Behavior Research Methods (2020) 52:236–263
https://doi.org/10.3758/s13428-019-01222-4

Detecting which variables alter component interpretation across multiple groups: A resampling-based method

Sopiko Gvaladze[1] · Kim De Roover[1,2] · Francis Tuerlinckx[1] · Eva Ceulemans[1]

Published online: 1 April 2019

Abstract
In psychology, many studies measure the same variables in different groups. In the case of a large number of variables when a strong a priori idea about the underlying latent construct is lacking, researchers often start by reducing the variables to a few principal components in an exploratory way. Herewith, one often wants to evaluate whether the components represent the same construct in the different groups. To this end, it makes sense to remove outlying variables that have significantly different loadings on the extracted components across the groups, hampering equivalent interpretations of the components. Moreover, identifying such outlying variables is important when testing theories about which variables behave similarly or differently across groups. In this article, we first scrutinize the lower bound congruence method (LBCM; De Roover, Timmerman, & Ceulemans in *Behavior Research Methods, 49,* 216–229, 2017), which was recently proposed for solving the outlying-variable detection problem. LBCM investigates how Tucker's congruence between the loadings of the obtained cluster-loading matrices improves when specific variables are discarded. We show that LBCM has the tendency to output outlying variables that either are false positives or concern very small, and thus practically insignificant, loading differences. To address this issue, we present a new heuristic: the lower and resampled upper bound congruence method (LRUBCM). This method uses a resampling technique to obtain a sampling distribution for the congruence coefficient, under the hypothesis that no outlying variable is present. In a simulation study, we show that LRUBCM outperforms LBCM. Finally, we illustrate the use of the method by means of empirical data.

Keywords Multigroup data · Principal component analysis · Resampling · Permutation test · Tucker′s congruence · Measurment invariance

Introduction

In many behavioral studies, a relatively large number of variables (say ten or more) are measured in different groups of participants. The International College Study (ICS; e.g., Diener et al., 2001; Kuppens, Ceulemans, Timmerman, Diener, & Kim-Prieto, 2006) is a good example. In this study, 10,018 students from 48 different countries indicated, among other things, how much they endorse 11 different values (happiness, intelligence, material wealth, etc.) on a 9-point Likert scale (Diener et al., 2001). Given such data, researchers are often not directly interested in the measured variables themselves, but in underlying constructs or dimensions that summarize the covariance structure of the variables, and in how these constructs are similar or different across groups. The previous analysis of the ICS value data by De Roover, Timmerman, and Ceulemans (2017) suggested that the covariances of the values can be summarized by means of two dimensions. The first dimension, "Showing success & benevolence," pertains to *material wealth*, *physical attractiveness*, *physical comforts*, *excitement/arousal*, *competition*, *heaven/afterlife*, and *self-sacrifice*. The second dimension, "Fun, happiness, & achievement," refers to *intelligence/knowledge*, *happiness*, *success*, and *fun*. Yet, in line with the known distinctions between, for instance, collectivistic and individualistic countries, some of the values had different covariance structures across the countries. For instance, self-sacrifice covaried less with the other values in the "showing success and benevolence" construct in more collectivistic countries.

✉ Sopiko Gvaladze
sofia.gvaladze@kuleuven.be

[1] Faculty of Psychology and Educational Sciences, KU Leuven, Leuven, Belgium

[2] Department of Methodology and Statistics, Tilburg University, Tilburg, Netherlands

Following De Roover, Timmerman, and Ceulemans (2017), we will call the latter variable "outlying." Detecting such outlying variables is important, as pinpointing such differences across groups is the main focus of comparative research (Triandis, 1988). Relatedly, if one aims to further compare the scores on measured constructs across groups (e.g., by computing means and variances), measurement invariance should be ensured first (Asparouhov & Muthén, 2014; Chan, Ho, Leung, Chan, & Yung, 1999; Meredith & Teresi, 2006; Sočan, 2016), which will fail in the presence of outlying variables.

To extract the underlying constructs, dimension reduction methods such as factor analysis (Yong & Pearce, 2013) and component analysis (Abdi & Williams, 2010; Jolliffe, 2005) are popular. Herewith, a further distinction can be made between exploratory and confirmatory approaches (Hurley et al., 1997; Schmitt, 2011; Tukey, 1980), depending on whether or not one has a preliminary idea about the positions of the variables on the constructs. In this article, we will focus on the *exploratory* case, in which no such a priori knowledge is available. In this respect, we focus on component analysis rather than exploratory factor analysis (EFA), because it is more computationally efficient than EFA, is applicable when the assumption of underlying constructs causing the covariances does not hold (Borsboom, Mellenbergh, & Van Heerden, 2003), and often leads to results that are very similar to those of EFA (Ogasawara, 2000; Velicer, Peacock, & Jackson, 1982). When the number of groups is larger than one, useful component analysis approaches are simultaneous component analysis (Kiers & ten Berge, 1994; Timmerman & Kiers, 2003; Van Deun, Wilderjans, Van den Berg, Antoniadis, & Van Mechelen, 2011) and its cluster-wise variants (De Roover, Ceulemans, Timmerman, Nezlek, & Onghena, 2013; De Roover, Ceulemans, Timmerman, & Onghena, 2013; De Roover et al., 2012). As multigroup extensions of standard principal component analysis (Jolliffe, 2002), these methods reduce the observed variables to a few components. Both approaches represent the relations between the variables and the components in the so-called loading matrix; these loadings are used to interpret the obtained components. The difference between simultaneous component analysis (SCA) and its cluster-wise extensions is that SCA imposes the loadings to be the same across groups, whereas cluster-wise SCA clusters the groups in a few clusters and yields a separate loading matrix for each cluster.

The loading matrices of cluster-wise SCA are a good starting point for detecting outlying variables, as the clustering of the groups is driven by the strongest loading differences (De Roover, Timmerman, De Leersnyder, Mesquita, & Ceulemans, 2014), while small loading differences are averaged out within clusters. When no outlying variables are present, we expect all the variables to have similar loading patterns across the (clusters of) groups. It is worth noting that we deliberately use the term "similar loading patterns" rather than "same loading patterns," because we are only interested in larger loading differences that alter the interpretation of the underlying constructs. Visually detecting outlying variables in the cluster-specific loading matrices is difficult and subjective, however: When are the loading patterns of a variable different enough across clusters to consider the variable outlying? To avoid making subjective and nonconsistent decisions, objective and replicable procedures are needed.

Therefore, De Roover, Timmerman, and Ceulemans (2017) recently proposed a detection heuristic to screen the cluster-wise SCA loading matrices for outlying variables. This approach is called the *lower bound congruence method* (LBCM), because it assesses the similarity of loadings by computing the Tucker's congruence (Tucker, 1951) of the corresponding components in the cluster-specific loading matrices and uses a congruence value of .95 (Lorenzo-Seva & ten Berge, 2006) as a lower bound (i.e., a congruence lower than .95 implies that the data contain at least one outlying variable). We will show that LBCM is not without problems, however, in that it is prone to output false positives for empirical data: Aside from the targeted larger loading differences, LBCM also detects small differences that sometimes are not practically important because they do not hamper the underlying constructs being interpreted the same across the clusters and may be due to error fitting.

The aim of this article is to overcome this sensitivity to differences that are either false positives and/or practically insignificant. To this end, we present and evaluate a new approach to detecting outlying variables: the *lower and resampled upper bound congruence method* (LRUBCM). This method detects outlying variables by comparing the loading matrices of two clusters. Thus, in the case that data are modeled using more than two clusters, the LRUBCM procedure is applied in a pair-wise manner (see "Illustrative applications" section). LRUBCM extends LBCM by adding two new features. First, in addition to the lower bound, an upper threshold is imposed, as well, to discard solutions with too many outlying variables (false positives). This upper threshold is dynamically derived from the sampling distribution of the Tucker's congruence coefficient under the null hypothesis that the data contain no outlying variables. This allows the user to take data characteristics into account that influence the size of the Tucker's congruence value—for instance, the numbers of observations, variables, groups, and components, as well as the size and structure of the loadings (Abdi, 2010; Lorenzo-Seva & ten Berge, 2006)—when setting the upper bound of the detection heuristic. Second, while building this sampling distribution, LRUBCM does not require that the loadings be identical across the two clusters, but checks for similarity instead, to avoid detecting practically insignificant differences. This is achieved by slightly perturbing the loadings when generating the sampling distribution. The amount of perturbing can be adjusted by users, giving them flexibility to decide which loading differences are

practically important when interpreting dimensions in their field. We will show through simulations and analyses of real data that the LRUBCM approach outperforms LBCM.

The remainder of the article is organized in five sections. After recapitulating the data structure and preprocessing in section 2.1, sections 2.2 and 2.3 give overviews of SCA-P and its cluster-wise extension, respectively. Section 3 discusses outlying-variable detection heuristics. First we review the recently developed LBCM (section 3.1), and in section 3.2 we show its drawbacks. To overcome these drawbacks, we propose the new method in section 3.3. Section 4 presents an extensive simulation study, showing that LRUBCM outperforms LBCM. In section 5, we apply LRUBCM to emotional acculturation data. In section 6 we discuss our findings and directions for future work.

Methodology

In this section, we first discuss the data structure and preprocessing. Next, we recapitulate both SCA-P and its cluster-wise extension.

Data structure and preprocessing

In this article, we assume that the data are composed of I data blocks, in which each data block $\mathbf{X}_i$ ($i = 1 \ldots I$) corresponds to the data of group i and consists of the scores of N_i subjects on the same J variables. For example, in the ICS value data, each data block contains the scores of a sample of students from a specific country on the 11 variables under study. The number of subjects in each block N_i is allowed to vary from block to block. The full data set $\mathbf{X}$ is obtained by vertically concatenating the $\mathbf{X}_i$ blocks, and thus it counts $N = \sum_{i=1}^{I} N_i$ rows. In the ICS value data, one student corresponds to one row, and the overall data set consists of 10,018 rows. Following De Roover, Timmerman, and Ceulemans (2017), the data are preprocessed before the analysis. Since we are interested in between-block differences in covariance structure, the variables are centered in each block, discarding between-block differences in means that might otherwise distort the obtained loadings (see Ceulemans, Wilderjans, Kiers, & Timmerman, 2016). Moreover, to give the variables the same weight in the analysis, each variable is rescaled so that its variance across the groups equals 1 (Ceulemans et al., 2016).

SCA-P

Model SCA-P (Kiers & ten Berge, 1994; Timmerman & Kiers, 2003) stands for *simultaneous component analysis while imposing an equal pattern matrix*. It extracts one set of Q components from the I data blocks. More specifically, SCA-P models each data block $\mathbf{X}_i$ ($i = 1 \ldots I$) as follows:

$$\mathbf{X_i} = \mathbf{F_i B}' + \mathbf{E_i} \quad (1)$$

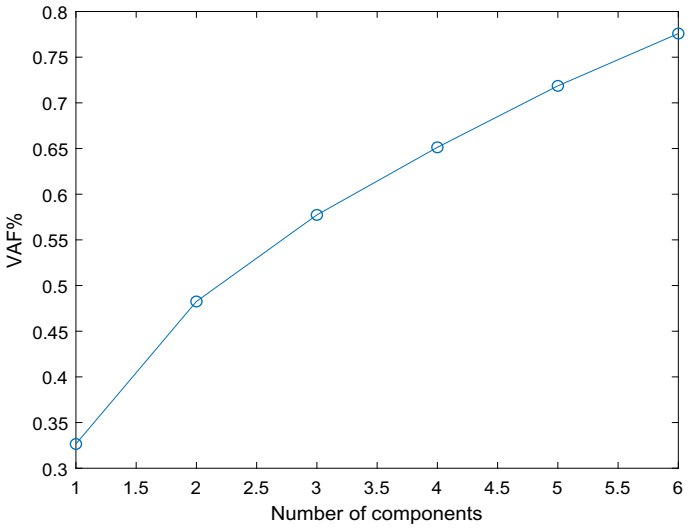
The matrix $\mathbf{F}_i(N_i \times Q)$ holds the scores of the N_i subjects in the ith data block on the Q components. To identify the model, the variance of the component scores across all the groups is fixed to one. Due to the centering of the data blocks, the means of the component scores equal zero for each data block. Given the preprocessing, the loadings in $\mathbf{B}(J \times Q)$ represent the correlations between the original variables and the components. The matrix $\mathbf{E}_i(N_i \times J)$ denotes the residual matrix.

Deciding on the number of components Q In exploratory SCA-P analyses, the optimal number of components that provide a good summary of the data without yielding an overly complex solution, is often not known beforehand. To assess this optimal number, a scree test can be conducted on a plot of the percentage of variance accounted for (VAF%) by solutions with increasing numbers of components. Next to visually investigating whether an elbow can be detected in the scree plot, the CHull procedure can be used (Ceulemans & Kiers, 2006; Wilderjans, Ceulemans, & Meers, 2013). This procedure first finds the upper boundary of the convex hull (CHull) of the considered solutions and retains the solutions located on this boundary. Next, from these CHull solutions, it picks the one after which the increase in VAF% levels off. The scree plot for the ICS value data (Fig. 1a) suggests that the optimal number of components amounts to two.

Rotation Once the decision is made on how many components to retain, one usually rotates the obtained components so that the loading matrix $\mathbf{B}$ has a simple structure, meaning that most variables load strongly on one component only. Following De Roover, Timmerman, and Ceulemans (2017), we will use the VARIMAX rotation in this article, which is an orthogonal rotation (Kaiser, 1958). Table 1 shows the VARIMAX-rotated SCA-P loading matrix for the ICS value data, revealing the two constructs mentioned in the introduction: "Showing success & benevolence" and "Fun, happiness, & achievement."

Cluster-wise SCA-P

Model For some data sets, restricting the loadings to be identical across the groups is an oversimplification. Therefore, De Roover, Ceulemans, Timmerman, and Onghena (2013) presented cluster-wise SCA-P, which clusters the data blocks in K clusters based on the between-block similarities and differences in loading structure and assigns each of the I data blocks to one of the K clusters. As a result, the groups with similar loading patterns will belong to the same cluster. Each cluster is



Cluster 1	Cluster 2
Bangladesh, Cameroon, Chile, Croatia, Egypt, Georgia, Ghana, India, Indonesia, Iran, Kuwait, Malaysia, Nigeria, Philippines, Poland, South Africa, Thailand, Turkey, Uganda, Zimbabwe	Australia, Austria, Belgium, Brazil, Bulgaria, Canada, China, Columbia, Cyprus, Germany, Greece, Hong Kong, Hungary, Italy, Japan, Mexico, Nepal, Netherlands, Portugal, Russia, Singapore, Slovakia, Slovenia, South Korea, Spain, Switzerland, United States, Venezuela

b)

Fig. 1 (**a**) Scree plot for the ICS value data, to aid in the selection of the adequate number of components, and (**b**) cluster memberships of the countries in the cluster-wise SCA-P solution with two clusters and two components for the ICS value data

separately modeled with SCA-P, yielding a cluster-specific loading matrix. Mathematically, cluster-wise SCA-P models each data block $\mathbf{X}_i$ ($i = 1 \ldots I$) as follows:

$$\mathbf{X}_i = \sum_{k=1}^{K} \rho_{ik} \mathbf{F}_i \mathbf{B}^{(k)'} + \mathbf{E}_i. \quad (2)$$

The ρ_{ik} scores are the entries of the partition matrix $\mathbf{P}(I \times K)$; they are equal to one when data block i is assigned to cluster k and zero otherwise. The cluster-specific loading matrices are denoted as $\mathbf{B}^{(k)}$. Like De Roover, Ceulemans, Timmerman, and Onghena (2013), we restrict the number of components to be the same across the clusters in this article (for a more general cluster-wise SCA model, allowing the number of components to vary across the clusters, see De Roover, Ceulemans, Timmerman, Nezlek, & Onghena, 2013).

Deciding on the optimal numbers of components and clusters

In this article, we set the number of components for the cluster-wise extension equal to the number of components found in the SCA-P analysis of the data, as we expect to have the same number of components in each cluster. Otherwise, it

Table 1 Cluster-specific loadings of the cluster-wise SCA-P model with two clusters and two components for the ICS value data, after oblique rotation toward the orthogonally rotated SCA-P loadings

	Cluster I		Cluster II		SCA-P	
	Showing success & benevolence	Fun, happiness, & achievement	Showing success & benevolence	Fun, happiness, & achievement	Showing success & benevolence	Fun, happiness, & achievement
Material wealth	**.74**	.03	**.71**	.13	**.72**	.07
Physical attractiveness	**.80**	.05	**.76**	.12	**.77**	.09
Physical comforts	**.75**	.09	**.74**	.10	**.75**	.08
Excitement arousal	**.80**	.04	**.60**	.09	**.66**	.09
Competition	**.71**	.01	**.73**	.11	**.72**	.05
Heaven/afterlife	.28	.21	**.74**	– .16	**.60**	– .05
Self-sacrifice	.35	.24	**.63**	.00	**.54**	.08
Success	.31	**.59**	.35	**.74**	.30	**.66**
Fun	.14	**.48**	– .21	**.77**	– .15	**.72**
Happiness	.28	**.73**	.30	**.42**	.30	**.53**
Intelligence knowledge	– .10	**.85**	– .03	**.68**	– .08	**.75**

The full SCA-P loadings are also included in the table. Loadings with an absolute value higher than .40 are printed in bold

would not make sense to look for outlying variables. Once the number of components is fixed, the CHull procedure (for details, see "Lower bound congruence method" section) mentioned above is applied to the VAF% values for models with different numbers of clusters. For the ICS value data, the CHull indicates that the optimal number of clusters is two (see Fig. 1a). Panel b of Fig. 1 shows the associated clustering of the countries. Following De Roover, Timmerman, and Ceulemans (2017), the clusters can be labeled as *preindustrial* and *postindustrial* countries. Note that when more than two clusters are retained, LRUBCM will be applied to each pair of clusters, as we will demonstrate in "Illustrative applications" section.

Rotation Since cluster-wise SCA-P conducts a separate SCA-P analysis for each cluster, the cluster-specific component sets can be rotated independently of one another to improve interpretability. This article focuses on comparing the components across each pair of clusters, however. Therefore, we will rotate the component sets of both clusters in such a way that the corresponding components are maximally similar. To this end, we run an SCA-P analysis on the complete data, with the same number of components, to obtain one common loading structure and obliquely rotate the cluster-wise SCA-P components per cluster toward the VARIMAX-rotated SCA-P components. The resulting cluster-specific loading matrices for the ICS value data are shown in Table 1. We see, for example, that *physical comfort* loads similarly across the clusters, while *self-sacrifice* behaves differently in the two clusters.

Outlying-variable detection heuristics

The starting point of this article is that the cluster-specific loading patterns differ across two clusters for a few, unknown variables, while all the other variables have similar loadings. Our aim is to pinpoint the former set of variables, since they are outlying. To detect them, two things are needed. First, we have to specify some objective criterion for measuring the extent to which the cluster-specific loading matrices are similar. Second, we have to set a threshold for deciding that the matrices are similar enough, implying that the presence of outlying variables is unlikely.

Regarding the similarity criterion, De Roover, Timmerman, and Ceulemans (2017) used Tucker's congruence coefficient (Tucker, 1951), which is a well-established index of component and factor similarity (Chan et al., 1999; Lorenzo-Seva & ten Berge, 2006; Sočan, 2016). This coefficient measures the proportional similarity of the loadings on two components. For our problem, we compute it consecutively for each of the corresponding cluster-specific components—so, first for both first components, next for the second components, and so forth. Specifically, Tucker's congruence φ_q between the qth components of the two clusters is calculated as follows:

$$\varphi_q = \frac{\sum_j b_{jq}^1 b_{jq}^2}{\sqrt{\sum_j \left(b_{jq}^1\right)^2 \sum_j \left(b_{jq}^2\right)^2}}. \tag{3}$$

The obtained value varies from −1 to 1. A value close to zero means that the components are very dissimilar, whereas absolute values close to one indicate strong similarity.

Regarding the threshold value, below which we would conclude that the data still contain outlying variables, De Roover, Timmerman, and Ceulemans (2017) initially built on the results of Lorenzo-Seva and ten Berge (2006). These authors demonstrated that congruence values higher than .95 point toward virtual identity of the compared components in terms of their interpretation. Applied to our problem, this finding suggests that if the lowest congruence value across the components is lower than .95, the data probably would contain at least one more outlying variable, and thus that the .95 threshold can be used as a lower bound. This does not imply, however, that we are sure that no more outlying variables are present, if the congruence scores exceed .95. This makes sense, because the value of Tucker's congruence has been shown to depend on multiple data characteristics, such as the number of variables and components, the loading structure, and the amount of error in the data (Abdi, 2010; Lorenzo-Seva & ten Berge, 2006). This suggests that the congruence value that indicates sufficient similarity should be adapted to the data characteristics at hand. De Roover, Timmerman, and Ceulemans (2017) proposed to handle this issue by applying a scree-test-like procedure and put forward the LBCM.

In "Lower bound congruence method" section, we recapitulate the LBCM approach. In "The problem with LBCM and how to resolve it" section, we show its main weakness—that is, a tendency to yield false positives—which questions its applicability to real data. To overcome this weakness, in "Lower and resampled upper bound congruence method" section we propose to also impose an upper bound, based on the characteristics of the data. The new method is called the *lower and resampled upper bound congruence method*.

Lower bound congruence method

LBCM is a stepwise approach, consisting of $J-Q$ steps.[1] In each of these steps the most outlying variable is removed and

[1] After removing a variable, LBCM re-estimates the SCA-P solutions in each cluster. Therefore, we cannot remove more than $J-Q$ variables, because at least Q variables are needed to extract Q components. As a result, the substeps are executed $J-Q$ times, each time removing one variable.

the resulting increase in the Tucker's congruence is recorded. To this end, in each step the following three substeps are performed:

1. Compute the minimum and mean component congruences φ^{min} and φ^{mean} across the Q components, for the remaining variables.[2] The congruence coefficients are computed after the cluster-specific SCA-P loadings are obliquely rotated toward (i.e., target rotation) the orthogonally rotated overall SCA-P loadings. For example, the initial component-specific congruences for the ICS value data amount to .9346 and .8801, implying that the first recorded values are $\varphi^{min} = .8801$ and $\varphi^{mean} = .9074$ (Fig. 2a). φ^{mean} and φ^{min} are used for different purposes. The minimum congruence is compared to the lower bound, as will be explained below, to safeguard that the final set of retained variables will behave sufficiently similarly across the two clusters. The mean congruence is used for deciding on the number of outlying variables, as will be described below. Also, the assessment of which variable is the most outlying is based on φ^{mean} (Substep 2). One might wonder why De Roover, Timmerman, and Ceulemans (2017) used mean rather than minimum congruence in order to detect the number of outlying variables. The authors indicated that pilot simulations had shown that applying the minimum congruence yields even more false positives and is less robust than the mean congruence, when used for selecting the most outlying variable.
2. To decide which of the remaining variables is the most outlying, compute variable-specific congruence-after-exclusion scores, quantifying the mean congruence obtained by excluding each variable one by one. The variable with the highest congruence after-exclusion score is declared to be the most outlying variable. For the ICS value data, when considering all 11 variables, removing the variable *Heaven/afterlife* yields the highest congruence-after-exclusion score, and this variable is thus the most outlying.
3. Remove the most outlying variable and update the SCA-P solutions in each cluster and across all data blocks. Afterward, perform the target rotation described in Substep 1. Record the number of removed variables (which equals the step number minus one), the most outlying variable from Substep 2, and the φ^{min} and φ^{mean} values from Substep 1. In the first step, the recorded values are 0 (since no variables are removed yet), *Heaven/afterlife*, $\varphi^{min} = .8801$ and $\varphi^{mean} = .9074$ (see Fig. 2a).

[2] As we assume two clusters in this paper, we will get one congruence value per component and φ^{min} and φ^{mean} are the minimum and average thereof.

Afterward, the φ^{mean} values are plotted against the number of removed variables. Next, the CHull procedure is performed on the plot in order to find the number of outlying variables. Therefore, this procedure finds the number of outlying variables after which the increase in φ^{mean} due to additional removals levels off. Specifically, CHull first selects the solutions on the upper boundary of the convex hull of this plot and orders them according to the number of variables removed. Next, the following *st* criterion is computed on the S selected solutions.

$$st_s = \frac{\left(\frac{\varphi_s^{mean} - \varphi_{s-1}^{mean}}{l_s - l_{s-1}}\right)}{\left(\frac{\varphi_{s+1}^{mean} - \varphi_s^{mean}}{l_{s+1} - l_s}\right)} \tag{4}$$

Here, l_s denotes the number of variables removed in the sth CHull solution ($s = 1 \ldots S$). The solution with the highest st_s value will yield the final set of outlying variables, if its φ^{min} value exceeds .95. The latter restriction is imposed to make sure that the retained variables have sufficiently similar loadings. For the ICS value data, the solutions with zero, one, and two removed variables are excluded for this reason. This does not change the final conclusion, however, which indicates that eight variables should be removed.

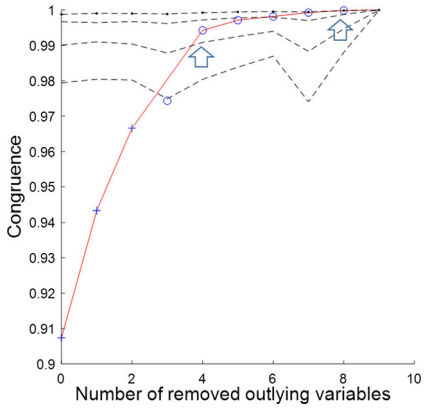
The problem with LBCM and how to resolve it

When applied to the ICS value data, LBCM indicates that eight variables should be removed, leaving us with three variables only. This is not very helpful: If we think of outlying-variable detection and removal as a prestep to prepare the data for further analysis and comparison, no applied researcher would be pleased with the suggestion to permanently remove eight out of 11 variables from the data and conduct the analysis with the remaining three variables only. Moreover, visually inspecting the solid line in the plot (Fig. 2b) would rather point toward four outlying variables, since the congruence scores are very high from that solution onward and hardly improve anymore. This result immediately demonstrates the main weakness of the LBCM approach: Although the method performs generally well on simulated data, it often removes a lot of, and probably too many, variables when applied to real data (see also the other applications in "Illustrative applications" section). In fact, the selected number of outlying variables is the maximum number that the method could have picked. We have observed the same selection behavior when the LBCM was applied to more challenging simulated data. Thus, for difficult settings, LBCM is prone to yield false positives.

This tendency to yield false positives is probably caused by a weakness of the CHull procedure. Analyzing Eq. 4 shows the following: The more variables are removed, the more φ_s^{mean} approximates φ_{s+1}^{mean} and the smaller the denominator of the ratio becomes, implying that a very small gain in

The number of removed variables	φ^{mean}	The most outlying variable	φ^{min}	Upper threshold $p_{max} = .05$	St values
0	.9074	Heaven/afterlife	.8801	.9959	-
1	.9433	Fun	.9225	.9962	-
2	.9666	Success	.9472	.9959	1.6783
3	.9743	Self-sacrifice	.9625	.9951	-
4	.9942	Excitement/arousal	.9942	.9964	**4.8069**
5	.9971	Physical comforts	.9949	.9972	2.6853
6	.9981	Happiness	.9969	.9974	-
7	.9993	Competition	.9992	.9958	1.7230
8	.9999	Material wealth	.9998	.9979	**5.4001**
9	.9999	Intelligence/knowledge	.9999	.9999	-

a)



b)

Fig. 2 (**a**) Outlyingness ranking matrix for the ICS value data using p_{max} of .05. The two highest *st* values are marked in bold and correspond to the arrows in panel b. LBCM yields eight outlying variables. This solution is discarded by the LRUBCM method using p_{max} of .05, because of the upper threshold; therefore, LRUBCM returns four outlying variables. (**b**) CHull/LRUBCM: The dashed lines from top to bottom represent the upper bounds using a p_{max} of .00, .05, .10, and .15, respectively. The solutions that are discarded because of the lower bound are marked as "+." Using a p_{max} of .00, LRUBCM yields eight outlying variables, and the other three p_{max} values result in four outlying variables

Tucker's congruence due to removing a variable will yield a very high *st* value. This weakness of CHull has been pointed out before, when it was used to solve model selection problems (i.e., to find the solution with the best goodness of fit vs. model complexity balance). In this context, Wilderjans et al. (2013) proposed to handle the issue by discarding a model if it does not imply a fit increase of at least 1% in comparison with the preceding model in the CHull plot. Although this 1% rule might be appropriate for the model selection problem, it is less applicable when using Tucker's congruence measure, as it is not clear what the minimal similarity gain should be, as this will probably depend on the data at hand.

Therefore, we propose a different way of discarding solutions, which explicitly takes the data characteristics into account. Specifically, we propose introducing an upper bound that is derived in each step of the method by building a sampling distribution of the Tucker's congruence measure, under the null hypothesis that the data do not contain outlying variables. If the observed mean congruence in a specific step is extremely low in comparison to this distribution (i.e., a low one-sided *p* value), this suggests that the data still contain outliers. Nonextreme congruence measures, on the other hand, indicate that all outlying variables have already been removed.[3] Therefore, we propose to discard all but the first solution that yields a nonextreme congruence, because selecting one of the previous solutions would imply finding false positives.

Lower and resampled upper bound congruence method

The new LRUBCM performs the same steps as the original LBCM procedure.[4] There are two modifications, however: In Substep 1, not only φ^{min} and φ^{mean} are computed, but also the sampling distribution of φ^{mean}, under the null hypothesis that no outlying variables are present. Each time, the upper threshold is saved, which is set to the fifth percentile of this sampling distribution.[5] The CHull step is also refined further, to incorporate this upper threshold. In particular, LRUBCM picks the solution that has the highest *st* value, after removing the solutions for which φ^{min} is lower than .95, or for which holds that a solution with less outlying variables has already crossed the upper bound.

Computing the sampling distribution of φ^{mean} (See the Appendix for the associated Matlab code; the full code of LRUBCM can be requested from the first author.) To be able to compute the sampling distribution of φ^{mean} under the null hypothesis that the data contain no outlying variables, we have to define what this hypothesis exactly entails. Thus, we revisit

[3] One might wonder why we do not simply use the sampling distribution of the Tucker's congruence measure under the null hypothesis to perform a statistical significance test. The presimulations showed that this type of test was not robust and produced false positives. This is due to the ambiguity of Tucker's congruence, implying that the measured congruence might still be extremely low in comparison to this distribution, while the data do not contain any more outlying variables. Using the distribution to construct an upper bound for the CHull procedure results in lower chances of picking a solution that includes false positives. Indeed, such solutions will probably result in a relatively small increase of the congruence.

[4] The LRUBCM is equally applicable when there are only two data blocks present (i.e. each cluster consists of one data block). In this case, during the third substep of the original procedure, the cluster-specific solutions can be obtained by performing ordinary principal component analysis.

[5] For the pilot simulations, we have tried significance levels equal to .01 and .05. Afterward, we retained the .05 significance level as it yielded more robust results across the "between" factors of the simulation design.

the introduction where we stated that we are only interested in larger loading differences, which alter the interpretation of the underlying constructs. To operationalize this notion of larger loading differences in a flexible way, we put forward that loading differences between variables do not alter the interpretation of the components as long as their absolute value does not exceed a user-specified threshold. For instance, it makes sense to assume that researchers will not be too alarmed about absolute differences that amount to .05 or .10 only. We will refer to this user-specified threshold as the maximally allowed perturbing value p_{max}. Note that users can obviously decide to be really strict and set p_{max} to zero. After prespecifying p_{max}, the sampling distribution is constructed by generating M resampled data sets that—in the case that p_{max} is nonzero—are simulated on the basis of two loading matrices with up to p_{max} absolute differences between the corresponding loadings. The number of resampled data sets, M, is set to 1,000 by default. Specifically, for each resampled data set, the matrices $\mathbf{B}^{(k)}$, $\mathbf{F}_i$, and $\mathbf{E}_i$ are obtained as follows, respectively:

- To obtain the two cluster-specific loading matrices $\mathbf{B}^{(1)}$ and $\mathbf{B}^{(2)}$, the observed data $\mathbf{X}$, minus the variables removed so far, are modeled with SCA-P, using the same number of components as in the original analysis. The SCA-P loading matrix $\mathbf{B}$ is VARIMAX-rotated toward simple structure, yielding $\mathbf{B}^{(1)}$. Next, to acquire $\mathbf{B}^{(2)}$, $\mathbf{B}^{(1)}$ is perturbed using p_{max}. Specifically, each of the loadings has 50% chance of being altered. If a loading is to be altered, the applied amount of perturbing is drawn randomly from the interval $[p_{max}/2, p_{max}]$. In 50%of the cases, the sampled value is added to the original loading, in the other cases, it is subtracted. We call $\mathbf{B}^{(2)}$ the "perturbed" loading matrix. Table 2 shows the VARIMAX-rotated SCA-P loadings for the ICS data and one example of a perturbed loading matrix, using $p_{max} = .10$.
- The $\mathbf{F}_i$ matrices of the groups are obtained by counter-rotating the corresponding original $\mathbf{F}_i$ matrices of the SCA-P solution, to compensate for the VARIMAX rotation. The counter-rotation boils down to postmultiplying $\mathbf{F}_i$ with $\mathbf{T}$, where $\mathbf{T}$ denotes the orthogonal rotation matrix. Indeed, $\mathbf{F}_i\mathbf{T}(\mathbf{BT})' = \mathbf{F}_i\ \mathbf{TT}'\mathbf{B}' = \mathbf{F}_i\ \mathbf{B}'$. Since we have perturbed the loading matrix of the second cluster, we reestimate the associated $\mathbf{F}_i$ matrices conditional upon the $\mathbf{X}_i$ matrices and $\mathbf{B}^{(2)}$, and subject to the SCA-P constraints on the $\mathbf{F}_i$ matrices.
- The error matrices $\mathbf{E}_i$ differ across the resampled data sets. They all are based on the residual matrices that can be computed for the original data, making use of the $\mathbf{F}_i$ and $\mathbf{B}^{(k)}$matrices obtained above: $\mathbf{X}_i - \mathbf{F}_i\mathbf{B}^{(k)'}$, where $\mathbf{B}^{(k)}$ indicates the cluster-specific loading matrix of the cluster to which $\mathbf{X}_i$ belongs. For each resampled data set, we generate new $\mathbf{E}_i$ matrices, by first computing residual matrices and then randomly permuting the entries within each column. The resulting reshuffled matrices, $\mathbf{E}_i$, are combined with the $\mathbf{F}_i$ and $\mathbf{B}^{(k)}$ matrices yielding the final resampled data set.

Table 2 Illustration of the perturbing step in the computation of the sampling distribution, using the ICS value data and $p_{max} = .10$

$\mathbf{B}^{(1)}$		$\mathbf{B}^{(2)}$	
.30	– .53	.30	– .59
– .08	– .75	– .14	– .83
.72	– .07	.72	– .14
.77	– .09	.67	– .02
.75	– .08	.75	– .14
.66	– .09	.66	– .03
.72	– .05	.72	– .05
.60	.05	.60	.15
.54	– .08	.49	– .13
.30	– .66	.39	– .66
– .15	– .72	– .15	– .81

Refining the CHull step As before, the CHull step first selects the solutions on the upper boundary of the convex hull of φ^{mean} versus the plot of the number of removed variables. These "hull" solutions are ordered according to the number of variables removed and the *st* values are calculated. Afterward, the solutions that do not satisfy the threshold criteria are discarded. In particular, CHull solutions with φ^{min} lower than .95 are set aside (lower bound), as well as all but the first solution for which φ^{mean} is higher than the upper resampled threshold (upper bound). From the retained solutions, the one with the highest *st* value is selected, and the associated set of variables is considered to be outlying.

For example, Fig. 2b shows the LRUBCM plot for the ICS value data for four different p_{max} values: .00, .05, .10, and .15. There are five CHull solutions, with two, four, five, seven, and eight associated sets of removed variables, before discarding any of them on the basis of the threshold criteria. The solution with two removed variables is discarded on the basis of the lower threshold restriction, as its φ^{min} is lower than .95. Which solutions are removed on the basis of the upper bound restriction depends on the p_{max} value used. For p_{max} equal to .00, .05, .10, and .15, the final sets of solutions to choose from are {4, 5, 7, 8}, {4, 5}, {4}, and {4}, respectively. The solution with the highest *st* value pertains to eight outlying variables, and the second highest to four. As a result, applying the LRUBCM with p_{max} equal to .00 indicates that the data contain eight outlying variables, while using any of the other p_{max} values points toward four outlying variables.

Simulation study

Problem

To compare the overall performance of the LBCM and LRUBCM approaches, we conducted a simulation study, based on the one by De Roover, Timmerman, and Ceulemans (2017). From the latter simulation study, we retained the two factors that affected performance most: the degree of outlyingness and the amount of error in the data. As we have already discussed above, we are not interested in small differences across the cluster-specific loadings. Thus, in order to evaluate performance in this regard, we have added one more factor: the maximum size of absolute differences in the loadings of nonoutlying variables. We hypothesize that a lower degree of outlyingness, a larger amount of error, larger absolute differences in the loadings of nonoutlying variables and fewer number of observations will complicate outlying-variable detection.

Design and procedure

Following De Roover, Timmerman, and Ceulemans (2017), the number of data blocks I was fixed at ten. Each simulated data set consisted of two equally sized clusters, with nine nonoutlying and four outlying variables. The number of underlying components Q was set to three per cluster. Four "between" factors were systematically varied in a complete factorial design:

1. The degree of outlyingness, at four levels: very high, high, medium, and low;
2. The error level, which is the expected proportion of error variance, v, in the data blocks, at two levels: .20 and .40;
3. The maximal amount of absolute differences, d_{max}, in the loadings of nonoutlying variables, at three levels: .00 .05, and .10;
4. The number of observations per data block: 25 and 75.

The factorial design thus contains $4 \times 2 \times 3 \times 2 = 48$ cells. For each cell of this design, 300 data matrices $\mathbf{X}$ were generated, yielding 14,400 datasets. Each data matrix consists of 10 $\mathbf{X}_i$ data blocks and, for each data block, a component score matrix $\mathbf{F}_i$ was randomly sampled from a multivariate normal distribution. To this end, the component variances and correlations were sampled between 0.25 and 1.75 and between −.5 and .5, respectively. To generate the partition matrix $\mathbf{P}$, half of the blocks were randomly selected and assigned to one of the clusters, and the rest were assigned to the other cluster.

To generate the cluster-specific loading matrices $\mathbf{B}^{(k)}$, the following steps were performed. To each of the three components, three different nonoutlying variables were assigned. If a nonoutlying variable is assigned to a component, it has a loading equal to 1; otherwise, the loading equals 0. In Cluster 1, outlying variables had a loading of 1 on a single component, whereas in the other cluster they received a loading b_{outl1} on the same component, but also a loading b_{outl2} on another component. Since there are four outlying variables and three components, one component contains two outlying variables, and the other two components only one. The loadings b_{outl2} were assigned in the same manner. The values of b_{outl1} and b_{outl2} depend on the degree of outlyingness. For very high, high, medium, and low degrees of outlyingness, the combinations of b_{outl1} and b_{outl2} are $\sqrt{.25}$ and $\sqrt{.75}$, $\sqrt{.50}$ and $\sqrt{.50}$, $\sqrt{.75}$ and $\sqrt{.25}$, and $\sqrt{.85}$ and $\sqrt{15}$, respectively.

In line with how we perturb when generating the sampling distribution, the perturbing of nonoutlying variables was implemented as follows: First, the loading matrices are rescaled by multiplying them by $\sqrt{1-v}$. Next, the loadings of the nonoutlying variables in the second cluster are perturbed, where the perturbing probability equals .5. If the loading is perturbed, then the perturbing amount is sampled randomly from the following interval $[d_{max}/2, d_{max}]$. In 50% of the cases, the sampled value is subtracted from the original loading; in the other 50%, it is added.

For the first cluster, the error matrix $\mathbf{E}_i$ was randomly sampled from a multivariate normal distribution with zero mean and identity covariance matrix and rescaled by multiplying it with $\sqrt{v}$. For the second cluster, the columns of $\mathbf{E}_i$ corresponding to outlying and nonoutlying variables are scaled differently as the sum of the squared loadings of the nonoutlying variables in the second cluster is slightly different due to the perturbing (i.e., it is distributed around $\sqrt{1-v}$, rather than exactly equal to it, for each variable). These changes are accounted for when rescaling the errors, to make sure that the variance of each variable equals one, once the error is added. Outlying variables are rescaled with v as they were not altered by the perturbing. Finally, each $\mathbf{X}_i$ is computed as $\mathbf{F}_i\mathbf{B}^{(k)'} + \mathbf{E}_i$. When analyzing the simulated data, the number of clusters was fixed to two and the number of components to three. Both the LBCM and the LRUBCM procedures were applied. For the LRUBCM, we selected the following three p_{max} values: .00, .05, and .10.

Results

Table 3 shows the percentages of datasets for which LBCM and LRUBCM correctly detected all outlying variables. By *correct detection*, we imply that the method identified exactly the four outlying variables that were present in the simulated data, neither less nor more. On average, the rounded percentages of correct detection equaled 71%, 76%, 78%, and 72 % ,for LBCM and LRUBCM with p_{max} values of .00, .05, and .10, respectively; LRUBCM thus performs better than LBCM overall. To test whether the performance differences between LBCM and LRUBCM with different p_{max} values are significant, we conducted three paired t tests comparing the results obtained with

Table 3 Percentage of simulated data sets for which the outlying variables are correctly identified by LBCM and LRUBCM with p_{max} of .00, .05 and .10, as a function of d_{max}, error%, degree of outlyingness and the number of observations per data block

d_{max}	Methods		20% Error	40% Error	Very high degree of outlyingness	High degree of outlyingness	Medium degree of outlyingness	Low degree of outlyingness	25 observations per block	75 observations per block	Overall
.00	LBCM		90	62	95	90	68	49	62	89	76
	LRUBCM	p_{max} .00	93	69	98	94	75	59	69	93	81
		p_{max} .05	94	71	98	95	77	60	70	94	82
		p_{max} .10	80	61	99	96	69	20	63	79	71
.05	LBCM		87	60	95	88	66	45	61	86	73
	LRUBCM	p_{max} .00	92	66	97	91	74	54	68	91	79
		p_{max} .05	93	68	98	92	76	57	68	93	81
		p_{max} .10	84	62	98	94	71	29	63	83	73
.10	LBCM		75	50	92	80	52	27	53	73	63
	LRUBCM	p_{max} .00	80	56	95	86	59	33	59	78	68
		p_{max} .05	84	59	96	88	63	38	61	82	71
		p_{max} .10	85	59	97	91	66	34	61	83	72
Overall	LBCM		84	57	94	86	62	40	59	83	71
	LRUBCM	p_{max} .00	89	64	97	90	69	49	65	87	76
		p_{max} .05	90	66	97	92	72	52	66	90	78
		p_{max} .10	83	61	98	93	69	28	62	81	72

LBCM in the 48 cells of the design with the corresponding results for the three LRUBCM variants. These tests revealed that LRUBCM with p_{max} of .00 [$t(47) = 7.86$, $p < .0001$] or .05 [$t(47) = 8.35$, $p < .0001$] clearly outperformed LBCM, whereas this did not hold for the differences between LBCM and LRUBCM with a p_{max} value of .10 [$t(47) = 0.64$, $p = .5248$].

The performance of the methods was not constant across the design, however. The general trend was for the performance of both LRUBCM and LBCM to decrease with a larger amount of error, a lower degree of outlyingness, and higher d_{max} values, and to be impacted even more by combinations thereof. One might wonder whether LRUBCM can outperform LBCM in the settings of the original simulation study (De Roover, Timmerman, & Ceulemans, 2017), implying a d_{max} of .00. To be as strict as possible, we compared the performance of LBCM with that of LRUBCM with $p_{max} = .00$. As one can see, LRUBCM not only performed better on average [$t(15) = 3.72$, $p = 0.0020$], but it also yielded better results for each of the corresponding cells of Table 3.

To gain insight into the tendency of both methods to output false positives, we inspected the difference between the observed and the true numbers of outlying variables. The mean values of these differences are shown in the Table 4. Positive numbers indicate that a method is inclined toward false positives, and negative numbers indicate that methods produce false negatives. From Table 4, we conclude that LRUBCM performs generally better than LBCM, except when LRUBCM is applied with a value of p_{max} that exceeds the value of d_{max} that was used to generate the data. In this case the comparison is not so straightforward, in that LBCM produces false positives, while LRUBCM generates false negatives.

To investigate whether incorrect detection results were due to incorrectly ranking the variables, we checked the ranking. Specifically, the ranking was considered to be incorrect if the first four most outlying variables in the outlyingness ranking were not the "true" outlying variables, regardless of the order. The order did not matter, since the level of outlyingness did not vary within the data sets. Table 5 shows the numbers of data sets with incorrect rankings across different d_{max} values and different numbers of observations per data block, error%s, and levels of outlyingness. The number of incorrect rankings drastically decreased when we increased the number of observations per data block. However, incorrect rankings often occur for the data sets with 25 observations per block and/or a low or medium degree of outlyingness. When the data sets contain 75 observations per block, incorrect rankings are only encountered when a low degree of outlyingness is combined with 40% error. Thus, for the rest of the cases we can conclude that when the outlying variables are detected incorrectly, this is mostly due to selecting the wrong number of outlying variables, rather than to an incorrect ranking.

For LRUBCM, one might expect that the performance using different p_{max} values would align to some extent with

Table 4 Mean differences between the detected number of outlying variables and the true number of outlying variables, after applying LBCM and LRUBCM with p_{max} of .00, .05 and .10, as a function of d_{max}, error%, degree of outlyingness, and the number of observations per data block

d_{max}	Methods		20% Error	40% Error	Very high degree of outlyingness	High degree of outlyingness	Medium degree of outlyingness	Low degree of outlyingness	25 Observations per block	75 Observations per block
.00	LBCM		0.24	0.69	0.18	0.24	0.62	0.85	0.67	0.25
	LRUBCM	p_{max} .00	0.02	0.08	0.04	0.05	0.05	0.06	0.04	0.05
		p_{max} .05	– 0.04	– 0.05	0.02	0.02	– 0.06	– 0.20	– 0.09	– 0.01
		p_{max} .10	– 0.40	– 0.50	0.00	– 0.03	– 0.40	– 1.63	– 0.45	– 0.44
.05	LBCM		0.33	0.67	0.18	0.31	0.62	0.99	0.65	0.35
	LRUBCM	p_{max} .00	0.03	0.12	0.04	0.09	0.06	0.08	0.04	0.10
		p_{max} .05	– 0.02	– 0.02	0.03	0.03	– 0.03	– 0.17	– 0.06	0.01
		p_{max} .10	– 0.28	– 0.40	0.00	– 0.03	– 0.30	– 1.23	– 0.36	– 0.31
.10	LBCM		0.59	0.93	0.22	0.55	1.05	1.42	0.81	0.68
	LRUBCM	p_{max} .00	0.24	0.30	0.07	0.21	0.36	0.54	0.20	0.32
		p_{max} .05	0.08	0.08	0.04	0.11	0.11	0.04	0.05	0.10
		p_{max} .10	– 0.18	– 0.28	0.02	0.02	– 0.29	– 0.89	– 0.27	– 0.19

Table 5 Numbers of incorrectly ranked data sets as a function of d_{max}, the number of observations per data block, error%, and degree of outlyingness

Data characteristics		Very high degree of outlyingness with 20% error	Very high degree of outlyingness with 40% error	High degree of outlyingness with 20% error	High degree of outlyingness with 40% error	Medium degree of outlyingness with 20% error	Medium degree of outlyingness with 40% error	Low degree of outlyingness with 20% error	Low degree of outlyingness with 40% error
25 observations per block	d_{max} .00	0	1	0	8	0	117	26	227
	d_{max} .05	0	2	0	9	3	146	29	242
	d_{max} .10	0	1	0	17	12	142	73	252
75 observations per block	d_{max} .00	0	0	0	0	0	2	0	27
	d_{max} .05	0	0	0	0	0	2	0	32
	d_{max} .10	0	1	0	0	0	17	10	94

the d_{max} value used to generate the data. This hypothesis would imply that for the data sets generated with a d_{max} value of .00, the best-performing LRUBCM variation would use a p_{max} value of .00; for the data sets generated with a d_{max} value of .05, LRUBCM with a p_{max} value of .05 would be best; and so forth. This was generally true for the data sets generated with d_{max} values of .05 and .10, whereas for the data sets generated with a d_{max} value of .00, p_{max} of .05 yielded slightly better performance than p_{max} of .00 (Table 3), since adding error to the data yielded small loading differences. LRUBCM performed notably worse when p_{max} exceeded d_{max} (Table 3), because it yielded too few outlying variables, and thus was prone to false negatives (Table 4). This tendency was also influenced by the degree of outlyingness of the outlying variables and the error level on the data. A lower degree of outlyingness and a higher error level led to a higher number of false negatives. In such cases, the congruence value for the observed data, even before removing all the outlying variables, was rather high. On the other hand, the congruence values for the resampled data sets would be relatively low, because during resampling a relatively high p_{max} is used. Therefore, the original congruence value would be higher than most of the resampled congruence values, implying that too few outlying variables would be detected.

To conclude, LRUBCM showed better performance than LBCM, not only in terms of correctly identifying all the outlying variables, but also in terms of false positives or negatives (Table 4). This pattern did not hold, however, for the data sets with $d_{max} = .00$ when LRUBCM was applied with a $p_{max} = .10$ value.

Illustrative applications

In this section, we will reanalyze another data set that was already screened for the presence of outlying variables (De Roover, Timmerman, De Leersnyder, et al., 2014). The data come from a study on emotional acculturation that was carried out by De Leersnyder, Mesquita, and Kim (2011). The starting point from this study was that, aside from many similarities, emotional experience also differs across cultures. Thus, the question is whether and how emotional experiences change when people move from one culture to another. The concept of emotional acculturation implies that immigrants start experiencing emotions in ways similar to those of their host culture (De Leersnyder et al., 2011). De Leersnyder et al. investigated two different host cultures (USA and Belgium) and included minority groups from different heritage cultures (the cultures from which the immigrants stemmed), yielding 13 samples of participants (see Table 6). Participants rated on 7-point Likert scales the degrees to which they had experienced 17 emotions (displayed in Table 7) in one to four different situations (for more details, see De Leersnyder et al., 2011). Similar to De Roover, Timmerman, De Leersnyder, et al. (2014), we considered each situation–participant combination as an individual observation. Note that observations with missing values were removed from the data set. The 17 variables were centered within each block and standardized across the blocks.

Analyzing these data with cluster-wise SCA-P, De Roover, Timmerman, De Leersnyder, et al. (2014) decided to retain a solution with three clusters and two components. The clustering of the participant samples is shown in Table 6. The first cluster consisted of the different cultural groups living in the USA as well as the Koreans. The second cluster consisted of the indigenous Belgian groups and second-generation Turkish immigrants living in Belgium, whereas the first-generation Turkish immigrants were clustered together with Turkish students living in Turkey in the third cluster. The assignment difference for the first- and second-generation Turkish immigrants is interesting, in that it shows that second-generation immigrants showed evidence of emotional acculturation.

De Roover, Timmerman, De Leersnyder, et al. (2014) used a precursor of the LBCM procedure to detect possibly outlying variables. This precursor procedure would probably output a different outlyingness ranking, as its ranking is based on φ^{min} instead of φ^{mean}. Another difference between this procedure and LBCM pertains to the stopping criterion, in that the former procedure did not rely on CHull, but rather went on until the minimum congruence between the clusters exceeded .96. The analysis with the precursor procedure indicated that seven variables could be considered outlying: *strong*, *proud about myself*, *surprised*, *relying*, *resigned*, *bored*, and *indebted*. Beyond

Table 6 Numbers of observations and cluster membership of the samples of the acculturation data, according to the clusterwise SCA-P solution with three clusters and two components

Samples	Retained observations	Cluster membership
European Americans 1 (USA)	120	1
Korean immigrants (USA)	126	1
Mexican immigrants (USA)	188	1
East-Asian immigrants (USA)	159	1
Latino immigrants (USA)	142	1
European Americans2 (USA)	122	1
Koreans (Korea)	298	1
Flemish students1 (Belgium)	183	2
Flemish students2 (Belgium)	516	2
Belgian community (Belgium)	166	2
Turkish 2nd generation immigrants (Belgium)	157	2
Turkish 1st generation immigrants (Belgium)	143	3
Turkish students (Turkey)	699	3

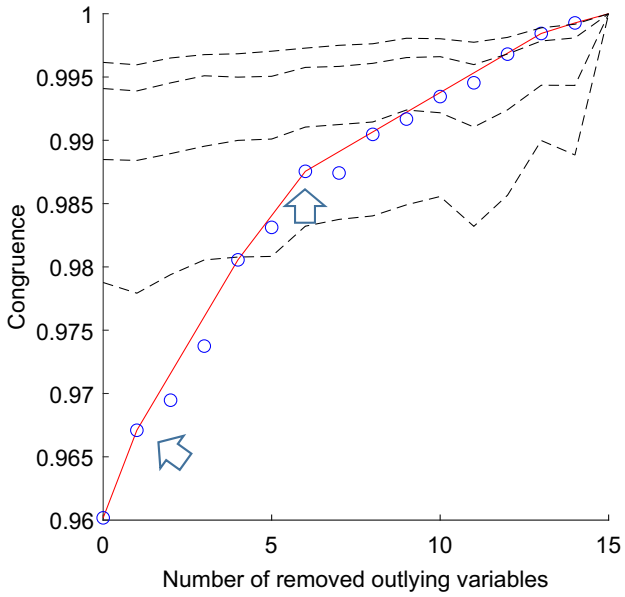
Table 7 Cluster-specific and overall SCA-P loadings for the pairwise comparisons of the Belgian cluster, Turkish cluster, and USA & Korea cluster

Emotions	Belgium and Turkey						USA & Korea and Turkey						USA & Korea and Belgium					
	Belgium		Turkey		Overall		USA & Korea		Turkey		Overall		USA & Korea		Belgium		Overall	
	N	P	*N*	P	*N*	P	*N*	P	*N*	P	*N*	P	*N*	P	*N*	P	*N*	P
Respect	− .17	**.79**	− .15	**.79**	− .17	**.79**	− .22	**.77**	− .18	**.76**	− .22	**.77**	− .24	**.77**	− .23	**.76**	− .24	**.77**
Interested	− .24	**.68**	− .08	**.66**	− .17	**.67**	− .29	**.68**	− .10	**.65**	− .23	**.66**	− .30	**.68**	− .28	**.64**	− .30	**.66**
Helpful	− .09	**.69**	− .12	**.58**	− .11	**.64**	− .22	**.72**	− .14	**.56**	− .20	**.66**	− .23	**.72**	− .14	**.67**	− .19	**.70**
Close	.01	**.77**	− .20	**.75**	− .08	**.75**	− .27	**.66**	− .22	**.71**	− .26	**.69**	− .28	**.66**	− .04	**.76**	− .18	**.69**
Strong	**− .47**	**.54**	− .24	**.69**	− .36	**.62**	− .31	**.64**	− .26	**.65**	− .31	**.66**	− .33	**.65**	**− .51**	**.46**	− .39	**.59**
Proud about myself	**− .56**	**.53**	− .31	**.68**	**− .44**	**.61**	**− .47**	**.62**	− .33	**.63**	**− .43**	**.63**	**− .48**	**.63**	**− .60**	**.45**	**− .52**	**.56**
Relying	.02	**.77**	− .05	**.72**	− .01	**.75**	.29	**.58**	− .07	**.71**	.13	**.65**	.27	**.58**	− .03	**.77**	.12	**.68**
Surprised	− .09	.34	− .08	**.68**	− .05	**.53**	.30	.29	− .10	**.66**	.13	**.48**	.30	.29	− .12	.33	.13	.35
Ill feelings	**.58**	**− .41**	**.52**	− .39	**.57**	− .38	**.58**	− .35	**.54**	− .31	**.57**	− .33	**.59**	− .35	**.61**	− .32	**.61**	− .33
Upset	**.64**	**− .41**	**.60**	**− .50**	**.62**	**− .44**	**.75**	− .31	**.62**	**− .40**	**.71**	− .35	**.76**	− .31	**.67**	− .31	**.72**	− .31
Irritated	**.51**	**− .54**	**.70**	− .24	**.60**	− .39	**.72**	− .22	**.71**	− .13	**.72**	− .20	**.73**	− .22	**.55**	**− .47**	**.67**	− .32
Embarrassed	**.64**	− .21	**.76**	− .19	**.71**	− .19	**.75**	− .10	**.78**	− .07	**.76**	− .11	**.75**	− .10	**.66**	− .11	**.72**	− .09
Ashamed	**.75**	− .23	**.78**	− .19	**.76**	− .21	**.74**	− .07	**.79**	− .06	**.76**	− .10	**.74**	− .07	**.77**	− .12	**.76**	− .10
Guilty	**.76**	− .19	**.76**	− .29	**.75**	− .25	**.72**	− .18	**.77**	− .17	**.74**	− .20	**.73**	− .18	**.78**	− .08	**.73**	− .15
Bored	.36	− .27	**.50**	**− .47**	**.43**	− .35	**.51**	.12	**.52**	− .38	**.53**	− .13	**.50**	.12	.38	− .22	**.48**	− .02
Indebted	**.70**	.20	**.57**	.26	**.64**	.23	**.41**	**.49**	**.57**	.36	**.46**	**.40**	.39	**.49**	**.68**	.30	**.52**	.37
Resigned	.38	.05	.24	.31	.34	.19	**.64**	− .07	.23	.35	**.48**	.13	**.65**	− .07	.37	.11	**.52**	.02

For each comparison, the cluster-specific loadings are obliquely rotated toward the orthogonally rotated SCA-P loadings. The loadings with an absolute value higher than .40 are printed in bold

The number of removed variables	φ^{mean}	The most outlying variable	φ^{min}	Upper threshold $p_{max} = .1$	St values
0	.9602	Irritated	.9590	.9788	-
1	.9671	Resigned	.9658	.9788	**1.5416**
2	.9695	Surprised	.9672	.9799	-
3	.9737	Proud about myself	.9637	.9805	-
4	.9806	Strong	.9761	.9806	1.2836
5	.9831	Bored	.9830	.9815	-
6	.9876	Interested	.9829	.9830	**2,2472**
7	.9874	Upset	.9873	.9836	-
8	.9905	Ill feelings	.9876	.9837	-
9	.9917	Embarrassed	.9898	.9856	-
10	.9935	Guilty	.9933	.9851	-
11	.9945	Close	.9908	.9830	-
12	.9968	Helpful	.9956	.9864	-
13	.9984	Ashamed	.9969	.9891	1,8658
14	.9993	Relying	.9988	.9885	1.1474
15	1.0000	Respect	1.0000	1.0000	-

a)



b)

Fig. 3 (**a**) The outlyingness ranking matrix for Belgium and Turkey data using p_{max} of .15. The two relevant highest *st* values are marked in bold and correspond with the arrows in panel b. LBCM yields six outlying variables. This solution is discarded by LRUBCM using p_{max} of .15, because of the upper threshold; therefore, LRUBCM returns one outlying variable. (**b**) CHull/LRUBCM: The dashed lines from top to bottom represent the upper bounds using p_{max} of .00, .05, .10, and .15, respectively. Using a p_{max} of .15, LRUBCM yields one outlying variable, and the other three p_{max} values result in six outlying variables

these seven variables, LBCM indicated that *irritated*, *close*, and *ill feelings* are outlying as well.

Because LRUBCM is tailored to comparing two clusters, we applied LBCM and LRUBCM to each pair of clusters (Belgium–Turkey, USA & Korea–Turkey, USA & Korea–Belgium) separately. Below we discuss the results for each pair.

Belgium–Turkey

The orthogonally rotated SCA-P loadings structure for the Belgian and Turkish clusters in Table 7 show that the positive emotions mainly load on the second component, whereas the negative emotions load on the first component. The emotion *proud about myself* has a strong negative loading on the negative component, which implies that when the participants of these samples experience negative emotions, they do not feel proud about themselves.

Implementing LBCM led to six outlying variables: *irritated*, *resigned*, *surprised*, *proud about myself*, *strong*, and *bored*, based on the elbow in the CHull plot in Fig. 3b. We used four perturbing values when building the LRUBCM sampling distribution of the congruence coefficient: .00, .05, .10, and .15. The CHull plot in Fig. 3b shows that applying the lowest three perturbing values led to the same six outlying variables. In contrast, the .15 perturbing value yielded only one outlying variable—*irritated*.

USA & Korea–Turkey

The loadings in Table 7 point toward the same component interpretation in terms of positive and negative emotions, but again with some subtle differences in whether and how strongly certain emotions are (inversely) related to a component. As is visualized in Fig. 4b, applying LBCM to these data indicates that 14 out of the 17 emotions are outlying, retaining only *guilty*, *strong*, and *ashamed*. Yet, visually investigating the CHull plot suggests that the data contain four outlying variables only. Once these four variables are removed, the mean congruence φ^{mean} exceeds .99. Thus, this cluster pair clearly showcases the tendency of LBCM to yield false positives. No applied researcher would like to remove 14 out of 17 variables from the data permanently in order to conduct cross-cultural comparative analysis.

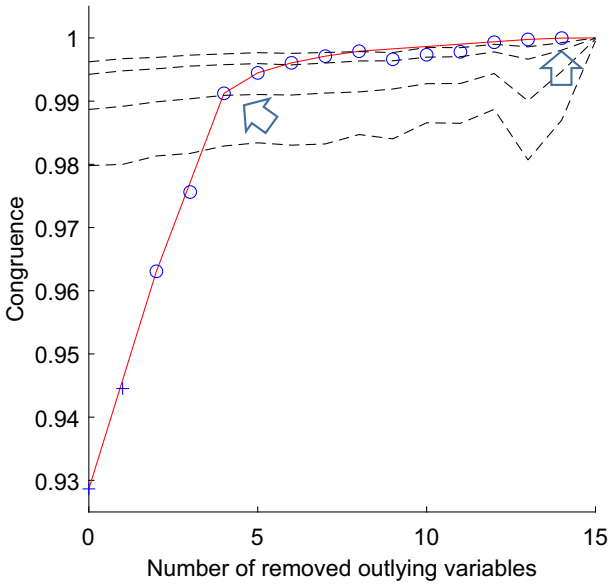
LRUBCM performed better for these data, since using any of the four considered perturbing values successfully eliminated the solution with 14 variables from the pool of possible solutions. Specifically, LRUBCM always suggested removing *resigned*, *surprised*, *bored*, and *relying*.

USA & Korea–Belgium

When comparing the USA & Korea cluster to the Belgian one, LBCM indicated that ten variables were outlying and should be removed (Fig. 5b). This is again probably too high a number.

The number of removed variables	φ^{mean}	The most outlying variable	φ^{min}	Upper threshold $p_{max} = .1$	St values
0	.9286	Resigned	.9214	.9796	-
1	.9445	Surprised	.9415	.9801	-
2	.9631	Bored	.9581	.9809	1.2224
3	.9756	Relying	.9643	.9818	-
4	.9912	Indebted	.9886	.9825	**4.3481**
5	.9945	Irritated	.9944	.9830	2.0402
6	.9961	Close	.9947	.9827	1.5079
7	.9971	Helpful	.9970	.9836	1.3417
8	.9979	Interested	.9972	.9837	2.1039
9	.9966	Ill feelings	.9966	.9838	-
10	.9973	Respect	.9955	.9863	-
11	.9978	Upset	.9971	.9864	-
12	.9993	Proud about myself	.9989	.9891	-
13	.9998	Embarrassed	.9997	.9806	1.7673
14	1.0000	Guilty	1.0000	.9889	**7.0706**
15	1.0000	Strong	1.0000	.9995	-

a)



b)

Fig. 4 (**a**) The outlyingness ranking matrix for USA & Korea and Turkey data using p_{max} of .15. The two highest *st* values are marked in bold and correspond with the arrows in panel b. LBCM yields 14 outlying variables. This solution is discarded by LRUBCM using p_{max} of .15 because of the upper threshold; therefore, LRUBCM returns four outlying variables. (**b**) CHull/LRUBCM: The dashed lines from top to bottom represent the upper bounds using p_{max} of .00, .05, .10, and .15, respectively. The solutions that are discarded because of the lower bound are marked as "+." Using p_{max} of .00, .05, .10, or .15, LRUBCM yields four outlying variables

The recommendations resulting from applying LRUBCM depended on the p_{max} value used. This is because the CHull plot shows two elbows: one associated with four outlying variables, and another with ten. The last elbow has a higher *st* value, so when both solutions are in the pool of considered elbows, it will be picked. The first elbow, with four associated outlying variables, is picked when using p_{max}values of .10 or.15, as this eliminates the last elbow. The latter choices of p_{max} make sense from a practical perspective, in that lower absolute loading differences probably do not change the interpretation of the components at all, so we recommend discarding the following four variables: *surprised*, *relying*, *bored*, and *resigned.*

Summary

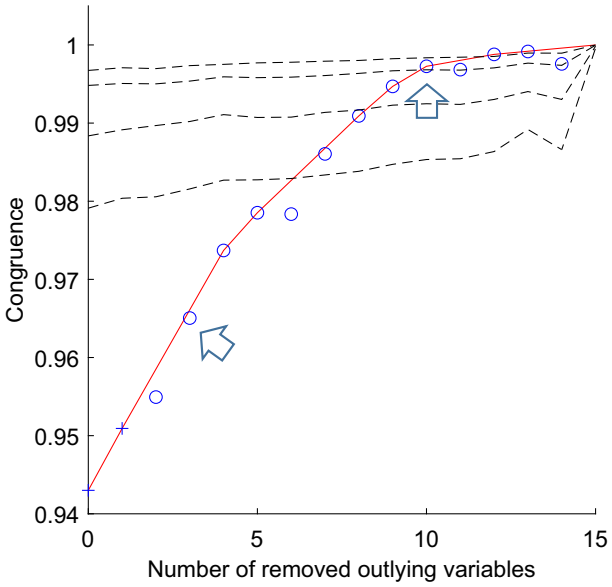
This application showed again that LBCM tends to return a high number of outlying variables. Applying LRUBCM with a p_{max} value of .15 effectively handles this overextraction and yields the strongest outlying variables only. If we focus on the results obtained with .10 and .15 p_{max} values, the variables *surprised*, *bored*, *resigned*, and r*elying* were declared outlying when comparing the USA & Korea cluster to each of the other two clusters. Although *relying* seems to be a positive emotion in the Belgian and Turkish clusters, it has a negative undertone as well for Americans and Koreans. *Surprised* loaded similarly high on the positive and negative emotion components in the USA & Korea cluster, whereas for the Belgian and Turkish clusters, it loaded more strongly on the positive emotion component. *Resigned* loaded primarily on the positive emotion component in the Turkish cluster, whereas for the USA & Korean and Belgian clusters it loaded mainly on the negative emotion component. The variable *bored* had a small positive loading on the positive emotion component for the USA & Korea cluster, whereas it has stronger negative loadings on this component in the other two clusters. Finally, *irritated* had a different loading pattern in the Belgian and Turkish clusters, in that it was more clearly a negative emotion in the Belgian cluster.

Discussion

There is a clear need for methods to detect outlying variables when performing exploratory dimension reduction in psychology. Outlying variable detection can be applied as a pre-step when measurement invariance (Meredith, 1993) is a concern and one has no strong a priori hypothesis about the underlying dimensions. It allows to have a first look at which variables may be causing noninvariance or to identify the least outlying variables that are suitable as anchor items for a subsequent multigroup EFA or CFA. Finding outlying variables is also of great interest

The number of removed variables	φ^{mean}	The most outlying variable	φ^{min}	Upper threshold $p_{max} = .1$	St values
0	.9430	Surprised	.9344	.9792	-
1	.9509	Relying	.9492	.9806	1.0435
2	.9549	Bored	.9500	.9804	-
3	.9650	Resigned	.9569	.9816	-
4	.9737	Irritated	.9660	.9824	**1.5715**
5	.9785	Indebted	.9700	.9827	1.1698
6	.9784	Strong	.9712	.9828	-
7	.9861	Proud about myself	.9847	.9834	-
8	.9909	Ill feelings	.9903	.9840	1.0935
9	.9947	Close	.9933	.9853	1.4683
10	.9973	Ashamed	.9971	.9859	**3.3724**
11	.9968	Helpful	.9944	.9856	-
12	.9988	Guilty	.9987	.9860	1.8956
13	.9992	Upset	.9988	.9887	-
14	.9976	Interested	.9955	.9850	-
15	1.0000	Respect	1.0000	.9996	-

a)



b)

Fig. 5 (**a**) The outlyingness ranking matrix for USA & Korea and Belgium data using $p_{\max}$ of .15. The two relevant *st* values are marked in bold and correspond with the arrows in panel b. LBCM yields ten outlying variables. This solution is discarded by LRUBCM using $p_{\max}$ of .15, because of the upper threshold; therefore, LRUBCM returns four outlying variables. (**b**) CHull/LRUBCM: The dashed lines from top to bottom represent the upper bounds using $p_{\max}$ of .00, .05, .10, and .15 respectively. The solutions that are discarded because of the lower bound are marked as "+." Using $p_{\max}$ of .00 and .05, LRUBCM yields ten outlying variables, and the other two $p_{\max}$ values result in four outlying variables

for comparative research (Triandis, 1988; Esser & Hanitzsch, 2013), well beyond the measurement invariance issue, as it can help distinguish between culture-specific and universal features.

In this article, we started from an existing heuristic, LBCM, because of its many strengths and advantages. LBCM yields objective, computation-based information and is therefore perfectly reproducible. Interestingly, it also ranks the variables in term of outlyingness and sheds light on how the cluster-specific component structures converge toward a common structure when removing one variable after the other. Finally, it accounts for the influence of outlying variables on component extraction and rotation, as the components are re-estimated and rotated after every removal, which avoids false positives and negatives. Nevertheless, we showed that LBCM is not without issues, in that it is still prone to pick up small loading differences that may be false positives. This hampers applying the method in practice, because users are left with too few variables. For example, for the ICS value data, eight out of 11 variables were detected as outlying, which is the maximum number that the method could have picked.

To overcome this false-positive issue, we have presented the new LRUBCM approach. LRUBCM extends LBCM by adding two new features. First, next to the lower bound, an upper threshold is imposed as well, to discard solutions with too many outlying variables. This upper threshold is derived from the sampling distribution of the Tucker's congruence coefficient under the null hypothesis that the data contain no outlying variables. Second, while building this sampling distribution, LRUBCM does not impose the loadings to be exactly the same across the two clusters, but checks for similarity instead by tolerating small loading differences, which are quantified by the selected $p_{\max}$ value.

In comparison with LBCM, LRUBCM shows better performance on simulated data, not only in terms of the number of data sets for which outlying variables are correctly identified, but also in terms of the number of false positives. LRUBCM yields a fair amount of false negatives, however, when p_{max} is much higher than the $d_{\max}$ value that the data was simulated with. This means that it is fairly important that researchers have a good idea about which size of loading differences are irrelevant and choose the value of $p_{\max}$ accordingly. Furthermore, the most appropriate $p_{\max}$ value depends, amongst others, on the amount of observations and the amount of noise in the data as both aspects impact the reliability of the loadings. In this regard, one might argue that we have replaced one arbitrary choice (when is Tucker's congruence high enough to be reasonably sure that the data contain no outlying variables anymore) with another (which $p_{\max}$ value should be used). However, we think that while, from an interpretation point of view, the meaning of a congruence

coefficient is sometimes hard to grasp as it is influenced by many factors, choosing the maximum absolute loading difference that one is willing to tolerate is more intuitive and moreover adds to the flexibility of the method.

One may wonder in which respects our procedure differs from the bootstrap-based approach for outlying variable detection that was proposed by Chan et al. (1999). Indeed, Chan also uses the congruence coefficient as a similarity measure and applies it in three different ways: to compare the loadings of a specific variable or a specific factor or to evaluate the complete loading matrix. As these authors suggest, the overall and factor similarity could be used to assess the overall and factor-specific invariance, while variable congruence (computing the congruence row-wise) can be used to detect outlying variables. This method also differs from LRUBCM. The first difference pertains to the strictness of the invariance assumption. Chan's method evaluates exact equality of the rotated loadings of nonoutlying variables, while LRUBCM tests for similarity, tolerating small differences according to user preferences. The second difference between Chan's bootstrap on the one hand and LRUBCM and LBCM on the other hand is that the first method assesses the invariance of all variables simultaneously, while LRUBCM and LBCM are stepwise approaches and remove one outlying variable only in each step. We argue that the stepwise approach is justified by the influence that the outlying variables may have on the loadings of the nonoutlying variables, where they affect both which components are extracted and how they are rotated. This can be observed in the results of the Chan and colleagues simulation study, as well: Nonoutlying variables had a higher chance to be declared as significantly different across the data sets as the overall number of outlying variables increased.

An attentive reader might argue that when building the sampling distribution under the null hypothesis of no outlying variable, we treat the error matrix **E** as purely random, which actually does not hold in component analysis models, in contrast to common factor models. Indeed, it is very likely that we cannot always explain all of the systematic variance in the data by means of the retained components. As a result **E** will contain some systematic structure also and this structure will be broken after randomly permuting the entries of the matrix **E** within each column. Also, whereas the original **E** matrix is of rank $J-Q$, the rank of the permuted version equals J. Given our simulations results, we do not think that this is a major issue, however. Yet, it might be an interesting direction for future research to replace the component analysis approach with a factor analysis one, to avoid this issue.

The choice of the number of components and how it affects detection results deserves further investigation as well in the future. For now, we hypothesize that we might end up with different sets of outlying variables if we analyze the same data with different number of components. Indeed, due to the rotation, different sets of retained components will have different loading patterns and therefore different interpretations; consequently, which variables are the most outlying ones in terms of these components might differ as well. Moreover, when one overextracts and retains components that are noise driven, we expect the noise components to be very different across the two clusters, even before rotation, implying that many variables should be removed. In case of underextraction, on the other hand, some of the components might be merged or averaged in some way, and this merging or averaging might again differ between clusters. This difference would lower the similarity of the components and thus affect outlying-variable detection.

As a final note, in this article we applied LRUBCM in a specific setting, pertaining to exploratory component analysis of many data blocks. However, the method is not limited to this setting, as it can be used to compare any two loading structures for the same variables, which may result from various types of component or factor analysis. As examples, LRUBCM can be used to detect which variables hamper the similarity of loading matrices obtained in a mixtures of factor analyzers (De Roover, Vermunt, Timmerman, & Ceulemans, 2017; McLachlan & Peel, 2000; Yung, 1997), a subspace k-means analysis (Timmerman, Ceulemans, De Roover, & Van Leeuwen, 2013) or a switching principal component analysis (De Roover, Timmerman, Van Diest, Onghena, & Ceulemans, 2014).

Author note The research leading to the results reported in this article was supported in part by the Research Fund of KU Leuven (GOA/15/003) and by the Interuniversity and Attraction Poles program financed by the Belgian government (IAP/P7/06). For the simulations, we used the infrastructure of the Flemish Supercomputer Center (VSC), funded by the Hercules foundation and the Flemish Government, department EWI. The data and materials can be requested from the first author, and the study was not preregistered.

Appendix: Matlab code for producing a the sampling distribution of φ^{mean}

For the future, for LRUBCM we plan to develop a Matlab standalone package, which can be used without having a Matlab environment installed on the computer. Meanwhile, the code, used in this paper, can be requested from the first author. As we think that technical details of the algorithm might be better conceptualized when accompanied with an actual implementation, we decided to deliver the main functions. Some functions—"makecongruencedistribution.m," "makeresampledperturbeddata.m," "perturbperloading.m," "reshufflematrixcolumnwise.m," and "clusterwiseSCAP-withpartition.m"—were developed specifically for this article/project, while the rest of the functions used—"normvari.m," "nrm.m," "OblProcrRot.m," "phi.m," "procr.m," "cent.m," "ssq.m," and "varim.m"—are a part of an available software package by De Roover, Ceulemans, and Timmerman (2012).

Makecongruencedistribution.m

```
function [sortedCongruencesOfResampledData] =
makecongruencedistribution(data, K, partvec, Q, C, numOfPerm, allowedPert)
% data       - Multivariate multiblock data
% K          - Vector (size:I) with # occasions for block 1,...,I
% partvec    - Partition matrix, which represents how I data blocks are
%              partitioned into two mutually exclusive clusters.
%              Vector elements take values: 1 2
% Q          - The number of components
% C          - The number of clusters
% numOfPerm  - The number of resampled data sets
% allowedPert - The maximally allowed perturbing value

I = length(K);
kcum=zeros(I,2);
for n=1:I,
    kcum(n,1)=(sum(K(1:n-1))+1); % makes I by 2 matrix, where kcum(i,1)
and kcum(i,2)  corresponds to the first and last row in the data for i-th
block
    kcum(n,2)=(sum(K(1:n)));
end;
% Preprocessing
% Center in each block and normalize across blocks
data_processed=[];
for n=1:I,
    Xi=data(kcum(n,1):kcum(n,2),:);
    Z=cent(Xi);  % Gives columnwise centered version of matrix Xi
    data_processed=[data_processed;Z];
end
% nrm makes columnwise normalized version of data_processed
data_processed=nrm(data_processed).*sqrt(sum(K));

% nPermData(l,m,1), nPermData(l,m,2), .... are resampled data sets
[nPermData] = makeresampledperturbeddata(data_processed, partvec,
K,Q,C,numOfPerm,allowedPert);

congruencesOfResampledData = zeros(1,numOfPerm);
% Calculate the statistics
for j = 1: numOfPerm
    [U,S,V]=svd(nPermData(:,:,j),0);
    jSCA_P_B=sqrt(1/sum(K))*V(:,1:Q)*S(1:Q,1:Q);
    % B1 and B2 are returned side by side
    [jclustterwise_SCA_P_B, ~, ~ ] =
clusterwiseSCAPwithpartition(nPermData(:,:,j ), I, K, Q, C, partvec);
    % Procurus rotation
    % Rotate each cluster loading matrix obliquely towards SCA-P solution
    [B1roteted,~,~]=OblProcrRot(jclustterwise_SCA_P_B(:,1:Q),
normvari(jSCA_P_B) ,0,1e-6,10000);

[B2rotated,~,~]=OblProcrRot(jclustterwise_SCA_P_B(:,Q+1:Q*C),normvari(jSCA_
P_B),0,1e-6,10000);

    congruencesOfResampledData(j) = mean( diag( phi(B1roteted,
B2rotated)));
end
    sortedCongruencesOfResampledData = sort(congruencesOfResampledData);
end
```

Makeresampledperturbeddata.m

```
function [ nPermData ] = makeresampledperturbeddata( DataProcessed,
partition, K,Q,C,numOfPermut,perturbValues )
% DataProcessed - Centered  and normalized data
% partition     - Partition matrix, which represents how I data blocks
% are partitioned into two mutually exclusive clusters.
% Vector elements take values: 1 2
% K             - Vector (size:I) with # occasions for block 1,...,I
% Q             - The number of components
% C             - The number of clusters
% numOfPermut   - The number of resampled data sets
% perturbValues  - The maximally allowed perturbing value

I = length(K);
kcum=zeros(I,2);        % makes I by 2 matrix, where kcum(i,1) and
kcum(i,2)  corresponds to the first and last row in the data for i-th block
for n=1:I,
    kcum(n,1)=(sum(K(1:n-1))+1);
    kcum(n,2)=(sum(K(1:n)));
end;

[l,m] = size(DataProcessed);

% Get SCA-P solutions for the full data
[U,S,V] = svd(DataProcessed,0);
SCA_P_B = sqrt(1/sum(K))*V(:,1:Q)*S(1:Q,1:Q);
SCA_P_F = sqrt(sum(K))*U(:,1:Q);

% Rotate using varimax rotation / T is rotation matrix
[SCA_P_B_rotated ,T ,~] = normvari(SCA_P_B);
% Counter-rotate F
% as a result SCA_P_F*SCA_P_B' = SCA_P_F_counterRotated*SCA_P_B_rotated'
SCA_P_F_counterRotated = SCA_P_F * T;

% nPermData(l,m,1), nPermData(l,m,2), .... are resampled data sets
nPermData = zeros(l,m,numOfPermut);
for i = 1:numOfPermut

    BJperturbedPerLoading = [];
    numRows = size(SCA_P_B_rotated,1);
    % This for loop will go through each row of the loading matrix and
    % perturb each row individually.
    for ind = 1:numRows
        row = SCA_P_B_rotated(ind,:);
        loopNumber = 0;
        % try to perturb each row until the sum of squared loadings is
        % less than 1. Try to perturb each row max 100 times, Otherwise
        % leave it the way it is.
        while true
            perturbedRow= [];
            if sum(row.^2, 2) > 1.0 | loopNumber > 100
                perturbedRow= row;
                break;
            end
            perturbedRow= perturbperloading(row ,perturbValues);
            if sum(perturbedRow.^2, 2) < 1.0
                break;
            end
            loopNumber = loopNumber + 1;
        end
        BJperturbedPerLoading = [BJperturbedPerLoading; perturbedRow];
    end

    % Collect cluster data together / we really only need XC2
    XC1 = [];
    XC2 = [];
    for n=1:I
        Xn = DataProcessed(kcum(n,1):kcum(n,2),: );
        if partition(n) == 1
            XC1 = [XC1 ; Xn];
        else
            XC2 = [XC2 ; Xn];
        end
    end

    subj_c2=find(partition==2)';
    K_c=K(subj_c2); % getting indices for the blocks from second cluster

    % Re-estimate F for the second cluster
    [Un, Sn, Vn] = svd((XC2*BJperturbedPerLoading), 'econ' );
    Frees2 = sqrt(sum(K_c ))* Un*Vn'; % F for cluster 2

    % We will need this indices to get block score matrices from re-
estimated cluster
    kcum_c=zeros(I,2);
    for n=1: length(K_c)
        kcum_c(n,1)=(sum(K_c(1:n-1))+1);
        kcum_c(n,2)=(sum(K_c(1:n)));
    end;

    j = 1; % To go through F_rees_2
    FBreestimated = [];
```

```
    for n=1:I

        if partition(n) == 1 % original - NOT changed data
            Fcurrent = SCA_P_F_counterRotated(kcum(n,1):kcum(n,2),:);
            Bcurrent = SCA_P_B_rotated;
        else  % Re-estimated / B_perturbed_per_loading's equivalent F
            Fcurrent = Frees2(kcum_c(j,1):kcum_c(j,2),: );
            j = j +1;
            Bcurrent = BJperturbedPerLoading;
        end
        FBcurrent = Fcurrent * Bcurrent';
        FBreestimated = [ FBreestimated; FBcurrent];
    end

    % Re-estimate and reshuffle E in columns
    Ereestimated = DataProcessed - FBreestimated;
    E_reestimated_reshuf = reshufflematrixcolumnwise(Ereestimated);

    % Final resampled data
    Data_altered = FBreestimated + E_reestimated_reshuf;

    % Center in each block and normalize across blocks
    permData = [];
    for n=1:I,
        Xi=Data_altered(kcum(n,1):kcum(n,2),:);
        Z=cent(Xi);
        permData=[permData;Z];
    end
    permData=nrm(permData).*sqrt(sum(K));

    nPermData(:,:,i) = permData;
end
end
```

perturbperloading.m

```
function [ Bperturbed ] = perturbperloading( B, perturbValues )
% B            - Loading matrix that needs to be perturbed, can be
% a vector.
% perturbValues - The maximally allowed perturbing value.
% Each of the loading has 50% chance of being altered.
% If a loading is to be altered, the applied amount of perturbing
% is drawn randomly from the interval [perturbValues/2 , perturbValues].
% In 50% of the cases, the sampled value is added to
% the original loading, in the other cases, it is subtracted.

Bperturbed = zeros(size(B));
for ind = 1:numel(B) % Goes through all the B matrix elements

    willBePerturbd = binornd(1,1./2); % Returns 1 or 0 by 0.5 chance

    if willBePerturbd == 1 % then perturb
        %perturbning value is a random number [perturb_values/2 ,
perturb_values]
        perturb_values_adj = perturbValues/2 + rand(1,1)*perturbValues/2;

        dirrectionOfPerturbing = binornd(1,1./2);

        if dirrectionOfPerturbing == 1 % perturb in '+' direction
            Bperturbed(ind) = B(ind) + perturb_values_adj;
        else % perturb in '-' direction
            Bperturbed(ind) = B(ind) - perturb_values_adj;
        end

    else   % Don't perturb
        Bperturbed(ind) = B(ind);
    end
end

end
```

reshufflematrixcolumnwise.m

```
function [ columnwiseReshMatrix ] = reshufflematrixcolumnwise( matrix )
% Reshuffle elements in columns of the matrix

[row, col] = size(matrix);
columnwiseReshMatrix = [];
for j = 1:col
    jCol = randsample(matrix(:,j), row);
    columnwiseReshMatrix = [columnwiseReshMatrix, jCol ];
end
end
```

clusterwiseSCAPwithpartition.m

```
function [ B, F, loss ] = clusterwiseSCAPwithpartition( Zsup, I, K, Q, C,
partvec )
% Calculate the SCA-P solutions for each cluster with fixed partition
% I - number of subjects (Blocks)
% K - vector with number of observations for subjects in it
% Q - number of components
% C - number of clusters
% partvec - partition vector

kcum=zeros(I,2);
for n=1:I,
    kcum(n,1)=(sum(K(1:n-1))+1);
    kcum(n,2)=(sum(K(1:n)));
end;

B=[ ];
F=zeros(sum(K),C*Q); % Initialization of Fsup
for c=1:C % Perform SCA-P per cluster
    subj_c=find(partvec==c)';
    K_c=K(subj_c); % A vector with the K_n's for the subjects assigned to
cluster cl
    Zsup_c=[ ];
    kcum_c=zeros(length(subj_c),2);
    for n_c=1:length(subj_c)
        Z=Zsup(kcum(subj_c(n_c),1):kcum(subj_c(n_c),2),:);
        Zsup_c=[Zsup_c;Z];
        kcum_c(n_c,1)=(sum(K_c(1:n_c-1))+1);
        kcum_c(n_c,2)=(sum(K_c(1:n_c)));
    end;

    [U,S,V]=svd(Zsup_c,0);
    cB=sqrt(1/sum(K_c))*V(:,1:Q)*S(1:Q,1:Q);
    B=[B,cB];
    for n_c=1:length(subj_c)
        F(kcum(subj_c(n_c),1):kcum(subj_c(n_c),2),(c-
1)*Q+1:c*Q)=sqrt(sum(K_c))*U(kcum_c(n_c,1):kcum_c(n_c,2),1:Q);
    end
end
% Find the loss of a model
FB = F*B';
loss = ssq(Zsup - FB);
end
```

Normvarim.m

```
function [B,T,f]=normvarim(A);
% Produces NORMALIZED varimax rotated version of A and rotation matrix T
%
% input: A    (matrix to be rotated)
% output: B (rotated A)
%     T (rotation matrix)
%     f (varimax function)
% Uses: varim.m

[m,r]=size(A);
d=sum(A.^2')';  % check on near zero rows, which will not be weighted
w=d<1e-16;
d(w)=ones(sum(w),1);
A=A./((d.^.5)*ones(1,r));
[B,T,f]=varim(A);
B=((d.^.5)*ones(1,r)).*B;
end
```

Varim.m

```
function [AT,T,f]=varim(A);
% produces varimax rotated version of A and rotation matrix T
% Based on nevels 1986; see also pascal source
%
% input: A    (matrix to be rotated)
% output: AT (rotated A)
%     T (rotation matrix)
%     f (varimax function)
% uses: ssq.m

conv=.000001;
[m,r]=size(A);
%T=eye(r);
T=eye(r);
B=A;

f=ssq((A.*A)-ones(m,1)*sum(A.*A)/m);
fold=f-2*conv*f;
if f==0, fold=-conv; end;
iter=0;

while f-fold>f*conv
  fold=f;iter=iter+1;
  for i=1:r
    for j=i+1:r
      x=B(:,i);y=B(:,j);
      xx=T(:,i);yy=T(:,j);
      u=x.^2-y.^2;v=2*x.*y;
      u=u-ones(m,1)*sum(u)/m;v=v-ones(m,1)*sum(v)/m;
      a=2*sum(u.*v);b=sum(u.^2)-sum(v.^2);c=(a^2+b^2)^.5;
      if a>=0; sign=1; end;
      if a<0; sign=-1; end;
      if c<.00000000001
        vvv=-sign*((b+c)/(2*c))^.5;
        cos=1;sin=0;
      end;

      if c>=.00000000001
        vvv=-sign*((b+c)/(2*c))^.5;
        sin=(.5-.5*vvv)^.5;cos=(.5+.5*vvv)^.5;
      end;

      v=cos*x-sin*y;w=cos*y+sin*x;
      vv=cos*xx-sin*yy;ww=cos*yy+sin*xx;

      if vvv>=0       % prevent permutation of columns
        B(:,i)=v;B(:,j)=w;T(:,i)=vv;T(:,j)=ww;
      end;
      if vvv<0
        B(:,j)=v;B(:,i)=w;T(:,j)=vv;T(:,i)=ww;
      end;
    end;
  end;
  f=ssq((B.*B)-ones(m,1)*sum(B.*B)/m)   ;

AT=B;
end
```

OblProcrRot.m

```
function [AT,T,f]=OblProcrRot(A,Y,start,conv,maxit,T)
% [AT,T,f]=OblProcrRot(A,Y,start,conv,maxit);
% Using general procedure by Jennrich (2002; PM, p.7-20)
%
% minimizes || A*T - Y ||
% constraint Diag(T^-1 T^-1') = I
% NOTE: Procedure assumes ORTHOGONAL factor scores at start
%
% input: A original laodings
%       Y target
%       start=0: rational: orthogonal procr; 1 =random 2= current
%       conv=convergenc criterion
%       maxit=max number of iterations
% output: AT rotated A
%         T output rotation matrix
%         f loss function value
%
%   uses procr.m ssq.m nrm.m

[m,r]=size(A);
if start==0
    [T,AT]=procr(A,Y);
end;
if start==1
    T=(inv(nrm( randn(r,r) )))';
end;
AT=A*T;
Phi=inv(T)*inv(T)';


% compute function value

f=ssq(AT-Y);
%fprintf(' function value at start is %12.8f \n',f);
iter=0;
fstart=f;
fold=f+2*conv*f;
while abs(fold-f)>conv*fold & iter<maxit & f>conv*0.01
  iter=iter+1;
  fold=f;
  S=inv(T');
  % cycling through alpha'
  f1=fold+1;
  alfa=2;
  %Gq=2*(AT-Y);
  %TempValue = (AT'*Gq*T')';
  while fold<f1
    alfa=alfa/2;
    Gq=2*(AT-Y);
    Su=nrm(S+alfa*(AT'*Gq*T')');
    %Su=nrm(S+alfa*TempValue);
    f1=ssq(A*inv(Su')-Y);
  end;
  T=inv(Su');
  AT=A*T;
  f=ssq(AT-Y);
 % fprintf(' function value after %g iterations is %12.8f \n',iter,f1);
  if iter==maxit,disp(' NOTE NOTE: MAXIMUM NUMBER OF ITERATIONS USED; NOT
CONVERGED YET!!');end;

end
AT=A*T;
end
```

Ssq.m

```
function t = ssq(a)
%SSQ    SSQ(A) is the sum of squares of the elements of matrix A.
%t = sum(sum(a.^2));
t=a(:)'*a(:);
end
```

Procr.m

```
function [t,yhat]=procr(x,y)
% [T,Yhat]=Procr(X,Y) finds Yhat=XT that approximates Y; T orthonormal
[p,d,q]=svd(x'*y,0);
t=p*q';
yhat=x*t;
end
```

Cent.m

```
function Ac=cent(A)
% gives columnwise centered version of matrix A
Ac=A-ones(length(A(:,1)),1)*mean(A);
end
```

Phi.m

```
function p = phi(a,b)
% phi makes matrix of phi coefficients among columns of A and B
% columnvise congruence
p = inv(diag(diag(a'*a))^.5)*a'*b*inv(diag(diag(b'*b))^.5);
end
```

Nrm.m

```
function [N,d]=nrm(A)
% makes columnwise normalized version of A
[n,m]=size(A);
d=(sum(A.^2)).^.5;
N=A./(ones(n,1)*d);
end
```

References

Abdi, H. (2010). Congruence: Congruence coefficient, RV coefficient, and mantel coefficient. In N. J. Salkind (Ed.), *Encyclopedia of research design* (pp. 222–229). New York, NY: Sage.

Abdi, H., & Williams, L. J. (2010). Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, *2*, 433–459.

Asparouhov, T., & Muthén, B. (2014). Multiple-group factor analysis alignment. *Structural Equation Modeling*, *21*, 495–508. https://doi.org/10.1080/10705511.2014.919210

Borsboom, D., Mellenbergh, G. J., & Van Heerden, J. (2003). The theoretical status of latent variables. *Psychological Review*, *110*, 203–219.

Ceulemans, E., & Kiers, H. A. (2006). Selecting among three-mode principal component models of different types and complexities: A numerical convex hull based method. *British Journal of Mathematical and Statistical Psychology*, *59*, 133–150. https://doi.org/10.1348/000711005X64817

Ceulemans, E., Wilderjans, T. F., Kiers, H. A., & Timmerman, M. E. (2016). MultiLevel simultaneous component analysis: A computational shortcut and software package. *Behavior Research Methods*, *48*, 1008–1020. https://doi.org/10.3758/s13428-015-0626-8

Chan, W., Ho, R. M., Leung, K., Chan, D. K.-S., & Yung, Y.-F. (1999). An alternative method for evaluating congruence coefficients with Procrustes rotation: A bootstrap procedure. *Psychological Methods*, *4*, 378–402. https://doi.org/10.1037/1082-989X.4.4.378

De Leersnyder, J., Mesquita, B., & Kim, H. S. (2011). Where do my emotions belong? A study of immigrants' emotional acculturation. *Personality and Social Psychology Bulletin*, *37*, 451–463.

De Roover, K., Ceulemans, E., & Timmerman, M. E. (2012). How to perform multiblock component analysis in practice. *Behavior Research Methods*, *44*, 41–56. https://doi.org/10.3758/s13428-011-0129-1

De Roover, K., Ceulemans, E., Timmerman, M. E., Nezlek, J. B., & Onghena, P. (2013). Modeling differences in the dimensionality of multiblock data by means of clusterwise simultaneous component analysis. *Psychometrika*, *78*, 648–668.

De Roover, K., Ceulemans, E., Timmerman, M. E., & Onghena, P. (2013). A clusterwise simultaneous component method for capturing within-cluster differences in component variances and correlations. *British Journal of Mathematical and Statistical Psychology*, *66*, 81–102. https://doi.org/10.1111/j.2044-8317.2012.02040.x

De Roover, K., Ceulemans, E., Timmerman, M. E., Vansteelandt, K., Stouten, J., & Onghena, P. (2012). Clusterwise simultaneous component analysis for analyzing structural differences in multivariate multiblock data. *Psychological Methods*, *17*, 100–119.

De Roover, K., Timmerman, M. E., & Ceulemans, E. (2017). How to detect which variables are causing differences in component structure among different groups. *Behavior Research Methods*, *49*, 216–229. https://doi.org/10.3758/s13428-015-0687-8

De Roover, K., Timmerman, M. E., De Leersnyder, J., Mesquita, B., & Ceulemans, E. (2014). What's hampering measurement invariance: Detecting non-invariant items using clusterwise simultaneous component analysis. *Frontiers in Psychology*, *5*, 604. https://doi.org/10.3389/fpsyg.2014.00604

De Roover, K., Timmerman, M. E., Van Diest, I., Onghena, P., & Ceulemans, E. (2014). Switching principal component analysis for modeling means and covariance changes over time. *Psychological Methods*, *19*, 113–132.

De Roover, K., Vermunt, J. K., Timmerman, M. E., & Ceulemans, E. (2017). Mixture simultaneous factor analysis for capturing differences in latent variables between higher level units of multilevel data. *Structural Equation Modeling*, *24*, 506–523.

Diener, E., Kim-Prieto, C., Scollon, C., et al. (2001). [International Collage Survey 2001]. Unpublished raw data.

Esser, F., & Hanitzsch, T. (Eds.). (2013). *The handbook of comparative communication research*. Basingstoke, UK: Routledge.

Hurley, A. E., Scandura, T. A., Schriesheim, C. A., Brannick, M. T., Seers, A., Vandenberg, R. J., & Williams, L. J. (1997). Exploratory and confirmatory factor analysis: Guidelines, issues, and alternatives. *Journal of Organizational Behavior*, 667–683.

Jolliffe, I. T. (2002). *Principal component analysis*. New York, NY: Springer.

Jolliffe, I. T. (2005). Principal component analysis. In B. S. Everitt & D. C. Howell (Eds.), *Encyclopedia of statistics in behavioral science*. Chichester, UK: Wiley. https://doi.org/10.1002/0470013192.bsa501

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

Kiers, H. A. L., & ten Berge, J. M. F. (1994). Hierarchical relations between methods for simultaneous component analysis and a technique for rotation to a simple simultaneous structure. *British Journal of Mathematical and Statistical Psychology*, *47*, 109–126. https://doi.org/10.1111/j.2044-8317.1994.tb01027.x

Kuppens, P., Ceulemans, E., Timmerman, M. E., Diener, E., & Kim-Prieto, C. H. (2006). Universal intracultural and intercultural dimensions of the recalled frequency of emotional experience. *Journal of Cross-Cultural Psychology*, *37*, 491–515.

Lorenzo-Seva, U., & ten Berge, J. M. (2006). Tucker's congruence coefficient as a meaningful index of factor similarity. *Methodology: European Journal of Research Methods for the Behavioral and Social Sciences*, *2*, 57–64. https://doi.org/10.1027/1614-2241.2.2.57

McLachlan, G. J., & Peel, D. (2000). Mixtures of factor analyzers. In P. Langley (Ed.), *Proceedings of the Seventeenth International Conference on Machine Learning* (pp. 599–606). San Francisco, CA: Morgan Kaufmann.

Meredith, W. (1993). Measurement invariance, factor analysis and factorial invariance. *Psychometrika*, *58*, 525–543.

Meredith, W., & Teresi, J. A. (2006). An essay on measurement and factorial invariance. *Medical Care*, *44*, S69–S77.

Ogasawara, H. (2000). Some relationships between factors and components. *Psychometrika*, *65*, 167–185.

Schmitt, T. A. (2011). Current methodological considerations in exploratory and confirmatory factor analysis. *Journal of Psychoeducational Assessment*, *29*, 304–321.

Sočan, G. (2016). Comparison of principal component solutions in two populations: A bootstrap test of the perfect congruence hypothesis. *Methodology: European Journal of Research Methods for the Behavioral and Social Sciences*, *12*, 11–20. https://doi.org/10.1027/1614-2241/a000099

Timmerman, M. E., Ceulemans, E., De Roover, K., & Van Leeuwen, K. (2013). Subspace K-means clustering. *Behavior Research Methods*, *45*, 1011–1023. https://doi.org/10.3758/s13428-013-0329-y

Timmerman, M. E., & Kiers, H. A. (2003). Four simultaneous component models for the analysis of multivariate time series from more than one subject to model intraindividual and interindividual differences. *Psychometrika*, *68*, 105–121.

Triandis, H. (1988). Collectivism v. individualism: A reconceptualisation of a basic concept in cross-cultural social psychology. In *Cross-cultural studies of personality, attitudes and cognition* (pp. 60–95). London, UK: Palgrave Macmillan.

Tucker, L. R. (1951). *A method for synthesis of factor analysis studies*. Princeton, NJ: Educational Testing Service.

Tukey, J. W. (1980). We need both exploratory and confirmatory. *American Statistician*, *34*, 23–25.

Van Deun, K., Wilderjans, T. F., Van den Berg, R. A., Antoniadis, A., & Van Mechelen, I. (2011). A flexible framework for sparse

simultaneous component based data integration. *BMC Bioinformatics*, *12*, 448. https://doi.org/10.1186/1471-2105-12-448

Velicer, W. F., Peacock, A. C., & Jackson, D. N. (1982). A comparison of component and factor patterns: A Monte Carlo approach. *Multivariate Behavioral Research*, *17*, 371–388.

Wilderjans, T. F., Ceulemans, E., & Meers, K. (2013). CHull: A generic convex-hull-based model selection method. *Behavior Research Methods*, *45*, 1–15. https://doi.org/10.3758/s13428-012-0238-5

Yong, A. G., & Pearce, S. (2013). A beginner's guide to factor analysis: Focusing on exploratory factor analysis. *Tutorial in Quantitative Methods for Psychology*, *9*, 79–94.

Yung, Y. F. (1997). Finite mixtures in confirmatory factor-analysis models. *Psychometrika*, *62*, 297–330.