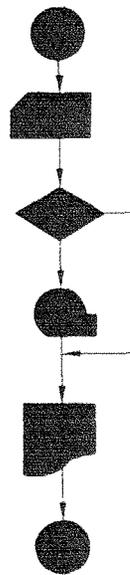


**THE NORTHEASTERN
FOREST-INVENTORY
DATA-PROCESSING SYSTEM.
VII. INFORMATION FOR
PROGRAMMERS
SUBSYSTEM TABLE.**



by
**Robert C. Peters and
Robert W. Wilson Jr.**

U. S. FOREST SERVICE RESEARCH PAPER NE-75
1967

NORTHEASTERN FOREST EXPERIMENT STATION, UPPER DARBY, PA.
FOREST SERVICE, U.S. DEPARTMENT OF AGRICULTURE
RICHARD D. LANE, DIRECTOR

About the Authors

ROBERT C. PETERS obtained his Bachelor's degree from the University of California in 1960 and his Master's at Yale University in 1961. He joined the Forest Service in 1961 as a research forester, and was assigned to the Station's biometrics unit from 1961 until 1965, when the unit was discontinued. Mr. Peters played a key role in the development of the data-processing system reported here.

ROBERT W. WILSON, JR. took his Bachelor's degree at The Pennsylvania State University in 1947 and his Master's and Ph.D. degrees at Yale University in 1948 and 1965, respectively. He joined the U. S. Forest Service in 1948 and has worked in various research capacities for the Northeastern Forest Experiment Station. From 1961 to 1965 he was in charge of the Station's biometrics unit at New Haven, Conn. He is assigned at present to the Forest Insect and Disease Laboratory at West Haven, Conn.

THE NORTHEASTERN FOREST-INVENTORY DATA-PROCESSING SYSTEM. VII. INFORMATION FOR PROGRAMMERS SUBSYSTEM TABLE.

Contents

A. INTRODUCTION	1
B. USE OF THE CALCULATE SUBROUTINE	2
C. MODIFICATION OF DIMENSIONED SPACE	4
Number of input records per sampling unit	5
Number of subunits per sampling unit	5
Number of data fields per record	5
Number of variable data fields per record	6
Number of semivariable data fields per record	7
Number of constant data fields per record	7
Number of cells in all input and output tables	8
Number of input tables	9
Number of output tables	9
Number of output table exceptions	10
D. PROGRAMMING FEATURES	10
Tape assignments	10
Use of sense switches and sense lights	11
Use of program halts	11
Use of the overlay feature	11
Bit manipulation	11
Subprogram names and functions	11
Important arrays and variables	12
E. OUTPUT TABLE FORMATS	15
F. SUMMARY OF ESTIMATING PROCEDURES	18
OPTION 1. — Transform and sum sampling-unit attributes over sets of sampling units	19
OPTION 2. — Transform and compute means of sampling-unit attributes over sets of sampling units	20
OPTION 3. — Transform and compute means and variances of sampling-unit attributes over sets of sampling units	20
OPTION 4. — OPTION 3 modified to include computation of covariances for ratio estimates	21

PREFACE

THIS paper is the seventh in a series of ten papers prepared to describe the forest-inventory data-processing system of the Northeastern Forest Experiment Station. This system was devised for using modern, large-scale, high-speed computers in processing forest-inventory data. The series will comprise the following papers:

- I. Introduction.
- II. Description of subsystem EDIT.
- III. Operation of subsystem EDIT.
- IV. Information for programmers — subsystem EDIT.
- V. Description of subsystem TABLE.
- VI. Operation of subsystem TABLE.
- VII. Information for programmers—subsystem TABLE.
- VIII. Description of subsystem OUTPUT.
- IX. Operation of subsystem OUTPUT.
- X. Information for programmers — subsystem OUTPUT.

VII-A. INTRODUCTION

ONE of the major projects of the U. S. Forest Service is a nationwide forest survey, which is designed to obtain useful and timely information about the timber resources of the United States. In the course of the surveys, which are made mainly on a state-by-state basis, great masses of detailed data are collected about timber volumes, growth, timber cut, and other characteristics of the timber resource.

In recent years the volume of information obtained from forest-survey field plots has increased greatly. The task of compiling and analyzing this mass of data with mechanical computing machines was both cumbersome and time-consuming.

A solution to this problem was seen in the development of the high-speed electronic computers. The Northeastern Forest Experiment Station, which was responsible for conducting the forest survey of the heavily forested Northeastern States, investigated the possibilities and devised the Northeastern Forest-Inventory Data-Processing System.

This paper presents information for programmers about part of the system, subsystem TABLE. Program TABLE is designed specifically to reduce large volumes of sample data to tables of statistics for the samples. The output of these sample summaries, in turn, is designed for use with program OUTPUT (see part VIII of this series) to produce equivalent tables of statistics for the sampled populations.

A general description of the program and detailed instructions for its use in solving data-processing problems are given in parts V and VI of this series. In the following chapters will be found selected programming information that will be useful if the programs must be modified for any reason. The program writeups and information on the program source decks may be obtained from the Northeastern Forest Experiment Station, 6816 Market Street, Upper Darby, Pennsylvania 19082.

This program is written in the standard IBM FORTRAN IV language, and is operative at the Yale University Computer Center on an IBM 7094/7040 direct coupled system under the IBSYS DCS operating system with IBJOB processor.¹ It will operate with little or no modification on other comparable systems. The main requirements for a machine on which to operate the standard version of the program are a 32K word core, a minimum of 36 bits per word, binary arithmetic capability, and 5 tape drives or equivalent input/output devices.

VII-B. USE OF CALCULATE SUBROUTINE

The normal version of program TABLE provides a dummy calculate routine named CALCUL. If the user wishes, he may program this routine either to generate new data fields in the input matrix or to change existing data fields. The subroutine which he must substitute for the dummy has the following calling sequence and DIMENSION statement:

```
SUBROUTINE CALCUL(TREE, ITREE, PLOT, IPLOT, POINT,  
1 IPOINT, NCORR, LTRCD, LTRVR, LPLVR, LPTCRD, LPTVR,  
2 NZAPS, NCARD, KPOINT)
```

```
DIMENSION TREE(LTRCD, LTRVR), ITREE(LTRCD, LTRVR),  
1 PLOT(LPLVR), IPLOT(LPLVR), POINT(LPTCRD, LPTVR),  
2 IPOINT(LPTCRD, LPTVR), NCORR(NZAPS)
```

Two additional subroutines are provided to retrieve (GETNO) and to store (STONO) data fields from the input matrix. All the user need do is supply the appropriate information in the calling sequence of these routines to carry out either operation. These routines are called in the following manner:

```
CALL GETNO (NDAFL, NFXFL, ANS, IANS, NXE, TREE,  
1 ITREE, PLOT, IPLOT, POINT, IPOINT, NCORR, LTRCD,  
2 LTRVR, LPLVR, LPTCRD, LPTVR, NZAPS, NCARD, KPOINT)
```

```
CALL STONO (NDAFL, NFXFL, ANS, IANS, NXF, TREE,  
1 ITREE, PLOT, IPLOT, POINT, IPOINT, NCORR, LTRCD,  
2 LTRVR, LPLVR)
```

The user supplies values for the variables NDAFL, NFXFL, ANS, IANS, and NXF. They mean the following:

¹ Mention of a particular product should not be construed as an endorsement by the Forest Service or the U. S. Department of Agriculture.

<i>VARIABLE</i>	<i>STONO</i>	<i>GETNO</i>
NDAFL	Number of data field to be stored in matrix.	Number of data field to be retrieved from matrix.
NFXFL	Mode of data field to be stored: 1 = fixed; 2 = floated	Mode of data field to be retrieved: 1 = fixed; 2 = floated
ANS	If mode is 2, value of data field to be stored.	If mode is 2, value of data field retrieved.
IANS	If mode is 1, value of data field to be stored.	If mode is 1, value of data field retrieved.
NXF	Index to determine position of data field in matrix. If data field to be stored has been defined on the CONSTANT (item 337), level NXF = 0. If data field to be stored has been defined on SEMI VARIABLE LEVEL (item 336), NXF runs from 1 to KPOINT. If data field to be stored has been defined on VARIABLE level (item 335), NXF runs from 1 to NCARD.	

For example, a routine used to test GETNO and STONO is the following:

```

SUBROUTINE CALCUL (TREE, ITREE, . . .
DIMENSION TREE(LTRCD, LTRVR), . . .
DO 1 I = 1, NCARD
NXF = 1
CALL GETNO (10, 2, ANS, IANS, NXF . . .
STOAN = ANS * 10.
CALL STONO (10, 2, STOAN, IANS, NXF . . .
1 CONTINUE
RETURN
END

```

This routine takes data field number 10, which is a floating-point number and has been defined on the VARIABLE card (part VI, item 335), and multiplies it by 10.0 and then stores the product of data field 10 and the constant 10.0 back into data field 10. This is done NCARD times thus performing the operation on data field 10 in each of the records making up the sampling unit.

VII-C. MODIFICATION OF DIMENSIONED SPACE

The standard version of program TABLE carries restrictions on both the dimensions and the overall size of problem that can be handled in a single-processing run. These restrictions are a result of the manner in which dimensioned space has been allocated (table 1) and the total space available in a given operating system. The program has been written so that all modifications of

Table 1. — Summary of dimensioned space restrictions, and associated program variables and arrays

Item	Restrictions	Variable	Arrays
Maximum number of input records per sampling unit	160	LTRCD	TREE, ITREE
Maximum number of subunits per sampling unit	10	LPTCRD	POINT, IPOINT
Maximum number of data fields per record	132	LIVR*	R
Maximum number of variable data fields	24 + 1	LTRVR*	TREE, ITREE
Maximum number of semi-variable data fields	24 + 1	LPTVR*	POINT, IPOINT, LPOINT
Maximum number of constant data fields	24 + 1	LPLVR*	PLOT, IPLOT, LPLOT
Maximum number of cells in all input and output tables	10,000	NDIMEN	AIMP, IMP
Maximum number of input tables	80	LINTB	NTAB
Maximum number of output tables	40	LDETB	INF, NADD, TNAME
Maximum number of output table exceptions	150	LNEX	EXCEP, NEXCEP

* NZAPS must always be equal either to LIVR or to the sum of the values of LTRVR, LPTVR, and LPLVR, whichever is larger. In the DIMENSION statement, the dimension of NCORR must always be equal to NZAPS, also.

dimensioned space can be made in the subprogram called MAINT. No other parts of the program need be touched for this purpose. The use of dimensioned space and the means of changing dimensions are discussed in detail below.

Number of Input Records per Sampling Unit

In the standard version of program TABLE, up to 160 input records may be processed per sampling unit. To change this maximum, the following steps (and only these) must be taken:

1. In the subprogram called MAINT, the variable named LTRCD must be set equal to the desired maximum value.
2. In the subprogram called MAINT, the DIMENSION statement must be changed so that the first dimension of the arrays TREE and ITREE equals the new value of LTRCD.

Number of Subunits per Sampling Unit

In the standard version of program TABLE, up to 10 subunits may be processed per sampling unit. To change this maximum, the following steps (and only these) must be taken:

1. In the subprogram called MAINT, the variable named LPTCRD must be set equal to the desired maximum value.
2. In the subprogram called MAINT, the DIMENSION statement must be changed so that the first dimension of the arrays POINT and IPOINT equals the value of LPTCRD.

Number of Data Fields per Record

In the standard version of program TABLE, up to 132 data fields may be contained in a record. To change this maximum, the following steps (and only these) must be taken:

1. In the subprogram called MAINT, the variable named LIVR must be set equal to the desired maximum value.
2. In the subprogram called MAINT, the DIMENSION statement must be changed so that the dimension of the array R equals the new value of LIVR.

Number of Variable Data Fields per Record

In the standard version of program TABLE, up to 24 data fields may be specified as containing variable values (see item 335). To change this maximum, the following steps (and only these) must be taken:

1. In the subprogram called MAINT, the variable named LTRVR must be set equal to the desired maximum value plus one.
2. In the subprogram called MAINT, the DIMENSION statement must be changed so that the dimension of LTREE and the second dimension of arrays TREE and ITREE equals the new value of LTRVR.
3. In the subprogram called MAINT, the variable named NZAPS must be set equal either to LIVR or to the sum of the values of the constants LTRVR, LPTVR, and LPLVR, whichever is greater.
4. In the subprogram called MAINT, the DIMENSION statement must be changed so that the dimension of array NCORR equals the new value of NZAPS.
5. In the control deck, if the desired maximum value is greater than 24, the input record variable fields card (item 335) must be followed by additional cards of the following format:

<i>Columns</i>	<i>Contain —</i>	<i>Explanation</i>
1-3	XXX	= 3 numeric characters, giving the identification number of the 25th data field that contains variable values, as described for item 335.
4-78	XXX . . . X	= Repetitions of columns 1-3 format, giving the identification numbers of the remaining data fields that contain variable values.

Number of Semivariable Data Fields per Record

In the standard version of program TABLE, up to 24 data fields may be specified as containing semivariable values (see item 336). To change this maximum, the following steps (and only these) must be taken:

1. In the subprogram called MAINT, the variable named LPTVR must be set equal to the desired maximum value plus one.
2. In the subprogram called MAINT, the DIMENSION statement must be changed so that the second dimension of arrays POINT and IPOINT equals the new value of LPTVR.
3. In the subprogram called MAINT, the DIMENSION statement must be changed so that the dimension of array LPOINT equal the new value of LPTVR.
4. In the subprogram called MAINT, the variable named NZAPS must be set equal either to LIVR or to the sum of the values of the constants LTRVR, LPTVR, and LPLVR, whichever is greater.
5. In the subprogram called MAINT, the DIMENSION statement must be changed so that the dimension of the array NCORR equals the new value of NZAPS.
6. In the control deck, if the desired maximum value is greater than 24, the input record semivariable fields card (item 336) must be followed by additional cards of the following format:

<i>Columns</i>	<i>Contain —</i>	<i>Explanation</i>
1-3	XXX	= 3 numeric characters, giving the identification number of the 25th data field that contains semi-variable values, as described for item 336.
4-78	XXX . . . X	= Repetitions of the columns 1-3 format, giving the identification number of the remaining data fields which contain semi-variable values.

Number of Constant Data Fields per Record

In the standard version of program TABLE, up to 24 data fields may be specified as containing constant value (see item 337). To change this maximum, the following steps (and only these) may be taken:

1. In the subprogram called MAINT, the variable named LPLVR must be set equal to the desired maximum value plus one.
2. In the subprogram called MAINT, the DIMENSION statement must be changed so that the dimension of the arrays PLOT, IPLOT, and LPLLOT equal the new value of LPLVR.
3. In the subprogram called MAINT, the variable named NZAPS must be set equal either to LIVR or to the sum of the values of the constants LTRVR, LPTVR, and LPLVR, whichever is greater.
4. In the subprogram called MAINT, the DIMENSION statement must be changed so that the dimension of the array NCORR equals the new value of NZAPS.
5. In the control deck, if the desired maximum value is greater than 24, the input record constant fields card (item 337) must be followed by additional cards of the following format:

<i>Columns</i>	<i>Contain —</i>	<i>Explanation</i>
1-3	XXX	= 3 numeric characters, giving the identification number of the 25th data field that contains constant values, as described for item 337.
4-78	XXX . . . X	= Repetitions of the columns 1-3 format, giving the identification numbers of the remaining data fields that contain constant values.

Number of Cells in All Input and Output Tables

In the standard version of program TABLE, up to 10,000 locations are available for storing all input and output tables. To change this maximum, the following steps (and only these) must be taken:

1. In the subprogram called MAINT, the variable named NDIMEN must be set equal to the desired maximum value.
2. In the subprogram called MAINT, the DIMENSION statement must be changed so that the dimension of the arrays AIMP and IMP equals the new value of NDIMEN.

If the space required for the input and output tables specified in a given control deck exceeds the dimensioned space, message 3 or message 6 will be printed during the reading of the control

deck, and processing will halt. The space required can be computed as follows:

$$\left[\sum_{i=1}^m (e_i) \right] + K \left[\sum_{j=1}^n (r_j + 1) \times (c_j + 1) \right] + [(r_{\max} + 1) \times (c_{\max} + 1)]$$

where

m = the total number of input tables.

e_i = the total number of entries in the i th input table.

K = a multiplier, the value of which depends upon the output option (item 302, column 9); as follows:

<i>Option</i>	<i>Value of K</i>
1 or 2	1
3	2
4	3

n = the total number of output tables.

r_j = the number of rows in the j th output table, as specified in columns 24-25 of item 321.

c_j = the number of columns in the j th output table, as specified in columns 27-28 of item 321.

r_{\max} = the number of rows in the largest output table.

c_{\max} = the number of columns in the largest output table.

Number of Input Tables

In the standard version of program TABLE, up to 80 input tables may appear in the control deck. To change this maximum, the following steps (and only these) must be taken:

1. In the subprogram called MAINT, the variable named LINTB must be set equal to the desired maximum value.
2. In the subprogram called MAINT, the DIMENSION statement must be changed so that the first dimension of the array NTAB equals the new value of LINTB.

Number of Output Tables

In the standard version of program TABLE, up to 40 output tables may be specified in the control deck. To change this maxi-

mum, the following steps (and only these) must be taken:

1. In the subprogram called MAINT, the variable named LDET_B must be set equal to the desired maximum value.
2. In the subprogram called MAINT, the DIMENSION statement must be changed so that the first dimension of the arrays INF and NADD equals the new value of LDET_B.
3. In the subprogram called MAINT, the DIMENSION statement must be changed so that the dimension of the array TNAME equals the new value of LDET_B.

Number of Output Table Exceptions

In the standard version of program TABLE, up to 150 output table exceptions (item 323) can appear in the control deck. To change this maximum, the following steps (and only these) must be taken:

1. In the subprogram called MAINT, the variable named LNEX must be set equal to the desired maximum value.
2. In the subprogram called MAINT, the DIMENSION statement must be changed so that the first dimension of the arrays EXCEP and NEXCEP equals the new value of LNEX.

VII-D. PROGRAMMING FEATURES

The following items will be of interest to programmers who plan to modify the standard version of program TABLE for use on other computers or under other operating systems.

Tape Assignments

In the standard version of program TABLE the FORTRAN logical tape assignments are as follows:

<i>Unit</i>	<i>Use</i>
5	Monitor input for program deck and job control deck.
6	Monitor print for job summary and debug output.
15	Sorted input file in binary or BCD.
19	Output of sample summary tables in binary only.

These tape assignments can be changed to fit local conditions by loading appropriate file routines with the program. See your systems representative or the section entitled FORTRAN files in the IBM IBJOB processor manual, file number 7090-27.

Use of Sense Switches and Sense Lights

No sense switches are used in program TABLE. All sense switches will be set at normal monitor settings.

No sense lights are used.

Use of Program Halts

There are no halts in program TABLE.

Use of the Overlay Feature

The standard version of program TABLE is constructed so that the overlay feature can be used when sufficient storage is not available for program and data. The configuration is as follows:

<i>Link</i>	<i>Contains Subprograms</i>
0	MAINT
1	CONTRL
2	TABLE, CALCUL, STONO, GETNO, VARIAN

Bit Manipulation

Program TABLE uses 36-bit, fixed-point words for all input table entries. Certain types of tables (LOOKUP and RANGE operation) use several numbers packed in a single word. These numbers are packed and unpacked by the use of binary arithmetic. The program will not operate on a machine that uses either a larger or smaller number of bits in fixed-point operation unless the shift constants that are used to pack and unpack these words are changed to correspond with the word length.

Subprogram Names and Functions

CONTRL	Reads all cards in the control deck and sets up storage array for processing input data.
TABLE	Reads input data and forms tables defined in control deck for each sample unit and data set.
VARIAN	Computes variances and means for a data set and writes final table on tape 19. Called by subprogram TABLE.
CALCUL	User-written subprogram to compute attributes not present in a usable form on the input data. Called by subprogram TABLE.
STONO	Used to store calculated data field in the input matrix. Called from subprogram CALCUL.
GETNO	Used to retrieve data field from the input matrix. Called from subprogram CALCUL.

MAINT Main calling program in which dimensions of all arrays are set. Calls subprograms CONTROL and TABLE.

Important Arrays and Variables

The following are the principal arrays and variables used in program TABLE:

<i>Array</i>	<i>Dimension</i>	<i>Description</i>
AIMP, IMP	NDIMEN	Floating- and fixed-point storage array for all input and output tables.
NTAB	LINTB x 4	Indexing information for all input tables; where LINTB is the number of input tables, and locations in the second dimension are used as follows: 1 = the alphameric name of the input tables. 2 = the beginning location of a table in IMP. 3 = the last location of a table in IMP. 4 = the number of entries in a table.
INF	LDETB x 35	Indexing information for all output tables; where LDETB is the number of output tables, and the locations in the second dimension are used as follows: 1 = The number of rows in an output table, plus 1. 2 = The number of columns in an output table, plus 1. 3 = The beginning location in IMP of a facsimile output table, less 1 plus the number of rows in the table. 4 = The beginning location in IMP of a final output table, less one plus the number of rows in the table. 5 = The beginning location in IMP of the sums of squares for a final output table, less one plus the number of rows in the table. 6 = The beginning location in IMP of the sums of cross-products, if any, less one plus the number of rows in the table. 7 = The number of entries to be made in an output table. 8 = The name of the input table, if any, to be used in defining the column index for the first entry in an output table. 9 = The name of the operation to be used in defining the column index for the first entry in an output table. 10 = The identification number of the data field or value of the constant to be used

<i>Array</i>	<i>Dimension</i>	<i>Description</i>
		in defining the column index for the first entry in an output table.
		11-13 = Same as 8-10, except that the information relates to defining the row index of the first entry in an output table.
		14 = The identification number of the data field to be used as the first entry in an output table.
		15-21 = Same as 8-14, except that the information relates to the second entry to be made in an output table, if any.
		22-28 = Same as 8-14, except that the information relates to the third entry to be made in an output table, if any.
		29-35 = Same as 8-14, except that the information relates to the fourth entry to be made in an output table, if any.
EXCEP, NEXCEP	LNEX x 3	Storage array for read-in information about table entry exceptions, where LNEX is the number of exceptions in the job and the locations of the second dimension are used as follows: 1 = The identification number of the data field to which the exception applies. 2 = The value of the constant to be used in making the exception. 3 = The relational operator to be used in making the exception.
NADD	LDETB x 8	Indexing information for input table entry exceptions stored in EXCEP; where LDETB is the number of output tables, and the locations of the second dimension are used as follows: 1 = The index of the row in EXCEP where the first exception for the first entry in an output table is stored. 2 = The number of sequential exceptions applying to the first entry in an output table. 3-4 = Same as 1-2, except that the information applies to the second entry in an output table, if any. 5-6 = Same as 1-2, except that the information applies to the third entry in an output table, if any. 7-8 = Same as 1-2, except that the information applies to the fourth entry in an output table, if any.
NCORR	NZAPS	Storage array for the storage (machine) locations of the data fields of an input record, where NZAPS

<i>Array</i>	<i>Dimension</i>	<i>Description</i>
		is equal to or greater than the number of data fields in the record.
PLOT, IPLOT	LPLVR	Floating- and fixed-point storage array for data fields in the input records defined as constant fields (item 337).
TREE, ITREE	LTRCD x LTRVR	Floating- and fixed-point storage array for data fields in the input records defined as variable fields (item 335).
POINT, IPOINT R	LPTCRD x LPTVR LIVR	Floating- and fixed-point storage array for data fields defined as semivariable fields (item 336). Storage array into which each individual input record is read from the input tape, where LIVR is the number of data fields in the record.
LPLOT	LPLVR	Storage array for the identification numbers of the input data fields defined as constant fields (item 337).
LPOINT	LPTVR	Storage array for the identification numbers of the data fields defined as semivariable fields (item 336).
LTREE	LTRVR	Storage array for the identification numbers of the data fields defined as variable fields (item 335).
IDPLOT	6	Storage array for the identification numbers of the input data fields defined as sampling-unit identification (item 334).
IDPT	1	Storage for the identification number of the data field defined as subunit identification, if any (item 334).
IDPI	6	Storage array for the identification numbers of the data fields defined as data-set identification (item 334).
IDPIO	6	Storage array for the identification data fields for the current data set.
IZDPIO, PIDPIO	6	Storage array for the identification data fields for the previous data set.
IDPLO	6	Storage array for the identification data fields of the current sampling unit.
KDP	1	Storage for the identification data field for the current subunit.
FMT	28	Storage array for the read-in input record format specification.
IVAR	1	The number of data fields in each input record.
NREAD	1	The total number of records to read from the input file.
NCOUT	1	The current number of records read from the input file.
MODE	1	The mode in which the input file is written: 1 equals binary; 2 equals BCD.

<i>Array</i>	<i>Dimension</i>	<i>Description</i>
NPLOT	1	The number of sampling units processed in the current data set.
KPOINT	1	The number of subunits read for the current sampling unit.
NCARD	1	The number of records read for the current-sampling unit.
NPSTEP	1	The total number of output tables defined in the control deck.
TNAME	IDETB	Storage array for the names (alphanumeric) of the output tables in the order that they are defined in the control deck.
ISTEP	1	The index number of the output table currently being processed.
INDEX	2	Temporary storage for final table indexes.

VII-E. OUTPUT TABLE FORMATS

Two alternative output table formats are available. The option is exercised in columns 12-16 of the output option card (item 302).

In normal job processing, these columns are left blank (not punched) and the table output is written on magnetic tape in binary mode. This form of output provides for rapid transmission of the output tables to the OUTPUT program in which they may be weighted, summed, labeled and printed, according to the sampling and table selection options available in that program.

If columns 12-16 of the output option card (item 302) are punched with the word DEBUG, the table output will be printed in BCD mode (E specification). As the control word implies, this alternative will normally be used only in debugging program changes or control decks. It does not provide for any further processing of the output data.

The general order of the table output is the same for both alternatives. All table output from the job is in a single file. For the binary tape output, this means that there is only one end-of-file mark in the output and it appears after the last record output from the job.

Within the file, the sets of output tables for each input data set are in the same order as the data sets appear in the input file. Within the data set, the output tables are in the same order as are

the corresponding output table definition cards (item 321) in the control deck. The output for a given table varies with the output option given in column 9 of the output option card (item 302). If the option is:

1. Only the table sums are given for each output table.
2. Only the table of means is given for each output table.
3. The table of means, followed immediately by the table of the variances of the means, is given for each output table.
4. The table of means, followed immediately by the table of the variances of the means, in turn followed immediately by the table of covariances of means, is given for each output table.

In the binary tape output, three types of record are used. The first type is repeated as the first record of each data-set output. The pair of records of the second and third types are repeated for every output table within the data set. The three types of record are described below.

<i>Record number</i>	<i>Word number</i>	<i>Description</i>
1	1	A number equal to one plus the number of identification words that follow in this record. This number is one greater than the number of data set identification fields specified in columns 13-27 of the input record identification fields card (item 334).
	2	The value of the data-set identification field specified in columns 13-15 of the input record identification fields card (item 334).
	3-6	Values of the remaining data-set identification fields specified in columns 16-27 of the input record identification fields card (item 334).
	7	The total number of output tables for the data set which follow this record. This number is equal to the number of output table definition cards (item 321) in the control deck.
	8	The total number of sampling units in the data set.

<i>Record number</i>	<i>Word number</i>	<i>Description</i>
2	1	An output table name, as given in columns 14-19 of an output table definition card (item 321).
	2	The number of rows, including column totals, in the output table named in word 1 of this record. This number is one greater than the number punched in columns 24-25 of the output table definition card (item 321) for this output table.
	3	The number of columns, including row totals, in the output table named in word 1 of this record. This number is one greater than the number punched in columns 27-28 of the output table definition card (item 321) for this output table.
3	1	The value in the first element or cell of the first column of the output table named in the preceding record 2, word 1. If the output option specified in column 9 of the output option card (item 302) is 1, the value will be a sum over the sampling units of the data set. Otherwise, it will be a mean over the sampling units.
	2-r	The values in the remaining cells of the first column of the table, where r is the number of rows given in the preceding record 2, word 2.
	(r+1) — (rxc)	The values in the cells of the remaining columns of the output table, where r and c are the numbers of rows and columns given in the preceding record 2, words 2 and 3, respectively.
	((rxc) + 1) — 2(rxc)	The values of the variances of the cells of the output table named in the preceding record 2, word 1. These words appear in the record only if output option 3 or 4 is punched in column 9 of the output option card (item 302). Otherwise the record ends with word (rxc).
	(2(rxc) + 1) — 3(rxc)	The values of the covariances between the cells and the total of the output table named in the preceding record 2,

<i>Record number</i>	<i>Word number</i>	<i>Description</i>
		word 1. These words appear in the record only if output option 4 is punched in column 9 of the output option card (item 302). Otherwise, the record ends with word (rxc) (output options 1 and 2) or with word 2 (rxc) (output option 3).

In the printed BCD output the numbers are in floating-point format (E specification). Essentially the same arrangement is followed as with the binary output, except that the output is broken into lines. There are eight entries (or cells) per line in columnar sequence, and as many lines are printed as necessary to record all the table entries.

For example, a table that has been defined as having 10 rows and 15 columns will appear printed in the following way: line 1 will represent column 1, rows 1-8; line 2 will represent column 1, rows 9-11 and column 2, rows 1-5. Row 11 of column 1 is the total of column 1 that has been provided by the program. The next line represents column 2, rows 6-11, and column 3, rows 1-2, and so forth through the entire table. If the option provides for tables of variances, these follow the last element of the means; and if the option provides for tables of covariances, these follow the last element of the table of variances.

The table is identified by a printed line that gives the table name, the entire number of rows and columns, and the entire number of cells represented in the printed tables. This number is merely the number of rows plus one times the number of columns plus one times the number of tables represented. If only means are printed, the number is one; if means and variances are united the number is two; if means, variances and covariances the number is three.

VII-F. SUMMARY OF ESTIMATING PROCEDURES

In this chapter the four output options available in program TABLE are presented in detail.

Vector notation is used to make the presentation of computing procedures compact and easy to read. An input vector, $\overset{1}{Y}$, is a one-

dimensional array representing a sampling unit attribute. An output vector, $\overset{O}{Y}$, represents an output table (in general, a two-dimensional array or matrix) summarizing the sampling-unit attribute. A final output vector, $\overset{F}{Y}$, represents an estimate of the population attribute that corresponds to the sampling-unit attribute. Elements of these vectors are represented by y_i, y_i, y_i .

It must not be inferred from what follows that the arithmetic is the arithmetic of vectors or matrices although, in general, it is correct vector arithmetic as shown. What is implied is simply the sequential and independent application of the indicated operation to each pair of equivalent elements from the two vectors. In this sense, the procedures will generalize to the case of matrices; otherwise they will not.

Other notational conventions adopted here are the use of a bar over an attribute symbol (Y) to symbolize the arithmetic mean of an attribute, and the use of a dot that replaces a subscript ($y_{j.k}$) to indicate the sum over-all members of the set represented by the subscript.

OPTION 1.—Transform and Sum Sampling-Unit Attributes Over Sets of Sampling Units

Compute: $\overset{O}{Y}_j$

Given: A set ($j = 1$) of $\overset{I}{Y}_{jk}$, and T

Where:

j = Subscript for the j th sample stratum or set of sampling units.

k = Subscript for the k th sampling unit

$\overset{O}{Y}_j$ = An output vector (output table) containing the sum of the sampling-unit output vectors, $\overset{O}{Y}_{jk}$, which represent a summary of the sampling-unit attribute input vectors, $\overset{I}{Y}_{jk}$

$\overset{I}{Y}_{jk}$ = An input vector containing a sampling-unit attribute (a data field from the input-data matrix for a sampling unit)

T = A set of rules whereby the elements of the input data vector, $\overset{I}{Y}_{jk}$, are redistributed (transformed) to form the output vector (table), $\overset{O}{Y}_{jk}$

Procedure: $\overset{O}{Y}_{jk} = \overset{I}{TY}_{jk}$

$$\text{Output: } \overset{\circ}{Y}_j = \sum_{k=1}^{P_j} \overset{\circ}{Y}_{jk}$$

OPTION 2.—Transform and Compute Means of Sampling-Unit Attributes Over Sets of Sampling Units

Compute: $\overset{\circ}{Y}_j$.

Given: Sets of $\overset{\circ}{Y}_{jk}$, P_j , and T

Where:

j = Subscript for the j th sample stratum set of sampling units

k = Subscript for the k th sampling unit

$\overset{\circ}{Y}_j$ = An output vector (output table) containing the arithmetic mean of the sampling-unit output vectors, $\overset{\circ}{Y}_{jk}$, which represent a summary of the sampling-unit attribute input vectors, $\overset{\circ}{Y}_{jk}$

$\overset{\circ}{Y}_{jk}$ = An input vector containing a sampling-unit attribute (a data field from the input-data matrix for a sampling unit)

P_j = The number of input sampling units in the stratum or set

T = A set of rules whereby the elements of the input-data vector, $\overset{\circ}{Y}_{jk}$, are redistributed (transformed) to form the output vector

(table), $\overset{\circ}{Y}_{jk}$

Procedure: $\overset{\circ}{Y}_{jk} = TY_{jk}$

$$\text{Output: } \overset{\circ}{Y}_j = \frac{\sum_{k=1}^{P_j} \overset{\circ}{Y}_{jk}}{P_j}$$

OPTION 3.—Transform and Compute Means and Variances of Sampling-Unit Attributes Over Sets of Sampling Units

Compute: $\overset{\circ}{Y}_j, V\overset{\circ}{Y}_j$.

Given: Sets of $\overset{\circ}{Y}_{jk}$, P_j , and T

Where:

j = Subscript for the j th sample stratum or set of sampling units.

k = Subscript for the k th sampling unit

\bar{Y}_j^0 = An output vector (output table) containing the arithmetic mean of the sampling-unit output vectors, \bar{Y}_{jk}^0 , which represent a summary of the sampling-unit attribute input vectors, \bar{Y}_{jk}^1 .

$V\bar{Y}_j^0$ = The variance of \bar{Y}_j^0 .

\bar{Y}_{jk}^1 = An input vector containing a sampling-unit attribute (a data field from the input-data matrix for a sampling unit)

P_j = The number of input-sampling units in the stratum or set

T = A set of rules whereby the elements of the input-data vector, \bar{Y}_{jk}^1 , are redistributed (transformed) to form the output vector (table), \bar{Y}_{jk}^0

Procedure: $\bar{Y}_{jk}^0 = T\bar{Y}_{jk}^1$

$$\text{Output: } \bar{Y}_j^0 = \frac{\sum_{k=1}^{P_j} \bar{Y}_{jk}^0}{P_j}$$

$$V\bar{Y}_j^0 = \frac{\sum_{k=1}^{P_j} (\bar{Y}_{jk}^0 - \bar{Y}_j^0)^2}{P_j (P_j - 1)}$$

OPTION 4.—OPTION 3 Modified to Include Computation of Covariances for Ratio Estimates

Compute: \bar{Y}_j^0 , $V\bar{Y}_j^0$, $C\bar{V}_j^0$.

Given: Sets of \bar{Y}_{jk}^1 , P_j and T

Where:

j = Subscript for the j th sample stratum or set of sampling units

k = Subscript for the k th sampling unit

\bar{Y}_j^0 = An output vector (output table) containing the arithmetic mean of the sampling-unit output vectors, \bar{Y}_{jk}^0 , which represent a summary of the sampling-unit attribute input vector \bar{Y}_{jk}^1

$V\bar{Y}_j^0$ = The variance of \bar{Y}_j^0 .

$\overset{0}{CV}_j$ = The mean covariance of output vectors for sampling units, $\overset{0}{Y}_{jk}$,
and the sums (totals) of elements in these vectors, $\overset{0}{Y}_{jk}$

$\overset{1}{Y}_{jk}$ = An input vector containing a sampling-unit attribute (a data field
from the input-data matrix for a sampling unit)

P_j = The number of input-sampling units in the stratum or set

T = A set of rules whereby the elements of the input-data vector, $\overset{1}{Y}_{jk}$,
are redistributed (transformed) to form the output vector
(table), $\overset{0}{Y}_{jk}$

Procedure: $\overset{0}{Y}_{jk} = TY_{jk}$

Output: $\overset{0}{Y}_j = \frac{\sum_{k=1}^{P_j} \overset{0}{Y}_{jk}}{P_j}$

$$\overset{0}{VY}_j = \frac{\sum_{k=1}^{P_j} (\overset{0}{Y}_{jk} - \overset{0}{Y}_j)^2}{P_j (P_j - 1)}$$

$$\overset{0}{CV}_j = \frac{\sum_{k=1}^{P_j} (\overset{0}{Y}_{jk} - \overset{0}{Y}_j) (\overset{0}{y}_{jk} - \overset{0}{y}_j)}{P_j (P_j - 1)}$$

■