WORKING PAPER NO. SSMD-89-004 E

METHODOLOGY BRANCH

ANALYSIS OF EDITS IN A GENERALIZED EDIT AND IMPUTATION SYSTEM

SSMD-89-004 E

Philip Giles
Social Survey Methods Division, Statistics Canada

RÉSUMÉ

Ce document décrit, en détail, une partie des étapes du Contrôle et de l'Imputation dans une enquête ou un recensement, c'est-à-dire l'élaboration des contrôles et l'analyse rigoureuse de ces contrôles afin d'identifier les erreurs et les erreurs potentielles qu'ils contiennet. En particulier, on considère l'incorporation des spécifications de contrôle et leur analyse dans un système généralisé de contrôle et d'imputation; c'est-à-dire, un système qui est élaboré afin de rencontrer les besoins de plusieurs enquêtes. Le rapport discute de tous les types de données, bien que la majeure partie du rapport porte essentiellement sur le traitment des données numériques.

ABSTRACT

This document examines, in detail, part of the Edit and Imputation (E&I) process in a survey or census context, namely the edit specification and the rigourous analysis of these edits in order to identify errors and potential errors in them. In particular, the edit specifications and edit analysis are examined in terms of their implementation in a generalized E&I system; i.e., a system which is designed to meet the needs of several surveys. The report considers all data types, although the major focus of the report is on the processing of numerical data.

CONTENTS

ANALYSIS OF EDITS IN A GENERALIZED EDIT AND IMPUTATION SYSTEM

Philip Giles, Social Survey Methods Division

March 14, 1989

1. Introduction

In any survey or census situation, one must deal with nonresponse as well as reported data which are erroneous or inconsistent. The exact definition as to which parts of the survey process are comprised in the Edit and Imputation (E&I) steps varies. However, in the context of this report, E&I refers to the automated processing after data collection and follow-up which detects data errors and inconsistencies (editing), provides values for missing data and changes erroneous data(imputation).

Statistics Canada has undertaken to develop generalized survey systems. These are procedures and accompanying computer systems which can be used by several surveys, thus saving on multiple development costs. Among these is a generalized E&I system.

For a more complete description of E&I, see I.G. Sande(1982). Kovar, MacMillan and Whitridge(1988) and Giles(1988) provide more details on a generalized E&I system at Statistics Canada. This latter paper describes E&I as being comprised of five components: edit specification, edit analysis, edit application, error localization, and imputation. The edit specification is the step where the survey taker inputs the edit rules used to detect errors in the data. The edit analysis provides various diagnostics which indicate whether or not the specified edits are defined correctly. Briefly, the three subsequent stages evaluate each data record against each of the edits, identify which variables must be imputed, for each data record which has missing values of which fails at least one edit, and then provides imputed values in order that the data record, after imputation, has no missing values and satisfies all edits. The purpose of this report is to provide a more complete documentation of the edit analysis step.

The edits can be defined once the data content, and perhaps also the questionnaire, are finalized for a survey. The edit-analysis function has two objectives: determining the minimal set of edits required to define the acceptance region of the data, and examining the acceptance region for errors and inconsistencies. Since processing time is directly related to the number of edits, it is useful to remove unneeded edits, working only with the minimal set of edits in further processing. Diagnostics which provide descriptions of the acceptance region may indicate errors in the edits which are not evident in the original edit specifications.

As a result of edit analysis, a respecification of the edits may be required. The respecified edits may then be analyzed. This would save unnecessary delays in processing that would occur if problems with the edits were only discovered after data collection. However, situations may still occur where the survey officer may decide to change an edit specification during data processing. Edit analysis only reduces such occurrences.

The remaining sections of the report will discuss the following topics: types of edits, types of analyses, and algorithms for performing the analyses. It must be noted that these algorithms are based on work previously done at Statistics Canada by Fellegi and Holt(1976) and Sande (1976), for categorical and numerical data, respectively.

2. Edit Specifications

The edits define logical relationships and constraints on individual variables or between variables. In the context of this report, they are checks only on data collected for the same unit or respondent. When the data are collected from the same respondent repeatedly, an edit may include a comparison with data from a previous round of the survey, for the same unit.

Each edit may be specified as a condition which classifies data as either "acceptable" or "erroneous". For example consider the edit

$a_1x_1 + a_2x_2 \leq b$, where x_1, x_2 are the data values, and a_1, a_2, b are specified scalars. Suppose that values of x_1, x_2 which satisfy this condition are considered to be valid and no imputation is required. This edit is then said to be specified as "acceptable". However, an alternative and equivalent way to specify this edit is to specify the condition as $a_1x_1 + a_2x_2 > b$, where data values satisfying this condition would indicate that at least one of x_1, x_2 is in error, and would require imputation. This second edit is specified as "erroneous". Since one can always determine an equivalent "acceptable" edit for each "erroneous" one, and vice versa (i.e., the conditions are complementary), the survey taker should be allowed to specify each edit in a form which is convenient. However, for each specified edit, one must also specify whether the edit indicates an "acceptable" or an "erroneous" data combination. As a whole, the edit set defines the acceptance region of the data; that is, those points which would satisfy all edits. After the imputation phase (i.e., at the completion of the E&I processing), all data records will be in the acceptance region.

This document does not intend to further discuss the form in which a survey taker may specify the edits. A balance must be reached between ease of specification and ease of manipulation by the system. The following sections will treat edits in a form which is suitable for explanation of the edit analysis.

3. Types of Edits

The edits may be classified under four types:

(i) numerical,
(ii) categorical,
(iii) conditional numerical,
(iv) conditional categorical.

(i) A numerical edit can be represented as an equality or inequality in the survey variables. Since linear-programming techniques have been utilized for many of the algorithms used to process numerical data, numerical edits have generally been restricted to a linear form. Nonlinear

edits can often be handled by applying a uniquely invertible transformation. For example, if the nonlinear edit is $X_1 \leq X_2 * X_3$, three new variables, Y_1, Y_2, Y_3, would be created, where $Y_i = \log(X_i)$. Then the edit becomes $Y_1 \leq Y_2 + Y_3$. The transformation can be reversed after processing. One problem with this approach is that it is unclear what one then does with other linear edits involving X_1, X_2, X_3.

Another constraint imposed by the use of linear-programming techniques is that all variables be nonnegative. Deviations from this constraint can be handled through the standard linear-programming technique of expressing negative variables as the difference of two nonnegative variables, one of which is zero.

Some algebraic notation will be introduced here for numerical edits. This notation will be useful in subsequent sections.

Let

n	be the number of variables,
m_1	be the number of inequality edits,
m_2	be the number of equality edits,
$m = m_1 + m_2$	be the total number of numerical edits,
x_j, $j = 1,\ldots,n$	be the data value of variable j,
a_{ij}, $i = 1,\ldots,m$, $j = 1,\ldots,n$	be the coefficient of x_j in edit i,
b_i, $i = 1,\ldots,m$	be the constant of edit i,
A	be the edit matrix with entries a_{ij},
B	be the edit constant vector with entries b_i,
X	be the vector of data values x_j.

Then, in matrix form, the numerical edits can be expressed as

$$A_1X \leq B_1,$$
$$A_2X = B_2,$$

where A and B have been partitioned appropriately.
The acceptance region of these edits is convex.

(ii) A categorical edit can be represented as a logical statement containing a set of survey responses joined by the logical operators AND and OR. The symbols "&" and "|" will be used in this paper to denote AND and OR respectively. By "survey response" is meant, for example, "SEX = MALE", "MARITAL_STATUS = SINGLE", or "OCCUPATION=STATISTICIAN". Note that a survey response could be a numerical condition such as "A + B $\leq$ 10". Two examples of categorical edits are:

(a) RELATIONSHIP_TO_HHLD_REFERENCE_PERSON = 'WIFE' AND SEX = 'MALE',
and,

(b) (MARITAL_STATUS='MARRIED'|MARITAL_STATUS='WIDOWED'|MARITAL_STATUS ='DIVORCED') & AGE $\leq$ 15.

For each of these two edits, the indicated combination of data values would define a situation of unacceptable data. That is, a data record satisfying the edit would be in error and require correction. Fellegi and Holt(1976) suggest that specifying unacceptable combinations of data values for categorical edits is easier than specifying acceptable combinations.

A matrix format is proposed for considering categorical edits. Using similar notation to that used for numerical edits,
Let

n	be the number of variables,
m	be the number of categorical edits,
x_j, $j = 1,\ldots,n$	be the data value of variable j,
d_j, $j = 1,\ldots,n$	be the number of valid discrete values which x_j can assume,
e_{ijk}, $i=1,\ldots,m$, $j=1,\ldots,n$, $k=1,\ldots,d_j$,	be the coefficient representing edit i and value k of variable j,
$D = \sum_{j=1}^{n} d_j$	the length of each edit.

The coefficients e_{ijk} can take on values of 0 or 1 only, and are defined as:

(a) $e_{ijk} = 1$, for $k = 1,\ldots,d_j$ if variable x_j is not in edit i,

(b) $e_{ijk} = 1$, for those discrete values of k of variable x_j which are in edit i,

(c) $e_{ijk} = 0$, for those discrete values of k of variable x_j which are not in edit i, providing that edit i has at least one discrete value of variable x_j.

Examples of defining the values of e_{ijk} will serve to clarify the definition. Suppose that a survey has three categorical variables A, B, C. These variables can assume 2, 3, 2 different values respectively, namely, a_1, a_2, b_1, b_2, b_3, c_1, c_2. The edits and their representation in matrix format are as follows. All edits define situations of erroneous data.

1. $A = a_1$ & $B = b_2$ & $C = c_1$,
2. $A = a_2$ & $C = c_2$,
3. $(B = b_1 \mid B = b_3)$ & $C = c_2$,
4. $A = a_1$ & $(B = b_1 \mid C = c_1)$,
5. $(A = a_2$ & $B = b_1) \mid (B = b_2$ & $C = c_2)$.

Then the edit matrix will have 7 rows and 7 columns as follows:

	A		B			C	
Edit	a_1	a_2	b_1	b_2	b_3	c_1	c_2
1	1	0	0	1	0	1	0
2	0	1	1	1	1	0	1
3	1	1	1	0	1	0	1
4a	1	0	1	0	0	1	1
4b	1	0	1	1	1	1	0
5a	0	1	1	0	0	1	1
5b	1	1	0	1	0	0	1

Note that edits 4 and 5 were split into two parts due to the OR condition, but that this was not required for edit 3. This is due to the

fact that variable B cannot assume both values b_1 and b_3 simultaneously. Edit 3 could have been split out in a similar fashion as edits 4,5, but this notation allows savings of space in situations where values for a particular variable are grouped in an edit.

The final notation required for this section is to represent each row in the edit matrix as E_i, $i=1,\ldots,m$. E_i will have D values. The complete edit matrix will be referred to as E. E will have m' rows and D columns, where m' denotes that some edits will be split for representation in E.

An additional complexity is added when one considers the more realistic situation where there may be nonresponse to one or more of the data items. This can be handled in a straightforward manner by adding an additional column for each variable. Each column would correspond to the situation of nonresponse to that variable. Every specified edit would have an entry of one in all these columns. Additional edits would be added, one for each variable. Each of these edits would represent the fact that a nonresponse to a particular variable is considered an error, and that imputation is required. Since the addition of these columns does not change the algorithms used to perorm the E&I functions, it will, for the most part, be ignored for the remainder of this paper, for the sake of simplicity.

(iii) and (iv) A conditional edit is of the form

IF "logical statement" THEN "edit".

The "logical statement" is as defined for categorical edits. If the "edit" is numerical, as in (i), the complete edit is a <u>conditional numerical</u> edit. If the "edit" is categorical, as in (ii), the complete edit is a <u>conditional categorical</u> edit. Examples of these two types of edits are, respectively,

(a) IF SALARY $\leq$ 5000 THEN TAXES = 0,

and,

(b) IF HOURS_WORKED = 0 THEN WORK_STATUS = 'UNEMPLOYED'.

Both these edits specify conditions of acceptable data.

Combining the notation introduced in the numerical and categorical edit sections, one can refer to conditional numerical edits as having two parts, one referring to the "logical statement" and the other referring to the "edit". Note that the classification of a conditional numerical edit as "acceptable" or "erroneous" refers to the "edit" part, and not to the "logical statement". That is, the condition expressed by the "logical statement" refers to the fact that the "edit" is applicable only to a subset of the respondents, those who satisfy the condition.

For example, suppose that a survey has five numerical variables and one categorical variable, SEX. Some edits are applicable to all respondents whereas some are particular to either MALE or FEMALE respondents only. The edits can be specified using the E, A, B matrices as defined above. Matrices A, B define the linear constraints, whereas matrix E will have two columns. For each edit i,

$$E_i = \begin{cases} (1 \quad 0) & \text{if the edit is relevant for males only} \\ (0 \quad 1) & \text{if the edit is relevant for females only} \\ (1 \quad 1) & \text{if the edit is applicable to all respondents.} \end{cases}$$

It should be noted that, on occasion, conditional numerical edits can be recast as a simple numerical edit. Kovar et al(1988) give the following example. The edit is :

IF SALES GT 0 THEN PURCHASES GT 0.

This can be restated as:

PURCHASES $\geq$ constant * SALES,

where "constant" is a specified sufficiently small positive number.

Conditional categorical edits can be cast into the E matrix format used for categorical edits. The conditional categorical edit "IF p THEN q", where p, q are logical statements is equivalent to the unconditional categorical edit "p & NOT q" (in "erroneous" format). An example, taken from Fellegi and Holt(1976), is used to illustrate this. The specification is "If a person's age $\leq$ 15 years or (s)he is an elementary school student, then the relationship to head of the household must not be head and marital

status must be single." The edit is transformed into the desired format as follows:

Edit: [(Age$\leq$15) | (Elementary School)] implies [(not Head) & (Single)],

Edit(Erroneous): [(Age$\leq$15) | (Elementary School)] & not[(not Head) & (Single)],

Edit(Erroneous): [(Age$\leq$15) | (Elementary School)] & [(Head) | (not Single)],

Edit(Erroneous): (Age$\leq$15) & (Head)
(Age$\leq$15) & (not Single)
(Elementary School) & (Head)
(Elementary School) & (not Single)

Note that, as was the case with some of the edits in the example of categorical edits, this one edit became more than one edit (actually four) in converting it into the required form.

4. Edit Analysis for Different Edit Types

While the specific analyses which can be performed are not discussed until subsequent sections, it is important to consider globally how the edit analysis is performed when the set of edits are a mixture of types. Also, from this point on, all numerical and conditional numerical edits will be considered to be "acceptable", and all categorical and conditional categorical edits will be considered to be "erroneous". This choice follows the implementation made by Fellegi and Holt, and Sande in the CANEDIT and NEIS systems. Overviews of these systems are given by Hill(1978) and Sande(1979), respectively.

As noted earlier, conditional categorical edits can be cast in the same format as categorical edits. If required, numerical edits can always be cast in the same form as conditional numerical edits (simply by adding a condition which is always satisfied, to each unconditional edit). Therefore, in processing edit sets, one need only consider the following four situations.

1. All edits are numerical.
2. All edits are categorical.

3. All edits are conditional numerical.
4. Some edits are conditional numerical and others are categorical.

Obviously, category 1 is a subset of category 3. Although the implementation algorithms for the most part are identical for the two categories, some simplification is gained if there are no conditions on the edits.

Edit analysis for strictly numerical or strictly categorical data is straightforward using the techniques described in subsequent sections. A separate edit analysis is required for all subsets of the population defined by the various conditions, when considering edits which are all conditional numerical. Using the same example as was used in the specification of conditional numerical edits, an edit analysis is done for all edits relevant to males and another for all edits relevant to females. Note that some edits are subject to both analyses; those which are applicable to all respondents. These edits would be included for analysis in both sets. The number of subsets of the population, as defined by the conditional numerical edits may be large. However, if the subsets can be considered to be treated independently (i.e., different edits for different age groups, for different groups of occupations, or for different types of businesses), one would often wish to process each group separately. In a situation such as this, all edits within each group would be strictly numerical.

The final possible combination of edit types can be split into the set of conditional numerical edits and the set of categorical edits. Each of these two groups can be analyzed separately as discussed above.

E&I processing for other functions, such as error localization, need not treat datasets subject to mixed edit sets in the same fashion; that is, by splitting the edits and datasets. The discussion as to an appropriate method for processing mixed edit sets for those functions is outside the scope of this report.

5. Types of Edit Analyses

The edit analysis provides diagnostics and descriptions of the acceptance region defined by the specified edits. The results of the analyses would be reviewed by a subject-matter expert. As indicated in the introduction, the edit-specification and edit-analysis steps may be recursive, with fine tuning of the specifications resulting from the analyses. Once the subject-matter expert is satisfied with the results of the analyses, the minimal set of edits required to define the acceptance region are output to the subsequent stages of processing.

As explained in the previous section, edit analysis for all edit sets may be handled by providing algorithms to handle either a set of strictly numerical edits or a set of strictly categorical edits. There are seven types of analyses proposed for numerical edits. Three of these are also appropriate for categorical edits. While the concept of the edit analyses are equivalent for all types of edits, the algorithms used to implement them are not. The implementation of the concepts is addressed in the next section. The seven proposed analyses for numerical edits are: checks for consistency, redundancy, determinacy, and hidden equalities, the generation of a set of implied edits and extremal points, and the determination of bounds on the variables. Of these, checking for consistency and redundancy, and the generation of the implied edits are proposed for categorical edits.

A set of edits is inconsistent if the acceptance region is empty. In other words, there exist no data values which can simultaneously satisfy all edit constraints. It should be noted that a check for consistency of an edit set can only indicate whether or not it is consistent. If the set is found to be inconsistent, the edit(s) in error cannot be identified automatically. One could implement some type of decision rules to drop edits one at a time until consistency is achieved. However, the optimal choice may not be unique. Due to this, as well as to the fact that a review of the edits by a subject-matter expert should pinpoint the problem, a manual approach is recommended to handle inconsistent edits. It should be pointed out that if a set of categorical edits identify certain values

for a variable as being entirely outside the acceptance region (i.e., it is not possible to find values of the other variables which along with these identified values will pass all edits), then the consistency check will fail.

Redundant edits are those which are not active in defining the acceptance region. They may be removed from the edit set without any affect on further processing. Generally, redundant edits occur because there are other, more restrictive edits. Practically, a redundant edit is one which cannot fail unless at least one other edit fails. Therefore, a redundant edit is not required to identify that there are errors in the data.

In the numerical context, determinacy is a situation where the edits define only a single point, or only single values for certain variables as acceptable. Determinacy does not indicate a definite problem with the edits, although this is usually not a desirable situation with numerical data. It is likely that a situation of determinacy is an indication that the edit specifications are too restrictive.

Hidden equalities are numerical edits which were specified as inequalities, but which may be expressed (more restrictively) as equalities while not changing the acceptance region. Detection of hidden equalities may be an indication that there are errors in the edits, since the acceptance region is much tighter than was believed when the edits were specified. If not, the conversion of inequalities to equalities will reduce processing time at further stages.

For numerical edits, a useful diagnostic is the production of the lower and upper bounds of each variable as defined by the set of edits. This information is useful in several ways. First, the bounds can be checked to verify that they correspond to subject-matter knowledge. Edits may be added (removed) if the bounds are not restrictive enough (too restrictive). Second, in order for the algorithms to operate efficiently, it may be useful to have finite upper bounds on variables. (Positivity constraints mentioned earlier guarantee that zero is the minimum possible

lower bound, although it may be higher.) For variables which can assume very large values (i.e., are essentially unlimited in value), edits can be added to set "artificial", but finite upper bounds. These upper bounds should be set sufficiently high such that these artificial edits never fail. In this way the algorithms can operate at maximum efficiency without affecting the data. Another use for the variable bounds is to identify "near-determinacy". Consider, for example, a situation where all variables are bounded by zero and one thousand, except one which is bounded by 10.1 and 10.2. Effectively, this is a situation of determinacy and should be examined in the same way. However, it will not be identified by the determinacy check.

Another useful diagnostic is the group of implied edits. Implied edits are discussed in Fellegi and Holt(1976). They show the implicit relationships between variables as defined by the specified edits. An examination of the implied edits by the subject-matter expert may indicate relationships between variables which are known, or are thought, to be false. This then would show that errors exist in the edit specifications. While the set of implied edits is used as a diagnostic tool only, for numerical edits, error localization for categorical data uses the set of implied edits.

The extreme points are the vertices of the acceptance region defined by a set of numerical edits. The list of extreme points is one method to describe the multidimensional acceptance region. As with the implied edits, (possibly) false relationships between variables may be indicated by examining the extreme points. Also, examination of the extreme points might provide a useful insight for the subject-matter expert as to the implications of the edit specifications. To illustrate how this might work, consider a survey with four variables X_1, X_2, X_3, X_4. Suppose that one of the generated extremal points is (10,0,20,0), and further assume that the subject-matter expert knows that $X_1 > X_3$ in situations where X_2, X_4 are both zero. The generated extremal point violates this condition, which indicates that the specified edits need to be reviewed and modified.

6. Example to Illustrate Edit Analyses

In order to illustrate some of the definitions provided in the previous section, a numerical example is provided. The example here is identical to that which appears in Giles(1988). For categorical edits, the concepts of consistency and redundancy are equivalent to those for numerical data. It is hoped that the numerical example will be illustrative enough to convey an understanding for categorical edits as well.

The edit analyses may be performed graphically. However, most surveys involve many variables, thus rendering this method impossible (or, at least, very difficult). This example involves two variables only, x_1 and x_2. In this section, graphical techniques will be used to explain the analyses. This example will be used in sections 7.1.1 and 7.1.2 to illustrate the automated algorithms.

Nine edits will be used in this example, as given below.

I. $x_1 \leq 8$,

II. $x_2 \leq 10$,

III. $x_1 - x_2 \leq 6$,

IV. $2x_1 + x_2 \leq 22$,

V. $x_1 + x_2 \geq 20$,

VI. $x_1 + x_2 \leq 20$,

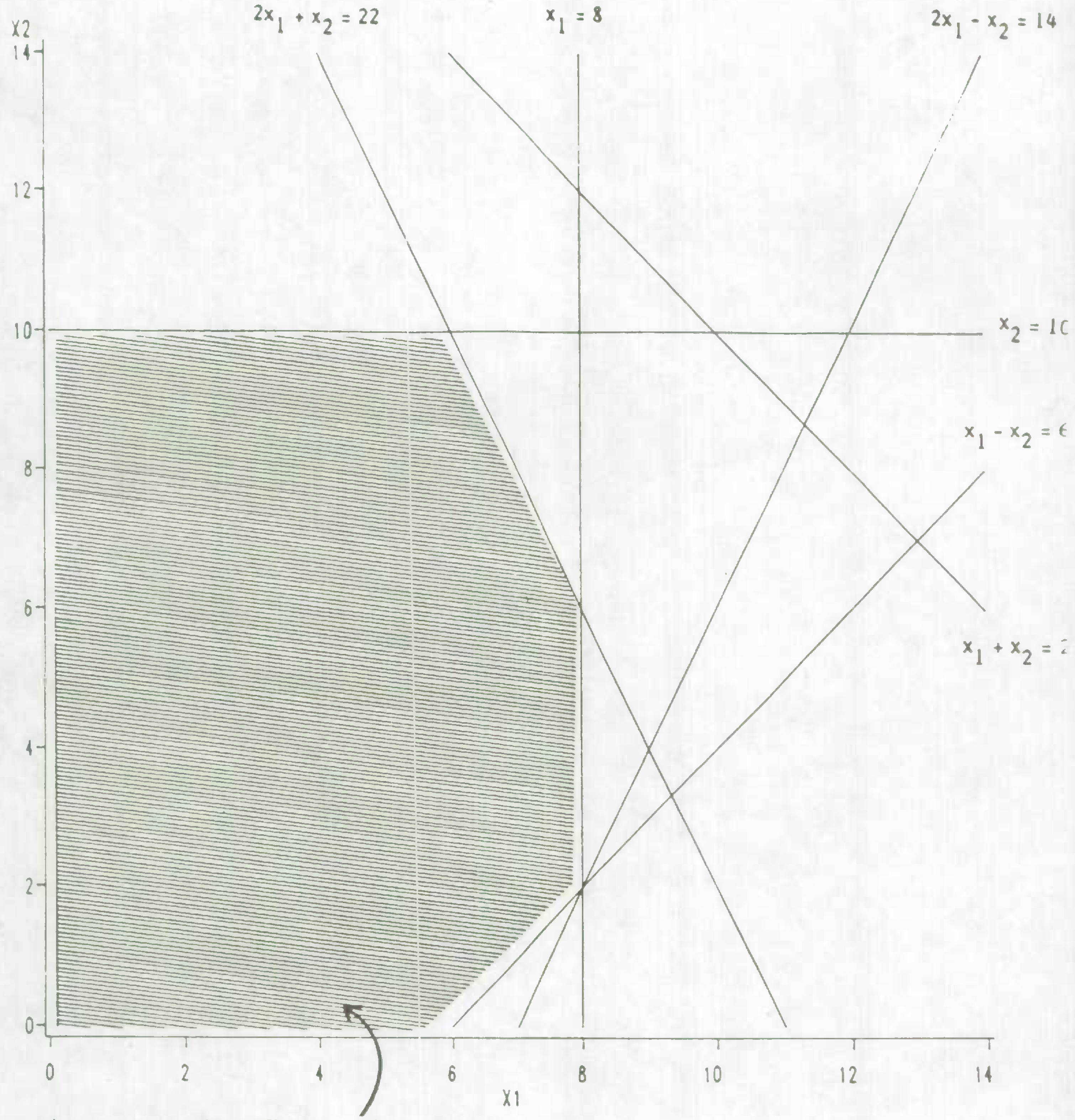
VII. $2x_1 - x_2 \leq 14$,

VIII. $2x_1 - x_2 \geq 14$,

IX. $x_1 - x_2 \geq 6$.

In addition, the two implicit positivity edits are $x_1 \geq 0$ and $x_2 \geq 0$.

Figure 1 shows the boundaries of the regions defined by each of the edits, as well as the acceptance region defined by edits I - IV plus the two positivity edits. In referring to groups of edits throughout the rest of the paper, the positivity edits are always assumed to be included. One will recall that in two space, the equation $ax_1 + bx_2 = c$, for any scalars a,b,c defines a line which separates the space into two parts. One part of the space is defined by the inequality $ax_1 + bx_2 \leq c$, while the other is



Acceptance region defined by edits: I. $x_1 \leq 8$,

II. $x_2 \leq 10$,

III. $x_1 - x_2 \leq 6$,

IV. $2x_1 + x_2 \leq 22$.

FIGURE 1 : Example to illustrate edit analysis

defined by $ax_1 + bx_2 \geq c$. One can decide which halfspace is related to each inequality by substituting the values of a point, to which part it is known to belong, into the expression $ax_1 + bx_2$ and examine whether it is greater than or less than c. (The point (0,0) is a convenient choice if $c \neq 0$.)

First consider the region defined by adding edit V to edits I - IV. It can be seen that there are no data points which can simultaneously satisfy all edits. Therefore the edit set comprising edits I - V is inconsistent.

In order to illustrate redundant edits, consider adding edits VI, VII to edits I - IV. One can verify that the region defined by these edits is identical to the region defined by edits I - IV. Therefore, edits VI, VII are redundant. It is interesting to note that edit VII is redundant, even though it hits the acceptance region defined by the edits. In addition, it can be seen that the removal of any of edits I-IV would result in a different acceptance region.

Next, consider the acceptance region defined by edits I-IV, VIII. In fact it is the single point (3,2). Determinacy occurs for this edit set.

The final edit set to consider is that comprising edits I-IV, IX. Edit III and edit IX can be removed and replaced by the equality edit $x_1 + x_2 = 6$. Although this example is a trivial situation of a hidden equality, they may occur in higher dimensions in a more complex fashion.

7. Algorithms for Edit Analysis

As indicated in Section 4, one need only provide edit analysis algorithms to handle either strictly numerical edit sets or strictly categorical edit sets. This section, therefore, will be split according to algorithms for numerical edits and those for categorical edits.

7.1 Numerical Edit Sets

Edit analysis for numerical edits makes use of linear-programming techniques. The linear program (LP), using notation introduced in Section 3, is:

$$\begin{aligned} &\text{Maximize } c'X \\ &\text{Subject to } A_1X \leq B_1, \\ &\qquad\qquad A_2X = B_2, \\ &\qquad\qquad X \geq 0. \end{aligned}$$

As will be shown, the various edit analyses will be performed by suitably choosing, and varying, the values of the vector c. This paper will not show how to determine the solution to the linear program. Those wishing details on this are referred to one of the many books devoted to linear programming, such as Chvatal(1983). In addition, many software packages provide a module to solve linear programs. It should be noted that the LP may have no solution, a unique solution, or many equally optimal solutions. Also, finding the minimum value of $c'X$ is equivalent to maximizing $-c'X$. The solution is as given, but the optimal value must be multiplied by -1.

7.1.1 Consistency

The edits can be checked for consistency by solving the LP with all elements of the vector c set to zero. This means that one needs only a possible solution in order to determine that the edit set is consistent. If this LP has no solution then the edit set is inconsistent, and should be reviewed and revised by a subject-matter expert.

One note of caution should be raised with regards to implementation. Depending on the algorithm used to solve the LP, an objective function (vector c) with all zeroes may not process efficiently. It is important to note that, theoretically, when checking for consistency, any choice of objective function is sufficient. Thus, the use of a zero objective function would yield the fastest result in most cases, as any feasible solution would be optimal.

7.1.2 Equality Edits - Redundancy and Determinacy

The first step in edit analysis of numerical edits is to examine the equality edits in isolation. In the course of checking for redundancy and determinacy, inconsistency of the equality edits is identified if present. Therefore these steps could be performed prior to the consistency check as described above in section 7.1.1. However, it is suggested that one perform that overall check first, as it can be done very quickly.

The process is to take the equality edits one at a time and add them to a "verification" matrix. This matrix has n+1 columns, the first n representing the coefficients of $x_1,\ldots,x_n$ in the matrix A_2, and the last column representing the elements of the vector B_2. The number of rows will change as each edit is added. By using matrix theory on combining rows, the matrix is modified as each edit is added so that it is in upper triangular form. That is, all matrix elements below the diagonal are zero. For example the matrix $\begin{pmatrix} 1 & 1 & 3 & 20 \\ 2 & 4 & 1 & 16 \end{pmatrix}$ can be modified to the upper

triangular matrix $\begin{pmatrix} 1 & 1 & 3 & 20 \\ 0 & -2 & 5 & 24 \end{pmatrix}$.

After each edit has been added to this verification matrix, and the suitable modifications have been made to render the matrix upper triangular, the following checks are performed.

i) If any row has all zero values, then the last edit added is redundant. This row is dropped from the verification matrix. The reason for this is that this edit was found to be a linear combination of the previous equality edits and therefore adds no further restrictions on the data.

ii) If any row has all zero values, except for that in the last column, then the edit set consisting of all edits previously added to the verification matrix, including the one currently under consideration, is inconsistent. This is due to the fact that a linear combination of the edits has resulted in producing $0 = w$, where $w \neq 0$. Since this equality is impossible to satisfy, the edit set is inconsistent.

iii) If the number of rows in the verification matrix is n, and inconsistency has not been found for the set of edits added to date, then determinacy has occurred for all variables.

iv) If after adding all equality edits to the verification matrix, any row has only one non-zero entry in the first n columns, then determinacy has occurred for that variable. For example, using notation introduced in Section 3, if the element is a_{2ij}, then x_j is deterministic, and the value is b_{2i}/a_{2ij}. If this value is negative, the edit set is inconsistent.

The appendix gives, in pseudocode, the algorithm used to perform the equality-edit analysis in both the NEIS and the current development of a generalized E&I system at Statistics Canada, GEIS.

7.1.3 Inequality Edits - Redundancy and Hidden Equalities

After successful completion of the edit analysis for equality edits, consider the edit set as a whole. Note that the notation introduced in Section 3 requires inequalities to be "less than". "Greater than" inequalities are converted to the required format by multiplying each coefficient by -1.

First, for each inequality edit i in A_1, maximize the quantity $A_{1i}x - b_{1i}$. This can be done by setting the values in the objective vector c as, $c_j = a_{1ij}$, $j = 1,\ldots,n$, and solving the LP. Subtract b_{1i} from the optimal solution. If this value is not zero, then inequality edit i is redundant and can be dropped from matrix A_1. The logic behind this strategy is as follows. The acceptance region defined by the edits is convex. In addition, the inequalities have all been converted to be "less than or equal to" inequalities. Therefore, in order for an edit to form part of the boundary of the acceptance region (i.e., not be redundant), it must take its maximum value on this boundary. Along the boundary the inequality becomes an equality. By obtaining the edit's maximum value and subtracting the value on the right-hand side, this maximum value will be zero for nonredundant edits.

If inequality edit i is not redundant, then minimize the same quantity

$A_{1i}x - b_{1i}$. As noted in the discussion of the LP, one can find this value by setting $c_j = -a_{1ij}$, $j = 1,\ldots,n$, and solving the LP. The optimal value is multiplied by -1 and b_{1i} is subtracted from it in order to obtain the desired minimum value. If this value is zero, then the inequality is a hidden equality. That is, the edit can be expressed, more restrictively, as an equality edit. In other words, the implication of having both a minimum and maximum value of zero for an edit is that the entire acceptance region lies along the boundary defined by this edit. As stated above, this boundary is the set of points for which the left-hand side and right-hand side of the edit are equal.

If an edit has been found to be not redundant by the first check and not a hidden equality by the second check, one additional check is required before determining that the edit is required in order to define the acceptance region. Geometrically, this check identifies edits which hit the acceptance region but which can be removed from the edit set without changing the acceptance region.

This final check again maximizes the quantity $A_{1i}x - b_{1i}$ as above, with one exception. Edit i is removed from the set of constraints before solving the LP. In other words, maximize the edit subject to all edits except itself. If the maximum value is zero, then the removal of the edit has not resulted in the removal of that particular boundary. Therefore the edit is not needed to define the region and is redundant. (Note that edits identified as redundant by the first check above would not be identified as redundant by this check. Therefore, it is important to discard redundant edits at the point at which they are identified.)

Once all inequality edits have been checked for redundancy and for hidden equality, one must return to the analysis-of-equality-edits algorithm in order to process those inequality edits which are found to be hidden equalities. The hidden equalities are added one at a time to the final verification matrix in order to check for redundancy and determinacy.

In order to illustrate the analytical techniques for determining redundancy of inequality edits, the example given in Section 6 will be used. The edits I-IV, VI, VII are considered.

The maximum value of each edit subject to all six edits are as given in Table 1 below, in the row labelled "max". These values are all zero, except for edit VI which attains a maximum of -4. Therefore edit VI is redundant and removed from further consideration.

The following row, labelled "min", shows the minimum value of each edit. Since none of these values are zero, none of these edits is a hidden equality.

Finally, the last row of Table 1 (max'), gives the maximum value of each edit subject to all other edits. All edits have strictly positive maximum values, with the exception of edit VII which has a maximum value of zero. Therefore edit VII is removed as redundant.

These two edits (VI, VII) were those identified graphically as redundant in Section 6.

Table 1: Computational Results of Edit Analysis Example

Edit	I	II	III	IV	VI	VII
Max	0	0	0	0	-4	0
Min	-8	-10	-16	-22	N/A	-24
Max'	1	12	1	4	N/A	0

7.1.4 Determinacy and Variable Bounds

The final analyses of numerical edits is the determination of variable bounds and the identification of further situations of determinacy. That is, in addition to the determinacies found in the checking of equality edits (although those will be detected again).

In order to determine the bounds of variable x_j, $j = 1,\ldots,n$, solve the LP with vector c set to maximize x_j (i.e., $c_j = 1$ and all other elements of the vector are zero), and then solve another LP to minimize x_j (i.e., set $c_j = -1$ and all other elements of the vector are zero, and the optimal value is multiplied by -1).

Determinacy occurs for a particular variable if the upper and lower bounds are equal.

7.1.5 Implied Edits

Fellegi and Holt (1976) provide an algorithm for the generation of implied edits from a set of inequality edits. Equality edits can always be expressed as two inequalities. An implied edit can be generated from edit r and edit s, if there is a variable k for which the coefficients in the rth and sth edits, $a_{rk} > 0$ and $a_{sk} < 0$. The implied edit, t, is calculated as:

$$a_{tj} = a_{sj}a_{rk} - a_{rj}a_{sk}, \qquad j = 1,\ldots,n,$$
$$b_t = b_s a_{rk} - b_r a_{sk}.$$

In descriptive terms, the implied edit is a linear combination of the two edits. The coefficients of the linear combination are chosen such that variable k does not appear in the implied edit (i.e., $a_{tk} = 0$). In this sense, there is a reduction in dimensionality in the implied edit, although the dimensionality of the implied edit may be larger than either of the generating edits. This is true since the implied edit will contain most variables with nonzero coefficients in at least one of the edits r and s, except variable k.

An implied edit can itself generate another implied edit. Therefore, the generation of implied edits continues iteratively, until no new implied edits are generated.

GEIS has implemented a different algorithm to generate the set of implied edits, as it was felt to be more efficient than that described above. For more details see Schiopu-Kratina and Kovar (1988). The Fellegi and Holt description is provided here as it is intuitively easier to understand.

7.1.6 Extremal Points

The extreme points are the vertices of the acceptance region defined by the edits. Each vertex is the result of the intersection of n edits (n = the number of variables). Since m (the number of edits) includes one positivity edit for each variable, $m \geq n$. A brute-force method to generate the extreme points is to take all possible combinations of n out of m edits ($C(m,n) = m! / n!(m-n)!$), and solve them as a set of linear equations. There may be no solution, the solution may be outside the acceptance region, or the solution is an extreme point of the acceptance region. To illustrate these possibilities, return to the example in Section 6, and consider the edits I-IV. Since n = 2, potential extreme points are found at the intersection of two edits. If one considers the intersection of the boundaries of edit I and the positivity edit, $x_1 \geq 0$, it can be seen that the lines are parallel and therefore there is no solution. If one considers edits III and IV, the intersection of the boundaries occurs outside the acceptance region due to the constraint of edit I. However the intersection of the boundaries of edits I, III, which occurs at the point (8,2), satisfies all other edits and therefore is an extremal point. Since C(m,n) can be a very large number, even for small values of m,n this approach is not an efficient one for processing. Schiopu-Kratina and Kovar (1989) provide details on the algorithm used for the generation of extremal points in GEIS.

One interesting additional point to mention is that Chvatal(1983), Chapter 18 proves that an upper bound on the number of extreme points is given by

$$C(m - [(n+1)/2],\ m - n) + C(m - [(n+2)/2],\ m - n),$$

where [x] = integer part of x.

By choosing various values of m, n one can quickly verify that this upper bound is much lower than the value of C(m,n). Also, in a number of edit sets examined by the author, the upper bound is not approached.

7.2 Categorical Edits

To date, the work on the current generalized E&I system at Statistics Canada, GEIS, has concentrated almost entirely on the processing of numerical data. Therefore, not as much detail can be provided in this section as in the previous one. When the methodology is developed for the E&I processing of categorical data, it is expected that ideas will be borrowed from Fellegi and Holt(1976) and from the development of the CANEDIT system. Despite the lack of additional detail, relevant ideas from Fellegi and Holt will be given here for the sake of completeness. The reader is reminded that the major difference in edit specifications between numerical and categorical edits, as used throughout most of this report, is that numerical edits specify constraints on the data values which must hold, whereas categorical edits specify situations of data conflicts, or errors, which indicate that some imputation is required.

At this point in the discussion, it is important to note two operational constraints. While all E&I functions for categorical edits can be handled using the E matrix representation, as first presented in Section 3, a particular survey application may exceed computer storage limitations. This is due to the fact that one column in the E matrix is required for each possible response (including one for nonresponse) for all data items. If the number of columns is very large, the software may not be able to operate. The second operational constraint is that multiple responses to a particular data item are not explicitly permitted. One can get around them by transforming multiple responses to additional single responses. For example, suppose that variable A can assume three values a_1, a_2, a_3, and that multiple responses are plausible. One then transforms the responses to variable A to a single response structure for a variable A' as follows:

A	a_1	a_2	a_3	a_1&a_2	a_1&a_3	a_2&a_3	a_1&a_2&a_3	Nonresponse
A'	1	2	3	4	5	6	7	0

The edits are specified using the values of A'. Obviously, this generates an additional complication to the task of specifying the edits.

7.2.1 Example of Categorical Edits

In order to illustrate the edit analysis of a set of categorical edits, an example will be used. This example has three variables A, B, C, which can assume 2, 3, 3 discrete values respectively. The valid responses for these variables are as follows:

A : a_1, a_2,

B : b_1, b_2, b_3,

C : c_1, c_2, c_3.

The edits to be used in the example, which identify combinations of data values which are in error, are:

I : $A = a_1$ & $(B = b_1 \mid B = b_2)$,

II : $A = a_2$ & $(B = b_2 \mid B = b_3)$ & $C = c_3$,

III : $(B = b_1 \mid B = b_2)$ & $C = c_2$,

IV : $B = b_2$ & $C = c_1$,

V : $(B = b_1 \mid B = b_3)$ & $(C = c_1 \mid C = c_2)$,

VI : $A = a_1$ & $(C = c_1 \mid C = c_3)$.

In addition, an edit 0 is considered where

0 : $A = a_1$ & $B = b_1$ & $(C = c_2 \mid C = c_3)$.

The matrix E for these edits is given in Table 2 below.

Table 2: Example of Categorical Edits

Edit	A		B			C		
	a_1	a_2	b_1	b_2	b_3	c_1	c_2	c_3
0	1	0	1	0	0	0	1	1
I	1	0	1	1	0	1	1	1
II	0	1	0	1	1	0	0	1
III	1	1	1	1	0	0	1	0
IV	1	1	0	1	0	1	0	0
V	1	1	1	0	1	1	1	0
VI	1	0	1	1	1	1	0	1

A set of data from a particular respondent can be represented in a similar format to the edits, as a string of zeroes and ones, with a one indicating each response. As mentioned in Section 3, nonresponse is handled by adding additional columns. In this way, the representation of a nonresponse to a particular data item is easily handled. One evaluates the data values against each edit by taking the scalar product of the vector of data values and the edit being evaluated. Since the edits are specified as "erroneous" conditions (i.e., indicate combinations of data values which are in error), the edit will fail if the scalar product is equal to n, the number of variables. (This assumes that multiple responses to any data item are not allowed.)

7.2.2 Redundancy

In order for an edit to be redundant, there must be another edit which has a 1 in E for every 1 in the row in E corresponding to the redundant edit. One will recall that a redundant edit is one which cannot indicate data errors unless at least one other edit does as well. Using this fact as well as the information on determining edit failures as given in the previous section, it can be quickly verified that if there exists two edits such that one edit has an entry of 1 in every corresponding position as the other, then the edit with the lowest cardinality (i.e., lowest number of one's) is redundant.

The method used to check for redundancy is to take the scalar product of every pair of rows. If the result for a particular pair is equal to the minimum cardinality of the two edits, then the edit with minimum cardinality is redundant and can be removed from the edit set. The cardinality of an edit is the number of elements in E with the value one.

Referring to the example and Table 2, examine edits 0, I, II. The cardinality of these edits are 4, 6, 4, respectively. The scalar product of edits 0, I is 4, that of 0, II is 1, and that of I, II is 2. Since the scalar product of edits 0, I is equal to the cardinality of edit 0, edit 0 is declared redundant.

7.2.3 Implied Edits and Consistency

This section provides the tools required to derive the implied edits. The details are essentially those given in Fellegi and Holt(1976). As noted in that article, an efficient means of implementing the strategy is required. The check for consistency is a byproduct of the derivation of the implied edits, as will be shown.

The edit set is inconsistent if an implied edit can be generated which has zero values for one variable only. Using the example, assume that an implied edit has a value of one for all columns, except the first. The interpretation of this implied edit is that a data record with $A = a_1$ is in error. However, by definition, $A = a_1$ is a valid value. This contradiction leads to the conclusion that the edit set is inconsistent.

As with implied edits for numerical data, at least one variable is eliminated in the combination of the edits to derive an implied edit. Also, as seen before, implied edits can be used to derive further implied edits. However, for categorical edits, it is not sufficient to examine only pairs of edits. Combinations of three, four, and so on, edits must be considered as an implied edit can be generated from two or more edits.

Three rules which can be used to determine if a combination of edits can potentially derive an implied edit are:

i) There must exist a variable which has at least one zero value for each edit in the combination under consideration, and, independently, for each discrete value in this variable, at least one of the edits in the proposed combination must have a value one. In other words, the variable must explicitly be present in all edits and all values of the variable must be present in at least one edit each.

ii) The combination of a previously derived edit with a subset of edits which were used to derive it will not produce an essentially new implied edit, and therefore need not be considered further.

iii) A proposed combination of edits with a variable identified by rule (i) above will not yield an essentially new edit if some subset of these edits, using the same variable has already generated an implied edit.

The derivation of a potential implied edit from a proposed combination of edits, using a variable identified by rule (i) above is performed as follows. (In Fellegi-Holt terminology, this variable is called the generating field.)

Step 1. For values of the variable denoted as the generating field, the value in the "new" edit is one.

Step 2. For values of all other variables, the value in the "new" edit is zero, unless all of the values in the contributing edits are one, in which case a value of one is assigned to the "new" edit.

Once the values are assigned to all elements of the "new" edit, a check is made to determine whether to retain it or not. This "new" edit is identified as an implied edit unless either all values for any variable are zero or else the variable is redundant with another existing edit (original or implied), as identified by the check described in Section 7.2.2.

In order to demonstrate the derivation of implied edits, consider edits I-VI in the example. (Edit 0 is dropped since it was found to be redundant.) The first step is to examine all pairs of these edits. The results are shown below in Table 3. Pairs of edits for which a generating field cannot be found are omitted.

Inconsistency was found when combining edit pair III, V, as well as IV, V. Therefore some changes to the edits are required. As mentioned previously, the detection of inconsistency does not identify the source of the problem. In this case, one must remove edit V, remove both edits III, IV, or make modifications to these edits. Assume that edit V is removed. This, of course, removes all implied edits generated by edit V. This only leaves two valid "new" edits; those generated by edit I, II and by II, VI. However the first is redundant with the second and is not needed. Label the implied edit generated by edits II, VI, as edit VII. Even though edit II is redundant with edit VII, it is left in the edit set for now. The reason for this will be apparent later.

Table 3: Example: Implied Edits Generated by Pairs of Original Edits

Edit Pair	Generating Field	A: a_1	A: a_2	B: b_1	B: b_2	B: b_3	C: c_1	C: c_2	C: c_3	Result
I,II	A	1	1	0	1	0	0	0	1	Valid
I,V	B	1	0	1	1	1	1	1	0	Valid
II,III	B	0	1	1	1	1	0	0	0	Invalid(C)
II,V	B	0	1	1	1	1	0	0	0	Invalid(C)
II,V	C	0	1	0	0	1	1	1	1	Valid
II,VI	A	1	1	0	1	1	0	0	1	Valid
III,V	B	1	1	1	1	1	0	1	0	Inconsist.
III,VI	C	1	0	1	1	0	1	1	1	Redundant,I
IV,V	B	1	1	1	1	1	1	0	0	Inconsist.
V,VI	C	1	0	1	0	1	1	1	1	Valid

The next step is to examine pairs involving this new edit. Only two pairs have suitable generating fields as given in Table 3A below.

Table 3A: Example: Implied Edits Generated by Pairs of Edits

Edit Pair	Generating Field	A: a_1	A: a_2	B: b_1	B: b_2	B: b_3	C: c_1	C: c_2	C: c_3	Result
I,VII	B	1	0	1	1	1	0	0	1	Redund.,VI
III,VII	B	1	1	1	1	1	0	0	0	Invalid(C)

No new implied edits are found. Now examine all triplets of edits with results as shown in Table 4. As before, combinations of edits without a suitable generating field are omitted from the table. Only one valid edit is found; that which is the result of combining edits II, III, IV. Label it edit VIII. Also inconsistency is found when combining edits III,IV,VII.

Table 4: Example: Implied Edits Generated by Triplets of Edits

Edit Triple	Generating Field	A a_1	A a_2	B b_1	B b_2	B b_3	C c_1	C c_2	C c_3	Result
I,II,III	B	0	0	1	1	1	0	0	0	Invalid,A,C
I,II,IV	B	0	0	1	1	1	0	0	0	Invalid,A,C
I,II,VI	A	1	1	0	1	0	0	0	1	Redund.,VII
I,II,VII	B	0	0	1	1	1	0	0	1	Invalid(A)
I,III,IV	B	1	0	1	1	1	0	0	0	Invalid(C)
I,III,VII	B	1	0	1	1	1	0	0	0	Invalid(C)
I,IV,VII	B	1	0	1	1	1	0	0	0	Invalid(C)
II,III,IV	B	0	1	1	1	1	0	0	0	Invalid(C)
II,III,IV	C	0	1	0	1	0	1	1	1	Valid
II,III,VI	C	0	0	0	1	0	1	1	1	Invalid(A)
II,III,VII	B	0	1	1	1	1	0	0	0	Invalid(C)
III,IV,VI	C	1	0	0	1	0	1	1	1	Redundant,I
III,IV,VII	B	1	1	1	1	1	0	0	0	Invalid(C)
III,IV,VII	C	1	1	0	1	0	1	1	1	Inconsist.
III,VI,VII	C	1	0	0	1	0	1	1	1	Redundant,I

Without modifying the original edits, an examination of the results to date indicate that there are four alternatives to consider as the modified set of edits. A choice of any of these edit sets will have no inconsistencies or redundancies. These four alternatives, with generated implied edits in brackets are :

i) Edits I, II, III, VI (VII),

ii) Edits I, II, IV, VI (VII),

iii) Edits I, III, IV, VI,

iv) Edits II, III, IV (VIII).

It is important to note that these four alternative edit sets are not equivalent representations. They define "data acceptability" differently.

One must rely on subject-matter knowledge in order to decide upon the "correct" edit set. One may also choose to modify the original edit specifications, and redo the analysis.

If one of the first two sets is chosen, edit II can be dropped, and edit VII added to the set (and, not left as an implied edit). This is due to the redundancy between the two as indicated earlier. However, edit II cannot be dropped from the fourth set of edits, as edit VII is not an implied edit from the fourth set.

Once one chooses the "correct" edit set, quadruples of edits can be checked for the generation of additional implied edits. In this example, no further implied edits can be generated.

8. Conclusion

This report has attempted to document the methodology related to the edit-analysis stage of the E&I processing. In some instances, algorithms have been provided for performing the analyses.

It has been stated that the edit-analysis stage is unnecessary if the edit specification has been done carefully. This is true to a certain extent. However, errors can still be made, and attempting to find errors in the edit specifications after the data are collected will result in delays in the survey production. More importantly though, the diagnostics provided by the edit analysis may result in supplying additional insight into the impact of the edit specifications as well as into the imputation process. This is due to the fact that the imputed values are determined based on the edit specifications.

REFERENCES

1. Chvatal, V.(1983). Linear Programming. W.H. Freeman, New York.

2. Fellegi, I.P., and, Holt, D.(1976). A systematic approach to automatic edit and imputation. J. Amer. Statist. Assoc., 71, 17 - 35.

3. Giles, P.(1988). A model for generalized edit and imputation of survey data. Can. Jour. Statist., 16 Supplement, 57-73.

4. Hill, C.J.(1978). A report on the application of a systematic method of automatic edit and imputation to the 1976 Canadian Census. Proc. Surv. Res. Meth. Section, Amer. Statist. Assoc.

5. Kovar, J.G., MacMillan, J.H., Whitridge, P.(1988). Overview and strategy for the Generalized Edit and Imputation System. Statistics Canada Working Paper BSMD-88-007E.

6. Sande, G.(1976). Diagnostic capabilities for a numerical edit specifications analyser. Statistics Canada technical report, Business Survey Methods Division.

7. Sande, G.(1979). Numerical edit and imputation. Int. Assoc. Statist. Comp., 42nd Session, Inter. Statist. Inst., Manila.

8. Sande, I.G.(1982). Imputation in surveys: coping with reality. Amer. Statist., 36, 145 - 152.

9. Schiopu-Kratina, I., and, Kovar, J.G.(1989). Use of Chernikova's algorithm in the Generalized Edit and Imputation System. Statistics Canada Working Paper BSMD-89-001E.

APPENDIX --- CHECKING OF EQUALITY EDITS

This appendix provides pseudocode for an efficient method of implementing the algorithms for the checking of equality edits which are described in Section 7.1.2.

```
1.  Designate NV  =  # of variables (> 0),
              NEQ =  # of input equality edits (>0),
              A   =  matrix containing the input equality edits, of size
                     NEQ X (NV + 1)  (i.e., the last column corresponds to
                     the right-hand-side scalar of the equality)
    Let  MAXI =  0
         I    =  1

2.  Read edit I from A into the vector DATA  (DATA is of size NV + 1).
    Let OMAXI = MAXI.

3.  Call SUBROUTINE CHECK.
     Arguments:   EQMTRX    (Vector of size (NV + 2)(NV + 1) / 2),
                  DATA      (Input edit),
                  NV, MAXI  (Input as above).
    EQMTRX (the verification matrix) stores modified, previously accepted
    edits.  Initially, EQMTRX is zero-filled.  The subroutine will attempt
    to add the new edit to the vector EQMTRX.

4.  On return from the subroutine, the following conditions are checked:

      (i)    IF MAXI = OMAXI AND DATA(NV+1) =  0,  then  the  latest edit is
             redundant.  This  edit  has  not  been  added  to  EQMTRX,  and
             processing can continue.
     (ii)    IF MAXI = OMAXI AND DATA(NV+1)  NE  0,  then  the  set of edits
             added to EQMTRX to date are inconsistent.  Processing of  these
             equality edits should stop, and these edits examined manually
             for errors.
```

(iii) IF MAXI = NV, then determinacy has occurred. All variables have a uniquely defined value, by the edits.

5. Increment I by 1.
   IF I > NEQ (i.e., no more equality edits) then STOP.

6. GO TO STEP 2.

SUBROUTINE CHECK

INDEX is an intrinsically defined function,

$$INDEX(I,J) = (2(NV+1) - I)(I - 1) / 2 + J.$$

Note that I in the subroutine is different from I in the calling program. INDEX is required since the two-dimensional verification matrix is being stored in a vector. Since the verification matrix is upper triangular, space is saved by not storing the values below the diagonal which are always zero.

```
     NDATA = NV + 1
     I = 1
     J = 1

100  IF J > NV THEN RETURN
     IJ = INDEX(I,J)
     IF DATA(J) NE 0 THEN GO TO 300
     IF EQMTRX(IJ) NE 0 THEN GO TO 200
     J = J + 1
     GO TO 100

200  IF I > MAXI THEN MAXI = I
     I = I + 1
     J = J + 1
     GO TO 100
```

```
300   IF EQMTRX(IJ) NE 0 THEN GO TO 500
      DO 400  K = J, NDATA
      IK = INDEX(I,K)
      T = EQMTRX(IK)
      EQMTRX(IK) = DATA(K)
      DATA(K) = T
400   CONTINUE

      IF I > MAXI THEN MAXI = I
      I = I + 1
      J = J + 1
      GO TO 100

500   IJ = INDEX(I,J)
      D1 = EQMTRX(IJ)
      D2 = DATA(J)
      D = SQRT(D1 * D1 + D2 * D2)
      C = D1 / D
      S = D2 / D
      DO 600  K = J, NDATA
      IK = INDEX(I,K)
      X = EQMTRX(IK) * C + DATA(K) * S
      Y = EQMTRX(IK) * S - DATA(K) * C
      EQMTRX(IK) = X
      DATA(K) = Y
600   CONTINUE

      IF I > MAXI THEN MAXI = I
      I = I + 1
      J = J + 1
      GO TO 100
```

#74848
c.1