PNL-SA 9219
CONF-810342--2

Data Editing on Large Data Sets *

MASTER

James J. Thomas
Robert A. Burnett
Janice R. Lewis

Pacific Northwest Laboratory

Richland, Washington

Abstract

The process of analyzing large data sets often includes an early exploratory stage to first, develop a basic understanding of the data and its interrelationships and second, to prepare and cleanup the data for hypothesis formulation and testing. This preliminary phase of the data analysis process usually requires facilities found in research data management systems, text editors, graphics packages, and statistics packages. Also this process usually requires the analyst to write special programs to cleanup and prepare the data for analysis. This paper describes a technique now implemented as a single computational tool, a data editor, which combines a cross section of facilities from the above systems with emphasis on research data base manipulation and subsetting techniques. The data editor provides an interactive environment to explore and manipulate data sets with particular attention to the implications of large data sets. It utilizes a relational data model and a self describing binary data format which allows data transportability to other data analysis packages. Some impacts of editing large data sets will be discussed. A technique for manipulating portions or subsets of large data sets without physical replication is introduced. Also an experimental command structure and operating environment are presented.

Keywords: Exploratory Data Analysis, Relational Data Base, Data Editing, Statistics, Data Analysis, Subsetting

* Work Supported by U.S. Department of Energy
Under Contract DE-AC-06-76RLO 1830

I. Data Editing in the Data Analysis Process

The data analysis process can be represented by an iterative sequence of gaining understanding of the data, data cleanup and preparation, hypothesis formulation, hypothesis testing, and summarizing results. Much of this initial process of gaining the basic understanding through hypothesis formulation is called exploratory data analysis[1]. For the purposes of this paper the following simple model of the data analysis process will be utilized:

Gaining Basic
Understanding

Data Manipulation
and Cleanup

Hypothesis
Formulation

Hypothesis
Testing

Summarizing
Results

The first step in the model is to gain a basic understanding of the data. The data to be analyzed must come with some minimum syntactic attributes. Most often there is also a minimum semantic description of the raw data. This information along with the initial exploratory phases of data analysis lead to the basic understanding. The next step in the model is to cleanup and prepare the data for analysis. This often involves converting to common engineering units, handling missing values, categorization, transformations or other data manipulation operations required to develop a hypothesis. After developing a hypothesis the analyst may be required to again transform the data. Then the hypothesis is formulated and tested for validity. This provides the data analyst with a better understanding of the data, leading to the resultant analysis or subsequent iterations. The data editing operations are usually found in the first phases: gaining the basic understanding through hypothesis formulation.

Our experience with data analysis of medium to large data sets indicates that gaining this basic understanding and data preparation phase constitutes a time-consuming and often cumbersome process. This process typically involves the utilization of uniquely written computer programs which only delay and distract the data analyst. Therefore an interactive tool was designed to include base level capabilities from a collection of computer science disciplines to form an interactive data editor specifically tailored to edit large data sets.

The data editor is one of several techniques being developed by a joint group of computer scientists and statisticians at PNL with direction to develop methodologies to Analyze Large Data Sets(ALDS)[2-5]. Further references to the data editor will be by its name: ADE, ALDS Data Editor.

II. Data Editing Requirements

The data editing requirements will be divided into those that were initially defined and those requirements that became defined as a direct result of experiences with editing large data sets. The latter set of requirements will be discussed in Section III.

From observations of data analysis sessions, it is obvious that a highly interactive environment must exist. The process of gaining a basic familiarity with the data and then cleaning up and preparing the data for analysis is often exploratory in nature. Therefore a regimented batch process would not fit this phase of data analysis. The majority of the operations in the initial phases of the analysis of large data sets were found to be less than exotic. Therefore only the base level capabilities in each functional area were initially required to provide the minimum data editing requirements. Some of the required techniques can be commonly found in relational data management systems, statistical sampling packages, interactive graphics systems, and text editors. A few of the base level capabilities required in an interactive data editing environment provide the data analyst with the capibilities to:

* logically model the data to correspond closely to the data analyst's conceptual framework (usually a case by variable model),

* logically define and manipulate groups of data by case, variable, or value,

* modify data on a case by case basis or on a more global basis,

* modify variables as functions of other variables, constants, and arithmetic functions,

* create and delete variables and cases,

* ramdomly select groups of cases,

* specify general expressions with schemes to tailor them to specific data sets,

* view data graphically.

The data editor can now be defined as a computational tool for the exploration, manipulation, modification, and preparation of data prior to and during the hypothesis formulation phases of data analysis.

III. Observations and Impact of Large Data Sets.

When editing large data sets several observations and subsequent data editing requirements were formulated. These are as follows:

* The initial data manipulation process is awkward. The largeness of the data has an impact on how the data analyst proceeds. Such questions as, "will it fit?", " If not how can I partition the data set meanfully?", are common.

* The data's semantic descriptive information is vital in order to grasp the essential structure of large data sets. Minimum descriptive statistics should be provided as standard descriptive information e.g. range, mean, mode, distributions, missing value indicators, etc.

* Replication of portions of the data set are a significant problem. When subsetting the data into homogeneous clusters, one often creates a copy of a significant portion of the data set for each subset created. In a multi-million item data set, severe physical limitations can arise.

* Large data sets are often collected over time and under diverse conditions, making missing data the norm rather than the exeption.

* The human interface between the data analyst and the computional tools must be obvious to the data analyst. The constant distraction of an unfriendly interface becomes dominate when working an large data sets. Also different data analysts work in different interaction styles. One may prefer an elegant command syntax while another literally fumbles at the keyboard and prefers a graphic interaction (menu) style.

* The data analyst must have the ability to take small samples of the data, perform an operation (i.e. transformation) on that data, and verify the results, all in a temporary environment. Then after verification the operation may be performed on the entire data set.

* The review of the selection criteria, transformations, etc. is required prior to execution when working with large data sets. An interactive environment also provides a greater opportunity for making mistakes. The review process should be the default condition with an option to turn it off.

* A context-sensitive interrupt is required to provide the data analyst the capibility to examine a process in progress and determine its status, intermediate results, etc. An option to terminate the interrupted process is also required. This becomes particularly important when performing global operations on large data sets.

* Automatic command and data modifcation history is essential due

to the number of operations possible and flexibility offered in an interactive environment. The history logging should become a part of the file description.

IV. The ALDS Data Editor (ADE)

ADE utilizes a self-describing binary data file structure[6,7]. This file structure provides a conceptual relational model of the data which is easy to visualize. The model is that of a flat file of rows and columns. In the data analyst's terminology, these are cases and variables, respectively. The following paragraphs illustrate a few of the most commonly used facilities in the data editor to manipulate data in the early phases of the analysis of large data sets. All of the currently available ADE commands are listed and briefly described in Appendix A.

The first and probably the most often used facility is the specification of a complex relational clause to apply to subsequent operations. This clause must be a well-formed expression, e.g.:

```
DATE > 1975 & SAMPLE < 200
(VAR1 > VAR2) ! (VAR3 <= 25.5)
```

where DATE, SAMPLE, VAR1, VAR2, and VAR3 are variable (column) labels.

The following example will list the first 10 cases that satisfy the condition VAR1 < VAR2:

```
CONDITION VAR1 < VAR2
LIST 10
```

The transform facility allows the data analyst to modify existing values with a simple assignment or a complex function. Data value modifications can be performed on a single item or on the entire data set. A simple example to change the value of the current case of the variable named RADAR1 to 42.5 would be:

```
CHANGE RADAR1 = 42.5
```

A more complex operation to tranform all values in TOTAL1 to a function of several other variables would be:

```
XFORM TOTAL1 = VAR1 * (1-ABS(X))/VAR4
REPLACE ALL
```

Note the above XFORM statement specifies the transformation while the REPLACE statement performs the operation. The above facilities can now be combined to perform varaible transformations under specified conditions:

```
CONDITION VAR4 > 20.5
XFORM     VAR3 = VAR4 + VAR3
REPLACE ALL
```

The above example replaces all values in VAR3 by the sum of VAR4 and the old value of VAR3 for only those cases (rows) where VAR4 is greater than 20.5. The ability to create and delete variables and cases is also provided. Note that on creation of a variable the data analyst will be requested to provide semantic information about that variable. Early experience indicates that typically only a few additional variables are created during data analysis. These new variables are usually transformations of other variables which are generated to facilitate hypothesis testing. The delete operation is seldom used.

When analyzing large data sets it is mandatory to have the capability to define a complex operation and then perform that operation on the entire data set. These types of operations are called global as compared to local operations on individual cases and variables. Both global and local operations are supported. The analyst may perform a single operation or a group of operations by executing a group of commands in a command file. Combining this with a synonym capability allows the data analyst to specify a formula in a generalized expression and then specify a synonym to map the expression into the particular variable set for the current file.

Currently only a single missing value code is supported in all three data types (real, integer, and character). Data analysts at PNL have stated the need for at least five different missing value codes. A missing value is denoted by the '#' symbol. A search for all cases that contain a missing value in variable VAR1 would be initiated by the following commands:

```
CONDITION VAR1 = #
SELECT ALL
```

Handling large data sets usually requires an exploratory process of cutting the data set (i.e. forming subsets). Forming physical subsets (actual replications of parts of the original data set) when handling large data sets becomes a limiting factor in the data analysis process. This limitation may be either a function of external storage space (disk) or access time. These data sets are often temporary and are used to verify a hypothesis. Therefore a technique is introduced to create a virtual subset (V-SET). A V-SET appears from the analyst's perspective to be a separate logical file which all commands can reference. The following command sequence illustrates how a V-SET is formed by first providing a relational condition and then forming the V-SET definition via the TABLE or SELECT command.

```
CONDITION  FIRSTVAR < 20 & B > 15
TABLE
```

The V-SET is composed of a table of case indices referring back to the orginal file and associated descriptive information. The analyst can interrogate the table directly although most of the references are to the selected cases defined by the V-SET. The V-SET maintains its own sequencing as well as the original case sequence from the master file. Either may be used for referencing purposes. The formation of a V-SET can be controlled by individual or group case selection, case selection based on a relational condition, random selection, or any combination of the above. From the data analyst's point of view this provides a straightforward technique to perform stratified sampling. The following command sequence selects 50 cases randomly from the subset of cases that satisfy the condition THIRDVAR = 20.

```
CONDITION THIRDVAR = 20
RAND 50
TABLE
```

The data analyst may perform stepwise refinement analysis by a sequence of TABLE operations on resultant tables (in effect a "re-table" operation). Assuming the above sequence has been executed to form a V-SET, then the following sequence will form a new V-SET containing five cases randomly selected from the 50 previously chosen:

```
RAND 5
TABLE
```

When the analyst wants to create a separate physical file the SAVE command can be used. This command generates a new file containing the subset of cases and variables as defined by the current V-SET. Note that a subset of variables can be selected and/or reordered if desired. When passing data on to other analysis systems this becomes an important capability.

V. Human Factors and the Editing Environment

The editing environment is designed to be highly interactive. A menu driven syntax or a command syntax could be implemented as the user interface. The current implementation provides a free form command interface with response occurring after a command is entered.

There are two editing modes: MASTER and SUBFILE. MASTER mode provides editing on the master file with access to all the cases in the file. SUBFILE mode provides access to only those cases that are defined by the V-SET case table. On entry the user is asked for a master file name. Then the user may create V-SETs and switch back and forth between editing the master file or the V-SET. On entry the user is asked if a backup copy of the file is desired. If the data set is very large, however, generation of a backup copy may take a considerable length of time and consume additional mass storage space. Backup of large data sets should normally be accomplished by other means (e.g., batch during off hours).

Early in the implementation of the data editor we realized that the editing mode and the current case number were not obvious and allowed for confusion. Therefore a variable prompt was developed to display the status of the editor at all times.

The ADE prompt takes the form of either:

```
           ADE:Mn>>
or         ADE:Sn>>
```

where M = master mode
      S = subset (V-SET) mode
      n = current case number within the master file or the V-SET

ADE:M3>> illustrates editing in master file mode with the current case pointer pointing to Case 3 of the full data set. ADE:S242>> illustrates editing in subset mode with the current case being the 242nd case in the V-SET table.

Editing large data sets often involves careful setup of complex operations. Before starting a complex operation, it is desireable if not mandatory that the data analyst be provided with a display of the specification status and a verify option for the specified operation. This verification step can be turned off at the user's request. An example verification response would be:

```
  ADE:M1>> SELECT ALL
    CONDITION: A>B & C>20
    TRANSFORMATION: NONE
    VARIABLES SELECTED: ALL
    CASES SELECTED: ALL
    Do you want to continue y/n?
```

A dynamic interrupt facility is also required to provide the data analyst with the capabilities to suspend the operation in progress, determine the current status, and then optionally continue. This is especially important when performing complex operations on large data sets. The following sequence illustrates an interrupt during a REPLACE operation with 85000 cases in the file.

```
  ADE:M1>> REPLACE ALL
    Replace set for 85000 cases
    CONDITION A>B & C>20
    TRANSFORMATION: VAR1 = A+B
    VARIABLES SELECTED: ALL
    CASES SELECTED: ALL
    Do you want to continue y/n? Y
   -- CTRL/C During select at case 44381
    Do you want to continue y/n? Y
```

Another implication of dealing with large data sets is that the analyst cannot view all the data on a video screen at one time. One should utilize the terminal as a viewport into the data base. With this concept in mind, both vertical and horizontal scrolling are provided in the data editor.

Experience with large data sets indicates that typing all the variable names needed for an operation can be tedious and error prone. A simple notion of grouped variables helps this problem. The command VARSEL SAM34:SAMLAST will select all variables between SAM34 and SAMLAST inclusively in subsequent operations. The command CASSEL 10:50 specifies that subsequent operations will be performed on cases 10 through 50 inclusive.

A facility that is vital when editing large data sets is an automatic logging of all commands. This provides for a trail to improve data validity as well as a history for file integrity. This also provides the system developers with a tool for logging and verifying program correctness. The log files were also found useful for initial introduction and training purposes.

VI. ADE Command Syntax

The command syntax is designed with the philosophy that it is important for a command to be obvious rather than simple. The commands as implemented in ADE usually reflect the logical phrases in a complex operation.

The commands are divided into three classes:

Specification commands
Information commands
Operational commands

Specification commands provide the ability to specifiy relational conditions, selection of cases and variables, random sampling criteria, etc. These specifications are then applied to a variety of information and operational commands providing multi-dimensional viewing and subsetting on the data set.

Information commands provide the analyst with selected file viewing and printing. Information commands also provide access to the current specification status, i.e. CONDITIONS, SYNONYMS, etc. Listing implies output to the interactive terminal while printing implies output to the line printer.

Operational commands include the creation of new data variables, subsetting, locating, deleting, replacement, command file execution, synonym definition, and other file manipulation commands. If desired, one can combine commands together to form a more complex command, as in the following examples:

```
CON A<25 & B>50 AND CASSEL 10:50 SELECT ALL
LIST 20 CONDITION A<25
```

All ADE commands are listed and briefly described in Appendix A.

VII. Conclusion

The data editor concept provides a unique approach to the initial data preparation phases of data analysis on large data sets. Its capabilities are what the authors define as the minimum functional needs to start exploratory data analysis and effective data manipulation. The concept of a V-SET is well understood by the data analysts with little instruction. It seems natural in the highly interactive mode of data analysis. The command structure is still under evaluation at PNL. The automatic verification of command specifications and automatic logging of all commands seemed natural and of great benifit both during implementation and analyst use.

Many of the techniques discusssed in specific reference to large data sets may also apply to small and medium size data sets. However, when dealing with large data volumes, these techniques become a necessary part of data manipulation. The techniques described by no means encompass all techniques to handle large data sets but hopefully will stimulate additional thought.

Acknowledgements

The authors wish to thank the statisticians at PNL (Wes Nicholson, Dan Carr, and David Hall) for their guiding comments. Also the authors wish to thank Harvard Holmes of Lawerence Berkeley Laboratory and George Rogers of George Washington University for their editing comments.

References

[1] Tukey J.W., Exploratory Data analysis, Addison-Wesley Co., Reading Mass., 1977

[2] Thomas J.J. et al, Analysis of Large Data Sets on a Minicomputer, Proceedings of the Computer Science and Statistics: 12th Annual Symposium on the Interface 1979, Univ. of Waterloo, pp. 442-447

[3] Nicholson W.L., Analysis of Large Data Sets, Proceedings of the 1979 DOE Statistical Symposium, Oak Ridge, Tn., September 1980.

[4] Carr D.B., The Many Facets of Large, Proceedings of the 1979 DOE Statistical Symposium, Oak Ridge, Tn., September 1980.

[5] Burnett R.A., The ALDS Project - Computer Science Research Areas, Proceedings of the 1979 DOE Statistical Symposiom, Oak Ridge, Tn., September 1980.

[6] Thomas J.J. et al, Distributed Data Analysis in a Mobile Real Time and Minicomputer Network Environment, Proceedings Trends and Applications, 1979, IEEE Cat. No. 79CH2402-7C.

[7] Burnett R.A., A Self-Describing Data File Structure for Large Data Sets, Computer Science and Statistics: 13th Symposium on the Interface, Carnegie-Mellon University, Pittsburgh,Pa, 1981.

Appendix A: ADE Command Brief

SPECIFICATION COMMANDS:

```
   SEQPTR n               set sequence pointer to n.
   CONDITION  r           set relational condition r.
   CASSEL (case-list)     set sequential case selection
   VARSEL (var.-list)     set variable selection
   XFORM  arith.          set transform for variable
   RAND   {n,.fr}         select n samples / probability sample function
   VERIFY {ON,OFF}        set verify switch
   MASTER                 set mode to MASTER
   SUBFILE                set mode to SUBFILE (tabled subset)

      where: n is an integer;
             r is a Boolean expression which may include the
                operators <,>,<=,>=,=,^= (not equal), ! (OR),
                & (AND);
             case-list is a list of case numbers, e.g.  1, 5, 20:40:2 ;
             var.-list is a list of variable names, e.g. V1,V2,V5:V10 ;
             arith. is an arithmetic expression which may include
                +,-,*,/,EXP,LN,LOG,SQRT,SIN,COS,TANH,ATAN
```

INFORMATION COMMANDS:

```
 * LIST {n,ALL}         list n (or all) cases on the terminal
 * PRINT                print subfile on the line printer
   SHOW  { TABLE(cc)  VERIFY  CONDITION  CASSEL  VARSEL  XFORM
           SEQPTR  MODE  DESCRIPTION  SYNONYM  }
                          show status of command specification.
```

OPERATION COMMANDS:

```
 (cc) * TABLE (SELECT)        form a table of cases defining a subset
 (cc) * FIND  n               find nth case which meets specifications
 (cc) * REPLACE {n,ALL}       replace variables in n (or all) cases
 (cc) * DLCAS  n              delete n cases
      * DLVAR                 delete variables
        ERASE                 erase case table selections
        RESET                 reset all specifications
      * SAVE 'filename'       save subset in case table to 'filename'
        RELAB oldlab,newlab   change label "oldlab" to "newlab"
        NWVAR                 add a new variable to data set
        NWCAS                 add a new case to data set
        CHANGE  arith.        change single variable
 (cc)   EXECUTE 'filename'    execute a command file
        SYNONYM equiv,name    define synonym "equiv" for "name"
        EDIT                  close existing file and restart editor
        END                   end the editing session
```

* commands operate only on variables and cases qualified by SPECIFICATION commands.

cc -- command is CTRL/C interruptible