Online Supplementary Material

Section 1: Algorithms

Algorithm 1: CSSCA

Input: the concatenated data matrix **X** ($N \times J$), the number of clusters K, and the number of starts Q, the number of sparseness-induced zeros Z (in the loading matrix), and the position vector W_{dis} that indicates the positions of the distinctiveness-induced zeros.

Initialize the converge rate ε, the maximal number of iterations allowed $Iter_{max}$ and the minimal total loss $Loss_{global}$ (initially $Loss_{global}$ is set to an arbitrary large number)

Scale the data matrix **X**, such that the mean and the variance of each variable equals 0 and 1, respectively

FOR each start q (q = 1,2,…,Q)**, DO**

  Initialize the semi-random starting partition H_q, with the i^{th} element H_q[i] reprsenting the cluster index of observation i (see Section 2 for details)

  Initialize the total loss of the current start $Loss_q$ to an arbitrary large number, the total loss calculated at a specific iteration $Loss_{current}$ to 0, and the number of iterations that have been executed $Iter_q = 0$

  WHILE (($Loss_q - Loss_{current}$) > ε and ($Iter_q < Iter_{max}$)), **DO**

    Update $Iter_q = Iter_q + 1$

    IF the first iteration, **DO**

      FOR each cluster k, **DO**

        Estimate the Sparse DISCO-SCA solution (see Algorithm 2) with multiple random starts and obtain the cluster-specific loss $Loss_{q\_k}$, the cluster-specific score matrix $\mathbf{T}_{q\_k}$ and loading matrix $\mathbf{P}_{q\_k}$.

      END FOR

      Calculate the total loss value $Loss_{current} = \sum_k Loss_{q\_k}$

    END IF

    Update $Loss_q = Loss_{current}$

    FOR each observation i in the concatenated data matrix **X, DO**

      Obtain the current cluster assignment of i: $c = H_q$[i]

      FOR each cluster k, **DO**

        IF k equals c, **DO**

          Estimate the Sparse DISCO-SCA solution for projected cluster s that removes the observation i. A single user-specified start $\mathbf{P}_{q\_k}$ is used (see Algorithm 2). Obtain $Loss_{q\_s}$, $\mathbf{T}_{q\_s}$, and $\mathbf{P}_{q\_s}$.

          When the observation i stays at cluster k, the total loss will not change $Loss_{current\_k} = Loss_{current}$

        END IF

        IF k does not equal c, **DO**

          Estimate the Sparse DISCO-SCA solution for projected cluster b that adds in the observation i. A single user-specified start $\mathbf{P}_{q\_k}$ is used. Obtain $Loss_{q\_b}$, $\mathbf{T}_{q\_b}$, and $\mathbf{P}_{q\_b}$.

          Compute the total loss $Loss_{current\_k}$ if the observation i is moved from cluster c to cluster k in the iteration: $Loss_{current\_k} = Loss_{current} - Loss_{q\_k} - Loss_{q\_c} + Loss_{q\_s} + Loss_{q\_b}$

        END IF

      END FOR

      Assign the i Cluster b that minimizes $Loss_{current\_k}$: H_q[i] = b where $Loss_{current\_b} = \min_k Loss_{current\_k}$

      Update the total loss $Loss_{current} = Loss_{current\_b}$, and the score and loading matrices of the newly formulated clusters (i.e. s and b)

    END FOR

  END WHILE

  If $Loss_{current} < Loss_{global}$, **DO**

    Update $Loss_{global} = Loss_{current}$, and update the corresponding cluster partitions, score and loading matrices

  END IF

END FOR

Algorithm 2: Sparse DISCO-SCA

Input: the concatenated data matrix of a specific cluster $\mathbf{X}$ ($N_k \times J$), the number of starts Q, the number of sparseness-induced zeros Z (in the loading matrix), and the position vector W_{dis}. When the loading matrix is expected to be specified by the user, the starting loading matrix $\mathbf{P}_{rational}$ is also required

Initialize converge rate ε, the maximal number of iterations allowed $Iter_{max}$ and the minimal loss $Loss_{global}$ (initially $Loss_{global}$ is set to an arbitrary large number)

Column-wise mean-center the data matrix $\mathbf{X}$

IF loading matrix is expected to be specified by the user, **DO**
  $Q = 1$, $\mathbf{P}_q = \mathbf{P}_{rational}$
END IF
FOR each start q ($q = 1,2,\ldots,Q$), **DO**
  Initialize the loss of the current start $Loss_q$ to an arbitrary large number, the loss calculated at a specific iteration $Loss_{current}$ to 0, and the number of iterations that have been executed $Iter_q = 0$
  IF loading matrix is expected to be randomly generated, **DO**
    Initialize loading matrix $\mathbf{P}_q$**:** each entry p_{kr} is generated from a uniform distribution U [-1, 1]
  END IF
  WHILE (($Loss_q - Loss_{current}$) > ε and ($Iter_q < Iter_{max}$)), **DO**
    Update $Iter_q = Iter_q + 1$
    IF not the first iteration, **DO**
      Update $Loss_q = Loss_{current}$
    END IF
    Perform singular value decomposition $\mathbf{P_q}^T\mathbf{X}^T = \mathbf{U\Sigma V}^T$
    Update the score matrix $\mathbf{T}_q$ $\mathbf{T_q} = \mathbf{VU}^T$
    Update the loading matrix $\mathbf{P}_q$ $\mathbf{P_q} = \mathbf{X}^T\mathbf{T_q}$
    Impose the distinctiveness-induced zeros on $\mathbf{P}_q$ based on W_{dis}
    Impose the sparseness-induced zeros on $\mathbf{P}_q$: the smallest Z loadings of $\mathbf{P}_q$ (except for the distinctive-induced zero loadings) are imposed to be zero.
    Calculate and Update the loss $Loss_{current} = ||\mathbf{X} - \mathbf{T_q}\mathbf{P_q}^T||_2^2$
  END WHILE
  IF $Loss_{current} < Loss_{global}$, **DO**
    Update $Loss_{global} = Loss_{current}$, and update the corresponding score and loading matrices
  END IF
END FOR

Section 2: Technical Minutiae of Algorithm 1

1. **The starting partitions of the algorithm.** As explained in the main text, to reduce the probability of *ending in a local minimum* (instead of a global one), we utilize a multi-start procedure in the algorithm. The multiple starting partitions are created on the basis of the partitioning results of the two other clustering methods: Clusterwise PCA and iCluster.

   Clusterwise PCA is in principle a simplified version of CSSCA that is applied on the concatenated dataset. The major difference between Clusterwise PCA and CSSCA is that the former estimates non-sparse loading matrices and does not distinguish between common components and distinctive components. We have noticed that similar approaches have been proposed in statistics (for example McWilliams, & Montana, 2014), despite notable differences in estimation procedures. Because of its model configuration, we expect the resulting cluster recoveries of Clusterwise PCA to be fairly similar to the true clusters when the component structure accounts for a large proportion of the total variance (e.g., b = 10%). The algorithm of Clusterwise PCA is also implemented in the package ClusterSSCA.

   On the other hand, as argued in the main text, iCluster partitions the observations mainly based on the mean structure. Therefore, we expect the resulting cluster recoveries of iCluster to be similar to the true clusters when the proportion of mean-level differences is large (e.g., b = 90%). The algorithm of *iCluster* is provided in the R package iCluster (Shen et al., 2008, 2012).

   To ensure that CSSCA performs well at different levels of b, we generate the starting partitions of CSSCA based on the results of both Clusterwise PCA and iCluster. More specifically, two of the starts (which are called "user-specified starts") are cluster partitions produced by Clusterwise PCA (which is labelled H_c; we further name its resulting partition H_{cr}) and iCluster (which is labelled H_i; we further name its resulting partition H_{ir}). The other starts (which are called "semi random starts") are generated by randomly changing the cluster memberships of a certain amount of observations in H_c or H_i. When the similarity between H_c and H_{cr} is larger than the similarity between H_i and H_{ir}, the component structure is probably more important, therefore, H_c is used to generate the semi random starts; otherwise, H_c is used to generate the semi random starts.

2. **Restrictions on the model parameters.** Some restrictions concerning model parameters apply to CSSCA. First, during the iterations, the number of observations of *every cluster* should always be larger than the number of components. If this condition is not met, the CSSCA analysis with the package ClusterSSCA will automatically cease estimation following the current starting partition, and the total loss associated with the current starting partition will be set at an invalid value of 1e9. We should also note that the failure to meet the restriction indicates that the number of clusters is potentially over-estimated.

Section 3: Data Generation Procedure

A partition matrix **H** with size $N \times K$ was first generated (note that this is different from the partition indictor *vector* H as described in the algorithm, though **H** and H are easily transferable), to represent the true cluster partitions. This was a binary matrix with the "1"s indicating cluster memberships. For example, if subject n belonged to cluster k, then the entry in row n and column k was equal to one while the other entries in that row n equaled zero, since CSSCA restricts each subject to only belong to a cluster. As a result, each row of **H** contained one and only one non-zero entry.

Equation (4) in the main text expresses the observed data matrix $\mathbf{X}^{\mathbf{con}}$ as the addition of three parts: the component structure part $\mathbf{X}_{\mathbf{comp}}$, the mean structure part $\mathbf{X}_{\mathbf{mean}}$, and the noise part **E**. In the simulation, the average variance of the variables was equal to 1, among which e was attributable to **E** and a total of $1-e$ to $\mathbf{X}_{\mathbf{comp}}$ and $\mathbf{X}_{\mathbf{mean}}$. Subsequently, a fraction b of the remaining variance was further attributed to the mean structure and $1-b$ to the component structure (note that as the average variance of all variables was equal to one, the average variance attributable to the component structure was $(1-e)(1-b)$). In what follows, the data generation procedures for the three parts are detailed.

To construct the component structure part, for each cluster k, the *component score matrix* (with dimension $N_k \times R$) was generated as follows: (1) each entry was initially sampled from the univariate standard normal distribution, (2) the resulting matrix was column-wise mean-centered and (3) to ensure that the component scores were orthonormal, the Gram-Schmidt orthonormalization was applied to each score matrix. To set the variance – rather than the sum-of-squares – of each component equal to 1, we then multiplied each entry of the score matrices by the square root of the corresponding cluster size.

We then constructed the *component loading matrices* (with dimension $J \times R$), where we first generated a component loading matrix for each cluster, and then imposed distinctiveness-induced zeros and sparseness-induced zeros, as follows.

1. A different procedure is used to create the non-sparse version of the loading matrices in the high-congruence and low-congruence conditions.
    - For the low-congruence condition, each element in the cluster-specific loading matrices was

obtained initially by uniformly sampling from the range of −1 to 1. Subsequently, the resulting matrix was rescaled such that the sum-of-squares of each row equaled 1.

- For the high-congruence condition, in addition to the cluster-specific matrices, generated as described above, a common base matrix was also generated. The entries of the common base matrices were also uniformly sampled from the range of −1 to 1. Afterward, these matrices were re-scaled such that the sum-of-squares of each row equaled 0.7 for the common base matrix and 0.3 for the cluster-specific matrices. The final cluster-specific loading matrices were then obtained by adding the common base matrix to the cluster-specific matrices.

2. The *distinctiveness-induced zero loadings* were introduced to the cluster-specific loading matrices, in order to structure the distinctive components, as shown in Figure 2. More specifically, for the l^{th} (l = 1, 2) distinctive component, the loadings of the variables that did not belong to the l^{th} data block were set to zero.
3. Then, *the sparseness-induced zero loadings* were also imposed on the cluster-specific loading matrices. The number of the sparseness-induced zero loadings Z is jointly determined by the level of sparsity S, the block-specific number of variables and components. In the current simulation, Z equaled $3 \times J \times S$. We selected Z of the remaining non-zero entries in each cluster-specific loading matrices – after imposing the distinctive-induced zero loadings in the previous step – and imposed these entries to zero. For the low-congruence condition, the Z positions in each cluster-specific loading matrix were selected randomly. For the high congruence condition, about $70\% \times Z$ (or more concretely, the largest positive integer that is smaller than $70\% \times Z$) zero positions were random selected and were identical across all clusters while the remaining zero positions were selected randomly for each cluster.
4. Finally, we re-scaled the cluster-specific loading matrices such that the average sum-of-squares of each row equaled $(1 - b)$ $(1 - e)$.

For each cluster, the component structure part of the data was constructed by multiplying its score matrix and the transpose of its loading matrix. $\mathbf{X}_{\mathbf{comp}}$ was then created by stacking together vertically the cluster-specific component-structure part according to the cluster assignment of each observation.

To quantify the degrees of similarities between the resulting cluster-specific loading matrices, we computed Tucker's congruence coefficient φ for each pair of the corresponding components and averaged them across all components and clusters. Formally, φ between two vectors **x** and **y** is defined as their normalized inner product: $\varphi = \frac{x'y}{\sqrt{x'x}\sqrt{y'y}}$ (Tucker, 1951). In the simulated datasets, the average congruence coefficients equaled 0.18 (*SD* = 0.07) in the low congruence conditions and 0.53 (*SD* = 0.01) in the high congruence conditions.

As the first step in the creation of $\mathbf{X}_{\mathbf{mean}}$, the $K \times J$ cluster centroids matrix **M** was created where each row k (k = 1, 2, …, K) represented the centroids of cluster k. Each entry in **M** was randomly sampled from the univariate uniform distribution **U** (-1, 1). We then created a preliminary version of the mean structure $\mathbf{X}_{\mathbf{mean\_pre}}$ by multiplying **H** and **M**. Subsequently, we re-scaled each column of $\mathbf{X}_{\mathbf{mean\_pre}}$ such that the variance equaled b * (1 - e) in the resulting mean-structure part $\mathbf{X}_{\mathbf{mean}}$.

Last, the each entry of the error matrix **E** was randomly sampled from a univariate normal distribution N (0, $\sqrt{e}$)

The final concatenated data was constructed by summing $\mathbf{X}_{\mathbf{comp}}$, $\mathbf{X}_{\mathbf{mean}}$ and **E.**

Section 4: Supplementary Report on the Clustering Accuracy of CSSCA

In addition to the results reported in the section Simulation Studies of the original article, we report hereafter the average clustering accuracy of CSSCA as a function of the other six factors. We found that, on average, CSSCA resulted in better clustering accuracy when (1) the total number of variables J was larger (ARI = 1 when $J = 65$ and ARI = 0.99 when $J = 30$), (2) the number of clusters K was larger (ARI = 0.997 when $K = 4$ compared to ARI = 0.991 when $K = 2$), (3) the cluster size N_k was larger (ARI = 0.996 when the largest cluster includes 100 observations, and ARI = 0.992 when the largest cluster includes 50 observations), (4) the cluster sizes were identical across all clusters (ARI = 0.995 when all clusters have the equal number of observations compared to ARI = 0.993 when all clusters have unequal number of observations), (5) the congruence between the cluster-specific loading matrices φ was lower (ARI = 0.997 when φ = about 0.2 , and ARI = .992 when φ = about 0.55), and (6) the level of sparsity S was larger (ARI = 0.995 when $S = 0.5$ or 0.7, and ARI = 0.985 when $S = 0.3$),