Arteaga Moreno, FJ.; Ferrer, A. (2103). Building covariance matrices with the desired structure. Chemometrics and Intelligent Laboratory Systems. 127:80-88. doi:10.1016/j.chemolab.2013.06.003.

Building covariance matrices with the desired structure

Francisco Arteaga[a*] and Alberto Ferrer[b]

a. Universidad Católica de Valencia San Vicente Mártir. Departamento de Contabilidad, Finanzas y Control de Gestión. C/ Jorge Juan 18, 46004, Valencia, Spain (francisco.arteaga@ucv.es)

b. Universitat Politècnica de València. Departamento de Estadística e Investigación Operativa Aplicadas y Calidad. Camino de Vera s/n, Edificio 7A, 46022, Valencia, Spain (aferrer@eio.upv.es)

* Corresponding author.

Keywords

Covariances; simulation

Abstract

The problem of building a covariance matrix by fixing their diagonal values (variances) and all or a subset of its eigenvalues has been solved in different ways in the literature. In this paper we propose an iterative heuristic method to build such covariance matrix when a subset of its covariances is also chosen from the user. This provides a more flexible approach than those available in the literature for designing covariances matrices with the desired structure. As in other related approaches, the proposal can be useful for testing the performance of chemometric methods with data sets matching the theoretical conditions for their applicability or checking their robustness when the hypothesized properties fail.

1. Introduction

In a related paper, Arteaga and Ferrer [1] propose a singular value decomposition (SVD) based method with two approaches to simulate N by K multivariate normal data sets with a desired correlation structure, the *approach* 1 and *approach* 2, respectively. In *approach* 1 the user specifies the desired covariance matrix (that can be singular), yielding a data set with a sample covariance matrix exactly matching the specified by the user. This can be seen as an alternative to the popular Cholesky decomposition approach [2]. In *approach* 2 the user specifies the correlation structure by fixing a subset of the eigenvalues of the covariance matrix and the variances for the variables of the data set, yielding a column-wise centred data set. This resultant data set verifies that the sample covariance matrix has the desired variances and eigenvalues. An iterative heuristic method that uses an initial random matrix data set $\mathbf{X}$ is used. The role of matrix $\mathbf{X}$ is to assure a feasible covariance matrix $\mathbf{S} = \mathbf{X}^{\mathrm{T}}\mathbf{X}/(N-1)$.

The problem of building a correlation matrix with specified eigenvalues has a popular solution: the Bendel and Mickey algorithm [3,4] (Bendel and Mickey, 1978; Lin and Bendel, 1985). This algorithm takes a matrix having the specified eigenvalues and uses a finite sequence of rotations to introduce 1's on the diagonal.

In this paper we propose a new iterative algorithm to build covariance matrices not only by fixing their variances and eigenvalues (as in Arteaga & Ferrer's *approach* 2 [1] but also a subset of their covariances as in the so-called *completion problem* [5]. The method also handles null eigenvalues, yielding singular covariance matrices. This increases the flexibility of the user when designing covariance matrices. The method is deterministic if the user defines a symmetric seed matrix $\mathbf{S}$ and the complete set of the eigenvalues; otherwise, if the user employs a randomly generated symmetric seed matrix, or some of the eigenvalues remain unfixed, the method becomes stochastic.

The paper is organized as follows: Section 2 introduces the proposed algorithm; in Section 3 its performance is illustrated with several examples; in section 4 some conclusions are drawn. The commented MatLab code for the algorithm is detailed in the Appendix.

2. The algorithm

Let $\{v_j\}_{j=1,\ldots,K}$ be the pre-specified variances for the K variables, $\{c_{ij}\}$ the fixed subset of covariances and $\{\lambda_a\}_{a=1,\ldots,A}$ the desired subset of A eigenvalues (note that some of them can be zeros), with $A \leq K$, and $\sum_{a=1}^{A} \lambda_a \leq \sum_{j=1}^{K} v_j$. The remaining $K - A$ eigenvalues can be generated as random non-negative values that sum $\sum_{j=1}^{K} v_j - \sum_{a=1}^{A} \lambda_a$ (in this case the algorithm becomes stochastic). Note that if $K \geq N$, no more than $N - 1$ non-null eigenvalues should be specified. It must be noted also that, when we combine the fixed eigenvalues with those randomly generated, the K eigenvalues must be sorted in descending order, and the position for each non-null fixed eigenvalue must be recorded for tracking purposes, because the convergence is attained in terms of the fixed covariances and the fixed eigenvalues.

Let **L** be a diagonal K by K matrix with diagonal values equal to the eigenvalues, in descending order, i.e. $l_{a,a} = \lambda_a$, and let r be the desired rank for **S**. The proposed algorithm is outlined in the following:

Step 0: define **S** as an arbitrary symmetric K by K seed matrix.

Step 1: define **V** as the first r eigenvectors of **S**.

Step 2: replace **S** by $\mathbf{S} = \mathbf{VLV}^{\mathrm{T}}$.

Step 3: scale **S** to have the desired variances.

Step 4: replace the $\{s_{ij}\}$ values of matrix **S** with the $\{c_{ij}\}$ desired covariances.

Step 5: if **S** has negative eigenvalues, replace them with their absolute value and scale **S** to have the desired variances.

Step 6: repeat steps 1 to 5 until convergence on the desired eigenvalues and covariances.

In step 0 **S** can be either specified by the user or randomly generated. Note that, if all eigenvalues are fixed, the resultant covariance matrix is completely determined by the initial **S**. This is not true if a subset of the eigenvalues remain unfixed because they are randomly generated.

In step 2 the eigenvectors of the current **S** matrix are combined with the desired eigenvalues. Let n_1 be the number of non-null fixed eigenvalues. If $n_1 < r$, the n_1 fixed eigenvalues replace the corresponding eigenvalues of the current **S** matrix (remember that their positions have been recorded); the remaining $r - n_1$ non-null eigenvalues of **S** are re-scaled to assure they sum up the part of the total variance not explained by the previously n_1 fixed non-null eigenvalues. In this step **S** is constrained to have the desired eigenvalues.

In step 3 the current covariance matrix is scaled by multiplying each of its $\{s_{ij}\}$ values with the weighting factor $\sqrt{v_i v_j / s_{ii} s_{jj}}$. This guaranties the desired variances for **S**, at the expense of changing the desired eigenvalues.

In step 4 **S** is constrained to have the desired covariances; consequently, the eigenvalues and the variances change.

The imputed covariances in Step 4 can lead us to an invalid covariance matrix, with negative eigenvalues. If this is the case, in Step 5 the negative eigenvalues are replaced by their absolute value, and **S** is scaled to have the desired variances.

The iterative nature of this approach allows that, at each iteration, if the algorithm converges, the differences between the desired and obtained eigenvalues and covariances are smaller (note that step 4 guaranties the variances). If this is the case, the matrix **S** in step 6 will converge to a matrix matching the desired variances, eigenvalues and covariances.

In our implementation, the algorithm converges when for each fixed covariance c_{ij}, the current **S** matrix satisfies $\left(c_{ij} - s_{ij}\right)^2 < 10^{-10}$, and for each fixed eigenvalue l_a there is an eigenvalue λ_a in **S** such that $(\lambda_a - l_a)^2 < 10^{-10}$. This tolerance threshold (10^{-10}) may be modified by the user.

Figure 1 illustrates this algorithm.

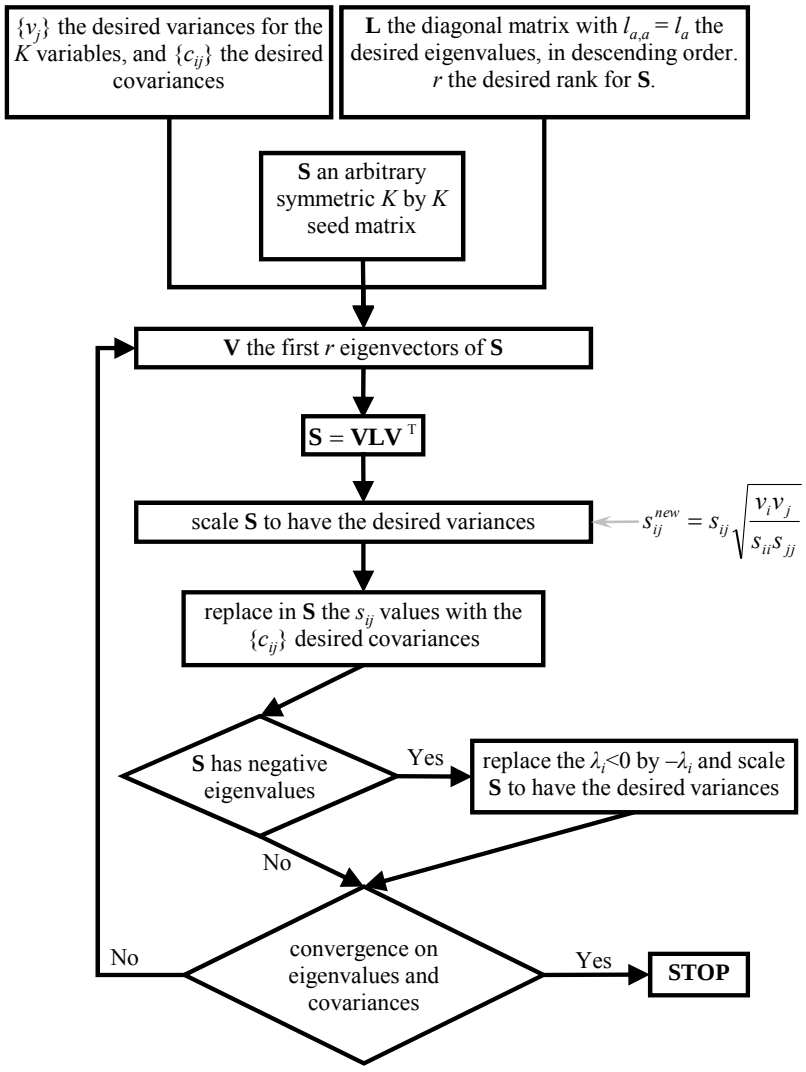

Figure 1. Schedule for the algorithm to build a covariance matrix with pre-specified variances, eigenvalues and a fixed subset of covariances.

For convergence, only the desired covariances and eigenvalues have to be surveyed. Note that for tracking the non-null eigenvalues at each iteration we need to know their positions along the *K* eigenvalues (the null fixed eigenvalues being the last ones).

This algorithm performs well for variances and a subset of the covariances, or for variances and a subset of the eigenvalues, but when the variances, a subset of the eigenvalues, and a subset of the covariances are fixed, convergence problems may arise.

It must be noted that some combinations of variances, covariances, and eigenvalues may be unfeasible, that is, there is no covariance matrix satisfying those restrictions, and the algorithm fails to converge. Kurowicka and Cooke [5] show that an incomplete covariance matrix can be filled out when all partial correlations that can be calculated from the variances and the known covariances of the incomplete covariance matrix belong to (-1, 1). The problem is that when some of the eigenvalues are constrained, Kurowicka and Cooke's method may not be useful to guarantee a feasible covariance matrix.

The MatLab code for this new algorithm (the *sbuild* function) is given in Appendix A.

3. Examples

In this section the algorithm's performance is illustrated by using some examples. The MatLab command is `S=sbuild(K,eig,var,co,S0)`, where `S` is the resultant covariance matrix, `K` is the number of variables, `eig` is a row vector with the desired eigenvalues (this can be [] when no eigenvalues are fixed), `var` is a row vector with the desired variances (if `var` is set to 0 all the variances are 1's), `co` provides the desired covariances (this can be []), `S0` is the seed covariance matrix (this can be []).

3.1. Two correlation matrices with the same specified structure

Two different 5 by 5 correlation matrices with rank 4, with fixed eigenvalues {2.5, 1.0} and with fixed covariances $\{s_{12} = 0.5, s_{13} = -0.5, s_{24} = 0.3, s_{35} = -0.7,\}$ are simulated. Both covariance matrices are shown in Table 1. For the first case the algorithm took 46 iterations to converge, while for the second case 29 iterations were needed.

The MatLab command for both covariance matrices is:

```
S=sbuild(5,[2.5 1 0],0,[1 2 .5;1 3 -.5;2 4 .3;3 5 -.7],[])
```

Table 1. Two different simulated covariance matrices with fixed variances, some fixed correlations and some fixed eigenvalues. Fixed values are in bold emphasis.

	X_1	X_2	X_3	X_4	X_5
X_1	**1.0000**	**0.5000**	**-0.5000**	0.2172	0.2249
X_2	**0.5000**	**1.0000**	-0.2456	**0.3000**	0.2729
X_3	**-0.5000**	-0.2456	**1.0000**	-0.6427	**-0.7000**
X_4	0.2172	**0.3000**	-0.6427	**1.0000**	0.0233
X_5	0.2249	0.2729	**-0.7000**	0.0233	**1.0000**
	λ_1	λ_2	λ_3	λ_4	λ_5
	2.5000	**1.0000**	0.9465	0.5535	**0.0000**

	X_1	X_2	X_3	X_4	X_5
X_1	**1.0000**	**0.5000**	**-0.5000**	-0.2122	0.0004
X_2	**0.5000**	**1.0000**	-0.6367	**0.3000**	0.5449
X_3	**-0.5000**	-0.6367	**1.0000**	0.0842	**-0.7000**
X_4	-0.2122	**0.3000**	0.0842	**1.0000**	-0.2858
X_5	0.0004	0.5449	**-0.7000**	-0.2858	**1.0000**
	λ_1	λ_2	λ_3	λ_4	λ_5
	2.5000	1.2281	**1.0000**	0.2719	**0.0000**

3.2. A covariance matrix that contains some pre-specified submatrices

A 9 by 9 correlation matrix with rank 6, with an eigenvalue equals to 1.0, that incorporates three 3 by 3 correlation matrices, $\mathbf{S}_1$, $\mathbf{S}_2$ and $\mathbf{S}_3$ as shown in Figure 2, is simulated.

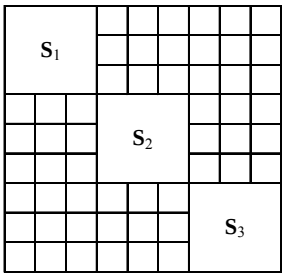

Figure 2. A 9 by 9 matrix that incorporates three 3 by 3 correlation matrices.

If $\mathbf{S}_1 = \begin{bmatrix} 1 & 0.5 & 0.4 \\ 0.5 & 1 & 0.7 \\ 0.4 & 0.7 & 1 \end{bmatrix}$, $\mathbf{S}_2 = \begin{bmatrix} 1 & 0.5 & -0.4 \\ 0.5 & 1 & -0.7 \\ -0.4 & -0.7 & 1 \end{bmatrix}$ and $\mathbf{S}_3 = \begin{bmatrix} 1 & -0.5 & 0.4 \\ -0.5 & 1 & -0.7 \\ 0.4 & -0.7 & 1 \end{bmatrix}$, `co=[1 2 .5;1 3 .4;2 3 .7;4 5 .5;4 6 -.4;5 6 -.7;7 8 -.5;7 9 .4;8 9 -.7]` and the MatLab command is `S=sbuild(9,[1 0 0 0],0,co,[])`, the resultant correlation matrix, obtained after 36 iterations, is shown in Table 2.

Table 2. A full rank 9 by 9 correlation matrix that incorporates three 3 by 3 correlation matrices. Fixed values are in bold emphasis.

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9
X_1	**1.0000**	**0.5000**	**0.4000**	0.0677	0.4031	0.0249	0.2568	-0.1873	-0.3047
X_2	**0.5000**	**1.0000**	**0.7000**	-0.3362	-0.0092	0.2583	-0.0470	-0.1841	0.0141
X_3	**0.4000**	**0.7000**	**1.0000**	-0.1222	-0.1409	-0.0770	0.1102	0.0845	0.1669
X_4	0.0677	-0.3362	-0.1222	**1.0000**	**0.5000**	**-0.4000**	0.8028	-0.0817	-0.0194
X_5	0.4031	-0.0092	-0.1409	**0.5000**	**1.0000**	**-0.7000**	0.3009	-0.3502	-0.1620
X_6	0.0249	0.2583	-0.0770	**-0.4000**	**-0.7000**	**1.0000**	-0.0466	-0.0819	0.0594
X_7	0.2568	-0.0470	0.1102	0.8028	0.3009	-0.0466	**1.0000**	**-0.5000**	**0.4000**
X_8	-0.1873	-0.1841	0.0845	-0.0817	-0.3502	-0.0819	**-0.5000**	**1.0000**	**-0.7000**
X_9	-0.3047	0.0141	0.1669	-0.0194	-0.1620	0.0594	**0.4000**	**-0.7000**	**1.0000**
	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	λ_8	λ_9
	2.6740	2.2173	1.8601	**1.0000**	0.9646	0.2840	**0.0000**	**0.0000**	**0.0000**

3.3. A correlation matrix with a highly restricted structure

In this example a full rank 5 by 5 correlation matrix, with fixed eigenvalues {2.0, 1.5} and with fixed covariances $\{s_{12} = s_{23} = s_{34} = s_{45} = s_{14} = s_{25} = 0.0\}$ is simulated. The resultant correlation matrix, obtained after 31 iterations, is shown in Table 3.

Table 3. A full rank 5 by 5 correlation matrix with fixed variances, some fixed correlations and some fixed eigenvalues. Fixed values are in bold emphasis.

	X_1	X_2	X_3	X_4	X_5
X_1	**1.0000**	**0.0000**	0.7813	**0.0000**	-0.5037
X_2	**0.0000**	**1.0000**	**0.0000**	0.5000	**0.0000**
X_3	0.7813	**0.0000**	**1.0000**	**0.0000**	-0.1457
X_4	**0.0000**	0.5000	**0.0000**	**1.0000**	**0.0000**
X_5	-0.5037	**0.0000**	-0.1457	**0.0000**	**1.0000**
	λ_1	λ_2	λ_3	λ_4	λ_5
	2.0000	**1.5000**	0.8678	0.5000	0.1322

3.4. Retrieving the raw data in a PLS Path Modelling application

In PLS Path Modeling applications [6,7] a data set $\mathbf{X}$ with N rows (objects) and K columns (variables named indicators) is arranged in B blocks $\mathbf{X}_1$, $\mathbf{X}_2$, …, $\mathbf{X}_B$, with K_1, K_2, …, K_B indicators, respectively, being $K = \sum_{b=1}^{B} K_b$.

Each one of the B blocks of indicators, $\mathbf{X}_b$, is a scale for measuring, in an indirect manner, a latent variable (i.e. a construct) named $\mathbf{t}_b$. For this example, let us assume that

each $\mathbf{X}_b$ can be expressed as $\mathbf{X}_b = \mathbf{t}_b \mathbf{p}_b^{\mathrm{T}} + \mathbf{E}_b$, that is, each one of the indicators of the b^{th} block is a reflect of their latent variable $\mathbf{t}_b$ (this is the so-called reflective way).

We assume that there are causal relationships among the constructs, reflected in the so-called structural model. In Figure 3 a simple example with three constructs with four, four and two indicators, respectively, is shown. Raw data are available for this example in a Gefen and Straub's work [8].

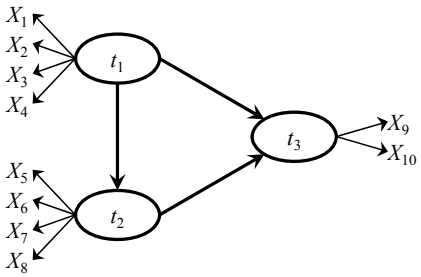

Figure 3. Structural model for the Gefen and Straub's example

The estimation of a PLS Path Model is determined by the K by K correlation matrix for the indicators $\mathbf{S} = \mathbf{X}^{\mathrm{T}}\mathbf{X}/(N-1)$, but this is not often published in literature. Therefore, potential readers cannot replicate the estimation nor fit alternative PLS path models [9]. However, researchers often publish: i) the between constructs correlations $\mathbf{Q} = \mathbf{T}^{\mathrm{T}}\mathbf{T}/(N-1)$, where $\mathbf{T} = [\mathbf{t}_1\ \mathbf{t}_2 \cdots\ \mathbf{t}_B]$, ii) the correlations between each construct and their indicators: the loadings $\mathbf{X}_b^{\mathrm{T}}\mathbf{t}_b/(N-1)$, and iii) the correlations between each indicator and the other constructs: the cross-loadings $\mathbf{X}_b^{\mathrm{T}}\mathbf{t}_c/(N-1)$, $b \neq c$. If we call $\mathbf{L}$ the matrix that groups the loadings and the cross-loadings, i.e. $\mathbf{L} = \mathbf{X}^{\mathrm{T}}\mathbf{T}/(N-1)$, we obtain the following correlation matrix:

$$\mathbf{R} = \frac{\begin{bmatrix}\mathbf{X}^{\mathrm{T}} \\ \mathbf{T}^{\mathrm{T}}\end{bmatrix}[\mathbf{X} \quad \mathbf{T}]}{N-1} = \begin{bmatrix}\mathbf{S} & \mathbf{L} \\ \mathbf{L}^{\mathrm{T}} & \mathbf{Q}\end{bmatrix}$$

This correlation matrix is incomplete because the correlations between the indicators, grouped in $\mathbf{S}$, are usually not provided in papers [9]. In the case of the Gefen and Straub's example, the structure of the correlation matrix $\mathbf{R}$ is shown in Figure 4.

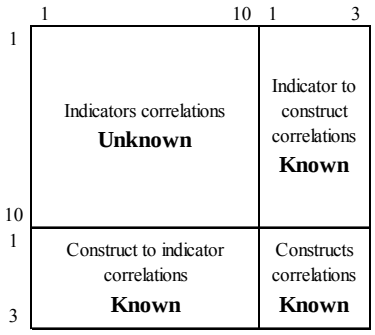

Figure 4. Correlation matrix (indicators plus constructs) for the Gefen and Straub's example.

As we have an incomplete correlation matrix, we can use the *sbuild* function to build a complete correlation matrix that matches the known correlations. Due to the strong dependence between the indicators and the constructs (each $\mathbf{X}_b$ block of variables associated to the $\mathbf{t}_b$ construct is essentially one-dimensional) and the fact that we know the correlations between the constructs (the $\mathbf{Q}$ matrix), it is expected that the estimated K by K indicators correlation submatrix, $\hat{\mathbf{S}}$, is an accurate approximation of the unknown correlation matrix $\mathbf{S}$. If we re-estimate the model, from $\hat{\mathbf{S}}$, and compare our

results with the published ones, we can measure how accurate our estimation is and, if this is the case, we can estimate alternative PLS path models as if we would have got the original data.

Tables 4, 5 and 6 show the **S** matrix for the Gefen and Straub's example, the loadings and cross-loadings matrix **L** after the estimation of the PLS path model from Figure 3, and the constructs correlation matrix, **Q**, respectively.

Table 4. Unpublished indicators correlation matrix **S** for the Gefen and Straub's example.

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
X_1	1	0.5160	0.6562	0.5503	0.3152	0.2586	0.3241	0.3688	0.1796	0.2510
X_2	0.5160	1	0.6377	0.5794	0.2809	0.2486	0.3671	0.4001	0.1883	0.2373
X_3	0.6562	0.6377	1	0.6438	0.1936	0.2014	0.3260	0.4226	0.1050	0.2491
X_4	0.5503	0.5794	0.6438	1	0.4214	0.2752	0.4571	0.4332	0.1288	0.2470
X_5	0.3152	0.2809	0.1936	0.4214	1	0.5911	0.6229	0.5426	0.2717	0.3235
X_6	0.2586	0.2486	0.2014	0.2752	0.5911	1	0.5565	0.5570	0.1854	0.1385
X_7	0.3241	0.3671	0.3260	0.4571	0.6229	0.5565	1	0.6771	0.3672	0.3467
X_8	0.3688	0.4001	0.4226	0.4332	0.5426	0.5570	0.6771	1	0.3540	0.4006
X_9	0.1796	0.1883	0.1050	0.1288	0.2717	0.1854	0.3672	0.3540	1	0.6343
X_{10}	0.2510	0.2373	0.2491	0.2470	0.3235	0.1385	0.3467	0.4006	0.6343	1

Table 5. Published **L** matrix with the correlations among the indicators and the constructs for the Gefen and Straub's example (loadings, in bold, and cross-loadings).

	t_1	t_2	t_3
X_1	**0.8085**	0.3873	0.2399
X_2	**0.8171**	0.4018	0.2366
X_3	**0.8677**	0.3619	0.1994
X_4	**0.8464**	0.4901	0.2108
X_5	0.3714	**0.8140**	0.3304
X_6	0.2974	**0.7603**	0.1780
X_7	0.4471	**0.8782**	0.3943
X_8	0.4875	**0.8641**	0.4185
X_9	0.1806	0.3699	**0.8945**
X_{10}	0.2948	0.3856	**0.9130**

Table 6. Published correlation matrix for the constructs, **Q**, for the Gefen and Straub's example.

	t_1	t_2	t_3
t_1	1	0.4968	0.2658
t_2	0.4968	1	0.4182
t_3	0.2658	0.4182	1

The MatLab command is now `R=sbuild(13,[],0,co,[])`, with `co=[1 11 .8085;1 12 .3873; … ;10 13 .9130;11 12 .4968;11 13 .2658;12 13 .4182]`. Table 7 shows the 10 by 10 estimated submatrix $\hat{\mathbf{S}}$ associated to the indicators, extracted from matrix **R**.

Table 7. Estimated indicators correlation matrix $\hat{\mathbf{S}}$ for the Gefen and Straub's example.

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
X_1	**1**	0.6533	0.5751	0.5661	0.2483	0.3208	0.2508	0.4145	0.3325	0.1481
X_2	0.6533	**1**	0.5627	0.5542	0.1901	0.2984	0.4238	0.3846	0.1488	0.3211
X_3	0.5751	0.5627	**1**	0.6903	0.3710	0.2512	0.3745	0.2819	0.0120	0.2423
X_4	0.5661	0.5542	0.6903	**1**	0.4270	0.1648	0.4001	0.5026	0.1831	0.2210
X_5	0.2483	0.1901	0.3710	0.4270	**1**	0.6119	0.6839	0.5324	0.2723	0.2181
X_6	0.3208	0.2984	0.2512	0.1648	0.6119	**1**	0.6087	0.4895	0.2093	0.1270
X_7	0.2508	0.4238	0.3745	0.4001	0.6839	0.6087	**1**	0.6606	0.2754	0.3850
X_8	0.4145	0.3846	0.2819	0.5026	0.5324	0.4895	0.6606	**1**	0.3758	0.4544
X_9	0.3325	0.1488	0.0120	0.1831	0.2723	0.2093	0.2754	0.3758	**1**	0.6687
X_{10}	0.1481	0.3211	0.2423	0.2210	0.2181	0.1270	0.3850	0.4544	0.6687	**1**

The difference between $\mathbf{S}$ and $\hat{\mathbf{S}}$ should not be measured element-wise but by comparing the resultant estimated models with $\mathbf{S}$ and $\hat{\mathbf{S}}$. In order to do so, we re-estimate the PLS path model from $\hat{\mathbf{S}}$, yielding a new constructs correlation matrix $\widehat{\mathbf{Q}}$, shown in Table 8.

Table 8. Correlation matrix for the constructs for the Gefen and Straub's example, from the estimated indicators correlation matrix $\hat{\mathbf{S}}$.

	t_1	t_2	t_3
t_1	1	0.4891	0.2651
t_2	0.4891	1	0.4101
t_3	0.2651	0.4110	1

By comparing Table 6 and Table 8, it is concluded that the between constructs correlations matrices $\mathbf{Q}$ and $\widehat{\mathbf{Q}}$ are quite similar.

The PLS path model estimation yields also the coefficients for the paths. In Table 9 we compare these path coefficients from both models.

Table 9. Estimated coefficients for the Gefen and Straub's example with both the original and the estimated indicators correlation matrix.

From	To	Original	Estimated
t_1	t_2	0.4968	0.4891
t_1	t_3	0.0771	0.0842
t_2	t_3	0.3799	0.3698

From Figure 3 we see that $\mathbf{t}_2$ and $\mathbf{t}_3$ are two endogenous constructs and in Table 10 we compare their R^2 from both models. Again we see that the model estimated from the estimated indicators correlation matrix agrees with the original model.

Table 10. Estimated R^2 coefficients for the endogenous constructs for the Gefen and Straub's example with both the original and the estimated indicators correlation matrix.

	Original	Estimated
t_2	24.68%	23.92%
t_3	17.93%	17.43%

The PLS path model estimated yields also other useful statistics (weights, communalities, reliability measures, …) that researchers often publish. As a suitable strategy, the *sbuild* function can be applied several times, say M, to achieve M estimated indicators correlation matrices, $\{\hat{\mathbf{S}}_m\}_{m=1}^{M}$, and choose among them the one that better matches with the previously mentioned published statistics.

3.5. Simulating multi-phase data

Processes with several stages or phases, i.e. processes with dynamics of different order and changes in the correlation structure among variables are quite common in practice. The multi-stage nature of both batch and continuous processes can also be the result of the different processing units and the distinguishable operations inside a unit. Even in the same unit or stage of a batch process, the correlation structure and process dynamics may change as the batch is being processed [10].

In this example a correlation matrix from a process with 3 phases (or stages) and 4 variables in each phase will be simulated. Let $\mathbf{S}_1$, $\mathbf{S}_2$ and $\mathbf{S}_3$ be the correlation matrices for each of the 3 phases. Each $\mathbf{S}_i$ is assumed to have an eigenvalue equals to 3 to assure a strong relationships between the 4 variables in each phase. In this case *sbuild* function is used to build a correlation matrix for the 12 variables resulting after arranging the 4 variables from each phase. The diagonal blocks $\mathbf{S}_1$, $\mathbf{S}_2$ and $\mathbf{S}_3$ are fixed in advance letting the algorithm to obtain only the correlations out of the diagonal blocks. Table 11

shows the fixed correlation blocks for the multi-phase example storage in variable `co=[1 2 -.634; 1 3 .890;5 6 -.739;…;7 8 -.381;9 10 .587;…;11 12 -.739]`.

Table 11. Fixed correlations for the multi-phase example.

1	-0.634	0.890	-0.776								
-0.634	1	-0.680	0.498								
0.890	-0.680	1	-0.488								
-0.776	0.498	-0.488	1								
				1	-0.739	-0.791	0.378				
				-0.739	1	0.831	-0.821				
				-0.791	0.831	1	-0.381				
				0.378	-0.821	-0.381	1				
								1	0.587	-0.962	0.760
								0.587	1	-0.440	0.427
								-0.962	-0.440	1	-0.739
								0.760	0.427	-0.739	1

This procedure is run M times and, by averaging the resulting complete M correlation matrices, the final desired **S** matrix is obtained. This is a correlation matrix (12×12) with 3 diagonal blocks $\mathbf{S}_i$ (4×4) matching the fixed values shown in Table 11 and with the values out of the diagonal blocks close to 0 (i.e. they are averages of random correlation values, ranging from -1 to 1). The largest the number M of simulated matrices the closest to 0 the out of diagonal blocks correlations.

Table 12 shows the outcome from M=10 by using the following MatLab command:

`S=zeros(12,12);for i=1:10,S=S+=sbuild(12,[],0,co,[]);end;S=S/10,` with variable `co` as specified before.

Table 12. Simulated multi-phase correlation matrix. Fixed values are in bold emphasis.

1	**-0.634**	**0.890**	**-0.776**	0.029	-0.008	-0.021	0.006	0.001	0.056	0.065	0.059
-0.634	**1**	**-0.680**	**0.498**	0.010	0.022	0.063	0.020	0.053	-0.068	-0.113	-0.005
0.890	**-0.680**	**1**	**-0.488**	0.025	0.049	0.007	-0.076	0.041	0.061	0.029	0.108
-0.776	**0.498**	**-0.488**	**1**	-0.074	0.061	0.094	-0.038	0.024	-0.066	-0.057	0.022
0.029	0.010	0.025	-0.074	**1**	**-0.739**	**-0.791**	**0.378**	0.018	0.003	0.041	-0.017
-0.008	0.022	0.049	0.061	**-0.739**	**1**	**0.831**	**-0.821**	-0.004	-0.009	-0.046	0.066
-0.021	0.063	0.007	0.094	**-0.791**	**0.831**	**1**	**-0.381**	-0.023	-0.020	-0.027	0.018
0.006	0.020	-0.076	-0.038	**0.378**	**-0.821**	**-0.381**	**1**	-0.020	-0.006	0.046	-0.085
0.001	0.053	0.041	0.024	0.018	-0.004	-0.023	-0.020	**1**	**0.587**	**-0.962**	**0.760**
0.056	-0.068	0.061	-0.066	0.003	-0.009	-0.020	-0.006	**0.587**	**1**	**-0.440**	**0.427**
0.065	-0.113	0.029	-0.057	0.041	-0.046	-0.027	0.046	**-0.962**	**-0.440**	**1**	**-0.739**
0.059	-0.005	0.108	0.022	-0.017	0.066	0.018	-0.085	**0.760**	**0.427**	**-0.739**	**1**

4. Evaluation study

Given the iterative nature of the method, this section provides an evaluation study of the proposal in three scenarios. Calculations were run on a 2.4 GHz, 4 Gb RAM, Intel Core i5 computer with a 64-bit operating system.

In the first scenario the performance of the algorithm was evaluated for a 5 by 5 correlation matrix, with rank equals to 4 and a fixed eigenvalue equals to 1, and fixed covariances $\{s_{12}, s_{13}, s_{14}, s_{24}, s_{25}, s_{45}\}$ randomly chosen from $(-1, 1)$. The MatLab command is `S=sbuild(5,[1 0],0,[1 2 s12;1 3 s13;1 4 s14;2 4 s24;2 5 s25;4 5 s45],[])`. The MatLab code for each s_{ij} is `rand()*(-1)^round(rand())`.

One hundred different sets of fixed covariances $\{s_{12}, s_{13}, s_{14}, s_{24}, s_{25}, s_{45}\}$ were simulated and, for each combination, the algorithm was run 100 times, and the

efficiency of the algorithm was evaluated by registering the convergence (or not), the number of iterations to converge and the computing time, in seconds. It should be noted that convergence failure may be due not only to a fail in the algorithm (bad initial), but also to the randomly chosen covariances and fixed eigenvalues that may lead to an unfeasible covariance matrix (See Section 2). This justifies the need for running 100 times the algorithm, for each combination of fixed covariances.

For 63 combinations of covariances the algorithm never converged (probably due to unfeasible covariance matrix). For 7 combinations of covariances the number of times that the algorithm converged was less than or equal to 75; for the 30 remaining combinations of covariances the algorithm converged more than 75 times and, among then, for 10 combinations the algorithm always converged. The average number of iterations needed in the runs that the algorithm converged was 198. The average time in the runs that the algorithm converged was 0.0366 seconds. The no convergence was verified after 5000 iterations, and the average time for the no convergent runs was 0.8844 seconds.

In the second scenario the convergence properties of the algorithm, avoiding the case when the randomly chosen covariances are jointly incompatibles, or incompatibles also with the fixed eigenvalues, were studied. The previous simulation was run, but ensuring compatibility between the fixed covariances and the fixed eigenvalue. To do this, using the Arteaga and Ferrer's *approach* 2 [1], we first simulated an initial (5×5) covariance matrix by fixing only an eigenvalue equals to one. From the simulated covariance matrix, the covariances $\{s_{12}, s_{13}, s_{14}, s_{24}, s_{25}, s_{45}\}$ were selected. Following this procedure, one hundred different sets of fixed covariances $\{s_{12}, s_{13}, s_{14}, s_{24}, s_{25}, s_{45}\}$ were simulated and, for each combination, the *sbuild* algorithm was run 100 times. Unlike in the first scenario now there is no case for which the algorithm never converges. In fact, in the worst case the algorithm converges 52 out of the 100 times. In 30 cases the algorithm always converges; in 60 cases the algorithm converges between 76 and 99 out of the 100 times; and only in 10 cases the algorithm converges between 50 and 75 out of the 100 times. Note that in this scenario, given that the feasibility of the covariance matrix is assured, convergence problems arise only due to bad initials. For this second scenario, the average number of iterations needed in the runs that the algorithm converges is 195. The average time in the runs that the algorithm converges is 0.0368 seconds. The no convergence is verified after 5000 iterations, and the average time for the no convergent runs is 0.8918 seconds.

In the third scenario the algorithm was evaluated for a 50 by 50 correlation matrix with five fixed eigenvalues equals to 10, 8, 6, 4 and 2, respectively, and by fixing 200 correlations. To avoid the above mentioned inter covariances and eigenvalues incompatibilities, we built an initial correlation matrix with the desired eigenvalues with the Arteaga and Ferrer's *approach* 2 [1] and we select randomly 200 correlations to be fixed. With this restriction we run the *sbuild* algorithm 10 times. This procedure (building the initial covariance matrix with approach 2 and running 10 times the *sbuild* algorithm) was repeated 100 times. In this case the algorithm always converged with a mean time of 0.4899 seconds and with a mean of 123 iterations.

To evaluate if the convergence is affected by the tolerance we simulated a 5 by 5 correlation matrix with two fixed eigenvalues (0 and 1) and fixed covariances $\{s_{12}, s_{13}, s_{14}, s_{24}, s_{25}, s_{45}\}$, in a similar manner as in the first scenario described before. The MatLab command is `S=sbuild(5,[1 0],0,co,[])`, where `co` defines the fixed covariances vector. Different tolerance values were studied.

The algorithm was run 500 times (i.e. 500 different initials) for each one of the tolerance values considered $\{10^{-3}, 10^{-4}, \ldots, 10^{-14}\}$. For each tolerance value, the efficiency of the algorithm was evaluated by registering the percentage of convergence and the average number of iterations to converge. Figures 5 and 6 illustrate the results:

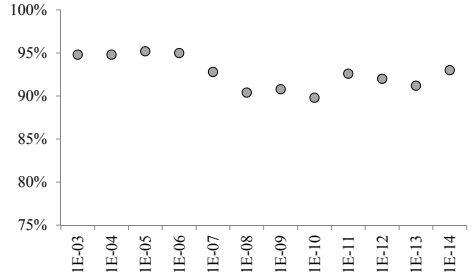

Figure 5. Convergence ratio vs tolerance

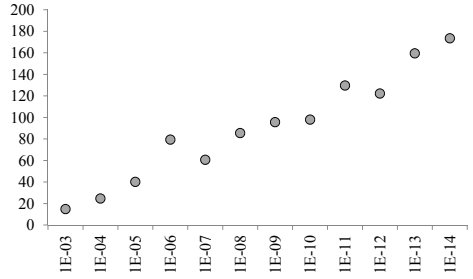

Figure 6. Average number of iterations needed in convergent runs

Figures 5 shows that convergence ratio is relatively robust to tolerance values in the studied range. On the other hand, Figure 6 shows that, as expected, the average number of iterations to converge increases as the tolerance decreases.

This results validate the tolerance value (10^{-10}) used in the paper. Anyway, as already commented, this threshold may be modified by the user.

5. Conclusions

In this work an iterative heuristic method to build a covariance matrix by fixing their variances, a subset of its covariances, and all or a subset of its eigenvalues is proposed. The new approach enhances Arteaga and Ferrer's *approach* 2 [1] in two important aspects:

a. It avoids using an **X** matrix at each iteration and works only by transforming the covariance matrix **S**.

b. It generalises *approach* 2 allowing the user to fix a subset of covariances.

In case of imposing high restrictions on the covariance matrix **S** the algorithm may not converge due to the unfeasibility of the designed covariance matrix or bad initials in the algorithm. If this is the case, the algorithm should be run till a feasible **S** is obtained. If convergence is not reached the designed covariance matrix is probably unfeasible.

The tolerance threshold used by default in the algorithm (10^{-10}) may be modified by the user. Convergence ratio is relatively robust to tolerance values in the range $\{10^{-3}, 10^{-4}, \ldots, 10^{-14}\}$.

The covariance matrix obtained from the new algorithm can be used as input of Arteaga and Ferrer's *approach* 1 [1] to generate a multivariate normal data set that exactly matches the built covariance matrix.

The proposal can be useful for testing the performance of chemometric methods with data sets matching the theoretical conditions for their applicability or checking their robustness when the hypothesized properties fail.

Acknowledgements

This research was supported by the Spanish Ministry of Science and Innovation - MICINN (grant DPI2011-28112-C04-02) and the Spanish Ministry of Science and Technology (grant SEJ2010-17475/ECON).

Appendix A

In this appendix we show the MatLab code for the algorithm.

```
function [S It conv]=sbuild(K,eig_val,var,co,S0)
% inputs
% K:        number of variables for the wanted S matrix
% eig_val:  list of the specified eigenvalues [l1 l2 ...]
% var:      specified variances [v1 v2 ... vK]. if var=0 then all
%           the variances are ones (S will be a correlation matrix)
% co:       specified covariances [i1 j1 co1;i2 j2 co2;...] can be []
% S0:       seed matrix. If S0=[] sbuild becomes stochastic
% outputs
% S:        wanted covariance matrix
% It:       number of iterations to achieve convergence
% conv:     1 if the algorithm converges, 0 otherwise
% calls
% S=sbuild(5,[2.5 1.5 0],0,[],[])
%       S is a 5 by 5 correlation matrix with 3 eigenvalues equal to
%       2.5, 1.5 and 0. Two eigenvalues randomly determined such that
%       they sum up 5-(2.5+1.5)=1.
% S=sbuild(5,[2.5 1.5 0],0,[1 2 .5;1 3 .5;2 4 -.5],[])
%       S is as above, but S12=0.5, S13=0.5 and S24=-.5

% if the variances are not specified, they are fixed as ones
if var==0, var=ones(1,K); end

if size(S0)==[0 0],
  S=cov(rand(K+1,K));
else
  S=S0;
end
ifixcov=[]; fixcov=[];
if size(co,1)>0,
  for i=1:size(co,1);
    ifixcov=[ifixcov (co(i,2)-1)*K+co(i,1) (co(i,1)-1)*K+co(i,2)];
    fixcov=[fixcov co(i,3) co(i,3)];
  end
end

TV=sum(var);
e1=sort(eig_val(find(eig_val~=0)),'descend');
ne1=size(e1,2);ve1=sum(e1);
RV=TV-ve1;
if RV<0,
  error('Fixed eigenvalues cannot sum up more than the variances');
end
ne3=size(eig_val(find(eig_val==0)),2);

ne2=K-ne1-ne3;

alea=rand(1,ne2);
e2=sort(RV/sum(alea)*alea,'descend');

[e ie]=sort([e1 e2],'descend');

conv=0;It=0;Tol=1e-10;OKeig=1;

while 1,
  It=It+1;
  if It>=5000, break, end
  [v,d]=svd(S);
```

```
  v=v(:,1:ne1+ne2);  % STEP 1
  d=diag(d)';
  d=d(1:ne1+ne2);

  dalea=d(ie(ne1+1:ne1+ne2));
  dalea=dalea*RV/sum(dalea);
  [e ie]=sort([e1 dalea],'descend');

  S=v*diag(e)*v';     % STEP 2
  S=rescale(S,var);   % STEP 3 re-scale S
  S(ifixcov)=fixcov;  % STEP 4 fix covariances
  [v d]=eig(S);
  d=abs(d);           % STEP 5 avoids negative eigenvalues
  S=v*d*v';
  S=rescale(S,var);

  % test for covariances
  OKcov=1;
  if size(fixcov,1)>0 && max((S(ifixcov)-fixcov).^2)>Tol, OKcov=0; end

  % test for eigenvalues
  OKeig=0;
  if ne1+ne3>0,
    OKeig=zeros(1,ne1+ne3); ei=diag(d);
    for i=1:ne1+ne3,
      for j=1:K,
        if (ei(j)-eig_val(i))^2<Tol && OKeig(i)==0,
          OKeig(i)=1; ei(j)=-1;
        end
      end
    end
  end

  % check for convergence
  if sum(OKeig)+OKcov==ne1+ne3+1, conv=1;break;end
end

if size(fixcov,1)>0 && OKcov==0,
  warning('fail in matching the fixed covariances');
end
if ne1+ne3>0 && sum(OKeig)~=ne1+ne3,
  warning('fail in matching the fixed eigenvalues');
end

function S=rescale(S,var)
K=size(S,1);
for i=1:K-1
  for j=i+1:K,
    S(i,j)=S(i,j)*sqrt(var(i)*var(j)/(S(i,i)*S(j,j))); S(j,i)=S(i,j);
  end
end
for i=1:K, S(i,i)=var(i); end
```

References

[1] F. Arteaga, A. Ferrer, How to simulate normal data sets with the desired correlation structure, *Chemometrics and Intelligent Laboratory Systems* 101 (2010) 38-42.

[2] J.E. Gentle, Cholesky Factorization, *Numerical Linear Algebra for Applications*, Springer-Verlag, Berlin, 1998.

[3] R.B. Bendel, M.R. Mickey, Population correlation matrices for sampling experiments, *Communications in Statistics–Simulation and Computation* B7:2 (1978) 163-182.

[4] S.P. Lin, R.B. Bendel, Algorithm AS 213: Generation of Population Correlation Matrices with Specified Eigenvalues, *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 34 (1985) 193-198.

[5] D. Kurowicka, R.M. Cooke, Completion problem with partial correlation vines, *Linear Algebra and its Applications* 418 (2006) 188-200.

[6] C. Guinot, J. Latreille, M. Tenenhaus, PLS Path modelling and multiple table analysis. Application to the cosmetic habits of women in Ile-de-France, *Chemometrics and Intelligent Laboratory Systems* 58 (2001) 247-259.

[7] M. Tenenhaus, V.E. Vinzi, Y.M. Chatelin, C. Lauro, PLS path modeling, *Computational Statistics & Data Analysis* 48 (2005) 159-205.

[8] D. Gefen, D. Straub, A practical guide to factorial validity using PLS-Graph: Tutorial and annotated example, *Communications of the Association for Information Systems* 16 (2005) 91-109.

[9] W.W. Chin, How to write up and report PLS analyses, in: V. Esposito Vinzi, W.W. Chin, J. Henseler, and H. Wang (Eds.), Handbook of partial least squares: Concepts, methods, and applications, Springer, Berlin, 2010, pp. 655–690.

[10] J. Camacho, J. Picó, A. Ferrer, Bilinear modelling of batch processes. Part I: theoretical discussion, Journal of Chemometrics 22 (2008) 299–308.