with unequal blocks, where the latter is based on the fact that we weight the blocks according to the number of variables that they include. Factor (2) pertains to the amount of error on the data and factor (3) to the sample size. For these factors we expect that the performance will deteriorate with increasing amount of error and decreasing sample size. Factor (4) pertains to the structure of the loading matrix and reflects the complexity of the underlying block structure. We presume that this complexity becomes larger when the number of zero target vectors per block reduces, and when the block structure is violated in that some variables have a non-zero loading on a component while the zero vectors indicate a zero.

3.1.2. Design

Each simulated data matrix was generated according to Eq. (1). The component score matrix **F** and the residual matrices $\mathbf{E}_k$ were randomly sampled from multivariate normal distributions, with $\mathbf{F} \sim N(\mathbf{0},\mathbf{I})$, and $\mathbf{E}_k \sim N(\mathbf{0},\mathbf{D}_k)$, where $\mathbf{D}_k$ is a diagonal matrix. The diagonal elements of $\mathbf{D}_k$ were chosen such that each variable has the required proportion of expected residual variance to the expected variance of **X** (factor 2). The number of components (R) and the number of blocks (K) were both fixed at 4. The number of variables per block (J_k) was fixed at either 3 or 6.

Four factors were systematically varied in a complete factorial design (with number of levels):

1. Equality of block size (2): Equal (with $J_k = 6$, $k = 1, \ldots, 4$); Different (with $J_k = 3$, $k = 1,2$; $J_k = 6$, $k = 3,4$)
2. Error level, the proportion of expected residual variance to the expected variance of **X** (2): .25; .50;
3. Sample size (2): $I = 100$; 500
4. Loading structure (4): Easy, Moderate, Difficult, Very Difficult.

The loading matrices were generated as follows. In the Easy condition, the true weight matrix $\mathbf{W}^{\text{true}}$ (i.e., $\mathbf{W}^{\text{true}}$ equals **W** in Eq. (8)) was an identity matrix. This implies that all variables within a block were uniquely associated to one component, thus yielding $p = 12$ true zero vectors. In the Moderate condition, the matrix was $\mathbf{W}^{\text{true}} = \begin{bmatrix} 1&0&0&0 \\ 1&1&0&0 \\ 0&0&1&0 \\ 0&0&1&1 \end{bmatrix}$, implying that two blocks were associated with one component, and two blocks were associated with two components. In the Difficult condition the matrix was $\mathbf{W}^{\text{true}} = \begin{bmatrix} 1&1&0&0 \\ 1&0&1&0 \\ 1&0&0&1 \\ 1&0&0&1 \end{bmatrix}$. This implies that all variables within a block were associated with two components; one component was shared among all blocks, two components were uniquely associated with one block, and one component was associated with two blocks. This structure is deemed fairly difficult to recover, because the information in the last three components jointly can be expected to somewhat overlap that in the first component, so it may be difficult to disentangle this information. The true loading matrices $\mathbf{A}^{\text{true}}$ were generated by setting a value of 0 for the loadings associated with a value of 0 in $\mathbf{W}^{\text{true}}$, and sampling other elements from unif(.25:.75), that is, the uniform distribution on the interval [.25:.75]. In the Very Difficult condition, we included a consistent violation of the perfect block structure. Departing from the same true loading matrix $\mathbf{A}^{\text{true}}$ as in the Difficult condition, we used the following structure as a basis for the loading matrix: $\begin{bmatrix} \times & \times & \boldsymbol{n} & \boldsymbol{n} \\ \times & \mathbf{n} & \times & \boldsymbol{n} \\ \times & \mathbf{n} & \mathbf{n} & \times \\ \times & \mathbf{n} & \mathbf{n} & \times \end{bmatrix}$, with 2/3 of the variables in **n** having zero-loadings, and 1/3 having a value sampled from unif(.125:.250). In the Difficult and Very Difficult conditions, the number of true zero vectors was $p = 8$; in the Moderate condition $p = 10$.

For each cell of the factorial design, 100 data matrices **X** were generated according to Eq. (1), yielding 3,200 data matrices. We centered each simulated data matrix, and subsequently applied (1) SCA followed by Blockwise Simplimax rotation to the SCA component loadings and (2) Sparse group SCA.

For (1), SCA was applied to each matrix, using the true numbers of components R. Subsequently the loading matrix was rotated with the Blockwise Simplimax algorithm, using the true number of zero target vectors p, and 102 starts. We used 100 random starts, 1 rational varimax-based start, and 1 true start, in which the true configuration of zeroes was used. The latter start was included to examine the sensitivity to local minima; the estimated solution that was retained was the one with lowest loss resulting from the 100 random runs and the single rational start.

For (2), the simulated data matrices were subjected to Sparse group SCA, again using the true number of components R. We used the MATLAB code provided as Additional files by Van Deun et al. [14], using their default settings, including 20 random starts. Like Van Deun et al., we normalized λ_G by taking it equal to $f_G\|\mathbf{X}\|^2/\sum_k\|\mathbf{P}_k^{\text{init}}\|^2$ where $\mathbf{X} = [\mathbf{X}_1|\ldots|\mathbf{X}_K]$ and $\mathbf{P}_k^{\text{init}}$ is obtained by unconstrained SCA. To select a proper value for the group lasso parameter f_G, which would enable a fair comparison between the Sparse group SCA and the Blockwise Simplimax solutions, we selected f_G such that the estimated loading matrix had p zero block vectors.[4] To find a value for f_G yielding Sparse SCA loadings with p zero block vectors, we used the bisection method [24] starting with the extreme values of $f_G = 10^{-6}$ and $f_G = 1$, respectively. In cases where that failed, we even took $f_G = 10^{-10}$ and $f_G = 100$.

Before presenting the results, it seems worth noting that computation time[5] of Blockwise Simplimax rotation (with 100 random starts) turned out to be 3 to 5 times shorter than a single Sparse group SCA analysis (with 20 random starts). Since in practice quite a few analyses have to be done (i.e., for determining p or f_G) this makes SCA (which in itself takes very little time) followed by Blockwise Simplimax an attractive competitor to Sparse group SCA. But of course, what ultimately counts is quality of recovery, see the next section.

3.1.3. Results

3.1.3.1. Optimization: sensitivity to local optima. In this section, we investigate how sensitive Blockwise Simplimax and Sparse group SCA are to local minima, and thus give an indication how likely it is that they fail to identify the global minimum of the loss function. Because in the presence of residuals the global optimum is unknown, we resort to a

[4] Note that values concerned gradually tend to 0 as f_G increases, and do not become exactly zero. To recognize such very small values as zeroes, values were set equal to 0 whenever they are, in absolute sense, smaller than .0005.

[5] Computation times are platform and implementation dependent, but to give some indication: typical computation times for data with 24 variables were 0,6 sec for 100 Blockwise Simplimax runs, and 5.3 s for 20 Sparse Groups SCA runs. Interestingly, there seems to be no serious practical time limitation to Blockwise Simplimax: 100 runs of a random loading matrix of size 200x10, with 7 blocks, and $p = 25$ still took only 18 s.

Table 2
Mean Proportion of starts across data sets and starts for which a suboptimal solution was found, per level of the manipulated factors, for the Block Simplimax algorithm (random, rational start) and the Sparse group SCA algorithm (random starts only).

Factor	Level	Blockwise Simplimax rational start	Blockwise Simplimax random start	Sparse group SCA — random start
Block size	Equal	.39	.60	.84
	Different	.40	.59	.86
Loading structure	Easy	.00	.07	.58
	Moderate	.05	.73	.94
	Difficult	.76	.79	.94
	Very Difficult	.78	.80	.94
Error level	.25	.42	.63	.84
	.50	.37	.56	.86
Sample size	100	.40	.59	.81
	500	.39	.60	.89

proxy for the global minimum.[6] For the Blockwise Simplimax algorithm, we take as the proxy the solution associated with the minimal loss function out of 102 starts, pertaining to 100 random, 1 rational and 1 true configuration starts. For the Sparse group SCA, we take as the proxy the best solution of the 20 random starts. A solution is considered suboptimal if the loss value is higher than $(1 + 10^{-6})f_{\text{proxy}}$, with f_{proxy} the loss value of the proxy.

For Blockwise Simplimax, it turned out that the estimated solution (i.e., the best out of 1 rational and 100 random starts) had a larger loss value than the proxy, and thus pertained to a local minimum for sure, for only 1 out of the 3,200 data sets. To further assess the sensitivity of the Blockwise Simplimax and Sparse group SCA algorithms, we considered the mean proportion of starts across data sets for which a suboptimal solution is found, per level of the four manipulated factors, per algorithm and type of start (i.e., Blockwise Simplimax with 1 rational start; Blockwise Simplimax with 100 random starts and Sparse group SCA with 20 random starts).

As can be seen in Table 2, suboptimal solutions occur least frequently in the condition with an Easy Loading structure, for both algorithms and both type of starts. In the Easy and Moderate conditions, the rational (Varimax) start rarely hits a local optimum, while this happens most of the time in the Difficult and Very Difficult conditions. For both methods, the other manipulated factors had a relatively small influence on the proportion of suboptimal solutions. The Blockwise Simplimax algorithm appears to be clearly less sensitive to suboptimal solutions than the Sparse group SCA algorithm, with overall proportions of suboptimal solutions resulting from a multiple random start procedure (across all conditions) equal to .60 and .85, respectively. Further, this performance difference holds in every cell of the design.

For the Sparse group SCA algorithm, the proportion of suboptimal solutions appeared to be .94 in the Moderate, Difficult and Very Difficult Loading structure conditions. Because for this algorithm we considered the best out of 20 random starts as the proxy, the proportion of .94 in fact indicates that for many data sets, the 20 runs of the algorithm yielded only 1 solution with the minimal loss value. This implies that the minimal loss values are highly instable across runs.

3.1.3.2. Recovery of the loadings. To evaluate how well the Blockwise Simplimax and Sparse group algorithms recover the loading matrices, we computed for all estimated loading matrices the loading recovery statistic (LRS). The LRS is defined as the mean of the congruence coefficients [25] between the columns of the true loading matrix $\mathbf{A}^{\text{true}}$ (i.e., the underlying loading matrix with 12 zero block vectors in the Easy condition, 10 in the Moderate condition, and 8 in the Difficult and Very Difficult conditions) and the estimated rotated loading matrix $\hat{\mathbf{A}}$, after taking into account the (arbitrary) permutations and reflections in the rotated components. To this end, we computed the LRS for all possible permutations and reflections of $\hat{\mathbf{A}}$, and retained the solution with the maximal value of LRS. The LRS coefficient expresses to what extent the true and rotated loading matrices are proportional, with a value of +1 indicating perfect proportionality and a value of 0 no proportional relationship. Note that in case of a zero column in $\hat{\mathbf{A}}$ the congruence coefficient is not defined. To repair this, we set the associated congruence coefficient to zero. This happened in 2 (out of the 3,200) analyses, with associated LRS values of .67 and .65.

In Table 3, the mean LRS and its standard deviation across all replicates is presented for each level of the four manipulated factors. From this table, it can be seen that, for both methods, the recovery of the loadings is affected most by the Loading structure. The other manipulated factors showed relatively small effects on the recovery for both methods, and when an effect occurred it was in the expected direction, i.e., worse performance for Different Block size, higher Error level and smaller Sample size. Therefore, we focus on differences between the Loading structure conditions.

Recall that the Easy, Moderate, and Difficult conditions involve a structure that complies perfectly with the block structure, and the Very Difficult condition involves clear violations of the block structure. For both algorithms, the recovery is almost perfect when the Loading structure is Easy, and decreases with increasing difficulty. It has been verified that in the Moderate, Difficult and Very Difficult conditions Blockwise Simplimax performs considerably better than Sparse group SCA in each cell of the design (mean differences ranging from .05 to .22, with standard errors ranging from .006 to .011). In individual replications of these conditions, the LRS for Blockwise Simplimax was at least .05 higher than that of Sparse group SCA in 1,719 (out of 2,400) cases, while the reverse was true in 67 cases. In the Very Difficult condition, the violation of the block structure was of course quite severe, because we had even modified the matrix $\mathbf{A}^{\text{true}}$ before constructing the data, and still tested to what extent the *unmodified* $\mathbf{A}^{\text{true}}$ was recovered. We expected that this would be advantageous to Sparse group SCA, since it aims at actually forcing whole blocks of loadings to be zero, and hence it could easier recover the full set of zero loadings in cases where some of the 'meant to be zero loadings' actually were nonzero during the data construction, but apparently this was not the case.

The largest differences in recovery between Blockwise Simplimax and Sparse group SCA appear to be among the various conditions of Loading structure and Error level. To visualize these effects, box plots of the difference in LRS scores between the two methods (i.e., $\text{LRS}_{\text{Blockwise Simplimax}}$–$\text{LRS}_{\text{Sparse group SCA}}$) for the eight combinations of Loading structure and Error level are depicted in Fig. 1 (upper part). A positive value indicates that Blockwise Simplimax outperforms Sparse group SCA. Thus, it can be concluded that Blockwise Simplimax outperforms Sparse group SCA considerably in the Moderate, Difficult and Very Difficult loading conditions with Low error level. Further, there are instances for which Sparse group SCA outperforms Blockwise Simplimax, but they are relatively scarce.

3.1.3.3. Recovery of the weight matrix. To evaluate to what extent the weight matrix $\mathbf{W}^{\text{true}}$ was recovered, we computed the position recovery statistic (PRS) [26] as

$$\text{PRS} = 1 - \frac{\sum_{k,r=1}^{K,R} |W_{kr}^{\text{true}} - \hat{w}_{kr}|}{KR}, \tag{10}$$

[6] We tested for cases where the global optimum is known, to what extent Blockwise Simplimax with 100 random starts would recover the original population loading matrix. That is, we rotated the 6 population loading matrices with zero-residuals (i.e., 2 (Equality of block size) × 3 (Loading structure)) 100 times with a random orthogonal rotation matrix, yielding 600 rotated matrices. To each matrix, the Blockwise Simplimax was applied. For 599 out of 600 rotated matrices, the global minimum was found. This is a reassuring starting point for further examining the quality of the algorithm.

Table 3
Mean LRS (and standard deviation) and Mean PRS (and standard deviation) per level of the manipulated factors, for Block-Simplimax and the Sparse group SCA

Factor	Level	LRS — Block-Simplimax	LRS — Sparse group SCA	PRS — Block-Simplimax	PRS — Sparse group SCA
Block size	Equal	.94(.08)	.84(.11)	.97(.08)	.83(.15)
	Different	.90(.10)	.83(.12)	.94(.10)	.82(.15)
Loading structure	Easy	.98(.03)	.99(.02)	1.00(.01)	1.00(.01)
	Moderate	.94(.08)	.83(.10)	.98(.06)	.83(.11)
	Difficult	.90(.10)	.77(.05)	.94(.10)	.72(.13)
	Very Difficult	.86(.10)	.73(.04)	.90(.12)	.74(.11)
Error level	.25	.96(.07)	.84(.11)	.97(.08)	.82(.15)
	.50	.88(.10)	.82(.12)	.93(.10)	.82(.15)
Sample size	100	.91(.10)	.83(.12)	.95(.10)	.82(.15)
	500	.93(.09)	.84(.12)	.96(.09)	.83(.15)

where the estimated $\hat{\mathbf{W}}$ was permuted in the same way as the corresponding $\hat{\mathbf{A}}$.[7] In Table 3, the mean PRS and its standard deviation is presented for each level of the five manipulated factors. As can be seen by comparing the effects of the manipulated factors on the PRS and the LRS (see Table 3) the effects are strikingly similar. That is, the Loading structure has the largest effect on the recovery of the position. In the Easy loading condition, recovery for both methods is almost always perfect, with only 9 and 6 (out of 800) exceptions for Blockwise Simplimax and Sparse group SCA, respectively. Blockwise Simplimax performs, on average, reasonably to very well for Moderate, Difficult and Very Difficult structures, as in each cell of the design the mean PRS > .83. Sparse group SCA performs rather poor in the Difficult and Very Difficult conditions, in that in each cell of the design the mean PRS never exceeded .78. For individual replications of the Moderate, Difficult and Very Difficult conditions, the PRS for Blockwise Simplimax was better than of Sparse group SCA in 1795 (out of 2,400) cases, while the reverse was true in 55 cases. This pattern can also be seen in Fig. 1 (lower part), which provides box plots of the difference in PRS scores between the two methods (i.e., $PRS_{\text{Blockwise Simplimax}}$–$PRS_{\text{Sparse group SCA}}$) for the eight combinations of Loading structure and Error level.

3.2. Simulation study 2

To investigate the performance of the CHull procedure in selecting the degree of simplicity (i.e., the number p) in Blockwise Simplimax, we applied the CHull procedure (as described in Section 2.2.3) to the 3,200 simulated data sets from Simulation study 1. The CHull involves a series of Blockwise Simplimax estimations for different values of p. For each estimation, we used 100 random starts and 1 rational Varimax-based start. For each data set, we recorded whether CHull indicated the value of p^{true} as the first indicated solution (i.e., as 'the optimal model').

In Table 4, the percentage of data sets for which CHull indicated the value of p^{true} as the first indicated solution, across all replicates is presented for each level of the four manipulated factors in column 3. Because the Very Difficult Loading structure condition involves clear violations of the block structure, we also present these percentages averaged across the Loading structure conditions that comply perfectly with the block structure (i.e., Easy, Moderate and Difficult) in column 4 of Table 4.

From Table 4, it can be concluded that, the performance of CHull is substantially affected by the Loading structure, and to a somewhat lesser extent by Block size, Error level and Sample size. The CHull performance in the Easy loading condition is almost perfect, with the correct indication of the number of zero vectors in 99% of the cases. The performance in the Moderate condition is still very good (84%). In the Difficult condition, the performance was well in the low Noise conditions (91%), but poor in the high Noise conditions (38%).

In practice, one will never know the true value of p. Therefore, it is interesting to see what happens to the recovery of the loadings if one would take p equal to the value suggested by the CHull procedure. We inspected this by computing the LRS for all data sets with the thus obtained p. As can be seen in Table 4 by comparing column 5 (LRS with true p) and column 6 (p indicated by CHull), it was found that the results indeed deteriorated slightly (i.e., average LRSs decreased at most .01), but were still, on average, quite good (i.e., never below .85) in all conditions. So even when the value used for p is based only on the CHull procedure, the Blockwise Simplimax method on average still yields a good to very good recovery of the loadings.

3.3. Discussion of results of the simulation studies

From the simulation results presented above it appears that the Blockwise Simplimax algorithm performs well in optimizing the loss function. The use of multiple starts is important to reduce the local minima problem, where 100 appeared to be sufficient in all conditions of our simulation study. We also saw that the use of the rational start is very useful in the conditions where a clear simple structure is present, but not in the Difficult and Very Difficult conditions, where a 'general factor' distorts the idea of simple structure. Unless it is known that a clear simple structure should be feasible, it is advised to keep using many random starts in addition to the rational start. The Sparse group SCA algorithm appears to be very sensitive to local minima. Therefore, we advise to use many more starts than the default 20.

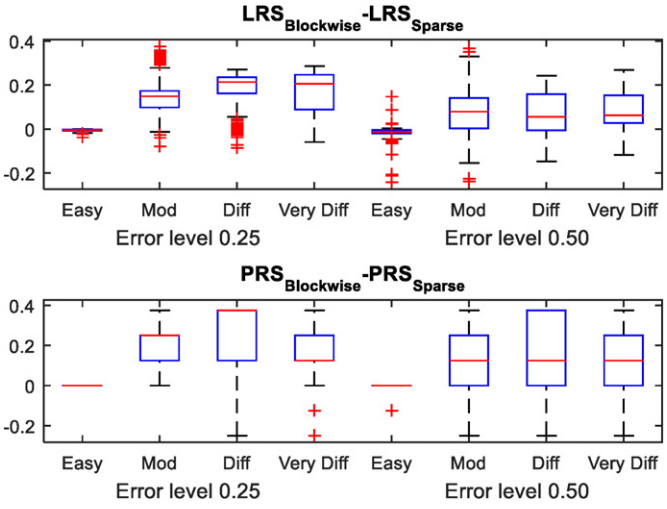

Fig. 1. Box plots of the difference between Blockwise Simplimax (Blockwise) and Sparse group SCA (Sparse) in LRS values (upper part) and in PRS values (lower part) as a function of Loading structure and Error level; a positive value indicates that Blockwise Simplimax outperforms Sparse group SCA. Mod = Moderate; Diff = Difficult.

[7] Theoretically, the PRS has a minimum of 0, but in the easy loading condition its minimum is 0.50, and in the other conditions, due to the permutation employed, values clearly higher than 0 would be expected too. In our simulation, the minimum encountered over all conditions and methods was 0.50.

Table 4
Percentages of data sets for which CHull indicated the correct value of p as the first indicated model (columns 3 and 4), and mean LRS for Block Simplimax with true p (column 5) and optimal p as indicated by CHull (column 6), per level of the manipulated factors.

Factor	Level	Across all levels of the design	Across loading structures Easy, Moderate and Difficult only	LRS — Block-Simplimax, with true p	LRS — Block-Simplimax, with optimal p indicated by CHull
Block size	Equal	78	86	.94	.93
	Different	69	79	.90	.90
Loading structure	Easy	99	99	.98	.98
	Moderate	84	84	.94	.94
	Difficult	65	65	.90	.89
	Very Difficult	47	–	.86	.85
Error level	.25	88	96	.96	.96
	.50	59	69	.88	.88
Sample size	100	69	79	.91	.90
	500	78	86	.93	.93

Further, the Blockwise Simplimax algorithm appears to recover the loading matrix and the weight matrix $\mathbf{W}^{true}$ reasonably well to very good, in conditions complying with a perfect block structure. In those conditions, the performance is affected little by issues as sample size (considering minimally N = 100) and error level. The recovery in conditions with Equal block sizes is better than with Unequal block sizes, especially in more difficult conditions, though the differences are relatively small in our simulation study. The differences in performance may be due to the fact that in our study the conditions with unequal blocks actually involved rather small blocks (i.e., involving 3 variables, rather than 6), which contributes to the instability of solutions, and hence may affect the performance. As could be expected, the performance drops in cases of a clear violation of the block structure. The Blockwise Simplimax algorithm clearly outperformed Sparse group SCA in the Moderate, Difficult, and Very Difficult conditions, while in the Easy condition the two methods performed about equally well (and both excellent). Therefore, we favor Blockwise Simplimax over Sparse group SCA to identify simplicity in block structures.

As to the performance of the CHull model selection procedure, we conclude that CHull can be a useful aid to indicate the number of zero vectors p, but that there is room for improvement as well. That is, CHull performs well in the Easy and Moderate conditions, but less so in the Difficult and especially the Very Difficult condition. We presume that a key factor for simplicity is that the proportion of explained variance of each true component is substantial. If the structure becomes more complicated, the CHull performance may get affected detrimentally by increasing noise levels, decreasing sample size and unequality of block sizes (though this effect might also result from the small block sizes involved in our

Table 5
Loadings after Blockwise Simplimax rotation of the SCA solution with 2 components and $p = 32$ zero-vectors. To ease the presentation, the loadings per attribute of the 8 panelists are positioned next to each other. Loadings that are rotated to a small value (i.e., associated with a value of 0 in W) are printed in a gray cell; loadings larger than .25 in absolute value are printed in bold face.

	Component 1								Component 2							
Panelist	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
N - Cream	.02	-.07	.05	.14	-.02	.04	-.00	-.08	.08	**.24**	.05	**.21**	-.03	-.03	-.06	.08
N - Acidic	-.03	-.01	.03	.02	-.00	.08	.11	.01	-.03	**-.20**	-.01	-.02	-.02	.02	-.04	-.03
N - Butter	.03	-.02	.03	.09	.04	-.01	-.04	-.02	.15	.16	.03	**.22**	.03	.07	-.01	-.01
N - Old milk	-.03	.02	-.04	.05	.01	-.04	-.02	.01	**-.28**	-.14	-.04	-.19	-.01	-.11	-.09	-.01
E - White	-.01	.10	.02	.12	.02	.03	.15	.13	**-.20**	**-.25**	-.15	**-.24**	-.17	-.15	**-.20**	-.10
E - Grey	.04	.04	.05	.03	.03	.03	-.04	.17	.03	-.12	-.09	-.06	-.15	.01	-.14	**-.32**
E - Yellow	-.03	-.11	.01	-.05	.01	-.06	-.05	-.02	**.27**	**.33**	.16	**.21**	.18	.19	**.23**	.19
E - Green	.07	-.08	.03	-.07	.10	.00	.04	.04	.09	.01	-.04	**.26**	-.01	-.06	-.07	.00
H - Resistance	**.29**	**.26**	**.27**	**.30**	**.25**	**.31**	**.26**	**.34**	-.08	-.01	-.09	.07	.01	.05	-.03	.17
E - Grainy	.20	.06	.14	**-.25**	.01	.18	.02	.00	-.04	-.09	.03	-.05	-.15	-.17	-.01	-.08
E - Shiny	**-.36**	**-.24**	**-.22**	**-.37**	**-.31**	**-.28**	-.18	-.17	.01	.04	.00	-.10	.00	-.02	.00	-.03
M - Firm	.17	**.23**	**.29**	**.32**	**.29**	**.30**	**.39**	**.35**	-.05	-.07	-.12	-.10	-.05	-.07	-.11	-.00
M - Melt down	**-.20**	-.09	**-.29**	**-.33**	**-.26**	**-.29**	**-.23**	-.18	.03	.15	.12	.14	.09	.08	.06	.10
M - Resistance	.11	.07	**.21**	**.36**	**.22**	**.29**	**.36**	**.28**	-.05	-.14	-.12	-.15	-.11	-.14	-.03	-.04
M - Creaminess	-.07	.11	.04	.29	.08	.05	-.03	.02	**.40**	.01	.05	-.15	.09	-.05	**.30**	.12
M - Grainy	**.14**	-.09	-.02	-.07	-.04	-.09	-.03	-.03	-.05	.22	.03	.05	.02	.06	.01	.01
M - Chalky	.06	-.01	.10	.02	.03	.11	.02	.07	**-.33**	**-.21**	-.18	**-.37**	-.21	-.23	**-.28**	-.14
M - Cream	.00	.03	.05	.01	.05	-.01	.06	-.03	.19	**.22**	.17	**.36**	.05	-.01	**.21**	.10
M - Fat	-.05	-.03	.04	.11	-.01	**.20**	.03	.04	**.42**	**.30**	.11	**.27**	.09	-.07	**.35**	.09
M - Butter	.00	-.01	-.00	.11	-.04	-.05	-.02	.00	**.28**	**.25**	**.27**	**.30**	.13	**.34**	**.37**	.08
M - Salt	-.14	-.05	-.04	-.16	-.09	**-.22**	-.16	.04	**.25**	.20	.04	.14	.01	**.44**	.22	.05
M - Sour	-.01	-.00	-.05	-.13	-.05	-.08	-.06	.05	-.02	**-.35**	**-.26**	-.14	-.15	-.10	-.18	-.16
M - Sweet	-.02	-.01	-.02	.04	.00	-.07	**.21**	-.01	.16	.14	-.02	.13	.10	-.12	**.32**	.05

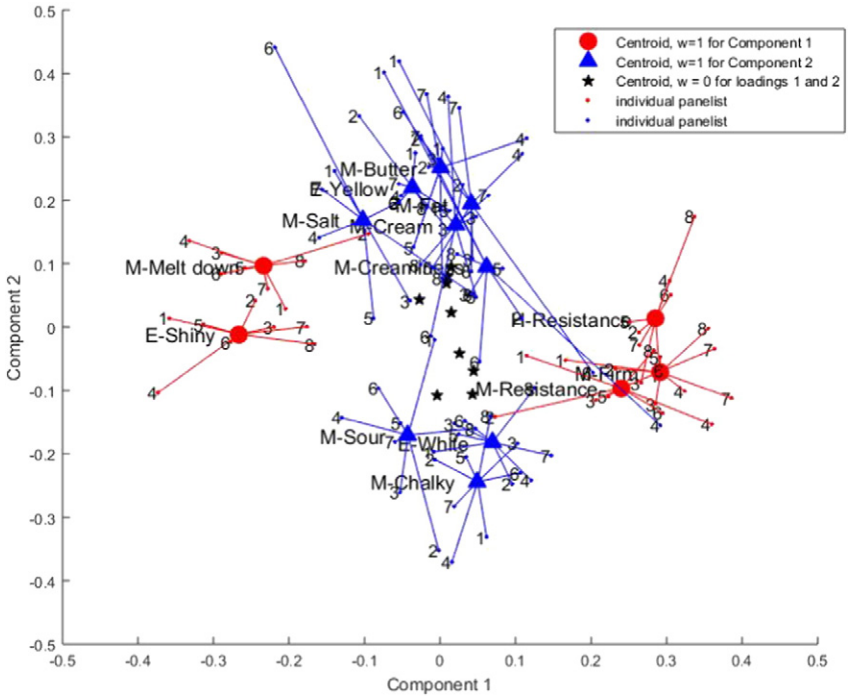

Fig. 2. Loadings of the 23 attribute centroids of the 8 panelists on two SCA components, after Blockwise Simplimax rotation with $p = 32$ zero-vectors, and the individual panelists loadings for those attributes associated with $w = 1$, indicated by their identification numbers.

Unequal block size condition). Furthermore, CHull cannot deal with the clear violations of the block structure in the Very Difficult condition, but here one could argue that the underlying structure is one of *no* zero blocks at all, hence $p = 0$ could also be seen as a proper outcome. Therefore, one might argue that this condition is too difficult to expect good results from the CHull method. Nevertheless, when using CHull to select p and next applying Blockwise Simplimax with this value of p, it turns out that the recovery of the loadings hardly changes, and is still good to very good on average for all levels. In other words, even if we would have a method that would be better able to recover the true number of zero blocks, this would hardly affect the recovery of the loadings. So, one may conclude that Blockwise Simplimax with the CHull selected value of p turns out to lead to good average recovery of the loadings.

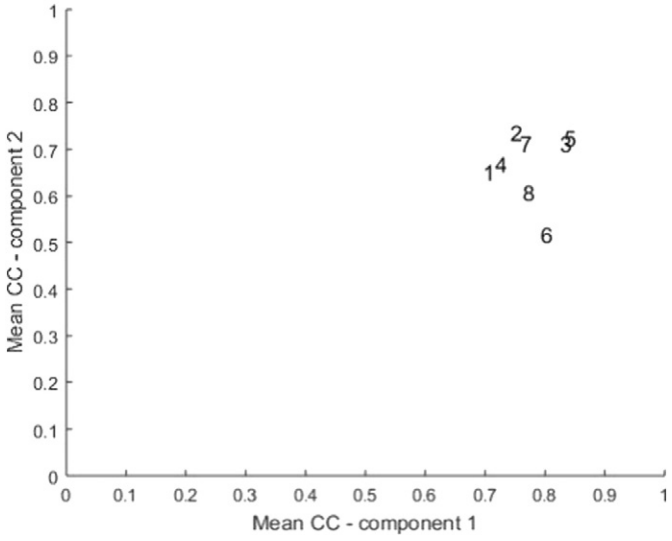

Fig. 3. For components 1 and 2, per panelist, the mean congruence coefficients (CC) between the loadings of the panelist concerned with all (7) other panelists.

4. Illustrative application of Blockwise Simplimax

To illustrate Blockwise Simplimax, we will re-analyze data from a sensory profiling study on cream cheese [27] (downloaded from http://www.models.life.ku.dk/datasets). These data pertain to 30 cheese samples rated by 8 trained panelists on 23 attributes. The 30 samples resulted from 3 samples of 8 different cheese types, and 6 samples of one cheese type. The attributes were scored using a 15 cm unstructured line scale. For details on the study see Bro et al. [5] and Frøst [27].

The data are three-way three mode data, and have been analyzed with three-way models previously [5,28]. Here, we apply a multi-set model. Treating the three-way data as multi-set data may reveal interesting structural information that is kept hidden with a three-way modeling approach [29]. Similar to the three-way modeling approaches applied so far, we focus on identifying the underlying dimensions of experience and the positions of the food samples on the dimensions of experience. However, we leave more freedom for revealing individual differences between panelists than is done in a three-way analysis. We do so by considering the three-way data as multi-set data, where each attribute refers to a data block $\mathbf{X}_k$ ($k = 1, \ldots, K = 23$) containing the ratings of the $I = 30$ samples by $J_k = 8$ panelists, with $J = \sum_{k=1}^{K} J_k = 23 \times 8 = 184$ ratings across all blocks. In this way, each block refers to one attribute rated by the eight panelists. Blockwise simplicity is a useful target in this context, because it aims at identifying simple structure in the loadings per attribute, with as little difference between the panelists as possible in the low loadings. Any structural deviations for specific panelists from the equality of loadings are of interest to identify deviating rating behavior of the (trained) panelists involved.

In line with the three-way modeling performed earlier, we first centered the data across the samples (i.e., the mean across the samples is zero for each j_k) and applied blockwise scaling, i.e., scaled the data per attribute k to a sum of squares of one.[8] Next, we applied SCA. We selected

[8] Note that since the block sizes are equal, the weights that result from the normalizing division are identical across blocks.

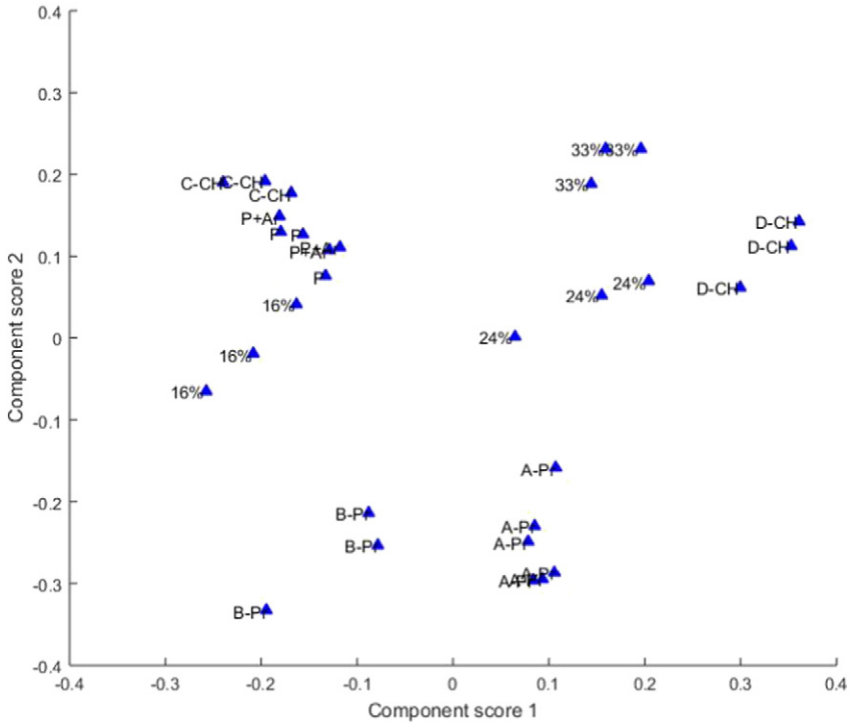

Fig. 4. Component scores of the cheese samples on the two SCA components, after Blockwise Simplimax rotation with $p = 32$.

the number of components R using CHull [22,23], considering the models with 1 up to 8 components (as previous models included 6 (constrained) components at maximum [28]), taking the explained variance as the goodness of fit and the number of components as the complexity measure. CHull indicated 2 as the optimal number of components, which is in line with the PARAFAC model of this data [5] as well as other multiset analyses [30,31]. This SCA solution accounted for 38.0% of the variance. Next we applied CHull for Blockwise Simplimax rotations, following the procedure discussed in Section 2.2.3, to select the number p. The CHull's first suggestion was $p = 32$, and as this solution appeared well interpretable to us, we selected this one to illustrate the method.

The Blockwise Simplimax rotated loadings on the two components are presented in Table 5, where loadings rotated to small values (i.e., associated with $w_{kr} = 0$, thus the $p = 32$ zero vectors) are indicated by gray cells — for ease of presentation, each row pertains to an attribute, while the loadings of the 8 panelist are positioned next to each other (rather than as a 184×2 matrix). In Fig. 2, we visualized the freely rotated loadings of the individual panelists (i.e., associated with $w_{kr} = 1$) and the centroids of the loadings across the 8 panelists of all (23) attributes (computed per component and attribute as the mean of the loadings across panelists). From Fig. 2 and Table 5, we conclude that Component 1 relates to texture properties, with firm texture resisting break down in the mouth (H-Resistance, M-Firm and M-Resistance) versus soft and shiny (E-Shiny and M-Melt). Component 2 contrast the attributes M-Butter, E-Yellow, M-Fat, M-Salt, M-Creaminess and M-Cream with E-White, M-Sour and M-Chalky. Note that the centroids of the Blockwise Simplimax loadings are rather similar to the PARAFAC loadings (see Fig. 5 in Bro et al. [5]).

As can also be seen in Table 5 (rowwise, per component) and Fig. 2 (lines from the individual panelists to centroids), the individual panelists show some dispersion, which is larger for some attributes (e.g., M-Creaminess) than for others (e.g., E-Yellow). Further, some panelists appear more deviating than others. Per panelist and component, we quantify the degree of correspondence between panelists as the mean of the congruence coefficients (CC) between the loadings of the panelist concerned with all (i.e., 7) other panelists. The mean CC of all panelists for the two components are visualized in Fig. 3. As can be seen, the panelists show a higher level of agreement for Component 1 than for Component 2. Further, Panelists 3 and 5 show the highest levels of agreement, and Panelist 6 the lowest level. The level of agreements can be used as an indication of panelist quality. In our view, this provides a better expression of panelist quality than the relative size of the panelist loadings of the PARAFAC solution, as proposed by Bro et al. [5]. The latter measure is influenced by the level of agreement between panelists as well as the variability in ratings across samples and attributes. The latter aspect is not necessarily related to panelist quality, thereby making panelist PARAFAC loadings less suitable to express panelist quality.

In Fig. 4, the rotated component scores of the 30 cheese samples are presented. It can be seen that the replicates are close for some of the cheese types (as C-CH) and differ more for others (as B-PR).

We have also carried out a Sparse group SCA on these data, using the same penalty tuning values as Van Deun et al. [14], and following their line of reasoning in the model selection. Specifically, we used as f values 0, .0001, .001, .01, .1, .2, .5 and 1. In all cases we took $R = 2$. We inspected the numbers of zero blocks thus attained, and the residual sums of squares (i.e., $||\mathbf{X}\text{-}\mathbf{F}\mathbf{A}^T||^2$), as presented in Table 6. Considering that the residual sums of squares of the solutions up to $f_G = .1$ are hardly higher than that of the SCA solution in the first line, and that these solutions

Table 6
Summary of outcomes from Sparse group SCA analyses with different values for penalty parameter.

f_G	λ_G	P	$\|\|X\text{-}FA^T\|\|^2$
0	0.00	0	14.26
.0001	0.00	0	14.26
.001	0.00	0	14.26
.01	0.01	0	14.27
.1	0.09	5	15.03
.2	0.18	19	16.70
.5	0.46	37	21.60
1	0.92	46	23.00

Table 7
Loadings from Sparse group SCA solution with 2 components and $f_G = .2$, leading to 19 zero vectors. To ease the presentation, the loadings per attribute of the 8 panelists are positioned next to each other. The 19 zero blocks have been indicated by 0 and corresponding cells are colored gray; loadings larger than .15 in absolute value are printed in bold face.

	Component 1								Component 2							
Panelist	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
N - Cream	0	0	0	0	0	0	0	0	.02	.06	.01	.05	-.01	-.01	-.01	.02
N - Acidic	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N - Butter	0	0	0	0	0	0	0	0	.02	.02	.00	.03	.01	.01	-.00	-.00
N - Old milk	0	0	0	0	0	0	0	0	-.08	-.04	-.01	-.05	-.00	-.03	-.03	-.00
E - White	0	0	0	0	0	0	0	0	-.10	-.13	-.08	-.12	-.09	-.08	-.10	-.05
E - Grey	0	0	0	0	0	0	0	0	.01	-.04	-.04	-.02	-.05	.00	-.06	-.12
E - Yellow	0	0	0	0	0	0	0	0	**.17**	**.21**	.09	.13	.11	.12	.14	.12
E - Green	0	0	0	0	0	0	0	0	.01	.00	-.00	.02	-.00	-.01	-.01	-.00
H - Resistance	**.20**	**.18**	**.19**	**.21**	**.18**	**.21**	**.19**	**.23**	-.00	-.00	-.00	.00	.00	.00	-.00	.00
E - Grainy	.07	.02	.05	-.08	.01	.07	.01	.01	-.00	-.00	.00	-.00	-.00	-.00	-.00	-.00
E - Shiny	**-.25**	**-.17**	**-.16**	**-.25**	**-.21**	**-.19**	-.13	-.12	0	0	0	0	0	0	0	0
M - Firm	.12	**.16**	**.20**	**.23**	**.21**	**.20**	**.27**	**.25**	0	0	0	0	0	0	0	0
M - Melt down	-.13	-.06	**-.18**	**-.21**	**-.17**	**-.18**	-.14	-.12	.00	.01	.01	.01	.00	.00	.00	.01
M - Resistance	.07	.04	.13	**.23**	**.15**	**.19**	**.23**	**.18**	-.00	-.01	-.01	-.01	-.01	-.01	-.00	-.00
M - Creaminess	-.02	.02	.01	.06	.02	.01	-.01	.00	**.21**	.01	.03	-.07	.05	-.03	**.16**	.06
M - Grainy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M - Chalky	0	0	0	0	0	0	0	0	**-.21**	-.13	-.12	**-.23**	-.13	-.14	**-.18**	-.09
M - Cream	0	0	0	0	0	0	0	0	.10	.12	.09	.19	.03	-.01	.11	.05
M - Fat	0	0	0	0	0	0	0	0	**.26**	**.19**	.07	**.17**	.06	-.04	**.21**	.06
M - Butter	0	0	0	0	0	0	0	0	**.19**	**.17**	**.18**	**.19**	.08	**.22**	**.24**	.05
M - Salt	-.04	-.02	-.01	-.04	-.02	-.06	-.04	.01	.14	.12	.03	.09	.01	**.25**	.13	.03
M - Sour	0	0	0	0	0	0	0	0	-.01	-.19	-.14	-.08	-.08	-.06	-.09	-.09
M - Sweet	0	0	0	0	0	0	0	0	.06	.06	-.01	.06	.04	-.05	.12	.02

hardly increase the simplicity (with only 5 zero blocks), it seems we have to choose from the last three solutions. The solutions for $f_G = .5$ and $f_G = 1$ seem to lead to a relatively big increase in residual sums of squares, so we first consider the solution for $f_G = .2$, yielding 19 zero blocks.

Table 7 gives the loadings resulting from Sparse group SCA, with the 19 blocks of zeroes indicated by "0". We can see that these refer to a subset of the blocks-rotated-to-small-values from Blockwise Simplimax. Specifically, for component 1, the attributes E-Grainy, M-Creaminess and M-Salt are not modeled by zero blocks, while for component 2, the attributes N-Cream, N-Butter, N-Old Milk, E-Grey, E-Green, H-Resistance, E-Grainy, M-Melt down, M-Resistance, and M-Sweet are not modeled by zero blocks.

Comparing the loadings further, it can be seen that the Sparse group SCA loadings are generally smaller than those in Blockwise Simplimax. This is due to the penalty, which not only steers blocks of loadings toward zero, but also tends to reduce the size of all loadings. Upon comparing the blocks of large loadings, it turns out that in both solutions the components are associated with exactly the same attributes, while in Sparse group SCA the relations are less pronounced, due to the diminished size of the loadings. The components of both solutions have a similar interpretation, as expressed by the rather high congruence coefficients [25] between the sets loadings of .92 for both components 1 and 2.

We also considered the Sparse group SCA solution for $f_G = .5$, yielding 37 zero blocks. These resulted in a subset of non-zero blocks compared to the Blockwise Simplimax, namely on component 1 of H-Resistance, E-Shiny, M-Firm, M-Melt down, M-Resistance, and on component 2 of M-Chalky, M-Fat, M-Butter. The interpretation of both solutions is somewhat different, as expressed by the moderate to low congruence coefficients between the sets loadings of .82 for component 1, and 0.56 for component 2. Generally, attributes that have relatively high Blockwise Simplimax loadings for all panelists are not set to zero, whereas attributes that have only for some panelists somewhat higher loadings are sacrificed (e.g., M-Salt of component 2). This Sparse group SCA solution thus offers less insight into the presence of deviating panelists.

5. Concluding remarks

We proposed a new rotation criterion, called Blockwise Simplimax, for rotating the components resulting from an SCA analysis of object-wise linked multiset data toward blockwise simplicity. This rotation can be used to reveal in which blocks the specific components play an important role. Further, we proposed a model selection criterion, to identify the required degree of simplicity for a data set under study. Our simulation study showed that Blockwise Simplimax performs reasonably well in recovering the underlying simple structure. Further, Blockwise Simplimax outperformed its sparse competitor Sparse group SCA [14], in the sense that in the Easy condition it performed similarly, while in the other conditions it performed clearly better. The latter is rather remarkable, because both methods essentially have the same goal. Based on our simulation results we would favor Blockwise Simplimax over Sparse group SCA.

Our CHull based procedure for selecting an optimal required level of simplicity for Blockwise Simplimax performed well in simple conditions, but improvement of the procedure would be welcome for some more difficult ones. As a possible remedy for the worse behavior in the unequal block size condition, we considered an alternative to the applied complexity measure, namely the total number of zero elements (rather than the number of zero vectors p). However, when using this alternative measure, the performance deteriorated rather than improved. Finding a better model selection method therefore remains a challenge for future research. For now, in empirical practice, applying CHull may offer a rough indication of interesting solutions to consider. The final decision among different levels of simplicity should be based on the interpretability of the associated rotated solutions. It is reassuring that we have seen that SCA followed by Blockwise Simplimax with the number of zero target blocks chosen according to the CHull procedure, on average recovered the loadings well to very well, indicating that a mistaken choice of the number of zero target blocks may not have too bad consequences.

Blockwise Simplimax is related to the rotation criterion considered in DISCO-SCA [15,32]. Both methods minimize a loss function that rotates the loadings toward a target matrix that imposes zeroes according

to the block structure in the data. The key difference is that DISCO-SCA explicitly aims at identifying components that are either common or distinctive, while the goal of Blockwise Simplimax is to achieve a simple block structure. This difference in goals is clear when considering the model selection procedures that have been proposed for the two methods: while the procedure for Blockwise Simplimax indicates how many zero vectors can be easily imposed, the most important model selection step in the DISCO procedure is determining how many of the simultaneous components are common and how many distinctive. Finally, as DISCO-SCA has only been applied to the two block case so far, it is unclear to what extent the method is as easy to use as Blockwise Simplimax with many blocks.

We introduced Blockwise Simplimax in the context of multiset data coupled in the row mode, implying that the objects are the same across all blocks. However, Blockwise Simplimax can be applied as well to multiset data coupled through the variables. Then, block simplicity would be required in the component scores, rather than in the loadings. For such data, it is instructive to compare Blockwise Simplimax to clusterwise SCA [12,29,33]. Clusterwise SCA simultaneously partitions the data blocks into a few clusters and fits a separate SCA model per cluster. Both methods are similar in that both aim at having zero vectors in the component scores. The key differences are that (1) clusterwise SCA yields exact zeroes, because it is a hard clustering approach, while Blockwise Simplimax just aims at zeroes, without strictly imposing them, as it is a rotation method; (2) in clusterwise SCA each block is allocated to a single cluster only and is only related to the other blocks that are assigned to the very same cluster, whereas in Blockwise Simplimax each block may be associated with all other blocks via different components; the latter can be viewed as an overlapping clustering. Based on the nature of the methods, we conjecture that clusterwise SCA would be better at identifying a limited number of subsets (e.g., 2 or 3, depending on the nature of the problem) of similarly behaving blocks, while the subsets behave rather differently; further, we conjecture that Blockwise Simplimax would be better at identifying the components associated with each block individually, especially if those components explain relatively much of the variance, irrespective of whether the individual blocks would behave similarly or not. However, it would require more detailed comparisons, based on theory, simulation and applications, to provide a well-founded picture.

Conflict of interest

Timmerman, Ceulemans and Kiers declare that they have no conflict of interest.

Compliance with ethical requirements

Timmerman, Ceulemans and Kiers declare that they comply with Springer's ethical policies.

References

[1] I. Van Mechelen, E. Ceulemans, Component- and factor-based models for data fusion in the behavioral sciences, Proc. IEEE 103 (9) (2015) 1621–1634.
[2] D. Lahat, T. Adali, C. Jutten, Multimodal data fusion: an overview of methods, challenges, and prospects, Proc. IEEE 103 (9) (2015) 1449–1477.
[3] R. Bro, H.J. Nielsen, F. Savorani, K. Kjeldahl, I.J. Christensen, N. Bruenner, et al., Data fusion in metabolomic cancer diagnostics, Metabolomics 9 (2013) 3–8.
[4] I. Måge, E. Menichelli, T. Næs, Preference mapping by PO-PLS: separating common and unique information in several data blocks, Food Qual. Prefer. 4;24 (1) (2012) 8–16.
[5] R. Bro, E.M. Qannari, H.A. Kiers, T. Næs, M.B. Frøst, Multi-way models for sensory profiling data, J. Chemother. 22 (1) (2008) 36–45.
[6] K. Van Deun, A. Smilde, dW. van, H. Kiers, I. Van Mechelen, A structured overview of simultaneous component based data integration, BMC Bioinformatics 10 (2009) 246.
[7] K. Van Deun, A.K. Smilde, L. Thorrez, H.A.L. Kiers, I. Van Mechelen, Identifying common and distinctive processes underlying multiset data, Chemom. Intell. Lab. Syst. 11/15;129 (2013) 40–51.
[8] H.A.L. Kiers, J.M.F. ten Berge, Hierarchical relations between methods for simultaneous component analysis and a technique for rotation to a simple simultaneous structure, Br. J. Math. Stat. Psychol. 47 (1) (1994) 109–126.
[9] M.E. Timmerman, H.A.L. Kiers, Four simultaneous component models for the analysis of multivariate time series from more than one subject to model intraindividual and interindividual differences, Psychometrika 03;68 (1) (2003) 105–121.
[10] O.E. de Noord, E.H. Theobald, Multilevel component analysis and multilevel PLS of chemical process data, J. Chemother. 19 (5–7) (2005) 301–307.
[11] A. Brose, K. De Roover, E. Ceulemans, P. Kuppens, Older adults' affective experiences across 100 days are less variable and less complex than younger adults', Psychol. Aging 30 (1) (2015) 194.
[12] K. De Roover, E. Ceulemans, M.E. Timmerman, K. Vansteelandt, J. Stouten, P. Onghena, Clusterwise simultaneous component analysis for analyzing structural differences in multivariate multiblock data, Psychol. Methods 03;17 (1) (2012) 100–119.
[13] M.E. Timmerman, H.C. Hoefsloot, A.K. Smilde, E. Ceulemans, Scaling in ANOVA-simultaneous component analysis, Metabolomics 1–12 (2015).
[14] K. Van Deun, T. Wilderjans, dB. van, A. Antoniadis, I. Van Mechelen, A flexible framework for sparse simultaneous component based data integration, BMC Bioinformatics 12 (1) (2011) 448.
[15] M. Schouteden, K. Van Deun, S. Pattyn, I. Van Mechelen, SCA With Rotation to Distinguish Common and Distinctive Information in Linked Data, 45 (3) (2013) 822–833.
[16] M.W. Browne, An overview of analytic rotation in exploratory factor analysis, Multivar. Behav. Res. 36 (1) (2001) 111–150.
[17] L.L. Thurstone, Multiple Factor Analysis, University of Chicago Press, Chicago, 1947.
[18] H.A.L. Kiers, SIMPLIMAX: oblique rotation to an optimal target with simple structure, Psychometrika 59 (4) (1994) 567–579.
[19] I.T. Jolliffe, Principal Component Analysis, second ed. Springer-Verlag, New York, 2002.
[20] H.F. Kaiser, The varimax criterion for analytic rotation in factor analysis, Psychometrika 23 (3) (1958) 187–200.
[21] M.W. Browne, Orthogonal rotation to a partially specified target, Br. J. Math. Stat. Psychol. 25 (1) (1972) 115–120.
[22] E. Ceulemans, H.A.L. Kiers, Selecting among three-mode principal component models of different types and complexities: a numerical convex hull based method, Br. J. Math. Stat. Psychol. 05;59 (1) (2006) 133–150.
[23] T.F. Wilderjans, E. Ceulemans, K. Meers, CHull: a generic convex-hull-based model selection method, Behav. Res. Methods 45 (1) (2013) 1–15.
[24] R.L. Burden, J.D. Faires, The bisection method, Numerical Analysis, nineth ed.Brooks/Cole, Australia 2010, pp. 48–56.
[25] L.R. Tucker, A Method for Synthesis of Factor Analysis Studies, Department of the Army, Washington, DC, 1951 (Report No.: 984).
[26] D. Depril, I. Van Mechelen, B. Mirkin, Algorithms for additive clustering of rectangular data tables, Comput. Stat. Data Anal. 52 (11) (2008) 4923–4938.
[27] M.B. Frøst, The Influence of Fat Content on Sensory Properties and Consumer Perception of Dairy Products, 2002.
[28] T.F. Wilderjans, E. Ceulemans, Clusterwise Parafac to identify heterogeneity in three-way data, Chemom. Intell. Lab. Syst. 129 (2013) 87–97.
[29] K. De Roover, M.E. Timmerman, I. Van Mechelen, E. Ceulemans, On the added value of multiset methods for three-way data analysis, Chemom. Intell. Lab. Syst. 129 (2013) 98–107.
[30] E. Ceulemans, M.E. Timmerman, H.A.L. Kiers, The CHull procedure for selecting among multilevel component solutions, Chemom. Intell. Lab. Syst. 106 (2011) 12–20.
[31] E. Ceulemans, T.F. Wilderjans, H.A. Kiers, M.E. Timmerman, MultiLevel simultaneous component analysis: a computational shortcut and software package, Behav. Res. Methods 1–13 (2015).
[32] M. Schouteden, K. Van Deun, T.F. Wilderjans, I. Van Mechelen, Performing DISCO-SCA to search for distinctive and common information in linked data, Behav. Res. Methods 46 (2) (2014) 576–587.
[33] K. De Roover, E. Ceulemans, M.E. Timmerman, P. Onghena, A clusterwise simultaneous component method for capturing within-cluster differences in component variances and correlations, Br. J. Math. Stat. Psychol. 66 (1) (2013) 81–102.