Computation of Multiple Correspondence Analysis, with code in R

Oleg Nenadić[1] **and Michael Greenacre** [2]

[1] Institut für Statistik und Ökonometrie
Georg-August-Universität Göttingen
Platz der Göttinger Sieben 5
37073 Göttingen. Germany

E-mail: onenadi@uni-goettingen.de

[2] Departament d'Economia i Empresa
Universitat Pompeu Fabra
Ramon Trias Fargas, 25-27
08005 Barcelona. Spain

E-mail: michael@upf.es

Acknowledgement
The support of the BBVA Foundation (Fundación BBVA) in Madrid in sponsoring this research is gratefully acknowledged.

Computation of Multiple Correspondence Analysis, with code in R

Oleg Nenadić[1] **and Michael Greenacre** [2]

Abstract

The generalization of simple correspondence analysis, for two categorical variables, to multiple correspondence analysis where they may be three or more variables, is not straighforward, both from a mathematical and computational point of view. In this paper we detail the exact computational steps involved in performing a multiple correspondence analysis, including the special aspects of adjusting the principal inertias to correct the percentages of inertia, supplementary points and subset analysis. Furthermore, we give the algorithm for joint correspondence analysis where the cross-tabulations of all unique pairs of variables are analysed jointly. The code in the R language for every step of the computations is given, as well as the results of each computation.

Keywords

Adjustment of principal inertias, Burt matrix, correspondence analysis, multiple correspondence analysis, R language, singular value decomposition, subset analysis.

1 Introduction

Multiple correspondence analysis (MCA) is essentially the application of the simple correspondence analysis (CA) algorithm to multivariate categorical data coded in the form of an indicator matrix or a Burt matrix (see, for example, Greenacre, 1984, 1993, 2005). Blasius and Greenacre (1994) described in detail the computations involved in CA. In this paper we describe the steps involved in computing MCA solutions as well as related results such as the coordinates of supplementary points and the adjustment of principal inertias (eigenvalues). We shall also describe an algorithm for joint correspondence analysis (JCA). Our computing environment is mainly the freeware program R (www.R-project.org) but all the analyses described in this Appendix have also been implemented in the XLSTAT

package (www.xlstat.com) and the results have all been corroborated using XLSTAT. In Section 8 we give the actual code in R that computed all the results described here.

The computing steps are illustrated using the western German sample taken from the International Survey Program on Environment (ISSP, 1993). There were four questions on attitudes to science, labelled "*A*" to "*D*", with responses on a 5-point scale (1=agree strongly to 5=disagree strongly), as well as three demographic variables, *sex* (2 categories), *age* (6 categories) and *education* (6 categories).

	A	*B*	*C*	*D*	*sex*	*age*	*education*
1	2	3	4	3	2	2	3
2	3	4	2	3	1	3	4
3	2	3	2	4	2	3	2
4	2	2	2	2	1	2	3
5	3	3	3	3	1	5	2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
871	1	2	2	2	2	3	6

TABLE 1: *Extract from the ISSP survey (ISSP, 1993).*

Table 1 shows a part of the survey data for western Germany. The columns of this data matrix contain Q (= 4) questions corresponding to the active variables and Q' (= 3) questions corresponding to the supplementary variables. Each question q, active or supplementary, has a certain number J_q of response categories. In the R language, each of the questions defines a factor, and each factor has a certain number of levels. In our example $J_q = 5$ for each active factor and the supplementary factors have 2, 6 and 6 levels respectively. Initially we only consider the MCA of questions *A*, *B*, *C* and *D*. The supplementary variables *sex*, *age* and *education* are used at a later stage for showing the computations for supplementary points in MCA.

2 Computations based on the indicator matrix

The most classical and standard approach to MCA is to apply a simple CA to the indicator matrix $\mathbf{Z}$. The indicator matrix $\mathbf{Z} = \{z_{ij}\}$ corresponds to a binary coding of the factors - instead of using a factor with J_q levels one uses J_q columns containing binary values, also called dummy variables. Table 2 illustrates a part of the indicator matrix for the ISSP survey data:

	A					*B*					*C*					*D*				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
1	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0
2	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0
3	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0
4	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0
5	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
871	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0

TABLE 2: *Extract from the indicator matrix for the first four columns of Table 1.*

We assume here that a (simple) CA program is available, based on the singular value decomposition (SVD). (see Greenacre and Blasius (1994) for more details of the computations involved).

The number of nonzero singular values of an indicator matrix based on Q factors with a total of J levels ($J = \sum_q J_q$) is $J - Q$, which in our example is $20 - 4 = 16$. Table 3 gives some of the 16 principal inertias (squares of the singular values) and the explained percentages of inertia.

s	1	2	3	4	...	16
λ_s	0.457	0.431	0.322	0.306	...	0.125
Explained inertia (in %)	11.4	10.8	8.0	7.7	...	3.1

TABLE 3: *Some principal inertias and explained inertia for the MCA of Table 2.*

The column standard and principal coordinates (b_{js} and g_{js}, respectively) for the first two dimensions ($s = 1, 2$) are shown in Table 4. The results are given only for the responses to the questions A and D.

	A						*D*				
	1	2	3	4	5	...	1	2	3	4	5
b_{j1}	1.837	0.546	– 0.447	– 1.166	– 1.995	...	1.204	– 0.221	– 0.385	– 0.222	0.708
b_{j2}	– 0.727	0.284	1.199	– 0.737	– 2.470	...	– 1.822	0.007	1.159	0.211	– 1.152
g_{j1}	1.242	0.369	– 0.302	– 0.788	– 1.349	...	0.814	– 0.150	– 0.260	– 0.150	0.479
g_{j2}	– 0.478	0.187	0.787	– 0.484	– 1.622	...	– 1.196	0.005	0.761	0.138	– 0.756

TABLE 4: *Some column standard and principal coordinates for the first two dimensions.*

A small part of the row principal coordinates f_{is} is displayed in Table 5

i	1	2	3	4	5	...	871
f_{i1}	– 0.210	– 0.325	0.229	0.303	– 0.276	...	0.626
f_{i2}	0.443	0.807	0.513	0.387	1.092	...	0.135

TABLE 5: *Some row principal coordinates for the first two dimensions.*

Figure 1 gives the complete result as a (symmetric) map for the first two dimensions. The four questions from the survey are coded with different symbols and the rows (i.e. individuals) are displayed as small dots. Notice that a potential inversion in sign with respect to the axes is irrelevant to the interpretation and is a function of the algorithm used to calculate the solution.

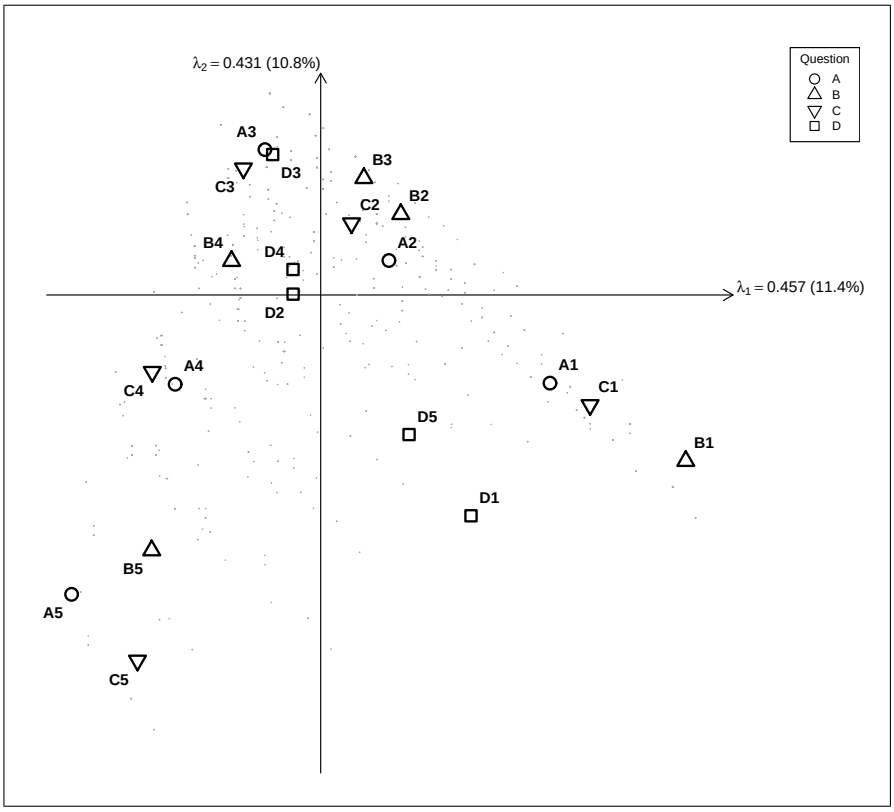

Figure 1: *Symmetric map of the ISSP dataset.*

3 Computations based on the Burt matrix

The Burt matrix $\mathbf{C}$ is obtained directly from the indicator matrix $\mathbf{Z}$: $\mathbf{C} = \mathbf{Z}^T\mathbf{Z}$. Table 6 shows a part of the Burt matrix from the ISSP dataset, corresponding to questions A and D.

		A						D				
		1	2	3	4	5	...	1	2	3	4	5
	1	119	0	0	0	0		15	25	17	34	28
	2	0	322	0	0	0		22	102	76	68	54
A	3	0	0	204	0	0	...	10	44	68	58	24
	4	0	0	0	178	0		9	52	28	54	35
	5	0	0	0	0	48		4	9	13	12	10
	⋮			⋮			⋱			⋮		
	1	15	22	10	9	4		60	0	0	0	0
	2	25	102	44	52	9		0	232	0	0	0
D	3	17	76	68	28	13	...	0	0	202	0	0
	4	34	68	58	54	12		0	0	0	226	0
	5	28	54	24	35	10		0	0	0	0	151

TABLE 6: *Data of the ISSP survey in the form of a Burt matrix.*

The computation of MCA is again the application of the (simple) CA algorithm to the Burt matrix $\mathbf{C}$. Notice, however, the following properties of this analysis and its relation with the CA of the indicator matrix $\mathbf{Z}$.

- Since $\mathbf{C}$ is symmetric, the solution for the rows and for the columns is identical
- The analysis of $\mathbf{C}$ only gives a solution for the response categories (that is, what were previously the columns of $\mathbf{Z}$)
- The standard coordinates of the rows (equivalent to columns) of $\mathbf{C}$, are identical to the standard coordinates of the columns of $\mathbf{Z}$.
- The principal inertias of $\mathbf{C}$ are the squares of those of $\mathbf{Z}$.
- Since the matrix of standardized residuals in the analysis of $\mathbf{C}$ is positive definite symmetric, the singular values in the analysis of $\mathbf{C}$ are also eigenvalues.

It is useful here to give the steps for a minimal stand-alone computing algorithm for performing an MCA starting from the Burt matrix $\mathbf{C}$:

1. Divide $\mathbf{C}$ by its grand total $n = \sum_{i,j} c_{ij}$ to obtain the correspondence matrix $\mathbf{P}$:
$$\mathbf{P} = \{p_{ij}\} = c_{ij}/n \tag{1}$$
and calculate the row totals (masses) r_i (equal to column masses).
2. Perform an eigenvalue-eigenvector decomposition on standardized residuals $\mathbf{A}$ (which - as pointed out above - is the same as the SVD)
$$\mathbf{S} = \{s_{ij}\} = (p_{ij} - r_i r_j)/\sqrt{r_i r_j} \tag{2}$$

The decomposition returns the eigenvectors $\mathbf{U} = \{u_{is}\}$ and the eigenvalues λ_s from the solution of $\mathbf{S} = \mathbf{V}\Lambda\mathbf{V}^T$. The eigenvalues (= singular values) are equivalent to the λ_s from Table 3, hence are the principal inertias of $\mathbf{Z}$. If the principal inertias of $\mathbf{C}$ are required, they need to be squared. Table 7 gives the principal inertias for the MCA based on the Burt matrix.

s	1	2	3	4	...	16
λ_s^2	0.2092	0.1857	0.1036	0.0939	...	0.0157
Explained inertia (in %)	18.6	16.5	9.2	8.3	...	1.4

TABLE 7: *Some principal inertias and explained inertia for the CA of Table 6.*

3. The i-th row (or column) standard coordinate for the s-th dimension is obtained as

$$a_{is} = v_{is}/\sqrt{r_i} \tag{3}$$

4. The corresponding principal coordinates are given by

$$f_{is} = a_{is}\lambda_s \tag{4}$$

Table 8 contains a part of the eigenvectors for the first two dimensions (u_{i1} and u_{i2}), the (row or column) category masses (r_i) and the (row or column) standard and principal coordinates (a_{i1} and a_{i2} and f_{i1} and f_{i2}, respectively). The principal coordinates differ from those reported in Table 4, because they are the standard coordinates scaled by λ_s whereas in Table 4 they are scaled by $\sqrt{\lambda_s}$.

	A						*D*				
	1	2	3	4	5	...	1	2	3	4	5
u_{i1}	0.339	0.166	– 0.108	– 0.264	– 0.234	...	0.158	– 0.057	– 0.093	– 0.056	0.147
u_{i2}	– 0.134	0.086	0.290	– 0.167	– 0.290	...	– 0.239	0.002	0.279	0.054	– 0.240
r_i	0.034	0.092	0.059	0.051	0.014	...	0.017	0.067	0.058	0.065	0.043
a_{i1}	1.837	0.546	– 0.447	– 1.166	– 1.995	...	1.204	– 0.221	– 0.385	– 0.222	0.708
a_{i2}	– 0.727	0.284	1.199	– 0.737	– 2.470	...	– 1.822	0.007	1.159	0.211	– 1.152
f_{i1}	0.840	0.250	– 0.204	– 0.533	– 0.913	...	0.551	– 0.101	– 0.176	– 0.101	0.324
f_{i2}	– 0.314	0.123	0.517	– 0.318	– 1.064	...	– 0.785	0.003	0.499	0.091	– 0.496

TABLE 8: *The eigenvectors, masses, standard and principal coordinates for the analysis of the Burt matrix.*

4 Adjustment of Inertias

The so-called "percentage of inertia problem" can be improved by using adjusted inertias:

$$\lambda_s^{\text{adj}} = \left(\frac{Q}{Q-1}\right)^2 \left(\lambda_s - \frac{1}{Q}\right)^2 \tag{5}$$

The adjusted inertias are calculated only for each singular value λ_s that satisfies the inequality $\lambda_s \leq 1/Q$. They are expressed as a percentage of the average off-diagonal inertia, which can be calculated either by direct calculation on the off-diagonal tables in the Burt matrix, or from the total inertia of $\mathbf{C}$ as follows:

$$\frac{Q}{Q-1}\left(\text{inertia}(\mathbf{C}) - \frac{J-Q}{Q^2}\right) \tag{6}$$

where inertia($\mathbf{C}$) is the sum of the principal inertias $\sum_s \lambda_s^2$ in Table 7. The value of (6) in our ISSP example is 0.17024 and Table 9 lists the adjusted inertias for the six dimensions that satisfy $\lambda_s \leq \frac{1}{4}$.

s	1	2	3	4	5	6
λ_s^{adj}	0.07646	0.05822	0.00920	0.00567	0.00117	0.00001
Explained inertia (in %)	44.9	34.2	5.4	3.3	0.7	0.0

TABLE 9: *Adjusted principal inertias and explained inertia for the ISSP survey.*

5 Joint Correspondence Analysis

The main diagonal submatrices of the Burt matrix are the key issue. The JCA analysis of the ISSP example is performed here by using iteratively weighted least squares for updating the diagonal submatrices of the Burt matrix. The updating is carried out by calculating the MCA solution for the dimensions $1, \ldots, S^*$ where S^* is the required dimensionality of the solution (this has to be chosen in advance, since the solution is no longer nested).

The procedure is carried out in the following steps:

1. Set $\mathbf{C}^* = \{c_{ij}^*\} = c_{ij}$
2. Perform an MCA on $\mathbf{C}^*$.
3. Reconstruct the approximation of the data from the solution:

$$\hat{\mathbf{C}} = \{\hat{c}_{ij}\} := n\, r_i r_j \left(1 + \sum_{s=1}^{S^*} \lambda_s a_{is} a_{js}\right) \tag{7}$$

where S^* is the required dimensionality of the solution. (In the first iteration of the algorithm, one can optionally use adjusted inertias.)

4. Update main diagonal submatrices of $\mathbf{C}^*$ with the corresponding entries of $\hat{\mathbf{C}}$

5. Repeat steps 2 - 4 until convergence

One possibility to measure the convergence is given by considering the maximum absolute difference between the entries of the main diagonal matrices of $\mathbf{C}^*$ and $\hat{\mathbf{C}}$ in step 4). Table 10 shows the updated Burt matrix after 45 iterations, after achieving a maximum absolute difference less than 0.0001, based on a two-dimensional solution. Values in the main diagonal submatrices that were modified are typed in boldface.

		A						*D*				
		1	2	3	4	5	...	1	2	3	4	5
	1	**30.72**	**53.14**	**18.59**	**13.79**	**2.58**		15	25	17	34	28
	2	**53.14**	**130.55**	**76.80**	**51.80**	**9.71**		22	102	76	68	54
A	3	**18.59**	**76.80**	**62.59**	**38.86**	**6.80**	...	10	44	68	58	24
	4	**13.97**	**51.80**	**38.86**	**53.51**	**19.85**		9	52	28	54	35
	5	**2.58**	**9.71**	**6.80**	**19.85**	**9.06**		4	9	13	12	10
	⋮			⋮			⋱			⋮		
	1	15	22	10	9	4		**9.02**	**14.67**	**5.03**	**13.27**	**18.01**
	2	25	102	44	52	9		**14.67**	**62.46**	**55.78**	**60.90**	**38.20**
D	3	17	76	68	28	13	...	**5.03**	**55.78**	**63.56**	**56.49**	**21.14**
	4	34	68	58	54	12		**13.27**	**60.90**	**56.49**	**59.74**	**35.60**
	5	28	54	24	35	10		**18.01**	**38.20**	**21.14**	**35.60**	**38.04**

TABLE 10: *The updated (or modified) Burt matrix.*

In order to measure the quality of the approximation, it should be remembered that in the final CA of the modified Burt matrix $\mathbf{C}^*$ in Table 10, the total inertia includes contributions due to the modified diagonal blocks and that these are perfectly fitted by the two-dimensional solution. These must be discounted from both the total inertia and the first two principal inertias. The principal inertias and the explained inertia for the two-dimensional JCA solution are given in Table 11, corresponding to a total inertia of the modified Burt matrix of 0.18242.

s	1	2
λ_s^{JCA}	0.09909	0.06503
Explained inertia (in %)	54.3	35.6

TABLE 11: *Adjusted principal inertias and explained inertia for the JCA case.*

By direct calculation on Table 10, the contributions to the total inertia due to the different submatrix blocks are given in Table 12.

	A	B	C	D
A	0.00745	0.01486	0.01215	0.00329
B	0.01486	0.02244	0.01858	0.00530
C	0.01215	0.01858	0.02103	0.00966
D	0.00329	0.00530	0.00966	0.00381

TABLE 12: *Contributions to the total inertia of each submatrix of table 10.*

The total of Table 12 is 0.18242, the total inertia of the modified Burt matrix, of which the sum of the diagonal values 0.05474 needs to be discounted from the first two principal inertias as well as the total. This gives the proportion of inertia explained by the two-dimensional JCA solution as

$$\frac{0.09909 + 0.06503 - 0.05474}{0.18242 - 0.05474} = \frac{0.10938}{0.12768} = 0.8567$$

hence 85.7% of the (off-diagonal) inertia is explained by the JCA solution. This percentage is necessarily less than the percentage explained in the whole matrix (equal to 89.9%, see Table 11), since the latter calculation includes the modified diagonal blocks which are fitted perfectly.

Notice finally that the denominator 0.12768 of this proportion (and thus the difference of 0.05474 due to the diagonal blocks) can be obtained easily from our previous results concerning the average off-diagonal inertia, formula (6). We had calculated the average off-diagonal inertia to be 0.17024, which is the average of 12 inertias from individual off-diagonal tables, whereas our present calculation involves the average of 16 tables. Hence the inertia in the modified Burt matrix due to the off-diagonal tables is $\frac{12}{16}\,0.17024$, that is 0.12768. Hence, the part due to the modified tables on the diagonal is $0.18242 - 0.12768 = 0.05474$, and so it is actually not necessary to calculate the contributions on the diagonal of Table 12 directly.

6 Supplementary variables

In simple CA supplementary row or column points are commonly calculated as weighted averages of the column or row standard coordinates respectively: for example, to position a supplementary column in principal coordinates, the profile of the supplementary column is used to calculate a weighted average of the row standard coordinates. In the MCA of the indicator matrix there are two ways to represent supplementary variables: first, as regular supplementary columns as just described, which amounts to averaging respondent points

in standard coordinates, second as averages of respondent points in principal coordinates, which amounts to appending to the indicator matrix, as rows, the concatenated cross-tabulations of the supplementary variables with the active variables. It is this latter option that is preferable, since it is a unified strategy across all forms of MCA, in particular the MCA with adjusted inertias, which is the form we prefer.

Before treating the second option, let us first recall the standard case in the analysis of the indicator matrix $\mathbf{Z}$, i.e. supplementary column categories whose position is obtained by averaging over row standard coordinates. Suppose that we have a supplementary variable coded in indicator form as the matrix $\mathbf{Z}^*$, with (i,j)-th column element z^*_{ij} and $z^*_{\cdot j}$ its corresponding column sum. Given the standard row (respondent) coordinates a_{is} the supplementary column principal coordinates g^*_{js} are given as

$$g^*_{js} = \sum_{i=1}^{I} \frac{z^*_{ij}}{z_{\cdot j}} \, a_{is} \tag{8}$$

which, since the values z^*_{ij} are either 0 or 1, is the average of the standard coordinates of those respondents in category j. Table 13 shows these column coordinates.

	sex		*age*						*education*					
	1	2	1	2	3	4	5	6	1	2	3	4	5	6
g^*_{j1}	-0.143	0.137	-0.166	-0.087	-0.025	-0.031	0.016	0.281	0.180	0.161	-0.068	-0.227	-0.172	-0.308
g^*_{j2}	0.029	-0.028	-0.014	-0.081	-0.004	0.057	0.047	0.033	0.060	0.093	0.090	-0.279	-0.263	-0.291

TABLE 13: *Supplementary principal coordinates for the variables sex, age and education, as columns of the indicator matrix.*

The second (and preferable) method is based on averaging the respondent row points in principal coordinates, which is equivalent to appending the cross-tabulations $\mathbf{Z}^{*T}\mathbf{Z}$ (of the supplementary variable with the active variables) as supplementary rows of $\mathbf{Z}$. This gives the same numerical coordinates as appending $\mathbf{Z}^{*T}\mathbf{Z}$ as supplementary rows to the Burt matrix $\mathbf{C}$. Or, since the Burt matrix is symmetric, one can append the transposed cross-tabulations $\mathbf{Z}^T\mathbf{Z}^*$ to $\mathbf{C}$ as supplementary columns. To illustrate the calculations, suppose that $\mathbf{C}^*$ denotes the latter cross-tabulations $\mathbf{Z}^T\mathbf{Z}^*$ (stacked vertically) appended as columns to the Burt matrix, with general element c^*_{ij} and column sums $c^*_{\cdot j}$. In the analysis of $\mathbf{C}$ we denote the (row) standard coordinates of the active response categories by $\tilde{a}_{is}$ (these are not the same as the a_{is} of (8) which in that case refer to standard coordinates of respondent points i in the analysis of the indicator matrix $\mathbf{Z}$ - in this case i refers to an active row category of the Burt matrix and runs from 1 to J). Now the positions of the supplementary columns $\mathbf{C}^*$ are obtained by weighted averaging as follows:

$$\tilde{g}_{js} = \sum_{i=1}^{J} \frac{c^*_{ij}}{c^*_{\cdot j}} \, \tilde{a}_{is} \tag{9}$$

Table 14 gives the supplementary principal coordinates for the response categories of the variables sex, age and education in the ISSP example, which (to emphasise) can be considered either as supplementary rows of $\mathbf{Z}$ or as supplementary rows or columns of $\mathbf{C}$. Notice that, since Table 12 contains averages over standard coordinates and Table 14 has effectively calculated averages of row (respondent) points over principal coordinates, the values in Table 14 are those in Table 13 multiplied by square roots of corresponding principal inertias in the analysis of the indicator matrix $\mathbf{Z}$:

$$\tilde{g}_{js} = g^*_{js}\sqrt{\lambda_s}$$

For example, the coordinate $g^*_{11} = -0.143$ for sex1 (male) on the first dimension in Table 13 would be multiplied by the square root of 0.457 (see Table 3) to give: $-0.143\sqrt{0.457} = -0.097$ – this checks with the corresponding element $\tilde{g}_{11}$ in Table 14.

	sex		*age*						*education*					
	1	2	1	2	3	4	5	6	1	2	3	4	5	6
$\tilde{g}_{j1}$	-0.097	0.093	-0.112	-0.059	-0.017	-0.021	0.011	0.190	0.122	0.109	-0.046	-0.154	-0.116	-0.209
$\tilde{g}_{j2}$	0.019	-0.018	-0.009	-0.053	-0.003	0.038	0.031	0.022	0.039	0.061	0.059	-0.183	-0.172	-0.191

TABLE 14: *Supplementary principal coordinates for the variables sex, age and education, computed as supplementary rows of* $\mathbf{Z}$ *or supplementary rows or columns of* $\mathbf{C}$.

7 Subset analysis

In this section we detail briefly the adaptation needed to the basic CA algorithm to perform the subset analyses. For example, suppose we wished to exclude the middle "neither agree nor disagree" responses for questions A to D from the analysis. Again, we can approach this either from the viewpoint of the analysis of the indicator matrix $\mathbf{Z}$ or of the Burt matrix $\mathbf{C}$. The idea is to execute the same CA algorithm to the corresponding submatrix of $\mathbf{Z}$ or $\mathbf{C}$ but maintain the original row and column margins of the matrix. We would thus first calculate the complete matrix to be decomposed, centre them as usual with respect to their row and column margins, and then extract the submatrix of interest, excluding rows and/or columns to be ignored. In the case of $\mathbf{Z}$ in our example, this would give a 871×16 submatrix (excluding the 4 columns corresponding to the 4 "neither ... nor" categories, while in the case of $\mathbf{C}$ this would give a 16×16 submatrix (excluding 4 rows and columns). In terms of our earlier description, the latter option would mean performing an SVD on the submatrix of $\mathbf{S}$ calculated as in (2). The results are given in the Tables 15 and 16. Note that in this case the subset analysis also has 16 dimensions, since there are no linear dependencies between the rows or columns of the submatrix analysed. Notice further that the percentages calculated in Table 15 are relative to that part of the inertia contained in the 16×16 submatrix, which is inflated by values on diagonal blocks just as in MCA of the full Burt matrix. Whether there are simple ways in this case to adjust these principal inertias to obtain more realistic percentages of inertia still needs to be investigated.

s	1	2	3	4	...	16
λ_s^{sub}	0.2016	0.1489	0.0980	0.0721	...	0.0017
Explained inertia (in %)	23.4	17.3	11.4	8.4	...	0.2

TABLE 15: *Some principal inertias and explained inertia for the subset MCA of Table 6, excluding the categories "neither ... nor"*

	A					*D*			
	1	2	4	5	...	1	2	4	5
b_{j1}	1.837	– 0.538	– 1.316	– 2.449	...	0.888	– 0.273	– 0.222	0.482
b_{j2}	1.153	– 0.517	– 0.015	2.746	...	2.482	– 0.462	– 0.683	1.210
g_{j1}	0.761	0.242	– 0.591	– 1.100	...	0.399	– 0.123	– 0.100	0.216
g_{j2}	0.445	– 0.200	– 0.006	1.060	...	0.957	– 0.178	– 0.264	0.467

TABLE 16: *Some column standard and principal coordinates for the first two dimensions of the subset MCA of Table 6, excluding categories "neither ... nor".*

8 A sample session in R

In this section we give some selected code using the programming language R (R Development Core Team, 2005). The complete language and associated material such as program manuals and contributed packages from researchers all over the world are freely downloadable from *http://www.r-project.org*. The code that follows gives the results given in Tables 1 to 16 of the previous sections.

We assume that the dataset is loaded into R as a `data.frame` named `dat`, using the function `read.table()` with the option `colClasses="factor"` so that columns are declared to be factors. Suppose that the first lines of the data file look like this:

```
A B C D sex age edu
2 3 4 3   2   2   3
3 4 2 3   1   3   4
2 3 2 4   2   3   2
2 2 2 2   1   2   3
3 3 3 3   1   5   2
. . . .   .   .   .
```

that is, a header with the variable (column) names, followed by the data for each respondent. Further, suppose it is stored under the name WG93.txt in the current working directory. Then the R statement to input the data matrix would be:

```
dat <- read.table("WG93.txt", header = TRUE, colClasses = "factor")
```

The following R statements that are displayed with an indent are used to calculate the numerical results given in the previous tables. The results for the tables are displayed with the R commands written without indent at the bottom of each subsection.

Table 1 (response pattern matrix)

```
  sup.ind <- 5:7
  dat.act <- dat[,-sup.ind]
  dat.sup <- dat[,sup.ind]
  I       <- dim(dat.act)[1]
  Q       <- dim(dat.act)[2]
dat[c(1:5,I),]
#     A B C D sex age edu
# 1   2 3 4 3   2   2   3
# 2   3 4 2 3   1   3   4
# 3   2 3 2 4   2   3   2
# 4   2 2 2 2   1   2   3
# 5   3 3 3 3   1   5   2
# 871 1 2 2 2   2   3   6
```

Table 2 (indicator matrix)

```
  lev.n  <- unlist(lapply(dat, nlevels))
  n      <- cumsum(lev.n)
  J.t    <- sum(lev.n)
  Q.t    <- dim(dat)[2]
  Z      <- matrix(0, nrow = I, ncol = J.t)
  newdat <- lapply(dat, as.numeric)
  offset <- (c(0, n[-length(n)]))
  for (i in 1:Q.t)
    Z[1:I + (I * (offset[i] + newdat[[i]] - 1))] <- 1
  fn     <- rep(names(dat), unlist(lapply(dat, nlevels)))
  ln     <- unlist(lapply(dat, levels))
  dimnames(Z)[[2]] <- paste(fn, ln, sep = "")
  dimnames(Z)[[1]] <- as.character(1:I)
  ind.temp  <- range(n[sup.ind])
  Z.sup.ind <- (ind.temp[1]-1):ind.temp[2]
  Z.act     <- Z[,-Z.sup.ind]
  J         <- dim(Z.act)[2]
Z.act[c(1:5,I),]
#     A1 A2 A3 A4 A5 B1 B2 B3 B4 B5 C1 C2 C3 C4 C5 D1 D2 D3 D4 D5
# 1    0  1  0  0  0  0  0  1  0  0  0  0  0  1  0  0  0  1  0  0
# 2    0  0  1  0  0  0  0  0  1  0  0  1  0  0  0  0  0  1  0  0
# 3    0  1  0  0  0  0  0  1  0  0  0  1  0  0  0  0  0  0  1  0
# 4    0  1  0  0  0  0  1  0  0  0  0  1  0  0  0  0  1  0  0  0
# 5    0  0  1  0  0  0  0  1  0  0  0  0  1  0  0  0  0  1  0  0
# 871  1  0  0  0  0  0  1  0  0  0  0  1  0  0  0  0  1  0  0  0
```

Table 3 (inertias of Z)

```
  P    <- Z.act / sum(Z.act)
  cm   <- apply(P, 2, sum)
  rm   <- apply(P, 1, sum)
  eP   <- rm %*% t(cm)
  S    <- (P - eP) / sqrt(eP)
  dec  <- svd(S)
  lam  <- dec$d[1:(J-Q)]^2
  expl <- 100*(lam / sum(lam))
rbind(round(lam[c(1:4,(J-Q))], 3),round(expl[c(1:4,(J-Q))], 1))
#        [,1]   [,2]  [,3]  [,4]  [,5]
# [1,]  0.457  0.431 0.322 0.306 0.125
# [2,] 11.400 10.800 8.000 7.700 3.100
```

Table 4 (column standard / principal coordinates)

```
  b.s1   <- dec$v[,1] / sqrt(cm)
  b.s2   <- dec$v[,2] / sqrt(cm)
  g.s1   <- b.s1 * sqrt(lam[1])
  g.s2   <- b.s2 * sqrt(lam[2])
round(rbind(b.s1,b.s2,g.s1,g.s2)[,c(1:5,16:20)], 3)
#          A1    A2     A3     A4     A5     D1     D2     D3     D4     D5
# b.s1  1.837 0.546 -0.447 -1.166 -1.995  1.204 -0.221 -0.385 -0.222  0.708
# b.s2 -0.727 0.284  1.199 -0.737 -2.470 -1.822  0.007  1.159  0.211 -1.152
# g.s1  1.242 0.369 -0.302 -0.788 -1.349  0.814 -0.150 -0.260 -0.150  0.479
# g.s2 -0.478 0.187  0.787 -0.484 -1.622 -1.196  0.005  0.761  0.138 -0.756
```

Table 5 (row principal coordinates)

```
  f.s1   <- dec$u[,1] * sqrt(lam[1]) / sqrt(rm)
  f.s2   <- dec$u[,2] * sqrt(lam[2]) / sqrt(rm)
  a.s1 <- f.s1 / sqrt(lam[1])
  a.s2 <- f.s2 / sqrt(lam[2])
round(rbind(f.s1,f.s2)[,c(1:5,I)], 3)
#           1      2     3     4      5   871
# f.s1 -0.210 -0.325 0.229 0.303 -0.276 0.626
# f.s2  0.443  0.807 0.513 0.387  1.092 0.135
```

Table 6 (Burt matrix)

```
  B <- t(Z.act) %*% Z.act
B[c(1:5,16:20), c(1:5,16:20)]
```

```
#      A1  A2  A3  A4  A5  D1  D2  D3  D4  D5
# A1 119   0   0   0   0  15  25  17  34  28
# A2   0 322   0   0   0  22 102  76  68  54
# A3   0   0 204   0   0  10  44  68  58  24
# A4   0   0   0 178   0   9  52  28  54  35
# A5   0   0   0   0  48   4   9  13  12  10
# D1  15  22  10   9   4  60   0   0   0   0
# D2  25 102  44  52   9   0 232   0   0   0
# D3  17  76  68  28  13   0   0 202   0   0
# D4  34  68  58  54  12   0   0   0 226   0
# D5  28  54  24  35  10   0   0   0   0 151
```

Table 7 (principal inertias of Burt matrix)

```
  P.2     <- B / sum(B)
  cm.2    <- apply(P.2, 2, sum)
  eP.2    <- cm.2 %*% t(cm.2)
  S.2     <- (P.2 - eP.2) / sqrt(eP.2)
  dec.2   <- eigen(S.2)
  delt.2  <- dec.2$values[1:(J-Q)]
  expl.2  <- 100*(delt.2 / sum(delt.2))
  lam.2   <- delt.2^2
  expl.2b <- 100*(lam.2 / sum(lam.2))
rbind(round(lam.2, 3),round(expl.2b, 1))[,c(1:4,16)]
#        [,1]   [,2]  [,3]  [,4]  [,5]
# [1,]  0.209  0.186 0.104 0.094 0.016
# [2,] 18.600 16.500 9.200 8.300 1.400
```

Addendum: "check" that δ_s is equivalent to λ_s:

```
rbind(round(delt.2, 3),round(expl.2, 1),
      round(lam, 3),round(expl, 1))
```

Table 8 (eigenvectors, column masses, column sc/pc's)

```
  u.s1  <- dec.2$vectors[,1]
  u.s2  <- dec.2$vectors[,2]
  a2.s1 <- u.s1 / sqrt(cm.2)
  a2.s2 <- u.s2 / sqrt(cm.2)
  f2.s1 <- a2.s1 * sqrt(lam.2[1])
  f2.s2 <- a2.s2 * sqrt(lam.2[2])
round(rbind(u.s1,u.s2,cm,a2.s1,
            a2.s2,f2.s1,f2.s2), 3)[,c(1:5,16:20)]
#           A1    A2     A3     A4     A5     D1     D2     D3     D4     D5
# u.s1   0.339 0.166 -0.108 -0.264 -0.234  0.158 -0.057 -0.093 -0.056  0.147
# u.s2  -0.134 0.086  0.290 -0.167 -0.290 -0.239  0.002  0.279  0.054 -0.240
# cm     0.034 0.092  0.059  0.051  0.014  0.017  0.067  0.058  0.065  0.043
```

```
# a2.s1  1.837 0.546 -0.447 -1.166 -1.995  1.204 -0.221 -0.385 -0.222  0.708
# a2.s2 -0.727 0.284  1.199 -0.737 -2.470 -1.822  0.007  1.159  0.211 -1.152
# f2.s1  0.840 0.250 -0.204 -0.533 -0.913  0.551 -0.101 -0.176 -0.101  0.324
# f2.s2 -0.314 0.123  0.517 -0.318 -1.064 -0.785  0.003  0.499  0.091 -0.496
```

Table 9 (adjusted inertias)

```
  lam.adj   <- (Q/(Q-1))^2 * (delt.2[delt.2 >= 1/Q] - 1/Q) ^ 2
  total.adj <- (Q/(Q-1)) * (sum(delt.2^2) - ((J-Q)/Q^2))
rbind(round(lam.adj, 5),100 * round(lam.adj / total.adj, 3))

#          [,1]     [,2]   [,3]    [,4]    [,5]  [,6]
# [1,]  0.07646  0.05822 0.0092 0.00567 0.00117 1e-05
# [2,] 44.90000 34.20000 5.4000 3.30000 0.70000 0e+00
```

Table 10 (updated Burt matrix)

```
  nd      <- 2
  maxit   <- 1000
  epsilon <- 0.0001
  lev     <- lev.n[-sup.ind]
  n       <- sum(B)
  li      <- as.vector(c(0,cumsum(lev)))
  dummy   <- matrix(0, J, J)
  for (i in 1:(length(li)-1)) {
    ind.lo <- li[i]+1
    ind.up <- li[i+1]
    ind.to <- diff(li)[i]
    dummy[rep(ind.lo:ind.up, ind.to) +
          (rep(ind.lo:ind.up, each = ind.to)-1) * J] <- 1
    }
  iterate <- function(obj, dummy, nd, adj = FALSE) {
    Bp      <- obj/n
    cm      <- apply(Bp, 2, sum)
    eP      <- cm %*% t(cm)
    cm.mat  <- diag(cm^(-0.5))
    S       <- cm.mat %*% (Bp - eP) %*% cm.mat
    dec     <- eigen(S)
    lam     <- dec$values
    u       <- dec$vectors
    phi     <- u[, 1:nd] / matrix(rep(sqrt(cm), nd), ncol = nd)
    if (adj)
      lam <- (Q / (Q - 1))^2 * (lam[lam >= 1 / Q] - 1 / Q) ^ 2
    for (s in 1:nd) {
      if (exists("coord")) {
```

```
        coord <- coord + lam[s] * (phi[,s] %*% t(phi[,s]))
        } else {
          coord <- lam[s] * (phi[,s] %*% t(phi[,s]))
          }
      }
    obj * (1 - dummy) + n * eP * dummy * (1 + coord)
    }
  # first iteration (adjusted lambda)
  B.star <- iterate(B, dummy, 2, adj = TRUE)
  # subsequent iterations
  k  <- 1
  it <- TRUE
  while (it) {
    temp    <- iterate(B.star, dummy, 2)
    delta.B <- max(abs(B.star - temp))
    B.star  <- temp
    if (delta.B <= epsilon | k >= maxit) it <- FALSE
    k <- k + 1
    }
round(B.star [c(1:5,16:20), c(1:5, 16:20)], 2)
#       A1     A2    A3    A4    A5    D1     D2    D3    D4    D5
# A1 30.72  53.14 18.59 13.97  2.58 15.00  25.00 17.00 34.00 28.00
# A2 53.14 130.55 76.80 51.80  9.71 22.00 102.00 76.00 68.00 54.00
# A3 18.59  76.80 62.95 38.86  6.80 10.00  44.00 68.00 58.00 24.00
# A4 13.97  51.80 38.86 53.51 19.85  9.00  52.00 28.00 54.00 35.00
# A5  2.58   9.71  6.80 19.85  9.06  4.00   9.00 13.00 12.00 10.00
# D1 15.00  22.00 10.00  9.00  4.00  9.02  14.67  5.03 13.27 18.01
# D2 25.00 102.00 44.00 52.00  9.00 14.67  62.46 55.78 60.90 38.20
# D3 17.00  76.00 68.00 28.00 13.00  5.03  55.78 63.56 56.49 21.14
# D4 34.00  68.00 58.00 54.00 12.00 13.27  60.90 56.49 59.74 35.60
# D5 28.00  54.00 24.00 35.00 10.00 18.01  38.20 21.14 35.60 38.04
```

Table 11 (JCA inertias)

```
  P.3    <- B.star / sum(B.star)
  cm.3   <- apply(P.3, 2, sum)
  eP.3   <- cm.3 %*% t(cm.3)
  S.3    <- (P.3 - eP.3) / sqrt(eP.3)
  delt.3 <- eigen(S.3)$values
  lam.3  <- delt.3^2
  expl.3 <- 100*(lam.3 / sum(lam.3))
rbind(round(lam.3, 3),round(expl.3, 1))[,1:2]
#        [,1]   [,2]
# [1,]  0.099  0.065
# [2,] 54.300 35.600
```

Table 12 (Inertia conributions of submatrices)

```
  subinr <- function(B, ind) {
    nn   <- length(ind)
    subi <- matrix(NA, nrow = nn, ncol = nn)
    ind2 <- c(0,cumsum(ind))
    for (i in 1:nn) {
      for (j in 1:nn) {
        tempmat   <- B[(ind2[i]+1):(ind2[i+1]),
                       (ind2[j]+1):(ind2[j+1])]
        tempmat   <- tempmat / sum(tempmat)
        ec        <- apply(tempmat, 2, sum)
        ex        <- ec%*%t(ec)
        subi[i,j] <- sum((tempmat - ex)^2 / ex)
        }
      }
    subi / nn^2
    }
  si <- subinr(B.star, lev)
round(si, 5)
#         [,1]    [,2]    [,3]    [,4]
# [1,] 0.00745 0.01486 0.01215 0.00329
# [2,] 0.01486 0.02244 0.01858 0.00530
# [3,] 0.01215 0.01858 0.02103 0.00966
# [4,] 0.00329 0.00530 0.00966 0.00381
```

Table 13 (supplementary principal coordinates as columns of Z)

```
  Z.star  <- Z[,Z.sup.ind]
  I.star  <- dim(Z.star)[1]
  cs.star <- apply(Z.star, 2, sum)
  base    <- Z.star / matrix(rep(cs.star, I.star), nrow = I.star,
                             byrow = TRUE)
  b.star1 <- t(base) %*% cbind(a.s1, a.s2)
round(t(b.star1), 3)

#        sex1   sex2   age1   age2   age3   age4  age5  age6
# a.s1 -0.143  0.137 -0.166 -0.087 -0.025 -0.031 0.016 0.281
# a.s2  0.029 -0.028 -0.014 -0.081 -0.004  0.057 0.047 0.033
#       edu1  edu2   edu3   edu4   edu5   edu6
# a.s1  0.18 0.161 -0.068 -0.227 -0.172 -0.308
# a.s2  0.06 0.093  0.090 -0.279 -0.263 -0.291
```

Table 14 (supplementary principal coordinates via cross-tabulation)

```
  ct.star   <- t(Z.star)%*%Z.act
  I.star2  <- dim(ct.star)[2]
  cs.star2 <- apply(ct.star, 1,sum)
  base2    <- ct.star / matrix(rep(cs.star2, I.star2),
                               ncol = I.star2)
  b.star2  <- base2 %*% cbind(a2.s1, a2.s2)
round(t(b.star2), 3)
#         sex1   sex2   age1   age2   age3   age4  age5  age6
# a2.s1 -0.097  0.093 -0.112 -0.059 -0.017 -0.021 0.011 0.190
# a2.s2  0.019 -0.018 -0.009 -0.053 -0.003  0.038 0.031 0.022
#        edu1  edu2   edu3   edu4   edu5   edu6
# a2.s1 0.122 0.109 -0.046 -0.154 -0.116 -0.209
# a2.s2 0.039 0.061  0.059 -0.183 -0.172 -0.191
```

Table 15 (principal inertias from subset analysis)

```
  sub.ind  <- c(3,8,13,18)
  P.4      <- B / sum(B)
  cm.4     <- apply(P.4, 2, sum)
  eP.4     <- cm.4 %*% t(cm.4)
  S.sub    <- ((P.4 - eP.4) / sqrt(eP.4)) [-sub.ind,-sub.ind]
  dec.sub  <- eigen(S.sub)
  lam.sub  <- dec.sub$values[1:(J-Q)]^2
  expl.sub <- 100*(lam.sub / sum(lam.sub))
rbind(round(lam.sub, 4),round(expl.sub, 1))[,c(1:4,(J-Q))]
#         [,1]    [,2]   [,3]   [,4]   [,5]
# [1,]  0.2016  0.1489  0.098 0.0721 0.0017
# [2,] 23.4000 17.3000 11.400 8.4000 0.2000
```

Table 16 (column standard and principal coordinates from subset analysis)

```
  cm.sub   <- cm.4[-sub.ind]
  u.sub.s1 <- dec.sub$vectors[,1]
  u.sub.s2 <- dec.sub$vectors[,2]
  a.sub.s1 <- u.sub.s1 / sqrt(cm.sub)
  a.sub.s2 <- u.sub.s2 / sqrt(cm.sub)
  f.sub.s1 <- a.sub.s1 * sqrt(lam.sub[1])
  f.sub.s2 <- a.sub.s2 * sqrt(lam.sub[2])
round(rbind(a.sub.s1,a.sub.s2,f.sub.s1,
            f.sub.s2), 3)[, c(1:4,13:16)]
#             A1     A2     A4     A5    D1     D2     D4    D5
# a.sub.s1 1.696  0.538 -1.316 -2.449 0.888 -0.273 -0.222 0.482
# a.sub.s2 1.153 -0.517 -0.015  2.746 2.482 -0.462 -0.683 1.210
# f.sub.s1 0.761  0.242 -0.591 -1.100 0.399 -0.123 -0.100 0.216
# f.sub.s2 0.445 -0.200 -0.006  1.060 0.957 -0.178 -0.264 0.467
```

9 R functions for CA, MCA and JCA

The above code and more has been implemented in an R package `mjca`. The package comprises two core functions: `ca()` for simple correspondence analysis, and `mjca()` for multiple and joint forms of correspondence analysis. Each core function has methods for printing, summarizing and plotting (in two dimensions and three dimensions).

A short description of these functions, extracted from their help files, follows.

`ca` *Simple correspondence analysis*

Description

Computation of simple correspondence analysis.

Usage

```
ca(obj, nd = NA, suprow = NA, supcol = NA, subsetrow = NA, subsetcol = NA)
```

Arguments

`obj`	A two-way table of non-negative data, usually frequencies.
`nd`	Number of dimensions to be included in the output; if NA the maximum possible dimensions are included.
`suprow`	Indices of supplementary rows.
`supcol`	Indices of supplementary columns.
`subsetrow`	Row indices of subset.
`subsetcol`	Column indices of subset.

Details

The function `ca` computes a simple correspondence analysis based on the singular value decomposition. The options `suprow` and `supcol` allow supplementary (passive) rows and columns to be specified. Using the options `subsetrow` and/or `subsetcol` result in a subset CA being performed.

Value

`sv`	Singular values
`rownames`	Row names
`rowmass`	Row masses
`rowdist`	Row chi-square distances to centroid
`rowinertia`	Row inertias
`rowcoord`	Row standard coordinates
`rowsup`	Indices of row supplementary points
`colnames`	Column names
`colmass`	Column masses

`coldist`	Column chi-square distances to centroid
`colinertia`	Column inertias
`colcoord`	Column standard coordinates
`colsup`	Indices of column supplementary points

`mjca` *Multiple and joint correspondence analysis*

Description

Computation of multiple and joint correspondence analysis.

Usage

```
mjca(obj, nd = 3, lambda = "adjusted", suprow = NA, supcol = NA)
```

Arguments

`obj`	A response pattern matrix containing factors.
`nd`	Number of dimensions to be included in the output; if NA the maximum possible dimensions are included.
`lambda`	Gives the scaling factor for the eigenvalues. Possible values include `"indicator"`, `"Burt"`, `"adjusted"` and `"JCA"`. Using `lambda = "JCA"` results in a joint correspondence analysis.
`suprow`	Indices of supplementary rows.
`supcol`	Indices of supplementary columns.

Details

The function `mjca` computes a multiple or joint correspondence analysis based on the eigenvalue decomposition of the Burt matrix.

Value

`Ev`	Eigenvalues
`lambda`	Scaling mathod for the eigenvalues
`levelnames`	Names of the factor/level combinations
`levels.n`	Number of levels in each factor
`rownames`	Row names
`rowmass`	Row masses
`rowdist`	Row chi-square distances to centroid
`rowinertia`	Row inertias
`rowcoord`	Row standard coordinates
`rowsup`	Indices of row supplementary points
`colnames`	Column names
`colmass`	Column masses
`coldist`	Column chi-square distances to centroid

`colinertia` Column inertias
`colcoord` Column standard coordinates
`colsup` Indices of column supplementary points
`Burt` Burt matrix
`subinertia` Inertias of sub-matrices
`call` Return of `match.call`

10 XLSTAT implementation of CA and MCA

XLSTAT is a commercial statistical package for performing a range of univariate and multivariate statistical analyses in a Microsoft Excel environment. The package includes simple and multiple correspondence analysis. We have collaborated to update these two programs in XLSTAT so that they contain additional features not found in regular correspondence analysis software, such as the adjustment of principal inertias in MCA and the subset correspondence analysis option. The advantage of XLSTAT is its simple interface and the fact that all results are returned in the Excel environment. This means that it is very easy to make additional computations on the results as well as modify maps (changing the coordinate values, labels, etc). An example of its ease of use is the following. In the context of our present example, we can read the whole data matrix into Excel, both the four substantive variables as well as the three demographic variables, and then produce the complete Burt matrix of all cross-tabulations of these seven variables, in Excel format, using the MCA program. Then we can freely select with the mouse which part of the Burt matrix we want to analyse, using the CA program. The fact that both data and results are contained in the same format in Excel worksheets means that there is a seamless interface that is very easy to use, especially for Excel users.

References

Greenacre, M.J. and Blasius, J. (1994). Computation of Correspondence Analysis. In Greenacre and Blasius (eds) *Correspondence Analysis in the Social Sciences*, Academic Press, London.

ISSP (1993). International Social Survey Program on the Environment. Central Archive for Empirical Social Research, University of Cologne, Germany.

R Development Core Team (2005). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria.
ISBN 3-900051-07-0, URL http://www.R-project.org