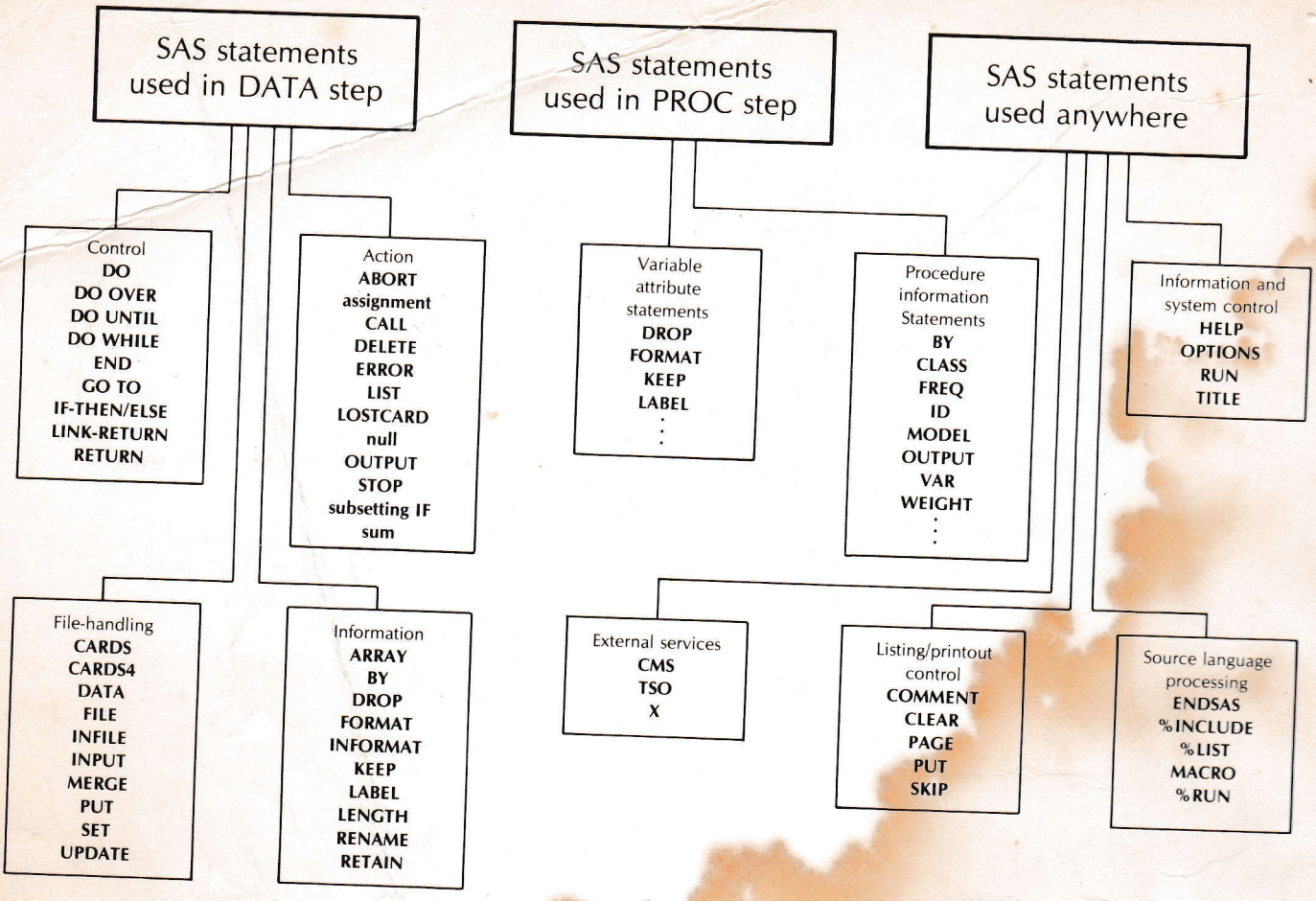
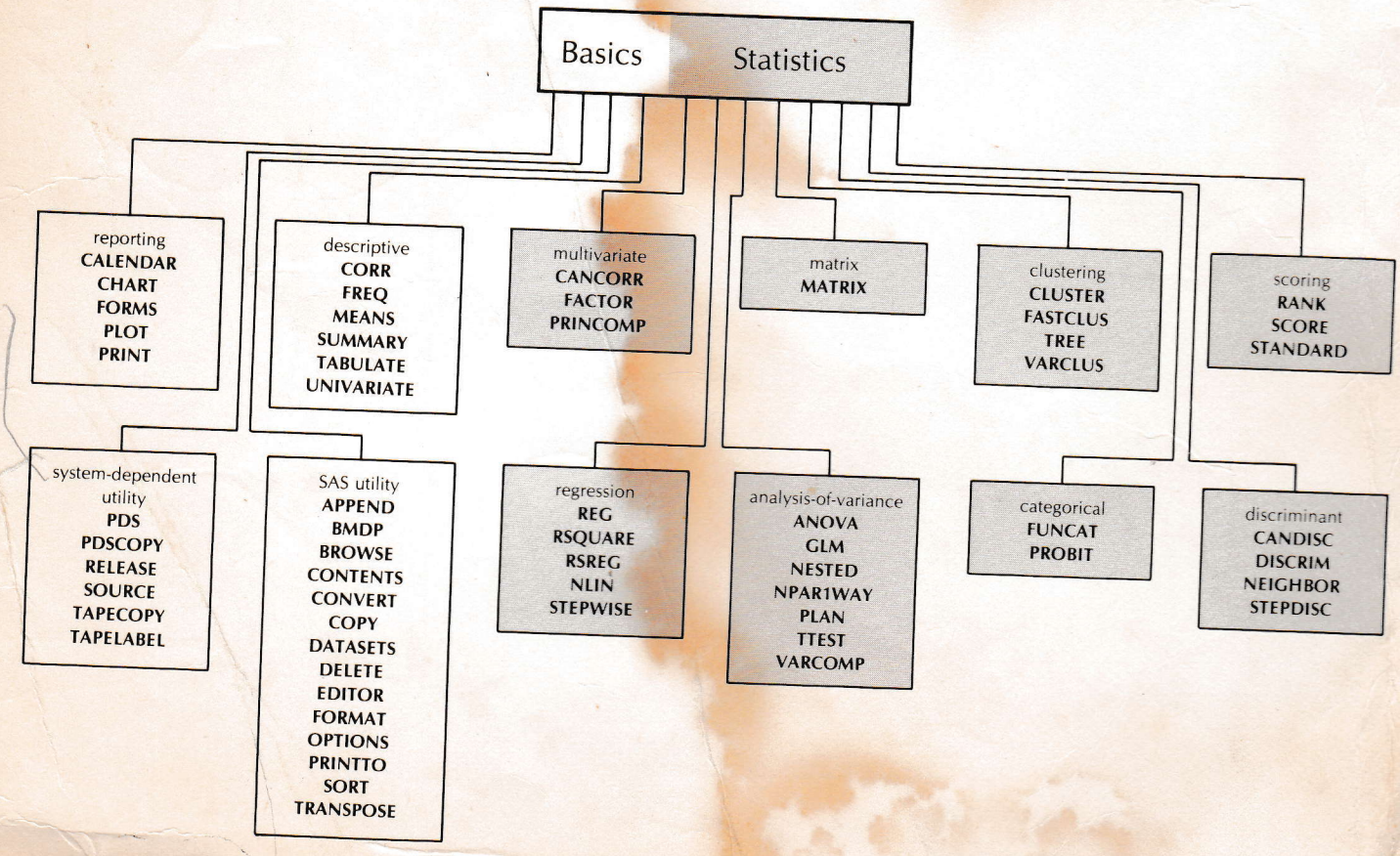
SAS USER'S GUIDE: Statistics

1982 Edition

SAS Institute
Statistical Analysis System


SAS statements used in DATA step
SAS statements used in PROC step
SAS statements used anywhere
Control
DO
DO OVER
DO UNTIL
DO WHILE
END
GO TO
IF-THEN/ELSE
LINK-RETURN
RETURN
Action
ABORT
assignment
CALL
DELETE
ERROR
LIST
LOSTCARD
null
OUTPUT
STOP
subsetting IF
sum
Variable attribute statements
DROP
FORMAT
KEEP
LABEL
Procedure information Statements
BY
CLASS
FREQ
ID
MODEL
OUTPUT
VAR
WEIGHT
Information and system control
HELP
OPTIONS
RUN
TITLE
File-handling
CARDS
CARDS4
DATA
FILE
INFILE
INPUT
MERGE
PUT
SET
UPDATE
Information
ARRAY
BY
DROP
FORMAT
INFORMAT
KEEP
LABEL
LENGTH
RENAME
RETAIN
External services
CMS
TSO
X
Listing/printout control
COMMENT
CLEAR
PAGE
PUT
SKIP
Source language processing
ENDSAS
%INCLUDE
%LIST
MACRO
%RUN

Procedures


Basics
Statistics
reporting
CALENDAR
CHART
FORMS
PLOT
PRINT
descriptive
CORR
FREQ
MEANS
SUMMARY
TABULATE
UNIVARIATE
multivariate
CANCORR
FACTOR
PRINCOMP
matrix
MATRIX
clustering
CLUSTER
FASTCLUS
TREE
VARCLUS
scoring
RANK
SCORE
STANDARD
system-dependent utility
PDS
PDSCOPY
RELEASE
SOURCE
TAPECOPY
TAPELABEL
SAS utility
APPEND
BMDP
BROWSE
CONTENTS
CONVERT
COPY
DATASETS
DELETE
EDITOR
FORMAT
OPTIONS
PRINTTO
SORT
TRANSPOSE
regression
REG
RSQUARE
RSREG
NLIN
STEPWISE
analysis-of-variance
ANOVA
GLM
NESTED
NPAR1WAY
PLAN
TTEST
VARCOMP
categorical
FUNCAT
PROBIT
discriminant
CANDISC
DISCRIM
NEIGHBOR
STEPDISC

Alice Allen Ray edited the *SAS User's Guide: Statistics, 1982 Edition*. **John P. Sall** was contributing editor. **Marian Saffer** was copy editor with assistance from **Stephenie P. Joyner** and **Judith K. Whatley**.

Data entry and text-editing were accomplished by **Andrea U. Littleton**.

Composition and production were provided by **Stephen K. Douglas**, **Elisabeth C. Smith**, and **June L. Woodward** under the direction of **W. Wayne Lindsey**.

Program authorship includes design, programming, debugging, support, and preliminary documentation. The SAS staff member listed first currently has primary responsibility for the procedure; others give specific assistance.

ANOVA J.H. Goodnight
CANCORR W.S. Sarle
CANDISC W.S. Sarle
CLUSTER W.S. Sarle
DISCRIM J.H. Goodnight
FACTOR W.S. Sarle, J.P. Sall
FASTCLUS W.S. Sarle
FUNCAT J.P. Sall
GLM J.H. Goodnight, J.P. Sall, W.S. Sarle
MATRIX J.P. Sall
NEIGHBOR J.H. Goodnight, W.S. Sarle
NESTED J.P. Sall
NLIN J.H. Goodnight, J.P. Sall
NPAR1WAY J.P. Sall
PLAN J.P. Sall
PRINCOMP W.S. Sarle
PROBIT D.M. Delong, J.H. Goodnight
RANK K.D. Kumar, J.P. Sall
REG J.P. Sall
RSQUARE W.S. Sarle, J.H. Goodnight
RSREG J.P. Sall
SCORE J.P. Sall
STANDARD J.H. Goodnight
STEPDISC W.S. Sarle
STEPWISE J.H. Goodnight
TREE G.K. Howell, W.S. Sarle
TTEST J.H. Goodnight
VARCLUS W.S. Sarle
VARCOMP J.H. Goodnight

Probability Routines D.M. Delong, J.H. Goodnight
Multiple Comparisons Routines W.S. Sarle, D.M. Delong
Multivariate Routines W.S. Sarle
Other Numerical Routines W.S. Sarle, D.M. Delong, J.P. Sall, J.H. Goodnight
Other Library Routines W.S. Sarle, R.D. Langston, J.P. Sall
Consulting Statisticians D.M. Delong, J.H. Goodnight, W.S. Sarle, J.P. Sall, H.J. Kirk, F.W. Young

(If you have questions or encounter problems, call SAS Institute and ask for technical support rather than an individual staff member.)

The correct bibliographic information for this manual is:

SAS Institute Inc. *SAS User's Guide: Statistics, 1982 Edition*. Cary, NC: SAS Institute Inc., 1982. 584 pp.

SAS User's Guide: Statistics, 1982 EDITION

ISBN 0-917382-37-4

83 7 6 5 4 3 2

Acknowledgments

Hundreds of people have helped SAS® in many ways since its inception. The individuals that we remember here have been especially helpful in the development of statistical procedures. An acknowledgment for SAS generally is in *SAS User's Guide: Basics, 1982 Edition*.

Anthony James Barr, Barr Systems
Wilbert P. Byrd, Clemson University
George Chao, Arnar-Stone Laboratories
Daniel Chilko, University of West Virginia
Richard Cooper, USDA
Sandra Donaghy, North Carolina State University
David B. Duncan, Johns Hopkins University
R.J. Freund, Texas A & M University
Wayne Fuller, Iowa State University
A. Ronald Gallant, North Carolina State University
Charles Gates, Texas A & M University
Thomas M. Gerig, North Carolina State University
Francis Giesbrecht, North Carolina State University
Harvey J. Gold, North Carolina State University
Harold Gugel, General Motors Corporation
Donald Guthrie, University of California at Los Angeles
Gerald Hajian, Burroughs Wellcome Company
Frank Harrell, Duke University
Walter Harvey, Ohio State University
Ronald Helms, University of North Carolina at Chapel Hill
Jane T. Helwig, Seasoned Systems Inc., Chapel Hill
Don Henderson, ORI
Harold Huddleston, Data Collection & Analysis, Inc., Falls Church, Virginia
David Hurst, University of Alabama at Birmingham
Emilio A. Icaza, Louisiana State University
William Kennedy, Iowa State University
Gary Koch, University of North Carolina at Chapel Hill
Kenneth Koonce, Louisiana State University
Clyde Y. Kramer (deceased), Virginia Polytechnic Institute, University of Kentucky, and Upjohn
Ardell C. Linnerud, North Carolina State University
Ramon C. Littell, University of Florida
H.L. Lucas (deceased), North Carolina State University
David D. Mason, North Carolina State University
J. Philip Miller, Washington University
Robert J. Monroe, North Carolina State University
Robert D. Morrison, Oklahoma State University
Kenneth Offord, Mayo Clinic
Robert Parks, Washington University
Richard M. Patterson, Auburn University

Introduction to the MATRIX Language

This chapter introduces MATRIX, which is both a SAS procedure and a programming language. The purpose of this section is to explain how MATRIX is used and to compare MATRIX to other programming languages.

MATRIX is a programming language. While most SAS procedures are prepackaged routines where the specifications merely control the details of an analysis, MATRIX is a complete programming language, and MATRIX statements are executable programming statements.

MATRIX makes SAS open-ended—suitable for custom work as well as packaged analyses. Analysts need two kinds of tools. They need packaged analysis programs for routine work and a general-purpose language for custom programming of methods too specialized or too new to be packaged. MATRIX provides a high-level programming language to serve the latter need. If a statistical method has not been implemented directly in a SAS procedure, you can program it using the MATRIX procedure.

MATRIX data elements are matrices. Most programming languages deal with single data elements; the fundamental data element in MATRIX is a matrix, a two-dimensional (row by column) array of numeric (double-precision real floating-point) values.

MATRIX expressions use operators that are applied to entire matrices. For example, the expression A+B in MATRIX adds the elements of the two matrices A and B. The expression A*B performs a matrix multiplication, while A#B does elementwise multiplication of the values in A and B.

MATRIX incorporates a powerful vocabulary of operators. Matrix operations that require calls to math-library subroutines in other languages are built into the language of MATRIX. Most programming languages have six to ten operators and a small function library. MATRIX offers 26 operators, 29 built-in functions, and all the functions of the DATA step; and MATRIX allows you to define your own functions.

MATRIX avoids housekeeping work typical in other programming languages. In many languages you have to explicitly declare, dimension, or allocate storage for a data item, and you cannot change its attributes once it is declared. In contrast, MATRIX does all the housekeeping automatically. The attributes of a matrix are determined when the matrix is given a value (this is called *late binding*). The procedure automatically finds space as needed for a matrix of any size. A matrix can change its size and attributes at any time.

MATRIX operations allow more direct programming. MATRIX eliminates most iterative specifications, calls to math subroutines, declarations, and allocations. For example, finding the sum of all positive elements of a matrix **X** requires one line in MATRIX:

```
S = SUM(X # (X>0)) .
```

The expression involves the sum of an elementwise product of two items. The first item is the matrix **X**; the second item is an indicator matrix showing the positive values of **X**. Once you have learned the matrix notation, this operation is clearer and more direct than an iterative specification that would be used in another programming language such as PL/I:

```
SUM=0;
DO I=1 TO N;
  DO J=1 TO M;
    IF X(I,J)>0 THEN SUM=SUM+X(I,J);
    END;
  END;
```

MATRIX notation is compact. Since MATRIX operations deal with arrays of numbers and the most commonly used mathematical and matrix operations are built directly into the language, programs that take hundreds of lines of code in other languages may take only a few lines in MATRIX.

MATRIX allows you to think directly in matrix algebra terms. If you are a statistician, you already know the formulas for many statistical methods. But in most languages it is no easy job to transcribe these formulas into a program. In MATRIX, you can transcribe almost directly from matrix algebra notation into MATRIX program statements. For example, the formula for the least-squares estimates of parameters in a linear model in matrix notation is written:

$$\mathbf{b} = (\mathbf{X'X})^{-1}\mathbf{X'y} .$$

It can be written in MATRIX as:

```
B=INV(X'*X)*X'*Y;
```

MATRIX is an alternative to APL. APL is another language that deals with matrices of values. Unlike MATRIX, however, APL works with a specialized character set that takes time to learn and is available only on special terminals. MATRIX can be learned more quickly than APL, since it builds on traditional and more familiar notation.

MATRIX has a rich set of control statements. MATRIX allows you to program with almost all of the control statements that exist in the DATA step, including LINK, GOTO, IF-THEN/ELSE, DO/END, iterative DO, and STOP.

MATRIX is preferred to DATA step programs for applications that must work across observations. The SAS DATA step is fine for programs that work with one observation at a time. The LAG function even allows you to work with a short set of observations. But to process across observations in ways that are impossible or difficult in the DATA step, use PROC MATRIX.