Research Report

KSTS/RR-90/001
Feb. 7, 1990

Data and Description Rule

by

R. Shibata, M. Sibuya and M. Takagiwa

R. Shibata, M. Sibuya and M. Takagiwa

Department of Mathematics
Faculty of Science and Technology
Keio University

Hiyoshi 3-14-1, Kohoku-ku
Yokohama, 223 Japan

Department of Mathematics
Faculty of Science and Technology
Keio University

Hiyoshi 3-14-1, Kohoku-ku, Yokohama, 223 Japan

KSTS/RR-90/001
February 7, 1990

Data and Description Rule

Ritei SHIBATA, Masaaki SIBUYA and Mutsumi TAKAGIWA
Department of Mathematics, Keio University
Hiyoshi, Kohoku-ku, Yokohama, JAPAN 223

Summary

Data and Description (D&D) is a dataset and its description organized together in a consistent and standard form. D&D is a list structure of character strings to be independent of any hardware or software systems. It improves the communication of data among statisticians through computer network. An intelligent statistical software understands the description, selects appropriate procedures and applies them to the dataset. This paper presents the concept of D&D and defines the rules to write D&D with some examples. An example of D&D processing system is also introduced.

This research was partly supported by the ISM Cooperative Research Program (89-ISM.CRP-6).

Contents

Introduction

In this paper we propose the concept of Data and Description (abbreviated as D&D) to organize both statistical dataset and its description together. D&D is a self-explanatory dataset of a formal but informative and versatile expression. It is intended to be a standard for collecting and disseminating data within an open and indefinite group of statisticians through computer networks. It can be, for example, a format to send data to an information system or submit it to electronic publications. More significantly, intelligent and expert statistical software of near future will check the description of D&D, select functions appropriate for its dataset, execute or suggest these functions, and output the results in an annotated and understandable form.

The rules to write D&D, a sort of computer language, are completely described in the Part 2 of this paper. Some examples of D&D taken from Cox and Snell (1981) are given in Part 3. The formal definition of D&D, in the form of yacc/lex (see Schreiner and Friedman (1985)) is listed in Part 4. An implementation of D&D processing system in S language is reported in Part 5. Before stating the D&D rules, we explain our basic ideas in due course of its development.

Part 1 Basic ideas

1. Motivation and background

Data In this paper, by 'dataset' or simply by 'data' we mean a collection of observed values by an experiment or by survey for a special purpose in science, technology or management. It may be a collection of recorded values during a period of industrial, business or economical activities. The collection of values is to be analyzed for confirmatory or exploratory purpose by some statistical procedures. It may be repeatedly used by statisticians for creating new methods of analysis, and by teachers as examples.

There are various types of data reflecting the complexity of the real world, and the multiplicity of people's viewpoints. Because of the variety, statistical science has numerous methodologies, models and algorithms. According to the design of experiments, sampling, observation items, measurement methods,

statistical models in mind, the structure and features of dataset may vary. A statistical dataset which is a mere collection of numbers or symbols is often difficult to use. Only well organized and described dataset in a universal way is useful. Because of the steady development of statistical methodology, this form of the rule may become unable to cover innovative new types of data. The D&D, however, is flexible enough to accept such types of data by a minor modification.

Organization and documentation of data When numbers or symbols are observed, data already have a form on a sheet or in a field note, or as a file of computer system. If a group of persons are involved in the observation there should be an agreement to organize the observed values. The method of organization will be, however, specific for the purpose of observation and is not unique. If a standard is provided, it should be useful for collection and dissemination in a wider group.

Description details are often text data and narrative. The documentation of basic items, like date, title of data, units or scales of measurements is rather easy. The D&D aims at organizing other items as formal as possible, too.

Statistical and scientific databases In many fields of science large databases are constructed as information systems, using sophisticated database management systems. Statistical and scientific databases have some characteristic features and needs, and have been an object of research, see Shoshani (1983) and Rafanelli et al. (1989). Typical examples are databases for national and international economy, seismic records, clinical tests of cancer treatments, and space research by satellites. Users of such an information systems typically retrieve only a part of data relevant to their concern.

A dataset in this paper is not large as a whole database but is of comparable size to the retrieved one. The purpose of this paper is closely related to the efforts to make Metadata or Data Description Dictionary more informative and useful. See Wertz (1986) for Data Description Dictionary. Such efforts are not independent of the Data Base Management System used, and we leave it as a future problem. There are also efforts to build an interface between Database Management System and Statistical Software. Our D&D provides an answer to such an approach. If a Database Management System is able to organize, along with D&D, the retrieved dataset and its adequate description from the Data Description Dictionary, then any specific interface to each Statistical Software would not be necessary.

Statistical software At present, each statistical software requires input data in its proper format. Users first have to reformat data to meet the requirement of the software.

To analyze the data, appropriate functions or subroutines are selected and called by the user. In this stage, a kind of computer programming is necessary. However, it is often the case when possible statistical models and analyses are conceived at the design stage of data collection. If these are documented in a form understandable by machine, the analyses can be automated.

The use of D&D is a sure approach to make statistical software be competent for the work. At least, it can be a universal intermediate expression linking a statistical software with other data handling systems.

Publications of applied statistics Statisticians emphasize the importance of the application of statistical methods to real data for advancing the methodology. Often real data are too bulky to be published, and only small size samples, summary statistics or fictitious data are printed. The data for statistical education are small, since the size of a textbook is limited and the data for exercises are thus far computed by hand.

Now paperless, electronic or on-line publication is practically available and is an ideal form of statistical publication. If an article of applied statistics on a new method is published with a real dataset, including the software used, then readers can comprehend the paper deeper. Conversely, the author will receive more immediate and significant response from readers.

For such a publication of data there must be a standard for organization and description of the data. Actually such a standard is a primary part of the submission rule of data to such a distribution system.

Electronic Journal of Data Analysis A project to create an Electronic Journal of Data Analysis (EJDA) is currently being carried out at the Institute of Statistical Mathematics, Tokyo. EJDA is an academic paperless journal which consists of three sections; Data, Analysis and Software. Contributions to any of these sections are regarded as articles. All articles will be reviewed through the network. Subscription is, of course, takes the same form, too. It aims at publishing such articles more purposively, efficiently and systematically than conventional hardcopy journals. We believe that such a journal will be a continuous source of stimulation to theoretical statisticians as well as a vehicle for accumulating knowledge to develop innovative statistical software. The D&D rule is a

prospective rule for submission of data to the Data Section of EJDA. See Sibuya and Shibata (1987).

2. Policy for designing the D&D rule

System independence The D&D rule is for the communication of data via computer networks. We designed the D&D as a sequence of characters in ISO code (American Standard Code for Information Interchange, ASCII), which is widely accepted and is easy to read and write through various media, tape, diskette or laser disk.

The D&D rule is designed so that both human beings and machines can understand easily. Thus, D&D has a list structure in the LISP sense, but is expressed in that of NAME=VALUE rather than the form of (NAME, VALUE) in LISP. It is still difficult for human being to faithfully follow the format for machines. An interface should be provided for users to organize together these data and description, following the rule. Such a interface will make D&D more friendly if it is oriented to a specific project or to an application field. At least two kind of interfaces are necessary to 'get' D&D into an operating system, a statistical software or other systems, and to 'put' them out of the systems. Our implementations are reported in the following Section 4 and in Part 5.

Structure of data Data in a computer system are structured for easier access by the system. In a database management system, data are structured for retrieval, update, or deletion. Both physical and logical structurings are important. For the present purpose, physical structuring is disregarded. The logical structure of the database reflects user's view, allowed to be different among users. In our case, the viewpoint, namely the data structuring, is that determined by those who collect or create the data.

An elementary structure of raw data, a traditional 'file', consists of 'records'. Each 'field' of a record is a number or a character string of fixed or variable length with an upper limit. A record represents usually an individual, an item or a transaction. A file of collected records can be regarded as 'a relation scheme'. It is a finite subset meaning a mathematical 'relation', namely a subset of 'the product space of the domains' of the columns. A relation scheme is the basic unit of relational data model.

On the other hand, for mathematical modeling, data are structured as vectors, arrays, and as graphs to reflect the intrinsic feature of the data. Array-data

are most convenient, efficient and general for numerical algorithms. Graphs are the most versatile way of structuring, including some specific type of graphs like a tree often appear. Such a graph can be represented by a matrix, too.

Thus considering our objectives and the present statistical methodologies it will be reasonable to limit data structures to relation schemes and arrays only. When both a relation scheme and an array are adequate for the given dataset, an array is recommended. Concerning such a choice see the discussions in Section 6, Uniqueness of description, of Part 2.

Processed data Both confirmatory and exploratory data analysis consist of several steps. Some intermediate summary statistics at these steps are important, and they may be published or archived. Such summary statistics may be organized as a D&D outside of the system for analysis. Then fairly significant part of descriptions of the D&D can be automatically inherited from the original D&D.

It is true that the structure of processed data depends on the system which analyzes the data. But it will be nice if the descriptions of the processed data are automatically generated in a conformal manner with the D&D. We believe that the D&D rule has enough capability to include such summary statistics or intermediate results (Mallows, 1983).

Experience The D&D rule can be justified only by its applicability to real data, and actually the rule has been elaborated by examining all the examples in Cox and Snell (1981) and Andrews and Herzberg (1985). The authors have reorganized all the datasets and their descriptions in these books as D&D's. Some results are listed in Part 4 and the whole list of the D&D's of Examples A to X in Cox and Snell (1981) is available as a technical report, Shibata et al.(1990). Similar experiments on other text-books will be instructive.

3. Defining the rule

Data type and relation scheme In some computer languages, including FORTRAN, data types of variables and their structures are 'declared' prior to use, and their initial values can be given by a 'data statement'. Essentially, the D&D rules are similar to those for variable declarations, but more sophisticated and complex. Let us start from the dataset itself.

'A relation scheme' is a simple rectangle table and each column has its own 'domain' and 'role', namely 'data type'. The columns may have the same domains. See Maier(1983). In the D&D, therefore, a relation scheme is a set of columns, and a column is a named 'vector', that is, a sequence of numbers given a name.

The data type of a vector is described in detail in the D&D. If a vector is of continuous type, then its range and measurement unit are specified. Discrete data type is classified further into logical, categorical, ordered categorical, sequence, count, interval, and others. The data of logical, categorical or other types are expressed by numerical codes 1,2,..., in the data body. Their corresponding character codes are given separately from the numerical codes.

The sequence number of experiments, or allocation indices in designed experiments appear as a patterned sequence of numbers in columns of a relation scheme. Therefore two operators for generating patterns are adopted:

(1) m:n stands for numerical sequence (m, m+1, m+2,..., n) if m < n, and (m, m-1, m-2,..., n) otherwise.

(2) m*n stands for m times repetition of a number n.

In a relation scheme, the data type of columns of covariates has some importance. For example, regression-type data may be distinguished from analysis-of-variance-type data by whether the covariates are continuous or discrete type. Furthermore, a set of columns may be internally related to each other, as discussed in the subsection 'Structure and structure type' below. Thus, such description is also important.

Multiple array and axis type An array is a typical structure and its entry is indexed by a set of natural numbers. Familiar examples are frequency tables and factorial experiment data. As a data-body an array is reformed into a vector, and its structure is indicated by a description. An array can be expressed by a relation scheme with added columns of indices which are codes for logical or categorical variables meaning 'levels'. Therefore, in order to describe an array, it is necessary to introduce the notion of 'axis' and its 'type'. The type of an axis adds further meaning to an array. For example, a pair of axes may mean symmetric measurements on left and right eye sights. An axis stands for a set of levels of a controlled factor, while another axis stands for repetition to decrease measurement error.

There is also a very special axis. Take, for examples, R.A.Fisher's iris data. There are 50 by 3 by 4 measurements; there are 50 flowers of three species, and petal length, petal width, sepal length and sepal width of each flower are measured. The first axis for flowers is repetition, the second axis for species is category, and the last axis means 'what to be measured'. In fact, this last axis means 'a multivariate', or a random vector. Therefore, an axis type 'variate' is introduced.

Structure and structure type Relation schemes and arrays, fundamental structures in the D&D, are defined as a 'structure type', which specifies columns to form a relation scheme or axes to form a relation. Structure type further describes details in relation schemes and relations in arrays. For example, in a relation scheme, two columns 'x' and 'y' mean coordinates of the points where the third column 'intensities' are measured. If the coordinates (x_i , y_j) are those of a regular rectangular lattice, the intensities are expressed as a matrix with x and y axes. Thus the pair 'x' and 'y' of columns (or of *dnames*) of a relation scheme, or of axes (or of *anames*) of an array have a special structure meaning 'Cartesian coordinates'. Similarly, two columns or axes, 'year' and 'month', means more specific 'month of year'. Through experiences the necessity of describing nonstandard structure can be anticipated, and a way to introduce user defined structure type is provided.

Thus, structure type (abbreviated as 'Stype') is for describing a structure among multiple columns or axes, while data type ('Dtype') and axis type ('Atype') describe the features of each column, value of an array, or its axis.

Designed experiments and surveys If a dataset is obtained by a designed experiment or sampling survey, its structure is balanced and patterned, and well established procedures can be applied to the dataset.

Design of an experiment is specified by its commonly used name with necessary parameters. The important description of 'factor type' is discussed in the following section.

A sampling survey is specified by its sampling procedures, which is usually nonsequential. That is, the procedures are fixed before observations and independent of the observed values. Random sampling is with or without replacement, and sampling ratio or sample size is specified. Systematic sampling starts from a unit and continues at a rate, or continues to reach a given size, following a pattern.

In designed experiments, 'treatments' are assigned to randomly sampled 'individuals' so that results within a laboratory can be used for a wide range of applications. Therefore sampling procedures are described as 'Observation.mechanism' which is common in experiments and surveys. See Smith and Sugden (1988) for the role of mechanisms for treatment assignment and unit selection in experiments and surveys.

Factors in experiments 'A factor', or 'a main effect' in contrast with 'an interaction', is a notion in the linear additive model of factorial experiments. A factor is a covariate which is considered influential to an response variate and selected for the study by an experiment. When a covariate is quantitative, it is usually fixed to one of finite number of values, called 'levels', which are of discrete data type. If the data of experiment is organized as an array, a factor corresponds to an axis and its levels to index values. It has been extended by McCullagh and Nelder (1989) to generalized linear models. It may be useful in broader situations where several covariates are involved. In 'factor analysis' a factor is a latent or hypothetical variable which cannot be measured, and it is quite different from that discussed here.

In order to design an experiment and to analyze the resulting data (typically by analysis of variance), many efforts have been made among quality control engineers to classify the nature of factors. The following classification is generally accepted, see for example Okuno and Haga (1969), although some variations have been proposed by others.

A principle in the classification is to realize the gap between the results of analysis within a laboratory and the applicability of these results to a real life situation. Typically, conditions which can be controlled to specific values in the laboratory can not be always controllable at application sites.

'Controllable': Factors of this type can be specified in any level at the experiment and in practice. Level sets of these factors are often called 'treatments', and the choice of the optimal level set of controllable factors is the main purpose of the experiment.

'Specification / Stratification': Conditions which are influential to the object variate but cannot be specified in practice must be included in the experiment. At the application sites, this factor is a given condition, and the condition varies from site to site. At the analysis, the interaction between these factors and controllable factors is an important question. For example, the optimal level set of

controllable factors may depend on the levels of this factor.

'Block': To provide homogeneous conditions to compare controllable factors, measurements must be blocked to prevent unexpected influences. The effect of blocks is considered but their interaction with other factors is negligible: the experiment should be designed to guarantee this.

'Explanatory': Factors of this type are quantitative and can be measured both in the laboratory and at application sites, but they are not controllable. If it is controllable in the laboratory it is like a Specific Factor, and some authors regard these as the same. At the application site those factors can be controlled, possibly at higher costs, if the influence is critical.

'Variable': Factors of this type are not controllable at the laboratory nor at application sites, and their effects are considered random variables. Moreover, levels are selected randomly, so that Random or Mixed Models of Analysis of Variances are applied.

'Repetition': This means repetition under the same conditions to improve the accuracy by averaging and also that of the error variance.

Among the above factors, Explanatory and Variable factors are of continuous data type, and the other factors are of discrete type. There are gray areas between factors of this classification. They should be classified according to reality and the purpose of the experiment.

Possible analysis If the designer of an experiment explicitly describe his or her intention in D&D, then statistical software will accept it as was intended, and process it. It is more straightforward than specifying the details of the experiment or survey. It is easier for software to process the dataset, too. At present this kind of specification is limited to a few examples and under development.

Miscellaneous A simple document of a dataset may contain the date, title, name of the collector, and key words. These items are short phrases or sentences. They are to be used for listing, indexing and 'help' or other human interfaces. Standardization of these items is like that of a library cataloguing, or formatting a journal.

Writing D&D, one feels the necessity of putting annotation everywhere. Just for simplicity, scattered comments should be avoided and 'Explanation' is the only place to write anything which cannot be written in the other places.

An outline of the D&D rule Now concluding this section we outline the D&D rule to explain how the issues of this section are implemented as a formal rule. Roughly, the rule is divided into four parts: (1) Preface (2) Overview (3) Data Description, and (4) Data body. They are in almost reverse order to that of paragraphs of this section.

The first Preface Part is 'Miscellaneous' description in text, and the second Overview Part is Design, Observation.mechanism and Possible.analysis. The Overview Part relating to the whole dataset contains the descriptions for higher-level softwares which are outside of the D&D. The third and fourth parts are more essential parts of the D&D. The third Data Description Part describes a relation scheme or a relation, or substructures composing them. It describes the elementary parts of dataset: columns of a relation, entries of an array or axes of an array. The last part is the data itself.

4. Implementation

D&D processing system A D&D processing system converts a D&D file into computer entities, and vice versa. The system depends on the computer environment, and primarily on the operating system and the computer language. In principle, the D&D can be transformed into some structured dataset in any environment. However, it is essentially a list structure in the LISP sense, and a computer language dealing with list structures is more convenient to implement a D&D processing system but not limited to such a language.

To check the feasibility and the usefulness of the D&D rule, a D&D processing system has been implemented. It reads a D&D and converts it into S objects (see Becker et al.(1988) for S language). The D&D is converted faithfully into a list in the S sense which has a special attribute 'd.d'. Then D&D forms a special object class.

S object class 'd.d' Once an S object class with the special attribute d.d is defined, many things can be done by writing S functions with the main argument as a d.d object. First, there are functions for printing or displaying each component of D&D. Another function picks up 'Possible.analysis' and if it is understandable one, for example 'regression analysis', then it asks the user to apply one of lsfit, l1fit or rreg. Other functions assemble S datasets, prompt the user to ameliorate insufficient descriptions, and put them together as a D&D file. The details are explained in Part 5 of this paper.

Acknowledgements

In creating the D&D rule, we owe our largest debt to our colleagues in the Cooperative Research Program (1-ISM.CRP-6) of the Institute of Statistical Mathematics, Tokyo. A partial list of individuals to whom we owe thanks includes H. Tamura, J. Kariya, and T. Tango.

Version 2.3

Part 2 D&D Rule

1. D&D

1.1. D&D file

A D&D file is a set of one or several D&D's with protocol on the top or between D&D's, which is explained in the next section.

```
# ...
  :
# ...
name = D&D
  :
name = D&D
```

1.2. Protocol

The protocol is composed of several lines beginning with #. These lines are information on the environment for D&D processing systems. If lines of the same kind appear more than once, then the latter line overrides the former one. The currently defined lines are shown in the following by examples.

```
# max_line_len  80
Maximum number of characters in a line.

# max_str_len   1023
Maximum number of characters in a string.

# str_cont      "\"
Code for continuation of a string.

# max_name_len      52
```

Maximum length of a string used as a name tag.

version 2.3
Identification of D&D generating system.

kanji sjis
Shift JIS kanji code.
kanji jis
JIS kanji code.
kanji euc
Extended UNIX kanji code.

include keio.local.h
File to be included
external *external.definition*

By "include" statement, an external file can be included into the D&D file. The "external" statement indicates an external dataset which is inserted as an element into Data.body of a D&D in the following line. For Data.body, see Section 5. The *external.definition* should have the form of NAME=STRING, in which, NAME is the same as that defined in Data (Section 4.3) and STRING gives the name of a file to be read. The content of the file should be a simple vector of NUMBERS. For the definition of NAME, STRING and NUMBERS, see the next section.

1.3. How to read the rule (Meta grammer)

In the following sections the D&D rule will be described in a formal way. We explain how to read the rule in this section. The notation *A*: B reads "nonterminal symbol *A* is defined by B". Vertical bars | in B, the right hand side of the definition, are separators meaning an exclusive choice of one of items listed. For example, *A* : a | b | c means "*A* is defined by one of a, b or c".

The basic item of D&D syntax is NAME=VALUE. Giving a null valuem NAME= , is allowed if VALUE is unknown, or the VALUE is provided by an *external.definition* of a # external line in protocol. The NAME can contain letters, digits, and periods, and must start with a letter. The VALUE is a terminal or nonterminal symbol, a list of either of them, or a parenthesized list of

items. Terminal symbols are defined soon later. A nonterminal symbol is written as *xx...xx* in italic. A parenthesized list of items has the form (a b c ...) separated by space or spaces. Not all of the items are obligatory, but at least one item should be in the list.

Ordering of elements in a list is insignificant if they are named, but can be significant if they are unnamed. Indention, spaces, new lines, or tabs can be used freely to improve the readability. This is only for human being and not meaningful for D&D systems.

The NAME is classified into three categories. The name Xx...x beginning with an upper case, .Data and .Dim in Data.body are keywords for initiating an appropriate action of a D&D system which processes D&D, that is, a sort of function name for the processing system. All-upper-case-name XX...X is also a function name, which is not recognized by the current D&D system. However, the description by such a name is expected to be helpful for later analysis of D&D or for any other systems of data analysis outside D&D. All-lower-case name xx...xx is a user defined name, which appears in the items, Data.structure, Axis or Data.body, or a local key name of parameters of the item with all-upper-case-name. Names defined in the item Data.body always appear also in the item Data for the description of its attributes. User defined names will be denoted by different symbols in the rule; *sname*, *aname* or *dname*, according to places where they appear; Data.structure, Axis, or Data.body, respectively.

Terminal symbols are

NUMBERS:	(NUMBER ...)
STRINGS:	(STRING ...)
NUMBER:	numeric
STRING:	string quoted by a double quote "

NUMBER is a decimal number, which is a single precision number of the form nnn.mmm, nnn.mmmekk or nnn.mmmEkk, or a double precision number of the form nnn.mmmdkk or nnn.mmmDkk. Logicals are only NUMBERs 0 or 1 and distinguished from numeric, as they are given Dtype "logical" (Section 4.3.2) or Atype "logical" (Section 4.2.2). The values like "True" or "False" are given through the item Code in Data (see Section 4.3.4).
Missing values are denoted by NA (not available). Detailed type of missing can be described through Stype INVALID (Section 4.1). Two operators, colon : and asterisk * are allowed in place of NUMBERS. The expression n : m is equivalent

to a sequence from n to m with the step +1 or -1. The expression n * m is equivalent to a sequence of n times repeated m. Each n and m should be a single NUMBER for each operator. Repeated application of the operators : and * , with or without parentheses, are not allowed.

A STRING referring to a name of dataset or of axis can appear in place of nonterminal symbols "*dname*" or "*aname*" in the rule. Such a name is defined in Data.body or in Axis, which should be unique across Data.body and Axis. To refer to *dname* or *aname* within a sentence of STRING, the name should be quoted by < and > like <*dname*> or <*aname*> to avoid quotes within the quote. The name *sname* given to a structure among several *dname*s and *aname*s is used to keep the notation of item NAME=VALUE consistent and not referred to, from the other parts of D&D. The following convention is also used. The *qname* means "a quoted *dname* or *aname*".

qnames: (*qname* ...)
anames: ("*aname*" ...)
dnames: ("*dname*" ...)
qname: "*dname*" | "*aname*"

1.4. Top structure of D&D

Items constituting D&D at the top level are as follows.

D&D : (
Title = STRING †
Date = *date* †
Contributor = *contributor* †
Research.field = STRING | STRINGS †
Keyword = STRING | STRINGS
Purpose = STRING | STRINGS
Source = STRING | STRINGS
Explanation = STRING | STRINGS
Primary.model = *primary.model*

Design = *design*
Observation.mechanism = *observation.mechanism*

Possible.analysis = *analyses*

Data.structure = (*sname* = *struct* ...) †
Axis = (*aname* = *axis* ...)
Data = (*dname* = *attr* ...) †

Data.body = (*dname* = *value* ...) †
)

Items with a dagger † at the end of the line are obligatory. The first items from Title to Primary.model at the top level are "Preface Part" of D&D, and the items from Design to Possible.analysis are "Overview Part". The items from Data.structure to Data are "Data Description Part". The last Data.body is a Part containig a set of data themselves. The four parts are explained in subsequent Sections 2–5.

2. Preface part

2.1. Title

This item is to give a title to the D&D, and used for later listing and indexing.

2.2. Date

date : (collected = STRING created = STRING modified = STRINGS)

This is to record the dates as a STRING of the format "yyyy-mm-dd" (ISO 8601). The 'collected' is the date of collection of the original data, 'created' is the date of creation of the D&D, and 'modified' is the date of modifications.

Example

Date=(created= "1965-08-31")
This D&D is created on the 31st of August, 1965.

2.3. Contributor

contributor: (investigator = STRINGS assembler = STRINGS)

The 'investigator' is the name of collector(s) of the data, and the item 'assembler' is the name of creator(s) of this D&D. Such names may accompany with their

address or other ways of access.

2.4. Research.field

Name of disciplines, application fields, business fields, which this D&D is related to.

2.5. Keyword

Keywords and phrases which do not appear in any STRING in Title or Research.field, or any NAMEs in the following Primary.model or Possible.analysis. Words which appear in any Long.name are also excepted.

2.6. Purpose

The purpose of this data collection written in free format.

2.7. Source

If this D&D is obtained from a published source, the source should be referred to in format free.

2.8. Explanation

Format free description of this D&D. Any information, difficult to describe in other formal items, may be described here. Reference papers, books or materials are also included here.

2.9. Primary.model

Framework of model building or analysis of this D&D, possibly conceived before starting the collection of the data. The following Possible.analysis is within this framework. This is described in format free. Some examples are:

primary.model : *model* | (*model* ...)
model : "accelerated test"
| "classification"
| "contingency"
| "dose response"
| "factorial experiment"
| "factor analysis"
| "functional relationship"

| "point process"
| "sampling survey"
| "survival analysis"

Examples

See Part 3 for examples of Preface Part of some D&D's.

3. Overview part

3.1. Design

If the data are collected by a designed experiment or sample survey, the design should be recorded through this item. Randomization mechanism is separately described in Observation.mechanism.

design : CR = (factors=*qnames* ftypes=NUMBERS response=*dnames*)
| RB = (factors=*qnames* ftypes=NUMBERS response=*dnames*)
| BIB = (v=NUMBER k=NUMBER b=NUMBER r=NUMBER
factors=*qnames* ftypes=NUMBERS
response=*dnames*)
| OA = (n=NUMBER m=NUMBER s=NUMBER d=NUMBER
factors=*qnames* assign="*dname*"
ftypes=NUMBERS response=*dnames*)
| SPLIT = (factors=*qnames* ftypes=NUMBERS
response=*dnames*)
| NESTED = (factors=*qnames* ftypes=NUMBERS
response=*dnames*)

3.1.1. Explanation of Design items

In any case, several *dname* can be given to the *response*.

CR Completely randomized block.

RB Randomized block experiment.

BIB Balanced incomplete block design with parameters, v = the number of treatments, k = the size of blocks, b = the number of block, and r = the number of repetitions. The number of concurrences $\lambda = r(k-1)/(v-1)$, which follows from the above parameters, is omitted.

OA Orthogonal array design with parameters, n = the number of rows in the array, m = the number of columns in the array, s = the number of levels, and d = the strength. Each column vector of the matrix "*dname*" given to 'assign' corresponds to each main effect in *qnames*, and is the vector of allocated level numbers of the factor.

SPLIT

Split plot. A model for the 'response' y_{ijk} is

$$y_{ijk}=\mu_{ij}+e_{ik}+e_{ijk}\ ,$$

where errors e_{ijk} , j=1,2,... are homogeneous, given i and k. The indices i and j stand for controllable factors and k stands for repetition factor under the condition that the (i,j) th 'factors' are given. Such a randomization mechanism is separately described in Observation.mechanism.

NESTED

Nested plot. A model for the 'response' y_{ijk} is

$$y_{ijk}=\alpha_i+e_{ij}+e_{ijk}\ ,$$

where errors e_{ijk}, k=1,2,... are homogeneous, given i and j. The indices i and j stand for controllable factors and k stands for repetition factor in the 'factors'. Such a randomization mechanism is separately described in Observation.mechanism.

3.1.2. ftypes

The ftypes is a vector of numbers giving the type of each main factor. The type number is one of the following numbers.

1 Controllable factor. Completely controllable factor whose levels can be set as designed. Main concern of factorial experiment is to know the effect of this factor to the 'response'.

2 Specification factor. This factor is controllable at the experiment but fixed and given in practice. An example is "raw material of a product". Stratification factor can be included in this Specification factor.

3 Block factor. This factor is usually introduced in the design to eliminate systematic error.

4 Explanatory factor. This factor is uncontrollable, unavoidable, influential, but not the direct aim of analysis implied by the experiment. It is

typically a measurement of environment for the experiment.

5 Variable factor. Variable term in random models or mixed models.

6 Repetition factor. This corresponds to an error term of the linear model, which is usually not regarded as a factor but included as a type to represent the axis of repetition.

Example H (Cox and Snell)

```
Design = ( RB = ( factors = ( "process" "batches" "purity" )
          ftypes = ( 1 3 4 )  response = "fault" ) )
```

Example J (Cox and Snell)

```
Design = ( CR = ( factors = ( "length" "amplitude" "load" )
          ftypes = (1 1 1)  response = "cycles" ) )
```

Example M (Cox and Snell)

```
Design=( BIB=(v=8 k=4 b=4 r=2
          factors=("factor1" "factor2" "block" "grade")
          ftypes=( 2 2 3 4 ) response="count") )
```

Example Q (Cox and Snell)

```
Design = ( NESTED = ( factors = ( "yarn" "bobbin" "samples" )
              ftypes = ( 1 5  6)   response = "strength" ) )
```

Example R (Cox and Snell)

```
Design = ( SPLIT = ( factors = ( "day" "sex" "treatment.a" "treatment.b" )
             ftypes = ( 3  2 1 1 )  response = "amount" ) )
```

3.2. Observation.mechanism

Sampling scheme or observation mechanism. Both random allocation mechanism in design of experiments and random sampling scheme can be described as follows.

observation.mechanism: *obs* | (*obs* ...)
obs : RND = (which = *qname* given = *qnames*
from = *qname* rate = "*dname*"

size = NUMBER | STRING replace = NUMBER)
| SYS = (which = *qname* given = *qnames*
from = *qname* rate = "*dname*"
size = NUMBER | STRING start = NUMBERS
pattern = *patterns* | STRINGS)
| RNDCENSOR = (which = *qname* censor = *dname*)
| CENSOR = (which = *qname* censor.time = "*dname*"
censor.number = NUMBER)
| POPULATION = (which = "*dname*" frame = STRINGS
from = "*dname*" size = NUMBER | STRING)

patterns: (*pattern1* = NUMBERS *pattern2* = NUMBERS ...)

3.2.1. Explanation of Observation.mechanism items

RND

Randomization mechanism. Randomization mechanism employed both in design of experiments and in sample survey is described. If the item 'from' is omitted, the elements of *qname* given to the item 'which' are regarded as random observations. If the item 'given' exists, the randomness is conditional. If *dnames* is given to 'given', the condition specified by each elements of the *dname* is parallel to the elements of *qname* given to 'which'. If *anames* is given to 'given', the condition specify entities of the array, that is, columns or rows of the array. For the 'rate', vector of sampling probabilities is given, corresponding to each *qname* given to the item 'given'. The item 'replace' is a flag indicating sampling with replacement or without replacement by 1 and 0, respectively. Default is 0, that is, without replacement. If the 'from' is given, 'which' is sampled 'from' with 'rate' under the condition 'given'. See the item POPULATION how to describe the size of the population itself.

SYS Fixed items, individuals or units, or systematic sampling mechanism. The term 'which' is such a fixed item or the result of systematic sampling from the term 'from' under the condition 'given'. The term 'rate' is the sampling rate, and the term 'start' is the start point of a systematic sampling. The term 'pattern' is a list of NUMBERS giving a pattern of indices of the elements of *qname* given to 'which', or the

name of systematic sampling like "zigzag". In case of design of experiments, the term 'which' corresponds to constant term like α_i or β_j.

RNDCENSOR

Random censoring. The *dname* given to the term 'censor' has Dtype "logical" and indicates if the corresponding element of *qname* given to 'which', that is usually time, is censored or uncensored. The 1 indicates censored and the 0 uncensored.

CENSOR

If 'censor.time' is given, the *qname* given to 'which' is censored with type 1, that is, by a limit of 'censor.time' span of observations. If 'censor.number' is given, the censoring is by 'censoring.number' of observations.

POPULATION

Sample from finite population. In case of vague sampling design, format free description can be given to 'frame' by STRINGS. Otherwise finite population is given to 'from' pararell to the data *dname* given to 'which'. The term 'size' can be STRING when the design is vague.

Example (A household survey)

```
Observation.mechanism=(
    RND=(which="city"  given="city.strata" size=1 )
    RND=(which="block"  given="city"  size="unfixed" )
    SYS=(which="unit" given=("start.year" "block" "city")
            size="unfixed but maximum 12" )
    SYS=( which="start.month"   given=("unit" "start.year" "block" "city" )
        size=12
        pattern=( a=(1 1 1 1 1 1 7 7 7 7 7 7) b=(2 2 2 2 2 2 8 8 8 8 8 8 )
        c=(3 3 3 3 3 3 9 9 9 9 9 9) d=(4 4 4 4 4 4 10 10 10 10 10 10 )
        e=(5 5 5 5 5 5 11 11 11 11 11 11 ) f=(6 6 6 6 6 6 12 12 12 12 12 12) )
            )
    RND=(which="household"
        given=("classification" "start.month" "unit" "start.year"
            "block" "city") size=6 )  )
```

This is a simplified version of the actual household survey conducted by Census Bureau of Japan as follows. A city is sampled from a strata of cities, several blocks are sampled from a city, 12 districts are systematically selected, the

starting month of recording is systematically chosen by one of 'patterns', and finally a household is randomly chosen according to the classification index, whether it is employed or unemployed.

Example Q (Cox and Snell)

```
Observation.mechanism=(
        SYS=(which="yarn")
        RND=(which="bobbin"  given="yarn"  size=6 )
        RND=(which="samples"  given=("bobbin" "yarn")  size=4  )
                        )
```

This example corresponds to that in Section 3.1. Six bobbins are randomly chosen from each of two types of worsted yarns and four strings of short length were chosen at random from each of six bobbins. In this case, yarn can be considered as a controllable factor. The design is NESTED plot as described in the Design item. The linear model for this experiment is written as

$$s_{ijk} = \alpha_i + e_{ij} + e_{ijk} ,$$

where s_{ijk} is 'strength' of yarn, and i, j and k stand for yarn, bobbin and repetition, respectively.

Example R (Cox and Snell)

```
Observation.mechanism=( RND=(which="day" given="sex"  size=4) )
```

This example corresponds to that in Section 3.1. It describes randomization mechanism of a SPLIT plot design.

3.3. Possible.analysis

This item describes possible analyses conceived at the stage of data collection in a formal way. Currently defined analyses are:

analyses : *analysis* | (*analysis* ...)

analysis : ANOVA = (explanatory = *qnames* response = "*dname*")
| LOG.LIN = (explanatory = *qnames* response = "*dname*")
| LOGIS.LIN = (explanatory = *qnames* response = *binary.response*)
| REG = (explanatory = *qnames* response = *dnames*)

binary.response : "*dname*"
| "*aname*"
| (yes = "*dname*" total = "*dname*")
| (no = "*dname*" yes = "*dname*")

3.3.1. Explanation of Possible.analysis items

ANOVA
Analysis of variance. The 'explanatory' is a list of main factors and the 'response' is the response variable. Possible interaction of factors can be described in Explanation.

LOG.LIN
Loglinear model. The 'explanatory' is a list of explanatory variables. The data given to 'response' has Dtype "count".

LOGIS.LIN
Logistic linear model. The 'explanatory' is a list of explanatory variables. The 'response' should take one of the forms in *binary.response*. In case of "*dname*", it should have Dtype "logical". If an "*aname*" is given, the corresponding Value of the array should have Dtype "count". In the last two forms in *binary.response*, *dname*s given to 'yes', 'no' and 'total' are "count" of negative, positive and total frequency, respectively

REG Regression. The 'explanatory' is a list of explanatory variables and 'response' is a list of response variables.

Example H (Cox and Snell)

```
Possible.analysis=( LOGIS.LIN=( explanatory=("purity" "process")
                    response ="fault" ) )
```

Example J (Cox and Snell)

```
Possible.analysis=( REG=( explanatory=("length" "amplitude" "load")
                    response="cycles") )
```

Example O (Cox and Snell)

Possible.analysis=(ANOVA=(explanatory=("silver" "iodine") response="weight"))

There is no Design item for this example. The aim of data collection is analysis of variance but the experiment is not designed to be balanced.

4. Data description part

4.1. Data.structure

A structure is defined by combining any number of *anames* or *dnames*. Any number of structures can be defined (see Example C below). All *anames* and *dnames* should be organized through this item.

struct: *str* | (*str* ...)
str : (Long.name = STRING Columns = *dnames*
Stype = *stypes*)
| (Long.name = STRING Axes = *anames*
Value = "*dname*" Stype = *stypes*)

The first form defines a "relation scheme", and the second form defines an "array". Just to define a relation scheme or an array the item Stype is not necessary. Relationship among several elements in Data can be described by giving the *dnames* to 'Columns'. The *dname* of an array data cannot appear in 'Columns' but only in a 'Value'. The ordering of *anames* given to Axes is significant and should be consistent with that of the values of .Dim element of the *dname* given to Value, in Data.body. One column matrix is an exception. In this case, only *aname* of an axis can be given to Axes and a simple vector without .Dim can be given to the Value.

Structure type, Stype, is in principle the attribute for a structured object. There are two types of the attributes: one is namely a "substructure", and another is an attribute for a data but some other data are referred to for the description. This point is different from Atype and Dtype even though the same word "type" is used. Attributes for individual dataset can be given by Atype or Dtype. Stype is only for individual *str* but not for the whole *struct*.

The *sname* (Section 1.4) given to *struct* is not referred to, in the other part of the D&D, but is given to follow the notation of item, NAME=VALUE. It is recommended to use a short *sname*, since this name may be frequently used to

access each dataset in the D&D system. It can be equal to some of *aname* or *dname*.

stypes: *stype* | (*stype* ...)
stype : SEQ = (which = *qnames* given = *qnames*)
| COORD = (x = *qname* y = *qname* z = *qname*)
| (r = *qname* theta = *qname*)
| (r = *qname* theta = *qname* phi = *qname*)
| (longitude = *qname* latitude = *qname* system = STRING)
| RADIX = (year = *qname* month = *qname*
day = *qname* hour =*qname*
minute = *qname*
second = *qname*)
| (degree = *qname* second = *qname*
system = STRING)
| ASSOCIATION = (which = (*aname aname*)
from = *aname* to =*aname*
what= STRINGS)
| PERM = (which = "*dname*" given = *qnames*)
| TIME.SERIES = (time = *qnames* value = *qnames*)
| SUM = (which = *qnames* given = *qnames*
total = NUMBER | total="*dname*"
| out.of = NUMBER | out.of ="*dname*"
)
| INTERVAL = (which = *qname*
min = "*dname*" max = "*dname*")
| SUBSET = (from = *anames*)
| GRAPH = (vertex = "*aname*" adjacent = "*dname*")
| INVALID = (which = "*dname*" where = NUMBERS | "*dname*"
what = NUMBERS | "*dname*" code = STRINGS)
| PRECISION = (which = *dnames* precision = NUMBER |
"*dname*")
| TRUNCATION = (which = "*dname*" where = NUMBERS | "*dname*"
left = NUMBERS | "*dname*"
right = NUMBERS | "*dname*")
| INDEX = (which = "*dname*" of = "*dname*")
| *user.defined.stype* = (which = *qname*

where = NUMBERS | "*dname*" *anything*)

4.1.1. Explanation of Stype items

Stypes above are only currently defined ones. Various other Stypes will be added along with the development of the D&D system.

SEQ Sequence. The ordering of the elements in the *qnames* of the item 'which' is meaningful. The ordering is under the condition that each element of 'given' is specified. This Stype may be needed for a relation scheme. In case of array, it can be indicated by setting the corresponding Atype as "sequence". See Example B below.

COORD

Coordinate system. A triplet 'x', 'y' and 'z' means orthogonal coordinates, a pair 'r' and 'theta' polar coordinates, and a triplet 'r', 'theta' and 'phi' spherical coordinates. The 'system' used with 'longitude' and 'latitude' describes "north latitude", "east longitude" and so on. An example is the description of the axes of a matrix of observations at rectangular lattice points, or "mesh data".

RADIX

Radix. The tuples of 'year', 'month, 'day', 'minute' and 'second' means the date. The triplet of 'degree', 'second' and 'system' means the angle. The base line and the clockwise or anticlockwise can be described by 'system'.

ASSOCIATION

The *anames* given to 'which' are axes for an association table. Direction of one axis to another can be indicated by a combination of 'from' and 'to'. This is also case of the digraph or a tree with the root. The item 'what' describes the meaning of the direction.

PERM

Permutation. The *dname* of the term 'which' is a permutation under the condition 'given'. Ties and mean ranks are allowed. Axes have Atype = "category" or = "sequence" if structured data is a matrix.

TIME.SERIES

Time series. The terms 'time' and 'value' indicate times and values of a time series. If the term 'time' is given by a radix, like hours, minutes

and seconds, then the *qnames* should form another structure with Stype RADIX. If the time series is observed on equispaced time points, then *qnames* to 'time' can be two element vector, giving the time of start and end of the time series. In this case, such *qnames* can not and should not appear in the Columns. If several *qname*s are given to 'value', the time series is vector valued or multiple time series. Point process is not regarded as a time series.

SUM

Sum constraint. The *qnames* given to 'which' have a constraint that the sum of corresponding elements is equal to that of 'total'. If 'given' exists, the 'total' is a NUMBER when the total is independent of the condition, otherwise elements of *dname* given to 'total' are corresponding to different conditions generated from *qnames* in 'given'. Composition ratio can be described by putting 'total' simply 1 or 100. The constraint reflects an aspect of the data generating mechanism. Actual data are not necessarily satisfy this condition because of accuracy or missing value problems (see Example M below). If 'out.of' is used instead of 'total', then 'which' is the number of positive responses whithin the total number of responses, 'out.of'.

INTERVAL

Interval data. An observation is only an interval, usually a time interval, on a real line including an event point. The terms 'min' and 'max' indicate endpoints of the interval. An example is a pair of days indicating a period in which a baby got a tooth. This is because exact observation is possible when the baby visits the clinic.

SUBSET

Subset data. A typical example of this kind of data is multiple answers in an opinion survey. Multiple choice is taken from the combination of *anames* given to 'from'. Then Atype of such *anames* is "category", and Value of the array has Dtype "logical" or "count".

GRAPH

Tree. The 'vertex' describes vertices or nodes of a graph or tree and the 'adjacent' is an adjacent matrix. The *aname* given to 'vertex' should appear in the Axes=("*aname*" "*aname*"). A digraph can be described by using Stype ASSOCIATION together. Rooted tree is described as a kind of digraph.

INVALID

Detailed description of invalid data. Description is element by element. The value of corresponding element of 'which' is NA (not available) or NUMBER. The 'where' is a vector of indices specifying elements of 'which' to be described as invalid, and 'what' is a vector of the same length specifying the invalid type by an index to the STRING given to the 'code'. If there is a lot of invalid values, 'where' and 'what' can be placed in Data.body, then their *dname*s are given to 'where' and 'what'. If there is only one type of invalid data, 'what' can be omitted. Examples of STRINGs to 'code' are:

"missing"	Missing data.
"refused to answer"	Refused to answer.
"never happen"	Never happens this categorical value.
"invalid answer"	Invalid answer.
"sic"	Apparently wrong data. An example is negative weight.
"inaccurate"	Inaccurate data because of error in experiments or by other reasons.

PRECISION

Accuracy or the least absolute value of possible observation. Quantized data can be indicated by using this item.

TRUNCATION

Truncation point. Truncation occurs typically because of limitation of instrument or time, or the method of measurement. The 'where' gives indices of 'which' to specify truncated elements of the data. The 'left' and 'right' give left (lower) and right (upper) limits, respectively.

INDEX

The elements of 'which' are indices of 'of'. This Stype can be used only for relation scheme.

user.defined.stype

Any user defined Stype can be given. The item 'which' specifies a dataset or axis to be described, which appeared in the 'Columns', 'Axes' or 'Value'. The item 'where' indicates the indices and 'what' is element by element description corresponding to 'where'. Any STRING, NUMBERS, *qnames*, or a list can be given to *anything*.

Example A (Cox and Snell)

```
Data.structure=(
    a.time=(Long.name="Arrival time of patients at intensive care units"
            Columns=("year" "day.of.week" "day" "month" "hour" "minute")
            Stype=( RADIX=( year="year" month="month" day="day"
                hour="hour" minute="minute" ) )
        )
```

Example B (Cox and Snell)

```
Data.structure=(
    interval=(Long.name="Mean intervals"
            Columns=("f.size" "births" "MM" "MF" "FM" "FF" )
            Stype = ( SEQ = ( which = "births" given="f.size" ) ) )
        )
```

Example C (Cox and Snell)

```
Data.structure=(
    freq=(Long.name="Frequency of occurrences"
            Axes=("works" "kai") Value="sentences" )
    total.number=( Long.name="Total number of kai's"
            Axes="works" Value="total.kai" )
        )
```

Example M (Cox and Snell)

```
Data.structure=(
    cauliflowers=( Long.name="Cauliflowers"
            Columns=("count" "block" "factor1" "factor2" "grade")
            Stype=(SUM=(which="count" given=("factor1" "factor2" "block")
                total=48) )
            )
        )
```

4.2. Axis

Description of each axis of a matrix or multiple array.

axis : (Long.name = STRING Atype = *atype*
Unit = STRING Levels = NUMBERS | STRINGS)

4.2.1. Long.name

Long.name of the axis, which may be used for pretty and informative printing, display or graphics of the matrix or array.

4.2.2. Atype

Type of axis of a matrix or multiple array. By nature of axis, Atype classifies indices into several classes. Atype is common with Dtype of discrete variables except the first two types, "variates" and "rank" which are axis proper types.

atype: "variates"
The axis corresponds to components of a vector of several variates, and it is typical for multivariate analysis data. For example, the first component is length and the second component is breadth of a petal of a flower.

| "rank"
Rank of some objects. Such data is obtained by an interview asking individuals to give rank orders to several commodities according to his/her preference. Then the array entries are the codes of object and the Dtype is "category". Another way of organizing such a preference data is to take an axis of "category" type representing objects and the array entries are ordinal numbers showing the order of preference. In both cases, the Stype PERM should be described in the Data.structure by giving the value of matrix to 'which' and an axis to 'given'.

| "logical"
Logical value, false or true. In this case, Levels is a vector of two string components, explaining logical values, false and true in this order. This type is a special case of "category", but it is recommended to use this type when only two choices exists, like male and female or left and right.

| "category"
Category or nominal scale. Code for each category can be given to Levels. This is distinguished from "ordered category" below. Code

for each category is specified as "category".

| "ordered category"
Totally ordered "category", which is distinguished from "equispaced", "specified values" and "interval class" below.

| "count"
The axis represents counts of an event. For example, when days of a year is classified by the number of traffic accidents in each day, the axis is the number of accidents and the values of the vector is the number of days. Levels is usually NUMBERS but STRINGS can be used to allow description like "10+" which means count greater than 10.

| "id"
Identification ndex or number. This is distinguished from the following "sequence".

| "sequence"
This is a kind of "id", but used when the order of component of axis may have some significance. Axis of repetition usually has this Atype "sequence" or "id".

| "equispaced"
Equally spaced discrete values, at which a continuous explanatory variable of Controllable Factor was set at the experiment. This is not "interval class". This suggests the use of Chebyshev-Fisher orthogonal polynomials for the regression. Levels should be NUMBERS. Unit is obligatory in this case.

| "specified values"
Similar to "equispaced" but not equally spaced. An example is, records of each runner in 800m, 1500m and 5000m running race.

| "interval class"
Continuous observations are discretized by classification into several intervals. The corresponding value of the vector is the count of observations falling into each interval and has Dtype = "count". Levels should be STRINGS. This is used in conjunction with CUT

in Transform. Note the difference from Stype INTERVAL, for a situation where only a time-interval including an instant of an event is observed and the interval varies observation to observation.

4.2.3. Unit

For convenience of later use. This may be used in conjunction with Levels if NUMBERS are given to Levels. This is meaningful only if Atype is one of the last three types, "equispaced", "specified values" or "interval class".

4.2.4. Levels

This indicates levels of each axis. Normally Levels are specified by STRINGS, which may be divided into NUMBERS to Levels and STRING to Unit. If this item is omitted, the D&D system will fill it by a simple sequence starting from 1.

Examples for Axis

Example C (Cox and Snell)

```
Axis=(
    works=(Long.name="Pauline works" Atype="category"
        Levels=("Romans(1-15)" "1st Corinth." "2nd Corinth." "Galat."
            "Philip." "Colos." "1st Thessal." "1st Timothy"
            "2nd Timothy" "Hebrews")
        )
    kai=(Long.name="Frequency of kai's in a sentence"
        Atype="count"
        Levels=("0" "1" "2" "3 or more")
        )
    )
```

Example J (Cox and Snell)

```
Axis=(
    length=(Long.name="Length of test specimen"
            Atype="equispaced"
            Levels=(250 300 350)
```

```
                Unit="1 mm"
                )
    amplitude=(Long.name="Amplitude of loading cycle"
                Atype="equispaced"
                Levels=(8 9 10)
                Unit="1 mm"
                )
    load=(Long.name="Load"
                Atype="equispaced"
                Levels=(40 45 50)
                Unit="1 g"
                )
    )
```

4.3. Data

Attributes of each *dname* appeared in Data.body are specified by *attr* in Data = (*dname* = *attr* ...).

```
attr  : ( Long.name = STRINGS
          Dtype = dtype
          Unit = STRING
          Code = STRINGS
          Range = range
          Transform = transforms
          )
```

4.3.1. Long.name

Label for the data, which may be used for pretty and informative printing, display or graphics.

4.3.2. Dtype

Fundamental aspect of the data in Data.body. Aspects of data of continuous numbers are not described by this type but by other attributes. For example, positiveness can be described by a combination of the item Range and Stype PRECISION. Therefore, Dtype is often omitted. Typical example is continuous response data of an experiment. The Dtype of discrete covariate has the same

meaning with the corresponding one for *atype*, but some types can be those of object variates and have slightly different meaning. The first two types, "score" and "sorted" are only for object variates and Dtype proper types, Explanation of the other Dtype common with Atype is duplicated for convenience.

dtype: "score"
Score like grade by examination. The range of the score should be specified by Range or Code.

| "sorted"
Original data is sorted at the stage of data collection by some reason. An example is a sorted population of countries.

| "logical"
Logical variable. False and true are indicated by 0 and 1 in Data.body , respectively. The semantics of true and false can be described by Code whose first element corresponding to the false 0. This type can be considered as a special case of "category", but it is recommended to use this type for the case when only two choices are available, like male and female or left and right. Also note that the codes are 0 or 1 for "logical" data but 1 or 2 for categorical data.

| "category"
Categorical variable or nominal scale. This can be regarded as a kind of logical variable which takes more than two values, or a vector of logical variables such that only one of them is true. If ordering of the categories is meaningful, use the following "ordered category".

| "ordered category"
Categorical variables when categories are totally ordered. This is distinguished from "equispaced", "specified values" and "interval" mentioned below.

| "count"
Count of events, objects, individuals, etc. The "interval class" should be used for the case when continuous variable is cut into

several classes and frequencies of data in the classes is counted.

| "id"
Identification number or index. It may be used later for showing results of analysis, like classification or outlier check. The name of individuals can also be given by STRING of Levels. An "id" may represent a specific combination of covariate values, and is distinguished from "sequence" below. More informative identification by symbols or names is possible by Code.

| "sequence"
Identification numbers whose ordering would be meaningful for later analysis.

| "equispaced"
Equally spaced discrete values, at which continuous explanatory variable of a Controllable Factor was set at the experiment, which is not "interval class". This suggests the use of Chebyshev-Fisher orthogonal polynomials for the regression.

| "specified values"
Similar to "equispaced" but not equally spaced. An example is, records of each runner in 800m, 1500m and 5000m running race.

| "interval class"
Continuous observations are discretized by classification into several intervals. This is used in conjunction with CUT in Transform. Note the difference from Stype INTERVAL, in which, only time-intervals including event-time are observed and varies observation to observation.

4.3.3. Unit

Physical unit. Percentage % can be specified here. Description like "1 g" instead of "g" is recommended to allow "0.1 g" etc. If Transform is present, the unit before transformation can be described by 'orig.unit' in it.

4.3.4. Code

Code for each value of *dname*. This is similar to Levels in Axes but the Levels is only used for axis. No characters are allowed in the Data.body, so that such coding is essential in case of categorical data. Codes may be outside of this D&D, and should be searched for in an external database. This is the case when the code is complex and lengthy.

4.3.5. Range

Range of possible values of data. This item is needed only when the range of values are explicitly limited by a nature, for example, positive However, trivial rage, like 0 to 59 for minutes, should be omitted. numbers or a finite set of values.

range: (min = NUMBER max = NUMBER)
| NUMBERS
| STRINGS

The 'min' and 'max' give the possible minimum and maximum of data. For the case when the data is discrete but not equally spaced, give a set of possible numbers by NUMBERS. For more complicated range, use STRINGS to specify all the possible values of the data.

4.3.6. Transform

Transformation applied to original data. Currently the following transformations are defined. The ordering of items are significant. Multiple transformations are applied in the order of descriptions.

transforms: (*transform* ...)

transform : LIN = (location = NUMBER scale=NUMBER
orig.unit = STRING)
| LOG = (base = NUMBER orig.unit=STRING)
| BOX.COX = (lambda = NUMBER orig.unit=STRING)
| CUT = (breaks = NUMBERS orig.unit=STRING)
| PW = (alpha = NUMBER orig.unit=STRING)
| RATIO = (denominator = NUMBER orig.unit=STRING)
| INDEX = (denominator = NUMBER orig.unit=STRING)

4.3.6.1. Explanation of Transform items

The 'orig.unit' describes the unit before transformation.

LIN Linear transform with 'location' and 'scale'.

LOG

Logarithm with 'base' as specified.

B.COX

Box-Cox transform with parameter 'lambda', λ;

$$Y = \begin{cases} \dfrac{X^{1-\lambda}-1}{\lambda} & \lambda \neq 0 \\ \log X & \lambda = 0 \end{cases}.$$

CUT Cut by 'breaks' into intervals. See "interval class" in Section 5.1.1 and 5.2.2.

PW Power transform with the power 'alpha', α;

$$Y = \begin{cases} X^{\alpha} & \alpha \neq 0 \\ \log X & \alpha = 0 \end{cases}.$$

RATIO

By setting 'denominator' 100 or 365, composition ratio, density, ratio of components etc., can be described. Value of the data is less than or equal to 1. If the total of the composition ratio is not 1, use Stype SUM to describe it.

INDEX

index relative to 'denominator'. An example is economic indices, relative to a base year. The difference from RATIO is that the index can exceed 1 whereas the RATIO cannot. If a STRING is given to 'denominator', then it is a literal explanation of the base of the index.

Geometric or Arithmetic means are not regarded as transformations but explained in Explanation of Preface Part (see Example B below).

Examples of Data

Example B (Cox and Snell)

```
Explanations="The values of <MM>, <MF>, <FM> and <FF> are geometric
    means of intervals in month between two births classified."
Data=(
    f.size=(Long.name="Number of children in the family"
        Dtype="sorted"
        Unit="1 person"
            )
    births=(Long.name="Successive births in a particular series"
        Dtype="category"
        Code=("1-2" "2-3" "3-4" "4-5" "5-6") )
    MM=(Long.name="Geometric mean of intervals between male and male babies"
        Unit="1 month" )
    MF=(Long.name="Geometric mean of intervals between male and female babies"
        Unit="1 month" )
    FM=(Long.name="Geometric mean of intervals between female and male babies"
        Unit="1 month" )
    FF=(Long.name="Geometric mean of intervals between female and female babies"
        Unit="1 month" )
        )
```

Example O (Cox and Snell)

```
Data=(
    weight=(Long.name="Ratio of reacting weight of silver and iodine"
        Dtype=(RATIO=(denominator=1)
            LIN = ( location = 1.176399 scale=10E6 ) )
    silver=(Long.name="Silver batch"  Dtype="category"
        Code=("A" "B" "C" "D" "E") )
    iodine=(Long.name="Iodine batch"  Dtype="category"
        Code=("I" "II") )
            )
```

Example R (Cox and Snell)

```
Data=(
    sentences=(Long.name="Number of sentences"
        Dtype="count"
```

```
        Unit="1 sentence"
        )
    total.kai=(Long.name="Total number of kai's"
        Dtype="count"
        Unit="1 word"
        )
)
```

5. Data body

Data.body is a set of data themselves.

value : NUMBERS
| (.Dim = NUMBERS .Data = NUMBERS)

The first form gives a vector of numbers and the second form gives a matrix or an array, where .Data is vector of values and .Dim (meaning dimension) is an integer vector giving sizes of axes. The definition of dimension of the array should be consistent with the axis definition in the corresponding Data.structure definition. All of the data should be organized by Data.structure definition as relation schemes or arrays. The data without value is regarded as an external data, which is described in "external" statement.

Example C (Cox and Snell)

```
Data.body=(
    sentences=( .Dim=(10 4)
            .Data = (386 424 192 128 42 23 34 49 45 155 141 152
            86 48 29 32 23 38 28 94 34 35 28 5 19 17 8 9 11 37
            17 16 13 6 12 9 16 10 4 24)
            )
    total.kai=(282 281 185 82 107 99 99 91 68 253)
  )
```

Example H (Cox and Snell)

```
Data.body=(
    purity=(
        7.2 6.3 8.5 7.1 8.2 4.6 8.5 6.9 8.0 8.0 9.1 6.5 4.9 5.3 7.1
        8.4 8.5 6.6 9.1 7.1 7.5 8.3
        7.2 6.3 8.5 7.1 8.2 4.6 8.5 6.9 8.0 8.0 9.1 6.5 4.9 5.3 7.1
```

```
        8.4 8.5 6.6 9.1 7.1 7.5 8.3)
    process=(22*1 22*2)
    fault=(0 1 1 0 1 1 0 1 0 1 0 0 1 1 0 1 0 1 0 1 0 0 0
        0 0 1 0 0 0 1 0 0 0 1 1 0 1 0 1 0 0 0 1 0)
    )
```

6. Uniqueness of description

6.1. Array or relation scheme ?

Relation scheme is quite powerful to represent a set of data but sometimes it becomes simpler if it is written as an array. If the values of data are homogeneous with respect to their Dtype, then it can be written as an array. The following kinds of data are appropriate to be organized as an array because of its simplicity, readability, and for easier numerical processing.

1) Observations obtained under control of design of experiment, like completely randomized, block experiment, completely randomized block, or split plot.
2) A contingency table of frequencies.
3) Polynomial regression model data: continuous response and continuous covariates.
4) Preference ordering. See the explanation of Atype "rank".

The following kinds of data are appropriate to be organized as a relation scheme because the length of data is variable even though the values are homogeneous.

1) ANOVA model response data with unbalanced blocks, or regression model data except polynomial regression.
2) A symmetric matrix representing similarity or dissimilarity distance among individuals.
3) Censored data
4) Time intervals with different time units like day and hour.

6.2. Attributes or data ?

There is no definite answer to describe auxiliary information like sensitivity of instrument or calibration as an attributes or as data. It may depend on the characteristic of information as well as its size.

Part 3 Examples of D&D

1. Cox and Snell (1981) Examples

	Possible.analysis	Primary.model	Design	Stype	Comments
B		point process		SEQ	
C		contingency			Arrays
H	LOGIS.LIN	paired dose response			0,1 response
J	REG	accelerated test	CR		Array
M	ANOVA		BIB	SUM	
O	ANOVA				Array
Q	ANOVA		NESTED		Array
R	ANOVA		SPLIT		Ragged array

D&D file

```
# max_line_len 512
# max_str_len 1023
# max_name_len 500
# version 2.3

Cox.B=
(
 Title="Intervals between adjacent births"
 Date=(created="1988-11-30" modified="1990-01-27")
 Contributor=(investigator="Greenberg and White (1963)" assembler="M.Takagiwa, Keio Univ.")
 Research.field=("demography"  "obsterics and gynecology")
 Keyword="genealogy"
 Purpose="Study of Natural variation of intervals between adjacent births due
          to (i) Systematic differences between the sex pairing MM, MF, FM and FF;
          (ii) Variation along with sequence number; and (iii) Pattern of
          interaction between (i) and (ii)."
 Source="B; Cox and Snell(1981), Applied Statistics, Chapman and Hall"
 Explanation="The values of <MM>, <MF>, <FM> and <FF> are geometric means of
          intervals in month between two births classified."
 Primary.model="point process"
 Data.structure=(
          interval=(Long.name="Mean intervals"
```

```
                    Columns=("f.size" "births" "MM" "MF" "FM" "FF")
                    Stype=(SEQ=(which="births" given="f.size")
                                )
                    )
          )
 Data=(
          f.size=(Long.name="Number of children in the family"
                    Dtype="sorted"
                    Unit="1 person"
                                )
          births=(Long.name="Successive births in a particular series"
                    Dtype="category"
                    Code=("1-2" "2-3" "3-4" "4-5" "5-6")
                                )
          MM=(Long.name="Geometric mean of intervals between male
                    and male babies"
                    Unit="1 month"
                                )
          MF=(Long.name="Geometric mean of intervals between male
                    and female babies"
                    Unit="1 month"
                                )
          FM=(Long.name="Geometric mean of intervals between female
                    and male babies"
                    Unit="1 month"
                                )
          FF=(Long.name="Geometric mean of intervals between female
                    and female babies"
                    Unit="1 month"
                                )
          )
 Data.body=(
          f.size=(2 2*3 3*4 4*5 5*6 )
          births=(1 1 2 1 2 3 1 2 3 4 1 2 3 4 5)
          MM=(39.8 31.0 42.8 28.4 34.2 43.1 25.3 30.3 33.7 41.6 24.2 27.6 29.8
                    34.2 40.3)
          MF=(39.5 31.5 43.7 28.1 34.4 44.3 25.6 30.1 34.0 42.1 24.4 27.7 30.2
                    34.2 41.0)
          FM=(39.4 31.4 43.3 27.5 34.3 43.3 25.6 29.9 33.7 41.9 24.0 27.5 30.3
                    34.1 40.6)
          FF=(39.3 31.1 43.4 27.8 35.0 42.8 25.5 30.0 34.7 41.3 24.5 27.6 30.8
                    33.4 39.9)
          )
)

Cox.C=
(
 Title="Statistical aspects of literary style"
 Date=(created="1988-11-30" modified="1990-01-27")
 Contributor=(investigator="Morton(1965)" assembler="M.Takagiwa, Keio Univ.")
 Research.field=("literary style" "Biblical")
 Purpose="Investigate the authorship of  10 Pauline works from the number
          of sentences having zero, one, two, ... occurrences of the word 'kai'"
 Source="C; Cox and Snell(1981), Applied Statistics, Chapman and Hall"
 Explanation="Reference: Morton A.Q. (1965) The authorship of Greek
```

```
            prose (with discussion) J. R. Statist. Soc. A 128 169-233"
Primary.model="contingency"
Data.structure=(
            freq=(Long.name="Frequency of occurrences"
                        Axes=("works" "kai")
                        Value="sentences")
            total =( Long.name="Total number of kai's"
                        Axes="works"
                        Value="total.kai"
                        )
            )
Axis=(
            works=(Long.name="Pauline works"
                        Atype="category"
                        Levels=("Romans(1-15)" "1st Corinth." "2nd Corinth." "Galat."
                                    "Philip." "Colos." "1st Thessal." "1st Timothy"
                                    "2nd Timothy" "Hebrews")
                        )
            kai=(Long.name="Frequency of kai's in a sentence"
                        Atype="count"
                        Levels=("0" "1" "2" "3 or more")
                        )
            )
Data=(
            sentences=(Long.name="Number of sentences"
                        Dtype="count"
                        Unit="1 sentence"
                        )
            total.kai=(Long.name="Total number of kai's"
                        Dtype="count"
                        Unit="1 word"
                        )
            )
Data.body=(
            sentences=(
                        .Dim=(10 4)
                        .Data = (386 424 192 128 42 23 34 49 45 155 141 152
                        86 48 29 32 23 38 28 94 34 35 28 5 19 17 8 9 11 37
                        17 16 13 6 12 9 16 10 4 24)
                                    )
            total.kai=(282 281 185 82 107 99 99 91 68 253)
                        )
)

Cox.H=
(
Title="Effect of process and purity index on fault occurrence"
Date=(created="1988-11-30" modified="1990-01-27")
Contributor=(
            investigator="D.R. Cox and E.J. Snell"
            assembler="M.Takagiwa, Keio Univ."
            )
Research.field=("industrial process"  "quality control")
Keyword=("matched pairs" "logistic linear model")
Purpose=("Test effects of modification of the process"
```

```
            "Batches of raw material of different purity were selected,
            and each of batches was divided into two equal sections; one of the
            sections was processed by the standard method and the other by a
            modified process."
            )
 Source="H; Cox and Snell(1981), Applied Statistics, Chapman and Hall"
 Explanation="Fictitious data based on a real investigation"
 Primary.model="dose response"
 Design=( RB=(factors=("process" "batches" "purity")
                        ftypes=( 1 3 4 )
                        response="fault"
                        )
            )
 Possible.analysis=( LOGIS.LIN=( explanatory=("purity" "process")
                        response ="fault"
                        )
            )
 Data.structure=( effect=(Long.name="Effect of process and purity"
                        Columns=("purity" "process" "fault")
                        )
            )
 Data=(
            purity=(Long.name="Purity index")
            process=(Long.name="Processes"
                        Dtype="category"
                        Code=("Standard" "Modified")
                        )
            fault=(Long.name="Fault"
                        Dtype="logical"
                        Code=("No Fault" "Fault")
                        )
            )
 Data.body=(
            purity=( 7.2 6.3 8.5 7.1 8.2 4.6 8.5 6.9 8.0 8.0 9.1 6.5 4.9 5.3 7.1
                        8.4 8.5 6.6 9.1 7.1 7.5 8.3
                        7.2 6.3 8.5 7.1 8.2 4.6 8.5 6.9 8.0 8.0 9.1 6.5 4.9 5.3 7.1
                        8.4 8.5 6.6 9.1 7.1 7.5 8.3)
            process=(22*1 22*2)
            fault=(0 1 1 0 1 1 0 1 0 1 0 0 1 1 0 1 0 1 0 1 0 0 0
                        0 0 1 0 0 0 1 0 0 0 1 1 0 1 0 1 0 0 0 1 0)
            )
)

Cox.J=
(
 Title="The number of cycles to failure of the length of worsted yarn under
            cycles of repeated loading"
 Date=(created="1988-11-30" modified="1990-01-27")
 Contributor=(investigator="A. Barella and A. Sust, Wool Textile Organization"
            assembler="M.Takagiwa, Keio Univ.")
 Research.field=("industry" "life test")
 Purpose="Life test of worsted yarn with the covariates regarding with
            the test"
 Source="J; Cox and Snell(1981), Applied Statistics, Chapman and Hall"
 Explanation="3-way factorial experiment model on log(cycles);
```

```
                or regression model of log(cycles) with covariates log(length),
                log(amplitude) and log(load)."
    Primary.model="accelerated test"
    Design=(
                CR=(
                        factors=("length" "amplitude" "load")
                        ftypes=( 1 1 1 )
                        response="cycles"
                        )
                )
    Possible.analysis=(
                REG=(
                        explanatory=("length" "amplitude" "load")
                        response="cycles"
                        )
                )
    Data.structure=(
                life.test=(Long.name="Life test"
                        Axes=("length" "amplitude" "load")
                        Value="cycles"
                        )
                )
    Axis=(
                length=(Long.name="Length of test specimen"
                        Atype="equispaced"
                        Levels=(250 300 350)
                        Unit="1 mm"
                        )
                amplitude=(Long.name="Amplitude of loading cycle"
                        Atype="equispaced"
                        Levels=(8 9 10)
                        Unit="1 mm"
                        )
                load=(Long.name="Load"
                        Atype="equispaced"
                        Levels=(40 45 50)
                        Unit="1 g"
                        )
                )
    Data=(
                cycles=(Long.name="The number of cycles to failure"
                        Dtype="count"
                        Unit="1 cycle"
                        )
                )
    Data.body=(
                cycles=(
                        .Dim=(3  3  3)
                        .Data=(674 1414 3636 370 1198 3184 292 634
                                2000 338 1022 1568 266 620 1070 210
                                438 566 170 442 1140 118 332 884 90 220 360)
                        )
                )
)
```

```
Cox.M=
(
 Title="Fertilizer experiment on growth of cauliflowers"
 Date=(created="1988-11-30" modified="1990-01-27")
 Contributor=(investigator="Mr.J.C. Gower, Rothamsted"
          assembler="M.Takagiwa, Keio Univ.")
 Research.field=("agriculture" "field test")
 Purpose="BIB experiments for the effect of nitrogen and potassium on the
          growth of cauliflowers."
 Source=("M;Cox and Snell(1981), Applied Statistics, Chapman and Hall"
          "Mr J.C. Gower, Rothamsted Experimental Station")
 Explanation=("For each fertilizer in each block 48 cauliflowers were grown,
          and the number of cauliflowers of 4 grades and unmarketable ones
          were counted. There are some missing values."
          "<grade> is the size measured by the number of cauliflowers
          fit into a standard crate.")
 Design=( BIB=(v=8 k=4 b=4 r=2
                    factors=("factor1" "factor2" "block" "grade")
                    ftypes=( 2 2 3 4 )
                    response="count"
                    )
          )
 Possible.analysis=(ANOVA=)
 Data.structure=(
          cauliflowers=( Long.name="Cauliflowers"
                    Columns=("count" "block" "factor1" "factor2" "grade")
                    Stype=(SUM=(which="count"
                              given=("factor1" "factor2" "block")
                              total=48)
                              )
                    )
          )
 Data=(
          count=( Long.name="Frequency count"
                    Dtype="count"
                    )
          block=(Long.name="Block"
                    Dtype="id"
                    Range=(1:4)
                    )
          factor1=( Long.name="Nitrogen"
                    Dtype="ordered category"
                    Code=(0 60 120 180 )
                    Unit="unit/acre"
                    )
          factor2=( Long.name="Portassium"
                    Dtype="ordered category"
                    Code=(200 300 )
                    Unit="unit/acre"
                    )
          grade=( Long.name="Grade"
                    Dtype="ordered category"
                    Code=("12" "16" "24" "30" "unmarketable")
                    )
           )
```

```
Data.body=(
        count=( 1 1 1 6 4 10 4 5 5 1 1 6 3 7 2 21 24 28 26 26 27 12 35 22
                8 22 27 8 16 31 13 24 13 12 9 14 13 28 5 22 33 17 14 30 22
                11 26 2 4 4 1 4 3 8 3 3 3 2 4 10 3 4 9 )
        block=( 6*1 3*2 4*3 2*4 4*1 4*2 4*3 4*4 4*1 4*2 4*3 4*4 4*1 4*2 4*3 4*4)
        factor1=( 3 4 1 3 2 4 4 2 3 2 1 4 3 3 4 1 3 2 4 4 2 1 3 2 1 4 3 1 3 4 2
                1 3 2 4 4 2 1 3 2 1 4 3 1 3 4 2 1 3 2 4 4 2 1 3 2 1 4 3 1 3 4 2)
        factor2=( 2 1 1 2 2 1 2 1 1 2 1 1 2 1 2 1 2 2 1 2 1 2 1 2 1 1 2 2 1 2 1
                1 2 2 1 2 1 2 1 2 1 1 2 2 1 2 1 1 2 2 1 2 1 2 1 2 1 1 2 2 1 2 1)
        grade=( 2*1 13*2 16*3 16*4 16*5 )
        )
)

Cox.O=
(
 Title="Atomic weight of iodine"
 Date=(created="1988-11-30" modified="1990-01-27")
 Contributor=(investigator=("Baxter and Landstredt(1940)" "Brownlee(1965)")
        assembler="M.Takagiwa, Keio Univ.")
 Research.field="chemistry"
 Purpose="Accurate determination of atomic weight of iodine from the
        ratios of reacting weight of iodine and silver, using five batches
        of silver and two batches of iodine."
 Source= "O; Cox and Snell(1981), Applied Statistics, Chapman and Hall"
 Explanation=("References: Statistical Methods in Medical Research,
        Armitage,P.(1971), Section 8.7, and Statistical Methods,
        Snedecor,G.W. and Cochran,W.G.(1967), Section 6.7"
        "Silver batch C in <silver> is a repurification of batch B,
        which in turn is a repurification of batch A." )
 Primary.model="Analysis of unbalanced data"
 Possible.analysis=(
        ANOVA=(
                explanatory=("silver" "iodine")
                response="weight")
                )
 Data.structure=(
        ratios=(Long.name="Ratios of reacting weight of iodine"
                        Columns=("weight" "silver" "iodine")
                )
        )
 Data=(
        weight=(Long.name="Ratio of reacting weight of silver and iodine"
                Transform=(
                        RATIO=(denominator=1)
                        LIN = ( location=1.176399  scale = 10E6 )
                        )
                )
        silver=(Long.name="Silver batch"
                Dtype="category"
                Code=("A" "B" "C" "D" "E")
                )
        iodine=(Long.name="Iodine batch"
                Dtype="category"
                Code=("I" "II")
                )
```

```
            )
 Data.body=(weight=(23 26 42 42 30 21 38 50 51 56 0 41 19 24 14 62)
            silver=( 1 1 2 2 3 3 3 4 4 5 1 1 1 2 2 4)
            iodine=( 10*1 6*2 )
            )
)


Cox.Q=
(
 Title="Strength of cotton yarn"
 Date=(created="1988-11-30" modified="1990-01-28")
 Contributor=(assembler="M.Takagiwa, Keio Univ.")
 Purpose="Estimating (i) the difference in mean strength of two worsted
            yarns produced by slightly different process, and (ii) the variation
            of strength between and within bobbins for yarns of this type."
 Source="Q; Cox and Snell (1981), Applied Statistics, Chapman and Hall"
 Research.field="industry"
 Design=(
            NESTED=(
                        factors=("yarn" "bobbin" "samples")
                        ftypes=(1 5 6)
                        response="strength"
                        )
            )
 Observation.mechanism=(
            SYS=(which="yarn")
            RND=(which="bobbin" given="yarn" size=6)
            RND=(which="samples" given=("yarn" "bobbin") size=4)
            )
 Possible.analysis=(ANOVA= )
 Data.structure=(
            loads=(Long.name="Breaking loads"
                        Axes=("samples" "bobbin" "yarn")
                        Value="strength"
                        )
            )
 Axis=(
            yarn=(Long.name="Yarn"
                        Atype="category"
                        Levels=("A" "B")
            )
            bobbin=(Long.name="6 bobbins selected at random"
                        Atype="sequence"
                        Levels=(1:6)
                        )
            samples=( Long.name="Samples"
                        Atype="id"
                        Levels=(1:4)
                        )
            )
 Data=(
            strength=(Long.name="Strength" Unit="1 oz")
            )
 Data.body=(
```

```
            strength=(.Dim=(4 6 2) .Data=(15.0 17.0 13.8 15.5 15.7
                        15.6 17.6 17.1 14.8 15.8 18.2 16.0
                        14.9 14.2 15.0 12.8 13.0 16.2 16.4 14.8 15.9 15.6 15.0 15.5
                        18.2 16.8 18.1 17.0 17.2 18.5 15.0 16.2 15.2 15.9 14.5 14.2
                        15.6 16.0 15.2 14.9 19.2 18.0 17.0 16.9 16.2 15.9 14.9 15.5)
                        )
            )
)

Cox.R=
(
 Title="Biochemical experiment on the blood of mice"
 Date=(created="1988-11-30" modified="1990-01-28")
 Contributor=(assembler="M.Takagiwa, Keio Univ.")
 Research.field="biochemistry"
 Purpose="To see the effect of treatments A, < treatment.a>,
            and B, <treatment.b>, on the amount, <amount>, of substances S
            in mice's blood"
 Source="R; Cox and Snell(1981), Applied Statistics, Chapman and Hall"
 Explanation="References: Snedecor and Cochran (1967, Section 12.12),
            Armitage (1971, Section 8.5), and Cox (1958, Section 17.4)"
 Design=(
            SPLIT=(
                        factors=("day" "sex" "treatment.a" "treatment.b")
                        ftypes=(3 2 1 1)
                        response="amount"
                        )
            )
 Observation.mechanism=( RND=(which="day" given="sex" size=4) )
 Possible.analysis=(ANOVA=)
 Data.structure=(
            effect=(Long.name="Amount of substances S"
                        Columns=("day" "sex" "treatment.a" "treatment.b" "amount") )
            )
 Data=(
            day=(Long.name="Day of treatment"
                        Dtype="sequence"
                        Unit="1 day"
                        )
            sex=(Long.name="Sex of mices"
                        Dtype="logical"
                        Code=("male" "female")
                        )
            treatment.a=(Long.name="Treatment A"
                        Dtype="logical"
                        Code=("absent" "present")
                        )
            treatment.b=(Long.name="Treatment B"
                        Dtype="logical"
                        Code=("absent" "present")
                        )
            amount=(Long.name="Amount of substance S"
                        )
            )
 Data.body=( day=( 4*1 4*2 4*3 4*4 4*5 4*6 4*7 4*8 )
```

```
        sex=( 8*0 4*1 4*1 8*1 4*0 4*1 )
        treatment.a=(0 2 0 2 0 2 0 2 2 0 2 0 2 0 2 0 2 0 2 0 2 0 0 2 0 2 0 2 2 0)
        treatment.b=(1 1 0 0 0 0 1 1 1 1 0 0 0 1 0 1 1 0 0 1 0 1 1 0 1 1 0 0 0 1 0 1)
        amount=(4.4 6.8 4.4 1.8 5.3 3.3 1.9 8.7 7.1 4.3 5.3 7.0 1.8 4.8 1.6 3.1
                    5.1 3.7 5.9 6.2 5.4 5.7 6.7 6.5 6.2 9.3 5.4 6.9 5.2 7.9 6.8 7.9)
        )
)
```

2. Earth tide data

D&D file

```
#external exnef="exnef.data"
Tide=
(
 Title="Earth tide at Esashi, Extensiometer NE (Free End)"
 Date=(created="1989-10-07")
 Contributor=(
            investigator=(
            "M.Ishiguro, The Institute of Statistical Mathematics"
            "National Astronomical Observatory"
                        )
            assembler="M.Takagiwa, Keio Univ."
            )
 Research.field="geophysics"
 Purpose="To study internal structure of the earth as a solid
            from the measurement of extensiometer reflecting earth tide"
 Explanation=( "Sensitivity of the instrument is corrected
            at each point of <s.point> from the level <s.prev> to the level <s.new>.
            One possible way of calibration is to re-scale <exnef> by the linearly
            interpolated sensitivity" "<gap.v> is the gap at <gap.point>,
            which is indicated by the re-scaled <exnef>."
            )
 Data.structure=(
            environ=(Long.name="Condition of measurement"
                        Axes="environ"
                        Value="environ.v"
                        )
            s.change=( Long.name="Sensitivity change of the extensiometer"
                        Columns=( "s.point" "s.prev" "s.new")
                        Stype=( INDEX=( which="s.point"  of="exnef"))
                        )
            gap=( Long.name="Gap after calibration"
                        Columns=("gap.point" "gap.v" )
                        Stype =( INDEX=( which="gap.point" of="exnef") )
                        )
            strain=(Long.name="Measurement for Earth tide survey"
                        Columns="exnef"
                        Stype=(
                                    TIME.SERIES=(
                                                time=("year" "month" "day" "hour"
                                                            "minute")
                                                value="exnef"
                                                )
                                    INVALID=(
                                                which="exnef"
                                                where=(1:168 1177:1190
                                                1773:1778 1822:1828 2447:2467
                                                3005:3020 3101:3114 4162:4169 5297:5305
                                                5322:5328 5346:5354 6567:6573 8051:8060
                                                8525:8538 8579:8781 10105:10106)
                                                code="missing"
```

```
                                                    )
                                          )
                                )
                    )
          Axis=(
                    environ=( Long.name="Environment"
                              Levels=("latitude" "longitude" "altitude" "gravity" )
                              )
                    )
          Data=(
                    year=(Long.name="Start and end year")
                    month=(Long.name="Start and end month"
                              Code=("Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep"
                                        "Oct" "Nov" "Dec")
                              )
                    day=(Long.name="Start and end day")
                    hour=(Long.name="Start and end hour")
                    minute=(Long.name="Start and end minute")
                    environ.v=( Long.name="Environment variable")
                    s.point=( Long.name="Sensitivity change point")
                    s.prev=( Long.name="Sensitivity of extensiometer before correction")
                    s.new=( Long.name="Sensitivity of extensiometer after correction")
                    gap.point=( Long.name="Gap point of rescaled values of extensiometer")
                    gap.v=( Long.name="Gap values")
                    exnef=(Long.name="Extensiometer"
                              Unit="1 STRAIN" )
                    )
          Data.body=(
                    year=(1979 1979) month=(6 12) day=(1 31) hour=(0 23) minute=(0 30)
                    environ.v=(39.148 141.335 370.0 980.122)
                    s.point=(1 5004 9372 10272)
                    s.prev=(NA -0.9849 -0.9948 -0.9948)
                    s.new=(-0.9863 -0.9849 -0.9948 -0.2429)
                    gap.point=(292 1217 1359 1383 1477 2068 2226 3364 4189 4338 5000 5525
                              5830 6922 7216 7357 7830 8262 8270 8992 9329 12132 12211)
                    gap.v=( -30 -4.5  4  3.5  -1.5  36.5  1.8  -4  31  5  -18  17.5  39
                              -1.5  34.5 4  -35.5  1  -2  -4  -6.5  21  -1.4)
                    exnef=
                    )
)
```

Part 4 Formal definition of D&D rule in yacc/lex

1. D&D rule in yacc/lex

makefile

```
# d.d2s to convert a D&D file to the dump-format file in S.
d.d2s: d.d2s.y d.d2s.l
     yacc d.d2s.y
     lex d.d2s.l
     cc -o d.d2s -DYYDEBUG -DLEXDEBUG  y.tab.c -ll
     /bin/rm y.tab.c lex.yy.c
# s2d.d to convert a dump-format file in S to the D&D file.
s2d.d: s2d.d.l
     lex s2d.d.l
     cc -o s2d.d lex.yy.c -DLEXDEBUG -ll
     /bin/rm  lex.yy.c
```

d.d2s.y

```
%{
# inclnde <stdio.h>
# include <ctype.h>
# include <string.h>
# define LPAREN  depth++ ;
# define RPAREN  printf(")"); depth--;
# define COMMA  printf(",")
# define NEWLINE printf("\n")
# define MAXLEN 500

int depth, used[16], check_error=0, nline=0;
char *d_d = "NONAME";
char buf[MAXLEN+20], extn[MAXLEN];
static char *name[]={
          "Title",
          "Date",
          "Contributor",
          "Research.field",
          "Keyword",
          "Purpose",
          "Source",
          "Explanation",
          "Primary.model",
          "Design",
          "Observation.mechanism",
          "Possible.analysis",
          "Data.structure",
          "Axis",
          "Data",
```

```
          "Data.body"
          };

static int optional[]={0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0 };

%}

%start dd_file
%union    { char a[MAXLEN]; }
%token <a> PROTOCOL DATA DIM NAME STRING NUMBER
%%   /* begin rule section   */

dd_file   : dd
     | protocols dd
     | dd_file { NEWLINE;} dd
     | dd_file { NEWLINE;} protocols dd
     ;

dd   :  '(' { LPAREN; no_assign(); } elements  ')'
          {  RPAREN; end_dd();  NEWLINE; }
     | NAME '=' '(' { LPAREN; assign($1); d_d =$1;} elements ')'
          {  RPAREN; end_dd();  NEWLINE; }
     ;

protocols: protocol
     | protocols protocol
     ;

protocol: PROTOCOL { proto($1);}
     ;

elements:  element
     |  elements { COMMA; NEWLINE; } element
     ;

element :  NAME '=' STRING
          { atomic($1, $3); check($1);}
     |  NAME '=' NUMBER
          { atomic($1, $3); check($1);}
     |  NAME '=' empty
          { empty($1); check($1); }
     |  NAME '=' '(' { LPAREN; strings($1) ; } strings ')'
          {  RPAREN; check($1);  }
     |  NAME '=' '(' { LPAREN; expression($1); } numbers ')'
          {  RPAREN; end_expression(); check($1);  }
     |  NAME '=' '(' { LPAREN; list($1);} elements ')'
          {  RPAREN; check($1); }
     |  NAME '=' '(' { LPAREN; array($1) ; } array ')'
          {  RPAREN; check($1);  }
     ;

array     : dim { COMMA; } data
     | data { COMMA; } dim
     ;
```

```
dim  :  DIM  '=' '(' { LPAREN; vector($1); } numbers ')'
          { RPAREN;}
     ;

data :  DATA '=' '(' { LPAREN; vector($1); } numbers ')'
          { RPAREN; }
     ;

strings   :  string
     |  strings { COMMA; } string
     ;

string    : STRING
          { printf("%s", $1); }
     ;

numbers :  number
     |  numbers { COMMA; } number
     ;

number    : NUMBER
          { if(nline > 13 ) { printf("\n"); nline=0;}
               nline++; printf("%s", $1 );}
     | NUMBER ':' NUMBER
          { printf("%s:%s", $1, $3); }
     | NUMBER '*' NUMBER
          { printf("rep(%s,%s)", $3, $1); }
     ;

empty     :
     ;

%%   /*   start of programs  */

#include "lex.yy.c"

proto(s) char *s;
{
     char *pname, *dname, *file, ss[MAXLEN];
     strcpy(ss, s);
     pname=strtok(ss, "# ");
     if(!strcmp("external", pname) )
     {
     if(!(dname=strtok(NULL, "="))) {yyerror(s); exit(1);}
     if(!(file=strtok(NULL, ""))) { yyerror(s); exit(1); }
     sprintf(buf, "%c%s%c%s", ',', dname, '=', file);
     strcat(extn, buf);
     }
     else fprintf(stderr, "%s\n", s);

}

no_assign()
{
      printf( "structure(.Data=list(\n" );
```

```
}

assign(s) char *s;
{
     printf( "%c%s%c <- structure(.Data=list(\n", '"', s , '"' );
}

atomic(s1, s2) char *s1, *s2;
{
     printf( "%c%s%c=%s", '"', s1, '"', s2);
}

vector(s) char *s;
{
     /* expressions inside .Dim and .Data to be evaluated  */
     printf( "%s=c(", s);
}

expression(s) char *s;
{
     /* protect from evaluation to save space */
     printf( "%c%s%c=expression(c(", '"', s, '"');
}

end_expression()
{
     printf(")");
}

empty(s) char *s;
{
     printf( "%c%s%c=NULL", '"', s, '"');
}

strings(s) char *s;
{
     printf( "%c%s%c=c(", '"', s, '"');
}

list(s) char *s;
{
     printf( "%c%s%c=list(\n", '"', s, '"');
}

array(s) char *s;
{
     printf( "%c%s%c=structure(\n", '"', s, '"');
}

check(s)
char *s;
{
     int i;

     for(i=0; (i<16) && strcmp(s, name[i]) ; i++);
```

```
          if(i==16) {  /* Not keyname */
               if(depth == 1){
               sprintf (buf, "unrecognized top level name \"%s\" ", s);
               yyerror(buf);
                         }
                    }
               else{  /* Keyname */
               if(depth > 1){
               sprintf(buf, "unbalanced parenthesis for \"%s\"", s);
               yyerror(buf);
               exit(1);
                         }
               if(used[i] == 1){
               sprintf( buf, "duplicated name \"%s\" ", s);
               yyerror(buf);
                         }
               else used[i] = 1 ;
               }
}

end_dd()
{

     int i;

     if(strlen(extn))
     printf(", external=list(%s)", extn);
     printf(", class=\"d.d\")\n");
     for(i = 0 ;  i < 16  ; i++){
          if(!optional[i] && !used[i] ){
               sprintf(buf, "not found \"%s\"", name[i]);
               yyerror(buf);
                              }
          used[i]=0;
     }
}

yyerror(s) char *s;
{
     fprintf( stderr, "Error: line %d; %s ", yylineno,  s);
     fprintf( stderr, "(D&D %s)\n", d_d);
     fprintf( stderr, "Read ahead text: %s\n", yytext);
     check_error=1;
}

main(){
     exit( yyparse() || check_error);
}
```

d.d2s.l

```
D [0-9]
E [DEde][-+]?{D}+
W [ ,\n\r\t]
R [^ ,\n\"=\r\t]*
%%
^{W}*#.*$           { strcpy(yylval.a, yytext);
                          return(PROTOCOL);}
[a-zA-Z]{R}              { if(!strncmp(yytext, "NA", 2)){
                         strcpy(yylval.a, yytext);
                         return(NUMBER);
                         } else {
                         strcpy(yylval.a, yytext);
                         return(NAME);
                              }
                         }

".Dim"                   { strcpy(yylval.a, yytext); return(DIM); }
".Data"                  { strcpy(yylval.a, yytext); return(DATA); }

\"[^"]*                  { if(yytext[yyleng-1] =='\\')
                         yymore(); /* \" */
                         else {
                         strcpy(yylval.a, yytext);
                         input(); /*  read " from buffer*/
                         yylval.a[yyleng]='"'; /* put "*/
                         yylval.a[yyleng+1]='\0';
                         return(STRING);
                              }
                         }

[-+]?{D}+ |
[-+]?{D}+"."{D}*({E})? |
[-+]?{D}*"."{D}+({E})? |
[-+]?{D}+{E}             { strcpy(yylval.a, yytext);
                         return(NUMBER);
                         }


"*"                      { return('*'); }

":"                      { return(':'); }

"="                      { return('='); }

"("                      { return('('); }

")"                      { return(')'); }

{W}                      {;}
```

s2d.d.l

```
%{
char buf[20] ;
int token1, token2 ;
int eflg=0;
%}
%%
\"[^"]*\"[ ]*<-         { printf( "%s=", strtok(yytext, "\"<-")); }
"structure(.Data = list("    { printf("("); }
"list(" |
"structure(" |
"c("               { printf("\n(");}
"expression("           { eflg=1;}

"NULL"              {;}

\"[^"]*                 { if(yytext[yyleng-1] =='\\')
                        yymore(); /* \" */
                        else {
                        input(); /*  read " from buffer*/
                        printf( "%s\"\n", yytext);
                             }
                        }
","                 {;}

"rep("[^)]*")"          { strcpy(buf, yytext);
                        token1=strtok(buf,"rep(,") ;
                        token2= strtok(NULL, ")");
                        printf("%s*%s", token2, token1);
                        }
")"                 { printf(")\n"); }
"))"                { if(eflg){
                        printf(")\n");
                        eflg=0;
                        }
                        else printf("))\n");
                        }
\t                      {;}
"class = \"d.d\")"          { ;}
^$\n                    {;}
%%
main()
{
     yylex();
     }
```

2. S functions

```
"d.d2s"<-
function(file)
{
     if(!exists("lib.loc", frame = 0))
          assign("lib.loc", unix("echo $SHOME/library"), frame = 0)
     t.file <- tempfile()
     command <- paste(lib.loc, "/d.d/", "d.d2s < ", file, " > ", t.file,
```

```
          sep = "")
     on.exit(unlink(t.file))
     if(!unix(command, out = F))
          invisible(eval(parse(n = -1, file = t.file), local = F))
     else stop("failed to get D&D's")
}
"s2d.d"<-
function(name, file = "d.d")
{
     if(!exists("lib.loc", frame = 0))
          assign("lib.loc", unix("echo $SHOME/library"), frame = 0)
     t.file <- tempfile()
     dump(name, file = t.file)
     command <- paste(lib.loc, "/d.d/", "s2d.d < ", t.file, " > ", file,
          sep = "")
     unix(command, out = F)
     on.exit(unlink(t.file))
}
```

Part 5 A D&D processing system in S language

1. Import and export

To get a D&D into S, an S function `d.d2s` is prepared. This function reads a D&D file and convert D&D's in it into S objects with the attribute "class" with "d.d", see Chambers(1989). Conversely several objects which belong to the class "d.d" can be converted to D&D's in a file by the function `s2d.d`. The definition of those S functions and the related yacc/lex programs are listed in Part 4 of this paper.

Currently the function `d.d2s` understands only "external" statement in the protocol, and adds it to the "d.d" object as an attribute "external" with the external file name. Other lines of protocol are only displayed on the terminal. If D&D's are named in a D&D file, then the converted objects are stored in the working directory of S with the given names. Otherwise, the function `d.d2s` returns the "d.d" object converted from the last D&D in the file. To fill the value of external data with actual value, use a function `fill.data`. This function looks for the "external" attribute of the given object, finds the file and gives value to the corresponding element in Data.body as far as it has the value NULL. If Axes are defined in Data.structure, the dimension attribute .Dim is automatically added to the data.

2. Printing

The most frequently used function is `print.d.d`. It is automatically called when a "d.d" object is displayed. This function will the display the object as pretty as possible with In order to print "d.d" object as a usual S list, use `Print`. The output is similar to the original form of the D&D.

3. Utilities

Corresponding to the top level items of D&D, thirteen S functions, from `Title` to `Data.structure`, are provided. Each function displays each corresponding item as pretty as possible. The function `Data` returns a list of

datasets in Data.body after structuring them following the description in Data.structure, Data and Axis. The functions **anames** and **dnames** will return a character string of *anames* and *dnames* of a "d.d" object.

4. Analysis

The function **analysis** first looks for Possible.analysis, and if it finds the item, it will ask user to choose one from "possible analyses" besides "others". If the user selects one appeared in Possible.analysis, then the function will invoke an appropriate function for the analysis. Otherwise, the user will be placed in a frame where the list of data as returned by **Data** is available. That is, the user is in such an operation mode as the system function **browser** is invoked.

5. Creating a d.d object

The function **d.d** is the function to create a "d.d" object from S data objects specified by its argument, or to change some descriptions of a "d.d" object.

6. S function documentation

The followings are the online documentation of S functions prepared for the D&D processing system.

Data — Extract Data of a "d.d" object — Data

```
Data(x, dnames)
```

ARGUMENTS

`x` "d.d" object

`dnames` vector of character strings giving *dnames* to be extracted. Optional.

VALUE

list of data structurized according to Data.structure in `x`. If the argument `dnames` is given, a simple list or an array is returned.

Title — Utilities for a "d.d" object — Title

```
Title(x)
Date(x)
Contributor(x)
Research.field(x)
Keyword(x)
Purpose(x)
Source(x)
Explanation(x)
Primary.model(x)
Design(x)
Observation.mechanism(x)
Possible.analysis(x)
Data.structure(x)
```

ARGUMENTS

`x` "d.d" object

VALUE

Each function prints in a pretty format the corresponding description in `x`.

analysis — Analyze a "d.d" object — analysis

```
analysis(x)
```

ARGUMENTS

`x` "d.d" object

User will be asked to choose one of "possible analyses" or "others". If "others" is selected, user can apply any S function to analyze data of `x` in the frame where the objects extracted from `x` by the function `Data` are available as local objects.

anames	Axis and Data names	anames

```
anames(x)
dnames(x)
```

ARGUMENTS

x "d.d" object

VALUE

character string giving *anames* or *dnames* of x.

d.d	Create a "d.d" object	d.d

```
d.d(...)
```

ARGUMENTS

... a "d.d" object or any S objects

If a "d.d" object is given, user will be interactively asked to change or fill each description. A blank line cause no change and ^D causes deletion of the description. Other input line will replace the content of the item. If S objects are given, a skeleton with given data is created.

VALUE

an "d.d" object.

d.d2s	D&D to a "d.d" object	d.d2s

```
d.d2s(file)
```

ARGUMENTS

file character string giving the name of a D&D file

This function reads the D&D file and convert it into a "d.d" object in S. If D&D's are named in the file, then converted objects are stored in the working directory of S with the given names.

VALUE

If D&D's are not named in the file, this function returns a "d.d" object converted from the last D&D in the file.

fill.data	Fill external data in Data.body of a "d.d" object	fill.data

```
fill.data(x)
```

ARGUMENTS

`x` "d.d" object

VALUE

The object `x` whose external value is filled from an external file specified by the "external" attribute of `x`.

print.d.d	Print a "d.d" object	print.d.d

```
print.d.d(x)
Print(x)
```

ARGUMENTS

`x` an "d.d" object

This function is automatically called when a "d.d" object is displayed. The "d.d" object is printed in a pretty format. In order to print `x` as a usual S list, use `Print`.

s2d.d	"d.d" S object to D&D	s2d.d

```
s2d.d(name, file="d.d")
```

ARGUMENTS

`name` character vector giving the names of "d.d" object

`file` character name of a file on which the D&D converted from the object `name` will be written

References

Andrews, D. F. and Herzberg, A.M. (1985)
Data: A Collection of Program from Many Fields for the
Student and Research Worker, Springer, New York.

Becker, R.A., Chambers, J., and Wilks, A.R. (1988)
The new S Language, Wadsworth & Brooks, Pacific Grove, CA.

Chambers, J.M. (1989)
Classes and Functions in S, personal communication.

Cox, D. R and Snell, E.J. (1981)
Applied Statistics: Principles and Applications, Chapman and Hall, MA.

McCullagh, P. and Nelder, J.A. (1989)
Generalized Linear Models, 2nd ed., Chapman and Hall, London.

Maier, D. (1983)
The Theory of Relational Database, Computer Science Press, Rockville, MD.

Mallows, C.L. (1983)
Data Description, in Scientific Inference, Data Analysis, and Robustness,
Eds. G.E.P Box et al. pp. 135-150, Academic Press, New York.

Okuno, T. and Haga, T. (1969)
Design of Experiments, Baifukan, Tokyo (in Japanese).

Rafanelli, M. et al (eds.) (1989)
Statistical and Scientific Database Management, Fourth International Working Conference
SSDBM, Rome, Italy, June 1988 proceedings, Lecture Notes
in Computer Science, No.339, Springer.

Schreiner, A.T. and Friedman, H.G., Jr. (1985)
Introduction to Compiler Construction with UNIX, Prentice-Hall, Englewood Cliff, N.J.

Shibata, R., Sibuya, M. and Takagiwa, M. (1990)
D&D examples, Research Report of Dept. Math., Keio University, KSTS/RR-90/002.

Shoshani, A. (1983)
Statistical database: Characteristics, problems, and some solution, in Geutle,
J.E. ed. Computer Science and Statistics: The Interface, North-Holland, Amsterdam.

Sibuya, M. and Shibata, R. (1987)
"Electronic Journal of Data Analysis": A proposal,
Proc. Inst. Satist. Math., Tokyo, vol.35, 81-87 (in Japanese).

Smith, T. M. F. and Sugden, R. A. (1988)
Sampling and assignment mechanisms in experiments, surveys and observational studies,
International Statistical Review, vol.56, 165-180.

Wertz, C.J. (1986)
The Data Dictionary: Concepts and Uses, North-Holland,Amsterdam.