Nord-DATA-68

Helsingfors, Otnäs 6—8. 6. 1968
Helsinki, Otaniemi 6—8. 6. 1968

Särtryck av konferensens föredragspublikation

Hakusana: SURVO

A STATISTICAL PROGRAMMING LANGUAGE SURVO 66

Timo Alanko, Seppo Mustonen, Martti Tienari
Helsingfors Universitet
Helsingfors

Introduction

The authors have co-operated since 1960 in programming statistical applications for electronic digital computers. We have worked through the usual stages of system development in this application field. We defined standard statistical programs for different methods requiring extensive computation: correlation, regression, factor analysis and other multivariate methods. We soon noticed the value of a common data standard for different programs, because many statistical problems required the application of different methods, often in an unpredictable sequence. It is, of cource, of great practical importance to be able to keypunch the data material just once although it is subsequently used in different statistical analysis programs. In the same way the intermediate results e.g. correlation matrices should be in a form conforming to the input requirements of the analysis programs. We also found it practical to compute different elementary statistical results e.g. means, variances and cross tabulations, of the data keypunched mainly for the subsequent heavy computer analysis. In this way we came to an integrated statistical program library for our computer, an Elliott 803B with 8196 words of 39 bit core memory. Similar integrated libraries, statistical program packages, have been reported for many computers e.g. IBM 7090 [1], [2] and IBM 1401 [4].

In the course of the extensive statistical computing service which has been maintained using the integrated statistical program library, we have been observing the behaviour of the scientists using computer services for their statistical research. The working habits of these scientists were changing. They dared to collect much more extensive data material, more attributes and more items than earlier. During the time of manual statistical computations the statisticians were close to the data. A decision to perform some statistical analysis came after careful reasoning. Now, the scientist -- once he has decided to make use of the computer -- is usually more careless. He often experiments with different analysis methods, sometimes even without any clear a priori hypothesis. The scientist is also often unable to look carefully at his data. The computer service must therefore provide for him thorough quality control, cross tabulation and plotting of the data. In manual computation one uses every conceivable trick and short-cut to avoid extensive straightforward computations. A computer user is tempted to exactly the opposite: a straightforward standard computation is no problem, whereas any fresh, simple idea might lead to slow and costly special programming or to manual computing. It is now wise to guide statistical work in such a way that one can make use of the standard statistical programs.

The observations presented above lead us to aim for radically more flexible statistical programs. There exist, however, some factors which limit the possibilities of an integrated chain of statistical standard programs. Added flexibility usually means added complexity of use; we would hope that the scientist need not be a computer specialist to be able to define in computer language his processing requirements. Many problems are left to the user of any integrated statistical program library with flexible processing facilities. The user is expected to furnish parameters for the programs in the statistical package. He must consider and fit together the different data structures used in the package, and required in his research. It is very difficult to provide adequate mnemonic labelling of different variables and results. A statistical package is usually unable to perform any parallel processing: each program handles the data completely before it is able to deliver control to the next program.

In the end, we felt that the only way to achieve drastically more flexibility in the statistical reseach process was to create a statistical language, which would be comprehensible to any scientist familiar with usual statistical methods. A specific design goal of the system SURVO 66 was to obviate any methods consulting staff between the scientist and the computer.

The process of implementing our ideas proceeded through several stages. In 1964, the first system design named SURVO 64 was elaborated. It was subsequently implemented in a reduced form which we called simply a generalized cross-tabulating system. The following stage was a plan called SURVO 65, which we could not agree to be worth the cost of implementing. Finally, a new design SURVO 66 emerged and was implemented. The system was released in December 1967 for computing service. The handbook of this general statistical data analysis system is published in Finnish [3]. The system is now in use at several university computer centers in Finland.

Basic principles of the language SURVO 66

SURVO 66 is a programming system tailored to the data processing requirements of elementary statistics. The data exposed to an analysis must conform to a special data standard. We presume that the data consists of numbers arranged in a data matrix. A row of the matrix, data vector, represents the data from an object under observation: a person, a unit of sample, a product item, a single experiment. The attributes of the objects are variables: numbers characterizing the object, test scores, replies to questions, measurements. Most statistical data materials can be organized according to this standard. To this end, any qualitative information must be coded in a numerical form; missing observations of attributes are coded as out-of-range numbers. If no symbolic names have been given to the variables, the system calls them X1, X2, X3, ... , XM.

The tasks which a SURVO program is able to do are:

1. Quality control of the data (range of variables, interrelationships of variables),
2. transforming the data,
3. estimation of basic statistical parameters: means, medians, standard deviations, fractiles, correlations,
4. frequencies and cross tabulations,
5. performing tests of significance: t-test, χ^2 - test,
6. simple statistical analysis: analysis of variance, regression analysis.

A task can be carried out selectively: the operations are applied only to the data vectors conforming to a predetermined condition. This feature allows, in effect, even handling of overlapping groups of data and comparing different data groups in a single computer run. All the objects referred to: variables, tables, correlation matrices, classification scales, classes, conditions etc., can be given alphanumeric names. This is in order to make the SURVO program easier to read. This practice also enables the SURVO system to label the result quantities in an easily comprehensible way.

For the sake of effeciency the SURVO 66 system applies a sort of parallel processing. The data material is usually too extensive to be stored in the fast random access memory. It must be held on an external data medium: magnetic tape, punched cards or paper tape. For any standard packaged computation of elementary statistics it is sufficient to have the data available one vector at a time. The cost of input makes many small statistical computations uneconomical, if they must be processed by independent programs. Therefore in the SURVO 66 system the data is exposed to several parallel statistical operations within one data input cycle.

As an introductory example we give a program which computes the means of 20 variables from 100 observations. The description of this job in SURVO language is simply:

```
M@20 N@100
MEAN@X1-X20
END@
```

The SURVO program is punched on paper tape and the data on paper tape or punched cards.

The run of a SURVO program can be divided into three stages: T1: translation of SURVO program, T2: input of the data under control of the translated program, T3: final computations on cumulated tables and output of results. During T1 the SURVO system program reads, checks and stores the program. Storage space is allocated and sum locations are cleared. The second stage, T2, consists of reading the data. The dimensions of the data matrix are read first, as well as a set of parameters describing the details of data format. While the data matrix is being read, just one observation vector is in the fast memory at the same time. The whole SURVO program is obeyed for each observation vector. Each SURVO instruction collects the information it needs from the current observation vector. For instance, the instruction CORREL collects a frequency count and sums, sums of squares and products of the variables referred to in the CORREL instruction. When all observation vectors have been read and treated in T2, the SURVO program is obeyed once more. At this stage the computer goes over the cumulated tables for the last time to get the final results and the output is generated.

In a sense the SURVO instructions have a dual interpretation. In stage T2 they lead to different internal function than in stage T3. From the point of view of the statistician, however, the instructions have a single meaning: give the defined results on the basis of the observation matrix.

Programming in SURVO 66

A SURVO program consists of the name of the program and of a sequence of instructions written in the SURVO 66 language. The name of the program is used in the output phase to label each page of results. The instructions are of the form

$$\langle \text{operator} \rangle @ \langle \text{list of parameters} \rangle$$

The delimiter symbol @ is used simply to terminate the operator identifier. The operator tells what should be done, and is expressed by a mnemonic operation code, e.g. MEAN, CORREL, END. The list of parameters has different requirements for different instructions. It establishes the necessary references which are needed in order to obey the instruction.

The instructions of a SURVO program are obeyed in the same order in which they are written in the program. The last instruction of any SURVO program is END@. Distinct instructions are to a large degree independent of each other. However, the SURVO objects (variables, tables, conditions), which are used in an instruction, must be defined in an earlier instruction.

The identifiers used in the list of parameters consist of letters, digits and special symbols (the six symbols @ :-()?exepted). They are terminated by the characters "space" or "line feed". The length of an identifier is unlimited; the system, however, considers only the first six characters. The program constants conform to usual programming language conventions.

A variable in the SURVO language may have several names. Each input variable is automatically associated with a standard name Xi, where i is the order number of the variable in the data vector. In order to get mnemonic programs and results it is customary to rename the variables using CALL-instructions. E.g. the instruction

```
CALL@ X3  WEIGHT
      X7  LENGTH
```

renames X3 and X7 as WEIGHT and LENGTH respectively. New variables and other SURVO objects are named in the same instruction where they are defined.

There exist means in the SURVO language to shorten long lists of names. The list X1, X2, ... , X20 can also be referred to by X1-X20. Other group references can be defined using the NAME-instruction. For instance the instruction

```
NAME@ PART1  X1 X2 X5 X6 X9
    @ PART2  X2 X4 X7 X8 X10
    @ALL     PART1  PART2
```

gives an easier means of reference: PART1 for variables X1, X2, X5, X6, X9, PART2 for variables X3, X4, X7, X8, X10 and an alternative reference ALL for X1-X10.

The variables and constants in SURVO 66 language are integers or fractions which are internally represented as integers scaled with a power of ten. There may also appear Boolean variables. No floating point variables are used, although the system makes use internally of floating point computing. The system is easiest to apply when all the data consists of integers: scaling requires some consideration by the programmer.

The parameter list of a SURVO instruction gives the SURVO objects to be operated upon. It also contains speciality parameters to specify the operation in more detail. The speciality parameters are expressed in the format

<speciality identifier> : <parameter identifier> .

In the following table we define the different speciality identifiers. They cannot all be used in connection with every SURVO instruction.

speciality identifier	function	parameter identifier	consequence of omission
N	give a name to a new SURVO object to be defined in the instruction	permissible identifier	a nameless SURVO-object
S	give the scaling of a new SURVO-variable	integer constant	depends on the instruction, usually omitted scaling
L	define the lower bound for a vari-able	constant	no lower bound
U	define the upper bound for a variable	constant	no upper bound
IF	define the selective condition which determines whether the instruction should be obeyed or omitted for the current data vector	Boolean variable	the instruction is obeyed for every data vector
M	suggest the use of a method which is better suited than the standard method	miscellaneous	normal method

T	refer to the variable to be cross-tabulated in TABLE-instruction	variable	the frequencies only are tabulated
W	refer to the variable to be used as a weight in MEAN, STDDEV and CORREL instructions	variable	no weighting applied

The instructions of SURVO 66 language can be grouped into <u>control</u> instructions, <u>transformation</u> instructions, <u>classification</u> and <u>tabulating</u> instructions, <u>Boolean</u> instructions and <u>analysis</u> instructions. We give here a tabular presentation of the main features of different instructions. The reader is referred to [3] for more detail.

<u>Control instructions</u>

END@	terminate the program list
WAIT@ IF: ⟨condition⟩	suspend program operation if the condition is satisfied
STOP@ IF: ⟨condition⟩	transfer to the next data vector if the condition is satisfied
M@m	give the length of the data vector (= m). This is usually the first instruction of any SURVO program.
N@n	give the number of data vectors (= n). This instruction may be omitted.
SPACES@ k	set the width of the result print-out to k characters.
COMMENT@ ⟨comment string⟩	the program can be made more readable by using comments
NAME@ ⟨identifier⟩ ⟨list of variables⟩	give a name to a group of variables.

CALL@ u_1 ⟨identifier 1⟩ ..... u_r ⟨identifier r⟩	give the variables $u_1, \ldots , u_r$ new names
DEF@ $u_1, u_2, \ldots , u_r$ L: ⟨lower bound⟩ U: ⟨upper bound⟩ S: ⟨scale⟩	the variables $u_1, u_2, \ldots , u_r$ are defined as having the properties defined by the speciality parameters. The variables will be checked for these properties during phase T2 of the SURVO system.

Transformation instructions

The transformations can be performed selectively using IF – conditions.

SET@ u u_1	$u := u_1$
ADD@ u $u_1 \ldots u_r$	$u := u_1 + \ldots + u_r$
SUB@ u u_1 u_2	$u := u_1 - u_2$
MULT@ u $u_1 \ldots u_r$	$u := u_1 \times u_2 \times \ldots u_r$
DIV@ u u_1 u_2	$u := u_1/u_2$
MOD@ u u_1	$u := \vert u_1 \vert$
SQRT@ u u_1	$u := \sqrt{u_1}$
LOG@ u u_1	$u := \ln u_1$
EXP@ u u_1	$u := \exp u_1$
MAX@ u $u_1 \ldots u_r$	$u := \max (u_1, \ldots , u_r)$
MIN@ u $u_1 \ldots u_r$	$u := \min (u_1, \ldots , u_r)$
ORDER@ u	u:= the sequence number of the data vector
LAG@ u u_1 k	u:= the value of the variable u_1 in the data vector which lies in the data matrix k rows earlier than the current vector.
PRINT@ $u_1 \ldots u_r$	A transformed data matrix is printed using the specified

M: ⟨number of output device⟩	output device. The vectors
IF: ⟨condition⟩	to be included in the transformed new matrix can be selected through the IF-condition.

Boolean instructions

EQUAL@	e u_1 u_2	e is <u>true</u> if $u_1 = u_2$
LESS@	e u_1 u_2	e " " " $u_1 < u_2$
LESSEQ@	e u_1 u_2	e " " " $u_1 \leq u_2$
BETWEEN@	e u_1 u_2 u_3	e " " " $u_1 \leq u_2 \leq u_3$
OR@	e $e_1 \ldots e_r$	$e := e_1 \vee e_2 \vee \ldots \vee e_r$
AND@	e $e_1 \ldots e_r$	$e := e_1 \wedge e_2 \wedge \ldots \wedge e_r$
NOT@	e e_1	$e := \neg e_1$

Classification and tabulating instructions

The CLASS-instruction is used to define a set of rules by which the variable values are mapped to class names or class number. Every set of classification rules is named to allow subsequent reference. The classification facility is used in TABLE- and TRANSF-instructions. The detailed format of the CLASS-instruction is

CLASS@ ⟨name of classification⟩
⟨class name 1⟩ ⟨lower bound⟩ ⟨upper bound⟩
.....
⟨class name r⟩ ⟨lower bound⟩ ⟨upper bound⟩
M: ⟨classification method⟩
S: ⟨scale⟩

The classification rule defined by a CLASS-instruction is available for use with any variable stored in the scale defined in the CLASS-instruction. The variable values x which fulfill the condition $a_i \leq x \leq b_i$ are mapped to the class i(i = 1, ... , r). The class names may be partially identical; the classes may thus consist of several distinct intervals. The class names are either nonnegative integers or any permissible SURVO identifiers.

The speciality parameter M has two possible values: FAST and SHORT. FAST guides the compiler to apply direct value indexing in table addressing. This method is sometimes wasteful in using the computer core memory. SHORT method applies a normal search strategy in table handling and therefore allows maximal storage economy.

Closely associated with the CLASS-instruction is a variable transformation instruction. This instruction is called TRANSF, and it defines a new variable applying a classification rule. The value of the new variable is the integer class number defined in a CLASS instruction or a simple count 1, 2, ... if alphanumeric class names have been used. The format of the TRANSF instruction is

```
TRANSF@ u  u1  c
     M:  m
    IF: <condition>
```

where u = the new variable, u_1 = the variable to be classified, c = the name of a classification rule defined earlier by a CLASS-instruction, m = the value to be given if the value of u_1 is outside the classification intervals.

The TABLE-instruction is used to tabulate frequency counts, percentages, mean values and standard deviations. The instruction is designed for construction of one-way and two-way tables. A TABLE-instruction performs r tabulating tasks with the same column variable. Tables in more dimensions are programmed applying conditional TABLE-instructions. The tables should be given names for later reference . The table may be used in analysis instructions. The CHI2-instruction can be used to compute a contingency test for a frequency table. The VARAN-instruction is able to perform a one-way or two-way analysis of variance using mean value and frequency count tables. The structure of the TABLE-instruction is as follows:

TABLE@ <column variable u_1><classification rule c>
<table name n_1> <row variable u_1> <classification rule c_1>
....
<table name n_r> <row variable u_r> <classification rule c_r>
T: <variable to be tabulated>
M: <output selection parameters>
IF: <condition>

Analysis instructions

Estimation of mean values, standard deviations and correlation coefficients is performed using MEAN-, STDDEV- and CORREL-instructions in the following format:

<operator>@ $u_1, \dots, u_r$
IF: <condition>
N: <name of moment table>
W: <weight variable>
M: <output specification>
T: <output specification>

where $u_1, \dots, u_r$ are variables. The sums of squares and sums of products are saved as the moment table, which should be named for later reference . These moments may be used in an analysis instruction, REGRAN or TTEST.

The MEAN-instruction computes mean values only. STDDEV-instruction estimates both mean values and standard deviations. CORREL-instruction computes, besides mean values and standard deviations, the product moment correlations of the variables $u_1, \dots, u_r$. In addition to other output options, the correlation matrix with mean values and standard deviations can be punched in an output form which conforms to the input requirements of standard multivariate analysis programs

The percentage points of empirical distributions can be examined using FRACT-instructions. The estimation of the percentage points is performed using the marginal distribution of a frequency table. The variable subject to investigation appears as a row variable in this table. The variable reference is hence performed indirectly using the table name . The general format of the FRACT-instruction is:

FRACT@ ⟨name of a table⟩ q r s ,

where the non-negative integers q, r, s give the selection rules for percentage points selected out of $P_0, P_1, \ldots, P_{99}$; P_i = the variable value which exceeds i percent of observed values. The instruction gives as results $P_q, P_{q+r}, P_{q+2r}, \ldots, P_s$.

The REGRAN-instruction fits a linear regression model

$$y = a_0 + a_1x_1 + \ldots + a_rx_r$$

to observations using the method of least squares. This analysis instruction is not designed to operate directly on the data. It needs a correlation matrix to get the necessary information. This arrangement has arisen from the experience that slightly different models are often estimated from the same set of variables. The format of the REGRAN-instruction is

REGRAN@ ⟨name of correlation matrix⟩
y
$x_1 \ldots x_r$

In the same way as the use of the REGRAN-instruction is based on an earlier CORREL-instruction, the VARAN-instruction uses a TABLE-instruction. The format of this instruction is simply

VARAN@ ⟨name of the table⟩ .

The specification of whether the analysis of variance is performed in one-way or two-way form, as well as the variable in question, appear implicitely by a reference to the table. The variable subject to the analysis of variance appears as a T-parameter in the corresponding TABLE-instruction. The classifications used in the tabulation specify the categories investigated using the analysis of variance, as well as whether one-way or two-way analysis is required. There is a problem in two-way analysis of variance when observation vectors fill the category table in an uneven manner. In SURVO language a heuristic method is used as an approximate solution in that case.

Any frequency table can be analysed for independence of its tabulating variables using the χ^2 - test. This happens applying a CHI2-instruction in the format

CHI2@<name of frequency table>.

The mean values in different groups are tested for equality using the TTEST-instruction. The sums and sums of squares needed for the computations are provided by earlier STDDEV or CORREL instructions. This information must have been given a reference name as a moment table. The format of the TTEST-instruction is

either TTEST@ <moment table 1><moment table 2>
or TTEST@ <moment table 1><variable u_1>
<moment table 2><variable u_2>.

In the former case it is requires that the variables to be compared appear in the same order in the moment tables.

An example of SURVO 66 programming

In order to illustrate SURVO programming we consider a recent statistical research by Dr. Knight on computer characteristics [5]. In this interesting paper the author investigates the functional dependence of computer power and its rental cost. This particular data has been chosen because we felt that most computer people are familiar with the concepts of this research.

The material which Dr. Knight has treated statistically contains 92 data vectors derived from production models of electronic digital computers. The attributes he has measured of each computer are: date introduced, scientific power in operations per second, commercial power in operations per second and inverse of computing cost in seconds of computing per dollar. The data matrix in [5] is of the following form:

Date introduced Month	Year	Scientific power(op/sec)	Commercial power(op/sec)	Inverse unit cost (sec/$)
4	63	21420	9079	44.54
7	63	67660	23420	23.98
....				
2	67	3127266	2755760	15.59
9	67	1086342	1021365	29.69

Computer no 303 is omitted here because of an obvious printing error.

We investigate the interdependence of the scientific power P of the computer and the computing cost C using the technological age T of the computer as an external variable to be compensated. The units of measurement for P, C and T are 1000 op/sec, $/hour, and month respectively. We will fit a logarithmic regression model

$$\ln P = a_0 + a_1 \ln C + a_2 T$$

to the data. We also cross-tabulate the average power of computers in three cost categories for each year 1963, ... , 67 of computer announcement. As data validity checks we require that the variables "month" and "year" should not be outside the intervals 1-12 and 63-67 respectively.

A reproduction of results is included. We can see that Grosch´s famous law $P = kC^2$ seems to fit well to Dr. Knight´s data.

SURVO program:

```
EVOLVING COMPUTER PERFORMANCE 1963-1967,DATAMATION,JAN.1968
M@5
CALL@ X1 MONTH
    @ X2 YEAR
DEF@ X5 S:1
   @ MONTH  L:1  U:12
   @ YEAR  L:63  U:67
DIV@ SPEED X3 1000    S:1
DIV@ COST 3600 X5  S:3
SUB@ Y1 68 YEAR
MULT@ Y2 12 Y1
SUB@ AGE Y2 MONTH

LOG@ LSPEED X3     S:3
   @ LCOST  COST   S:3

CLASS@ COSTCL
       CHEAP    0      30.000
       MODER   30.001  90.000
       EXPNS   90.001 500.000
       M:SHORT   S:3

TABLE@ YEAR -
        DEVEL  COST COSTCL
        T:SPEED

CORREL@ LSPEED LCOST AGE        N:CORR

REGRAN@ CORR
         LSPEED      LCOST AGE

END@
```

Results of the SURVO program:

```
EVOLVING COMPUTER PERFORMANCE 1963-1967,DATAMATION,JAN.1968

CLASSIFICATION: COSTCL
 CLASS           LIMITS
 CHEAP .0000000 30.00000
 MODER 30.00100 90.00000
 EXPNS 90.00100 500.0000

VARIABLES
 NO.    NAME   SCALE
   1   MONTH       0
   2    YEAR       0
   3      X3       0
   4      X4       0
   5      X5       1
   6   SPEED       1
   7    COST       3
   8      Y1       0
   9      Y2       0
  10     AGE       0
  11  LSPEED       3
  12   LCOST       3

EVOLVING COMPUTER PERFORMANCE 1963-1967,DATAMATION,JAN.1968
N=    91

TABLE: DEVEL

COLUMN VARIABLE:   YEAR
   ROW VARIABLE:   COST   CLASSIFICATION:COSTCL

FREQUENCIES
              63      64      65      66      67   TOTAL

  CHEAP        6       4      10       7       4      31
  MODER        7      11       9       5       1      33
  EXPNS        6       6       6       6       3      27
  TOTAL       19      21      25      18       8      91
```

MEANS OF SPEED

	63	64	65	66	67	TOTAL
CHEAP	5.5167	2.1000	20.810	1.6571	36.600	13.148
MODER	13.243	54.909	50.500	439.08	154.80	106.10
EXPNS	198.32	1371.6	1123.9	1875.8	1419.7	1173.2
TOTAL	69.247	421.05	296.24	747.89	570.05	391.06

EVOLVING COMPUTER PERFORMANCE 1963-1967,DATAMATION,JAN.1968
N= 91

CORR

VARIABLE	MEAN	STDDEV
LSPEED	9.963143	3.113192
LCOST	3.905297	1.233960
AGE	32.97802	14.36120

CORRELATION MATRIX: CORR

	LSPEED	LCOST	AGE
LSPEED	1.000	.8069	-.1797
LCOST	.8069	1.000	.0539
AGE	-.1797	.0539	1.000

EVOLVING COMPUTER PERFORMANCE 1963-1967,DATAMATION,JAN.1968

REGRESSION ANALYSIS

CORRELATION MATRIX: CORR

VARIANCE OF DEPENDENT VARIABLE LSPEED	9.6920
RESIDUAL VARIANCE	2.9632
MULTIPLE CORRELATION	.83322

REGRESSION COEFFICIENTS AND STANDARD DEVIATIONS:

VARIABLE	COEFF	STDDEV	T
CONSTANT	3.4946	.71522	4.8860
LCOST	2.0662	.14726	14.031
AGE	-.04853	.01265	-3.8356

Experiences and conclusions

Our experiences so far indicate that the idea of a statistical language seems to be feasible. We shall proceed to implement the system for a larger computer. We have also found that researchers have been able to specify their statistical data processing jobs in the SURVO language without any expert help.

We have observed a remarkable increase in the use of computers in statistical applications. Part of this increase in due to the ease of use when the researcher is able to specify himself his information processing needs. Part of the increase comes from new applications where the prohibiting cost of special programming is now to a large extent removed.

There also exist some negative aspects which we have found in our system. The method of scaling we have used in the system may sometimes cause unpleasant pitfalls. When transferring the system to a faster computer we will introduce more floating point computing to remedy this drawback. There also exists a steady demand from the users'side for more sophisticated statistical techniques in the SURVO system. A computer with larger memory capacity is needed to satisfy this demand. A final goal is an integrated system for all statistical manipulation needed in usual statistical research.

In system design we have aimed at simplicity where possible. Therefore the syntax of the SURVO language is chosen more in favour of simple compiling than of syntactical beauty.There have been, however, enough reasons to promote this research project as an interdisciplinary effort in co-operation with computer scientists, statisticians and users of computing services.

Acknowledgements

We are grateful to Oy Nokia Ab, Electronics Division and the University of Tampere for the support they have given to this research. In the implementation phase several persons have participated in the project. We want especially to mention the valuable contributions of Leena Lankinen, Tatu Kalin, Matti Ylinen as well as those of Pentti Kanerva and Kari Kärkkäinen.

Literature

[1] Couch, A.S: The Data-Text System Manual, Dept of Social Relations, Harvard University, Cambridge, Massachusetts, 1967.

[2] Dixon, W.J: Manual of BMD: Biomedical Computer Programs, Health Sciences Computing Facility, School of Medicine, University of California, Los Angeles, 1964.

[3] Mustonen, Seppo: Tilastollinen tietojenkäsittelyjärjestelmä SURVO 66, Monistesarja, Tampereen yliopiston tietokonekeskus, Moniste no 2, Tampere, 1967 (Statistical Data Processing System SURVO 66, Reports of the Computing Centre in the University of Tampere, Report no 2, Tampere, 1967). In Finnish.

[4] Pollack, Seymor: Establishing an Integrated Statistical Program Library, 18th Annual ACM Conference.

[5] Knight, E.K: Evolving Computer Performance 1963-67. Datamation Magazine, January 1968, pp. 31-35.