Survo is an interactive computing environment for processing text and numerical data. Its current version SURVO MM works in Windows, but it has emerged during a 40-year-old history that includes, for example, the statistical programming language SURVO 66, and SURVO 76, one of the first interactive statistical packages. These various generations of the Survo system have played an essential part in the tradition of statistical computing in Finland.

SURVO CROSSINGS

Seppo Mustonen
seppo.mustonen@survo.fi

The fundamental concept of Survo is the editorial approach, which surpassed the menu-based interface of SURVO 76 in 1979, and still establishes the heart of the current SURVO MM. In this approach, all functions of the system are controlled by a specific text editor that distributes the tasks between numerous independent program modules. Thus SURVO MM is not a single huge program but a large family of small program modules.

The Survo editor is the center of all the activities, and the system, as a whole, is a general environment for various tasks related not only to statistical analysis and computing. In fact, functions of Survo have been extended to many areas that support statistical research and teaching of statistics.

Essentially, using Survo is like working with a combined word processor and spreadsheet program with enhanced capabilities in various directions. Thus in Survo, one can maintain the whole statistical research process so that each step of that process is automatically documented. For these purposes, Survo includes functions for data input and screening, general data management, statistical graphics and analysis, matrix computations, making reports in printable form, desktop publishing etc.

Survo also includes a powerful macro language for making expert applications by combining automatically and conditionally ready-made functions of the system. The same technique can be used for creating teaching programs on any topic, for example, statistical methods.

Survo puzzles

Since the editorial approach of Survo provides plenty of means for the user to express ideas and to organize computational experiments in a personal way, it gives scope for imagination and inventiveness.

Survo (cross sum) puzzles provide an example of these features since without the experience offered by working with Survo, I would have hardly come across the idea of such a number game.

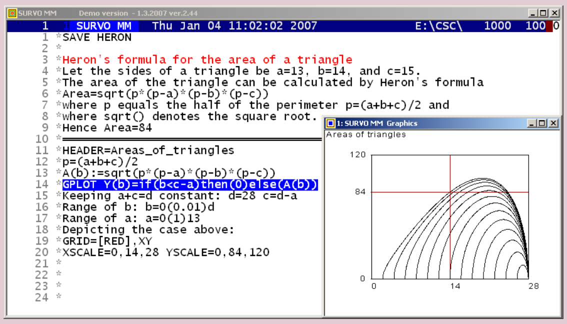
In a Survo puzzle the task is to fill an $m \cdot n$ table by integers $1,2,\ldots,mn$, so that each of these numbers appears only once, and their row and column sums are equal to integers given on the bottom and the right side of the table. Often some of the integers are given readily in the table in order to guarantee uniqueness of the solution and/or for making the task easier.

Survo puzzles have been eagerly studied and solved among the active Survo users and their friends beginning from the spring of 2006. In harder cases, various computational features of the Survo system have been helpful. Some people have found Survo puzzles more interesting and more challenging than the popular Sudoku or Kakuro puzzles.

Although it is fairly easy to make algorithms for solving any Survo puzzle systematically, in many cases heuristic treatment of the problem at hand may lead to a more elegant solution. It is obvious that

Heron

The user can type text, formulas, commands, *etc.* in the edit field. In this example, a typical computation scheme is presented for computing the area of a triangle. The essential ingredients are given as formulas naturally within the text commenting the task. When Area= on the line 9 is activated, Survo identifies and evaluates the pertinent formulas and immediately writes the result (84) in its place.



```
    1  SURVO MM   Thu Jan 04 11:02:02 2007              E:\CSC\    1000   1000
 1 *SAVE HERON
 2 *
 3 *Heron's formula for the area of a triangle
 4 *Let the sides of a triangle be a=13, b=14, and c=15.
 5 *The area of the triangle can be calculated by Heron's formula
 6 *Area=sqrt(p*(p-a)*(p-b)*(p-c))
 7 *where p equals the half of the perimeter p=(a+b+c)/2 and
 8 *where sqrt() denotes the square root.
 9 *Hence Area=84
10 *=============================================
11 *HEADER=Areas_of_triangles
12 *p=(a+b+c)/2
13 *A(b):=sqrt(p*(p-a)*(p-b)*(p-c))
14 *GPLOT Y(b)=if(b<c-a)then(0)else(A(b))
15 *Keeping a+c=d constant: d=28 c=d-a
16 *Range of b: b=0(0.01)d
17 *Range of a: a=0(1)13
18 *Depicting the case above:
19 *GRID=[RED],XY
20 *XSCALE=0,14,28 YSCALE=0,84,120
21 *
22 *
23 *
24 *
```

All the work in Survo is controlled from an edit field having often thousands of lines and hundreds of columns. A part of it is always visible in the main window.

In the second part, a family of curves is plotted by activating a GPLOT command on the line 14. Now various specifications and comments are given freely around the command for adjusting the details. The graph itself appears in a separate window located, in this case, upon the main window.

Even this small example indicates one major difference between Survo and typical mathematical software (like Mathematica or Matlab). In Survo, the steps of computing can be presented as free format descriptions and no strict order is required. This form of literal or self-documenting programming has already been introduced in Survo in the late seventies.

A simple example

	6			30
8				28
		3		30
27	16	10	25	

The table has to be filled by integers from 1 to 12 so that each of these numbers appears only once and their row and column sums are equal to integers given on the bottom and the right side of the table.

Solution step by step

	6			30
8				28
		3		30
27	16	10	25	

	6			30
8	1			28
	9	3		30
27	16	10	25	

12	6			30
8	1			28
7	9	3		30
27	16	10	25	

12	6			30
8	1			28
7	9	3	11	30
27	16	10	25	

12	6	2	10	30
8	1			28
7	9	3	11	30
27	16	10	25	

12	6	2	10	30
8	1	5	4	28
7	9	3	11	30
27	16	10	25	

there are no simple general rules for solving these puzzles manually as in Sudoku, for example. Of course, various computational tools, like routines for finding restricted partitions of integers (for example, the COMB program in Survo) are helpful in more demanding problems.

Many people seem to tackle Survo puzzles by a mixture of rational reasoning and guesswork. It is usually a much more demanding task to solve these puzzles, so that besides reaching the solution, one has also proved that there are no other solutions.

Basic information and guided solutions may be found in my expository paper

www.survo.fi/papers/puzzles.pdf and general information in

www.survo.fi/puzzles.

The open Survo puzzles (where only the row and column sums are given) call for special attention. They are like statistical tables of frequencies with given marginal frequencies but all cell frequencies missing. In general, the marginal distributions have very little to say about the joint distribution of two statistical variables.

The marginals practically never define cell frequencies uniquely. Surprisingly, in open Survo puzzles for any dimensions m,n there are plenty of cases where we have a unique solution for the "cell frequencies". This is due to the very strict condition that the cells have to be occupied by numbers $1,2,...,mn$ in some order.

However, open Survo puzzles with a unique solution comprise a minority among all tables with valid marginal sums. When the dimensions m and n grow, the number $S(m,n)$ of essentially different and uniquely solvable Survo puzzles seem to grow steadily but the proportion of them seem to decrease (to zero as a limit, I presume).

It is known, for example, that $S(2,2)=1$, $S(3,3)=38$, and, $S(4,4)=5327$, but already computation of $S(5,5)$ may be a very hard task.

Music, statistics and data processing

Survo has an outstandingly long history, since the first Survo was in use already in the 1960s. The father of Survo, professor emeritus **Seppo Mustonen** from the University of Helsinki, Dept. of Mathematics and Statistics, remembers the first moments of Survo well:

"It was a hot summer day in 1962. **Martti Tienari** and I were then working in the Electronics Department of the Finnish Cable Company (predecessor of the current Nokia) and now attending the second IFIP Conference in Munich. We had an idea of a general programming language for processing of statistical data and presented it to Professor **Olli Lokki**. After a short discussion he readily supported our idea. Because I was for responsible of making statistical software in our company, it was my task to launch the project."

Automatic data processing had started at the Finnish universities at the beginning of the 60s; the University of Helsinki had received its first computer, ESKO, in 1960, and by the middle of the decade, the universities were using the IBM 1620, IBM 1130, and Elliott 803 machines. The first version SURVO 66 was created for Elliott 803 by a work group led by Seppo Mustonen who actually was responsible for the basic solution and most of the program code. After a year or two Mustonen became professor of Statistics at the University of Helsinki and Tienari became Professor of Computer Science in the same university.

About working with different kinds of computers, Mustonen says, "the first computers in the sixties were exciting. It was inspirational to work hands-on with the computers." A little later the universities got their data processing centers, and everything had to go through these: "You sent in a pack of punched cards, and a day or two later got the results of your job." This was not a satisfying way of working, "I was getting a little desperate, but perked up in 1975, when our department bought a Wang 2200 minicomputer."

The first interactive version SURVO 76 was developed by Mustonen for this minicomputer. "When demonstrating it, some specialists came to claim that I was merely playing with a computer, interactivity will never be needed in statistical computing and data processing! Of course, I didn't agree, and told them that, on the contrary, now is the right time to learn the rules how to communicate with a computer. At the beginning, working with SURVO 76 was like a conversation, the program posed questions and the user gave answers. One of the leading principles was that the program remembered earlier answers and offered them as defaults for the next time. However, the new idea of the editorial approach superseded this conversational working mode since I felt immediately that it was still more interactive and efficient way to get along with the computer.

"Surprisingly, the idea arose in connection with a musical application. I wanted to develop a program that would accurately and easily transcribe my son Olli's hand-written violin compositions into a printable form on the Wang minicomputer that was equipped with a drum plotter," Mustonen reminisces, "As a minor part of this task I also had to program a new editor and only after this experiment I realized that the same editor can be extended to computational and statistical applications."

Mustonen says that you can find a flash demo of the idea of the editorial approach at http://www.survo.fi/flash/e_idea.html.

Mustonen has a soft spot for music, and music gave him his inspiration for the editor in Survo. Says Mustonen, "You never know, *a priori*, what interesting things can emerge, when alien life forms meet."

Unconventional combinations, like music and statistics, can be fruitful, and Mustonen tries to encourage statisticians to fan out into other sciences as well.

Leena Jukka

Three Survo puzzles for the reader

			11
			14
			20
7	17	21	

Degree of difficulty = 45

					30
		9			41
			13		49
9	13	26	31	41	

Degree of difficulty = 275

				24
				29
				38
				45
11	28	42	55	

Degree of difficulty = 950

For those who have not time and patience to solve Survo Puzzles, quick games, for example, as a Java applet are available in http://www.survo.fi/java/quick5x5.html.

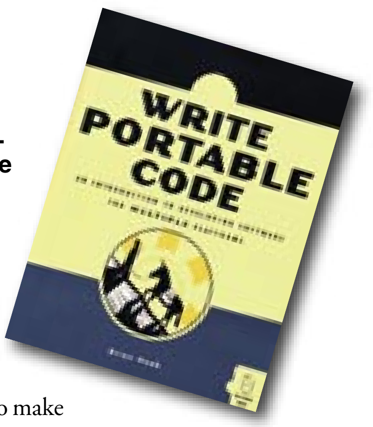
As quick games, Survo Puzzles give challenges of another kind and are excellent practice for logical reasoning, mental arithmetics, and even for recognition of musical intervals

Imprints

Juha Haataja

Moving programs

Write Portable Code—An Introduction to Developing Software for Multiple Platforms (Brian Hook; No Starch Press, 2005)



Brian Hook's guide to writing portable code motivates developers to cross-platform development, and answers many basic questions about code portability. Why not try to make code portable from the beginning? And what should you first do when you need to make a legacy code portable?

So, why write portable code? First of all, you reach a larger market, and avoid locking to a single platform. Also, portable code tends to be more robust, because sloppy assumptions and lazy coding habits tend to reveal themselves.

Moving a program from one platform to another may be a daunting task. Also, platforms themselves evolve. This forces programmers to port their codes even within a single platform. But if you want to remain competitive, you need to use new possibilities such as 64-bit processors.

As Hook discusses in the book, it is a surprise that there are almost no books on portable coding, even though cross-platform development is increasingly common. Software is ported from Linux to Windows, from Windows to Mac OS X, and from desktop systems to pda-type devices, or from desktops to supercomputers.

Hook's book is an excellent introductory review of the subject, although on most topics it covers just the basics. On the other hand, the subject by itself is demanding both in theory and practice, so the reader should know the practice of software development on the intermediate or advanced level.

The book consists of 18 short chapters, and focuses on cross-platform C/C++ development. There are extensive examples of portability issues with C and C++. In addition, scripting languages like Javascript and Python are discussed.

How do you edit your source code on multiple platforms? What about source control systems or portable build systems? It is a bit surprising to see how many different topics affect code portability.

Many aspects of hardware may limit portability, such as byte ordering, address space, and floating point arithmetic. Many compilers have "features" that may pose problems in porting codes.

Writing portable user interfaces poses challenges by itself. Should one use ready-made portable libraries, or code everything from scratch? Also, data portability, file management, and scalable algorithms pose challenges for the programmer.

Hook's approach to porting code is practical. He describes many case examples and concrete tools.

The practicality has a drawback, though. Within a few years many of the examples may not be relevant any more. However, the book is highly useful for a few years from now on.

The writer has personal knowledge about cross-platform development, and also shares his expertise with the readers. The practical advice, clear writing, and a good selection of essential topics make this book a treasure for anyone interested in software development.