Research Program on the
Management of Science and Technology

DATANAL: AN INTERPRETIVE LANGUAGE FOR ON-LINE ANALYSIS OF EMPIRICAL DATA

JAMES R. MILLER

August 1967 275-67

"Work reported herein was supported (in part) by Project MAC, an M.I.T. research program sponsored by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research Contract Number Nonr-4102(01). Reproduction in whole or in part is permitted for any purpose of the United States Government."

"The research described in this paper was supported (in part) by NsG-235 of The National Aeronautics and Space Administration for research studies in the organization and management of research and development. The findings and views reported herein are those of the author and do not necessarily reflect those of the sponsoring agency."

HD28
M414
no.275-67
c.2

ABSTRACT

The purpose of this working paper is to describe an interpretive language for data analysis. The name of the language is DATANAL. It has been designed to facilitate:

1. analysis of any kind of empirical data collected in any context;

2. on-line conversational interaction between a user and selected portions of his data base through the medium of a time-shared computer;

3. two-way communication between a user and his computer in English (i.e., no additional programming languages need be learned); and

4. immediate usability by individuals relatively naive with respect to computers and their idiosyncrasies.

Discussion is focused upon how DATANAL appears to the user and how he might formulate and answer empirical questions within its framework. Less attention is given to the programming details underlying its operation. Hence, this should be regarded as a user's manual rather than as a programmer's manual.

At the time of this writing, DATANAL stands partially implemented on Project MAC's Compatible Time-Sharing System (CTSS). An earlier, less efficient but no less encompassing version of DATANAL was completed in June 1967. It is hoped that this final version will be finished by the end of 1967, at which time transfer to the Computation Center is fully anticipated.

TABLE OF CONTENTS

DATANAL: AN INTERPRETIVE LANGUAGE FOR

ON-LINE ANALYSIS OF EMPIRICAL DATA

1.0 INTRODUCTION

The purpose of this working paper is to describe an interpretive language for data analysis. The name of the language is DATANAL. It has been designed to satisfy four major objectives.

First, DATANAL is context-free. It may be used to analyze scientific data gathered in the course of some research project (e.g., thesis research). It may equally well be used to analyze administrative data as part of a management information system.

Second, DATANAL is conversational. The language is designed for on-line interaction between a user and selected portions of his data base through the medium of a time-shared computer.

Third, DATANAL is in English. It permits two-way communication between a user and his data in a sub-set of the English language. No additional programming languages need be learned.

Fourth, DATANAL is immediately usable. Extensive knowledge of computers and their idiosyncrasies is not required. Relatively naive users have mastered the basic commands of DATANAL with less than one hour of practice.

The focus of this report is upon the user--how DATANAL appears to him and how he might formulate and answer empirical questions within its framework. The basic commands of DATANAL will be discussed in detail,

along with some indication of how each command might be used. Less attention will be paid to the programming structure underlying DATANAL's operation.

2.0 STATEMENT OF THE PROBLEM

A formal statement of the problem which DATANAL has been designed to solve might appear as follows:

1. given an already designed and currently existing data base (e.g., thesis data punched up on cards);
2. given that the contents of the data base are hierarchically ordered (note: the meaning of "hierarchically ordered" will be spelled out in section 3.3);
3. given that the hierarchical ordering is perfect (note: the meaning of "perfect" will also be spelled out in section 3.3);
4. given a user who desires to answer some particular question;
5. given that the data base contains at least one datum relevant to the question posed;
6. then the problem is to devise a set of general procedures which will translate any well-formulated question into a sequence of operations to be performed upon the data base such that, after the sequence of operations has been performed and a result or set of results has been obtained, the user will regard such results as an answer to his question.

The above problem definition says two kinds of things. Items 1 through 5 above serve to exclude those problems which DATANAL has <u>not</u> been designed to solve. It is assumed that all of these have achieved

adequate resolution before the user attempts to converse with his data. On the other hand, item 6 delineates the essential nature of DATANAL--a set of general procedures to translate well-formulated questions into computer solutions. In order to preclude later confusion, some of the givens in this problem definition will be amplified briefly.

First, it is assumed that a data base has already been designed and coded. DATANAL does not provide any assistance in either of these respects.

Second, it is assumed that all data to be included in the data base have been gathered and transferred to some permanent, computer-readable storage medium (e.g., punched on cards, stored on tape, etc.). DATANAL provides no assistance either to the data gathering or to the data recording process.

Third, DATANAL does not identify users, nor does it identify sensible questions. It is assumed that users will identify themselves and will produce spontaneously their own criteria of sensibility.

Fourth, DATANAL cannot guarantee that a given data base will contain data relevant to every possible question. Only if, in the user's judgment, the data base does contain relevant data, and only if he can then formulate his question in terms of such data, only then can DATANAL provide an answer. The relevance of data to a given question must first be perceived and specified by the user himself.

3.0 THE CONCEPT OF A DATA BASE

A data base is assumed to consist of a number of (at least one) discretely identifiable physical entities, each one of which is characterized by a number of (at least one) discretely identifiable properties. Thus, if a data base contains the results of a questionnaire survey conducted among a sample of defense contractors, then the particular defense contractors in the sample would constitute discretely identifiable cases being observed, and responses to each item in the questionnaire would constitute properties associated with each case (contractor). Let us develop further these notions of cases and properties.

3.1 Cases: The Basic Units of Analysis

Whenever empirical data are gathered, some physical process must occur in which observations are made on selected attributes of concrete physical entities. In the example above, the process involved responding to a questionnaire, and the objects of interest were the defense contractors (i.e., business organizations) in the survey sample.

It sometimes happens, however, that observations are made on physical entities which are not static objects at all. They are instead well-delineated sets of activities. A mission flown against the enemy

or a project to reach the moon are good examples. In order to permit this latter kind of entity to become a basic unit of analysis in DATANAL, and in order to avoid confused interpretations, the more general rubric "cases" will be used instead of "objects." A case can be anything the analyst chooses to regard as his basic unit of analysis so long as:

1. it is something physically identifiable; and
2. it adheres to certain rules of hierarchical ordering to be discussed in section 3.3.

3.2 Properties: The Basic Categories of Discussion

Having defined and delineated the various cases for inclusion in a data base, the next logical step is to establish basic conceptual categories in terms of which these cases may be discussed. Such categories of discussion and analysis are called properties. The responses to items in a questionnaire survey, the damage inflicted by a combat mission, and the date when the moon is reached all constitute properties of a responding contractor, a particular combat mission, and the overall mission of NASA, respectively. The contents of an empirical data base (i.e., the essential nature of empirical data) are observations of whether or not and possibly the degree to which each included case possesses various defined properties.

3.3 Hierarchical Ordering

Not all of the properties (e.g., responses to questionnaire items) need refer to cases at the same level of organization. Some properties may refer to the entire population (industry) of which each case (defense contractor) is a member. Other properties may refer to a particular case (contractor), and still other properties may refer to subunits of each case (departments or project groups within the contractor's corporate structure). It is in this sense of logical inclusion that both cases and properties referring thereto may be hierarchically ordered.

Stated more formally, one class of cases in a data base is said to occupy a higher position in the hierarchy than another if and only if a case from the first class is capable of containing a case from the second class as a member. Similarly, one property is said to occupy a higher position than another if and only if it refers to a higher-level case.

An immediate implication of the above definition is that all properties may be imputed downward to each lower-level case contained as a member, although upward imputation is not always possible. Thus, all defense contractors in our hypothetical questionnaire survey share certain properties related to the industry of which they are members (e.g., they all perform defense work, they all receive direct or indirect funding from the federal government, etc.), but the entire defense industry is not uniformly engaged in, let us say, missile production.

Another concept worth introducing at this juncture is perfect ordering. The contents of a data base are said to be perfectly ordered if and only if all the membership relationships between each pair of cases are either absent or assymetric. That is, given any pair of cases, either there is no membership relationship between them (i.e., neither contains the other or any part thereof as a member) or there is a membership relationship defined, and it is unilateral (i.e., the higher-level case contains the lower-level case as a member, but the lower-level case does not contain the higher-level case). An example of proper ordering would be a defense contractor who belongs to the defense industry, but who does not contain the entire defense industry within one of his project groups.

In the discussion that follows, we shall only consider perfectly ordered data bases. Also, uniform numbering conventions will be adopted such that the lowest level in a data base will be designated level one, the next-higher level will be designated level two, and so on. In addition, there will be established a "level zero" to accommodate "global" properties. These will be discussed in section 3.5.

3.4 Matrix Representation

Now that we have introduced the twin concepts of cases and properties, we may envision any data base as if it were a rectangular table or

matrix. The rows of the matrix are occupied by physical cases, and the columns are occupied by properties. Hence, a complete description of a given case would require enumerating all of the observations along its row of the data matrix, while a complete description of a given property would require enumerating all observations down its column.

If the data base contains more than one explicitly defined level, then we may envision separate matrices for each level. Complete separation permits each level to have its own number of cases and properties and yet still be represented in matrix form. However, when this occurs, cross-level linkages must be provided to indicate the higher-level case to which each lower-level case belongs. The introduction of level-linkage identity tags as explicit properties of each case will mediate complete identification (note: these level-linkage tags must appear as raw data within the data base).

3.5 Global Properties: Level Zero

It is convenient to characterize every data base as possessing an external level over and above however many levels are explicitly defined within. We shall refer to this level as the global level, and we shall refer to all its properties as global properties. To distinguish it numerically from explicitly defined levels within a data base, we shall henceforward refer to the global level as level zero.

One type of global property refers to the whole data base as a physical entity. Common characteristics of the data gathering process, the dates when data were gathered, and the purposes for gathering data illustrate this type of global property. It is rare that such properties achieve explicit representation within the data base (e.g., punches in specified card columns), but they are important. They are particularly relevant to the process of interpreting results obtained from the data base and generalizing conclusions drawn therefrom.

There exists another kind of global property which may be computed from individual property values. Thus, the mean value of any property (averaged over all cases on a given level) constitutes a global property. The same statement applies to a median, frequency count, or any other computed summary measure. Computed summary measures may always be imputed downward to cases on the level of computation, as well as to all lower-level cases. However, such properties may not, in general, be imputed to higher-level cases. Only if the summarizing computations are particularized for each higher-level case (e.g., a conditional mean is computed over all lower-level cases belonging to a single higher-level case) can upward imputation occur; for only then is a unique property value defined for each higher-level case.

A third kind of global property is a pure constant (e.g., the value of π) entered into the data base for analytical purposes. Reasons for entering constants into the data base will be discussed later.

4.0 THE CONCEPT OF AN EMPIRICAL QUESTION

An empirical question is here being defined as any interrogative statement referring to:

1. either a sub-set of the contents of a data base (e.g., "what were the responses to all questions answered by company X in the questionnaire survey?");
2. or statistics computed from a specified sub-set of the contents of a data base (e.g., "what was the mean response on item 37 in the questionnaire survey?");
3. or relationships among specified sub-sets of the contents of a data base (e.g., "was company X's response on item 37 higher than company Y's?");
4. or relationships among statistics computed from a specified sub-set of the contents of a data base (e.g., "was company X's mean reported project group performance higher than company Y's?").

An empirical question is really an interrogative form of an empirical hypothesis. As such, there are certain rules which must be adhered to before a candidate query may be called a well-formulated empirical question.

4.1 Rules for Constructing a Well-Formulated Empirical Question

First, a question must be phrased so as to be neither logically true (a tautology) nor logically false (a logical contradiction or inconsistency). An example of a tautological question would be, "Are all bachelors unmarried?" This is a tautology, since the definitional meaning of "bachelor" is "unmarried male," which makes the answer logically true. An example of a logically false question would be, "Is it possible for human beings to survive in a climate where no mammals can survive?"

Second, a question must be empirically testable with respect to the possible contents of the data base (i.e., there must exist at least one logically possible datum within the data base such that, if such data in fact exist, the question will be answered yes; and there must exist at least one other logically possible datum such that, if such data in fact exist, the question will be answered no).

One major purpose of DATANAL is to induce users to ask only well-formulated questions.

4.2 An Example

A reasonable empirical question which might be asked of the data base resulting from our hypothetical questionnaire survey appears below.

"Do project groups in company X tend to perform better, on the average, than groups in company Y?"

As stated, this question is in principle untestable. It fails to make clear what constitutes "project group," "better performance," or "average," to mention just a few terms. Consequently, it would not qualify as well-formulated. However, the situation can be repaired.

Let us now operationalize some of the above terms. Let us define "project group" as meaning whatever the respondent stated in his answer to item 32 of the questionnaire. Let us define "performance" as his response to item 37. Let us define "better performance" as a numerically higher response to item 37. Let us define "average performance" as an arithmetic mean of the responses to item 37 computed over all project groups (indicated by item 32) within a given company. Let us also formulate an acceptance criterion in terms of the results of a T-test. Let us agree to give an affirmative answer to the above question if and only if both of the following conditions are satisfied.

1. The mean performance of project groups in company X exceeds the mean performance of project groups in company Y.
2. The probability of occurrence of whatever value of T emerges from the analysis falls at or below .05.

Now the question is well formulated. Every term is defined operationally and with respect to the possible contents of the data base. A second purpose of DATANAL is both to require and to facilitate the

operationalization of empirical questions, as well as to compute and display answers.

4.3 The Process of Answering a Well-Formulated Empirical Question

The general process of answering a well-formulated empirical question might be broken down into the following four sequential phases.

1. Extract and/or compute from the contents of the data base whatever specific properties are defined operationally by the well-formulated question.
2. Select one or more specific analytical and/or display procedures to be performed upon the operationally defined properties. A well-formulated question identifies precisely which procedures are relevant.
3. Arrange the extracted/computed operational properties in a form compatible with the selected procedure(s).
4. Execute the procedure(s), and display the results.

Returning to the sample question formulated in section 4.2, this process might proceed as follows.

1. Identify all of the project groups associated with companies X and Y, respectively (by inspecting responses to item 32). Separate, thereby, two sub-samples of performance data (responses to item 37). The first sub-sample will contain

performances of company X project groups. The second will contain performances of company Y project groups.

2. Select a pre-compiled T-test routine from the DATANAL library. Its inputs include two variable size sub-samples of interval scale numbers. Its outputs include sub-sample means, the computed value of T, and an associated probability of occurrence.
3. Extract the sub-sample data from the data base, and re-package it in the form of two variable-length input vectors.
4. Transmit the two input vectors to the T-test routine, execute the routine, and display the results.

Whether the original question receives an affirmative answer would now depend upon the outputs of the T-test routine. If the project group mean achieved by company X exceeds company Y's mean, and if the associated probability of occurrence falls at or below .05, then an affirmative answer is appropriate. Otherwise, either a negative answer or a decision to suspend judgment pending further investigation would be appropriate.

5.0 THE CONCEPT OF A WORKING DATA BASE

Of central importance to DATANAL is the concept of a working data base. It is through this medium that DATANAL achieves much of its computing power, its analytical flexibility, and its convenience to the user. The remainder of this section will set forth the role played by a working data base in accomplishing these ends.

5.1 Its Purpose

The major purpose in creating a working data base is to provide a physically separate and completely self-contained collection of basic instructions, descriptive information, and data associated with an empirical question or set of closely related empirical questions. The conceptual unit here is the question itself or, as it is sometimes called, the working hypothesis. It will be recalled from section 4.3 that answering a well-formulated empirical question requires:

1. extracting whichever properties are relevant to answering the question from a raw data base;
2. selecting specific analytical and/or display procedures to operate on these properties;
3. arranging property data to serve as legitimate inputs to the selected procedures; and

4. executing the procedures and displaying the results.

All of these operations are performed and fully documented within the confines of a working data base.

5.2 Its Structure

Every working data base contains two kinds of information. First, there is descriptive information about the structure and contents of the data base itself. Second, there is the actual data either extracted from a raw data base or created internally by executing various analytical procedures.

Descriptive information is deposited on a single string of binary-packed words called the "P-string" ("P" stands for property). A P-string may be as long as 4096 words but will normally be less than 1000. Since it is binary-packed, a P-string is never displayed to the user in raw form. It must be interpreted by special programs, which are part of the DATANAL language.

Every P-string contains the following information describing the structure and contents of the working data base:

1. the name of the raw data base from which this working data base was created (the process of creating a working data base will be discussed in section 7);
2. the number of levels defined in the working data base;

3. the number of cases on each level;
4. the number of unit records (e.g., punch cards) required to contain all property data associated with each case on each level;
5. the number of blocks of data contained in this working data base (the meaning of a data block will be explained shortly);
6. the number of data words contained in each data block;
7. the number of currently defined properties;
8. for each property -
    a. its name or label,
    b. its origin (i.e., how it was originally defined),
    c. its property type (i.e., scalar, vector, or matrix),
    d. its dimensions (i.e., number of rows and columns),
    e. its level (i.e., the level of the cases to which it refers),
    f. whether or not the corresponding property data have been created and, if so, where such data are located,
    g. whether the definition of the property is temporary (i.e., may be erased) or permanent (i.e., may not be erased),
    h. whether the corresponding property data are temporary (i.e., may be destroyed) or permanent (i.e., may not be destroyed),
    i. an optional verbal description of its meaning or significance to the user (the user must compose such a description and type it into the console),

j. an operational definition indicating how the corresponding property data are computed (DATANAL composes, stores, and interprets this information internally).

The actual data are stored on blocks of up to 4096 words each. There can be approximately 500 blocks associated with every working data base. This means that a single working data base can accommodate approximately 200,000 data words, so long as sufficient disk space remains available for storage.

A single data block contains all of the data associated with at least one property. As mentioned previously, part of the definition of every property is an address pointer indicating the particular block containing that data, if such data exist.

P-strings and data blocks exist as permanent files on disk storage. Whenever any work is performed by one of the commands in DATANAL, the entire P-string and whichever data blocks are relevant get moved into core, updated, and returned to disk. In this manner, a working data base is always maintained up-to-date.

5.3 Its Implications for Preparing Raw Data

Inherent in the programming structure of DATANAL are several types of capacity limitations. These limitations in fact apply to operations on a working data base, but their primary significance (from the user's

point of view) relates to the preparation of raw data. Hence, a list of rules for preparing a raw data base has been compiled and appears below. Why some of these rules apply cannot be understood without a detailed knowledge of how DATANAL has been programmed. However, that they apply will become immediately apparent to any serious user.

1. No more than 10 levels of cases are permitted.
2. The hierarchical ordering across all levels must be perfect and must be explicitly indicated by level-linkage identity tags, rules for which are stated in 9 (see below).
3. No more than 4096 cases are permitted on a single level.
4. When preparing a data deck or tape for transfer onto disk, all of the unit records (e.g., cards) associated with all of the cases on a single level must be placed together in a block. There will be as many such blocks as there are levels.
5. The order of cases within each level block is arbitrary, but the order of level blocks must be ascending. That is, all of the data (e.g., cards) associated with level one (the lowest level) must precede all data associated with level two, which must precede level three, and so forth.
6. There is no limit to the number of unit records (e.g., cards) required to contain complete property data for each case, nor is there a limit to the number of properties per case.

7. However, if multiple records are used to describe a case, then the ordering of these multiple records within each case must be fixed. That is, the first unit record (e.g., card) must always precede the second, and so forth.

8. The arrangement of data within each unit record must adhere to fixed-field formatting conventions. The formatting conventions must be identical for all records (e.g., cards) of a given type, but may be different for different types. Thus, all of the i^{th} cards associated with each case on the j^{th} level must share the same fixed-field format, but this format may differ from other cards on the same level and/or different levels. No field on any record may exceed 12 characters (e.g., card columns) in width.

9. Each case must possess at least one explicit level-linkage identity tag. There must be as many tags associated with each case as there are levels at and above the level on which that case resides. Thus, if there are three levels in the raw data base, all first-level cases must possess three tags, all second-level cases must possess two tags, and all top-level cases must possess one tag. The total number of such tags, including all levels, is given by the expression $\sum_{i=1}^{N} n_i \, (N-i+1)$, where N is the number of levels and n_i is the number of cases on the i^{th} level. This total cannot exceed 4096. Each tag must appear on the first record (e.g., card) associated with

each case and, if multiple tags exist, must be entered in contiguous four-column fields, starting in column one. A unique, although not necessarily serially ordered nor justified tag must be assigned to each case on each level, and the tag of every higher-level case to which each case belongs must also be assigned. The order of tags in successive four-column fields must be as follows:

a. a tag to distinguish each case from all other cases on the same level (columns 1-4);

b. the tag of the case on the next-higher level (if one exists) to which each case belongs (columns 5-8);

c. the tag of the case two levels higher (if such a level exists) to which each case belongs (columns 9-12);

d. and so forth.

10. Although DATANAL will read nonnumeric data (i.e., it will not terminate upon encountering a nonnumeric character or improperly presented number), it will only interpret legal numbers. Both integers and decimal numbers with explicit decimal points are considered legal, but exponential or any more esoteric form of notation is not. Anything other than an integer or normal decimal number will be ignored by DATANAL and treated as a "missing" or nonexistent observation. The property value for that case will be undefined. In particular, blank fields are

considered nonnumeric by DATANAL (i.e., blanks are not interpreted as zeros). Hence, a blank field may serve (and, in fact, frequently does serve) as a missing data code.

5.4 Its Basic Philosophy

As stated previously, a working data base provides a concrete focus of attention in formulating and testing empirical hypotheses. This is its primary role. A working data base actually embodies such a question or hypothesis in the sense that all descriptive information relevant to its formulation and all basic instructions required to provide an answer are stored on the P-string. In addition, whatever raw data have been read in and whatever new data have been created are stored on the data blocks. However, not all of the actual data need be stored in order to obtain an answer. This is because the basic instuctions stored on the P-string, in conjunction with the CREATE command to be discussed in section 7, provide the capability of creating (or recreating) any property data for which an operational definition exists. This point is so central both to the philosophy and to the operation of DATANAL that it deserves amplification.

Perhaps the best way to begin is with an analogy from nature. The process by which all living organisms are created is directed by a combination of genetic and biological synthesizing mechanisms. The

instructions for this process are stored in highly coded form within strings of genetic material called chromosomes. Each chromosome contains numerous sub-strings called genes. Now these genes do not themselves combine to form living matter. Instead, they provide direction to external synthesizing mechanisms which, in turn, combine raw materials (e.g., amino acids) into intermediate sub-cellular building blocks (e.g., protein molecules), building blocks into cells, cells into component organs (e.g., arms and legs), and component organs into finished organisms (e.g., complete human beings). The three basic elements in this entire process are:

1. the original instructions contained in genetic code on the chromosomes;
2. the various synthesizing mechanisms which take their instructions from the chromosomes and which operate on raw and intermediate biological materials to form complete organisms; and
3. the raw materials out of which cells and more complete structures are formed.

DATANAL mimics this natural process very closely. The P-string plays the role of a chromosome, with each individual property definition playing the role of a single gene. Certain working programs (principally the CREATE command) play the role of biological synthesizing mechanisms. Raw data contained in the raw data base play the role of amino acids and other elementary biological materials. Intermediate results

computed out of raw data in the process of answering an empirical question play the role of component organs. Finally, an entire working data base, containing all property definitions and corresponding property data required to generate an answer to a given empirical question, plays the role of a complete organism of a given species.

The importance of the above analogy is three-fold. First, the internal programming structure of DATANAL bears close resemblance to its biological counterpart. In fact, DATANAL was explicitly designed with the genetic process in mind.

Second, recalling the biological analogy may help the user to understand the dramatic separation within DATANAL of property-defining and property-describing procedures on the one hand from property-creating and data-manipulating procedures on the other hand.

Third, and perhaps most important, DATANAL, like the genetic process, is primarily formulation-oriented and only secondarily execution-oriented. The principle role of the genetic process is to formulate and direct a sequence of biological procedures. Involvement in the actual execution of these procedures is limited, indirect, and of secondary importance. Similarly, DATANAL is oriented primarily toward formulating question-answering and hypothesis-testing procedures. It is only secondarily a means of executing such procedures, once formulated. The ability to concentrate upon formulating question-answering and hypothesis-testing procedures, with confidence that any well-formulated

procedure may be executed to obtain an answer, should permit many more and potentially better solutions to analytical problems. At least, this is the basic premise upon which DATANAL rests.

5.5 Some of Its Operating Consequences

The structural separation of a raw data base from its associated working data bases has several important operating consequences. First, this separation permits a user both to operate upon numerous raw data bases simultaneously and to create multiple working data bases out of each distinct raw data base. Great analytical flexibility is thereby realized. However, since most of the commands in DATANAL are oriented toward a particular working data base, this added flexibility is not purchased at the expense of stringent capacity limitations. The liberal capacity limitations stated on sections 5.2 and 5.3 still apply. Also, several tools are provided within DATANAL to remind the user of the various connections he has established among various data bases. These include:

1. the ability to assign a symbolic name to each working data base suggesting the particular question (hypothesis) which it was designed to answer (test);
2. special commands which permit free movement from one working data base to another; and

3. Immediate access to the name of and to complete structural information about the raw data base from which each working data base was created.

A second consequence flows from the physical discreteness of a working data base. Since it is stored on disk in physically separate files, it may be removed at any time, either temporarily or permanently, to a less expensive storage medium. Any working data base may be punched out on cards or stored on tape whenever disk space becomes limited. This serves both to free up disk space for additional analyses and to reduce storage costs for the results of analyses already performed. Furthermore, should a user wish to return to that data base at some future time, he may replace it on disk and re-open it. Recall that the P-string contains an always up-to-date description of its structure and contents. Hence, a user may review all of the operations previously performed to read in and manipulate property data and proceed from there. It is as if the data base had never been removed. The user is right back to the point where his last operation was completed.

A third consequence is the relatively great security provided against machine breakdown. Since DATANAL is oriented toward a working data base, and since both P-strings and data blocks are updated after every command and stored on disk, the effect of a machine breakdown is rarely serious. Unless physical damage is done to the disk, the worst that can happen is to nullify whichever operation is currently being

performed and to return the data base to its status following the last completed command. The entire day's work is not lost, as is usually the case with core-oriented analysis systems.

A fourth consequence is the complete buffering provided by a working data base between raw data and the various commands of DATANAL. Raw data are never altered. Data are first read selectively into a working data base and then operated upon. In this manner, the integrity of raw data is always preserved, but without any loss in flexibility.

A fifth consequence flows also from the complete buffering provided. It is possible to group properties in various ways so as to form vectors and matrices in a working data base, even though the original data existed in scalar form in the corresponding raw data base. More will be said about this very important feature of DATANAL in section 7.

6.0 THE SUPERVISOR

DATANAL is operated under the constant control of a special supervisor. The major purpose of this supervisor is to stand as a buffer between the user and the overall time-sharing monitor and to manage all interactions both within DATANAL and between DATANAL and all other routines available under time-sharing.

To initiate DATANAL, the special supervisor must be loaded into core. This is accomplished by issuing the command "RESUME DATANAL." The special supervisor will then maintain control of all subsequent interactions until either the user decides to terminate DATANAL voluntarily (by issuing the command "QUIT") or a fatal error occurs. In either case, control will be returned to the time-sharing monitor.

When first loaded, the special supervisor proceeds immediately to clear out any commands or fragments thereof remaining from previous interactions. This reduces the probability of incorrectly interpreting information from a previous command as relevant to the current command. It then reads a single typed line from the console.

The first word on each console line is interpreted by the supervisor as a command word. After comparing it with an internal list, each command word is interpreted as signifying one of the following three types of commands:

1. an internal supervisor command;

2. a command external to the supervisor but included within DATANAL;

3. neither of the above--either a command recognized by the time-sharing monitor or an error.

If an internal supervisor command is identified, control is maintained by the supervisor, and that command is executed. Following successful execution, the supervisor will print the message "OK" on the console, possibly accompanied by some timing information. The "OK" message indicates that the current command has been executed normally and that the supervisor is ready to read the next command line.

If an external DATANAL command is identified, control is passed temporarily to the external program embodying that command (i.e., the external program replaces the supervisor in core). Control will return automatically to the supervisor following its successful execution, and the "OK" message will appear as before.

If the command word is not recognized by the supervisor, control is passed conditionally to the time-sharing monitor in hopes that it may still be legitimate. If it is, appropriate action is taken under control of the time-sharing monitor, and, if all goes well, control will return to the special supervisor upon successful completion of that action. Return of control will be indicated by the usual "OK" message. However, if either the command word is unrecognized by the time-sharing monitor or an error occurs while under the control of the time-sharing

monitor, the special supervisor will not regain control automatically. This condition will be indicated by the absence of an "OK" message and the substitution of whatever diagnostic message the time-sharing monitor deems appropriate. When this occurs, the special supervisor must be re-loaded manually into core just as it was at the initiation of DATANAL.

One important consequence of these control-passing activities is that a user may move freely in and out of DATANAL, and, while in DATANAL, he need not sacrifice access to any commands available under the time-sharing monitor.

The remainder of this section will present a detailed description of the internal supervisor commands within DATANAL. Each command will be described in terms of its purpose, its format, its effects, and any diagnostic information it may provide.

6.1 The ATTACH Command

The ATTACH command is used to update the supervisor's internal list of recognized command names. DATANAL has been designed so that users may invent their own private commands and may attach these to "their version" of the system. Detailed instructions for preparing private commands will be postponed until section 9. However, once a private command has been prepared, it is entered into DATANAL by issuing the ATTACH command.

The ATTACH command is issued as follows.

ATTACH NAME1 NAME2 ... NAMEN

NAME1, NAME2, ..., NAMEN are the names which will be given to the new commands. Any name involving up to six BCD characters is legal, so long as no other command already exists under the same name. To determine the names of existing commands, the user may type "LIST COMMANDS." As many command names as can be typed on a single console line (each name separated by at least one blank space) may be entered with a single ATTACH command. The supervisor can handle up to 430 command names in all.

The effect of issuing the ATTACH command is to alter a special disk file called "CURENT COMNDS" so that it will include the newly attached command names. The CURENT COMNDS file is created by DATANAL the first time a user loads the special supervisor. This initial version of CURENT COMNDS contains the names of all commands mentioned in this report. Subsequent ATTACH commands (and certain other internal commands to be discussed shortly) alter the CURENT COMNDS file so that the supervisor will always be up-to-date.

An error message is issued whenever a user attempts to attach more than 430 command names to the supervisor's internal list.

6.2 The REMOVE Command

The REMOVE command is the antithesis of the ATTACH command. It serves to remove various command names from the supervisor's internal list. Such removals might be useful either to make room for additional command names (if the limit of 430 has been reached), to change the official name of an existing command, or to permit a more efficient search through the supervisor's internal list by deleting and/or resequencing some of them.

The REMOVE command is issued as follows.

```
REMOVE NAME1 NAME2 ... NAMEN
```

NAME1, NAME2, ..., NAMEN are the command names to be removed from the internal list. As many names as can be typed on a single console line (each name separated by at least one blank space) may be entered with a single REMOVE command. Should a user attempt to remove one or more non-existent command names, these requests will be ignored without damage and without any error message.

The effect of issuing the REMOVE command is to alter the CURENT COMNDS file--just like the ATTACH command.

There are no diagnostic messages associated with the REMOVE command.

6.3 The TIMON Command

The special supervisor is equipped with an optional capability of

providing either continuous or intermittent information to the user concerning how much computer time he has consumed. Exercise of this option is controlled by selective issuance of the TIMON, the TIMOFF, the RESET, and the QUIT commands.

The TIMON command is issued as follows.

TIMON (no additional parameters)

Should the user enter any additional parameters by mistake, these will be ignored without damage and without an error message.

The effect of issuing the TIMON command is to initiate the internal timing mechanism. This is accomplished by updating the CURENT COMNDS file--just like the ATTACH and REMOVE commands.

When the TIMON command is first issued after either the initial loading of DATANAL or a subsequent reloading of DATANAL (provided that a normal exit was made from the previous loading via the QUIT command) or after any issuance of the RESET command, the internal timing mechanism is both turned on and initialized to zero. However, no effect is apparent to the user until the next command is issued. Then, until the first issuance of the TIMOFF command, each subsequent command will be followed by an "OK TIME1 TIME2" message instead of the usual "OK." TIME1 is the incremental time in seconds and tenths of a second required to complete the current command. TIME2 is the cumulative time in seconds and tenths of a second used since either the initial loading of DATANAL, the last issuance of the QUIT command, or the last issuance of the RESET command,

whichever occurred most recently. Both TIME1 and TIME2 constitute aggregates of execution and swapping time, since it is these aggregate times which represent the total resource drain imposed by the user upon the computer.

Incremental and cumulative times will continue to appear with each "OK" message until the TIMOFF command is issued. When this occurs, the timing mechanism is turned off and printing of both incremental and cumulative times is suppressed. However, the timing mechanism is not initialized. The user may query the supervisor at any moment by re-issuing the TIMON command. This will cause both incremental and cumulative times to be reinstated in the "OK" message. The incremental time associated with re-issuing the TIMON command has a special interpretation. It includes all time consumed since the most recent TIMOFF command was issued. The corresponding cumulative time includes all time consumed since the first TIMON command was issued. Incremental and cumulative times associated with all subsequent commands have their usual interpretation.

6.4 The TIMOFF Command

The purpose of the TIMOFF command is to suspend operation of the internal timing mechanism and to suppress printing of both incremental and cumulative times in the "OK" message. Since the timing mechanism

itself consumes about one-half second per command, it should be turned off unless the user is really interested in timing himself.

The TIMOFF command is issued as follows.

```
TIMOFF (no additional parameters)
```

Should the user enter additional parameters by mistake, these will be ignored without damage and without an error message.

The effect of issuing the TIMOFF command was discussed in section 6.3. TIMOFF also updates the CURENT COMNDS file.

6.5 The RESET Command

The purpose of the RESET command is to initialize the timing mechanism (i.e., set incremental and cumulative times to zero) without turning it off and without terminating DATANAL.

The RESET command is issued as follows.

```
RESET (no additional parameters)
```

Should additional parameters be entered by mistake, they will be ignored without damage and without an error message.

The effect of issuing RESET is to zero out both incremental and cumulative times. This does not involve any changes to the CURENT COMNDS file. Incremental and cululative times will continue to be printed out with each subsequent "OK" message, but cumulative times will now include only that amount of time used since issuing RESET. Incidentally, incremental and cumulative times will be automatically re-set whenever a user crosses a shift barrier (e.g., crosses from the first to the second shift at 5:00 p.m. on any week-day).

Although the TIMON, TIMOFF, and RESET commands are designed to be issued in certain regular sequences, no damage will result from issuing them out of sequence. Thus, two successive issues of TIMOFF will be wasteful of computer time, but will not damage the internal timing mechanism.

6.6 The QUIT Command

The QUIT command is issued to terminate a session with DATANAL and to initialize the internal timing mechanism.

The QUIT command is issued as follows.

```
QUIT (no additional parameters)
```

Should additional parameters be entered by mistake, they will be ignored without damage and without an error message.

The effect of issuing the QUIT command is to re-set both incremental and cumulative times to zero and to return control to the time-sharing monitor. However, the internal timing mechanism is not turned off (i.e., no changes are made to the CURENT COMNDS file).

6.7 Certain SYNONYM Commands

The supervisor also provides for using different command names to call essentially the same procedure. This feature is useful for translating automatically what a user may regard as separate procedures from the analytical point of view into what the computer has been programmed to regard as a single procedure. Thus, a user may distinguish conceptually between performing a Chi square test of homogeneity, a Chi square test of independence, a Fisher exact test of either homogeneity or independence, and a median test. However, all five of these tests are extremely similar from a mathematical and computational point of view. For this reason, they have all been combined into a single procedure under the name CNTING (meaning contingency table analysis), which permits considerable programming efficiency and saving in disk storage space. Reference to this procedure can then be made by issuing any one of the following commands:

1. CNTING (plus parameters);
2. HOMNOM (plus parameters);

3. FISHER (plus parameters);

4. MEDTST (plus parameters).

The use of these procedures and their accompanying parameters will be explained more fully in section 8.

Synonym relationships are maintained within the core image of the supervisor itself. Each command line is intercepted and inspected for synonyms prior to calling an external program. Hence, changing established synonym relationships would require re-coding the supervisor by hand (in assembly language) and re-assembling the re-coded version.

7.0 THE EXTERNAL COMMANDS OF DATANAL

Beside the internal supervisor commands discussed in section 6, DATANAL consists of six types of external commands. These are:

1. commands to define, describe and create a working data base;
2. commands to define and describe properties within a working data base;
3. commands to implement property definitions (i.e., to execute the instructions comprising an operational definition of a property and to create thereby the corresponding property data);
4. commands to summarize and display property data in certain convenient ways (e.g., in terms of means, standard deviations, frequency counts, cross-tabulations, etc.);
5. commands to perform standard statistical analyses upon property data (e.g., contingency analyses, ordinal and cardinal correlation analyses, regression analyses, etc.); and
6. private commands prepared by a user and attached to "his version" of DATANAL.

The remainder of this section will discuss all but the statistical and private commands. These topics will constitute the subjects of sections 8 and 9, respectively.

7.1 Universal Operating Features

Before launching into a separate discussion of each command, it would be useful at this point to set forth some of the general features common to all commands. These universal operating features are described below.

7.1.1 Typing Errors

Typing errors are to be expected, even by frequent and professional users of DATANAL. For naive and infrequent users, they will constitute a major source of irritation. Obviously, the best remedy is experience and practice, whose effect will be to reduce both the frequency and the seriousness of such errors. In addition, several specific devices have been provided to mitigate their consequences.

The first device is a kind of "electronic eraser" which operates in three separate forms. Whenever a new command is issued, DATANAL automatically clears out any information remaining from previous commands--so long as that information was entered with a carriage return. (Note: Merely typing information on the console without pushing a carriage return will not transmit the message to the computer. A carriage return is essential at the end of every typed line. Failure to push the carriage return is one of the most common mistakes made by naive users.) This automatic clearing device is also present within those commands of DATANAL which require multiple lines of typed input (e.g., the DEFINE command).

However, typing errors are frequently detected by visual inspection before an input line has been completed and entered. Under these conditions, either of two kinds of "electronic erasure" may be effected manually. The quotation mark (") serves as a single character eraser and will wipe out the immediately preceding character. Multiple quotation marks will wipe out successive characters, counting backwards from the current character position. The question mark (?) serves as a complete line eraser. It will wipe out the entire contents of the line up to the point where it is entered. Multiple question marks are redundant, but nondamaging. After either type of erasure has been effected, typing may continue normally on the same line, just as if the erased portions had never existed.

The question mark provides a useful retort to occasionally incomprehensible diagnostics. Many of the commands of DATANAL require multiple lines of input. To expedite the input process, these commands have been programmed so that the computer will read, interpret, and check previously entered input lines while the user is typing subsequent lines. If an error is discovered while the user is typing, the computer will interrupt with a diagnostic message (recall that the computer cannot tell whether typing is currently going on until the carriage return is pushed). This leaves an unfinished piece of an input line to be interpreted by the next read cycle. Invariably, it will be misinterpreted. This frequently gives rise to an incomprehensible diagnostic.

To avoid the entire problem, the next typed line should start with a question mark.

A second device is the presence of extensive interpretive and diagnostic logic within all commands of DATANAL. A great deal of effort has been expended to incorporate specific and detailed error diagnostics, along with recovery procedures. In fact, many more computer instructions are dedicated to performing these kinds of functions than are involved in making actual computations.

A third device is the ability to abort any command at any point in its progress. This will be discussed more fully in section 7.1.6.

7.1.2 Incomplete Specifications

All commands will accept incomplete specifications. Only the command word itself (e.g., "ATTACH") is required to transfer control from the special supervisor to the appropriate command. This makes it possible for a naive or infrequent user to initiate any command without really knowing how to complete it. It also provides an explicit mechanism for obtaining guidance on issuing a command at the same time that the command is being issued. Guidance may be obtained by issuing only the command name. If that cannot be remembered, entering "LIST COMMANDS" will cause a complete list of currently defined command names to appear on the console.

In the face of an incomplete specification, every command will respond with detailed typing instructions. These will lead the user

through the remainder of the command specification on a word-by-word basis. However, any partially completed portion of a command need not be re-typed, so long as it was legal and properly ordered.

7.1.3 <u>Illegal Specifications</u>

Almost all commands require that certain logical and syntactic rules be adhered to in their specification. Whenever possible, violations are treated individually, and only the offending pieces of information need be corrected and re-submitted. If impossible, the entire specification must be repeated. In either case, a specific diagnostic message and re-submission request will appear on the console.

7.1.4 <u>Rapid vs. Guided Input Modes</u>

As suggested by previous discussion, all commands accept input specifications in either of two modes. The primary mode is completely devoid of any interaction with the computer. Hence, it is called the rapid mode. The secondary mode involves typing instructions and/or error diagnostics provided on an interactive basis. Hence, it is called the guided mode.

The user is placed in the rapid mode at the beginning of each command. He will stay in that mode so long as he provides a specification which is:

1. complete;
2. properly ordered;
3. legal in syntax; and
4. legal in content.

Incomplete specifications will cause an immediate shift to the guided mode for the incompleted portions. Improper ordering will require a complete respecification, sometimes in the rapid mode, sometimes in the guided mode, depending upon how many errors have been detected. Other syntax errors are treated similarly. Illegal contents are treated on an individual basis, wherever possible. No mode change will occur, unless the number of errors becomes excessive.

7.1.5 Error Counting

Every command records and counts the number of specification errors made by a user. As this number increases, the user is more likely to be thrown into the guided input mode (the exact conditions for such a mode change vary from command to command). However, if three errors occur in any command, that command is automatically aborted, and control is returned to the special supervisor (indicated by the usual "OK" message). If a legal specification is achieved before committing three errors, whether in the rapid or in the guided mode, the command will be executed, and an explicit message to that effect will appear, followed by "OK."

This automatic abort feature is particularly useful in limiting the duration of error loops (i.e., no more than three go-arounds). If an error loop is induced by confusion on the user's part (which is usually the case), its efficacy is mostly psychological. If the computer has become hopelessly confused, it serves to initialize machine conditions.

7.1.6 Manual Abort

It is also possible at any time in the course of any command to effect a manual abort. This may be accomplished by pushing the "break," "quit," or "interrupt" button, as it is variously called, either once or twice. (Note: This button is located somewhere on the face of the console depending upon the particular model being used). A single push will return control immediately to the DATANAL supervisor and will cause the usual "OK" message to be printed out. Two pushes will return control to the time-sharing monitor. The ability to effect a manual abort, particularly the single-push variety, is useful under any one of the following circumstances.

1. The user decides in the middle of a command that he does not wish to complete that command at all. He wishes to issue another command or the same command in a different manner.
2. The user has requested and received specific information from the computer, but it continues to pour out additional, irrelevant information (e.g., a command name is located by issuing "LIST COMMANDS" before the entire command list has been exhausted).
3. The user has made one or two specification errors within a command. The computer has therefore concluded that he needs assistance and has switched him into the guided

input mode. Meanwhile, the user has recalled the correct specification format and does not wish to be led by the hand (always a slow and torturous, although safe process). Consequently, he effects a manual abort and starts over. This causes the computer to "forget" how many mistakes he made and to reinstate the rapid input mode.

4. An unusually long response delay on the part of the computer has occurred (e.g., several minutes have elapsed without any action). The user wishes to determine the cause of the delay. Possible causes are:
   (a) failure to push the carriage return;
   (b) heavy usage of the time-sharing system so that every user's response time has deteriorated; and
   (c) temporary breakdown so that nobody gets any response at all.

A useful procedure, under such circumstances, would be first to push the carriage return twice, then the quit button once. Pushing the carriage return will remedy the first type of problem and should elicit a response fairly quickly. If heavy usage is the problem, merely pushing the carriage return will have no effect, nor will it damage DATANAL. However, a single push of the quit button will induce a manual abort, indicated by the usual "OK" message. If no response occurs to either of the above procedures, a temporary breakdown has probably occured, and there is nothing to do but wait.

7.1.7 Data Base Protection

Recall from section 5.2 that all relevant information pertaining to a working data base is stored on the P-string and its associated data blocks. It is essential, therefore, that these disk files be protected against damage or accidental loss. In addition to the various protective devices provided by the time-sharing system for all disk files, **DATANAL** affords special protection to these critical files. In particular, it prevents their loss due to disk storage overflow. Whenever a user has exceeded his storage quota (e.g., by creating new data within **DATANAL**), the P-string and/or its associated data blocks will be written out onto the disk in temporary mode instead of the usual permanent mode. **A** strong warning will appear on the console indicating that such has occurred and recommending that the user clear out some disk space immediately. It is imperative that this warning be heeded and that the temporary files be returned to permanent mode.

7.2 Data Base Commands

There are three data base functions which must be performed within **DATANAL**. First, when a raw data base is initially submitted to a user's disk file, or whenever an existing raw data base is structurally altered, **DATANAL** must be notified. Upon notification, all necessary machinery for creating working data bases therefrom will be set up. These functions are performed by the SETUP and ESTABLISH commands.

Second, there must be some way to activate the above machinery and thereby create working data bases. Also, there must be some way to move back and forth between working data bases which have already been created. These functions are performed by the OPEN and CLOSE commands.

Finally, even though DATANAL may know which working data base it is operating upon, which raw data base this came from, and which structural properties pertain to both, the user may be unclear on some of these issues. Hence, the LIST command may be issued to obtain such information.

7.2.1 The SETUP Command

The SETUP command is designed to notify DATANAL of the existence of a raw data base prepared externally in some machine-readable form. Typically, this means a deck of cards. Assuming cards, the prepared deck must be submitted off-line to the disk editor for transfer onto the user's disk file. All raw data bases must be given the second name "DATA." Any unique combination of one-to-six nonblank characters may serve as a first name. Also, raw data bases must adhere to the specifications set forth in section 5.3.

Assuming the rapid input mode, the SETUP command is issued as follows.

```
SETUP DBNAME NLV NCS1 NREC1 ... NCSN NRECN
```

DBNAME is the first name of the raw data base to be set up. The second name is assumed to be DATA. NLV is the number of levels in DBNAME. NCS1

is the number of cases on the first (lowest) level, and NREC1 is the corresponding number of unit records (e.g., cards) per case. If DBNAME contains more than one level, subsequent pairs of NCS and NREC parameters are entered up to NCSN NRECN, indicating the number of cases and records per case, respectively, associated with additional levels up to the N^{th} (highest) level.

The command word "SETUP" and all subsequent parameters must be typed on a single line with each word separated by at least one space. Multiple spaces will be ignored without damage and without an error message. However, the order of presentation of parameters is critical and must be exactly as shown above.

The first action taken is to inspect whatever specification has been entered by the user. This is done on a word-by-word basis. Checks are made to determine whether:

1. a disk file exists under the name DBNAME DATA;
2. NLV is a positive integer not exceeding 10;
3. each value of NCS is a positive integer not exceeding 4096;
4. each value of NREC is a positive integer;
5. The number of NCS NREC pairs following NLV is numerically equal to NLV;
6. the total number of level-linkage identity tags (see formula in section 5.3) falls at or below 4096; and
7. successive values of NCS comprise a sequence of (weak) monotone decreasing integers.

Absent, improperly ordered, and/or illegal parameters will cause specific error diagnostics to be typed out on the console, along with instructions for correction and re-submission. Under certain conditions (e.g., incomplete and/or multiple illegal specifications), the user will be thrown into the guided input mode.

Assuming a legal specification in the above formal senses, the next step is to check factual correspondence with the actual contents of DBNAME. Additional checks are made to determine whether:

1. the total number of records in DBNAME falls at or below the total number stipulated in the specification (note: specifying only a portion of DBNAME is perfectly legal, but this precludes incorporating the unspecified portion in any working data bases to be created at a later point in time);
2. explicit level-linkage identity tags appear as positive integers not exceeding 9999 in the appropriate fields of all first records associated with all cases on all levels;
3. within each level, level-linkage tags are unique (i.e., each tag is a different, though not necessarily serial integer); and
4. between each pair of levels, every lower-level case possesses the tag of the higher-level case to which it belongs (note: it is perfectly legal for a higher-level

case to possess no lower-level cases as members, but it is illegal for a lower-level case to allege membership in a nonexistent higher-level case).

If all formal and factual checks progress without error, no diagnostic messages will appear on the console. Instead, a message stating that "DBNAME DATA HAS BEEN SET UP" will be printed out, followed by the usual "OK." DATANAL is now ready for the next command.

Successful execution of the SETUP command produces two files on the user's disk. The first one is called "DBNAME PSETUP" and contains all structural information about DBNAME, as well as property definitions of the various level-linkage identity tags contained therein. The second file is called "DBNAME DSETUP" and contains the actual tags read from DBNAME. These two files will form the skeleton of all subsequent working data bases associated with DBNAME.

7.2.2. The ESTABLISH Command

The ESTABLISH command is a special-purpose version of the SETUP command. It is designed to accommodate small data bases which the user is willing to enter by hand from the console. Pre-test data gathered prior to conducting a full-scale questionnaire survey or experiment might best be handled by the ESTABLISH command, since no cards need be punched and submitted off-line to the disk editor.

Assuming the rapid input mode, the ESTABLISH command is issued as follows.

ESTABLISH DBNAME NCS

DBNAME is the first name to be given to the raw data base. NCS is the number of cases contained therein.

In reality, no raw data base called "DBNAME DATA" will exist on the user's disk file. However, DATANAL will be tricked into believing otherwise. This will allow the user to enter raw data manually by means of the SET and CHANGE commands. The exact procedure for completing this fraud will be postponed until the SET and CHANGE commands are discussed (see sections 7.3.3 and 7.3.10).

Successful execution of the ESTABLISH command will produce two disk files--"DBNAME PSETUP" and "DBNAME DSETUP." Their respective contents will be as described under the SETUP command. However, the ESTABLISH command makes the following uniform assumptions which the SETUP command does not.

1. There will be exactly one level of cases.
2. Each case will be assigned a serial integer from one to NCS, and these integers will serve as identity tags.
3. Each case will be assumed to require one unit record (this is only to protect the user against disaster in case he accidentally issues the READ command at some future time).

A single error check is made to insure that NCS is a positive integer not exceeding 4096.

7.2.3 The OPEN Command

The OPEN command serves two distinct functions. It may be used to create a new working data base from a raw data base which has already been set up or established. Alternatively, it may be used to re-open an existing working data base for further analysis.

Assuming the rapid input mode, the OPEN command is issued as follows.

```
OPEN DBNAME
```

If a new working data base is to be created from a raw data base, DBNAME is the name of that raw data base. If an existing working data base is to be re-opened, DBNAME is the name of that working data base. The effect in either case will be to produce a disk file called "CURENT PSTRNG," another file called "CURENT 1," and possibly some additional files, if multiple data blocks exist.

Action taken by the OPEN command is as follows. First, the user's disk is searched for an existing file under the name "CURENT PSTRNG." If one is found, the user is so informed, and control is returned to the special supervisor, indicated by the usual "OK." This prevents the user from inadvertently wiping out whichever working data base he is currently operating upon.

If CURENT PSTRNG is not found, the user's disk is then searched for a file called "DBNAME PSTRNG." This file will be found if an existing working data base under that name is to be re-opened. If found, it will be re-named "CURENT PSTRNG," and all associated data blocks will be given the first name "CURENT." Should one or more of the associated data blocks be missing, appropriate error messages will appear on the console. Finally, the message "DBNAME HAS BEEN OPENED" will appear, followed by "OK." This signifies successful execution of the command.

If DBNAME PSTRNG is not found, the user's disk is searched for a file called "DBNAME PSETUP." If found, it will be duplicated (not re-named) under the name "CURENT PSTRNG," and a duplicate of DBNAME DSETUP will be created under the name "CURENT 1." A successful execution message and "OK" will follow.

If neither of the above files is located, an error message will appear, and control will return to the special supervisor.

7.2.4 The CLOSE Command

The CLOSE command also serves two functions, but they are concurrent. First, it provides a mechanism for the user to assign a unique name to a working data base. Second, it clears the way for a new and/or different working data base to be opened for analysis.

Assuming the rapid input mode, the CLOSE command is issued as follows.

```
CLOSE DBNAME
```

DBNAME is the name to be given to the current working data base. Any unique combination of one-to-six nonblank characters is legal.

Action taken by the CLOSE command is as follows. First, the user's disk is searched for an existing file called "DBNAME PSTRNG." If one is found, the user is so informed, and control is returned to the special supervisor with an "OK." This prevents the user from inadvertently closing two working data bases under the same name.

If DBNAME PSTRNG is not found, the user's disk is then searched for a file called "CURENT PSTRNG." This file will only be found if the user was operating on a working data base prior to issuing the CLOSE command. If found, it will be re-named "DBNAME PSTRNG," and all of the current data blocks will be given the first name DBNAME. Should one or more of the current data blocks be missing, appropriate error messages will appear on the console. Finally, the message "DBNAME HAS BEEN CLOSED" will appear, followed by "OK." This signifies successful execution.

If CURENT PSTRNG does not exist, an error message will appear, and control will return to the special supervisor.

7.2.5 The LIST Command

The LIST command is designed exclusively to provide status information to the user. It displays three kinds of information on the console:

1. structural information pertaining to the current working data base, if one has been opened;
2. a list of the names of all properties defined within the current working data base; and
3. a list of all currently defined command names.

Assuming the rapid input mode, the list command is issued in either one of the following three forms.

$$\text{LIST}\begin{Bmatrix}\text{STATUS}\\ \text{PROPERTIES}\\ \text{COMMANDS}\end{Bmatrix}$$

LIST STATUS elicits structural information pertaining to the current working data base and the raw data base from which it was created (e.g., number of levels, number of cases, etc.). LIST PROPERTIES and LIST COMMANDS elicit property and command name lists, respectively.

The user need not inspect all of the information provided by the LIST command. A manual abort (see section 7.1.6) may always be executed to terminate the listing process.

Incidentally, the list of command names will only contain the last six characters of each name. Thus, the ESTABLISH command will be listed as "ABLISH." Similarly, if the user should assign property names of more than six characters, LIST PROPERTIES will uniformly display only the last six. This is because all names in DATANAL occupy a

single computer word, and computer words cannot accommodate more than six charactars. In general, however, typing names of more than six characters onto the console will not cause trouble, unless the last six characters are identical to some other, already-established name.

7.3 Property-Defining and Property-Describing Commands

There are eleven commands in DATANAL which perform one or more of the following four functions:

1. definition of new properties;
2. description of existing properties;
3. change in definition of existing properties; and
4. erasure of existing properties.

The definition of new properties may take any one of three forms. First, a property may be defined as basic. This means not in terms of and without reference to any previously defined properties. Raw data to be read from a raw data base, to be typed in from the console, or to be generated internally according to some mathematical rule (e.g., serial numbers) illustrate this concept. In fact, these three examples exhaust the various ways in which basic properties may be defined in DATANAL. They are implemented by the READ, SET, and ASSIGN commands, respectively.

Second, a property may be defined in terms of some algebraic and/or logical transformation of previously defined properties. As such,

this constitutes a nonbasic definition. The DEFINE, COMPUTE, RECODE, ASSIGN, and ECOUNT commands implement nonbasic definitions of this form.

Third, a property may be defined as a special re-arrangement of existing property data without any algebraic transformation and possibly without any logical transformation. Forming a vector property out of scalar properties and forming a matrix property out of vector properties illustrate one kind of special re-arrangement. This function is performed by the DEFINE and SET commands. Grouping cases on one level to compute conditional summary statistics applicable to the next-higher level illustrates another kind of special re-arrangement. This function is performed by the GROUP command. Both of these constitute nonbasic definitions.

A single command exists in DATANAL to describe existing properties. This is the DESCRIBE command. It contains numerous selectivity options, as well as a "complete" option to trace and display complete definitional "trees," starting with all relevant basic definitions and culminating in a single nonbasic definition. In this manner, complete documentation of any given analytical sequence may be generated automatically upon request.

A single command with many options exists to alter various portions of an existing property definition. This is the CHANGE command. Through it, the name, definition mode, creation mode, level, and/or verbal description of any existing property may be changed.

7.3.1 The READ Command

The READ command is the vehicle by which certain physical fields in a raw data base are singled out as containing data corresponding to properties defined in a working data base. In fact, this field address constitutes the operational definition of any property defined by the READ command. All such definitions are basic.

Assuming the rapid input mode, the READ command is issued as follows.

```
READ PLABEL LEVEL RECNO COLNO FWIDTH
```

PLABEL is the name or label to be assigned to whichever property is being read. LEVEL is the level of that property. RECNO is the serial number of the record (e.g., card) on which the corresponding property data is to be found (e.g., second card each case). COLNO is the starting column of the data field on that record (e.g., column 37). FWIDTH is the field width or number of columns occupied by that field (e.g., three columns).

Action taken by the READ command is as follows. First, the specification is inspected to determine whether:

1. it is complete;
2. it is properly ordered (i.e., just as shown above);

3. PLABEL is a legal property name (i.e.: not a pure number; not one of the arithmetic operators "+", "-", "*", "/", or "**"; not one of the punctuation marks "(",")", or ","; not one of the library function names "ABS", "SQRT", "LOG", "EXP", "SIN", "COS", "MAX", or "MIN"; not the same as any already-defined property label; and not either of the logical control words "CONT." and "OR");
4. LEVEL is a positive integer not exceeding the maximum number of levels assigned to this working data base by the SETUP command;
5. RECNO is a positive integer not exceeding the total number of records required by each case on this level;
6. COLNO is a positive integer not exceeding 80;
7. FWIDTH is a positive integer not exceeding 12; and
8. the sum of COLNO & FWIDTH is a positive integer not exceeding 80 (i.e., the total field does not extend beyond column 80).

Incomplete, illegal, and/or improperly ordered specifications are handled in the standard manner.

As soon as a legal specification is achieved, the user is given the option of appending a verbal description. This may be accomplished by responding "yes" to the option request which appears on the console. Any other response, including a carriage return, is

interpreted as meaning that no verbal description is desired. If "yes" is typed, the computer responds with directions for entering a verbal description of the property PLABEL. Verbal descriptions may be of any length, of any format, and in any language. They are stored as unedited textual information on the P-string.

Following the verbal description, a message stating that "PLABEL HAS BEEN READ" will appear on the console, and the user will be given the choice of either terminating the READ command or entering another specification. Subsequent specifications may be given without limit, except that:

1. three or more errors within any single specification (but not cumulated across successive specifications) will induce an automatic abort;
2. the user may execute a manual abort at any time;
3. a large number of long specifications may fill the computer's internal memory allocation, in which case an automatic abort is triggered between specifications, and additional internal storage is allocated.

In none of the above instances will already completed specifications be nullified.

Successful execution of each READ specification will serve to update the P-string. The new property name, the associated status information, the operational definition, and the verbal description (if

any) will be appended. Whenever a property is defined by the READ command, the following specifications are made automatically (i.e., without explicit directions from the user).

1. The origin of the property definition is noted as the READ command.
2. The property type is uniformly scalar.
3. Each property value, being a scalar quantity, has one row and one column as its dimensionality.
4. Both definition and creation modes are set to temporary, indicating, respectively, that the definition may be erased and that the corresponding property data, once read in, may be destroyed.
5. The operational definition is the field address of the corresponding property data on the raw data base.

As experience is gained through practice, users will wish to move more rapidly through the READ command. This statement applies equally to all commands in DATANAL. The fastest way through the READ command is to type a complete (and correct) specification on a single line, followed by three carriage returns. The first return serves to transmit the specification to the computer. The second serves as a negative answer to the optional verbal description request. The third serves as a negative answer to the "go-again" request. Under such circumstances, interaction is both minimized and expedited, since all option queries are answered (negatively) in advance.

This use of multiple carriage returns to avoid interactive options and to expedite specifications is a standard feature of many DATANAL commands. Another use of multiple carriage returns, not illustrated by the READ command, is to signal the end of variable-length specification lists. Consequently, DATANAL may appear to "get stuck" from time to time, when the only problem is an insufficiency of carriage returns. This situation is rendered more likely by the necessity of a carriage return after every input line, coupled with the tendency of users to forget it. When in doubt, an always safe and frequently effective remedy is to push two or more carriage returns. Unless the computer is down, or time-sharing demands are excessive, or DATANAL has requested (but not received) specific information, this maneuvre will generally elicit a response. At worst, desired options (e.g., a verbal description) may be inadvertently skipped. However, DATANAL is amply equipped to recover from this kind of oversight.

7.3.2 The DEFINE Command

If any single command were to be identified as the "heart" of DATANAL, it would be the DEFINE command. Although not the most complex from the programming point of view (the CREATE command has no competitors in this respect), it is both the most complex and the most central from the user's point of view. Also, it will be the most frequently issued of the property-defining commands, if not the most frequently issued of all commands. Within it are incorporated most of the basic concepts of the entire DATANAL language.

The DEFINE command performs three broad types of functions. It specifies algebraic transformations to be performed on property data (e.g., index formation, change of variable or functional form, change of scale, etc.). It specifies logical transformations to be performed both on property data (e.g., partitioning into sub-intervals) and on cases (e.g., partitioning into sub-samples). It also specifies the re-arrangements necessary to form both vector and matrix properties out of scalar properties. Logical transformations may be specified in addition to either algebraic transformations or vector/matrix definitions within a single command, but algebraic and vector/matrix manipulations must be specified by separate commands.

Assuming the rapid input mode, the DEFINE command is issued in either of the following two general forms.

```
DEFINE PLABEL = {some algebraic expression}
FOR ALL CASES WHERE
1.  PLABEL1 R1 {some algebraic expression}
2.  PLABEL2 R2 {some algebraic expression}
.
.
.
M.  PLABELM RM {some algebraic expression}


DEFINE PLABEL = PLABEL1 PLABEL2 ... PLABELN
FOR ALL CASES WHERE
1.  PLABEL1 R1 {some algebraic expression}
2.  PLABEL2 R2 {some algebraic expression}
.
.
.
M.  PLABELM RM {some algebraic expression}
```

Within both of the above two general forms there are numerous variations and options. These will be discussed below.

Let us begin with the first form of the DEFINE command. This is designed to specify algebraic transformations, in conjunction with one or more optional logical transformations or Boolean conditions, as they will henceforward be called. PLABEL is the name of the property to be defined. The equals sign separating PLABEL from the algebraic expression means "assign the computed value of." That is, if and when PLABEL is created, the computed value of the algebraic expression to the right of the equals sign will be assigned as the property value of PLABEL for each applicable case in the working data base. The algebraic expression to the right of the equals sign must adhere to certain rules of content and syntax, which will be discussed shortly. This completes the first line of the specification.

The second and all succeeding lines are optional. If no Boolean conditions are to be applied (i.e., if the indicated computation is to be performed and the result is to be assigned to every case on the level at which PLABEL resides), they may be omitted entirely. This is accomplished by entering a double carriage return following the first line. Alternatively, the same result may be achieved by entering a single carriage return and a second line stating "FOR ALL CASES," followed by another single carriage return.

If one or more Boolean conditions are to be applied, at least two additional lines following the first line are required (at least three lines in all). The second line must be "FOR ALL CASES WHERE." The last Boolean condition line must be flagged by entering a double carriage return thereafter.

In formulating Boolean conditions, several factors must be kept in mind. First, each elementary Boolean condition must be complete and of proper syntax. Second, it is frequently desirable to state complex Boolean conditions (i.e., elementary conditions which are interconnected with ANDs, ORs, and NOTs). Such interconnections must also adhere to certain rules of syntax. Finally, the entire specification must be arranged in blocks of typed lines so that the computer may decipher the user's intentions.

An elementary Boolean condition possesses three components in serial order (from left to right);

1. an already-defined property label;
2. a legal numeric relationship; and
3. a legal arithmetic expression.

The meaning of an already-defined property label has been discussed previously. Legal numeric relationships include:

1. E, meaning is numerically equal to;
2. NE, meaning is not numerically equal to;
3. GT, meaning is strictly greater than;

4. LT, meaning is strictly less than;

5. EGT, meaning is equal to or greater than; and

6. ELT, meaning is equal to or less than.

Legal arithmetic expressions will be discussed shortly.

The existence of a logical NOT or negative is maintained implicitly in the six numeric relationships displayed above. Thus, not equal may be represented simply as "NE," not greater than by "ELT," and so forth. This constitutes the first step toward formulating more complex conditions out of elementary conditions.

The provision of an explicit logical OR constitutes the second step. The general form of such a specification is as follows.

PLABEL1 R1 {some algebraic expression} OR

PLABEL2 R2 {some algebraic expression} OR ...

OR PLABELN RN {some algebraic expression}

In addition, there are two abbreviated forms which may be used under certain circumstances. If two or more OR-ed elementary conditions share the same PLABEL, but differ in their numeric relationships and/or in their corresponding algebraic expressions, the following abbreviation is acceptable.

PLABEL R1 {some algebraic expression} OR R2 {some algebraic expression}

If two or more such conditions share both the same PLABEL and numeric relationship, and if that relationship is "E," then further abbreviation is possible as shown below.

PLABEL E {some algebraic expression} OR {some algebraic expression}

This latter abbreviation simplifies the task of defining by enumeration either a sub-sample of cases or a sub-interval along a property scale.

The third step in formulating complex conditions is the provision of a logical AND. Whereas OR-ed conditions are typed horizontally along a single condition line, AND-ed conditions are numbered serially (for purposes of diagnostic reference) and typed as a vertical column of successive condition lines. Each successive AND-ed condition line may itself be a constellation of OR-ed conditions.

Finally, it is possible to extend any condition line beyond the right-hand margin of the console by entering the special control word "CONT." as the last word on that physical input line. (Note: the placement of "CONT." in any position except the final position will result in an error message.) This same technique of extending the logical length of specification lines may be used on the top line of the DEFINE command. It may also be used in any other command of DATANAL which involves variable-length specification lines. After entering "CONT.", push the carriage return once, and continue typing as if the physical line had been extended. Repeated extensions may be obtained by ending successive physical lines with "CONT.", so long as the extended logical line contains no more than 100 words. The termination of an extended logical line is signaled by the first appearance of a physical line ending in anything except "CONT.". If this occurs on the first line, then the physical and logical lines are co-extensive.

Let us now consider algebraic expressions. The contents of a legal algebraic expression include and are restricted to the following items:

1. pure numbers;
2. already-defined property labels;
3. algebraic operators;
4. punctuation marks; and
5. library functions.

A pure number is defined as any solid string of one-to-six digits, possibly preceded by either a "+" or a "-", and possibly containing a single "." (decimal point) anywhere in the string except preceding either "+" or "-". Anything else will not be interpreted as a number. In particular, the presence of one or more blanks or spaces within the string will preclude its numeric interpretation, although the string may be preceded and/or followed by any number thereof.

The user should bear in mind that numbers, like all other inputs and outputs of DATANAL, are transmitted to and from the console in free format. This relieves the user from having to compose and enter any format specifications. However, it also imposes some restrictions on the interpretation process. These are listed below.

1. All numbers are words and, therefore, must be preceded by at least one blank or space in any input message typed on the console.

2. Like all other words, the representation of a number must be confined to six characters or less, including "+", "-", and/or ".". This serves to limit both input and output numbers to fall between 999999 and -99999, inclusive. If either larger or smaller numbers are desired, an algebraic expression may be composed which specifies a scale change (e.g., multiplication by some power of ten). This will relax the effective range constraints, but the accuracy of computed results will remain limited to six significant digits at most.
3. If a user should enter a number involving more than six characters, it will be truncated from the left, and only the last six characters will be interpreted. For this reason, it is always a good idea to "play back" any algebraic specification by issuing the DESCRIBE command.
4. None of the above restrictions apply to numbers read from a raw data base. Up to twelve characters are permitted here, and up to eight significant digits of accuracy are possible.
5. All output numbers returned by DATANAL to the console are formatted automatically so that the one-to-six most significant (i.e., left-most) digits are displayed, in addition to a minus sign and/or a decimal point, if

appropriate. Leading plus signs and trailing zeros are uniformly truncated. Integers are represented without a decimal point. Decimals are rounded in the last significant digit.

Returning to the contents of a legal algebraic expression, the next item is an already-defined property label. Whether or not a candidate word satisfies this criterion is determined by consulting the current P-string.

There are five algebraic operators which may be included in any algebraic expression. These are:

1. +, meaning addition;
2. -, meaning subtraction;
3. *, meaning multiplication;
4. /, meaning division; and
5. **, meaning raised to a power.

As with pure numbers and property labels, each of these is a single word and must, therefore, be preceded by at least one blank or space to insure proper interpretation of typed inputs. (Note: It requires a little practice to remember that each word in an algebraic expression must be so separated from its neighbors.)

Since properties in DATANAL may be either scalars, vectors, or matrices, the above algebraic operators are interpreted in terms of matrix algebra conventions. This includes, but is not restricted to normal scalar conventions. In particular, the following conventions apply.

1. Any two properties/numbers may be added, so long as they are dimensionally equivalent (i.e., possess the same number of rows and columns, respectively). If equivalent, addition will take place element-by-element.
2. Subtraction and division are handled identically. (Note: It is possible to divide equivalent vectors and matrices in DATANAL element-by-element so as to obtain arrays of ratios and percentages.)
3. Any two properties/numbers may be multiplied, so long as either at least one of them is a scalar or they are dimensionally conformable (i.e., the number of columns of the first equals the number of rows of the second). If conformable, a matrix product will always result with an appropriate dimensionality. For purposes of dimensionality determination, vectors in DATANAL may be either row vectors or column vectors, whichever makes a multiplication conformable. If neither interpretation results in a conformable specification, an error message will ensue. Unless multiplication is indicated in an algebraic expression, vectors are normally considered by DATANAL to be row vectors. This convention has been adopted to conform with the row orientation of the console as a display device. (Note: Element-by-element multiplication may be substituted

for matrix multiplication of two dimensionally equivalent properties by specifying the reciprocal of {the reciprocal of one divided by the other}.)

4. Raising to a power applies only to scalars. That is, only scalars may be raised to a power, and only scalars may serve as powers. However, powers need not be integers.

There are three punctuation marks which may be included in any algebraic expression. These are:

1. (, meaning left parenthesis;
2.), meaning right parenthesis; and
3. ,, meaning comma.

Left and right parentheses are used to alter the order in which computations are performed when an algebraic expression is evaluated. The comma occurs only in conjunction with the MAX and MIN functions to be discussed immediately.

Finally, there are eight library functions built into DATANAL. These are:

1. ABS, meaning absolute value of;
2. SQRT, meaning positive square root of;
3. LOG, meaning natural logarithm of;
4. EXP, meaning exponentiation of (base e);
5. SIN, meaning sine of (argument in radians);
6. COS, meaning cosine of (argument in radians);

7. MAX, meaning maximum value of; and

8. MIN, meaning minimum value of.

All of the above eight functions must be followed immediately by a left parenthesis, then a legal algebraic expression, and then a right parenthesis. In the case of MAX and MIN, a sequence of at least two legal algebraic expressions must appear between the parentheses, with successive expressions set off by commas. (Note: Commas and parentheses are treated as single words and, therefore, must be separated from their neighbors by at least one blank or space.)

This completes our discussion of the first general form of the DEFINE command. The second form is used to define vectors and matrices, possibly subject to Boolean conditions. It is identical to the first form except on the top line. Here, a list of two or more already-defined property labels replaces the algebraic expression to the right of the equals sign. No parentheses nor commas nor other words may appear in the list. The extension option is possible by using "CONT.", so that an array of up to 97 elements may be specified. If the elements are scalars, the result will be a (row) vector property. If the elements are vectors, the result will be a matrix property whose rows are the component (row) vectors in the specification list. If the elements are matrices, an error message will ensue. If the elements are mixed and/or dimensionally nonequivalent, an error message will also ensue.

The first action taken by the DEFINE command is to inspect the top line. Checks are made to determine whether:

1. PLABEL (the second word on the line) is a legal property name (see criteria in section 7.3.1);
2. the third word is "="; and
3. all words to the right of the equals sign (and there must be at least one) are legal components of an algebraic expression.

Assuming that all of these are passed, the next action is to determine whether an algebraic transformation or a vector/matrix specification is intended. A vector/matrix specification is assumed if at least two words appear to the right of the equals sign, and if all of these are already-defined property labels. Otherwise, an algebraic transformation and, therefore, an algebraic expression is assumed.

If an algebraic expression is assumed, an extended series of checks is made on the syntax and dimensionality of the specification. Assuming no errors, the following information is stored on the P-string:

1. the level of PLABEL, which is determined as the level of the lowest-level property in the algebraic expression, excluding all zero-level properties (note: there must be at least one nonzero-level property for reasons that will become clear when we discuss the SET and COMPUTE commands);
2. the dimensionality of PLABEL, which is determined by simulating the process of evaluating the expression.

If a vector/matrix specification is assumed, additional checks are made to determine whether:

1. all properties are of the same type (e.g., all scalars) and share the same dimensions;
2. no properties are already defined as matrices.

If both checks yield affirmative results, the level and dimensions of PLABEL are computed and stored on the P-string.

Next comes a test for the existence of Boolean conditions. Detection of a double carriage return following the top line or a second line stating "FOR ALL CASES," followed by a single carriage return, signals the absence of any conditions. Otherwise, one or more are assumed to follow.

Boolean condition checks are made to determine whether:

1. the second specification line reads "FOR ALL CASES WHERE";
2. all subsequent condition lines are numbered serially;
3. each elementary condition is composed of an already-defined property label, followed by a legal numeric relationship (either literally or by implication), followed by a legal arithmetic expression;
4. all nonzero level properties referenced on both sides of every numeric relationship exist on at least as high a level as PLABEL;
5. each property on the left of a numeric relationship is dimensionally equivalent to the algebraic expression on the right; and

6. OR-ed conditions are stated in one of the three alternative forms previously discussed.

Assuming no errors, the DEFINE command will search for the first double carriage return following a condition line. This means that the last condition line has been reached. A request for an optional verbal description will follow immediately. Successful execution of the entire command will then be indicated by a message stating "PLABEL HAS BEEN DEFINED," followed by the usual "OK."

As with the READ command, the effect of the DEFINE command is to update the P-string. The new property name, the associated status information, the operational definition, and the verbal description (if any) is appended thereto. In addition to the specifications already discussed, additional items of information are added automatically.

1. The origin of the property definition is noted as the DEFINE command.

2. Both definition and creation modes are set to temporary.

For the experienced user, the fastest way through the DEFINE command is as follows.

1. If no Boolean conditions are intended and no verbal description is desired, enter the algebraic transformation or vector/matrix definition, followed by three carriage returns.

2. If Boolean conditions are intended, but no verbal description, proceed in the normal manner through the last condition line. Then enter three carriage returns.

7.3.3 The SET Command

The SET command is a rapid and specialized alternative to the DEFINE command. Its use is restricted to entering constants into level zero of a working data base. It cannot involve either algebraic expressions or Boolean conditions, but it may be used to define either scalar, vector, or matrix properties. In addition, it performs several special functions.

Assuming the rapid input mode, the SET command is issued as follows.

```
SET PLABEL = PLABEL1 PLABEL2 ... PLABELN
```

The meaning of the above terms is exactly as discussed under the vector/matrix alternative of the DEFINE command.

Action taken by the SET command parallels exactly the action taken by the DEFINE command, with the following exceptions.

1. All activities associated with Boolean conditions are omitted.
2. The list of words to the right of the equals sign may contain only one item.
3. Pure numbers are permitted in any mixture with already-defined property labels, provided that all referenced properties reside on level zero.

4. If only pure numbers are involved, then PLABEL is defined as a basic property. Otherwise, it is nonbasic.
5. The origin of PLABEL is noted as the SET command.

One special use of the SET command was alluded to in section 7.2.2. First, a phoney raw data base is set up by means of the ESTABLISH command. Second, raw data are entered from the console via the SET command. If N cases were specified in the ESTABLISH command, then strings of pure numbers, each of length N, must be typed via SET. Third, all properties are created by issuing the CREATE command. This may be accomplished in one shot, as will be explained in section 7.4.1. Finally, the level of each property is changed from 0 to 1 by issuing the CHANGE command (see section 7.3.10, LEVEL option).

Restrictions on the above procedure are as follows.

1. Only pure numbers are allowed in each issuance of the SET command.
2. Property data must be entered by property rather than by case (i.e., all of the data associated with each successive case on a given property must be entered serially).
3. There must be exactly N numbers entered under each property, and N cannot exceed 97. Also, N must agree with the number of cases specified by the ESTABLISH command.
4. The user is responsible for maintaining proper case-by-case ordering within each property.

If properly executed, this procedure is equivalent to a SET-OPEN-READ-CREATE sequence applied to an externally prepared data deck.

A second special use of the SET command involves vector and matrix manipulations. The CREATE command will automatically impute zero-level property data to any case on any level. Hence, special vectors and matrices may be entered via SET whose sole purpose is to extract specified rows, columns, or elements from vector and matrix properties. Alternatively, row sums, column sums, and matrix sums may be computed by means of other special vectors. From these, means, standard deviations, and many other useful statistical indices may be computed with great ease.

Finally, SET may be used to establish symbolic equivalents of recurrent scalars. Thus, the value of π or e may be set once and thereafter referenced symbolically rather than continually written out as a pure number.

7.3.4 The COMPUTE Command

The COMPUTE command is also a rapid and specialized alternative to the DEFINE command. Like the SET command, its use is restricted to entering constants into level zero, and it cannot involve Boolean conditions. However, unlike the SET command, it cannot accommodate

vector/matrix specifications. It operates exclusively on algebraic specifications.

Assuming the rapid input mode, the COMPUTE command is issued as follows.

COMPUTE PLABEL = {some algebraic expression}

The meaning of the above terms is exactly as discussed under the algebraic transformation alternative of the DEFINE command.

Action taken by the COMPUTE command parallels exactly the action taken by the DEFINE command, with the following exceptions.

1. All activities associated with Boolean conditions are omitted.
2. All properties referenced in the algebraic expression must reside on level zero.
3. If one or more zero-level properties are referenced, then the origin of PLABEL is noted as the COMPUTE command.
4. If no properties are referenced (i.e., if the algebraic expression is completely devoid of property labels), then the CREATE command acts as a desk-calculator, and an immediate answer appears on the console. This precludes updating the P-string.

The desk-calculator function of COMPUTE is particularly useful for evaluating fitted curves to empirical data. Results obtained from

either a linear or polynomial curve fit (see section 8.1.8) may be so tested.

7.3.5 The RECODE Command

The RECODE command is a specialized version of the DEFINE command. It performs both arithmetic and logical transformations of a limited nature. Arithmetic transformations are constrained to produce only positive integers as outputs, and logical transformations may only be specified in terms of a single property. However, despite these limitations, the RECODE command serves as the basis for all frequency counts, cross-tabulations, and related analytical procedures (e.g.: Chi square, Fisher, and median tests; one-way analysis of variance; two-way analysis of variance; etc.).

Assuming the rapid input mode, the RECODE command is issued as follows.

```
RECODE PLABEL PLABEL1 (SPEC1) (SPEC2) ... (SPECN)
```

PLABEL is the name of the property being re-coded. PLABEL1 is the name of the new property currently being defined as a re-coded version of PLABEL. (SPEC1), (SPEC2), ..., (SPECN) are Boolean specifications indicating which numeric values of PLABEL will be assigned, respectively, to category 1, category 2, ..., category N of PLABEL1. The form and interpretation of a Boolean specification will be explained shortly.

Action taken by the RECODE command is as follows. First, the entire command specification is inspected to determine whether:

1. it is complete;
2. it is properly ordered (i.e., just as shown above);
3. PLABEL is an already-defined property name identifying a scalar property not on level zero;
4. PLABEL1 is a legal name for a new property (see requirements listed in section 7.3.1); and
5. the entire specification does not exceed 200 words.

Assuming passage of the above checks, the various Boolean specifications are next inspected to determine whether:

1. at least one exists;
2. each successive specification is enclosed in parentheses; and
3. each successive specification is legal in both content and syntax.

Incomplete, illegal, and/or improperly ordered specifications are handled in the standard manner.

Legal contents of a Boolean specification include and are restricted to:

1. pure numbers;
2. the six numeric relationships listed in section 7.3.2; and
3. the word "TO."

These may be arranged to form complex Boolean conditions, just as in the DEFINE command. However, the notation is more compact.

A Boolean specification may assume any one of the following three forms, or any combination thereof, subject to an ordering restriction:

1. (N1 N2 ... NM);
2. (R N);
3. (N1 TO N2).

The first form above is a list of pure numbers separated by at least one blank or space and enclosed within parentheses. There is an implied OR linking each of the M listed numbers N1, N2, ..., NM. The interpretation, here, is that whenever PLABEL assumes any one of the listed numeric values (i.e., either N1 or N2 or ... or NM), then PLABEL1 is to be assigned the serial integer corresponding to the category indicated by this specification. Thus, if the value of a case on PLABEL is numerically equal to N1, N2, ..., or NM, and if this is the third Boolean specification, which defines the contents of the third category of PLABEL1, then the integer three will be assigned to PLABEL1 as the numeric value of that case. Otherwise, the numeric value of PLABEL1 will be something other than three or, possibly, undefined.

The second form above is a single numeric relationship followed by a pure number. These two terms are separated by at least one blank space and enclosed within parentheses. The interpretation, here, is

that whenever the elementary Boolean condition "PLABEL R N" is satisfied, then PLABEL1 is to be assigned the serial integer corresponding to the category indicated by this specification. Otherwise, some other integer is to be assigned, if the value of the case is defined. Any number of these elementary conditions may be OR-ed in a single Boolean specification. However, OR-ing must be implicit (i.e., the word "OR" must not be typed on the console). To illustrate, the following complex specification is legal and, incidentally, is logically equivalent to the illustration under the first form of specification previously discussed.

(E N1 E N2 ... E NM)

The third form above consists of the word "TO" sandwiched in between two pure numbers. As usual, each word must be separated from its neighbors by at least one blank or space, and the entire specification must be enclosed within parentheses. The purpose of this form of specification is to designate a closed interval along the scale of PLABEL. Its interpretation is that whenever PLABEL assumes a value between N1 and N2 (inclusive), then PLABEL1 is to be assigned the serial integer corresponding to the category indicated by this specification. Any number of these closed intervals may be OR-ed in a single Boolean specification. However, OR-ing must again be implicit.

Finally, OR-ing may occur between the above three forms as well as within them. Thus, any given specification may contain a mixture of the three forms. The only restriction is that the pure numbers

within every Boolean specification must form a (weak) monotone increasing sequence. This restriction does not apply across successive specifications. The "CONT." option is available to extend the effective length of the specification line.

As soon as a legal specification is achieved, a request for an optional verbal description will appear. This will be followed by the message "PLABEL HAS BEEN RECODED UNDER THE NAME OF PLABEL1" and the usual "OK," indicating successful execution.

Successful execution serves to update the P-string. The new property name, the associated status information, the operational definition, and the verbal description (if any) will be appended. In addition, the following specifications are made automatically.

1. The origin of PLABEL1 is noted as the RECODE command.
2. The property type is uniformly scalar.
3. Each property value, being a scalar quantity, has one row and one column as its dimensionality.
4. The level of PLABEL1 is equated with the level of PLABEL.
5. Both definition and creation modes are set to temporary.
6. The operational definition is the set of Boolean specifications which defines how values of PLABEL will be mapped into corresponding integral values (really serial category labels) of PLABEL1.

The primary function of the RECODE command is to prepare data for frequency counts, cross-tabulations, and related analytical procedures. Thus, if a frequency count of some property named PLABEL is desired, the RECODE command is issued to establish the categories or sub-intervals within its logical range. The resulting property PLABEL1 will encapsulate this partitioning of PLABEL. When created, property data corresponding to PLABEL1 will consist of serial integers indicating which category or cell each case falls into.

If a table is to be constructed around joint frequencies of two properties P1 and P2, then each property is first re-coded by issuing the RECODE command. Two new properties P11 and P21 are thereby defined. These indicate the row and column address, respectively, of each case in the table to be constructed.

There are two major advantages gained from setting up frequency counts and tables in this manner. First, it is very easy to change a specification by merely re-issuing the RECODE command. Categories, rows, and columns may thereby be collapsed and/or re-partitioned. Second, since the output of every RECODE command is a defined property, frequency counts and tables may be subjected to additional Boolean conditions via the DEFINE command. Thus, a conditional table may be specified via a pure logical transformation of the row and/or column addresses specified by the RECODE command. All analytical procedures which operate upon counts or tables may be similarly conditioned. This turns out to be a very powerful and useful device.

The fastest way through the RECODE command is to type a complete specification, followed by two carriage returns. The second return will obviate any verbal description. Additional returns are redundant, but nondamaging.

7.3.6 The ASSIGN Command

The ASSIGN command is used to define internally certain special properties. These special properties will facilitate analysis, although the corresponding property data will frequently lack any direct substantive interest.

Assuming the rapid input mode, the ASSIGN command is issued in any one of the following nine alternative forms.

$$\text{ASSIGN} \left\{ \begin{array}{l} \text{SERIAL} \\ \text{RANDOM} \\ \text{RANK} \end{array} \right\} \left\{ \begin{array}{l} \text{INTEGERS} \\ \text{NUMBERS} \\ \text{DECIMALS} \end{array} \right\} \text{ TO PLABEL PLABEL1}$$

In all nine forms, PLABEL is the name of the already-defined property which will constitute the basis of the special definition. PLABEL1 is the name to be given to the special property currently being defined.

The first form of the ASSIGN command is "ASSIGN SERIAL INTEGERS TO PLABEL PLABEL1." Its purpose is to define a new property named PLABEL1 on whichever level PLABEL resides. Property data corresponding to PLABEL1 will consist entirely of serial numbers ranging from 1 to NCS, where NCS is the number of cases on that level. These will be useful in certain ranking and sorting operations which the analyst may wish to perform.

The second form of the command is "ASSIGN SERIAL NUMBERS TO PLABEL PLABEL1." Its effect is identical to the first form.

The third form of the command is "ASSIGN SERIAL DECIMALS TO PLABEL PLABEL1." Its effect is identical to the first two forms, except that each serial number is transformed into a decimal fraction by dividing NCS into it.

The fourth form of the command is "ASSIGN RANDOM INTEGERS TO PLABEL PLABEL1." Here, serial integers are first assigned as above, but then their order is randomized. The result is one of the NCS permutations of the set of NCS serial integers. The randomizing device is such that every permutation is rendered equally likely. Also, successive issuances of the ASSIGN command will, in general, select a different permutation. Possible uses for these random serial integers are listed below:

1. to define a random sample of cases to be drawn from all cases on a given level (e.g., by issuing the DEFINE command containing a Boolean condition specifying a subinterval of these random integers);
2. to simulate statistical processes involving sampling without replacement; and
3. to facilitate split-half reliability testing on cases.

The fifth form of the command is "ASSIGN RANDOM NUMBERS TO PLABEL PLABEL1." This induces a set of NCS random numbers to be drawn

from a rectangular distribution defined over the closed interval $[0, 1]$ and to be assigned to the NCS cases on the level of PLABEL. Such random numbers might be useful in simulating statistical processes involving sampling with replacement.

The sixth form of the command is "ASSIGN RANDOM DECIMALS TO PLABEL PLABEL1." Its effect is identical to the RANDOM INTEGERS option, except that integers are converted to corresponding decimal fractions. This makes it possible to specify a random sub-sample as a percentage of all cases on a given level.

The seventh form of the command is "ASSIGN RANK INTEGERS TO PLABEL PLABEL1." This assigns rank numbers to PLABEL1 indicating the ordinal position of each case on PLABEL. Thus, the case with the numerically largest value on PLABEL will receive a one on PLABEL1, the second-highest case will receive a two, and so forth. Whenever two or more identical values are encountered on PLABEL, identical integers will be assigned to the corresponding cases on PLABEL1. Then, lower-valued cases will be assigned rank numbers as if no ties had occurred (e.g., 1 2 2 2 2 6 7 ...).

The eighth form of the command is "ASSIGN RANK NUMBERS TO PLABEL PLABEL1." This is identical to the RANK INTEGERS option, except that ties are handled differently. Strings of tied ranks are equated uniformly with their arithmetic mean (e.g., 1 3.5 3.5 3.5 3.5 6 7 ...).

The ninth form of the command is "ASSIGN RANK DECIMALS TO PLABEL PLABEL1." It is identical in effect to the RANK NUMBERS option,

except that rank numbers are converted to decimal fractions. Hence, they may be treated as percentile scores.

The ASSIGN command, in any of its nine forms, runs two error checks. These determine whether:

1. PLABEL is an already-defined property name identifying any property not on level zero (and scalar, if RANK is used); and
2. PLABEL1 is a legal name for a new property (see requirements in section 7.3.1).

If no errors are detected, a request for an optional verbal description will appear. This will be followed by the message "PLABEL1 HAS BEEN ASSIGNED," and then "OK."

Successful execution of the ASSIGN command updates the P-string in the usual manner. Automatic specifications are as follows.

1. The origin of PLABEL1 is noted as the ASSIGN command.
2. The property type is uniformly scalar, which implies a dimensionality of one row and one column per case.
3. The level of PLABEL1 is equated with the level of PLABEL.
4. Both definition and creation modes are set to temporary.
5. The operational definition is one of the nine (really eight) generating functions previously described.

The fastest way through the ASSIGN command is to type the single specification line, followed by two carriage returns. This will obviate any verbal description.

7.3.7 The Group Command

The GROUP command serves to group cases on one level in preparation for computing summary properties applicable to the next-higher level. In this manner, group means, standard deviations, etc. may be computed from individual case data.

Assuming the rapid input mode, the GROUP command is issued as follows.

```
GROUP PLABEL PLABEL1
```

PLABEL is the name of the property on the lower level whose cases are to be grouped. PLABEL1 is the name of the property on the next-higher level containing grouped data corresponding to PLABEL.

Action taken by the GROUP command is as follows. First, the specification is checked to determine whether:

1. PLABEL is an already-defined property name identifying either a scalar or a vector property not on level zero;
2. there exists at least one level higher than the level of PLABEL; and
3. PLABEL1 is a legal name for a new property (see requirements in section 7.3.1).

If no errors are detected, a request for an optional verbal description will appear. This will be followed by the message "PLABEL HAS BEEN GROUPED UNDER THE NAME OF PLABEL1," and then "OK."

Successful execution of the GROUP command updates the P-string in the usual manner. Automatic specifications are as follows.

1. The origin of PLABEL1 is noted as the GROUP command.
2. The property type is vector, if PLABEL is a scalar, or matrix, if PLABEL is a vector.
3. The number of rows in PLABEL1 is equated with the number of cases in the largest group formed, if PLABEL is a vector property, or one, if scalar.
4. The number of columns in PLABEL1 is equated with the number of cases in the largest group formed, if PLABEL is a scalar property, or vector length, if vector.
5. The level of PLABEL1 is set at one higher than the level of PLABEL.
6. Both definition and creation modes are set to temporary.
7. There is no operational definition of PLABEL1.

The number of cases contained in a group will not, in general, be the same across all groups. Hence, when property data corresponding to PLABEL1 are actually created (by the CREATE command), those groups containing fewer than the maximum number of cases will be assigned vector or matrix property values with one or more undefined elements. Undefined elements will always be the right-most elements in a row and the lowest elements in a column.

The fastest way through the GROUP command is to type the single specification line, followed by two carriage returns. This will obviate any verbal description.

7.3.8 The ECOUNT Command

The ECOUNT command counts the number of defined elements in any scalar, vector, or matrix property. It is particularly useful in counting the number of defined cases associated with any property defined by the GROUP command. It is also useful for various matrix algebra manipulations (e.g., computing the mean of all defined elements within a vector property).

Assuming the rapid input mode, the ECOUNT command is issued as follows.

```
ECOUNT PLABEL PLABEL1
```

PLABEL is the name of the property whose defined elements are to be counted. PLABEL1 is the name of the new property containing this count.

Action taken by the ECOUNT command is as follows. First, the specification is checked to determine whether:

1. PLABEL is an already-defined property name; and
2. PLABEL1 is a legal name for a new property (see requirements in section 7.3.1).

If no errors are detected, a request for an optional verbal description will appear. This will be followed by the message "PLABEL HAS BEEN COUNTED UNDER THE NAME OF PLABEL1," and then "OK."

Successful execution of the ECOUNT command updates the P-string in the usual manner. Automatic specifications are as follows.

1. The origin of PLABEL1 is noted as the ECOUNT command.
2. The property type is vector, if PLABEL is matrix. Otherwise, it is scalar.
3. The number of rows is uniformly set equal to one.
4. The number of columns is equated with the number of rows of PLABEL.
5. The level of PLABEL1 is equated with the level of PLABEL.
6. Both definition and creation modes are set to temporary.
7. There is no operational definition of PLABEL1.

The fastest way through the ECOUNT command is to type the single specification line followed by two carriage returns. This will obviate any verbal description.

7.3.9 The DESCRIBE Command

The DESCRIBE command is the single mechanism through which information about defined properties may be obtained. Such information can be useful as a reminder to the user of previous and possibly forgotten property definitions, as a means of "playing back" and, thereby, verifying current definitions, as a diagnostic device to ferret out the source of incomprehensible results, and as a vehicle of final documentation. The DESCRIBE command will provide information on any defined property, no matter which command was used to define it.

Assuming the rapid input mode, the DESCRIBE command is issued as follows.

DESCRIBE PLABEL (additional optional parameters)

PLABEL is the name of an already-defined property.

If the DESCRIBE command is issued as shown above without additional parameters, the effect will be a complete description of PLABEL. This will include the following items of information displayed on the console in the following order:

1. ORIGIN--the name of the command which defined PLABEL;
2. PTYPE--scalar, vector, or matrix, whichever property type PLABEL happens to be;
3. NROWS--the number of rows associated with each case of PLABEL;
4. NCOLS--the number of columns associated with each case of PLABEL;
5. LEVEL--the level on which PLABEL resides;
6. DFMODE--temporary or permanent, depending upon PLABEL's definition mode;
7. CRMODE--temporary or permanent, depending upon PLABEL's creation mode;
8. BLOCKN--the serial number of the data block within which corresponding property data reside, if such data have been created;

9. BLOCKA--the starting address of the first data word within the above data block, if such data have been created;
10. VDSC--a verbal description of PLABEL, if one exists; and
11. OPDEF--an operational definition of PLABEL, if one exists.

In many cases, the user will not wish all of this information. Any sub-set of the above items may be obtained by appending their keywords (the capitalized words immediately following the numbers on the above list) to the command line. These keywords constitute additional optional parameters to the DESCRIBE specification. They may be appended in any order. Also, any keyword may appear more than once, and undefined words may be interspersed with keywords. Redundancies and errors of this nature will be ignored without damage to DATANAL and without an error message.

In addition to these selectivity options, the DESCRIBE command provides a COMPLETE option. By appending the keyword "COMPLETE" anywhere after PLABEL on the specification line, whatever information was requested for PLABEL will also appear for all of the properties involved in the definition of PLABEL. The complete definitional "tree" is traced out, beginning with all of the basic properties underlying PLABEL, progressing through any intermediate definitions linking PLABEL to basic properties, and culminating in PLABEL itself. If PLABEL identifies a basic property, then only information on PLABEL will appear. This option is particularly useful in ferreting out the source of incomprehensible results and in providing automatic final documentation.

The DESCRIBE command checks only to see whether PLABEL is an already-defined property name. Its successful execution is indicated by the usual "OK" message.

7.3.10 The CHANGE Command

The CHANGE command serves to alter selected portions of an existing property definition. It also serves several special functions previously discussed.

Assuming the rapid input mode, the CHANGE command is issued in any one of the following five forms.

```
          / PLABEL   PLABEL   PLABEL1 \
         |  DFMODE   PLABEL   T  or P  |
CHANGE  <   CRMODE   PLABEL   T  or P   >
         |  LEVEL    PLABEL   O  or N  |
          \ VDSC     PLABEL      --   /
```

The first form of the CHANGE command is "CHANGE PLABEL PLABEL PLABEL1." The second word in this specification, "PLABEL," is a keyword and must be entered literally right after "CHANGE." It indicates that the label or name of a property is to be changed. PLABEL (the third word) is the name of the property which is to be changed. PLABEL1 is the new property name. A check is made to determine whether PLABEL is an already-defined property name. This is true for all five forms of the CHANGE command. If so, every reference to PLABEL in the P-string is changed to PLABEL1, except references contained within verbal descriptions.

The second form of the command is "CHANGE DFMODE PLABEL T or P." The keyword "DFMODE" indicates that the definition mode of PLABEL is to

be changed. "T" means changed to temporary. "P" means changed to permanent. Requesting that a mode be changed to the mode which already applies will cause no harm, nor will it induce an error message.

The third form of the command is "CHANGE CRMODE PLABEL T or P." Its effect is identical to the DFMODE form, except the creation mode is changed. (Note: neither the definition mode nor the creation mode of level-linkage identity tags may ever be changed from permanent to temporary.)

The fourth form of the command is "CHANGE LEVEL PLABEL 0 or N." Its effect is to change the level of PLABEL, subject to certain restrictions. The first action is to check whether PLABEL is an already-defined property name. The second action is to check whether any other properties are defined either directly or indirectly in terms of PLABEL. If so, an error message will appear, and the command will not be executed. The third action is to check whether a level change is requested to or from level zero. Scalar and vector properties may undergo a level change from any level to level zero. Vector properties may undergo a level change from level zero to any other level, provided that the number of columns in the vector equals exactly the number of cases on the new level. A scalar property on the new level will result. Matrix properties may undergo a level change from level zero to any other level, provided that the number of rows in the matrix equals exactly the number of cases on the new level. A (row) vector property on the new

level will result with the same number of columns as the original matrix. No other kinds of level changes are permitted.

The fifth form of the command is "CHANGE VDSC PLABEL." Assuming that PLABEL is an already-defined property name, explicit instructions will appear to guide the user in appending a verbal description to PLABEL.

None of these five forms of the CHANGE command affect property data. Only the P-string is affected. It is updated immediately.

Successful execution of all forms is indicated by the message "PLABEL HAS BEEN CHANGED," followed by "OK."

7.3.11 The ERASE Command

The ERASE command is used to delete property definitions and corresponding property data from a working data base. This can be useful as a storage-saving device and as a means of clearing out space on the P-string in the very unlikely event that an overflow occurs.

Assuming the rapid input mode, the ERASE command is issued as follows.

```
ERASE PLABEL1 PLABEL2 ... PLABELN
```

PLABEL1, PLABEL2, ..., PLABELN are already-defined names. The "CONT." option is available up to 100 words.

Action taken by the ERASE command is as follows. First, each listed property is inspected to determine whether:

1. it is an already-defined property;
2. its definition mode is temporary;
3. every property defined in terms of it (either directly or indirectly) is in temporary definition mode; and
4. property data corresponding to it and to every property defined in terms of it (either directly or indirectly) are in temporary creation mode, whether or not these data actually exist.

Violations on any of the above counts will induce an error message and will preclude execution.

For those listed properties which pass all of the above checks, corresponding property data are removed from the data blocks. Then, the property-defining information is removed from the P-string. Both the P-string and the data blocks are then re-packed to economize on disk storage space.

Successful execution is indicated by the message "PLABEL HAS BEEN ERASED" for each erased property. The last such message is followed by the usual "OK."

7.4 <u>Data-Creating and Data-Destroying Commands</u>

There is a single command in DATANAL to implement operational definitions generated by all of the property-defining commands. This is the CREATE command. It is the primary vehicle for introducing property data

into a working data base. Except for the SETUP and ESTABLISH commands (which create level-linkage identity tags) and certain rapid analysis commands (to be discussed in section 8.1), it is the sole vehicle.

There is another command, the DESTROY command, for removing data from a working data base. It may be used to eliminate intermediate results no longer of interest to the user, redundant data generated by duplicating various properties, and irrelevant data. It may also be used to free up disk storage space whenever a working data base is to be retained in a dormant state for an extended period of time. Recall that property data may always be re-created, so long as the property definition remains on the P-string.

7.4.1 The CREATE Command

Although the most complex from the programming point of view, the CREATE command is one of the simplest to use. Assuming the rapid input mode, the CREATE command is issued as follows.

```
CREATE PLABEL1 PLABEL2 ... PLABELN
```

PLABEL1, PLABEL2, ..., PLABELN are the names of already-defined properties. The "CONT." option is available up to 100 words.

The first step taken is to check each listed property to determine whether:

1. it is an already-defined property; and

2. the corresponding property data have already been created.

If the property is undefined, an error message will appear. If property data already exist, the specification will be ignored without damage and without an error message.

The next step is to determine which command defined each remaining property on the specification list. Those defined by the READ command are segregated for early processing, since these are uniformly basic. If none are found, the next few steps are skipped.

All properties defined by the READ command are then re-ordered according to their field address in the raw data base. This permits a single pass through the raw data base, unless the total amount of data overflows maximum core capacity, less reading instructions. Should an overflow appear imminent, multiple passes will be made. These activities follow a single error check to determine whether the raw data base still exists on the user's disk.

Successful execution is indicated by the message "PLABEL HAS BEEN CREATED" for each listed property. This means that the corresponding property data have been transferred to one of the data blocks and that both the block number and the block address have been placed on the P-string.

Whenever missing or illegal (i.e., nonnumeric) data are encountered in the raw data base, the corresponding data in the working data base will be undefined. This occurs without hesitation and without

any error message. The number of such undefined observations may be determined subsequently by issuing the DESTAT command (see section 7.5.2). Their identity may be determined by issuing the DISPLAY command (see section 7.5.1).

All remaining basic properties (i.e., pure numbers entered via the SET command and serial or random numbers generated by the ASSIGN command), and all remaining nonbasic properties are created next. The exact procedure varies with the definitional origin of each property.

Let us begin with properties defined by the DEFINE command. The overall procedure is as follows.

1. Start with the first case.
2. Determine whether any Boolean conditions exist. If so, proceed to the next step. Otherwise, proceed to step 7.
3. Evaluate the algebraic expression associated with an elementary condition.
4. Compare this computed value with the current value of (i.e., the value of this case on) the corresponding PLABEL according to the indicated numeric relationship.
5. If the indicated numeric relationship is satisfied, proceed to the next AND-ed condition, and return to step 3. Proceed to step 7, if all AND-ed conditions have been tested. If the indicated numeric relationship is not satisfied, and if this is either an AND-ed condition or the last in a string of uniformly unsatisfied OR-ed conditions,

proceed to step 6. Otherwise, proceed to the next condition in the OR-ed string, and return to step 3.

6. The set of Boolean conditions has not been satisfied. Consequently, the value of this case on the property being created is undefined. Proceed to the next case, and return to step 3.
7. The set of Boolean conditions (if any exist) has been satisfied. Consequently, evaluate the algebraic expression or collect property data, if a vector/matrix specification is involved, and store the results on the data block. Then, proceed to the next case, and return to step 2.

Within the above general procedures are imbedded several uniform conventions. These are listed below.

1. Boolean conditions are tested serially. The testing order is determined by the specification order in the DEFINE command. Consequently, the user may render the creation process more efficient by manipulating this specification order in two ways. First, he may order strings of OR-ed conditions in decreasing a priori likelihood of being satisfied. Second, he may order AND-ed conditions in increasing a priori likelihood of being satisfied. Although not essential to obtaining an answer, these maneuvres will generally obtain one more quickly.

2. When evaluating any expression in a Boolean condition, every term must be completely defined. Otherwise, that condition is considered unsatisfied. Thus, every element in a vector or matrix property involved in a Boolean condition must be defined to satisfy that condition. This is a necessary, but not a sufficient criterion.
3. In contrast, expressions involved in algebraic transformations and elements collected to form vectors/matrices need not be completely defined to obtain a defined result. If a vector or matrix is being created, some of its elements may be defined, while others are not.
4. The evaluation of all algebraic expressions is protected against termination due to illegal computations. This is accomplished by a series of "look-aheads" built into the evaluation mechanism. If an illegal computation appears imminent (e.g., division by zero, square root of a negative number, etc.), that computation is skipped, and the result is undefined. Processing will then continue normally.
5. Whenever vectors or matrices are involved in any algebraic expression, a resulting element will be defined if and only if all elements involved in the computation are defined, and the computation is legal. This statement does not apply to multiplication. Here, only enough

elements need be defined to obtain at least one term in the cumulative sum. The remaining undefined terms are ignored. This device is extremely useful for defining a series of alternative functions sharing the same range and domain, but applying under mutually exclusive Boolean conditions. Collecting these functional definitions in a vector and then post-multiplying them by a vector of ones serves to define a conditional function whose shape or form may be altered automatically according to the current values of whichever properties are contained in the Boolean conditions.

The creation of properties defined by the SET and COMPUTE commands proceeds as discussed above, except that no Boolean conditions are involved.

Properties defined by the RECODE command are created as follows.

1. Start with the first case.
2. Test the set of Boolean conditions associated with the first specification. Boolean conditions are tested as previously described. If satisfied, assign the integer one to this case, proceed to the next case, and repeat this step. Otherwise, proceed to the next specification, and repeat this step. Continue in this manner until either one of the specifications is satisfied or the list of specifications is exhausted. If satisfaction is achieved, assign the serial number of the satisfied

specification as the value of this case, proceed to the next case, and repeat this step. If all specifications are exhausted without satisfaction, this case is undefined. Proceed to the next case, and repeat this step.

Properties defined by all other commands are created in a straightforward manner. No special explanation is required.

One additional feature of the CREATE command is its ability to determine automatically which properties must have been created first in order to create a specified property, to determine the order in which such computations must be performed, and to create all necessary intermediate properties without explicit direction. Thus, if a lengthy definitional chain is established by the user, but he is only interested in the final results, he need only request that the last property in the chain be created. He need not remember the names of intermediate properties, nor request their creation explicitly. This is particularly helpful when a user returns to a working data base after several months of inactivity.

Successful execution of the CREATE command is indicated by the message "PLABEL HAS BEEN CREATED" for each property. An "OK" will follow the last such message. This means that all computed data have been transferred to data blocks and that block numbers and addresses have been appended to the P-string.

7.4.2 The DESTROY Command

The DESTROY command removes property data from a working data base and re-packs those data blocks affected by the removal to conserve disk storage space. This is accomplished without erasing any property definitions from the P-string.

Assuming the rapid input mode, the DESTROY command is issued as follows.

```
DESTROY PLABEL1 PLABEL2 ... PLABELN
```

PLABEL1, PLABEL2, ..., PLABELN are names of already-defined properties. The "CONT." option is available up to 100 words.

Undefined properties induce error messages, but redundant specifications are ignored without damage and without any explicit messages. Requests to destroy nonexistent data are treated as redundant.

Successful execution of the DESTROY command is indicated by the message "PLABEL HAS BEEN DESTROYED" for each property. An "OK" will follow the last such message.

The user should be prepared to see many changes in the data addresses of remaining properties following the DESTROY command. This is because all data blocks are re-packed to eliminate "holes" created thereby.

7.5 Data-Summarizing and Data-Display Commands

DATANAL contains five commands for summarizing and displaying already-created property data. The most basic of these is the DISPLAY command. It is to property data what the DESCRIBE command is to property definitions. Specific results may be selected for display on the console. However, no attempt is made within the DISPLAY command either to reorder or to summarize data in any manner.

The DESTAT command computes and displays four descriptive statistics for any given property:

1. its mean;
2. its median;
3. its standard deviation; and
4. the number of cases defined on it.

The RANK command ranks property data (numerically highest first) prior to display on the console. However, the order of cases is not changed within data blocks.

The COUNT command provides absolute and relative frequency information about a property. Categories must be pre-established by the RECODE command before a count may be effected.

The TABLE command produces and displays two-dimensional tables. The contents of each row and column of a table must have been pre-established by the RECODE command.

Before discussing these five commands individually, it would be well to describe the several conventions which apply uniformly to all of them. This is done below.

1. Execution of all commands is preceded by two error checks. One determines whether every specified property has been defined. The other determines whether the corresponding data have been created. Only if both checks are passed will action continue. Otherwise, the command will terminate with an appropriate error message. None of these commands alter the contents of the P-string or any data block.
2. All results are printed on the console without format specification. Each command figures out its own format according to the particular characteristics of the data involved. This includes integer/decimal discrimination, lining up decimal points for column arrays, horizontal spacing for vector displays, both horizontal and vertical spacing for matrix displays, and other such decisions. The user need never worry about these annoying details.
3. Asterisks (*) are displayed uniformly to indicate undefined property values.
4. Every command may be conditioned in either or both of the following ways. First, a sub-set of N cases may be obtained by appending an integer N to the specification list. Second, this

can be made a random sub-set by following N with the keyword "RANDOM." Without "RANDOM," the sub-set will be serial (i.e., it will involve the first N cases). This N RANDOM option is also available for all of the rapid analysis commands, as well as for all private, user-defined commands. It is particularly useful for "getting a feel" for one's data and for performing partial analyses without overworking the computer needlessly (e.g., by analyzing all cases).

5. Whenever a sufficient quantity of information has been displayed, a manual abort is always possible and frequently desirable. The one-push variety is recommended.

6. Successful execution of all five commands is indicated by "OK."

7.5.1 <u>The DISPLAY Command</u>

Assuming the rapid input mode, the DISPLAY command is issued as follows.

```
DISPLAY PLABEL N RANDOM (optional)
```

PLABEL is the name of an already-defined and already-created property. The N RANDOM option is available. N may be specified without RANDOM. If specified, RANDOM must be preceded by N. PLABEL must precede both. These statements apply to all commands.

Without the N RANDOM option, the effect is to display PLABEL on the console. All cases are displayed according to the following format conventions.

1. Each case is preceded by a serial integer. This is to facilitate visual reference only. Serial integers should not be confused with property data.
2. Scalar property data are displayed in a single column of successive cases.
3. Vector property data are displayed row-wise, if possible. That is, all of the elements in a vector are layed out along a single row, and the next vector associated with the next case appears as the next row. If vectors cannot fit along a row, they will be layed out column-wise (i.e., successive single-column displays, each preceded by a serial integer). Vectors can always be layed out column-wise, since the display process is indefinitely expandable in the vertical direction.
4. Matrix property data are displayed as is, if they can fit. Once again, the constraint lies in the horizontal direction. Users are urged to define matrices so that the number of columns never exceeds the number of rows. This will increase the likelihood of a fit, since the number and width of columns is the only thing that can cause an

overflow. Should an overflow occur, an error message will appear, and no display will occur.

If N is specified, but without RANDOM, the first N cases will be displayed according to the above conventions.

If RANDOM is also specified, N cases will be displayed as before, but it will be a random selection of N cases. Normally, the selection will differ over successive issuances of the same command.

7.5.2 <u>The DESTAT Command</u>

Assuming the rapid input mode, the DESTAT command is issued as follows.

```
DESTAT PLABEL N RANDOM (optional)
```

PLABEL is the name of an already-defined and already-created property.

Mean, median, standard deviation, and number of defined cases is computed and displayed for PLABEL. A separate scalar, vector, or matrix of results is displayed in each of the above four instances, depending on whether PLABEL is a scalar, vector, or matrix property, respectively. All summarizing computations are made on an element-by-element basis. Format conventions are as discussed under the DISPLAY command.

7.5.3. <u>The RANK Command</u>

Assuming the rapid input mode, the RANK command is issued as follows.

```
RANK PLABEL N RANDOM (optional)
```

PLABEL is the name of an already-defined and already-created property.

Property data under PLABEL are ranked according to the following conventions.

1. Highest numeric values appear first.
2. All undefined cases are pushed to the bottom and appear after the lowest defined case.
3. If PLABEL is a vector or matrix property, ranking will be performed on the basis of the first element.
4. Format conventions are as discussed under the DISPLAY command.

7.5.4 The COUNT Command

Assuming the rapid input mode, the COUNT command is issued as follows.

```
COUNT PLABEL N RANDOM (optional)
```

PLABEL is the name of an already-defined and already-created property.

The COUNT command will only operate upon properties defined by the RECODE command, possibly modified by a string of DEFINE commands involving only duplications subject to Boolean conditions (i.e., no algebraic transformations are permitted). Then, four kinds of frequency information are computed and displayed on the console:

1. absolute frequency of defined cases in each category (i.e., a simple count);
2. cumulative absolute frequencies;
3. relative frequency of defined cases in each category (i.e., a percentage distribution); and
4. cumulative relative frequencies.

Undefined cases are uniformly ignored.

7.5.5 The TABLE Command

Assuming the rapid input mode, the TABLE command is issued as follows.

```
TABLE PLABEL1 PLABEL2 N RANDOM (optional)
```

PLABEL1 and PLABEL2 are the names of two already-defined and already-created properties.

Like the COUNT command, the TABLE command operates only upon properties defined by the RECODE command, possibly modified by a string of restricted DEFINE commands. These restrictions apply to both PLABEL1 and PLABEL2.

The TABLE command produces and displays four tables:

1. joint absolute frequencies resulting from a cross-tabulation of PLABEL1 and PLABEL2;
2. corresponding joint relative frequencies;
3. corresponding conditional relative frequencies by row; and
4. corresponding conditional relative frequencies by column.

The categories of PLABEL1 form the rows of each table, while the categories of PLABEL2 form the columns. Only cases defined on both PLABEL1 and PLABEL2 are tabled. All others are ignored.

Row and column totals are computed internally. Also, display formats are determined without specification. To reduce the chances of a horizontal overflow in the display of these tables, they should be requested so that the number of columns never exceeds the number of rows. As in the DISPLAY command, horizontal overflows will induce an error message and will preclude the display of any further results.

8.0 THE STANDARD ANALYTICAL COMMANDS OF DATANAL

DATANAL comes equipped with a set of standard analytical commands. Most of these execute statistical procedures, many of which are nonparametric or distribution-free in terms of their assumptions.

Almost all of the statistical procedures are available in both a rapid and a guided version. The rapid analysis versions operate directly upon property data prepared by the various commands described in section 7. Guided versions are essentially independent of DATANAL. Once called (by issuing the EXECUTE command), they are completely self-contained. They provide explicit instructions for typing data into the computer. They also provide detailed diagnostic and interpretive information. Their primary value lies in training beginners and in testing hunches which involve small-to-intermediate amounts of data. In addition, they have proven highly successful as single-shot demonstration devices, particularly where the objective of the demonstration is pedagogical. However, the rapid versions are recommended for frequent and experienced users.

DATANAL also contains certain built-in synonym relationships. These permit a relatively small number of coded routines to be referenced by a larger number of names or labels. The details of these relationships will be discussed in section 8.2.

8.1 Rapid Analysis Commands

There are eight rapid analysis commands in DATANAL. All of these are issued with a single-line specification. Also, the N RANDOM option discussed in section 7.5 is universally available.

Unless otherwise specified, internal core capacity restrictions are variable. That is, imminent overflows will not be detected until execution is attempted, at which time an error message will appear. This situation can often be remedied by re-issuing the same command conditioned by some form of the N RANDOM option.

Some of these commands provide the option of saving computed results for subsequent use. None of them destroy prepared input data. When the save option is exercised, a new property is both defined and created in temporary mode. The P-string is updated accordingly, and corresponding property data (saved from the analysis) are deposited in one of the data blocks. Subsequent issues of the DESCRIBE command will identify the origin of such properties as "some procedure." All rapid analysis commands ignore or skip undefined property data.

8.1.1 The CNTING Command

The CNTING command performs statistical analyses on contingency tables. It may be used to execute a Chi square test of either homogeneity or independence, a Fisher exact test of either homogeneity or independence, and a median test on absolute frequency data.

Assuming the rapid input mode, the CNTING command is issued in either of the following two forms.

CNTING PLABEL1 PLABEL2 N RANDOM (optional)

PLABEL1 and PLABEL2 are the names of two already-defined and already-created scalar properties residing on the same nonzero level.

CNTING PLABEL N RANDOM (optional)

PLABEL is the name of an already-defined and already-created matrix property on level zero.

Restrictions on the CNTING command are as follows.

1. Both PLABEL1 and PLABEL2 must have been defined by the RECODE command, possibly modified by a string of DEFINE commands involving only logical transformations. This restriction applies to the first form of the command.
2. PLABEL must contain absolute frequency data only, if the second form of the command is used.
3. Contingency tables cannot exceed 25 rows and/or 25 columns.
4. There are additional restrictions. However, they are numerous and highly specific to the particular input data received. Also, when violated, appropriate error message will appear on the console. Consequently, they will not be discussed here.

Assuming the right input code, the OUTPUT command is issued in either of the following two forms

OUTPUT PLABEL1 PLABEL2 = NAME3 (random).

PLABEL1 and PLABEL2 are the names of two already-defined and already-created scalar properties [illegible] (or [illegible] level).

OUTPUT PLABEL [illegible]

PLABEL is the name [illegible] already-created matrix property [illegible]

[illegible] follows.

[illegible] by the

[illegible] of OUTPUT

[illegible] This

[illegible]

On the basis of whatever data are received, the CNTING command will select automatically a particular kind of analysis, perform it if possible, and print out the following information:

1. a joint absolute frequency table;
2. a declaration of what kind of analysis was performed (e.g., a Chi square or Fisher test);
3. the computed value of Chi square, if it was computed;
4. the exact 1-tail and 2-tail probabilities of occurrence of a Chi square value (or Fisher table) at least as extreme as the one generated, assuming either homogeneous samples or statistically independent properties; and
5. measures of predictive efficacy.

There is no save option associated with the CNTING command. The computations are so simple and the results are so few that it pays to repeat the command, if a review of results is desired.

8.1.2 The T-TEST Command

The T-TEST command performs an uncorrelated T test on two samples containing independently drawn cardinal data. As an added bonus, the Mann-Whitney U test is performed on the same data.

Assuming the rapid input mode, the T-TEST command is issued as follows.

```
T-TEST PLABEL1 PLABEL2 N RANDOM (optional)
```

PLABEL1 and PLABEL2 are the names of two already-defined and already-created properties residing on the same nonzero level.

If both PLABEL1 and PLABEL2 are scalar properties, results will be displayed on the console with no save option. If both are vector properties, analyses will be performed on a case-by-case basis. The results for each case will be displayed on the console, and the save option will be offered. If exercised, a vector property will be defined and created under a name provided, upon request, by the user. All other kinds of inputs will be rejected with an appropriate error message.

Additional restrictions exist. These will be flagged by error messages upon execution, if and only if they are violated.

Outputs include the following items in the following order (note: this is the order in which they will be stored, if a vector property is created by exercising the save option):

1. the sample mean under PLABEL1;
2. the sample mean under PLABEL2;
3. the computed value of T;
4. the exact 2-tail probability of occurrence associated with the computed value of T;
5. the computed value of U; and
6. the exact 2-tail probability of occurrence associated with the computed value of U. (Note: either the DESTAT or the ECOUNT command may be issued to determine relevant sample sizes.)

8.1.3 The TOTEST Command

The TOTEST command performs a correlated T test on two samples of matched pairs of independently drawn cardinal data. As added bonuses, the Wilcoxon matched-pairs signed-ranks test and the binomial sign test are performed on the same data.

Assuming the rapid input mode, the TOTEST command is issued as follows.

```
TOTEST PLABEL N RANDOM (optional)
```

PLABEL is the name of an already-defined and already-created property residing on any level except zero.

If PLABEL is a two-element vector property containing matched pairs of data, results will be displayed on the console with no save option. If PLABEL is a matrix property with two rows of matched pairs, analysis will be performed on a case-by-case basis. The results for each case will be displayed on the console, and the save option will be offered. If exercised, a vector property will be defined and created under a name provided, upon request, by the user. All other kinds of inputs will be rejected with an appropriate error message.

Additional restrictions exist. These will be flagged by error messages at execution time, if and only if they are violated.

Outputs include the following items in the following order (note: this is the order in which they will be stored, if a vector property is created by exercising the save option):

1. the mean difference computed by subtracting each second member of a pair from the first and averaging the results;
2. the computed value of T, assuming zero mean difference;
3. the exact 2-tail probability of occurrence associated with the computed value of T;
4. the computed value of the Wilcoxon matched-pairs signed-ranks statistic;
5. the exact 2-tail probability of occurrence associated with the Wilcoxon statistic;
6. the number of positive differences computed;
7. the number of negative differences computed;
8. the exact binomial probability of occurrence of the maximum of 6 or 7 above; and
9. the number of completely defined matched pairs involved in computing all of the above results.

8.1.4 The ANLVR1 Command

The ANLVR1 command performs a one-way analysis of variance on multiple samples of independently drawn cardinal data.

Assuming the rapid input mode, the ANLVR1 command is issued as follows.

```
ANLVR1 PLABEL1 PLABEL2 N RANDOM (optional)
```

PLABEL1 and PLABEL2 are the names of two already-defined and already-created scalar properties on the same nonzero level.

PLABEL1 serves as a classificatory or "cutting" property. It defines the sub-sample into which each case will fall. PLABEL2 is the property to be analyzed (frequently some measure of an experimental effect). PLABEL1 must have been defined by the RECODE command, possibly modified by a string of DEFINE commands involving only logical transformations. PLABEL2 must be a scalar property.

Additional restrictions exist. These will be flagged by error messages at execution time, if and only if they are violated.

Outputs include:

1. sub-sample means on PLABEL2;
2. sub-sample sizes (i.e., number of defined cases);
3. the computed value of the F ratio;
4. the two degree of freedom numbers associated with F; and
5. the exact 2-tail probability of occurrence associated with the computed value of F.

There is no save option available.

8.1.5 The ANLVR2 Command

The ANLVR2 command performs a two-way analysis of variance on independently drawn cardinal data.

Assuming the rapid input mode, the ANLVR2 command is issued as follows.

```
ANLVR2 PLABEL1 PLABEL2 PLABEL3 N RANDOM (optional)
```

PLABEL1, PLABEL2 and PLABEL3 are the names of three already-defined and already-created scalar properties on the same nonzero level.

Both PLABEL1 and PLABEL2 serve as classificatory or "cutting" properties. Together they define a two-way table (frequently indicating different experimental treatments and/or levels of treatment). PLABEL3 is the property to be analyzed (frequently some measure of an experimental effect). Both PLABEL1 and PLABEL2 must have been defined by the RECODE command, possibly modified by a string of DEFINE commands involving only logical transformations. PLABEL3 must be a scalar property.

Additional restrictions exist. These will be flagged by error messages at execution time, if and only if they are violated.

Outputs include:

1. a table of joint absolute frequencies indicating various cell sizes (note: the ANLVR2 command will only operate upon tables with identical cell sizes uniformly greater than one);
2. a table of means on PLABEL3 including all cell means, row means, column means, and the grand mean;

3. computed F ratios for the row effect, column effect, and interaction effect, respectively;
4. associated degrees of freedom for each of the above;
5. exact 2-tail probabilities of occurrence associated with each of the above; and
6. measures of predictive efficacy.

There is no save option available.

8.1.6 The NRMTST Command

The NRMTST command performs a general test of goodness-of-fit between cardinal sample data and an assumed normal population with unknown mean and variance. As an added bonus, the symmetry of the population is also tested.

Assuming the rapid input mode, the NRMTST command is issued as follows.

```
NRMTST PLABEL N RANDOM (optional)
```

PLABEL is the name of an already-defined and already-created property on any level except zero.

At least twenty defined cases are required to execute a normality test. Four cases suffice for the symmetry test. Additional restrictions will be flagged at execution time, if and only if they are violated.

If PLABEL is a scalar property, results will be printed on the console with no save option. If PLABEL is a vector property, analyses will be performed on a case-by-case basis. The results for each case will be displayed on the console, and the save option will be offered. If exercised, a vector property will be defined and created under a name provided, upon request, by the user. All other kinds of inputs will be rejected with an approperiate error message.

Outputs include the following items in the following order (note: this is the order in which they will be stored, if a vector property is created by exercising the save option):

1. the computed value of Chi square (indicating goodness-of-fit to a normal population);
2. the associated degrees of freedom;
3. the exact 2-tail probability of occurrence associated with the computed value of Chi square;
4. the computed vale of the Wilcoxon matched-pairs signed-ranks statistic (indicating goodness-of-fit to a symmetric population); and
5. the exact 2-tail probability of occurrence associated with the computed value of the Wilcoxon statistic.

8.1.7 The CORREL Command

The CORREL command performs both linear (Pearson) and rank-order (Kendall) correlation analyses. It also computes selected partial correlation coefficients, if requested.

Assuming the rapid input mode, the CORREL command is issued as follows.

```
CORREL PLABEL1 PLABEL2 .... PLABELN N RANDOM (optional)
```

PLABEL1, PLABEL2, ..., PLABELN are the names of already-defined and already-created properties residing on the same nonzero level.

If all of the listed properties are scalar, results will appear on the console with no save option. If all are vectors containing the same number of elements, and if the elements are matched across vectors, analyses will be performed on a case-by-case basis. The results for each case will be displayed on the console, and the save option will be offered. If exercised, a set of six matrix properties will be defined and created under names provided, upon request, by the user. All other kinds of inputs will be rejected with an appropriate error message.

No more than six properties may be intercorrelated in a single command. Additional restrictions will be flagged at execution time, if and only if they are violated.

Outputs include the following items in the following order:

1. a matrix of Pearson product-moment correlation coefficients;
2. a corresponding matrix of exact 2-tail probabilities of occurrence;

3. a corresponding matrix of sample sizes;
4. a matrix of Kendall rank-order correlation coefficients;
5. a corresponding matrix of exact 2-tail probabilities of occurrence; and
6. a corresponding matrix of sample sizes.

8.1.8 <u>The LINFIT Command</u>

The LINFIT command fits a linear function to cardinal property data via least squares. Alternatively, a polynomial may be fitted by the same technique.

Assuming the rapid input mode, the LINFIT command is issued as follows.

```
LINFIT PLABEL1 PLABEL2 N RANDOM (optional)
```

PLABEL1 and PLABEL2 are the names of two properties on the same nonzero level.

PLABEL1 is the property to be fitted (e.g., the dependent variable), and PLABEL2 contains the fitting data (e.g., the independent variable or variables). If PLABEL1 is a scalar property, then PLABEL must be either a scalar or a vector property. If PLABEL1 is a vector property, then PLABEL2 must be either a vector or a matrix property. All other inputs will be rejected with appropriate error messages.

If PLABEL1 is a scalar property, results will appear on the console with no save option. If PLABEL1 is a vector property, analyses will be performed on a case-by-case basis. The results for each case will be displayed on the console, and the save option will be offered. If exercised, a vector property will be defined and created under a name provided, upon request, by the user.

Additional restrictions will be flagged at execution time, if and only if they are violated.

Outputs include the following items in the following order (note: this is the order in which they will appear if a vector property is created by exercising the save option):

1. the linear fitting coefficients, starting with a constant-added term; and
2. the proportional variance reduced by utilizing information on the fitting data.

8.2 Established Synonyms

The rapid analysis commands discussed in the previous section are really complex programs with multiple entry points. Issuing such a command normally exercises all of the possible options contained therein. Alternatively, one of the synonym commands may be issued to exercise some or all of these same options. When only some of the options are

exercised, the resulting output is reduced accordingly. Those outputs listed in section 8.1 which are not relevant to the synonym command are simply omitted.

There are fourteen synonym commands built into DATANAL. Each of these will be discussed in the remainder of this section. Unless otherwise stated, the conventions which apply to the particular rapid analysis command for which each is a synonym apply equally to the synonym command itself.

Three synonyms exist for the CNTING command. These are "HOMNOM," "FISHER," and "MEDTST." Each of these may be issued by substituting the appropriate synonym for "CNTING" on the single-line command specification (see section 8.1.1). Unlike the other synonyms to be discussed in this section, these three do not alter the logical flow through the command at all. They serve only to suggest by their names somewhat different uses to which the same analytical procedure may be applied.

HOMNOM stands for a homogeneity analysis to be performed on strictly nominal or classificatory data. Thus, cases may be partitioned into two or more sub-samples according to one nominal property (e.g., human beings may be partitioned on the basis of their sex). Then, the conditional relative frequency distributions of each sub-sample may be computed on some other nominal property (e.g., hair color). If the conditional relative frequency distributions are homogeneous, then the partitioning property may be considered devoid of influence in terms of the

other property. If significant heterogeneity emerges, then some sort of influence (possibly causal) may be inferred. HOMNOM may be used in this manner as a nominal equivalent of the more popular one-way analysis of variance.

FISHER stands for the Fisher exact test of small, two-by-two contingency tables. It may be used to perform a test of homogeneity between two samples on the basis of a dichotomous property. It may also be used to test the statistical independence of two dichotomous properties. In either case, the total number of cases distributed within the two-by-two table cannot exceed thirty.

MEDTST stands for a median test. It may be used either as a homogeneity or as an independence test. In both cases, a two-by-two table is assumed, wherein either or both properties are defined in terms of dichotomous median splits. The DESTAT command may be used to determine where the median falls on any ordinal or cardinal property.

A single synonym exists for the T-TEST command. By substituting "U-TEST" for "T-TEST" in the command specification, only a Mann-Whitney U Test is performed, and only U Test statistics are printed out. Otherwise, the description in section 8.1.2 applies.

The TOTEST command possesses two synonyms. Substituting "WILCXN" in the command specification restricts both analyses and outputs to the Wilcoxon matched-pairs signed-ranks test and the binomial sign test. Substituting "SINTST" causes only the sign test to be performed and reported. Otherwise, the description in section 8.1.3 applies.

Execution of the NRMTST command may be restricted to performing and reporting a symmetry test by substituting "SYMTST" in the command specification (see section 8.1.4).

Substitution of "PEARSN" for "CORREL" will restrict both computations and outputs to Pearson product-moment coefficients. Substitution of "KENDAL" will produce Kendall rank-order coefficients only. Otherwise, the description in section 8.1.7 applies.

Finally, there are five synonyms for the LINFIT command. These alter the form of the fitting function, but they do not change the output format. The five synonyms are:

1. POLFIT--fits a polynomial function to collected data;
2. LINPOS--same as LINFIT, except all fitting coefficients are constrained to be nonnegative;
3. LINHOM--same as LINFIT, except a homogeneous linear function is fitted (i.e., the constant-added term is eliminated from the fitting function);
4. POSHOM--a combination of LINPOS and LINHOM, where a homogeneous linear function with nonnegative coefficients is fitted; and
5. WTFIND--same as POSHOM, except that the nonnegative coefficients are required to sum to one.

8.3 Guided Analysis Commands

Guided versions of all rapid analysis commands and their synonyms may be executed by issuing a single command. This is the EXECUTE command. Assuming the rapid input mode, it is issued as follows.

```
EXECUTE GCNAME
```

GCNAME is the name of the guided version desired. There are eighteen such guided commands available. Since all of these were written up in a separate memorandum, they will not be discussed here. The reader is referred to On-Line Analysis for Social Scientists, Sloan School Working Paper Number 226-66, November 1966.

9.0 PRIVATE USER-PREPARED COMMANDS

DATANAL is equipped to incorporate any number of private commands prepared by a user. The only restriction here is that no more than 430 command names may reside within the special supervisor (i.e., within the user's CURENT COMNDS file) at any given moment.

The procedure for preparing and attaching a private command is as follows.

1. Code the private command in some language which may be compiled or assembled into machine instructions.
2. Insert a special subroutine called "LINK" at the head of the coded program. LINK will serve both to interpret the private command specification, when it is later issued, and to read all referenced property data into core storage. It also mediates the N RANDOM option.
3. Compile and load the private command.
4. Save the core image of the loaded program under some appropriate name. This name will henceforward be the command name.
5. Issue the ATTACH command to inform DATANAL that a new command exists. The name of the saved core image must be used in the ATTACH command.

From that moment on, DATANAL will recognize and execute the private command. The private command is issued as follows.

```
CNAME PLABEL1 PLABEL2 ... PLABELN N RANDOM (optional)
```

CNAME is the name of the private command. PLABEL1, PLABEL2, ..., PLABELN are the names of already-defined and already-created properties. As indicated, the N RANDOM option is universally available.

Detailed instructions for acquiring and using the LINK subroutine may be obtained by contacting either James R. Miller or Christopher R. Sprague at the Sloan School of Management, Massachusetts Institute of Technology.

10.0 OBTAINING ACCESS TO DATANAL

At the time of this writing, the programs of DATANAL are contained in the disk files of T169 2750 and T169 CMFL01 on Project MAC. It is hoped that the entire system will be moved to the Computation Center by the end of 1967. In the meantime, access to the system on MAC is freely available through linkages.

Links must be established to the following specific files on T169 CMFL01:

1. ATANAL SAVED
2. CREATE SAVED
3. SETUP SAVED
4. XECUTE SAVED
5. CNTING SAVED
6. T-TEST SAVED
7. TOTEST SAVED
8. ANLVR1 SAVED
9. ANLVR2 SAVED
10. NRMTST SAVED
11. CORREL SAVED
12. LINFIT SAVED

These links will mediate access to all commands in DATAMAL except the guided analysis routines. Additional links must be established to

T169 2750 to obtain access to these routines. Specific linkage instructions appear in <u>On-Line Analysis for Social Scientists</u>, Sloan School Working Paper Number 226-66. (Note: When using the guided routines, be sure that linkage is made to the version of XECUTE SAVED on T169 CMFL01 and not to the version on T169 2750. Otherwise, users will be bounced out of DATANAL following every guided routine.)

REFERENCES

1. Miller, J. R. On-Line Analysis for Social Scientists, Sloan School Working Paper Number 226-66, November 1966.

Copy 2