Teaching Matrices within Statistics

Kimmo Vehkalahti
Department of Social Research, Statistics
University of Helsinki, Helsinki, Finland
http://www.helsinki.fi/people/Kimmo.Vehkalahti/

24th International Workshop on Matrices and Statistics
Haikou, Hainan, China | 25–28 May 2015

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Preliminary thoughts

How to teach matrices – within statistics?

- Every statistician needs matrices in some form, both in theoretical and practical challenges.
- Learning the necessary skills requires time and work, as the multidimensional concept of matrix is far from trivial for most students entering the university.
- Students of statistics face matrices for the first time on an elementary course of linear algebra. Enthusiasm for matrices should be kindled...
- Approach often quite mathematical, and the connection to statistics and its applications may be hidden.

Preliminary thoughts

How to teach matrices – within statistics?

- Also advanced non-statisticians may need matrices, often with no mathematical background.
- Needed: courses that do not forget about the theoretical aspects of matrix theory, but focus primarily on **computational approach with appropriate software.**
- In this talk and the mini-symposium we consider various **examples of applications and approaches** where **matrices play a significant role** – and wonder how to teach matrices within statistics.

Outline of this talk

- **Introduction**

- **Aim:** to demonstrate the possibilities of **Survo R** in **teaching matrices.**

- **How:** by showing views of Survo R used in
  1) teaching factor analysis and
  2) solving Survo puzzles with matrices:

  Vehkalahti & Sund (2014).
  Solving Survo puzzles using matrix combinatorial products,
  Journal of Statistical Computation and Simulation.

- **Conclusions**

What is needed?

For practical aspects of matrices within statistics we need:

- interactive **"learning lab"** for students' experiments
- documented **work schemes**: **create, repeat, modify, learn**
- working **step-by-step**, checking any **intermediate results**
- **self-documenting** commands with **free-form** documentation
- **good connections** with other tools of **data analysis** etc.

R is a general tool for many of these needs. Some of the features are better provided with **Survo R** and its **editorial approach**.

1 Introduction

- **Survo R**, initiated in 2009 by Reijo Sund [8, 9], is an open-source implementation representing the newest generation of the **Survo** computing environment, the lifework of prof. Seppo Mustonen since the early 1960s [4, 10].
- Survo binds together a selection of useful tools through its unique **editorial approach**, invented by Mustonen in 1979 [3]. This approach is the heart of Survo and it lets the user freely create exciting mixtures of **computational schemes** and natural language **documentation** [4, 5, 10].
- Through this approach, Survo also offers an efficient and interactive environment for **teaching** (and learning) purposes, for example for teaching matrices (possibly within statistics).

View from Survo R: teaching factor analysis

Q (EXAMPLE): Survo R - General Editorial Computing Environment for Data Analysis - ver. 0.6.20

File Edit View Help

```
*SAVE EXAMPLE / teaching matrices within statistics using Survo R
*
*
*The following matrix B313 gives a measurement structure for the
*simulation studies in Vehkalahti, Puntanen & Tarkkonen (2006).
*
*MATRIX B313
*///       F1   F2   F3     /     The structure consists of
*A1       0.9   .    .      /
*A2       0.8   .    .      /     3 factors       F1, F2, F3
*A3       0.7   .    .      /                     and
*A4       0.6   .    .      /     13 variables    A1, A2, A3, A4,
*B1        .   0.8   .      /                     B1, B2, B3,
*B2        .   0.7   .      /                     C1, C2,
*B3        .   0.6   .      /                     Z0, Z1, Z2, Z3
*C1        .    .   0.7     /
*C2        .    .   0.6     /     A,B,C: a 'simple' structure
*Z0       0.5  0.5  0.5     /
*Z1       0.6 -0.6   .      /     Z: 'confounders' (not too simple)
*Z2        .   0.6 -0.6     /
*Z3      -0.6   .   0.6     /
*
*.......................................................................
*
*Save the factor matrix, compute the theoretical correlation matrix R13:
*
+MAT SAVE B313
+MAT DIM B313 /* rowB313=13 colB313=3
+MAT R13=B313*B313'+(IDN(p,p)-DIAG(B313*B313')) / p=rowB313
*
*
*
```

Row: 2 / 1000   Column: 1 / 300   Path: e:/vanhaa/conf/CARME11/2011/

View from Survo R: highlighted matrices

Q (EXAMPLE): Survo R - General Editorial Computing Environment for Data Analysis - ver. 0.6.20

File Edit View Help

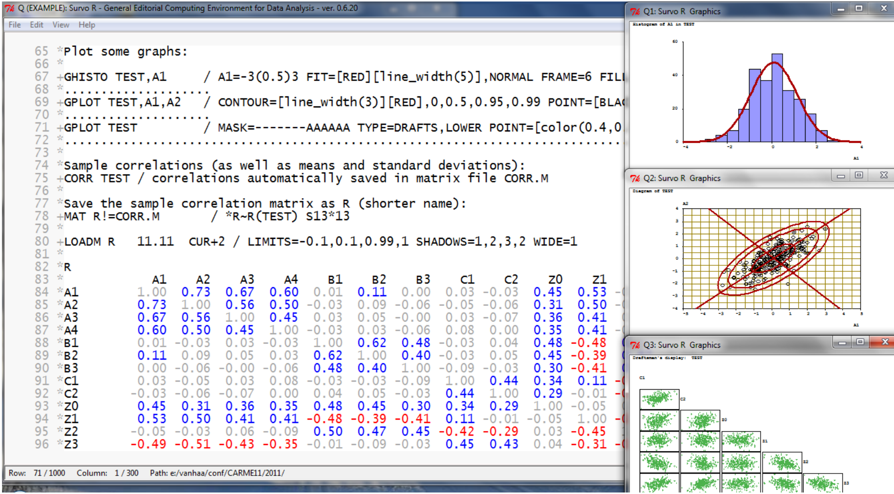
```
*
*
*Check the theoretical correlation matrix:
*
+LOADM R13 11.11 CUR+2 / LIMITS=-0.1,0.1,0.99,1 SHADOWS=1,2,3,2 WIDE=1
*
*B313*B313'+IDN-DIAG(B313*B313')
```

	A1	A2	A3	A4	B1	B2	B3	C1	C2	Z0	Z1	Z
A1	1.00	0.72	0.63	0.54	0.00	0.00	0.00	0.00	0.00	0.45	0.54	0.
A2	0.72	1.00	0.56	0.48	0.00	0.00	0.00	0.00	0.00	0.40	0.48	0.
A3	0.63	0.56	1.00	0.42	0.00	0.00	0.00	0.00	0.00	0.35	0.42	0.
A4	0.54	0.48	0.42	1.00	0.00	0.00	0.00	0.00	0.00	0.30	0.36	0.
B1	0.00	0.00	0.00	0.00	1.00	0.56	0.48	0.00	0.00	0.40	-0.48	0.4
B2	0.00	0.00	0.00	0.00	0.56	1.00	0.42	0.00	0.00	0.35	-0.42	0.4
B3	0.00	0.00	0.00	0.00	0.48	0.42	1.00	0.00	0.00	0.30	-0.36	0.3
C1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.42	0.35	0.00	-0.4
C2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.42	1.00	0.30	0.00	-0.3
Z0	0.45	0.40	0.35	0.30	0.40	0.35	0.30	0.35	0.30	1.00	0.00	0.
Z1	0.54	0.48	0.42	0.36	-0.48	-0.42	-0.36	0.00	0.00	0.00	1.00	-0.3
Z2	0.00	0.00	0.00	0.00	0.48	0.42	0.36	-0.42	-0.36	0.00	-0.36	1.
Z3	-0.54	-0.48	-0.42	-0.36	0.00	0.00	0.00	0.42	0.36	0.00	-0.36	-0.3

```
*
*Create a sample (n=250) from multinormal distribution, based
*on the previous correlation matrix (zero means assumed):
*
+MNSIMUL R13,*,TEST,250,0 / RND=rand(25052015)
*
*FILE SHOW TEST / browsing the data, observe the variable names
*                                  inherited from the matrix
*
*
*
```

Row: 62 / 1000 Column: 1 / 300 Path: e:/vanhaa/conf/CARME11/2011/

View from Survo R: graphics, typical work schemes



24th IWMS | Haikou, Hainan, China | May 2015 | K. Vehkalahti

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

View from Survo R: showing and manipulating matrices

Q (EXAMPLE): Survo R - General Editorial Computing Environment for Data Analysis - ver. 0.6.20

File Edit View Help

```
*
*Display the rotated factor matrix (AFACT.M):
*                        COLUMNS=SORT  POSDIR=1 SHADOWS=W,1,2,3,7       SUMS=2
+LOADM AFACT.M 12.12 CUR+1 / SORT=-TEST,0.6 LIMITS=-.5,-.3,.3,.6,1 WIDE=1
*A
*                 F1     F2     F3   Sumsqr
*A1             0.91   0.03   0.01   0.82
*A2             0.80  -0.03  -0.09   0.65
*A3             0.72   0.05  -0.05   0.53
*A4             0.66  -0.02   0.06   0.43
*B1            -0.00   0.81   0.04   0.66
*B2             0.09   0.73   0.00   0.55
*Z2            -0.07   0.64  -0.57   0.74
*B3            -0.04   0.61  -0.06   0.38
*Z1             0.61  -0.61   0.02   0.74
*C1             0.04  -0.07   0.70   0.49
*Z3            -0.56  -0.04   0.67   0.77
*C2            -0.04   0.03   0.59   0.35
*Z0             0.48   0.56   0.51   0.80
*Sumsqr         3.34   2.68   1.89
*
*Save the rotated factor matrix as matrix B. Also save the
*correlation matrix of the factors (RFACT.M) as matrix PHI,
*and compute PSI:
*
+MAT B!=AFACT.M
+MAT PHI=RFACT.M
+MAT NAME PHI AS \Phi      / internal name may be given explicitly
+MAT LOAD PHI CUR+4        / with orthogonal factors, PHI = I
+MAT PSI=DIAG(R-B*PHI*B') / PSI is diagonal in the factor model
+MAT I!=IDN(3,3)           / identity matrix I (to be used later)
*
```

Row: 155 / 1000   Column: 1 / 300   Path: e:/vanhaa/conf/CARME11/2011/

View from Survo R: matrix computations and comments

Q (EXAMPLE): Survo R - General Editorial Computing Environment for Data Analysis - ver. 0.6.20

File Edit View Help

```
*
*Create the factor score coefficient matrix A. Two methods:
*
+MAT IR=INV(R)
+MAT NAME IR AS R^{-1} / "LaTeX-like" internal name for INV(R)
+MAT A1=IR*B*PHI       / (*)
*
*This formula (*) is the shortest way to compute the factor score
*coefficients, but it involves inverting the matrix R. In practice,
*it would be preferable to apply Ledermann's (1938) trick, which is
*based on Schur complement and seems to be a simple special case of
*Woodbury inversion formula (see Puntanen & Styan 2005):
*
+MAT IP=INV(PSI)          / shorthand notation for INV(PSI)
+MAT NAME IP AS \Psi^{-1} / "LaTeX-like" name
+MAT GAMMA=B'*IP*B        / shorthand notation for B'\Psi^{-1}B
+MAT NAME GAMMA AS \Gamma / [assumed diagonal in ML estimation!]
*
*Compute A2, and compare the internal names of A1 and A2.
*Note that in A2 we have only inverted (diagonal) \Psi and
*I+\Gamma\Phi (3 x 3).
*
+MAT A2=IP*B*PHI*INV(I+GAMMA*PHI)
+MAT A2   / *A2~(\Psi^{-1})*B*\Phi*INV(I+\Gamma*\Phi) 13*3
+MAT A1   / *A1~(R^{-1})*B*\Phi 13*3
+MAT E=SUM(SUM(A1-A2,2)') / sum of the squared differences
+MAT LOAD E 1.1111111111111 CUR+2
*
*MATRIX E
*SUM(SUM((R^{-1})*B*\Phi-(\Psi^{-1})*B*\Phi*INV(I+\Gamma*\Phi),2)')
*///                 Sum2
*Sum      0.0000000000000
```

Row: 195 / 1000 Column: 1 / 300 Path: e:/vanhaa/conf/CARME11/2011/

1.1 Survo puzzle: playing with integer partitions

Survo puzzle is a basically simple, numerical puzzle that offers mathematical challenges both for novices and experts.

The task is to fill an $m \times n$ table by integers $1, 2, \ldots, mn$, so that **each number appears only once**, when the column and row sums are fixed. Here, $m = 3$ and $n = 4$.

	A	B	C	D	
1		**6**			30
2	**8**				18
3			**3**		30
	27	16	10	25	

Column C: $10 = 3 + 2 + 5$, **not** $3 + 6 + 1$, as 6 is already in use.

Finding solutions for a Survo puzzle is a **combinatorial problem**, where these **restricted integer partitions** play a crucial role. Survo puzzle was invented by prof. *Seppo Mustonen* in 2006 [6, 7].

Survo puzzles can be solved in many ways, but here we are especially interested in their relations to the **matrix theory**.

1.2 Solving Survo puzzles with matrices

We focus on a **stepwise method** for solving Survo puzzles **with matrices**, employing the matrix interpreter and other tools of Survo. Our method depends on **binary matrices** and matrix combinatorial products (**Hadamard**, **Kronecker**, **Khatri–Rao**) [2, 1].

Our method [11] consists of two stages:

1. **Constructing the code matrix** of the same size as the puzzle under study.
2. **Analysing the partitions** of the row sums and the column sums of the puzzle.

- We apply the **editorial approach** of Survo, which combines all the matrix and other data analytic operations needed.
- It provides an efficient way of **documenting and repeating** the steps, which is essential both in research and teaching.

2 Idea of the stepwise solving method

The minimal Survo puzzle is only 2×2, with an obvious solution:

	A	B	
1			3
2			7
	4	6	

	A	B	
1	1	2	3
2	3	4	7
	4	6	

In our method, we consider the partitions P as **binary vectors**:

$$P_1 = \{1, 2\} \quad \boldsymbol{P}_1 = \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix}$$

$$P_A = \{1, 3\} \quad \boldsymbol{P}_A = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}$$

etc. In general, these will be **matrices**, where the number of **rows** varies according to the number of partitions.

2.1 Combining the binary partitions

We consider the binary partitions of **one row** and **one column** at a time. Clearly, they have always exactly one common element. We combine the information by
1) multiplying the other one by 2 and 2) computing their sum.

Proceeding with row 1 and column A, we obtain a matrix (vector)

$$\boldsymbol{P}_{1A} = \boldsymbol{P}_1 + 2\boldsymbol{P}_A = \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 2 & 0 & 2 & 0 \end{bmatrix} = \begin{bmatrix} 3 & 1 & 2 & 0 \end{bmatrix}.$$

where we have **four codes** for the elements of the Survo puzzle:

- 3: the **common element** of the row 1 and column A
- 1: **own element** of the row 1
- 2: **own element** of the column A
- 0: **other element**, outside of the row 1 and column A

2.2 Coding scheme for the trivial puzzle

Placing these codes in a form of a Survo puzzle *(using gray color and no row sums, as these are codes, not elements)* we see *(on the left)* that in the trivial 2×2 example we already have a **unique code** for each element of the puzzle at this early stage:

	A	B
1	3	1
2	2	0

	A	B	C
1	3	1	1
2	2	0	0
3	2	0	0

	A	B	C
1	0	2	0
2	1	3	1
3	0	2	0

Exactly the same coding scheme works with larger puzzles, which can be seen above from the two 3×3 puzzles. There, the only unique code is 3, the common number of the row 1 and column A *(in the middle)* or similarly row 2 and column B *(on the right)*.

2.3 Coding scheme and uniqueness

In general, the codes and their counts in an $m \times n$ puzzle are:

- 3: 1 (the only) **common element** of the row and column
- 2: $(m-1)$ **own elements** of the column
- 1: $(n-1)$ **own elements** of the row
- 0: $(m-1)(n-1)$ **other elements** elsewhere in the puzzle

Clearly, $1 + (m-1) + (n-1) + (m-1)(n-1) = mn$.

The solution of a Survo puzzle is found immeaditely as soon as each element is represented by a unique code.

For puzzles larger than 2×2, the coding scheme must be expanded in order to retain the uniqueness of the codes.

We continue for a moment with the trivial example, although it is unnecessary for its solution (as we already found a unique code for each element). However, it is instructive for describing the stepwise method, takes less space, and reveals a unique coding scheme for a 3×3 Survo puzzle.

2.4 Recoding and Hadamard product

Proceeding similarly as with $\boldsymbol{P}_1$ and $\boldsymbol{P}_A$ we compute

$$\boldsymbol{P}_{2B} = \boldsymbol{P}_2 + 2\boldsymbol{P}_B = \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 2 & 0 & 2 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 1 & 3 \end{bmatrix}$$

so we now have two code matrices (of codes 3, 2, 1 and 0):

$$\boldsymbol{P}_{1A} = \begin{bmatrix} 3 & 1 & 2 & 0 \end{bmatrix} \text{ and } \boldsymbol{P}_{2B} = \begin{bmatrix} 0 & 2 & 1 & 3 \end{bmatrix}.$$

We need to **recode** these so that their **products** will be unique.

Applying simple transformations $2x + 1$ and 2^x, where x refers to the elements of the code matrices, we obtain new matrices

$$\boldsymbol{P}_{1A} = \begin{bmatrix} 7 & 3 & 5 & 1 \end{bmatrix} \text{ and } \boldsymbol{P}_{2B} = \begin{bmatrix} 1 & 4 & 2 & 8 \end{bmatrix},$$

which we multiply with each other using **Hadamard product:**

$$\boldsymbol{P}_{1A2B} = \boldsymbol{P}_{1A} \circ \boldsymbol{P}_{2B} = \begin{bmatrix} 7 & 12 & 10 & 8 \end{bmatrix}.$$

2.5 Recoded coding scheme

Recode 1:

- **7**: A1
- 3: row 1
- 5: column A
- 1: other

Recode 2:

- 1: other
- 4: column B
- 2: row 2
- **8**: B2

Product:

- **7**: A1
- **12**: B1
- **10**: A2
- **8**: B2

A 3×3 puzzle, where the non-unique codes 1,3,5 and 1,2,4 would be similarly repeated for the row 3 and column C, would be:

	A	B	C
1	7	3	3
2	5	1	1
3	5	1	1

	A	B	C
1	1	4	1
2	2	8	2
3	1	4	1

	A	B	C
1	7	12	3
2	10	8	2
3	5	4	1

The last puzzle *(on the right)* shows that we would already obtain a unique code for all elements of a 3×3 puzzle.

2.6 Constructing a 4×4 code matrix

To solve a 4×4 Survo puzzle we must still continue expanding the coding scheme, as the codes 1,2,3,4,5 become ambiguous again:

	A	B	C	D
1	7	12	3	3
2	10	8	2	2
3	5	4	1	1
4	5	4	1	1

In this phase, the elements A1, B1, A2, B2 already have unique codes 7, 12, 10, 8.

Continuing the recoding and multiplying (and choosing the codes carefully), we obtain unique codes for all elements.

We note that we can construct the code matrices **without any reference** to a particular Survo puzzle, because the coding schemes are general and based only on a) the codes 0,1,2,3, b) suitable recodings and c) their Hadamard products. **So let us now construct a** 4×4 **code matrix using Survo R**.

Creating a 4 $\times$ 4 code matrix (view from Survo R)

A view of the **editorial approach**, where ...

```
*/ACTIVATE +  /  This activates all rows that have a '+' in the control column.
+MAT X=ZER(4,4)  /  Create a 4 x 4 null matrix X with labels
+MAT X(0,1)="A"  /  corresponding to the Survo puzzle.
+MAT X(0,2)="B"
+MAT X(0,3)="C"
+MAT X(0,4)="D"
+MAT RLABELS NUM(1) TO X
+MAT X1=X  / X is the basis for each binary matrix.
+MAT XA=X  /                                         This is how the matrices
+MAT X2=X  / We need three pairs of rows and columns.   X1...XC look like before
+MAT XB=X  / Choose X1 and XA, X2 and XB, X3 and XC.    recoding and combining:
+MAT X3=X  /
+MAT XC=X  /                                         X1:        X2:        X3:
*                                                    1 1 1 1    0 0 0 0    0 0 0 0
*Mark the rows and the columns suitably with ones:   0 0 0 0    1 1 1 1    0 0 0 0
*(I# and J# refer to row and column indices)         0 0 0 0    0 0 0 0    1 1 1 1
*                                                    0 0 0 0    0 0 0 0    0 0 0 0
+MAT #TRANSFORM X1 BY F1 / F1=if(I#=1)then(1)else(0)
+MAT #TRANSFORM XA BY FA / FA=if(J#=1)then(1)else(0)   XA:        XB:        XC:
+MAT #TRANSFORM X2 BY F2 / F2=if(I#=2)then(1)else(0)   1 0 0 0    0 1 0 0    0 0 1 0
+MAT #TRANSFORM XB BY FB / FB=if(J#=2)then(1)else(0)   1 0 0 0    0 1 0 0    0 0 1 0
+MAT #TRANSFORM X3 BY F3 / F3=if(I#=3)then(1)else(0)   1 0 0 0    0 1 0 0    0 0 1 0
+MAT #TRANSFORM XC BY FC / FC=if(J#=3)then(1)else(0)   1 0 0 0    0 1 0 0    0 0 1 0
*
```

Creating a 4 × 4 code matrix (view from Survo R)

...the commands and comments are freely written and *activated*:

```
*Recode and combine (see the results on the right):     X1A:     X2B:     X3C:
+MAT X1A=X1+2*XA        / W=if(X#=0)then(01)else(a)     3 1 1 1  0 2 0 0  0 0 2 0
+MAT X2B=X2+2*XB        / a=if(X#=1)then(09)else(b)     2 0 0 0  1 3 1 1  0 0 2 0
+MAT X3C=X3+2*XC        / b=if(X#=2)then(11)else(c)     2 0 0 0  0 2 0 0  1 1 3 1
*U=2*X#+1 V=2^X#        / c=if(X#=3)then(13)else(X#)    2 0 0 0  0 2 0 0  0 0 2 0
*
*Further recoding (the rules U,V,W defined above):
+MAT #TRANSFORM X1A BY U / {1,3,5,7}                    The matrices X1A, X1B, X1C
+MAT #TRANSFORM X2B BY V / {1,2,4,8}                    after the both recodings:
+MAT #TRANSFORM X3C BY W / {1,9,11,13}
*                                                    /  X1A:     X2B:     X3C:
*Combine with Hadamard product X1A o X2B o X3C:      /  7 3 3 3  1 4 1 1  1 1 11 1
+MAT       X1A2B=#HADAMARD(X1A,X2B)                  /  5 1 1 1  2 8 2 2  1 1 11 1
+MAT     X1A2B3C=#HADAMARD(X1A2B,X3C)                /  5 1 1 1  1 4 1 1  9 9 13 9
+MAT NAME X1A2B3C AS 4x4                             /  5 1 1 1  1 4 1 1  1 1 11 1
+LOADM X1A2B3C 111 CUR+1  /  The final coding matrix
*4x4
*           A   B   C   D
*  1        7  12  33   3
*  2       10   8  22   2
*  3       45  36  13   9
*  4        5   4  11   1
```

Note: 3 row/column pairs are needed for coding a 4 × 4 puzzle.

3 Solving a demanding Survo puzzle

We turn to a more challenging problem of solving a particular 4×4 Survo puzzle, given by Seppo Mustonen in 2010 (see `http://www.survo.fi/puzzles/index.html#080210`).

Here is the puzzle *(on the left)* with its solution *(on the right)*:

	A	B	C	D	
1					27
2					12
3					52
4					45
	20	45	31	40	

	A	B	C	D	
1	3	10	6	8	27
2	1	5	2	4	12
3	9	16	12	15	52
4	7	14	11	13	45
	20	45	31	40	

It is clearly not a trivial puzzle. E.g., for the sum 27 (row 1) there are **61 possible partitions**.

3.1 Code matrix and the solution

Here, we repeat the solution *(on the right)* with the code matrix *(on the left)* being the one created earlier:

	A	B	C	D
1	7	12	33	3
2	10	8	22	2
3	45	36	13	9
4	5	4	11	1

	A	B	C	D	
1	3	10	6	8	27
2	1	5	2	4	12
3	9	16	12	15	52
4	7	14	11	13	45
	20	45	31	40	

It turns out that the solution vector that gives the 16 codes and their positions, is **one of 75 million alternatives**.

With more difficult puzzles, the number of alternatives will grow fast, and may be too large to be managed in practice. (In order to make a puzzle a bit easier, some of the numbers may be readily given – **as in our 6 x 6 case**.)

Creating a binary partition matrix (view from Survo R)

Listing the 61 partitions of 27 consisting of 4 parts:

```
*COMB ROW1 TO ROW1.TXT / ROW1=PARTITIONS,27,4 MIN=1 MAX=16 DISTINCT=1
*
*Partitions 4 of 27: N[ROW1]=61
*1 2 8 16
*1 2 9 15
*1 2 10 14
*1 2 11 13
*1 3 7 16
*1 3 8 15
*1 3 9 14
*1 3 10 13
*1 3 11 12
*[...] (44 lines omitted)
*3 6 8 10
*3 7 8 9
*4 5 6 12
*4 5 7 11
*4 5 8 10
*4 6 7 10
*4 6 8 9
*5 6 7 9
```

Creating a binary partition matrix (view from Survo R)

Saving the 61 partitions as a Survo data file of 16 variables:

```
*FILE SAVE ROW1.TXT TO ROW1
*
*  X1  X2  X3  X4  X5  X6  X7  X8  X9 X10 X11 X12 X13 X14 X15 X16
*   1   2   8  16   -   -   -   -   -   -   -   -   -   -   -   -
*   1   2   9  15   -   -   -   -   -   -   -   -   -   -   -   -
*   1   2  10  14   -   -   -   -   -   -   -   -   -   -   -   -
*   1   2  11  13   -   -   -   -   -   -   -   -   -   -   -   -
*   1   3   7  16   -   -   -   -   -   -   -   -   -   -   -   -
*   1   3   8  15   -   -   -   -   -   -   -   -   -   -   -   -
*   1   3   9  14   -   -   -   -   -   -   -   -   -   -   -   -
*   1   3  10  13   -   -   -   -   -   -   -   -   -   -   -   -
*   1   3  11  12   -   -   -   -   -   -   -   -   -   -   -   -
*[...] (44 lines omitted)
*   3   6   8  10   -   -   -   -   -   -   -   -   -   -   -   -
*   3   7   8   9   -   -   -   -   -   -   -   -   -   -   -   -
*   4   5   6  12   -   -   -   -   -   -   -   -   -   -   -   -
*   4   5   7  11   -   -   -   -   -   -   -   -   -   -   -   -
*   4   5   8  10   -   -   -   -   -   -   -   -   -   -   -   -
*   4   6   7  10   -   -   -   -   -   -   -   -   -   -   -   -
*   4   6   8   9   -   -   -   -   -   -   -   -   -   -   -   -
*   5   6   7   9   -   -   -   -   -   -   -   -   -   -   -   -
```

Creating a binary partition matrix (view from Survo R)

Moving the numbers to their right positions in the data:

```
*XALL ROW1,X1,16
*
*  X1  X2  X3  X4  X5  X6  X7  X8  X9 X10 X11 X12 X13 X14 X15 X16
*   1   2   -   -   -   -   -   8   -   -   -   -   -   -   -  16
*   1   2   -   -   -   -   -   -   9   -   -   -   -   -  15   -
*   1   2   -   -   -   -   -   -   -  10   -   -   -  14   -   -
*   1   2   -   -   -   -   -   -   -   -  11   -  13   -   -   -
*   1   -   3   -   -   -   7   -   -   -   -   -   -   -   -  16
*   1   -   3   -   -   -   -   8   -   -   -   -   -   -  15   -
*   1   -   3   -   -   -   -   -   9   -   -   -   -  14   -   -
*   1   -   3   -   -   -   -   -   -  10   -   -  13   -   -   -
*   1   -   3   -   -   -   -   -   -   -  11  12   -   -   -   -
*[...] (44 lines omitted)
*   -   -   3   -   -   6   -   8   -  10   -   -   -   -   -   -
*   -   -   3   -   -   -   7   8   9   -   -   -   -   -   -   -
*   -   -   -   4   5   6   -   -   -   -   -  12   -   -   -   -
*   -   -   -   4   5   -   7   -   -   -  11   -   -   -   -   -
*   -   -   -   4   5   -   -   8   -  10   -   -   -   -   -   -
*   -   -   -   4   -   6   7   -   -  10   -   -   -   -   -   -
*   -   -   -   4   -   6   -   8   9   -   -   -   -   -   -   -
*   -   -   -   -   5   6   7   -   9   -   -   -   -   -   -   -
```

Creating a binary partition matrix (view from Survo R)

Transforming the variables of the data to binary form:

```
*TRANSFORM ROW1 BY if(X=MISSING)then(0)else(1)
*
*  X1  X2  X3  X4  X5  X6  X7  X8  X9 X10 X11 X12 X13 X14 X15 X16
*   1   1   0   0   0   0   0   1   0   0   0   0   0   0   0   1
*   1   1   0   0   0   0   0   0   1   0   0   0   0   0   1   0
*   1   1   0   0   0   0   0   0   0   1   0   0   0   1   0   0
*   1   1   0   0   0   0   0   0   0   0   1   0   1   0   0   0
*   1   0   1   0   0   0   1   0   0   0   0   0   0   0   0   1
*   1   0   1   0   0   0   0   1   0   0   0   0   0   0   1   0
*   1   0   1   0   0   0   0   0   1   0   0   0   0   1   0   0
*   1   0   1   0   0   0   0   0   0   1   0   0   1   0   0   0
*   1   0   1   0   0   0   0   0   0   0   1   1   0   0   0   0
*[...] (44 lines omitted)
*   0   0   1   0   0   1   0   1   0   1   0   0   0   0   0   0
*   0   0   1   0   0   0   1   1   1   0   0   0   0   0   0   0
*   0   0   0   1   1   1   0   0   0   0   0   1   0   0   0   0
*   0   0   0   1   1   0   1   0   0   0   1   0   0   0   0   0
*   0   0   0   1   1   0   0   1   0   1   0   0   0   0   0   0
*   0   0   0   1   0   1   1   0   0   1   0   0   0   0   0   0
*   0   0   0   1   0   1   0   1   1   0   0   0   0   0   0   0
*   0   0   0   0   1   1   1   0   1   0   0   0   0   0   0   0
```

3.2 Partitions, matrices and Kronecker products

Recall that we need 3 row/column pairs to solve a 4×4 puzzle.

Applying the similar recoding as earlier, we will obtain the matrices $\boldsymbol{P}_{1A}$, $\boldsymbol{P}_{2B}$ and $\boldsymbol{P}_{3C}$. Now, the number of the partitions may vary freely, so we have to use Kronecker products of type

$$\boldsymbol{P}_{1A} = \left[(\boldsymbol{P}_1 \otimes \mathbf{1}\mathbf{1}') + (\mathbf{1}\mathbf{1}' \otimes 2\boldsymbol{P}_A)\right]\left[(\boldsymbol{I} \otimes \mathbf{1}) \circ (\mathbf{1} \otimes \boldsymbol{I})\right]$$

in order to make the intermediate matrices compatible. The latter term (which includes the Hadamard product) removes the unnecessary combinations of the columns by selecting the *principal columns* of the Kronecker product.

Here, the three rows/columns have 61, 23, 2, 38, 9 and 79 partitions, so the matrices $\boldsymbol{P}_{1A}$, $\boldsymbol{P}_{2B}$ and $\boldsymbol{P}_{3C}$ will consist of $61 \cdot 23 = 1403$, $2 \cdot 38 = 76$ and $9 \cdot 79 = 711$ rows, respectively.

The dimensions are still quite reasonable here, because the puzzle is not overly difficult. It has a ”weak point”, as the row 2 has only two possible partitions: $12 = 1 + 2 + 3 + 6 = 1 + 2 + 4 + 5$.

3.3 Recoded matrices and Khatri–Rao products

The solution will then be found using the three recoded matrices $\boldsymbol{P}_{1A}$, $\boldsymbol{P}_{2B}$ and $\boldsymbol{P}_{3C}$, by computing their Khatri–Rao products [2]

$$\boldsymbol{P}_{1A2B3C} = \boldsymbol{P}_{1A} \odot \boldsymbol{P}_{2B} \odot \boldsymbol{P}_{3C},$$

which will have $61 \cdot 23 \cdot 2 \cdot 38 \cdot 76 \cdot 9 \cdot 79 =$ **75 812 508** rows. Exactly **one** of these rows gives the codes for the solution.

The final challenge is to find that one row!

As we are not interested in the products of the columns (they represent the *mn* elements of the puzzle), the Khatri–Rao product we apply here could be called a "row-wise Kronecker product". It is also referred to as the "Khatri–Rao of first kind product" by [1].

3.4 Finding the codes of the solution

To avoid too large dimensions, we do *not* compute only one huge Khatri–Rao product. Instead, **we work stepwise**, and after each of the Kronecker and/or Khatri–Rao products, we **remove all illogical combinations**, that is, those products of codes that do not meet the conditions cumulated so far.

At the final stage, we will have exactly *mn* unique codes, and only one of the remaining rows will include all of them.

These steps are executed in Survo through several phases:
1) moving the matrices to data files,
2) applying the conditions, and
3) saving the selected rows back in the matrix form.

Finding the codes of the solution (view from Survo R)

The final phase, where the solution is found:

```
+MAT Q1A2B3C=#RAO_KHATRI(Q1A2B,Q3C)
+MAT DIM Q1A2B3C /* rowQ1A2B3C=1215 colQ1A2B3C=16
*
+FILE SAVE MAT Q1A2B3C TO Q1A2B3C /  save the matrix to a data file
+FILE MASK Q1A2B3C,CASE,1,-       /  mask the case ID out of computations
*Compute the number of each of the codes for all observations of the data:
+VARSTAT Q1A2B3C / VARSTAT=N1:1,N2:1,N3:1,N4:1,N5:1,N7:1,N8:1,N10:1,N12:1,&
*                          N9:1,N11:1,N13:1,N22:1,N33:1,N36:1,N45:1
*  N1=#VAL,1    N2=#VAL,2    N3=#VAL,3    N4=#VAL,4    N5=#VAL,5
*  N7=#VAL,7    N8=#VAL,8    N10=#VAL,10  N12=#VAL,12
*  N9=#VAL,9    N11=#VAL,11  N13=#VAL,13  N22=#VAL,22
*  N33=#VAL,33  N36=#VAL,36  N45=#VAL,45
*.........................
+FILE MASK Q1A2B3C,CASE,1,A       /  put the case ID back
+FILE MASK Q1A2B3C,N1,1,-...      /  mask the numbering variables out
*Save the reduced data back in to a matrix - applying the 16 conditions:
+MAT SAVE DATA Q1A2B3C TO Q1A2B3C / SELECT=A*B*C*D*E*F*G*H*I*J*K*L*M*N*O*P
*  A=N1,1  B=N2,1 C=N3,1  D=N4,1  E=N5,1  F=N7,1  G=N8,1  H=N10,1
*  I=N12,1 J=N9,1 K=N11,1 L=N13,1 M=N22,1 N=N33,1 O=N36,1 P=N45,1
+MAT DIM Q1A2B3C /* rowQ1A2B3C=1 colQ1A2B3C=16
```

Only one row remains! Does it really give the solution?

Checking the validity of the solution (view from Survo R)

We check by matching the codes with the 4×4 code matrix:

```
+LOADM X1A2B3C 111 CUR+1           /  the coding matrix (created earlier)
*4x4
*           A   B   C   D
*  1        7  12  33   3
*  2       10   8  22   2
*  3       45  36  13   9
*  4        5   4  11   1
*
+MAT V1A2B3C=VEC(X1A2B3C)          /  vectorize the coding matrix
+MAT V1A2B3C(0,1)="code"           /  set a column label
+MAT LOAD V1A2B3C' 111 CUR+1       /  display as a row vector
*MATRIX V1A2B3C'
*VEC(4x4)'
*///        1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
*code       7  10  45   5  12   8  36   4  33  22  13  11   3   2   9   1
*
*Do the same for the obtained code vector of the solution:
+MAT CLABELS NUM(1) TO Q1A2B3C     /  set column labels 1,2,3,...,16
+MAT Q1A2B3C(1,0)="code"           /  set a row label
+MAT P1A2B3C=Q1A2B3C'              /  transpose to a column vector
+MAT LOAD P1A2B3C' 111 CUR+1       /  display as a row vector
*MATRIX P1A2B3C'
*///        1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
*code      10  22   7   2   8  33   5   3  45  12  11  13   1   4   9  36
*
```

Checking the validity of the solution (view from Survo R)


```
+FILE SAVE MAT V1A2B3C TO CODES    / save both vectors to new data files,
+FILE SAVE MAT P1A2B3C TO SOLUTION / both will have CASE = 1,2,3,...,16
*..............................
 Match the data sets and get the numbers corresponding to the codes:
+FILE COPY SOLUTION TO CODES       /  VARS=CASE MATCH=code MODE=2
*.......................................................................
+VAR code=CASE TO CODES            /  replace the codes by the numbers
+MAT SAVE DATA CODES TO CODES      /  save the data file into a vector
+MAT CODES=VEC(CODES,4)            /  change the vector to a 4x4 matrix
+MAT RLABELS FROM X TO CODES       /  take labels from the matrix X that
+MAT CLABELS FROM X TO CODES       /   was the basis of the coding matrix
+MAT NAME CODES AS Solution        /  this is the solution through codes
+LOADM CODES 111 CUR+2 / SUMS=1    /  show the row and column sums, too
*
*Solution
*            A   B   C   D  Sum
*  1         3  10   6   8  27
*  2         1   5   2   4  12
*  3         9  16  12  15  52
*  4         7  14  11  13  45
*Sum        20  45  31  40
```

Indeed, we have found the solution for the Survo puzzle.

4 Conclusions

Survo R and teaching matrices within statistics:

- interactive **"learning lab"** for students' experiments
- documented **work schemes**: **create, repeat, modify, learn**
- working **step-by-step**, checking any **intermediate results**
- **self-documenting** commands with **free-form** documentation
- **good connections** with other tools of **data analysis** etc.

Survo R is freely available, see: www.survo.fi/muste

References (Thank you for your attention!)

[1] Al Zhour, Z. and Kiliçman, A. (2007). *Some new connections between matrix products for partitioned and non-partitioned matrices.* Computers and Mathematics with Applications, **54**, 763–784.

[2] Khatri, C. G. and Rao, C. R. (1968) *Solutions to some functional equations and their applications to characterization of probability distributions.* Sankhyā, Ser. A, **30**, 167–180.

[3] Mustonen, S. (1981) *On Interactive Statistical Data Processing.* Scandinavian Journal of Statistics, **8**, 129–136.

[4] Mustonen, S. (1992) *Survo – An Integrated Environment for Statistical Computing and Related Areas.* Survo Systems, Helsinki. http://www.survo.fi/books/1992/Survo_Book_1992_with_comments.pdf

[5] Mustonen, S. (1999) *Matrix computations in Survo.* Proceedings of the 8th International Workshop on Matrices and Statistics (IWMS), Department of Mathematical Sciences, University of Tampere, Finland. http://www.helsinki.fi/survo/matrix99.html

[6] Mustonen, S. (2006) *On certain cross sum puzzles.* http://www.survo.fi/papers/puzzles.pdf

[7] Mustonen, S. (2007) *Enumeration of uniquely solvable open Survo puzzles.* http://www.survo.fi/papers/enum_survo_puzzles.pdf

[8] Sund, R. (2011) *Muste – the R implementation of Survo.* Yearbook of the Finnish Statistical Society 2010, pp. 133–146. http://www.survo.fi/muste/publications/sund2011_muste_yearbook.pdf

[9] Sund, R., Vehkalahti, K. and Mustonen, S. (2012). *Muste – editorial computing environment within R.* Proceedings of COMPSTAT 2012, 20th International Conference on Computational Statistics, 27–31 August 2012, Limassol, Cyprus. http://www.survo.fi/muste/publications/sundetal2012.pdf.

[10] Vehkalahti, K. (2005) *Leaving useful traces when working with matrices.* Research Letters in the Information and Mathematical Sciences, Vol. 8, 143–154. Proceedings of the 14th International Workshop on Matrices and Statistics (IWMS), ed. by Paul S. P. Cowpertwait, Massey University, Auckland, New Zealand, pp. 143–154.

[11] Vehkalahti, K. and Sund, R. (2014). Solving Survo puzzles using matrix combinatorial products, *Journal of Statistical Computation and Simulation.* http://www.tandfonline.com/doi/full/10.1080/00949655.2014.899363