FUNCTIONAL DESCRIPTION OF THE BANFF SYSTEM FOR EDIT AND IMPUTATION

Banff

Version 2.07

Banff Support Team

March 2017

Statistics Canada Statistique Canada

Canada

In order to familiarize the users of Banff with its many uses, a number of documents, including this one, have been prepared by Statistics Canada. For a list of these documents, the reader is invited to consult the reference section.

This document describes the methodology used in the Banff procedures. It should be useful to those who want to understand the mathematical methods underlying each of the Banff functions, in order to evaluate the various options offered.

Copies of this document can be obtained from:

Statistics Canada
Business Survey Methods Division
Generalized System Methods Section
17th floor, R.H. Coats Building
Ottawa, Ontario
K1A 0T6

La version française de cette publication est disponible sur demande.

TABLE OF CONTENTS

1. INTRODUCTION

Background

The Banff system for edit and imputation is a collection of specialized SAS procedures developed at Statistics Canada, each of which can be used independently or put together in order to satisfy the edit and imputation requirements of a survey. Data to be processed by Banff are assumed to be numeric and continuous and the edits used in Banff must be expressed in linear form. The user can choose whether to accept negative data or to reject it as invalid for each execution of a procedure. The exceptions to this are the Historical and Ratio Methods in proc Outlier, where the data must be positive. It is also assumed that some preliminary editing has been done at data capture stage and that respondent follow-up is complete.

Banff is the replacement for the Generalized Edit and Imputation System (GEIS). The methodology currently used in Banff is nearly identical to that of GEIS. However, there are several major structural differences between the two systems. The first is that Banff is based on the SAS architecture, while GEIS used Oracle as its underlying database structure. Also, the individual SAS procedures in Banff are independent of one another, while the modules in GEIS were linked. Thirdly, not only Banff is available on the UNIX computing platform like GEIS, but it is also available in the Windows computing environment on the PC. Due to these differences, Banff is inherently much simpler to use and more flexible than GEIS was.

Independence of SAS Procedures

The independence of the SAS procedures in Banff gives the user a great deal of freedom and flexibility. However, this independence also entails more responsibility for the user in ensuring that the inputs are of good quality, and that the outputs are interpreted and applied correctly. In GEIS, the related data tables are updated automatically by the system after each step of processing because all of the modules are linked together. That is, original data are simply overwritten by new data. This imposes some difficult restrictions on the user, though, such as manually creating backup copies of these tables if they want to trace the history of changes to the data. The positive aspect is that because the modules are linked, the data flowing from one module to the next is assured of being correct if the system is run without problem.

In Banff, each of the procedures accepts independent inputs provided either by the user or by another Banff procedure. Thus, in the case of inputs being supplied by the user from outside the system, the user has the responsibility of guaranteeing the quality of the input; Banff will attempt to process whatever is provided.

As well, each of the procedures provides its own unique outputs. In the case of field status codes, the format of the SAS datasets is very similar from one procedure to the next, but the codes will be different. The data records output from Banff procedures contain only those data which have been changed from the input data. Thus, the user has the responsibility of incorporating these changes into their original data unless the user works with the Banff Processor (see the *Banff 2.07 Processor User Guide*).

BY Variables in Banff

Like regular SAS procedures, Banff procedures are able to process data in BY groups. For example, rather than process separate datasets for each one of several industrial groups, a user may include all industrial groups in a single dataset, and Banff will process each of these groups independently according to the BY variable which identifies the industrial group.

Purpose of this Document
The purpose of this document is to describe the methods used in each Banff procedure, to give the methodological reasons for the choice of these procedures and to illustrate the application of each procedure with simple examples. Advantages and disadvantages of the parameters associated with each procedure are discussed, and examples of many special situations and how to handle them in Banff are given. Methodologists and subject matter officers could use this guide to understand the logic behind Banff so that they may be better able to evaluate their different edit and imputation options.

It is not necessary to read the sections of this document in the order presented. The reader who is interested in one particular procedure may refer to that section immediately without reading the sections which precede it.

Other Available Documents
User guides have been written for each of the SAS procedures in Banff. They describe the syntax, the parameters which can be specified, and options which exist. Examples are also provided. They are all included in the document *Banff 2.07 Procedures User Guide* (Banff Support Team, 2017) which can be obtained from the Banff support team.

Banff can be run in SAS, in SAS Enterprise Guide with the Banff tasks or, with the Banff Processor. The Banff 2.07 Processor User Guide (Banff Support Team, 2017) explains how to use the Banff Processor.

The Banff 2.06 Tutorial (Banff Support Team, 2014), which applies equally to both Banff versions 2.06 and 2.07, provides the user with data based on an actual survey conducted at Statistics Canada. The accompanying manual steps the user through the entire system. The tutorial would be of interest to anyone who will actually run Banff or to anyone who prefers a "hands-on" environment in which to become familiar with Banff.

Morabito and Shields (1992) give advice in the GEIS Applications User's Guide on how to fine-tune edits, how to set appropriate parameters and how to customize procedures to meet the needs of a particular application. This document is intended to be a reference guide to be used during the detailed specification of the edit and imputation phase of a survey. Though this guide was written for GEIS, many of the ideas and advice can in most instances be applied to Banff as well.

Structure of this Document

Figure 1.1 shows the basic functions of Banff and the sections of this document which pertain to each SAS procedure. In addition, Section 11 provides references and the appendices give the details of the method of calculation of medians, quartiles and imputation estimators used in Banff. A short description of each section follows Figure 1.1.

The Banff procedures are presented in the order that a user might apply them in the context of carrying out all of the edit and imputation steps for a typical survey. **One must keep in mind, as stated earlier, that each of the procedures is an independent entity.** Thus, the user can select only those procedures which they would want to apply to the processing of their survey data. Still, it is instructive to display the following diagram as a flow from one procedure to another, from the typical early steps of editing to the completion of imputation.

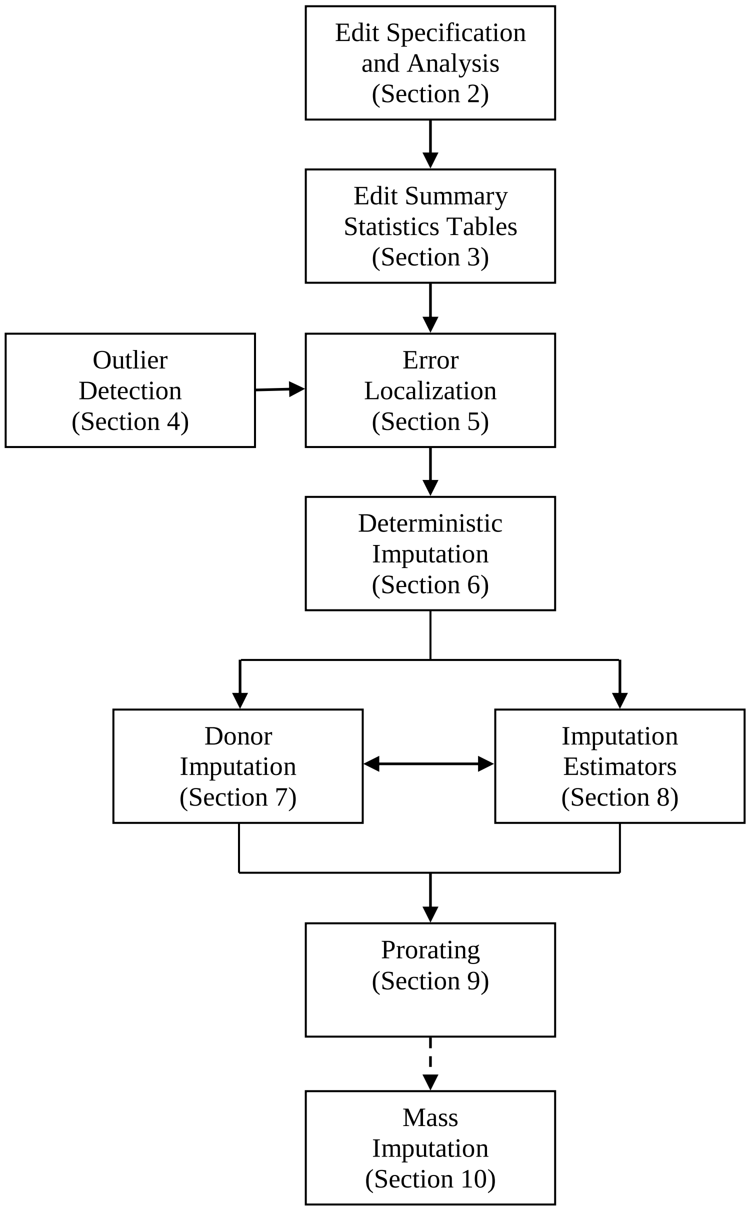

Edit Specification and Analysis (Section 2)
Edit Summary Statistics Tables (Section 3)
Outlier Detection (Section 4)
Error Localization (Section 5)
Deterministic Imputation (Section 6)
Donor Imputation (Section 7)
Imputation Estimators (Section 8)
Prorating (Section 9)
Mass Imputation (Section 10)

Figure 1.1 The logical order of the Banff procedures in the context of survey processing. Section numbers refer to sections of the document.

PROC VERIFYEDITS – Edit Specification and Analysis (Section 2)
When using the editing functions of Banff, a useful first step is to analyze the relationships which characterize an acceptable record. These relationships, referred to as edits, are of great importance to the proper performance of each procedure and to the quality of the data which are produced by Banff. Possible sources of these relationships are analysis of the questionnaire, analysis of existing data and subject matter knowledge. Section **2.1 Edit Specification** describes valid Banff edits, the formation of edit groups and the formation of data groups. Section 2 also describes several functions (**2.2 Check Edits, 2.3 Generate Extremal Points** and **2.4 Generate Implied Edits**) which help the user to understand the specified conditions and to ensure that the group of edits accurately represents the constraints which should be imposed on the data. In this step it is the edits themselves which are analyzed. The user does not specify the action to be taken when a record does not obey the specified relationships. In fact, respondent data are not used in these procedures.

PROC EDITSTATS – Edit Summary Statistics Tables (Section 3)
If historical or preliminary data are available, a preview of failure rates may be obtained by producing the Edit Summary Statistics Tables described in Section 3. This is the first procedure in which there is an interaction between the linear edits and actual respondent data. The five Edit Summary Statistics Tables give counts of edit and record status (pass, miss or fail) by variable and by individual edit as well as the distribution of records which pass, miss or fail a given number of edits. No information concerning which fields must be changed is given in this procedure.

The information supplied by the Edit Summary Statistics Tables may be used to fine-tune the edits or to estimate how many records will fail during the execution of the Error Localization procedure (and thus to estimate the cost of associated computer runs). This procedure may also be run after some or all of the imputation has been done in order to assess the effect of the imputation process.

PROC OUTLIER – Outlier Detection (Section 4)
This procedure offers two methods to identify outlying observations, one described by Hidiroglou and Berthelot (1986) and the Sigma-Gap method developed at Statistics Canada in the 1990s. Values of selected variables are compared across records rather than comparing fields within each individual record, as is done with the linear edit rules. The identification of outlying observations may also be based on the change in a field from a previous period, if historical data are available, or on the difference between the field of interest and an auxiliary field, if reliable auxiliary data are available.

PROC ERRORLOC – Error Localization (Section 5)
When edit rules have been finalized and the respondent data are available, the Error Localization procedure may be run. Section 5 describes how this procedure applies a group of edits to each individual record. In the case of a record which does not satisfy the edits, Error Localization detects which fields must be changed so that the record may be made to satisfy the edits while replacing the smallest possible (weighted) number of original fields. This procedure identifies the fields requiring imputation but does not perform any imputation.

Schiopu-Kratina and Kovar (1989) give a detailed description of the use of Chernikova's algorithm in the solution of the Error Localization problem.

PROC DETERMINISTIC – Deterministic Imputation (Section 6)
Banff offers four types of imputation. Deterministic Imputation, described in Section 6, identifies the cases in which there is only one possible value which will allow the record to pass the original edits. Imputation is performed by this procedure in such cases.

PROC DONORIMPUTATION – Donor Imputation (Section 7)
Section 7 describes Donor Imputation, a method which imputes all required fields of an individual record by transferring the corresponding values from the nearest neighbour record. Records imputed by donor imputation are guaranteed to pass the user-specified post-imputation edits, which may or may not be the same as the original edits. Section 7.1 Prepare for Donor Imputation describes the parameters and criteria which must be specified to provide the desired imputation results. The identification of the nearest neighbour record is based on fields which may be chosen by both the user and the system. Section 7.2 Find Matching Fields describes this process. In order to remove the effect of scale, all the matching fields are transformed to a (0,1) range, as described in Section 7.3 Transform Matching Fields. Section 7.4 Perform Donor Imputation describes the method used by Banff to search for the nearest neighbour record.

PROC ESTIMATOR – Imputation Estimators (Section 8)
Imputation Estimators, described in Section 8, provide imputation for individual fields using a variety of estimators. These estimators can be either mathematical expressions or derived from linear regression models. Banff provides 20 estimator algorithms which are hard-coded directly into the system, and the user also has the option of defining their own algorithms. Several algorithms may be specified and applied sequentially so that back up procedures are available if the preferred algorithm cannot be used in some individual cases.

PROC PRORATE – Prorating (Section 9)
Section 9 describes the Prorating procedure, which is able to analyze and adjust each record so that the components of a sum add to a specific total. The user supplies edit rules, weights and field status information which the process takes into account. A rounding mechanism ensures that all the final data produced by Prorating conforms to the number of decimal places required by the user.

PROC MASSIMPUTATION – Mass Imputation (Section 10)
Section 10 describes the Mass Imputation procedure. For operational reasons, in some surveys, detailed information is collected only for a subsample (or second phase sample) of units selected randomly from a large first phase sample. Classical estimates based on the subsample require the derivation of subsampling weights. The derivation of such weights can be quite complex. The Mass Imputation procedure applies an alternative technique which creates a complete rectangular file for all of the first phase sample units by donor imputing the missing information for the non-sampled units, after the editing and imputation has been completed for the second phase sample units.

Updates to this Document
The major functions of Banff have been developed and are available for application in a production environment. However, as there are ongoing improvements and fine-tuning, the corresponding section(s) of this document will be updated. More recent versions of any section may be obtained from:

Generalized System Section
International Cooperation and Corporate Statistical Methods Division
Statistics Canada

Questions or comments on this document or on any aspect of Banff are also welcome. If you need assistance, please contact the support team at: statcan.banff-banff.statcan@canada.ca.

2. PROC VERIFYEDITS – EDIT SPECIFICATION AND ANALYSIS

2.1 EDIT SPECIFICATION

Purpose

In the Verifyedits procedure (Proc Verifyedits), the user specifies the relationships, or edits, which determine if a record is acceptable. The user may specify the edits in a form which defines either a "pass" or a "fail" condition or may mix the two methods. Individual edits are assigned to one or more groups of edits which are analyzed in later procedures and are used in processing throughout the rest of Banff.

Edits in Banff

Banff edits data which are numeric and continuous. The user specifies edit rules which must be satisfied by the responses in the fields of each individual record. These edits must be linear equalities or inequalities of the form:

$$a_1x_1 + a_2x_2 + ... + a_nx_n = b \quad \text{or}$$
$$a_1x_1 + a_2x_2 + ... + a_nx_n \le b$$

where x_1 to x_n are the n responses supplied to the survey by a sampled unit, and a_1 to a_n and b are constants specified by the user. If the user specifies the rejectnegative option, edits of the form $x_i \ge 0$ are automatically added to the group of edits specified by the user for each variable that is being edited. Therefore, data which are negative or missing always fail at least one of these positivity edits. The user must indicate whether each edit describes a pass or a fail condition, but does not indicate the action to be taken when a record fails the edit. The following are examples of edits as they would be specified by the user.

$$\text{PASS:} \quad \text{FOOD} + \text{BEVERAGES} = \text{TOTAL\_SALES}$$
$$\text{PASS:} \quad \text{MILK\_LITRES} \ge 15 \ * \ \text{MILK\_COWS}$$
$$\text{FAIL:} \quad x_1 + x_2 + x_3 < 10$$

The first edit requires that the values in the variables FOOD and BEVERAGES add to the value in the variable TOTAL_SALES for each record being edited. The second edit requires that the value of MILK_LITRES be greater than or equal to 15 times the value of MILK_COWS. The third edit would cause a record to fail if the sum of x_1, x_2 and x_3 were less than 10.

Canonical Form

In Banff, all edits are stored internally in **canonical form**, that is, as a pass condition with all variables sorted alphabetically and appearing to the left of either an "=" or a "≤" operator. The user may enter edits in any form, but the system works only with canonical form and prints edits in canonical form whenever a procedure includes a list of edits as part of its output. A function which converts edits into canonical form is provided internally by Banff and is executed whenever edits are input or modified. The following edits show some examples of valid and invalid edits and the canonical form to which the edits are converted.

ORIGINAL EDITS			CANONICAL FORM		
PASS:	$A > B + 3$	(1)	PASS:	$-A + B \leq -3$	(1)
PASS:	$C = D$	(2)	PASS:	$C - D = 0$	(2)
PASS:	$Z < A$	(3)	PASS:	$-A + Z \leq 0$	(3)
PASS:	$M \neq N$	(4)	INVALID		
FAIL:	$A > B + 3$	(5)	PASS:	$A - B \leq 3$	(5)
FAIL:	$C = D$	(6)	INVALID		
FAIL:	$Z \leq A$	(7)	PASS:	$A - Z \leq 0$	(7)
FAIL:	$N \neq M$	(8)	PASS:	$-M + N = 0$	(8)

Note that strict inequalities are not allowed in the canonical form of pass condition edits and are automatically replaced by "≤", regardless of what was specified by the user. For example, in edit (3) the "<" entered by the user has been replaced be a "≤". Edits expressed as a fail condition have "≤" replaced by "<". For example, the canonical form of edit (7) is equivalent to FAIL: Z < A, even though edit (7) was specified as FAIL: Z ≤ A. It should also be noted that the "≠" is not a valid relational operator for a pass condition and the "=" is not valid in a fail condition. These restrictions exist because a set of edits must describe a convex region which contains its own boundary. This is discussed further in Section 2.2.

Non-linear Edits

The requirement that the edits be linear is due to the linear programming techniques which are used in several procedures of Banff. However, some types of non-linear edits may be linearized, as is seen in the following two examples.

$\mathbf{x_1 x_2 = x_3}$ Create new variables, $y_I = \log x_I$ for I = 1, 2, 3 and the edit becomes $y_1 + y_2 = y_3$. The variables x_1, x_2 and x_3 must be replaced by y_1, y_2 and y_3, otherwise inconsistencies would result if any of x_1, x_2, x_3, y_1, y_2 and y_3 were imputed. All edits which previously involved x_1, x_2 or x_3 must be replaced by edits in the y_1, y_2 and y_3. This may be impossible if the edit in question involves another variable, such as x_4.

If EMP > 0 then SALARIES > 0 This conditional edit may be re-expressed in the form SALARIES ≥ .000001 EMP - a slightly stronger restriction than the original. This ensures that if EMP is greater than zero, SALARIES must also be greater than zero. If EMP is equal to zero, there are no restrictions on SALARIES. Another possibility would be to put records with EMP equal to zero in a group and process them separately from the records whose EMP is greater than zero.

Even though it is possible to linearize many non-linear edits, a survey which is considering processing in Banff should not simply transform all existing edits into linear form. Instead, the user should take the opportunity to examine the relationships which exist, or should exist, among the survey variables and should base the edits on this, and other, information.

Edits for Processing Negative Values

Designing edits for processing negative values can present unexpected challenges and requires special considerations that may produce unexpected results when not taken into account. For more information and examples please see the document “Specifying Edits for Processing Negative Values with Banff” (Banff Support Team, 2006).

Edit Groups and Data Groups

In GEIS, the Error Localization module which identifies the fields requiring imputation can process up to about 40 variables at a time. In larger surveys, it is usually necessary to divide the variables into logical groups of about this size. The set of variables and the edits which define the relationships to be satisfied by the variables form an **edit group**. In Banff, this restriction on the number of variables does not exist. Still, it is sometimes useful to organize the edits into edit groups for the sake of simplicity and efficiency. The user can then apply one edit group for each execution of the Error Localization procedure.

The edits are later applied together to each record of a group of records. The simultaneous application of a group of edits is fundamental to Banff and is an important benefit to the user because it avoids the inconsistencies which can develop when edits are applied consecutively. Such inconsistencies occur when a record is altered so that it passes an edit, then the "corrected" record fails a subsequent edit and is changed in a way such that the first edit is no longer satisfied. This cannot happen when all edits are applied together.

Data group refers to a set of records to which an edit group is applied. Data groups could be defined according to the sampling strata, provincial boundaries, industrial classification, or any other way. In Banff, multiple data groups can be processed in a single execution of a Banff SAS procedure by using the BY statement.

Edits often change slightly from one geographical or industrial area to another, and separate edit groups must be created whenever a different set of edits is required. For example, suppose a survey has collected six variables, x_1, x_2, …, x_6, from respondents in several industries. Edit groups similar to the following might be created.

Edit Group: SERVICES		Edit Group: RETAIL		Edit Group: CONSTRUCTION		
$x_1 + x_2 = x_3$	(1)	$x_1 + x_2 = x_3$	(1)	$x_1 + x_2 = x_3$	(1)	…
$x_4 + x_5 = x_6$	(2)	$x_4 + x_5 = x_6$	(2)	$x_4 + x_5 = x_6$	(2)	…
$x_3 \geq 75 x_6$	(3)	$x_3 \geq 80\ x_6$	(4)	$x_3 \geq 100\ x_6$	(5)	…

Edits (1) and (2) are used in all industries and are assigned to all edit groups. The variable x_3 must be greater than or equal to a constant times x_6, but this constant varies from industry to industry. Therefore, the user defines edits (3), (4), (5) etc. separately and assigns one of them to each of the edit groups. Individual data groups are created with records from each industry, or possibly with both an industrial and geographical breakdown.

Data groups do not have to be the same for each edit group and vice versa. Consider a survey with data from four geographical regions and with variables pertaining to land, livestock and crops. There are too many variables to be edited together in one group so the land, livestock and crops variables are edited in three different groups.

	Land Variables $x_1 \; x_2 \ldots x_{20}$	Livestock Variables $x_{21} \; x_{22} \ldots x_{60}$	Crops Variables $x_{61} \; x_{62} \ldots x_{90}$
Region 1 Data Group	Edit Group: Land_1	Edit Group: Livestock_1	Edit Group: Crops_1234
Region 2 Data Group	Edit Group: Land_2	Edit Group: Livestock_234	
Region 3 Data Group	Edit Group: Land_3		
Region 4 Data Group	Edit Group: Land_4		

Figure 2.1 Example of data groups and edit groups in an agriculture survey

One possible structure of data groups and edit groups is illustrated in Figure 2.1. The land variables have slightly different edits according to the geographical region, so there are four land edit groups defined and four data groups. The livestock variables from region 1 records have their own set of edits. The livestock variables of records from the other regions are subject to another set of edits, so a data group containing records from regions 2, 3 and 4 is defined and the edit group Livestock_234 is used for the livestock variables in that data group. The crops variables in all four regions are subject to exactly the same edits so another data group is defined for all records and the Crops_1234 edit group is used to edit the crops variables for all records.

Once a group of edits has been created, the user has several tools available to ensure that the group of edits accurately represents the conditions which should be imposed on the data. These facilities are described in the next three sections:

- 2.2 Check Edits
- 2.3 Generate Extremal Points
- 2.4 Generate Implied Edits

2.2 CHECK EDITS

Purpose
This function in the Verifyedits procedure checks that the edits in a group of edits are consistent with each other and, if so, identifies any redundant edits, deterministic variables or hidden equalities. Once these features are identified, the minimal set of edits may be determined. No respondent data are used in this procedure; it is the edits themselves which are analyzed.

Definitions
A group of edits involving n variables defines a region, called a **feasible region** or an **acceptance region**, in the n-dimensional space. When the original survey values are substituted in the equalities and inequalities which make up the group of edits, records which satisfy all the edits fall inside the feasible region. Records which do not satisfy the edits fall outside the region. The feasible region used by Banff must be convex and must include its boundaries. A pass condition specified as a "<" is automatically changed to a "≤" in the canonical form so that the boundary is included in the feasible region. A pass condition cannot be expressed as a "≠" relation because the feasible region would be two separate regions on either side of the equality relation and would not be convex.

A feasible region may be described by any one of an infinite number of different sets of edits. A **minimal set of edits** contains the smallest number of edits necessary to define a certain feasible region. The Verifyedits procedure provides the user with the information necessary to create a minimal set of edits for the desired feasible region. This minimal set should be used in further processing to improve efficiency.

The Check Edits function of the Verifyedits procedure also provides information which should give the user a better understanding of the feasible region defined by a group of edits. In Banff the identification of fields requiring imputation, as well as the success of some types of imputation, depends on the group of edits so it is very important that the edits accurately represent the conditions that the user wishes to impose on the data.

In this procedure, a group of edits is checked for consistency, redundancy, determinacy and hidden equalities. These terms are defined and illustrated in this section, but the details of the actual methods which are used in Banff have not been included here. Many of these methods involve maximizing and/or minimizing a given edit subject to the other constraints. This is done using the Revised Simplex Method algorithm, (Product Form Inverse). The Given's Transformation is used to convert a matrix to triangular form when checking for hidden equalities. The reader who is interested in a more complete description of the methods used in the Verifyedits procedure of Banff should refer to Giles (1989).

Consistent and Inconsistent Groups of Edits

A **consistent** group of edits defines a non-empty feasible region. An **inconsistent** group of edits can never be satisfied by any data record because the group contains edits which contradict each other. Figures 2.2 and 2.3 illustrate consistent and inconsistent groups of edits.

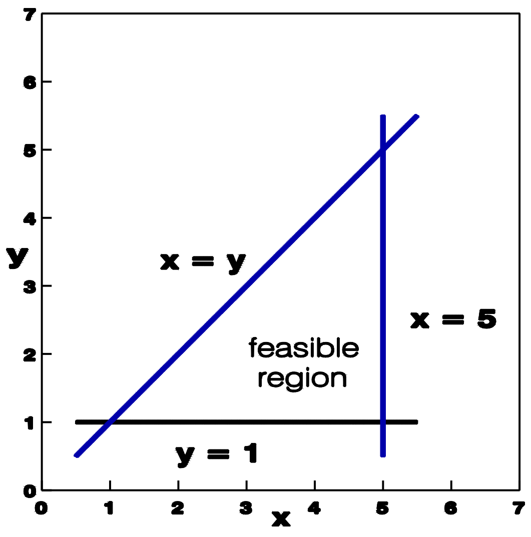


Fig. 2.2 A consistent group of edits

$x \geq y$ (1)
$x \leq 5$ (2)
$y \geq 1$ (3)

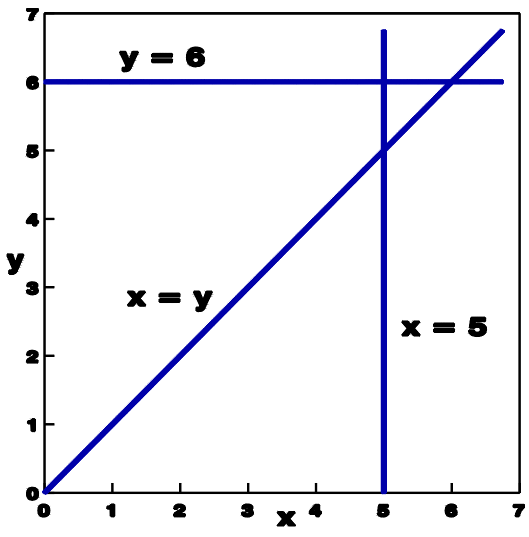


Fig. 2.3 An inconsistent group of edits

$x \geq y$ (1)
$x \leq 5$ (2)
$y \geq 6$ (4)

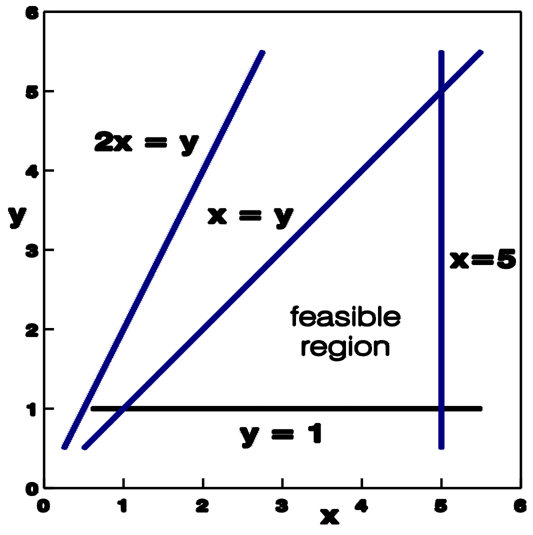
In Figure 2.2, the group of edits is consistent and defines a feasible region. In Figure 2.3, the group of edits is inconsistent. No feasible region can be identified because there is no point which can satisfy all three edits at the same time, although there exist regions which would satisfy each of the three possible pairs of edits. In the above example it is possible to use a graph to identify an inappropriate or incorrectly specified edit, but in a typical application the group of edits involves many variables and it is not possible to inspect the feasible region graphically.

When Banff detects an inconsistent group of edits, it provides the user with a sub-group of edits that must be removed so that the resulting group of edits is consistent.

Any inconsistent set of edits must be re-examined very carefully because the user has specified constraints which are contradictory. The first step is to verify that all the edits have been entered correctly since even a slight discrepancy can result in an edit which is very different from the one which was intended. If all edits have been specified correctly, the user may whish to remove edits singly or in combination in order to identify various consistent sets. The goal is not to remove as few edits as possible, but to determine a consistent set of edits which represents the relationships the user wishes to impose upon the data.

Redundant Edits: Tight but Non-restrictive Edits

The Check Edits function also identifies edits which are **redundant**. This means that the edit does not form part of the boundary of the feasible region and therefore imposes no restrictions on the acceptable values. A special case of a redundant edit occurs with edits which are **tight but non-restrictive.** These edits touch the boundary of the feasible region at a single point, but do not restrict the allowable values, given the other edits. Neither redundant edits nor tight but non-restrictive edits belong to the minimal set of edits needed to describe a feasible region. These edits should be omitted from further processing to improve efficiency. Examples of redundant and tight but non-restrictive edits are found in Figures 2.4 and 2.5.



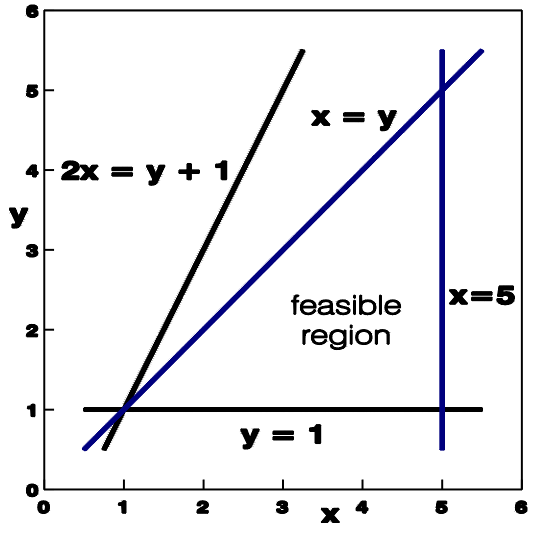


Fig. 2.4 Edit (4) is redundant

$x \geq y$	(1)	
$x \leq 5$	(2)	
$y \geq 1$	(3)	
$2x \geq y$	(4)	(redundant)

Fig. 2.5 Edit (5) is tight, non-restrictive

$x \geq y$	(1)
$x \leq 5$	(2)
$y \geq 1$	(3)
$2x \geq y + 1$	(5) (tight, non-restrictive)

In Figure 2.4, edit (4) does not form part of the boundary of the feasible region and therefore is redundant. Any record which satisfies edits (1), (2) and (3) will automatically satisfy edit (4). In Figure 2.5, edit (5) touches the feasible region at one point, but still does not restrict the acceptable values. Any record which satisfies edits (1), (2) and (3) will automatically satisfy edit (5). In both cases, the user should study the specified edits and decide if the redundant edits and the tight edits should be eliminated or if it is the other edits which are too restrictive. For these examples, the user could specify a new group of edits consisting of edits (1), (2) and (3) or could choose a new group consisting of edits (2), (3) and (4) for Figure 2.4 and edits (2), (3) and (5) for Figure 2.5. In any case, the new group would be used for all further processing.

Hidden Equalities

A group of edits may contain inequalities which, when taken with the restrictions imposed by the other edits, imply an equality. These edits, referred to as **hidden equalities**, may not be obvious when several variables are involved. The presence of hidden equalities usually indicates that the edits are more restrictive than the user had realized. When Proc Verifyedits identifies hidden equalities, the edits should be reexamined to determine if the hidden equality should be retained or if one or more of the edits has been incorrectly specified. Consider the following group of edits.

$$\begin{array}{rcl} x_1 + x_2 \qquad + x_4 + x_5 & \le 4 & (1) \\ x_2 + x_3 - x_4 + x_5 & \ge 2 & (2) \\ x_1 \qquad\qquad + x_4 \qquad & = 3 & (3) \\ x_3 - x_4 \qquad & = 1 & (4) \end{array}$$

In this example, the Verifyedits procedure would detect that edits (1) and (2) form a hidden equality and would identify edit (2) as a **redundant hidden equality**. The hidden equalities in this group of edits may be seen more clearly after performing simple algebraic substitutions.

Substitute edit (3) in edit (1) to obtain: $x_2 + x_5 \le 1$
Substitute edit (4) in edit (2) to obtain: $x_2 + x_5 \ge 1$

If the user decides that the edits are correctly specified, then both $x_2 + x_5 \le 1$ and $x_2 + x_5 \ge 1$ must be true, which of course implies that $x_2 + x_5 = 1$. Edits (1) and (2) should be removed and replaced by $x_2 + x_5 = 1$, or by a revised version of edit (1) or (2) restated as an equality. Any of these edits, along with the existing edits (3) and (4), define the same feasible region as the original edits. These three equivalent groups of edits are listed below.

Equivalent Groups of Edits

$$\begin{array}{rcl} x_2 \qquad + x_5 & = 1 & (5) \\ x_1 \qquad + x_4 \qquad & = 3 & (3) \\ x_3 - x_4 \qquad & = 1 & (4) \end{array} \qquad \begin{array}{rcl} x_1 + x_2 \qquad + x_4 + x_5 & = 4 & (1a) \\ x_1 \qquad\qquad + x_4 \qquad & = 3 & (3) \\ x_3 - x_4 \qquad & = 1 & (4) \end{array} \qquad \begin{array}{rcl} x_2 + x_3 - x_4 + x_5 & = 2 & (2a) \\ x_1 \qquad\qquad + x_4 \qquad & = 3 & (3) \\ x_3 - x_4 \qquad & = 1 & (4) \end{array}$$

On the other hand, the user may decide that the edits were not correctly specified and that, say, the "≤" in edit (1) should be a "≥". If this change is made, the Verifyedits procedure would identify edit (2) as tight but non-restrictive and would find no hidden equalities among the edits.

Upper and Lower Bounds; Determinacy

The Check Edits function also produces **upper and lower bounds** for each variable. These bounds are the maximum and minimum values that a variable could attain while remaining inside the feasible region described by the edits. The bounds do not represent actual data values since no respondent data are used in this procedure. The user should examine the bounds and consider adding or removing edits if the bounds do not seem to be reasonable. As an illustration of bounds, consider the following group of edits and the bounds which would be output by the Verifyedits procedure.

$$x_1 + x_2 + x_4 = 10 \quad (1) \qquad x_1 \geq 0 \quad (4)$$
$$x_1 + x_2 = 6 \quad (2) \qquad x_2 \geq 0 \quad (5)$$
$$x_3 + x_4 \geq 8 \quad (3) \qquad x_3 \geq 0 \quad (6)$$
$$x_4 \geq 0 \quad (7)$$

Variable	Lower Bound	Upper Bound	
x_1	0	6	
x_2	0	6	
x_3	4	*****	UNBOUNDED
x_4	4	4	DETERMINANT

Both x_1 and x_2 have minimum values of 0 and maximum values of 6. The variable x_3 cannot take on valid values less than 4, but has no limit on its maximum value. The variable x_4 has a lower bound of 4 and an upper bound of 4 and must therefore be equal to 4 at all times. This variable shows **determinacy**, that is, can take on only one possible value. Determinacy of one variable may be caused by an incorrectly specified edit anywhere in the group, so the user should review the entire group of edits to decide if the variable identified as deterministic should be limited to one possible value in the group of records being processed.

The upper and lower bounds give no indication of the fact that valid values in one field of a particular record may depend on the values in its other fields. In the above example, the edits do not allow both x_1 and x_2 to take on their maximum or minimum values at the same time. In fact, when one attains its maximum, the other is at its minimum, as is shown by their relationship in edit (2).

2.3 GENERATE EXTREMAL POINTS

Purpose
This function of the Verifyedits procedure generates all the extremal points, or vertices, of the feasible region described by a group of edits. These points represent the most extreme data records which would be acceptable and may therefore give the user a better understanding of the shape of the feasible region which is being specified.

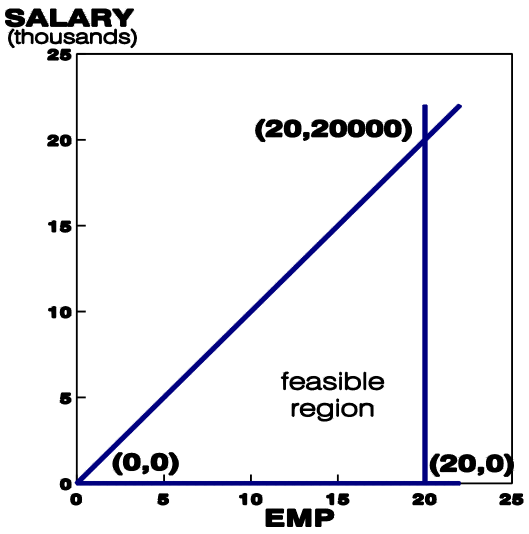
Description of the Method
Each **extremal point** is a vertex of the feasible region and may be represented by its coordinates in the n-dimensional space. Geometrically, extremal points occur at the intersection of n edits, where n is the number of variables. In the survey context, extremal points may be thought of as the most extreme that a respondent record is allowed to be while still being acceptable.

A complete description of the use of Chernikova's algorithm to generate all the extremal points of the feasible region may be found in Schiopu-Kratina and Kovar (1989). A matrix is constructed from the user-specified edits and the implied positivity edits. The matrix is transformed during each of several iterations. In each iteration, columns are either retained or are taken in linear combination with others to produce new columns. The iterations continue until all extremal points have been produced or until all extremal points with cardinality less than or equal to a user-specified limit have been produced. Here, **cardinality** refers to the number of non-zero coordinates of an extremal point. For example, the point (5, 0, 10) has a cardinality of two. The points easiest to interpret are often those which have many zero values because, in a sense, these variables are eliminated from consideration. So the user may choose to restrict the generation of extremal points to those whose number of non-zero coordinates is less than or equal to a user-specified limit. Limiting the cardinality may also reduce the execution time.

There is a theoretical upper bound to the number of extremal points associated with an edit group, but the actual number of extremal points is usually much smaller than this bound. Both the actual number and the theoretical bound are often very large, even for moderately sized edit groups. More details may be found in Giles (1989).

Example of Extremal Points

Consider the following example of edits for a group of small businesses in a certain industrial sector.



ORIGINAL EDITS

EMP $\leq$ 20

SALARY $\leq$ 1000 EMP

EMP $\geq$ 0 (implied positivity edit)

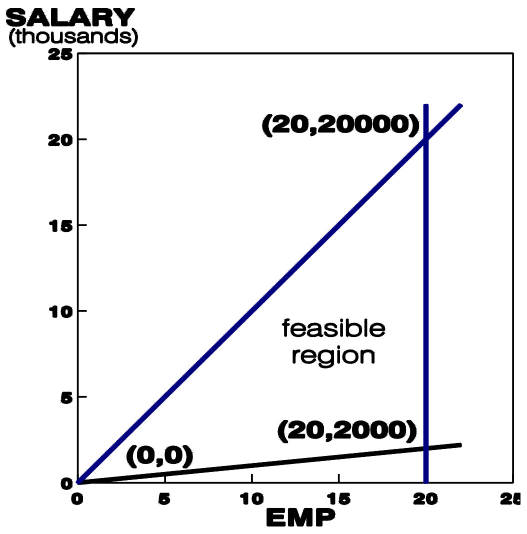
SALARY $\geq$ 0 (implied positivity edit)

EXTREMAL POINTS

(0 , 0)	cardinality of 0
(20 , 20000)	cardinality of 2

Figure 2.6 Extremal Points of the Original Group of Edits

Examination of the three extremal points shown in Figure 2.6 causes the user to question the point (20,0). Could a business have 20 employees and no salaries? If the user decides this is not acceptable, another edit could be added to impose a minimum average salary. A revised group of edits would be created and put through the Generate Canonical Form, Check Edits and Extremal Points functions of Proc Verifyedits. With the new edit group shown in Figure 2.7, the point (20,0) would be outside the feasible region and a respondent record with these values would be identified as unacceptable in a later procedure.



REVISED EDITS

EMP $\leq$ 20

SALARY $\leq$ 1000 EMP

SALARY $\geq$ 100 EMP

EMP $\geq$ 0

SALARY $\geq$ 0

EXTREMAL POINTS

(0 , 0)

(20 , 20000)

(20 , 2000)

Figure 2.7 Extremal Points of the Revised Group of Edits

2.4 GENERATE IMPLIED EDITS

Purpose
This function of the Verifyedits procedure generates additional edits which are implied by a group of edits. These implied edits may be examined to ensure that all relationships implied by the group of edits are acceptable to the user.

Description of the Method
The concept of implied edits was introduced in Fellegi and Holt (1976). In Banff, an **implied edit** is the result of a linear combination of *k* edits in which at least (*k*-1) variables have been eliminated. Implied edits are always redundant and therefore should not be added to the minimal set of edits. However, implied edits may involve fewer variables than the edits specified by the user and so may bring to light relationships of which the user was unaware.

A complete description of the use of Chernikova's algorithm to generate the implied edits may be found in Schiopu-Kratina and Kovar (1989). A matrix is constructed from the user-specified edits and the positivity edits. The transpose of this matrix undergoes successive transformations as each of its rows is processed. In each iteration, columns are taken in linear combination with others to produce additional columns. These iterations continue until all rows have been processed or until the number of implied edits produced exceeds a user-specified limit. The user may wish to apply this limit because a large number of implied edits may be generated from a relatively small number of edits, especially if an equality edit is present.

Example of Implied Edits
Consider the following group of two user-specified edits and three positivity edits added by the system.

$$\begin{aligned}
x_1 - 2x_2 + 3x_3 &\le 10 \quad (1) \\
x_1 + x_2 + x_3 &\le 5 \quad (2) \\
-x_1 &\le 0 \quad (3) \qquad \text{(positivity edit)} \\
-x_2 &\le 0 \quad (4) \qquad \text{(positivity edit)} \\
-x_3 &\le 0 \quad (5) \qquad \text{(positivity edit)}
\end{aligned}$$

Edits (1) and (2) may be combined to produce implied edit (6) which may itself be combined with edit (3) to produce edit (7).

$$\begin{array}{llll}
x_1 - 2x_2 + 3x_3 \le 10 & (1) & \qquad 3x_1 \quad + 5x_3 \le 20 & (6) \\
\underline{2x_1 + 2x_2 + 2x_3 \le 10} & \underline{(2)\times 2} & \qquad \underline{-3x_1 \qquad\quad \le \ 0} & \underline{(3)\times 3} \\
3x_1 \qquad + 5x_3 \le 20 & (6) & \qquad\qquad 5x_3 \le 20 & (7)
\end{array}$$

It is common to have a very large number of implied edits. This small edit group of two inequalities and three positivity edits generates nine implied edits. The complete list of implied edits is shown as it would appear in the output of the procedure. Note that the leading coefficient of the first variable of each equation is always either +1 or -1, so, for example, edit (7) appears as $x_3 \le 4$ instead of $5x_3 \le 20$ as it was derived above.

List of Implied Edits

$$x_2 + x_3 \le 5$$
$$-x_2 + 1.5x_3 \le 5$$
$$x_1 + 1.66667x_3 \le 6.66667 \quad (6)$$
$$x_1 + x_3 \le 5$$
$$x_3 \le 4 \quad (7)$$
$$x_1 + x_2 \le 5$$
$$x_1 - 2x_2 \le 10$$
$$x_2 \le 5$$
$$x_1 \le 5$$

In examining the implied edits, the user may find that there are relationships between variables which are unacceptable. If this is the case, the original edits should be examined and modified and the new set of edits should be put through the Generate Canonical Form, Check Edits, Generate Extremal Points and Generate Implied Edits functions of Proc Verifyedits.

3. PROC EDITSTATS – EDIT SUMMARY STATISTICS TABLES

Purpose

This procedure applies a group of edits to respondent data records and determines if each record passes, misses or fails each edit. The status codes created during this process are used only in this procedure and are not passed on to other procedures. Five tables summarizing the status codes are produced and may be used to fine-tune the group of edits, to estimate the resources required for later procedures or to evaluate the effects of imputation. When a record fails an edit, there is no attempt to identify how to change the record so that it would pass.

Description of the Method

For an edit group of *m* positivity edits and *n* user-specified edits with no redundant edits, a total of $m + n + 1$ status codes is assigned to each record. Redundant edits should be removed from the edit group by this stage, but if they are still included, the number of status codes to be generated will be reduced because redundant edits do not appear in the tables. The following procedure is performed independently for each data record.

- One status code is assigned to the data record for each edit, including the positivity edits. There are $m + n$ of these codes in total. The status is:
- PASS, if the record passes the edit,
- MISS, if the record has one or more missing fields involved in the edit, or
- FAIL, if the record fails the edit because of one or more non-missing values.

- An overall record status is derived from the $m + n$ status codes which have been assigned to the record based on the results of the application of the individual edits to that record. The status is:
- PASS, if each edit status is PASS, i.e., if the record has passed all the edits in the group,
- MISS, if one or more edit status is MISS and no edit status is FAIL, i.e., each edit which the record did not pass involved missing fields, or
- FAIL, if one or more edit status is FAIL, i.e., if the record failed edits because of non-missing values and possibly had failures due to missing values.

Example of Creation of Edit Status Codes

Consider the following group of edits and respondent records.

Positivity Edits		User Edits	
$x_1 \geq 0$	(1)	$x_1 + 1 \geq x_2$	(4)
$x_2 \geq 0$	(2)	$x_1 \leq 5$	(5)
$x_3 \geq 0$	(3)	$x_2 \geq x_3$	(6)
		$x_1 + x_2 + x_3 \leq 9$	(7)

There are three positivity edits and four user-specified edits, so there are 3 + 4 + 1 = 8 status codes to be assigned to each record. These codes are marked as Edit Status (1) to (7) and Overall Status in the following table.

	Respondent Records			Edit Status							Overall
	x_1	**x_2**	**x_3**	**(1)**	**(2)**	**(3)**	**(4)**	**(5)**	**(6)**	**(7)**	**Status**
record 1	**4**	**3**	**2**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**
record 2	**4**	**3**	**missing**	**P**	**P**	**M**	**P**	**P**	**M**	**M**	**M**
record 3	**6**	**3**	**2**	**P**	**P**	**P**	**P**	**F**	**P**	**F**	**F**
record 4	**6**	**3**	**missing**	**P**	**P**	**M**	**P**	**F**	**M**	**M**	**F**

Record 1 passes the edits and therefore has a PASS status for all seven edits and for the overall status. Record 2 passes all the edits except those involving the variable x_3 which is missing. Therefore edits (1), (2), (4) and (5) have PASS status and edits (3), (6) and (7) have MISS status. The overall status for record 2 is also MISS. Record 3 has no missing values, but fails edits (5) and (7). All edits have a PASS status except edits (5) and (7) which have FAIL status, so the overall status is FAIL. Record 4 fails edit (5) due to "incorrect" non-missing values and fails other edits due to missing values. Edits (1), (2) and (4) have PASS status, edit (5) has FAIL status and edits (3), (6) and (7) have MISS status. The overall status of record 4 is FAIL because FAIL takes precedence over the MISS status code when deriving the overall record status code.

Example of the Tables

A short description of the five Edit Summary Statistics Tables is given here, along with the tables which would be produced for the edits and data records used in the above example of the creation of status codes. The user should keep in mind that tables 1-1 and 1-2 are based on the edit status codes while table 1-3 is based on the overall record status codes. Table 2-1 is based on edit status codes which have been assigned to each field in the record according to rules described below. Table 2-2 is based on the record status codes which have been assigned to each field of the record according to rules described below.

TABLE 1-1

COUNTS OF RECORDS THAT PASSED, MISSED AND FAILED FOR EACH EDIT

EDITID	OBS_ PASSED	OBS_ MISSED	OBS_ FAILED
POSITIVITY EDIT x_1	4	0	0
POSITIVITY EDIT x_2	4	0	0
POSITIVITY EDIT x_3	2	2	0
EDIT (4)	4	0	0
EDIT (5)	2	0	2
EDIT (6)	2	2	0
EDIT (7)	1	2	1

Table 1-1 gives the number of records which passed, missed or failed each edit. The table is based on counts of the PASS, MISS and FAIL edit status codes which occur for each individual edit. For any given edit, each record is counted as passed, missed or failed, so the total in each row should be the same as the total number of records.

For example, for edit (1), the positivity edit on the variable x_1, all four records passed so all four are counted in the " OBS_PASSED" column and no records are counted in the " OBS_MISSED" or " OBS_FAILED" columns. On the other hand, only record 1 passed edit (7), while records 2 and 4 had edit status MISS and record 3 failed. The last line of Table 1-1 counts one under " OBS_PASSED", two under " OBS_MISSED" and one under " OBS_FAILED".

From Table 1-1, the user may determine if records tend to fail or miss some edits more often than others. This could indicate that the edits are too restrictive or that the data are of poor quality.

TABLE 1-2

DISTRIBUTION OF RECORDS THAT PASSED, MISSED AND FAILED K EDITS

NUMBER OF EDITS (K_EDITS)	OBS_ PASSED	OBS_ MISSED	OBS_ FAILED
0	0	2	2
1	0	0	1
2	0	0	1
3	1	2	0
4	1	0	0
5	1	0	0
6	0	0	0
7	1	0	0
TOTAL RECORDS	4	4	4

Table 1-2 gives the distribution of records which pass, miss or fail a given number of edits. The table is based on the edit status codes and gives the number of records for which each status code occurs zero times, once, twice, etc. Each record is counted once in each column (in the row corresponding to the number of edits it failed, missed or passed) so that the total of each column is equal to the number of records.

In this example, the first row of Table 1-2 indicates that there were no records which passed zero edits, while two records missed zero edits and two records failed zero edits. Moving down the table to the row with "3" in the left-hand column, one can see that one record passed three edits, two records missed three edits and no records failed three edits. The line beginning with "7" says that one record passed all seven edits, no records missed seven edits and no records failed seven edits.

From Table 1-2 the user may determine if the data being analyzed consist of a moderate number of records which pass all edits and a moderate number of records which fail a few edits, or if the data consist of a large group of records which pass all the edits and a much smaller group of records which fail most or all of the edits.

TABLE 1-3

OVERALL COUNTS OF RECORDS THAT PASSED, MISSED AND FAILED

OBS_ PASSED	OBS_ MISSED	OBS_ FAILED	OBS_ TOTAL
1	1	2	4

Table 1-3 gives the number of records with PASS, MISS or FAIL overall record status. Each record is counted once so the single row of this table should add to the total number of records.

In the example being used here, one record had a PASS record status code, one had a MISS and two had record status codes of FAIL. Note that records which have a FAIL record status may have one or more MISS edit status codes as well as at least one FAIL edit status code.

The sum of the OBS_MISSED and OBS_FAILED cells of Table 1-3 gives a preview of the number of failures to expect when the Error Localization procedure is run.

TABLE 2-1

COUNTS OF EDIT APPLICATIONS OF STATUS PASS, MISS OR FAIL THAT INVOLVE EACH FIELD

FIELDID	EDIT_ APPLIC_ PASSED	EDIT_ APPLIC_ MISSED	EDIT_ APPLIC_ FAILED	EDIT_ APPLIC_ NOT INVOLVED	EDITS_ INVOLVED
x_1	11	2	3	12	4
x_2	11	4	1	12	4
x_3	5	6	1	16	3

Table 2-1 gives the number of times each variable was involved in an edit which passed, missed or failed. For this tabulation the PASS, MISS and FAIL edit status codes which were generated for a particular record and edit are assigned to each field which is involved in the edit. A NOT INVOLVED code is assigned if the variable did not appear in the edit. Then the occurrences of these codes are summed for each variable. For a given row of the table (i.e., a given variable), each record is counted with each edit, so the total of all cells except the last is the number of records times the number of edits. The last column gives a count of the number of edits in which the particular variable is involved.

In the example being used here, x_1 was involved in a PASS edit status code 11 times - record 1 in edits (1), (4), (5) and (7), record 2 in edits (1), (4) and (5), record 3 in edits (1) and (4) and record 4 in edits (1) and (4). The variable x_1 was involved in a MISS edit status code twice - edit (7) with records 2 and 4 - and was involved in a FAIL edit status code three times - record 3 with edits (5) and (7) and record 4 with edit (5). There are three edits in which x_1 was not involved, giving a count of 3 edits x 4 records = 12 edit applications which did not involve x_1. The last column gives the number of edits which did involve x_1. There are four records in this example, so there should be 4 x 4 = 16 edit applications which involved x_1, and this should always be equal to the sum of the first three columns.

From Table 2-1, the user may judge if some variables tend to be included in edits which fail or miss more often than others.

TABLE 2-2

COUNTS OF RECORDS OF STATUS PASS, MISS, OR FAIL FOR WHICH FIELD *j* CONTRIBUTED TO THE OVERALL RECORD STATUS

FIELDID	OBS_ PASSED	OBS_ MISSED	OBS_ FAILED	OBS_NOT_ APPLICABLE
x_1	1	1	2	0
x_2	1	1	1	1
x_3	1	1	1	1

Table 2-2 gives the number of times each variable contributed to the overall record status. For this tabulation, the PASS, MISS and FAIL overall record status codes are assigned to each field according to the following rules.

- If the overall record status is PASS then the record passed all edits, all fields must be good and all fields are assigned a PASS status for the purposes of this table.

- If the overall record status is MISS, then MISS and PASS are the only possible values for edit status. The fields involved in the edits with status MISS are assigned a MISS and the fields not involved in any edits with status MISS are assigned a NOT APPLICABLE status.

- If the overall record status is FAIL, then at least one edit status must be FAIL and one or more edit status may be MISS. The variables involved in edits with FAIL edit status are assigned a FAIL and the variables not involved in any edit with a FAIL edit status are assigned a NOT APPLICABLE status.

Each record is counted once for each field, so the row total is equal to the number of records. For example, since record 1 passed all edits, it is counted under "OBS_PASSED" for all variables. Record 2 has a MISS record status and all its fields are involved in at least one edit which had a MISS status, so record 2 is counted in the "OBS_MISSED" column for all variables, even though only x_3 is actually missing on the respondent data record. Record 3 has a FAIL record status and each of its fields is involved in at least one edit which has a FAIL edit status. Therefore, record 3 is counted in the "OBS_FAILED" column for all variables. Record 4 also has a FAIL record status. The variable x_1 is involved in edit (5) which has a FAIL edit status, so it is counted in the "OBS_FAILED" column for x_1. The remaining two fields are not involved in any edits which have a FAIL edit status, so record 4 is counted in the "OBS_NOT_APPLICABLE" column for the rows referring to x_2 and x_3.

From Table 2-2, the user may determine if some variables tend to contribute to the MISS or FAIL record status more often than others. It should be noted that this table counts all fields involved in a FAIL or MISS edit as contributing to the overall record FAIL or MISS status. The Error Localization procedure will likely identify a subset of these fields as requiring imputation and allow the other fields involved in the edit to remain as they are.

Uses of the Edit Summary Statistics Tables

This is the first procedure in which there is an interaction between the linear edits and actual respondent data; in the Verifyedits procedure, the linear edits themselves are examined with no reference to data. In the Outlier Detection procedure, which may be run either before or after the production of the Edit Summary Statistics Tables, the data are edited using a statistical edit whose parameters are based on the data themselves, not on the linear edit rules specified by the user.

The Edit Summary Statistics Tables have three major uses during edit and imputation processing. These uses and the steps at which they occur are discussed in turn.

The first major use is to assess the suitability of individual edits or of a group of edits by observing the failure rates which occur when the edits are applied to a set of data records. A high failure rate for one particular edit might indicate that the user should modify that edit by changing the constants or by altering the form of the edit. If this is done, it would then be necessary to change the edit, regenerate the canonical form and resubmit the new edit group to edit analysis. It is also possible that anomalies observed in the Edit Summary Statistics Tables do not represent problems with the group of edits itself. For example, if one variable is involved in a large percentage of the edit failures, it might indicate that a review should be done of the questionnaire definitions or of the collection procedures used in the field. Of course it might not be possible to do this during the later stages of a survey.

When the Edit Summary Statistics Tables are being run in order to fine-tune the edits, it is likely that the final survey data are not yet available. The respondent records used in this procedure may be actual data taken from a pilot study or from historical sources. It is also possible to use data which have been generated by the user outside of Banff.

The second major use of the Edit Summary Statistics Tables is to document the status of the respondent data as they enter Banff and to estimate how many records will fail during Error Localization. This information would be used to predict how much computer time will be required for the execution of Error Localization. This step is done after the respondent data have been received and after the edits have been finalized.

The third major use is to assess the imputation process. This could be done between applications of imputation methods or after all imputation has been completed. The Edit Summary Statistics Tables produced at this stage would be compared to those produced before imputation began.

The user should consult the GEIS Applications User's Guide for more details concerning the tables and their interpretation.

Edits for Processing Negative Values

Designing edits for processing negative values can present unexpected challenges and requires special considerations that may produce unexpected results when not taken into account. For more information and examples please see the document “Specifying Edits for Processing Negative Values with Banff” (Banff Support Team, 2006).

4. PROC OUTLIER – OUTLIER DETECTION

Purpose
This procedure offers two methods to identify outlying observations, one described by Hidiroglou and Berthelot (1986) and the Sigma-Gap method developed at Statistics Canada in the 1990s. Values of selected variables are compared across records rather than comparing fields within each individual record, as is done with the linear edit rules. The user may also choose to identify values which are not extreme enough to be considered in error, yet are sufficiently unusual that they should not be used later in the imputation procedures. Outlier Detection may be limited to a selection of variables; it is not necessary to process all variables.

Options in Outlier Detection
The Outlier Detection procedure can identify two types of values. One, called **Outlier Detection Imputation (ODI),** refers to values which are so different from the other values of the same variable that they can be considered to be in error and should be imputed in a later procedure. Values of the second type are called **Outlier Detection Exclusion (ODE)**. These are values which are not sufficiently extreme to be considered in error, but are unusual enough that the user might not wish to donate them later in the Donor Imputation procedure or might not wish to have them contribute to the parameters (e.g., means, regression parameters) used by the imputation estimators. Parameters are defined so that the Outlier Detection procedure will identify ODIs, ODEs, or both. In both the Hidiroglou-Berthelot and Sigma-Gap methods, the bounds used to define these values are not fixed, but are a function of user-specified parameters and of the data themselves. In the output field status SAS dataset, ODI values are flagged as FTI (Field to Impute), while ODE values are flagged as FTE (Field to Exclude).

Both Hidiroglou-Berthelot and Sigma-Gap methods can detect outliers from values on the right (ODER, ODIR), on the left (ODEL, ODIL), or on both sides. Use the Banff parameter SIDE to specify on which side you want to perform outlier detection.

The MINOBS parameter determines the minimum number of records required for each By group to run outlier detection. MINOBS must be greater or equal to 3 for the Hidiroglou-Berthelot method, 5 for Sigma-Gap. The system will not detect any outlier if the number of records is equal to 3 for the Hidiroglou-Berthelot method. Be cautious with the interpretation of the results for By groups with less than 10 observations. The Banff Support Team recommends a minimum of 10 observations per By group for better results.

The REJECTZERO parameter will remove zero values before the detection of outliers is done. The opposite parameter, ACCEPTZERO, will allow zero values to be included in all calculations involving current data (see next paragraph). REJECTZERO is the default value when detecting outliers with ratios and historical trends. ACCEPTZERO is the default when detecting outliers with current data. Zero and negative values are never included in the detection of outliers using ratios and historical trends.

Outlier detection can be performed in three ways, using either the Hidiroglou-Berthelot or the Sigma-Gap method: with **Current** data, with **Ratios** and with **Historical Trends**. The appropriate choice may be dictated by the data to be examined. Next, the Hidiroglou-Berthelot method is first described in detail followed by the Sigma-Gap method.

Hidiroglou-Berthelot Method

If data are available from only one period and no appropriate and reliable auxiliary variable is available, then the **Current** data must be used and each value of the selected variable is compared to bounds which are based on the values in the other records. If an appropriate and reliable auxiliary variable is available, then the user may choose to analyse the selected variable with **Ratios**. When using Ratios, Banff compares a function of the ratio of the selected variable and the auxiliary variable in each record to bounds based on the same function of ratios from the other records. **Historical Trends** is a special case of detecting outliers using Ratios in which the auxiliary value is the historical value of the selected variable and which compares the selected variable's trend over time in each record to bounds based on trends from the other records. Therefore, if historical data are available, the user may choose to use Historical Trends instead of Ratios.

Description of the Hidiroglou-Berthelot Method Using Current Data

The following steps are performed for each selected variable.

- Calculate the first quartile, Q1, the median, M, and the third quartile, Q3, of the variable being processed. (See Appendix A for a definition of median and quartiles and a description of the method used by Banff for their calculation.) If no exclusions have been specified, these values are based on all records in the data group. Otherwise, the calculations are based on the subset of records which remains after all records satisfying the exclusion clause have been removed.

- Calculate d_{Q1} and d_{Q3}. These are normally the distances from the median to the first and third quartiles, respectively. Each of these distances is replaced by a user-controlled default when it is smaller than that default. This occurs when the median and one or both of the quartiles are so close that the calculated distance(s) would be very small. The default value is a function of the median and the Minimum Distance Multiplier, a user-specified parameter, referred to as A in the following equations.

$$d_{Q1} = \text{Max}\,(\,M - Q1, |\,A * M\,|\,)$$
$$d_{Q3} = \text{Max}\,(\,Q3 - M, |\,A * M\,|\,)$$

- Calculate the intervals into which values to be imputed and values to be excluded fall. These intervals are a function of d_{Q1}, d_{Q3} and of the user-specified Multiplier for Imputation Interval and Multiplier for Exclusion Interval parameters. These multipliers are referred to as C_I and C_E in the following equations which define the ODI and ODE regions. The user may specify values for either or both of C_I and C_E. If both are specified, then C_I must be greater than C_E. If C_I is not specified then there will be no values identified as ODI, although there may be ODEs. If C_E is not specified, then there will be no values identified as ODE, although there may be ODIs.

$$\text{ODI: impute if } \begin{cases} x_i < M - C_I d_{Q1} & \text{or} \\ x_i > M + C_I d_{Q3} \end{cases}$$

$$\text{ODE: exclude if } \begin{cases} M - C_I d_{Q1} \le x_i < M - C_E d_{Q1} & \text{or} \\ M + C_I d_{Q3} \ge x_i > M + C_E d_{Q3} \end{cases}$$

- Identify the records which fall into the ODI and ODE ranges. Write the results to the output field status SAS dataset as FTI and FTE flags, respectively.

Example of the Hidiroglou-Berthelot Method Using Current Data

Suppose that *x*, the variable to be examined by Outlier Detection, has only current data with a frequency distribution as shown in Figure 4.1 and that no records are to be excluded from the calculations. The 24 values of *x* in ascending order are:

-1, 4, 7, 7, 8, 8, 8, 8, 9, 9, 9, 9, 9, 9, 10, 10, 11, 11, 11, 12, 13, 13, 15 and 19.

As described in Appendix A, the first quartile is .75 times the sixth observation plus .25 times the seventh, the median is the average of the twelfth and thirteenth values and the third quartile is .25 times the eighteenth observation plus .75 times the nineteenth.

The median is 9, the first and third quartiles are 8 and 11 respectively and the distances from the median to the quartiles are 1 and 2. The bounds are calculated with the user-specified values of $C_I = 6$ and $C_E = 4$ for the imputation and exclusion multiplier parameters, respectively. Therefore, any values which are further from the median than 6 times the distance from the median to the corresponding quartile are identified as values to be imputed. In this example, the observation with a value of -1 would be identified as requiring imputation (ODI) because it is at a distance greater than 6 times the distance from the median to the first quartile. There would be no values identified as requiring imputation due to being too large, since there are no values further than 6 times the distance from the median to the third quartile.

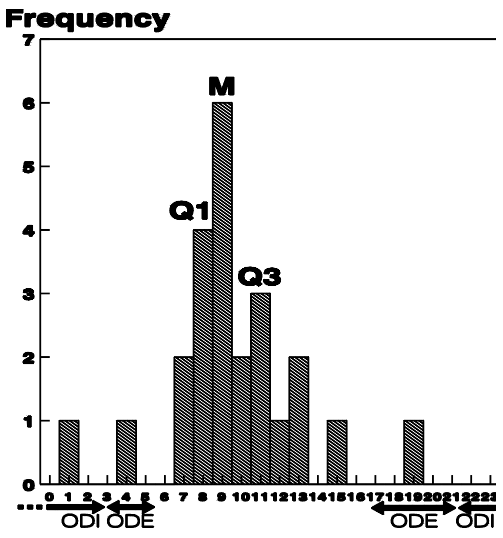


Figure 4.1 Example with Current data

$$9 - (6 * 1) = 3 \qquad\qquad \text{impute if } \begin{cases} x_i < 3 \text{ or} \\ x_i > 21 \end{cases}$$
$$9 + (6 * 2) = 21$$

Values which are less than 6 but more than 4 times the distance from the median to the quartiles are identified as values to be excluded. In this example, the observations with values of 4 and 19 would be identified as ODEs because they are between 4 and 6 times their respective quartile distances from the median.

$$9 - (4 * 1) = 5 \qquad\qquad \text{exclude if } \begin{cases} 3 \le x_i < 5 \text{ or} \\ 21 \ge x_i > 17 \end{cases}$$
$$9 + (4 * 2) = 17$$

The bounds used by the Hidiroglou-Berthelot method using Current data are linear with respect to the variable being investigated, so the user could create edits using the bounds calculated by the Outlier Detection procedure. These edits could then be incorporated directly in the set of linear edits.

Description of the Hidiroglou-Berthelot Method Using Ratios

The following steps are performed for each selected variable.

- For each record in which $x_i > 0$ and $y_i > 0$, calculate $r_i = \frac{x_i}{y_i}$, the ratio of x_i, the value of the selected variable in the ith record, to y_i, the corresponding value of the auxiliary variable in the ith record.

- Transform the r values so that an n-fold difference between x_i and y_i is the same on either side of the median difference.

$$s_i = \begin{cases} 1 - \frac{r_M}{r_i} & 0 < r_i < r_M \qquad \text{where } r_M \text{ is the median} \\ \frac{r_i}{r_M} - 1 & r_i \geq r_M \qquad \text{of the ratios } r_i \end{cases}$$

- Calculate the effect, e_i, of each record.

$$e_i = s_i \left[\max(x_i, y_i)\right]^{\exp}$$

Calculations similar to those done with the Current data previously are then performed on these transformed *e* values. The user may specify any value between 0 and 1 for *exp*, the exponent. An exponent of 0 treats all relative differences the same, regardless of the size of the unit, while an exponent of 1 gives greater importance to small deviations of large units.

- Calculate the first quartile, the median and the third quartile of the transformed *e* values of the variable being processed. If no exclusions have been specified, the *e* values are based on all records whose values of the selected and auxiliary variables are greater than zero. Otherwise, the calculations are based on the subset of records which remains after all records satisfying the user-specified exclusion clause have been removed.

- For the transformed *e* values, calculate d_{Q1} and d_{Q3}. These are normally the distances from the median to the first and third quartiles, respectively. Each of these distances is replaced by a user controlled default when it is smaller than that default. This occurs when the median and one or both of the quartiles of the transformed *e* values are so close that the calculated distance(s) would be very small. The default value is a function of the median and the Minimum Distance Multiplier, a user-specified parameter referred to as A in the following equation.

$$d_{Q1} = \mathrm{Max}(M - Q1, |A * M|)$$
$$d_{Q3} = \mathrm{Max}(Q3 - M, |A * M|)$$

- Calculate intervals based on the transformed values. These intervals will be used to identify the values to be imputed and the values to be excluded. The intervals are a function of the distances from the median to the quartiles and of the Multiplier for Imputation Interval and Multiplier for Exclusion Interval parameters specified by the user. These multipliers are referred to as C_I and C_E in the following equations which define the ODI and ODE regions. The user may specify values for either or both of C_I and C_E. If both are specified, then C_I must be greater than C_E. If C_I is not specified then there will be no values identified as ODI, although there may be ODEs. If C_E is not specified, then there will be no values identified as ODE, although there may be ODIs.

$$\text{ODI: impute if } \begin{cases} e_i < M - C_I d_{Q1} & \text{or} \\ e_i > M + C_I d_{Q3} \end{cases}$$

$$\text{ODE: exclude if } \begin{cases} M - C_I d_{Q1} \le e_i < M - C_E d_{Q1} & \text{or} \\ M + C_I d_{Q3} \ge e_i > M + C_E d_{Q3} \end{cases}$$

- Identify the records which have outlier and/or exclusion values. Write the results to the output field status SAS dataset as FTI and FTE flags, respectively.

When detecting outliers using Ratios, the calculated bounds are linear in the transformed e variable, but nonlinear in the original variable, unless an exponent of 0 was used. The example presented below using Historical Trends is an example of the Hidiroglou-Berthelot method using Ratios in the special case where the auxiliary values are the historical values of the selected variable.

Description of the Hidiroglou-Berthelot Method Using Historical Trends

This is a special case of the Hidiroglou-Berthelot method using Ratios in which the auxiliary variable, y_i , is the selected variable from the previous period. The steps performed when using Historical Trends are the same as those performed when using Ratios with the current value of the selected variable being denoted by x_{it} and the historical value of the selected variable being denoted by $x_{i(t-1)}$, where the second subscript denotes the time period, so that the ratio for the ith record is $r_i = \dfrac{x_{it}}{x_{i(t-1)}}$.

Example of the Hidiroglou-Berthelot Method Using Historical Trends

Consider an example with 22 observations in each of two time periods. Rather than reproducing all the calculations done by Banff, the results of Outlier Detection under various combinations of parameters are presented graphically. The following table gives the values of respondent data (x_t and $x_{(t-1)}$), along with r, the ratio of the current to the previous time period and s, the transformed values. The results of Outlier Detection are presented in Figures 4.3a, 4.3b, 4.4a, 4.4b, 4.5a and 4.5b. Figure 4.6 shows the results of the Hidiroglou-Berthelot method using Current data applied to only the data from the current time period. The following observations are sorted in ascending order according to their values of r.

Ident	x_t	$x_{(t-1)}$	r	s	Ident	x_t	$x_{(t-1)}$	r	s
01	60	160	0.375	-1.667	13	180	180	1.000	0.000
02	15	35	0.429	-1.333	14	200	200	1.000	0.000
03	150	192	0.781	-0.280	15	90	85	1.059	0.059
04	130	150	0.867	-0.154	16	100	85	1.176	0.176
05	40	45	0.889	-0.125	17	165	140	1.179	0.179
06	160	175	0.914	-0.094	18	30	21	1.429	0.429
07	70	75	0.933	-0.071	19	300	200	1.500	0.500
08	70	71	0.986	-0.014	20	100	50	2.000	1.000
09	40	40	1.000	0.000	21	195	97	2.010	1.010
10	62	62	1.000	0.000	22	160	60	2.667	1.667
11	75	75	1.000	0.000	23	5	0	drop	
12	100	100	1.000	0.000	24	-10	5	drop	

The effect of the s transformation may be seen by examining records 01 and 22. The value of x in record 01 has decreased from 160 in time (t-1) to 60 in time t and the value of x in record 22 has increased from 60 to 160 over the same period. These are similar changes and yet their untransformed r values, 0.375 and 2.667, are not the same distance from the median r value of 1.000. The transformation to the s values of -1.667 and 1.667 makes these similar changes symmetric about 0.0, the median of the s values.

Record 23 is ignored by the Hidiroglou-Berthelot method using Historical Trends because it has a zero in the historical data table. Record 24 is ignored because it has a negative value in the current time period.

Example using Historical Trends - Exponent = 0

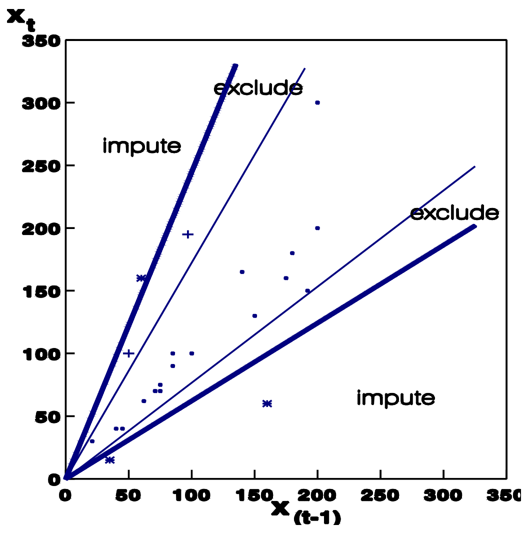


Figure 4.3a Current vs Previous Obs
Exponent = 0

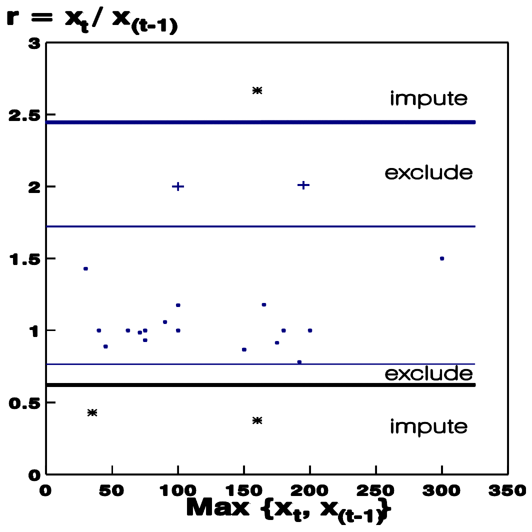


Figure 4.3b Ratio vs Maximum Value
Exponent = 0

Figure 4.3a shows each current observation, x_{It}, plotted against its corresponding historical value, $x_{I(t-1)}$ and Figure 4.3b shows the ratio of these two values plotted against the maximum of the values. In both figures, the Hidiroglou-Berthelot method with Historical Trends was used with an exponent of 0 and with multipliers of 6 and 3 for the imputation and exclusion bounds. Observations outside the heavy lines require imputation. In this example there are three such observations, each identified by an "*". Values between the heavy line and the lighter line are fields to exclude. Here there are two such values, identified by a "+". Note that the bounds are straight lines in both graphs when an exponent of zero is used.

Example using Historical Trends - Exponent = 1

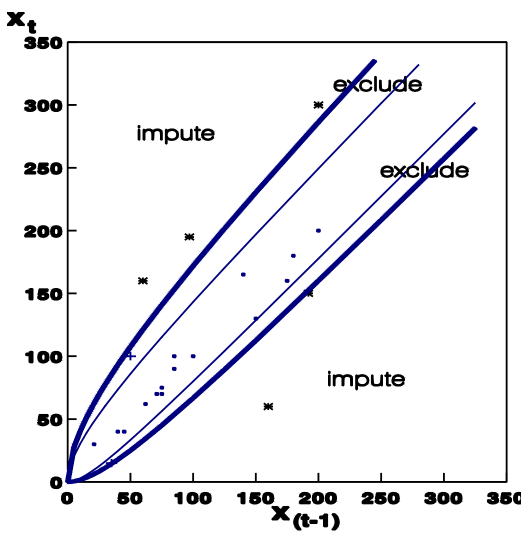


Figure 4.4a Current vs Previous Obs
Exponent = 1

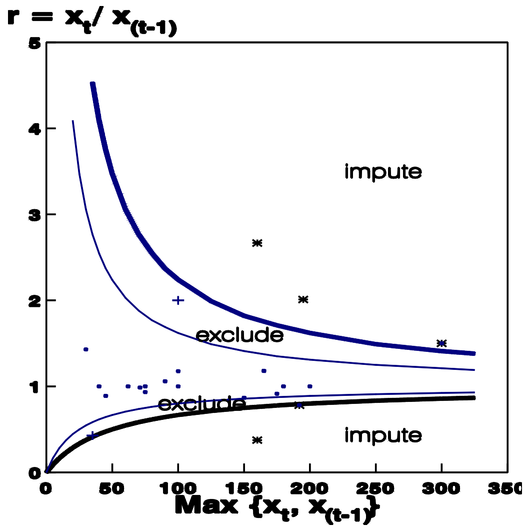
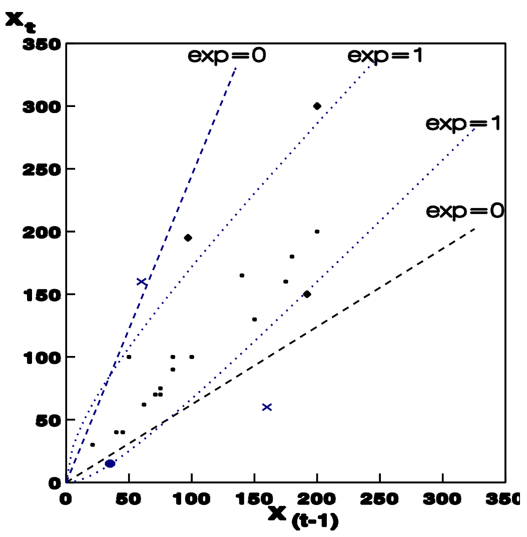


Figure 4.4b Ratio vs Maximum Value
Exponent = 1

Figure 4.4a shows each current observation, x_{It}, plotted against its corresponding historical value, $x_{I(t-1)}$ and Figure 4.4b shows the ratio of these two values plotted against the maximum of the values. In both figures, the historical trend method was used with an exponent of 1 and with multipliers of 6 and 3 for the imputation and exclusion bounds. Observations outside the heavy lines require imputation. In this example there are five such observations, each identified by an "*". Values between the heavy line and the lighter line are fields to exclude. There are two such values, identified by a "+". Note that the bounds are not straight lines when an exponent of 1 is used, but that the bounds curve allowing higher relative changes for the small units.



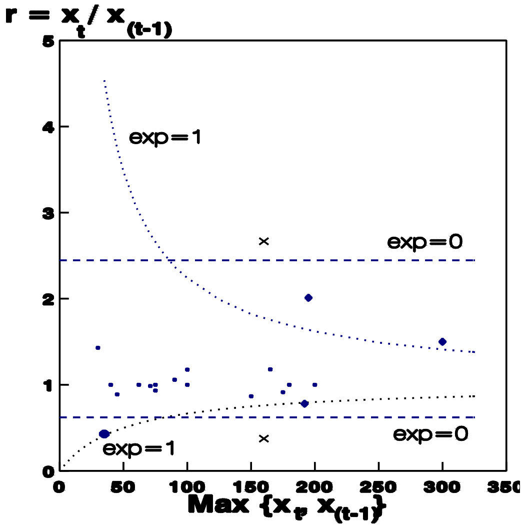


Figure 4.5a Current vs Previous Obs
Comparison of Exp=0 to Exp=1

Figure 4.5b Ratio vs Maximum Value
Comparison of Exp=0 to Exp=1

To facilitate comparison of the results of using the two different exponent values, the heavy lines indicating which observations require imputation have been taken from Figure 4.3a and Figure 4.4a and are shown together in Figure 4.5a. The heavy lines from Figures 4.3b and 4.4b are shown together in Figure 4.5b. The dashed lines represent the bounds which would be applied when an exponent of 0 is used and the dotted lines represent the bounds which would be applied when the exponent is 1. The exclusion bounds are not shown. The two observations marked as "x" were identified as requiring imputation when either 0 or 1 was used as the exponent. The three observations marked as "♦" were identified as requiring imputation when an exponent of 1 was used, but not when an exponent of 0 was used. The observation marked as "●" was identified as requiring imputation when an exponent of 0 was used, but not when an exponent of 1 was used.

Note that using an exponent of 1 tends to identify more large units as ODIs and to omit some small units which would be ODIs if an exponent of 0 were used. When the exponent is 1, modest deviations in a large unit become more likely to be identified as requiring imputation and medium deviations in a small unit become more likely to be acceptable. This is because the size of the observation itself has an impact on the value of e when the exponent is greater than 0 while the size has no impact when the exponent is equal to 0. This may be seen graphically as the bounds for exponent = 1 curve out to include medium relative deviations for small units and then tighten to allow only modest deviations in the large units. Using an exponent between 0 and 1 will result in bounds which are curved, but less than with an exponent of 1.

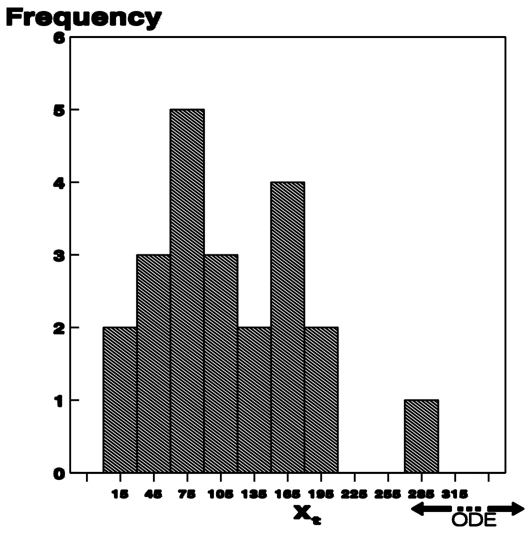
Comparison of Hidiroglou-Berthelot Using Current Data and Historical Trends

Now consider the results of applying the Hidiroglou-Berthelot method with Current data to only the x_t values of the data used in the Historical Trends example. In order to treat exactly the same group of records, records 23 and 24 must be dropped. The x_t data are listed below, sorted in ascending order by the value of x_t. The same data are shown in a frequency histogram in Figure 4.6.

Ident	x_t	Ident	x_t	Ident	x_t	Ident	x_t
02	15	07	70	20	100	17	165
18	30	08	70	04	130	13	180
05	40	11	75	03	150	21	195
09	40	15	90	22	160	14	200
01	60	12	100	06	160	19	300
10	62	16	100				

$$\begin{array}{rlll} \text{User specified parameters :} & A = .05 & C_I = 6 & C_E = 3 \\ \text{Calculated from the data :} & Q1 = 61.5 & M = 100 & Q3 = 161.25 \\ & d_{Q1} = 38.5 & & d_{Q3} = 61.25 \end{array}$$

$$\begin{array}{rlcl} \text{impute if :} & x_{it} < -131 & \text{or} & 467.50 < x_{it} \\ \text{exclude if :} & -131 \le x_{it} < -15.5 & \text{or} & 283.75 < x_{it} \le 467.50 \end{array}$$



Note that in this example, the Hidiroglou-Berthelot method with Current data would not identify a negative value as requiring imputation unless it was less than -131. Negative values which do not deviate enough from the median are not automatically flagged for imputation by this procedure. However, if the user continues with Banff processing and runs Error Localization with the option to reject negative values (Section 5), then all negative values will be identified as requiring imputation because Error Localization will automatically add a positivity edit for each variable.

The Hidiroglou-Berthelot method with Current data identifies the value 300 from record 19 as a value to be excluded and finds no values to be imputed. This is not the same result as was reached by the Hidiroglou-Berthelot method with Historical Trends. When the Hidiroglou-Berthelot method with Historical Trends was used, record 19 was identified as requiring imputation when the exponent was 1 but was identified as neither a field to impute nor to exclude when the exponent was 0.

The Hidiroglou-Berthelot method with Current data identifies the value 300 in record 19 as a value to be excluded because it is unusual compared to the other observations when only x_t is considered. The two applications of the Hidiroglou-Berthelot method with Historical Trends give different results for the reason that was discussed above: the change of 50% compared to the previous period is enough to warrant imputation in a unit larger than about 100 when the exponent is 1, but this change is not enough to warrant imputation in any unit, regardless of size, when the exponent is 0.

Sigma-Gap Method

The Sigma-Gap method can be used with Current data, Ratios or Historical Trends. Refer to the first paragraph of the Hidiroglou-Berthelot method to learn how to determine which way is more appropriate with your data.

Description of the Sigma-Gap Method Using Current Data
The following steps are performed for each selected variable.

- Calculate the deviation σ_x. The Sigma-Gap method offers two ways of calculating a deviation through the SIGMA parameter. The first is the *standard deviation* based on the mean (STD):

  $$\sigma_{STD} = \sqrt{\frac{\sum\left(\mathrm{w_i x_i} - \frac{\sum \mathrm{w_i x_i}}{n}\right)^2}{n-1}}$$

  where *x* is the original unweighted value of the variable X, and w is the associated weight provided with the WEIGHT parameter. The weight is 1 by default if the WEIGHT parameter is not specified.

  The other deviation available is derived from the median absolute deviation (MAD) which looks at the distance of each record to the median:

  $$\sigma_{MAD} = 1.4826 * med\left(\left|w_i x_i - med_j\left(w_j x_j\right)\right|\right)$$

  where the internal median $med_j\left(w_j x_j\right)$ is the median of n weighted values, and med is the external median of the absolute value of the gaps between the n weighted values and $med_j\left(w_j x_j\right)$. The adjustment factor 1.4826 makes σ_{MAD} a consistent estimator of σ_{STD} for a normal population (see Rousseeuw and Leroy, page 202). Example: the internal median of the values (5, 10, 15, 20, 25) is 15. The distances to the median in absolute values are, in the same order, |5-15|=10, |10-15|=5, |15-15|=0, |20-15|=5, and |25-15|=10 which, once ordered, gives, (0, 5, 5, 10, 10). The median absolute deviation is the external median of those values, and it is equal to 5. Therefore, $\sigma_{MAD} = 1.4826 * 5 = 7.413$ in this example.

  The deviation σ_{STD} can be largely influenced by extreme values compared to σ_{MAD}. This implies that potential outliers influence the deviation which serves to locate them. It is recommended to use σ_{MAD} for this reason. Here is an example which demonstrates the effect of outliers on the deviation. The values of variable X follow a normal distribution:

-27, -22, -21, -19, -16, -16, -15, -15, -12, -12, -8, -6, -5, -2, -2, -2, 1, 7, 8, 8, 9, 10, 14, 19, 24, 26, 29, 32, 36 and 45

The deviations are similar for these values: $\sigma_{STD} = 19.12$ and $\sigma_{MAD} = 19.27$. By modifying the last two values of the previous data to be extreme values,

-27, -22, -21, -19, -16, -16, -15, -15, -12, -12, -8, -6, -5, -2, -2, -2, 1, 7, 8, 8, 9, 10, 14, 19, 24, 26, 29, 32, 136 and 145,

the deviations become: $\sigma_{STD} = 39.20$ and $\sigma_{MAD} = 19.27$. The deviation σ_{STD} has doubled while the value of σ_{MAD} has remained the same. Consequently, the deviation σ_{MAD} tends to be less influenced by extreme values.

- Calculate the two sigma-gaps, one for detecting ODEs and one for ODIs. To do so, the deviation previously calculated is multiplied by the value of the BETAE parameter (β_E) for ODE and by BETAI (β_I) for ODI. If the OUTLIERSTAT parameter is specified, those sigma-gaps will appear in the output file, as EXCL_SIGMAGAP and IMP_SIGMAGAP respectively for each BY group:

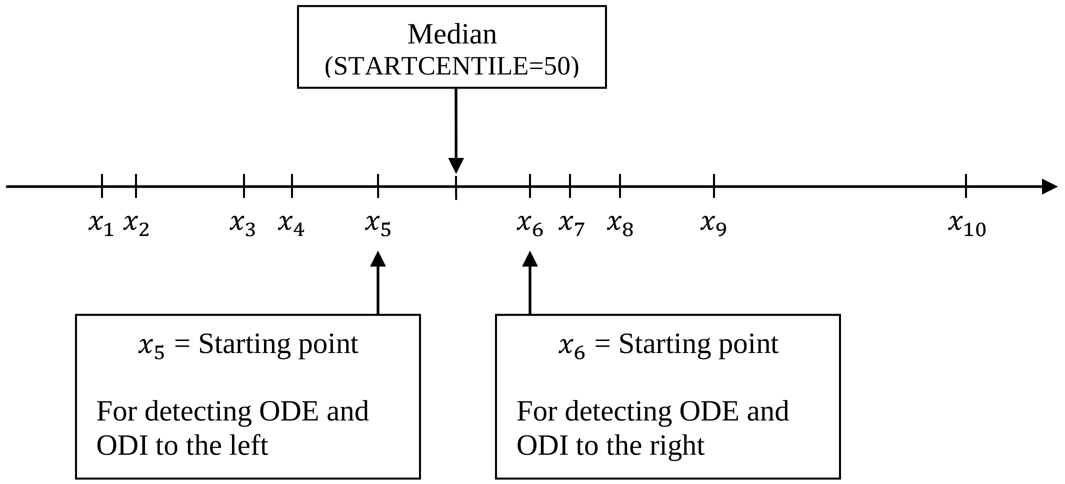
$$EXCL\_SIGMAGAP = \beta_E \sigma_{STD} \text{ } or \text{ } \beta_E \sigma_{MAD}$$

$$IMP\_SIGMAGAP = \beta_I \sigma_{STD} \text{ } or \text{ } \beta_I \sigma_{MAD}$$

- Determine the starting point. This feature is specific to the Sigma-Gap method. Although all values were used so far to calculate the exclusion sigma-gap and the imputation sigma-gap, it is possible to limit the outlier detection to only a portion of the data through the use of the STARTCENTILE parameter. The following examples will illustrate how this works.

  Let us say that the values for the variable X are available for 10 records and are already sorted in ascending order: $x_1, x_2, \dots, x_{10}$. By default, STARTCENTILE=0 when SIDE=LEFT or SIDE=RIGHT. Let us also say that the detection of outliers is performed on the right only. This means that, except for the smallest value x_1, all other values could become either ODE or ODI. However, we will set STARTCENTILE to 75. This means that outlier detection will be performed starting with the first value going to the right from the 75th centile inclusively, x_8 in this example. Thus, the starting point is x_8 which means only values for records x_9 and x_{10} could become either ODE or ODI. Note that if many records have the same value as the record at the centile position, Banff will choose the record furthest to the right as the starting point (see the example on page 4-12). If detection had been done on the left only, then outlier detection will be performed starting with the first value met going to the left from the 75th centile inclusively. Here, x_3 would be the starting point for detection on the left with STARTCENTILE=75, and only the values of records x_2 and x_1 could become either ODE or ODI. Again, if many records have the same value as the record at the centile position, Banff will choose the record furthest to the left as the starting point.

Using the same set of 10 records, an even number, with the parameter SIDE=BOTH, i.e. outlier detection is performed on both the right and left sides of the data. The minimum value for STARTCENTILE is 50, the median, when detection is performed on both sides. Here are the starting points when all the values of x_i are different:


Median
(STARTCENTILE=50)
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10}
x_5 = Starting point
For detecting ODE and ODI to the left
x_6 = Starting point
For detecting ODE and ODI to the right

Thus, with SIDE=BOTH and STARTCENTILE=50, the first record from the median going to the right is x_6. It is the starting point for detection on the right. The first record from the median going to the left is x_5 which is the starting point for detection on the left. The median is the record in the middle when there are an odd number of records. That record in the middle becomes the starting point for detection on both the left and the right sides.

It is important to note that the starting point can neither be ODE nor ODI. The default value for STARTCENTILE when SIDE=BOTH is 75.

- Calculate the gaps between each of the ordered values to identify the records which fall into the ODI and ODE ranges.

  For outlier detection on the right, x_{i+1} and all values greater or equal to x_{i+1} are ODE if $x_{i+1} - x_i > EXCL\_SIGMAGAP$, where the value of i is the position of the starting point and i+1 is the position of the next record to the right. For outlier detection on the right, x_{i+1} and all values greater or equal to x_{i+1} are ODI as soon as $x_{i+1} - x_i > IMP\_SIGMAGAP$.

  For outlier detection on the left, x_{i-1} and all values smaller or equal to x_{i-1} are ODE if $x_i - x_{i-1} > EXCL\_SIGMAGAP$, where the value of i is the position of the starting point and i-1 is the position of the next record to the left. For outlier detection on the left, x_{i-1} and all values smaller or equal to x_{i-1} are ODI as soon as $x_i - x_{i-1} > IMP\_SIGMAGAP$.

If the same gap triggers both the ODE and ODI ranges, then there will be only fields to impute (FTI). Write the results to the output field status SAS dataset as FTI and FTE flags, respectively.

First Example of the Sigma-Gap Method Using Current Data.
The data used for the Hidiroglou-Berthelot method with Current data is used again here for comparison purposes. The 24 values of *x* in ascending order are:

-1, 4, 7, 7, 8, 8, 8, 8, 9, 9, 9, 9, 9, 9, 10, 10, 11, 11, 11, 12, 13, 13, 15 and 19.

SIDE is set to BOTH to detect outliers on the left and on the right. The parameter STARTCENTILE is set to 75. The 75th centile going from the record with the lowest value to the right is x_{18} with a value of 11. Since there are three records with this value, Banff uses the last record of this group going to the right, i.e. x_{19}, as the starting point ($x_i = x_{19}$) for detection on the right. The first gap calculated between two consecutive values when checking to the right ($x_{i+1} - x_i$) is $x_{20} - x_{19}$ which gives 12-11=1, and it will be compared to the exclusion and imputation sigma-gaps. The second gap will be $x_{21} - x_{20}$ and so on.

The same process is applied for detection on the left. The 75th centile going from the record with the largest value to the left is x_5 with a value of 8. Since there are four records with this value, Banff uses the last record of this group going to the left, i.e. x_5, as the starting point ($x_i = x_5$) for detection on the left. The first gap calculated between two consecutive values when checking to the left ($x_i - x_{i-1}$) is $x_5 - x_4$ which gives 8-7=1, and it will be compared to the exclusion and imputation gaps. The second gap will be $x_4 - x_3$ and so on.

The mean absolute deviation, which will be used to calculate the gap, is 2.22 for this example. The multiplier for exclusion that we will use is β_E=1.5, and the multiplier for imputation is β_I=3.0.

The exclusion gap is calculated, and it will be used for detecting ODE on the left and the right sides:

$$EXCL\_SIGMAGAP = \beta_E \sigma_{MAD} = (1.5) * (2.22) = 3.33$$

Whenever the gap between any two consecutive values is greater than 3.33, the larger value (when going to the right) or the smaller value (when going to the left) is identified for exclusion (ODE). The imputation gap is calculated, and it will be used for detecting ODI on both the left and the right side:

$$IMP\_SIGMAGAP = \beta_I \sigma_{MAD} = (3.0) * (2.22) = 6.66$$

Whenever the gap between any two consecutive values is greater than 6.66, the larger value (when going to the right) or the smaller value (when going to the left) is identified for imputation (ODI).

In other words, any gap in the interval (3.33, 6.66] will trigger ODEs, and any gap greater than 6.66 will trigger ODIs. For example, the gap between x_{22}=13 and x_{23}=15 is 2, which is smaller than 3.33, and that means x_{23} is neither an ODE nor ODI. However, the gap between x_{23}=15 and x_{24}=19 is 4, which falls in the interval (3.33, 6.66], and thus makes x_{24} an ODE.

In this example with the Sigma-Gap method, x_1=-1 is detected as an outlier to exclude (ODE) on the left, and x_{24}=19 is an ODE on the right. In the previous Hidiroglou-Berthelot example using the same data, on the left, x_1=-1 was an outlier to impute (ODI), and x_2=4 was an ODE; on the right, x_{24}=19 was an ODE.

Second Example of the Sigma-Gap Method Using Current Data.
We have the following 20 values of *x* in ascending order:

5, 5, 5, 6, 6, 6, 6, 6, 7, 7, 7, 24, 24, 25, 25, 25, 25, 27, 28 and 100

In this case, we want to detect outliers by checking the gaps between all the values going to the right. When the parameter SIDE=RIGHT, the parameter STARTCENTILE=0 by default, which means that the record with the smallest value in ascending order, x_1=5, is the starting point (x_i=x_1), and all the other values will be checked to see if they become ODE or ODI. The sigma-gaps will once again be based on the calculated mean absolute deviation (SIGMA=MAD), which for this example is 2.97. As in the previous example with Current data, β_E=1.5 and β_I=3.0.

$$EXCL\_SIGMAGAP = \beta_E \sigma_{MAD} = (1.5) * (2.97) = 4.46$$

$$IMP\_SIGMAGAP = \beta_I \sigma_{MAD} = (3.0) * (2.97) = 8.91$$

Whenever the gap between any two points is in the interval (4.46, 8.91], the larger point is identified for exclusion (ODE). Whenever the gap between any two points > 8.91, the larger point is identified for imputation (ODI).

With the above data, the first gap greater than 4.46 or 8.91 is 17, between the values of x_{11}=7 and x_{12}=24. Since the gap exceeds the imputation sigma-gap which has a value of 8.91, the value 24 is flagged for imputation (ODI) along with all the values greater than it. Almost half of the values have been identified as outliers because they are too large. This is due to the fact that our data form two distinct groups of similar data, those from 5 to 7 and those greater than 23. By setting STARTCENTILE=80 instead of the 0 default, only the largest 20% of the data will be checked. In this case, there is only one outlier identified for imputation, the value of x_{20}=100.

Description of the Sigma-Gap Method Using Ratios
The following steps are performed for each selected variable.

- For each record in which $x_i > 0$ and $y_i > 0$, calculate $r_i = \frac{x_i}{y_i}$, the ratio of x_i, the value of the selected variable in the ith record, to y_i, the corresponding value of the auxiliary variable in the ith record. The next steps are the same as when using Current data. The only difference is the variable x_i which is replaced by r_i in the formulas. The steps are summarized next, and for more information, refer to the previous section about the Sigma-Gap method using Current data.

- Calculate the deviation σ_r:

$$\sigma_{STD} = \sqrt{\frac{\sum\left(\mathrm{w_i r_i} - \frac{\sum \mathrm{w_i r_i}}{n}\right)^2}{n-1}}$$

or

$$\sigma_{MAD} = 1.4826 * med\left(\left|w_i r_i - med_j\left(w_j r_j\right)\right|\right)$$

- Calculate the two sigma-gaps, one for detecting ODEs and one for ODIs for each BY group:

$$EXCL\_SIGMAGAP = \beta_E \sigma_{STD} \ or \ \beta_E \sigma_{MAD}$$

$$IMP\_SIGMAGAP = \beta_I \sigma_{STD} \ or \ \beta_I \sigma_{MAD}$$

- Determine the starting point using the parameter STARTCENTILE and the r_i sorted in ascending order: ascending order: $r_1, r_2, \dots, r_{10}$.

- Calculate the gaps between each of the ordered values (r_i) to identify the ratios which fall into the ODI and ODE ranges.

  For outlier detection on the right, r_{i+1} and all values greater or equal to r_{i+1} are ODE if $r_{i+1} - r_i > EXCL\_SIGMAGAP$, where the smallest value of i is the position of the starting point for detection on the right and i+1 is the position of the next record to the right. For outlier detection on the right, r_{i+1} and all values greater or equal to r_{i+1} are ODI as soon as $r_{i+1} - r_i > IMP\_SIGMAGAP$.

  For outlier detection on the left, r_{i-1} and all values smaller or equal to r_{i-1} are ODE if $r_i - r_{i-1} > EXCL\_SIGMAGAP$, where the largest value of i is the position of the starting point for detection on the left and i-1 is the position of the next record to the left. For outlier detection on the left, r_{i-1} and all values smaller or equal to r_{i-1} are ODI as soon as $r_i - r_{i-1} > IMP\_SIGMAGAP$.

- If the same gap triggers both the ODE and ODI ranges then there will be only fields to impute (FTI). Identify the records which have outlier or exclusion values. Write the results to the output field status SAS dataset as FTE and FTI flags. For example, if r_8 is ODE, then x_8 will get an FTE status.

Description of the Sigma-Gap Method Using Historical Trends

This is a special case of the Sigma-Gap method using Ratios in which the auxiliary variable, y_i, is the selected variable from the previous period. The steps performed when using Historical Trends are the same as those performed when using Ratios with the current value of the selected variable being denoted by x_{it} and the historical value of the selected variable being denoted by $x_{i(t-1)}$, where the second subscript denotes the time period, so that the ratio for the ith record is $r_i = \frac{x_{it}}{x_{i(t-1)}}$

The remaining steps are the same as for detecting outliers with the Sigma-Gap method using Ratios.

Example of the Sigma-Gap Method Using Historical Trends

The same data will be used as in the earlier example illustrating the Hidiroglou-Berthelot method using Historical Trends, 24 observations in each of two time periods which we will now use as ratios. Once again, records x_{23} and x_{24} will be ignored because they contain a zero and negative value respectively. As well, outliers are checked on both the left and right (SIDE=BOTH).

First, the historical trends r_i are computed and sorted in ascending order. The parameter STARTCENTILE=75. First, looking to the right, the 75th centile starting from the record with the lowest value is ratio r_{17}=1.179. Ratio r_{17} becomes the starting point ($r_i = r_{17}$). The first gap calculated when checking for outliers to the right ($r_{i+1} - r_i$) is between ratios r_{17} and r_{18}, 1.429-1.179=0.250, and it will be the first gap on the right to be compared to the exclusion and imputation sigma-gaps. The second gap will be $r_{19} - r_{18}$ and so on.

The same process is applied for detection on the left. We still look for the 75th centile but this time starting from the largest to the smallest value of ratios. Ratio r_6=0.914 is the starting point (r_i=r_6) for detection on the left. The first gap that will be calculated when checking for outliers to the left ($r_i - r_{i-1}$) will be between ratios r_6 and r_5, 0.914-0.889=0.025, and it will be the first gap on the left to be compared to the exclusion and imputation sigma-gaps. The second gap will be $r_5 - r_4$ and so on.

The calculated mean absolute deviation, which will be used to calculate the gap, is 0.18. The multipliers for exclusion and imputation are set to β_E=1.5 and β_I=3.0 respectively.

The exclusion gap will be used for detecting ODEs on the left and the right side:

$$EXCL\_SIGMAGAP = \beta_E \sigma_{MAD} = (1.5) * (0.18) = 0.27$$

Whenever the gap between any two consecutive values is greater than 0.27, the larger value of the two (when going to the right) or the smaller value (when going to the left) is identified for exclusion (ODE).

The imputation gap is calculated, and it will be used for detecting ODIs on both the left and the right side:

$$IMP\_SIGMAGAP = \beta_I \sigma_{MAD} = (3.0) * (0.18) = 0.54$$

Whenever the gap between any two consecutive values is greater than 0.54, the larger value of the two (when going to the right) or the smaller value (when going to the left) is identified for imputation (ODI).

In other words, any gap in the interval (0.27, 0.54] will trigger ODEs, and any gap greater than 0.54 will trigger ODIs. For example, the gap between ratios r_5 and r_6 is 0.025 which is smaller than 0.27, and that means the ratio of 0.889 for r_5 is neither ODE nor ODI. However, the gap between ratios r_2 and r_3 is 0.781- 0.429=0.352, which falls in the interval (0.27, 0.54], and thus makes the ratio r_2 an ODE. The fact that r_2 falls in the ODE interval means that x_2 will get a status of FTE.

Comparing this Sigma-Gap example to the Hidiroglou-Berthelot example using the same data, on the left, the values to be imputed (ODI) of the same two records by Hidiroglou-Berthelot are now to be excluded (ODE) by Sigma-Gap; x_1 and x_2 have an FTE status. On the right, the values of the same three records are detected using both methods; x_{20} and x_{21} still have an FTE status, and x_{22} still has an FTI status.

Order of Outliers on Output Data Set

The procedure's output data set contains an observation for each field that was identified as an outlier to exclude (FTE) or impute (FTI). Each observation contains the record number, field name, field status, and several other variables. The user may sort these observations in any order after the procedure has run by using the SORT procedure in SAS. By default, when using the Sigma-Gap method on the right or the Hidiroglou-Berthelot method on any side, all of the variables associated with each observation are automatically sorted from rightmost column to leftmost in ascending order except the record number.

However, when looking for outliers on the left using the Sigma-Gap method, the first record listed is the one whose field's value triggered the first sigma-gap. In this case, the remaining outliers on the left are then listed by the field's descending value.

5. PROC ERRORLOC – ERROR LOCALIZATION

Purpose

The purpose of the Error Localization procedure is to identify the fields which must be changed in each individual record in error so that the record can be made to pass all the edits. The original data are not changed in this procedure. The fields which require imputation are identified during the execution of Error Localization, but no imputation actually takes place. The values which will replace the original values in these fields are not determined until the imputation phase.

Description of the Method

Data to be edited by Banff are assumed to be numeric and continuous. In the edits, all variables have to be bounded either below or above or both. Positivity edits for each variable may be added through the option for rejecting negative values to the edits specified by the user to form the system of linear inequalities which is to be applied to each data record. The edits specified by the user may include equalities as well as inequalities. Letting the n variables supplied by a survey respondent be x_1 to x_n, a system with m user-specified inequalities and/or equalities and n edits to bound the variables may be written as:

$$\begin{array}{ccccc}
a_{11}x_1 + & a_{12}x_2 + & \cdots & a_{1n}x_n \leq & b_1 \\
. & . & \cdots & \leq & . \\
. & . & \cdots & . \leq & . \\
. & . & \cdots & . \leq & . \\
a_{m1}x_1 + & a_{m2}x_2 + & \cdots & a_{mn}x_n \leq & b_m \\
 & & & x_1 \geq & 0 \\
 & & & . \geq & . \\
 & & & . \geq & . \\
 & & & x_n \geq & -9999
\end{array}$$

The inequalities define a region, called a feasible region, in the n-dimensional space. When values supplied by the respondent are substituted in the inequalities, all records which satisfy the edits fall inside the feasible region. Records which fail one or more edits are outside the region. Some records which fail may contain missing or unknown values while others may have values which do not satisfy the edits.

The selection of fields to impute for records which do not pass the edits must be based on some criterion. The strategy used by Banff is to minimize the number of fields requiring imputation. In other words, it would be impossible to make a record pass the edits by changing fewer fields than the number of fields identified in the solution provided by Error Localization. This is an application of the **Rule of Minimum Change** as proposed by Fellegi and Holt (1976) and developed by Sande (1979). This approach is intuitively appealing because it retains as much as possible of the data supplied by the respondent. It should be noted that minimizing the number of fields to impute is not the same as minimizing the magnitude of the imputation. Error Localization would always choose to impute one field, even though it would have to change by a large amount, rather than choose two fields which would have to change by smaller amounts.

The purpose of the Error Localization procedure is to identify the fields which must be imputed in order to "move" a record into the feasible region. The identification of these fields is referred to as the solution to the Error Localization problem, while the number of fields requiring imputation is referred to as the **cardinality** of the solution. In reaching the solution, certain "corrections" to each record are calculated, but these "corrections" are never actually used to adjust the data. Indeed, the "corrections" would be unsuitable for that purpose since they move a record which fails the edits to a location on the boundary of the feasible region. No application would want all its edit failures imputed so that they were just at the extreme limits of acceptability.

A simplified version of the steps that Error Localization takes when a record fails one or more edits is given below. The term "corrections" is used in the Error Localization context; it is not the actual imputation that will eventually be made to the record.

- Initially, it is not known which fields require imputation so that the record may pass the edits. The approach taken is to consider independent corrections for each field. It is these corrections which become the variables in the Error Localization problem. It is likely that many of these corrections will eventually be found equal to zero and that the corresponding fields will not require imputation, but that is not known at the outset.

- Impose the condition that the number of non-zero corrections be minimized, or, equivalently, that as many of the corrections as possible be set to zero. This ensures that the number of fields requiring imputation is a minimum.

- When the corrections are applied to the fields, the resulting record must, by definition, pass the edits. Therefore, the corrected values are substituted into the edits and the linear system is solved to obtain values for the unknown corrections.

- A field does not require imputation if its corresponding correction was found to be equal to zero. Fields with non-zero corrections will be imputed in later procedures so that the record will fall inside the feasible region. Therefore, fields with non-zero corrections are identified as fields requiring imputation. These fields are given a FTI (Field To Impute) flag on the output field status SAS dataset.

<u>Multiple Solutions</u>
For any given record, there may be several solutions to the Error Localization problem which involve the identification of the same number of fields for imputation. For example, if a record fails the edit $x_1 + x_2 + x_3 = x_4$, then it does not matter which one of x_1, x_2, x_3 or x_4 is chosen for imputation, assuming that none of these values has failed other edits. Error Localization identifies all the possible solutions which involve the minimum number of fields and then selects one solution at random. This means that slightly different results may be obtained when Error Localization is run twice on the same set of data because different solutions may be chosen for records which have multiple solutions. It should be noted that only the selected solution is retained; all other multiple solutions are discarded by Error Localization.

Example of Error Localization

Consider the following set of edits which defines the feasible region shown in Figure 5.1.

$$x + y \geq 6 \quad (1)$$
$$x \leq 4 \quad (2)$$
$$y \leq 5 \quad (3)$$

Record A has the values (3,4) for x and y. It is inside the region and passes all the edits. Records B (2,3), C (4,1) and D (5,6) all fail one or more edits, but can be made to pass the edits by changing one or both of the reported values. For each individual record, Error Localization determines the minimum number of fields which must be changed and identifies that group as the Error Localization solution for that record.

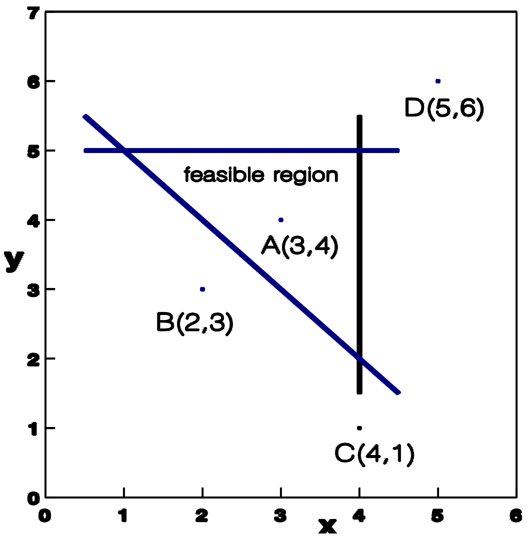


Figure 5.1 Example of Feasible Region

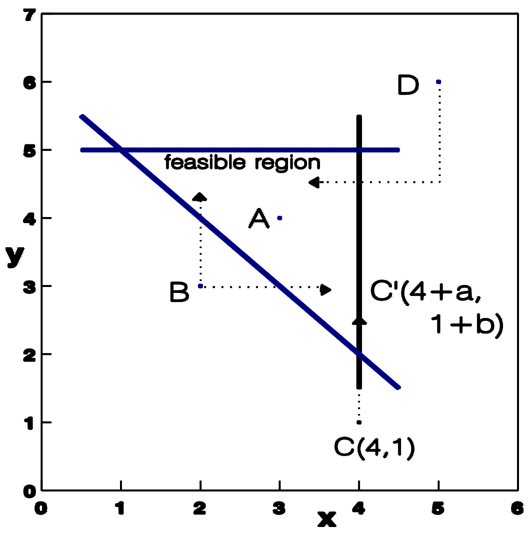


Figure 5.2 Solutions for records B, C and D

Figure 5.2 represents the possible solutions to the Error Localization problem for records B, C and D. Record B can be brought into the feasible region by changing either x, represented by the horizontal dotted line starting at B, or by changing y, represented by the vertical dotted line starting at B. This is an example of multiple solutions, each with a cardinality of one. Banff would randomly choose one of the two to use as the solution and would drop the other.

Record C can be brought into the feasible region by changing y, represented by the vertical dotted line from point C. Record C could also be made acceptable by changing both x and y, but changing y involves changing fewer fields and would therefore be the solution identified by Error Localization. The solution to this record is shown in detail below.

Record D also fails the edits and would need imputation of both x and y to enter the feasible region. This is represented in Figure 5.2 by a change in y, the vertical dotted line, followed by a change in x, the horizontal dotted line. It should be noted that x and y are identified as the group of fields requiring imputation in record D; the required change could also have been represented by a horizontal line followed by a vertical line.

As a more detailed illustration of the approach taken to reach an error localization solution, consider Record C (4,1) which lies outside the feasible region. In Banff, the corrections actually calculated would consist of the difference of two non-negative values, at least one of which would be required to be equal to zero. However, for simplicity, single corrections are applied to each field in this example.

Initially, it is not known which field or fields require imputation. Unknown corrections of a and b are considered for x and y respectively. This would "move" the original record C (4,1) to some unknown point C' (4+a,1+b) inside the feasible region. The values of the point C' must satisfy the edits because we require that C' be inside the feasible region, so, substituting into the inequalities from the original set of edits we obtain the following.

$$4 + a + 1 + b \geq 6 \quad (1)$$
$$4 + a \leq 4 \quad (2)$$
$$1 + b \leq 5 \quad (3)$$

Or, more simply,

$$a + b \geq 1 \quad (1)$$
$$a \leq 0 \quad (2)$$
$$b \leq 4 \quad (3)$$

Banff requires that the number of fields requiring imputation be minimized. There is an infinite number of possible solutions with a=0 and with b between 1 and 4. These solutions are equivalent to accepting the value of x and selecting y for imputation. There are no solutions with b=0. An infinite number of possible solutions also exist with both a and b not equal to zero, but that implies that two fields would be selected for imputation while selecting only one would suffice. Therefore, Error Localization would accept the x value and identify the value in y as requiring imputation.

Chernikova’s Algorithm

The Error Localization problem is expressed as cardinality constrained linear program (Sande, 1979) and solved using Chernikova's algorithm (Chernikova, 1964 and 1965), (Rubin, 1973). A matrix is constructed from the user-specified edits, any positivity edits and the original values of the record. Several iterations are usually necessary to reach the final solution. In each iteration, columns of the matrix are either retained or taken in linear combination with one or more other columns. In this way, the matrix often increases greatly in size, even though some columns are eliminated because they can never lead to a solution. The execution time necessary to reach a solution also increases as more columns are produced and as accompanying tests are performed more often.

The actual implementation is somewhat more complex than what has been indicated here. For a detailed description of the application of Chernikova's algorithm to Banff, see Schiopu-Kratina and Kovar (1989).

Negative Values in Error Localization

Designing edits for processing negative values can present unexpected challenges and requires special considerations that may produce unexpected results when not taken into account. For more information and examples please see the document “Specifying Edits for Processing Negative Values with Banff” (Banff Support Team, 2006).

Weights in Error Localization

In some cases, the user may wish to exert some influence on the fields that are selected for imputation. This may be accomplished by extending the rule of minimum change to include the use of weights for each variable. Error Localization then minimizes the sum of the weights of fields that are identified for imputation. For example, if a variable has a weight of two, Banff views changing that field as exactly the same as changing two fields which each have a weight of unity. In this situation, the cardinality of the solution is the sum of the weights of the fields identified as requiring imputation. Variables may be given non-integral weights. If one variable has a weight of 1.1 while all others have weights of unity, then a solution involving a field with the 1.1 weight has a lower cardinality than a solution involving two fields, but a higher cardinality than solutions involving any other single field. These weights apply to all records in the data group and edit group which are being processed; individual records cannot have different weight patterns. In other words, if the user wants error localization performed on the first data record with a certain set of weights, then error localization is done on the entire group of records using this set of weights.

Care must be taken in assigning weights since variables are usually involved in several edits with different combinations of other variables. Giving a certain variable an unusual weight in one edit may have an impact on the weights needed by the other variables involved in other edits. The use of weights may also affect the execution time necessary for Error Localization to reach its solutions. If the work matrix does not attain its maximum allowed size, the execution time may be reduced. However, if the work matrix grows so large that some elimination of columns must take place then the execution time may increase. When all variables have equal weights, removing all columns with the highest total weight, or cardinality, often results in the elimination of many columns. However, when the weights are unequal, removing all columns with the highest cardinality often results in the elimination of only a few columns. In addition, when columns with the maximum cardinality have been removed, a new maximum cardinality is calculated by subtracting .5 from the current maximum. When variables have different weights, it may take several passes to reduce the new maximum cardinality to a level that is effective in controlling the production of new columns so that the total matrix size does not exceed the storage limit. Several situations in which weights are used are described below.

Some variables within an edit group may be reported more reliably than others. For example, respondents may tend to give good values for Gross Business Income but may give poor values for Total Salaries because they are not sure if bonuses should be included when reporting Total Salaries. In this situation, Banff users often assign a higher weight to the more reliable variable so that in a choice between the two fields, the more reliable one is retained and the less reliable one is chosen for imputation. It should be emphasized again that the set of weights specified by the user applies to all records in the data group and edit group being processed.

For example, consider a variable which often has a reported value of zero and which is part of the following equality edit:

wheat + oats + barley + rye + caraway = total grain produced.

Very few farms produce caraway seeds, so if this edit fails it is likely that one of the other fields has been reported incorrectly and should be changed rather than changing caraway from zero to some other value. However, if the variables have equal weights, caraway is just as likely to be chosen for imputation as any of the other four variables. The solution is to give caraway a slightly higher weight than is given to the other components. This is a special case of one variable being reported more reliably than others. The user believes that the typical respondent knows the amount of caraway seeds produced better than the amounts of other types of grain because the caraway produced is almost always zero. It may be noted that caraway would never be chosen for imputation if the reported value is a zero and the reported total is less than the sum of the parts. This is because at least one of the parts would have to be reduced and the zero value could not be decreased and still pass the edits.

Weights may be used to ensure that certain fields can never be selected for imputation. This may be useful when a variable belongs to two or more edit groups. The user must avoid the situation in which a record passes the edits of one edit group (either originally or after imputation) and is then imputed in a second edit group in such a way that it no longer satisfies the edits of the first edit group. This problem may be avoided by assigning to any variable which is part of a previously imputed edit group, a weight higher than the sum of weights of all the other variables. Obviously, the order of processing of edit groups is very important in this situation and must be given careful consideration.

Weights are often used in conjunction with equality edits to make some variables less likely to be selected for imputation when that particular edit fails. Consider the case in which the sum of four variables must be equal to a fifth variable. Say the user wishes to retain the sum if one or two of the parts could be changed to satisfy the equality. However the user would rather change the sum than have to impute three or four of the parts. This may be accomplished by assigning weights of unity to each of the four parts and a weight greater than two and less than three to the sum. On the other hand, if the user prefers always to retain the parts and to readjust the sum, then the sum must be assigned a weight less than each of the other four variables.

Limit on Number of Fields in the Solution

Another feature of Error Localization is an optional user-specified limit on the maximum number of fields which are identified as requiring imputation in any solution. The same limit applies to all records in the data group and edit group being processed. The purpose of this limit is to restrict the number of fields (or the sum of weights of fields if weights are used) to be imputed in the final solution. Obviously this is effective only if the limit is less than the number of (weighted) fields.

No Solution Found – Manual Imputation Required

If the user has imposed no limit on the (weighted) number of fields to be identified as requiring imputation, then a solution is always possible. In the worst case, this solution identifies all fields for imputation, as it would for record D in the earlier example. However, if a limit is specified, there may be records which have no solution involving a (weighted) number of fields that is less than or equal to that limit. Since the only solution for Record D involved two fields, no solution could have been found if the limit on the number of fields to be imputed had been set to one. If no solution can be found because of this limit, Banff simply rejects the entire record and notifies the user of this. The advantage of limiting the solution is that the work matrix used in the calculations does not grow to the size which would have been necessary to find a solution involving many fields. This means that execution time is reduced, but possibly at the cost of having more records which need some kind of manual imputation because no acceptable solution could be found.

No Solution Found – Time Limit Exceeded

The user may limit the execution time to be spent finding the solution for any single record. If the solution of minimum cardinality has not been found in the specified time, the record is rejected and the user notified of this. In normal processing runs, a value should always be entered in this field to avoid the possibility of a very small number of records consuming a great deal of execution time. It is difficult to recommend a precise default value since the time taken to process records varies greatly from group of edits to group of edits and from record to record. Nevertheless, a time limit between 0.5 and 30 seconds could be entered for mainframe jobs and refined after the user has studied the number of TLE records which are produced.

Time Limit Exceeded records may be resubmitted with an increased time limit. It is up to the user to ensure that the time limit has actually been increased; Banff does not retain the previously specified limit(s) to verify this. If the limit is not greater than the run which produced the Time Limit Exceeded records, then no new records will be solved. The user may also deal with Time Limit Exceeded records outside Banff.

Other Sources of Fields to Impute

It is possible that certain fields of individual records have already been identified as fields requiring imputation before the record enters Error Localization. The user may have done this manually, or this can occur if the Outlier Detection procedure has been used to flag fields as fields to impute. See Section 4 of this guide for more details on Outlier Detection. To ensure that these fields are identified as fields to impute by Error Localization, the user should set the values of these fields to missing before the data enters Error Localization.

6. PROC DETERMINISTIC – DETERMINISTIC IMPUTATION

Purpose

The Deterministic Imputation procedure analyzes each field previously identified as requiring imputation to determine if there is only one possible value which would satisfy the original edits. If such a value is found, it is imputed during execution of this procedure.

Description of the Method

Deterministic Imputation performs the following steps for each record which has one or more fields requiring imputation.

- Eliminate the edits which are satisfied by the valid fields on the record. Therefore, all remaining edits involve fields which require imputation and, possibly, the acceptable fields.

- The original values of the fields which are to be retained are substituted into these remaining edits to obtain a reduced set of edits. This reduced set contains edits which involve only the fields to be imputed.

- The maximum and minimum values are found for each field, subject to the reduced set of edits and, if the option to reject negative values is in effect, to the positivity edits.

- If the maximum and minimum are the same, then only one value is possible and it is imputed.

One would not expect Deterministic Imputation to find an acceptable imputation for the majority of fields which have been previously identified as requiring imputation. Nevertheless, it is useful to run Deterministic Imputation because the solutions that are found in this procedure may never be found by the other imputation methods. In addition, running Deterministic Imputation serves to reduce the number of fields which will require imputation by subsequent methods. Generally, Deterministic Imputation would be expected to find more solutions if there are equality edits. However, deterministic solutions may exist, even without equality edits, as illustrated by the following example.

Example of Deterministic Imputation

Consider the following set of edits when the option to reject negative values is in effect and the values in the record after Error Localization.

Original Edits		Positivity Edits		Record
$x_1 + x_2 \le x_3$	(1)	$x_1 \ge 0$	(5)	$x_1 = ?$ (imputatio n required)
$.54\,x_3 + x_4 \le .9\,x_1$	(2)	$x_2 \ge 0$	(6)	$x_2 = 400$
$.6\,x_3 \le x_1$	(3)	$x_3 \ge 0$	(7)	$x_3 = 1000$
$x_3 \le 1500$	(4)	$x_4 \ge 0$	(8)	$x_4 = ?$ (imputatio n required)

Edits (1), (2), (3), (5) and (8) involve the fields which require imputation, so these edits cannot be satisfied at this point. However, edits (4), (6), and (7) involve only fields which are to be retained and these edits are satisfied by the acceptable values in the record. Therefore, the first step in finding the reduced set of edits is to eliminate edits (4), (6), and (7). Next, the values of the fields which are to be retained are substituted into the remaining five edits to obtain the reduced set of edits.

Reduced Set of Edits

$$\begin{array}{rlrl} x_1 \le 600 & (1) & x_1 \ge 0 & (5) \\ 540 + x_4 \le .9\, x_1 & (2) & x_4 \ge 0 & (8) \\ 600 \le x_1 & (3) & & \end{array}$$

The maximum value that x_1 can have while still satisfying the edits is 600 and the minimum value that it can have while still satisfying the edits is also 600. Therefore, the value 600 is imputed for x_1. Substituting 600 for x_1, satisfies edits (1), (3), and (5), so these edits are eliminated. Next, the values of the fields which are to be retained are substituted into the remaining two edits to obtain the reduced set of edits.

Reduced Set of Edits

$$x_4 \le 0 \quad (2) \qquad x_4 \ge 0 \quad (8)$$

Clearly, the only value satisfying both edits is 0, so the value 0 is imputed for x_4.

Consider the same set of original edits when the option to accept negative values is in effect and the values in the record after Error Localization.

Original Edits		Record
$x_1 + x_2 \le x_3$	(1)	$x_1 = ?$ (imputatio n required)
$.54\, x_3 + x_4 \le .9\, x_1$	(2)	$x_2 = 400$
$.6\, x_3 \le x_1$	(3)	$x_3 = 1000$
$x_3 \le 1500$	(4)	$x_4 = ?$ (imputatio n required)

Again, the first step in finding the reduced set of edits is to eliminate edit (4). Next, the values of the fields which are to be retained are substituted into the remaining three edits to obtain the reduced set of edits

Reduced Set of Edits

$$\begin{array}{rl} x_1 \le 600 & (1) \\ 540 + x_4 \le .9\, x_1 & (2) \\ 600 \le x_1 & (3) \end{array}$$

The only value of x_1 that satisfies these edits is 600, so the value 600 is imputed to x_1. Substituting this value into the remaining edits, produces the reduced set of edits

Reduced Set of Edits

$$x_4 \leq 0 \quad (2)$$

Deterministic Imputation can find no solution for x_4 in this case because x_4 is unbounded. Another method of imputation must be used.

Negative Values in Deterministic Imputation

Designing edits for processing negative values can present unexpected challenges and requires special considerations that may produce unexpected results when not taken into account. For more information and examples please see the document “Specifying Edits for Processing Negative Values with Banff” (Banff Support Team, 2006).

7. PROC DONORIMPUTATION – DONOR IMPUTATION

Purpose

The Donor Imputation procedure uses a nearest neighbour approach to find, for each record requiring imputation, the valid record that is most similar to it and that will allow the imputed recipient record to pass the user-specified post imputation edits. The imputation is performed if such a record is found. Donor Imputation is the preferred method of imputation for many applications because all fields requiring imputation are taken from the same donor record and relationships between the imputed variables are therefore retained.

Definitions

Recipient refers to a record which has at least one field requiring imputation in the edit group being processed. These records are identified by the Error Localization procedure, or by the user using some other method outside of Banff. Outlier Detection may have been run prior to Error Localization and may also have identified fields requiring imputation.

Donor refers to a record which has passed all the edits and which has no fields requiring imputation in the edit group being processed or in the user-specified matching fields outside the edit group. All fields of donor records should ideally be original respondent data or data imputed by Deterministic Imputation, although the user may specify that previously-imputed data also be eligible. The values which are imputed by Donor Imputation are taken directly from donor records. Potential donor records may have some fields which have been identified as Field To Exclude (FTE) fields and therefore cannot be donated.

Mixed refers to a record which has some characteristics of both donors and recipients. Such a record has no fields to impute inside the edit group being processed so it is not a recipient, but is not a donor either because it requires imputation in one or more of the user-specified match fields outside the edit group.

Field to Exclude (FTE) codes may refer to fields on any type of record. The user may define any field as FTE. As well, the Outlier Detection procedure has determined that such fields have values which are not extreme enough to require imputation, but are sufficiently unusual that they should not be donated to other records. These values are accepted in the original record because they have been supplied by the respondent. Such fields may be used as matching fields when they occur in a donor record. FTE fields have no effect in a recipient record.

Matching Fields are calculated individually for each recipient record. The values of the matching fields of a recipient do not have to match exactly to the values of the matching fields in the nearest neighbour donor record. The values in these matching fields are used to determine the distance between recipients and donor records. Since the search for a nearest neighbour is based on these fields, matching fields should be related to the fields which require imputation. It is possible for a record to have no matching fields.

User-specified Matching Field or **Must Match Field** is a field which is specified by the user to be a matching field for every recipient, regardless of whether or not it would have been chosen as such by the system. A user-specified matching field is used as a matching field for every recipient, except for cases in which the user-specified matching field of the recipient itself requires imputation. As with other matching fields, it does not have to match exactly to the corresponding field of the nearest neighbour donor record.

Post Imputation Edits form a user-specified system of linear inequalities which must be satisfied by records imputed in this procedure. No restrictions are placed on the relationship between the post imputation and original edits except that they must refer to the same fields. A common situation is to make the post imputation edits a more "relaxed" version of the original edits so that the feasible region of the post imputation edits completely contains the feasible region of the original edits. Equalities from the original set of edits are often replaced by two inequalities which allow some deviation from the strict equality. This will increase the chance of a successful imputation. On the other hand, the post imputation edit set could be made more restrictive than the original set by adding edits in order to ensure that certain relationships, such as an exact match on a classification variable, exist in the imputed data.

Example of Classification of Records

In the following example, the edit group consists of the variables x_1 to x_4 with x_5 as a user-specified matching field from outside the edit group.

		Inside Edit Group				Outside Edit Group
	Ident	x_1	x_2	x_3	x_4	x_5
donor	1	ok	ok	ok	ok	ok
donor	2	ok	ok	ok	ok	imputed
recipient	3	FTI	match-system	match-user	ok	match-user
recipient	4	FTI	match-system	match-user	ok	FTI
recipient	5	FTI	match-system	FTI	ok	match-user
recipient	6	FTI	FTI	match-user	ok	match-user
mixed	7	ok	ok	ok	ok	FTI

Records 1 and 2 are donors. Record 1 has valid original values in all fields under consideration. These values are labelled as "ok" in the above table. Record 2 has a value in x_5 which has been imputed in an earlier run of one of the imputation procedures. Record 2 is classified as a donor and the imputed value is used in distance calculations, although it would not be donated because it is outside the edit group.

Records 3, 4, 5 and 6 are recipients because each has one or more fields to be imputed (FTI) inside the edit group. The calculation of the nearest neighbour will be based on a group of matching fields which may be different for each recipient. The user has chosen two matching fields - x_3 inside the edit group and x_5 outside the edit group. These fields are labelled above as "match-user" when they contain acceptable values. The variable x_3 is used as matching field except in record 5 where it is not possible because x_3 requires imputation; x_5 is used as a matching field except in record 4 where it is not possible because x_5 requires imputation.

In addition to user-specified matching fields, Banff chooses matching fields based on an algorithm described in Section 7.2. For this example, the matching fields selected by the system are labelled as "match-system". The system chooses x_2 as a matching field for records 3, 4 and 5 but cannot choose it for record 6 where x_2 requires imputation. It is possible that a field would be a user-specified match field as well as be chosen as a matching field by the system. This has no effect on this example because it would not change the classification of records as donor, recipient or mixed. However, special codes are written to the output field status SAS dataset to indicate if a match field was chosen by the user, by the system or by both.

Record 7 is a mixed record. It is not a donor because its value in x_5 requires imputation and it is not a recipient because no imputation is required inside the edit group.

In Banff, there are four internal processes involved in the execution of Donor Imputation:

- 7.1 Prepare for Donor Imputation
- 7.2 Find Matching Fields
- 7.3 Transform Matching Fields
- 7.4 Perform Donor Imputation

A description of each of these processes follows in sections 7.1-7.4.

Negative Values in Donor Imputation
Designing edits for processing negative values can present unexpected challenges and requires special considerations that may produce unexpected results when not taken into account. For more information and examples please see the document "Specifying Edits for Processing Negative Values with Banff" (Banff Support Team, 2006).

Mass Imputation
Some surveys select samples of units from which a set of core information is collected. More detailed information may be collected from a sub-sample of these units. A procedure known as "mass imputation" is then used to impute by donor those items to units not in the sub-sample, in order to create a complete rectangular file. Mass imputation can be considered as a special case of donor imputation, where donors are selected based solely on user-specified matching variables. A separate procedure was developed to carry out mass imputation and is described in Section 10.

7.1 PREPARE FOR DONOR IMPUTATION

Purpose

The user must specify several parameters in order for the results of Donor Imputation to satisfy their requirements. The user may exclude certain records from the donor population and impose criteria which must be satisfied for donor imputation to proceed.

Previously Imputed Data in Donor Records

The user has the option to specify whether or not records are eligible donors if they have previously imputed values in at least one field in the edit group. Note that values imputed by Deterministic Imputation are treated as original respondent data, however. In any case, Proc Donorimputation always uses the closest available donor in the chosen set which allows the resulting record to pass the edits.

Keeping records with both original and previously-imputed data as donors makes the donor pool as large as possible and may therefore result in finding a suitable donor for a greater percentage of recipients. Because more records are included as donors, this choice may better represent the distribution of records in the actual sample. One disadvantage is that fields which were donated previously may have a greater chance of being used because they are repeated - once in the original records and again in the previously-imputed record(s).

Choosing not to keep donor records with previously-imputed values limits the potential donors to those which were originally clean and thus ensures that only original data can be used for imputation. Some users view this as an important advantage; others prefer to include as many donors as possible in the donor population. One of the disadvantages of reducing the number of donors is that imputation may be unsuccessful because no suitable donor is available. When imputation is successful the closest original donor is used, although there may have been clean records which were closer to the recipient but were not original donors.

Other Exclusions from the Donor Population

Banff identifies eligible records which have no fields requiring imputation in the edit group being processed or in the user-specified matching fields outside the edit group as donors. However, the user may exclude any record from being part of the donor population by identifying them in the input dataset.

Exclusion of other records will limit the records which are identified as donors. The user must balance the advantages of excluding certain types of inappropriate records from the donor population with the risk of reducing that population so much that donors cannot be found for many of the recipients.

Limiting the Use of a Donor

Besides being able to exclude certain records from the donor population, the user may also limit the number of times a donor is used by supplying at least one of two parameters: *NLIMIT* (number limit) and *MRL* (multiplier for ratio limit) which are used in the calculation of *DONORLIMIT* for each data group *j* as follows:

$$DONORLIMIT_j = \max\left\{NLIMIT, \left\lceil MRL * \left\lceil \frac{\#\, recipients_j}{\#\, donors_j} \right\rceil \right\rceil \right\}$$

The symbol $\lceil\ \rceil$ means rounded up to the nearest integer. The maximum between *NLIMIT* and the other statistics using *MRL* is used for each data group *j* as the *DONORLIMIT*$_j$. If only one parameter is provided, *NLIMIT* or *MRL*, then the other is excluded from the *DONORLIMIT* calculation. If neither parameter is specified, then there is no limit to the number of times a donor can be used.

Neither the minimum number of donors required (parameter *MINDONORS*) nor the minimum percentage of donors required (parameter *PCENTDONORS*) is impacted by *DONORLIMIT* since they are both calculated at the beginning of the donor imputation procedure.

A donor which has reached *DONORLIMIT* will be retired from the donor pool. It will be ignored and not counted in the maximum number of donors to try (parameter *N*).

Imputation will stop for a given data group *j* when all donors have reached *DONORLIMIT*$_j$. In such a case, there will not be any more attempts to find a donor for the remaining recipients. By using an *MRL* ≥ 1, at least one donor will be available for attempting to impute each recipient.

Criteria for Donor Imputation

The user may specify a percentage and number of donor records which must be available for imputation to proceed. If there is a possibility that the donor population will be drastically reduced by the specified exclusion expression, then these criteria may be used to ensure that at least a minimum percentage and number of donors are available. If the criteria are not satisfied, a message will be printed and donor imputation will not be performed.

7.2 FIND MATCHING FIELDS

Purpose

The Find Matching Fields function of the Donor Imputation procedure analyzes each recipient record individually to determine a set of fields to be used in the calculation of the distance from that recipient to the donor records. It is possible for a recipient to have no matching fields. When matching fields exist, they must be some, or all, of the valid, acceptable values of the recipient. Many different combinations of matching fields are usually found for any given edit group and data group. The selection of matching fields depends on each recipient's pattern of fields to impute, the values in its fields which are retained and, of course, the original edits. If user-specified matching fields have been declared, they are included in the recipient's set of matching fields.

User-specified Matching Fields or Must Match Fields

User-specified matching fields may be either inside or outside the edit group. A variable which has been declared a user-specified matching field is used in the distance calculation along with the other matching fields. These fields are sometimes called must match fields, although they do not have to match exactly.

User-specified matching fields from outside the edit group are especially useful when there is a variable which is thought to be correlated with the variables in the edit group but does not appear explicitly in the edits. For example, suppose that gross business income (GBI) does not enter the edits specified for the edit group containing employment and salary data. Even though a direct relationship has not been specified, the user may feel that the value of a recipient's GBI should have an impact on the record which will eventually be chosen as its donor. The user would therefore select GBI as a user-specified matching field from outside the edit group.

The user should ensure that all data for variables which might be used for matching fields have already been edited, either by the Error Localization procedure of Banff or by some other process. If not, the calculation of the nearest neighbour records could be based on values which will eventually be changed. This is especially true for user-specified matching fields which come from outside the edit group. It is the user's responsibility to consider the order of processing so that this does not become a problem. If imputation has been performed as well as error localization, donor values in the user-specified matching fields outside the edit group may be original "good" values or values which have already been imputed. In either case, the existing donor value is used to calculate distances but would never be donated because only fields inside the edit group are imputed.

The user may also select fields inside the edit group as user-specified matching fields. It should be emphasized that such fields are included as matching fields for every recipient being processed in that run, even if the system would not have chosen those variables as matching fields. The only situation in which a user-specified matching field is not used occurs when that field requires imputation in the recipient record so there is no valid value to be used in the distance calculation.

Description of the Method

The procedure performs these steps individually for each recipient record.

- Substitute the known, acceptable values of the recipient record into the original edits. Exclude any edits which are reduced to relations between constants, i.e., which no longer contain any variables.

- The remaining edits form a reduced set with the fields requiring imputation as the only unknowns. This set of edits defines a feasible region which contains all allowable values of the fields requiring imputation. Select the edits which limit these fields by determining the boundary of this feasible region. All other edits are redundant for the reduced set of edits and are dropped.

- Return to the original form of the edits which were identified in the previous step as part of the boundary of the feasible region of the reduced set of edits. The variables in these edits which are not fields requiring imputation are chosen as the matching fields.

Example of the Determination of Matching Fields

Original Edits

$x \geq y$ (1)	$x \geq 0$ (5)
$x \leq 5$ (2)	$y \geq 0$ (6)
$y \geq u$ (3)	$u \geq 0$ (7)
$y \leq 2v$ (4)	$v \geq 0$ (8)

Records 1 and 2 have valid values for u and v but require the imputation of x and y.

Record 1	Record 2
$x = ?$ (impute)	$x = ?$ (impute)
$y = ?$ (impute)	$y = ?$ (impute)
$u = 1$	$u = 1$
$v = 2$	$v = 3$

Substitute the acceptable values of both records to obtain the following relations.

Record 1		Record 2	
$x \geq y$ (1)	$x \geq 0$ (5)	$x \geq y$ (1)	$x \geq 0$ (5)
$x \leq 5$ (2)	$y \geq 0$ (6)	$x \leq 5$ (2)	$y \geq 0$ (6)
$y \geq 1$ (3)	$1 \geq 0$ (7)	$y \geq 1$ (3)	$1 \geq 0$ (7)
$y \leq 4$ (4)	$2 \geq 0$ (8)	$y \leq 6$ (4)	$3 \geq 0$ (8)

For both records, inequalities (7) and (8) are simply relations between constants and are therefore excluded. For record 1, edits (1) to (6) form the reduced set of edits with edits (1) to (4) defining the boundary of the feasible region as shown in the first part of Figure 7.1. Referring to the original edits, it may be seen that edits (1) to (4) involve x and y which are to be imputed and u and v which were found by Error Localization to be acceptable values. The Matching Fields procedure then chooses u and v as the matching fields for record 1.

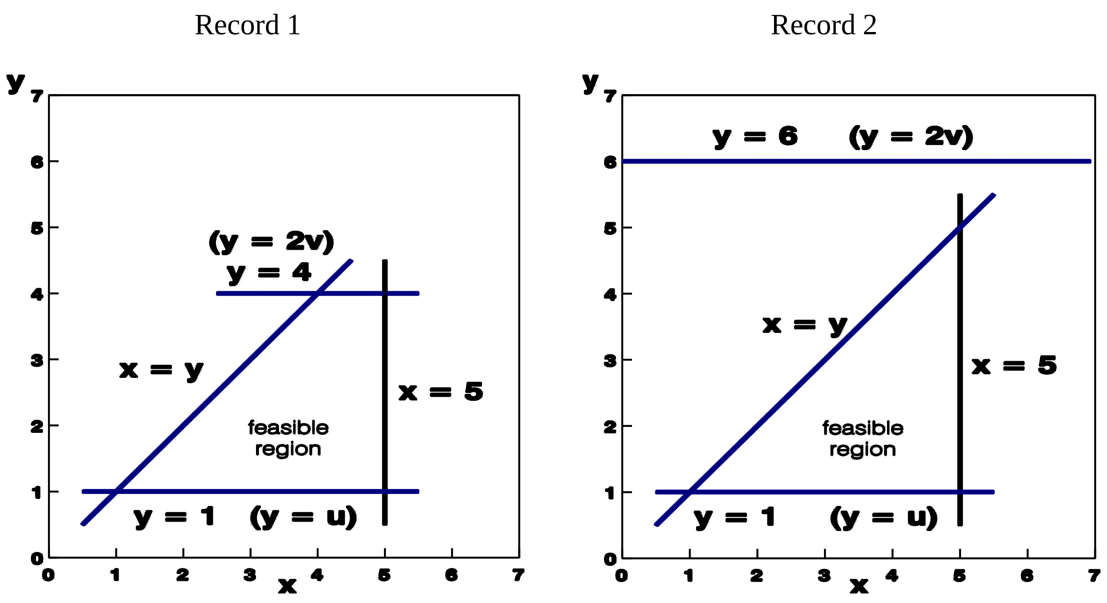


Figure 7.1 Feasible regions for the reduced set of edits - Record 1 and Record 2

For record 2, edits (1) to (6) also form the reduced set of edits. In this case the boundary of the feasible region is defined by edits (1) to (3) as is shown in the second part of Figure 7.1. The original edits (1) to (3) involve x and y as fields to impute and u which is an acceptable value and is chosen as the matching field for record 2.

Record 1 and record 2 are subject to the same edits, have the same fields to impute and have very similar values in their acceptable fields. Yet, their matching fields are different. This is because the value of v limits the value of y in record 1 but has no effect on the values of x or y in record 2.

Example of No Matching Fields

Now consider the following set of edits and the values in record 3.

Original Edits	Record 3
$x \geq 2$ (1)	$x = ?$ (impute)
$x \leq 5$ (2)	$y = ?$ (impute)
$y \geq 1$ (3)	$u = 3$
$y \leq 4$ (4)	$v = 4$
$u + v \leq 10$ (5)	

Substitute the acceptable values of record 3 to obtain the following relations.

$$x \geq 2 \quad (1)$$
$$x \leq 5 \quad (2)$$
$$y \geq 1 \quad (3)$$
$$y \leq 4 \quad (4)$$
$$7 \leq 10 \quad (5)$$

Edit (5) is excluded because it is simply a relation between constants. The remaining edits, that is edits (1), (2), (3), and (4), form the reduced set of edits. The feasible region described by the reduced set of edits is shown in Figure 7.2. The boundary of this feasible region is defined by edits (1), (2), (3) and (4).

Then next step is to examine the corresponding edits (1), (2), (3) and (4) in the set of original edits. It may be seen that edits (1) to (4) involve only x and y, which are both fields to impute. There are no acceptable fields involved in these edits, so there are no matching fields identified for this record. This is equivalent to saying that u and v, the acceptable fields for record 3, do not provide any information concerning the values which should be imputed for x and y.

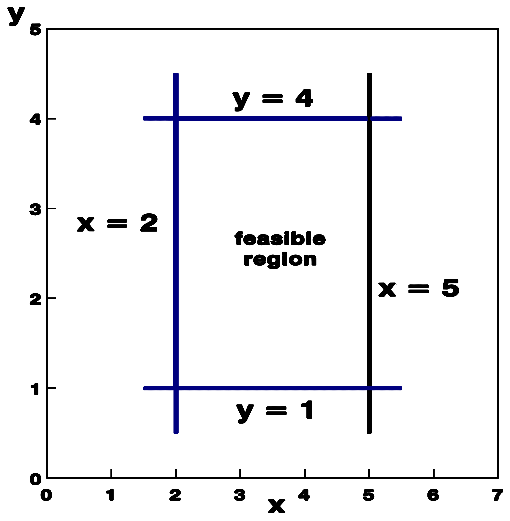


Figure 7.2 Feasible region for the reduced set of edits - Record 3

7.3 TRANSFORM MATCHING FIELDS

Purpose

The Transform function of the Donor Imputation procedure performs a rank value transformation on all valid values within an edit group for each variable which has been selected at least once as a matching field. This is done to remove the effect of scale from the data which are used in the calculation of the distance between two records. If no transformation were done, original data with wide ranges, such as dollar values, would always dominate the distance calculation.

Description of the Method

The Transform procedure performs the following steps independently for each variable which has been selected at least once as a matching field.

- Sort all the valid values of the field in ascending order. This includes donor records, recipient records for which the variable is a matching field, recipient records for which the variable is acceptable but is not a matching field and mixed records for which the variable is valid. Values which require imputation are ignored since they are not valid.

- Assign each value a rank. A group of tied values receives one rank for each occurrence of the shared value. Each record with this tied value is then assigned the average of these ranks.

- Divide each rank by the total number of valid values plus one. This divisor is likely to be different for each variable, since each variable may have a different number of records with valid fields.

Example of Transformed Values

Original Value (Sorted)	Rank	Transformed Values
-12	1	.1
26	2	.2
38	4	.4
38	4	.4
38	4	.4
47	6.5	.65
47	6.5	.65
53	8	.8
105	9	.9

The rank value transformation results in observations which, except for ties, are uniformly distributed across the range (0,1). Extreme values are pulled in close to the other observations and do not have an impact on the resulting transformed values. Note that in the above example, there would be no change in the transformed values if the largest original value were reduced to 54 or increased to 1000.

Distance Calculation

With numeric, continuous data, one would not expect to find an exact match between a recipient and a donor. Since Donor Imputation is based on transferring data from a record which is as similar as possible to the recipient, some method of deciding which record is closest, or most similar, must be established.

Banff uses the L^{∞} norm to define this distance between two records. The distance between a record with transformed values of matching fields of $(x_1, x_2, \ldots x_n)$ and a record with transformed values of matching fields of $(y_1, y_2, \ldots y_n)$ is defined as

$$\max(|x_1 - y_1|, |x_2 - y_2|, \ldots, |x_n - y_n|).$$

This is sometimes referred to as the minimax distance because the closest donor is the one with the smallest maximum absolute difference between the transformed values of its matching fields and those of the recipient. Note that all matching fields have the same weight.

Example of Distance Calculation

Consider the following records representing a recipient and three donors. The transformed values of the recipient's three matching fields, x_1, x_2 and x_3 are given below, along with the distances from the recipient to each donor.

	x_1	x_2	x_3	Distance to Recipient
recipient	.5	.5	.5	
donor 1	.6	.6	.6	max (|.5-.6|,|.5-.6|,|.5-.6|) = max (.1,.1,.1) = .1
donor 2	.5	.5	.4	max (|.5-.5|,|.5-.5|,|.5-.4|) = max (0, 0,.1) = .1
donor 3	.8	.5	.5	max (|.5-.8|,|.5-.5|,|.5-.5|) = max (.3, 0, 0) = .3

Note that donor 1 and donor 2 are the same distance from the recipient, even though all three fields of donor 1 are .1 from the corresponding fields of the recipient while two of the fields of donor 2 are exactly the same as those of the recipient and one field is .1 away. Donor 3 and donor 1 have the same total absolute difference, but donor 1 has this difference spread out among the three matching fields while donor 3 has all this difference concentrated in one matching field. This definition of distance considers a donor with moderate differences from the recipient in all matching fields to be closer than a donor which has a large difference in one matching field and little or no difference in the others.

7.4 PERFORM DONOR IMPUTATION

Purpose
The purpose of this function of the Donor Imputation procedure is to find, for each recipient record, the closest donor record whose values will allow the recipient record to pass the user-specified post imputation edits. As described in Friedman, Bentley and Finkel (1977), a tree structure is created in order to make the search for donors more efficient. Donors are found for records both with and without matching fields.

Description of the Method
Only one tree is constructed for each data group. The following steps are performed once for the entire set of k matching fields identified earlier by the Find Matching Fields procedure for the data group being processed.

- Create a tree structure to organize the records which have been identified as donors. The number of levels in the tree is controlled by the bucket size. See below for a more detailed description of how the tree is constructed.

- For each recipient with matching fields, traverse the tree and select a group of closest donors. The size of the group is the *n* parameter specified by the user.

- Beginning with the closest, test donors in the group until one is found which does not have FTE flags on fields to be imputed and which would result in an imputed record which passes the post imputation edits. If such a record is found, the imputation is performed and the procedure goes to the next recipient record.

- If none of the *n* potential donors can be used, the procedure reports that no donor could be found for the recipient.

- If a recipient has no matching fields, the user has the option of selecting donors at random until a suitable one is found. This normally occurs with the first donor selected. In this case, the k-d Tree is not used in the search for a donor. See below for a more detailed explanation.

Construction of the k-d Tree
It would be possible to carry out the search for donors without a tree structure, but efficiency is increased by using a k-dimensional tree, where k is the total number of matching fields used in the entire data group. In constructing the tree, the population of donor records is partitioned into smaller and smaller groups of similar records. In this way, donors which are close to a given recipient tend to be located together in certain branches of the tree. In addition, some branches of the tree can be excluded from the search because it can be deduced that they cannot contain donors closer than the ones which have already been found.

Each point at which the tree splits into two branches is called a **node**. The first, or highest, level is called the **root node** which represents all donor records in the tree. At the root node, the k matching fields used are examined and the field which has the largest range of transformed values is chosen. This is called the **branching field** of the root node. A **splitting value** is chosen such that it will divide the donors into two groups which are as equal as possible. If there is an even number of donors and if the central values are not identical, the splitting value is the same as the median. If there is an odd number of donors and if the central values are not identical, then the splitting value is the average of the $\frac{n+1}{2}$ and the $\frac{n+1}{2}-1$ values. If the central values are identical, Banff makes the two groups as equal as possible while still ensuring that any sets of identical records remain together. In this case, the splitting value is the average of the central values and the value immediately before or after the group of identically valued records. Once the branching field and splitting values have been calculated, two nodes are formed at the next lowest level, each representing about half of the donors of the higher level.

The selection of branching fields and splitting values continues until all nodes contain no more than 16 records (the **bucket size**). A node which has 16 or less donors is called a **terminal node**. A terminal node may contain more than 16 records only if all the donors represented by the node have exactly the same values in all the matching fields. In this case, the identical donors would be assigned to the same terminal node, even if their number were greater than 16. The set of all terminal nodes provides a partition of the k-dimensional space into mutually exclusive and exhaustive subsets of donors.

Example of Construction of a Tree

Consider an example with eight donors which are to put into a k-d tree with a bucket size of only two. Such a small bucket size is used here to keep the example manageable. Banff always uses a bucket size of 16 in practice. Note that only donors are given in the list below, so there may be transformed data values belonging to recipients which are in between those which appear here.

	Transformed Values		
	x	y	
donor 1	.10	.20	
donor 2	.30	.50	
donor 3	.40	.60	(Sorted
donor 4	.50	.55	by x)
donor 5	.50	.80	
donor 6	.50	.70	
donor 7	.85	.95	
donor 8	.95	.60	

The letters used to identify each node in the following description are the same as those used in Figure 7.3. At each node, the first step in the construction of the tree is to decide which variable is to be used as the branching field. Since the variables x and y are the only match fields in this example, the branching field will always be one or the other of these fields. At the root node, node A, the x values of all the donors have a range of .95 -.10 =.85 and the y values of the same donors have a range of .95 -.20 =.75. Therefore, x is chosen as branching field. The next step is to split the donors into two groups as evenly as possible. Normally the split would be between donors 4 and 5, but donors 4, 5 and 6 have the same value for x, so the split must be either just before or just after this group, that is, either between donors 3 and 4 or between donors 6 and 7. Splitting between donors 3 and 4 makes the split as even as possible. The splitting value is calculated as .45, the average of the values of donors 3 and 4. All records with values of $x \leq .45$ are associated with node B, while those with $x > .45$ are associated with node C. Node B represents all donors with $x \leq .45$ while node C represents all donors with $x > .45$. The equality is always assigned to the left node. It is possible for a recipient to have a value of x exactly equal to .45, but no donor can have this particular value since the splitting value is the average of two consecutive donor values.

Node B now represents three records (donors 1, 2 and 3) and cannot be a terminal node since the terminal bucket size is two. The range of x for these records is .30 and the range for y is .40, so y is chosen as the branching field for this node. An odd number of records cannot be divided exactly in half. In this situation, Banff always splits between the central value and the next larger value. Here, there are three records and Banff splits between the second and third. Donors 1 and 2 are associated with node D and donor 3 is associated with node E. Both these nodes represent two or fewer records, so they both become terminal nodes. All records at node D have $x \leq .45$ and $y \leq .55$ while those at node E have $x \leq .45$ and $y > .55$.

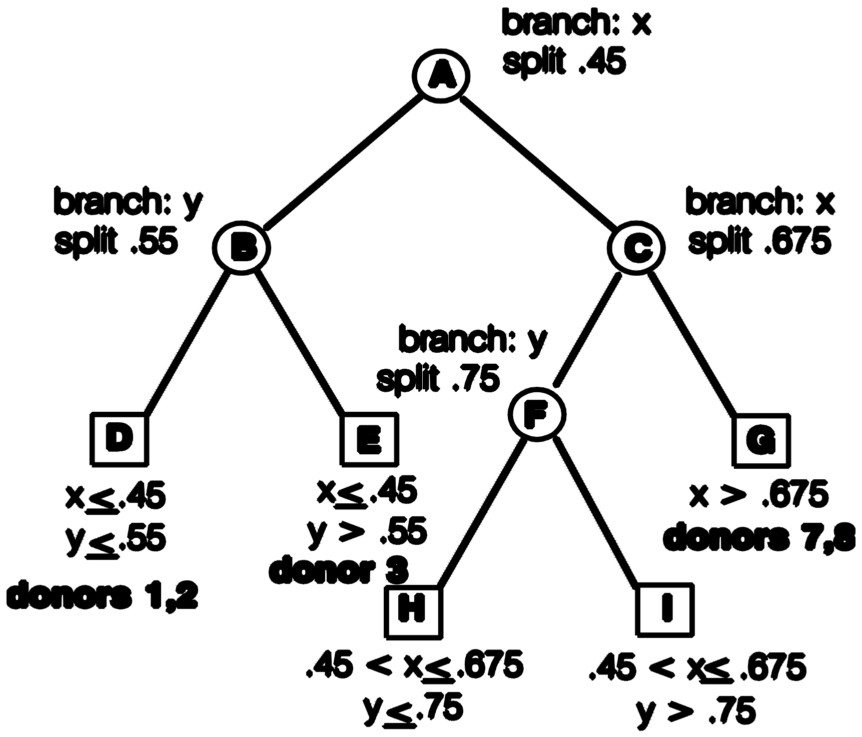


Figure 7.3 Example of the construction of a k-d tree

The construction of the tree continues at node C. The variable x has a range of .45 and y has a range of .40 so x is chosen as the branching field. It does not matter that x was also chosen as the branching field at node A. The donors are split at .675 with donors 4, 5 and 6 going to node F and donors 7 and 8 going to node G. Node G becomes a terminal node because it represents only two records. All records at node G must have values of $x > .675$, but may have values of y anywhere in the range (0,1).

At node F, y is used to split donors 4, 5 and 6 because it has the greater range. The splitting value is the average of the values of donors 5 and 6. Donors 4 and 6 go to terminal node H and donor 5 goes to terminal node I. Records at terminal node H have $.45 < x \leq .675$ and $y \leq .75$. Records at terminal node I must have $.45 < x \leq .675$ and $y > .75$.

Recipients with Matching Fields – Traversing the Tree

Now suppose that Banff is searching the tree which has just been constructed in the previous example to find the closest donor for a recipient with transformed values of .53 and .74 for x and y. A value of two will be used for the parameter n. This value is too small to be used in most applications, but is chosen here so that the example may be of manageable size. Since there are only two matching fields in this example, the partitioning of the donor population may be represented by a two dimensional graph. Figure 7.4 shows the areas of the plane which are represented by each of the terminal nodes. The donors are marked with an "x" and the recipient is marked with an "*". Figures 7.3 and 7.4 show different ways of representing the same k-d tree. Figure 7.3 emphasizes the hierarchical relationship of the nodes while Figure 7.4 emphasizes how the terminal nodes partition the donor space. Figure 7.3 is the usual way of representing the k-d tree; graphs such as Figure 7.4 cannot be represented in the higher dimensions which would be necessary for virtually all applications.

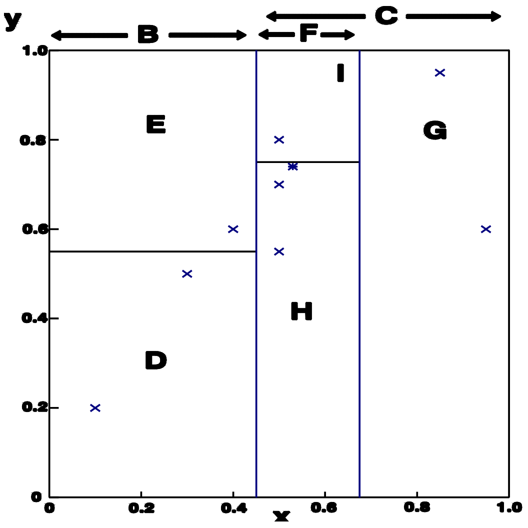


Figure 7.4 Terminal nodes partition the donor space

At the beginning of the search the set of n closest donors is empty. However, the set will acquire members and, as the search proceeds, may have some of the original members replaced by donors which are even closer to the recipient.

The search begins at node A, the root node. This is not a terminal node so the search proceeds to node C since the recipient's value for x is greater than .45 and so is in the range represented by node C. Node C is not a terminal node so the search continues. The recipient's value of x, the branching field, is less than .675 so the search goes to node F. Node F is not a terminal node so the search continues to node H because the recipient's value of y is less than .75. This is the first terminal node that has been encountered in the search so the set of n closest donors is empty. The parameter n is equal to two and there are two donors represented by node H so both donors are put into the set of closest donors.

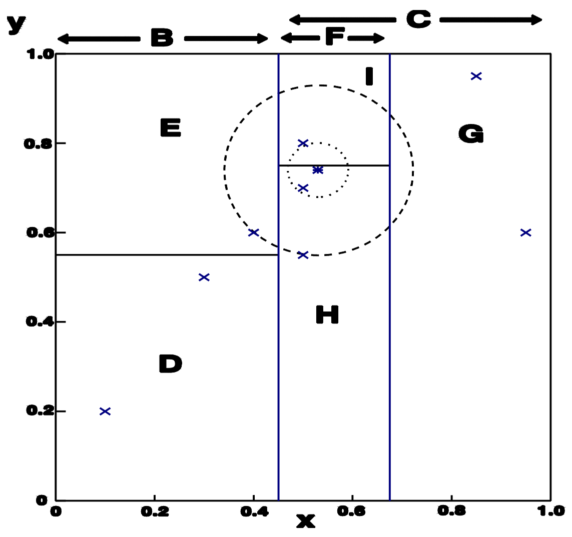


Figure 7.5 Use of the bounds-overlap-ball test

The next step is to decide whether or not there can be any donors at the opposite node, node I, which are closer to the recipient than the ones already selected. This decision is based on the results of a test called the "bounds-overlap-ball" test described by Friedman, Bentley and Finkel (1977). This test ensures that a node will not be searched if the closest possible donor represented by that node is farther away than the farthest donor which has already been selected in the group of n closest donors. The test determines if the bounds of the opposite node overlap a ball whose centre is the recipient and whose radius is the distance between the recipient and the farthest donor in the current group of n records. The donor in the current closest set which has the greatest distance from the recipient is donor 4 at a distance of .19. The dashed line representing a ball of radius .19 centred on the recipient at (.53, .74) overlaps the bounds of node I, as may be seen in Figure 7.5. In fact, there may be donors at node I which have a distance from the recipient of just over .01. This could occur if, for example, a donor had the same value of x as the recipient and a value of y slightly greater than .75. Thus, node I must be searched since it is possible to find a donor at that node which is closer to the recipient than the donors which are currently in the set of n closest donors.

Node I is a terminal node, so the distance between its single donor (donor 5) and the recipient is compared to those of the donors already in the set of n closest donors. This distance is .06 and is less than that of donor 4 so donor 5 is put into the set and donor 4 is dropped.

The search returns to node F and, since both node H and I have been visited, the next decision is whether or not to investigate node G. The bounds-overlap-ball is applied here, as was done at node H. The most distant in the group of n closest donors is now donor 5 which is at a distance of .06. In Figure 7.5, it may be seen that the dotted line representing a ball with radius .06 centred on the recipient does not overlap the bounds of node G. All donors at node G have values of x >.675 so all donors must be at least .675 -.530 =.145 away from the recipient, even if their y values are exactly the same as the recipient. Therefore it would be pointless to search node G.

The search goes back to node C and, since node F has been visited and node G has been tested, the next decision is whether or not to investigate node B. The most distant in the group of n closest donors is still donor 5 at a distance of .06. In Figure 7.5 it may be seen that the dotted line representing a ball with radius .06 centred on the recipient does not overlap the bounds of node B, so it is not necessary to search this branch of the tree. Donors belonging to node B must have x # .45 which places them at a distance of at least .08 from the recipient. This completes the search of the tree. Donors 5 and 6 are the members of the set of n closest donors.

When the group of n closest donors has been assembled, the closest donor is examined to ensure that it does not have FTE flags on fields to be imputed and to determine whether the recipient would pass the post imputation edits if the closest donor's values were transferred to the fields requiring imputation. If this is successful, the imputation is performed and the next recipient is processed. If the closest donor does not allow the recipient to pass the post imputation edits, then the next closest donor is tried. This continues until a donor is found or until all n donors in the group have proved unsuitable. If no suitable donor is found the procedure prints a message to that effect and goes to the next recipient.

In this small example, the branching field was always a member of the recipient's group of matching fields. In practice, there would likely be many cases in which this is not so. When this happens, the portions of the tree to the right and left of the node in question must both be searched because the branching field does not limit the suitable donors from the point of view of a recipient without that matching field. In this case, Banff always searches the left side on the way down the tree and the right side on the way back up.

Recipients With No Matching Fields

The k-d tree is not used to search for donors when the recipient record being processed has no matching fields. Instead, the user can choose to have a record selected at random from the donor population. If this option is not selected, Banff will not search for a donor. If this option is selected, a donor record is randomly selected and a check is done on the selected donor record to determine if there have been any fields identified as fields to exclude (FTE) which correspond to any of the recipient's fields to impute. If this is the case, the donor cannot be used and another donor is drawn randomly from the reduced donor population. Otherwise, if the record resulting from the imputation passes the post imputation edits, the imputation is performed.

While it might seem that a random search for a suitable donor could be quite long, this is unlikely to be the case. This is due to a particular feature of recipients which have no matching fields. Recall that when the acceptable values of the recipient record are substituted into the original edits, the edits which define the reduced set do not involve any variables other than those which require imputation. Since all donor records have passed the original edits, the fields to be transferred must be consistent with each other and will still be consistent after being transferred as a group into the recipient record. Nor can these fields cause other edits to fail since none of the acceptable fields are involved with them, given the actual values of that particular recipient.

This relationship is true only if the feasible region defined by the original edits is wholly contained in the feasible region defined by the post imputation edits. If there are portions of the original feasible region which are outside the post imputation region, then a donor which falls into one of these portions cannot be used successfully to impute for some recipients since the relationship required by the post imputation edits for the recipient is more restrictive than that which was required of the donor when it passed the original edits.

Post Imputation Edits

Banff places no constraints on the post imputation edits except that all variables referred to in the original edits must also be referred to in the post imputation edits, and vice versa, even if only in edits that bound the variables above or below. The post imputation edits are usually more relaxed than the original edits, although the user may choose to have them the same or even more restrictive. It is usual to replace equalities in the original edits by two inequalities which give an upper and lower bound on the total. Otherwise, for records which initially failed the equality edit, Donor Imputation will search until a donor is found which happens to have a value which will satisfy the equality exactly. This may result in the rejection of many potential donors which are close to the recipient and the eventual selection of a donor which is not close to the recipient, but happens to have that single required value. It is also quite possible that no donor would be found with the "right" value.

8. PROC ESTIMATOR – IMPUTATION ESTIMATORS

Purpose

This procedure imputes one variable at a time using a variety of imputation estimators. The user may choose from 20 pre-defined imputation estimator algorithms that are hard-coded into the system, or may specify their own custom-defined algorithms (the terms “imputation estimators” and “algorithms” are interchangeable here). There are two types of algorithms available in Banff: estimator functions and linear regression estimators. These algorithms may reference current and historical data. All historical data are assumed to be correct. The user may choose to base any of the required parameters (means or regression coefficients) on all acceptable values in the variable or may describe a subset of records to be used in the calculation of the parameters. Because the algorithms are applied independently to selected variables, the resulting imputed records may not pass the original edits. The user may reprocess the imputed records through the Error Localization procedure of Banff or their own error localization routine to determine if this has happened.

Types of Algorithms

Estimator functions are mathematical expressions involving current and/or historical values of some variables of the record being imputed, and current and/or historical means. The mathematical expressions may include parentheses and the arithmetic operators addition (+), subtraction (-), multiplication (*), division (/), and exponentiation (^).

Linear regression estimators are estimators which impute for variables using linear regression models of the general form

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{i1T_1}^{p_1} + \hat{\beta}_2 x_{i2T_2}^{p_2} + \hat{\beta}_3 x_{i3T_3}^{p_3} + \cdots + \hat{\beta}_m x_{imT_m}^{p_m} + \hat{\varepsilon}_i$$

where the T_j refer to current or historical periods, and the p_j are exponents. The variable y_i being imputed is the dependent variable in the model, and the auxiliary variables x_{ij} are the independent variables, or regressors. The $\hat{\beta}_j$ are the regression coefficients, the values of which are solved for by using the method of least squares. The $\hat{\varepsilon}_i$ is a random error term, which can be added to the model to introduce some variability into the fitted values of the y_i.

Description of the Method

The user must specify several options each time an imputation estimator algorithm is applied to a variable. Several variables may be processed in one execution of the procedure. More than one algorithm can be specified for a specific variable in that execution. If the first algorithm defined for a variable cannot provide an imputed value, it will try with the next algorithm.

Any parameters required by the algorithms are calculated before the procedure is executed, and the values of these parameters remain unchanged throughout the execution of the procedure. That is, the parameter values are not recalculated during execution of the procedure to reflect newly imputed variables. If the procedure is run a second time with different algorithms, all parameters required by the new second procedure are calculated and may reflect the imputation carried out in the previous procedure. This calculation of parameters is done each time a new procedure is to be executed, with all values imputed in previous runs of the procedure available for parameter calculation.

Specification of the Estimator

The following choices must be made for the application of Proc Estimator. The user must store this information in a SAS dataset, which is then accessed when the procedure is run.

- Algorithm Name, Field ID, Auxiliary Variables: these choices describe the basic imputation which is to be performed. The user specifies the imputation algorithm to be applied, the variable which is to be imputed and the auxiliary variables, if required by the selected algorithm.

- Weight Variable: if the selected algorithm uses means or is a linear regression, then the user may choose to name a variable to be used as a weight in the calculation of the parameters. The use of a weight variable is discussed further below.

- Variance Variable, Variance Exponent, Variance Period: The user may include a model variance variable in the definition of a linear regression estimator. This variable is taken into account in the calculation of the regression coefficients and the random error term; this is discussed in more detail below.

- Random Error Term: The user may also include a random error term in the estimator specification. This random error, or residual, is added to the value of the imputed variable in an attempt to create the same variability in the imputed values as in the non-imputed values.

- Exclusion from Calculation of Parameters: if the selected algorithm uses parameters, then the user may exercise some control over which records contribute to these parameters. Fields to be Excluded (FTE) fields may be omitted; fields which have already been imputed may be omitted, and other fields specified by the user may also be omitted. These options are discussed further below.

- Criteria for the Calculation of Parameters: the user may specify a minimum percentage and number of records which must be available for the parameters to be calculated. This is discussed further below.

Description of the Banff Pre-Defined Algorithms

Notation:	y_{iC}	is the field requiring imputation for the unit i at time C (the current period).
	y_{iH}	is the value of the field requiring imputation for the unit i at time H (the historical period).
	x_{iC}, x_{iH}	is the auxiliary variable for the unit i in the current or historical period, respectively.
	$u_{it}, v_{it}, w_{it}, z_{it}$	other auxiliary variables for the unit i at time t (either current or historical).
	$\bar{y}_C, \bar{y}_H$	are the means of the field requiring imputation taken over all eligible records in the current and historical files, respectively.
	$\bar{x}_C, \bar{x}_H$	is the mean of the auxiliary variable taken over all eligible records in the current and historical files, respectively.
	$\hat{\beta}_j$	is the jth regression coefficient in the regression model calculated over all eligible records in the current or historical files

Means and regression coefficients are based on all eligible records, i.e., the records which remain after user-specified exclusions have been made and after Banff has ensured that regression parameters or means in the numerator and denominator are based on the same records. Imputation is not performed if there are not enough records to satisfy the user-specified criteria for the calculation of parameters. Imputation is also not performed if the current or historical values of the auxiliary variables required by the algorithm are not valid (missing, needing imputation, or negative if applicable) for the record being imputed.

The following 20 algorithms are currently pre-defined in Banff. The format used to specify the algorithm within Banff is displayed for future reference; this same format will be utilized by the user when defining their own algorithms.

Estimator Functions

Algorithm:	AUXTREND
Equation:	$\hat{y}_{iC} = \frac{x_{iC}}{x_{iH}} y_{iH}$
Format:	aux1(c,v) * fieldid(h,v) / aux1(h,v)
Description:	The value from the previous survey for the same unit, with a trend adjustment calculated from an auxiliary variable, is imputed
Algorithm:	AUXTREND2
Equation:	$\hat{y}_{iC} = \frac{y_{iH}}{2}\left(\frac{u_{iC}}{u_{iH}} + \frac{v_{iC}}{v_{iH}}\right)$
Format:	fieldid(h,v) / 2 * (aux1(c,v)/aux1(h,v) + aux2(c,v)/aux2(h,v))
Description:	An average of two AUXTRENDs is imputed
Algorithm:	CURAUX
Equation:	$\hat{y}_{iC} = x_{iC}$
Format:	aux1(c,v)
Description:	The current value of a proxy variable for the same unit is imputed
Algorithm:	CURAUXMEAN
Equation:	$\hat{y}_{iC} = \bar{x}_C$
Format:	aux1(c,a)
Description:	The current average of a proxy variable is imputed
Algorithm:	CURMEAN
Equation:	$\hat{y}_{iC} = \bar{y}_C$
Format:	fieldid(c,a)
Description:	The mean value of all (user-defined) respondents for the current survey is imputed
Algorithm:	CURRATIO
Equation:	$\hat{y}_{iC} = \frac{\bar{y}_C}{\bar{x}_C} x_{iC}$
Format:	fieldid(c,a) * aux1(c,v) / aux1(c,a)
Description:	A ratio estimate, using values of all (user-defined) respondents from the current survey is imputed
Algorithm:	CURRATIO2
Equation:	$\hat{y}_{iC} = \frac{\bar{y}_C}{2}\left(\frac{u_{iC}}{\bar{u}_C} + \frac{v_{iC}}{\bar{v}_C}\right)$
Format:	fieldid(c,a)/2 * (aux1(c,v)/aux1(c,a) + aux2(c,v)/aux2(c,a))
Description:	An average of two CURRATIOs is imputed

Algorithm:	CURSUM2
Equation:	$\hat{y}_{iC} = u_{iC} + v_{iC}$
Format:	aux1 + aux2
Description:	The sum of two auxiliary variables from the current data table

Algorithm:	CURSUM3
Equation:	$\hat{y}_{iC} = u_{iC} + v_{iC} + w_{iC}$
Format:	aux1 + aux2 + aux3
Description:	The sum of three auxiliary variables from the current data table

Algorithm:	CURSUM4
Equation:	$\hat{y}_{iC} = u_{iC} + v_{iC} + w_{iC} + z_{iC}$
Format:	aux1 + aux2 + aux3 + aux4
Description:	The sum of four auxiliary variables from the current data table

Algorithm:	DIFTREND
Equation:	$\hat{y}_{iC} = \frac{\bar{y}_C}{\bar{y}_H} y_{iH}$
Format:	fieldid(c,a) * fieldid(h,v) / fieldid(h,a)
Description:	The value from the previous survey for the same unit, with a trend adjustment calculated from the difference of reported values for the variable, is imputed

Algorithm:	PREAUX
Equation:	$\hat{y}_{iC} = x_{iH}$
Format:	aux1(h,v)
Description:	The historical value of a proxy variable for the same unit

Algorithm:	PREAUXMEAN
Equation:	$\hat{y}_{iC} = \bar{x}_H$
Format:	aux1(h,a)
Description:	The historical average of a proxy variable for the same unit is imputed

Algorithm:	PREMEAN
Equation:	$\hat{y}_{iC} = \bar{y}_H$
Format:	fieldid(h,a)
Description:	The mean value from the previous survey of all (user-defined) respondents is imputed

Algorithm:	PREVALUE
Equation:	$\hat{y}_{iC} = y_{iH}$
Format:	fieldid(h,v)
Description:	The value from the previous survey for the same unit is imputed

Linear Regression Estimators

Algorithm:	CURREG
Equation:	$\hat{y}_{iC} = \hat{\beta}_0 + \hat{\beta}_1 x_{iC}$
Format:	intercept, aux1(c)
Description:	A simple linear regression based on one independent variable from the current data table

Algorithm:	CURREG_E2
Equation:	$\hat{y}_{iC} = \hat{\beta}_0 + \hat{\beta}_1 x_{iC} + \hat{\beta}_2 x_{iC}^2$
Format:	intercept, aux1(c), aux1(c)ϖ2
Description:	A regression based on the value and the squared value of a variable from the current data table

Algorithm:	CURREG2
Equation:	$\hat{y}_{iC} = \hat{\beta}_0 + \hat{\beta}_1 u_{iC} + \hat{\beta}_2 v_{iC}$
Format:	intercept, aux1(c), aux2(c)
Description:	A linear regression based on two independent variables from the current data table

Algorithm:	CURREG3
Equation:	$\hat{y}_{iC} = \hat{\beta}_0 + \hat{\beta}_1 u_{iC} + \hat{\beta}_2 v_{iC} + \hat{\beta}_3 w_{iC}$
Format:	intercept, aux1(c), aux2(c), aux3(c)
Description:	A linear regression based on three independent variables from the current data table

Algorithm:	HISTREG
Equation:	$\hat{y}_{iC} = \hat{\beta}_0 + \hat{\beta}_1 y_{iH}$
Format:	intercept, fieldid(h)
Description:	A linear regression based on the historical value of the variable to impute

User-Defined Algorithms

The user may choose to define one or more of their own imputation estimator algorithms, rather than only selecting only from among the 20 pre-defined algorithms. The definition of these algorithms must be done before the user specifies to Banff which of the pre-defined and user-defined algorithms are to be used for imputation, i.e. before the specification of the procedure is done as described earlier.

Several options must be specified when the user is defining their own algorithm. The user must store this information in a SAS dataset, which is then accessed when the procedure is run.

- Algorithm Name: the user can uniquely identify the algorithm that they are creating in this field by specifying a name with alphanumeric characters.

- Algorithm Type: the type of algorithm is identified in this field as being either an estimator function (EF) or a linear regression (LR).

- Algorithm Status: the user specifies an alphanumeric status code that they want to associate with the algorithm. This status code is used to update the field status table by recording the imputation method which was used to impute a field. Note that the system automatically adds an 'I', for imputation, in front of the code.

- Algorithm Formula: the algorithm is defined using a combination of variable names, pre-defined keywords, and mathematical operators. The syntax of the definition is dependent on the type of the algorithm.

- Description: comments about the algorithm can be entered in this field.

Weighted and Unweighted Parameter Calculations

The user may specify a variable to be used as a weight in the calculation of the parameters of one or more estimators. This would be appropriate if, for example, records in the data group being imputed had different probabilities of selection in the original sampling plan. If a weight variable is chosen, then any means used by that particular algorithm are weighted means, calculated as follows.

$$\bar{y}_C = \frac{\sum_i w_{iC} y_{iC}}{\sum_i w_{iC}}, \qquad \bar{y}_H = \frac{\sum_i w_{iH} y_{iH}}{\sum_i w_{iH}}$$

Similarly, this weight variable is taken into account in the calculation of the regression coefficients.

The specified weight may be any variable on the current data SAS dataset. It is the user's responsibility to ensure that the weight field is valid; Banff will terminate execution before attempting any imputation if any values are negative (if applicable) or missing. If the chosen algorithm requires a historical mean as well as a current mean, then the same variable is automatically used as the weight in both time periods. To calculate the historical mean, the weight values are taken from the corresponding variable in the historical data SAS dataset.

Model Variance Variable in Linear Regressions

The user may include a model variance variable in the definition of a linear regression estimator. This variable is taken into account in the calculation of the regression coefficients and the random error term. The model variance variable is related to the variable being imputed in the sense that the variance of the imputed variable is proportional to the model variance variable. That is, the variance for the variable being imputed is estimated to be $v_i\sigma^2$, where v_i is the model variance variable for record i, and σ^2 is the population variance. The variance σ^2 is assumed to be the same for all records in the data group, but is not required to be known. It is the user's responsibility to ensure that the model variance field is valid; if any values are missing or not greater than zero, Banff will terminate processing before attempting any imputation.

Random Error Term

The user may also include a random error term in the estimator specification, for both estimator functions and linear regression estimators. This random error, or residual, is added to the value of the imputed variable in an attempt to create the same variability in the imputed values as in the non-imputed values.

If the random error is specified, Banff will first calculate the value for the variable to be imputed, according to the specified estimator. Then, Banff will randomly select one of the eligible records which contributed to the parameter calculations. The random selection will incorporate the record weight in the probability of selection if the weight is to be included in the imputation step calculations. The residual for the selected eligible record will be calculated, which is the difference between the actual reported value and the value estimated by the estimator. As well, the model variance variable will be included in the calculation of the random error if a model variance variable is included in the definition of a linear regression estimator.

If the residual for the randomly selected record J is denoted by R_J, and the model variance variable by v, then the random error added to the imputed value y_i is:

$$\hat{\varepsilon}_i = R_J \sqrt{\frac{v_{iT_{m+1}}^{p_{m+1}}}{v_{JT_{m+1}}^{p_{m+1}}}}$$

Note that if no model variance variable is included in the estimator, then the random error added to the imputed value is simply the residual R_J from the randomly selected record J.

<u>Parameter Calculation in Linear Regression</u>
As noted earlier, the general form of the regression model used in specifying linear regression algorithms is

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{i1T_1}^{p_1} + \hat{\beta}_2 x_{i2T_2}^{p_2} + \hat{\beta}_3 x_{i3T_3}^{p_3} + \cdots + \hat{\beta}_m x_{imT_m}^{p_m} + \hat{\varepsilon}_i$$

where the T_j refer to current or historical periods, and the p_j are exponents. The variable y_i being imputed is the dependent variable in the model, and the auxiliary variables x_{ij} are the independent variables, or regressors.

The parameters in a linear regression algorithm are the regression coefficients, denoted by $\hat{\beta}_j$. The values of these parameters are calculated by applying the method of least squares. Suppose that the values of the independent variables x_{ij}, raised to the appropriate exponent, are represented in the matrix **X** as follows:

$$\mathbf{X} = \begin{pmatrix} 1 & x_{11T_1}^{p_1} & x_{12T_2}^{p_2} & x_{13T_3}^{p_3} & \cdots & x_{1mT_m}^{p_m} \\ 1 & x_{21T_1}^{p_1} & x_{22T_2}^{p_2} & x_{23T_3}^{p_3} & \cdots & x_{2mT_m}^{p_m} \\ 1 & x_{31T_1}^{p_1} & x_{32T_2}^{p_2} & x_{33T_3}^{p_3} & \cdots & x_{3mT_m}^{p_m} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1T_1}^{p_1} & x_{n2T_2}^{p_2} & x_{n3T_3}^{p_3} & \cdots & x_{nmT_m}^{p_m} \end{pmatrix}$$

The hth row of **X** represents the hth eligible record for the calculation of the regression coefficients, and the number of rows in **X** is the number of eligible records. The first column of **X** is associated with the intercept parameter $\hat{\beta}_0$. If the intercept is not included in the regression model, then this first column is excluded from the matrix **X**. The other m columns of **X** are associated with the independent auxiliary variables x_{ij}. There are as many of these columns in **X** as there are independent variables specified in the regression model.

For each of the eligible records, the current values for the variable being imputed are represented by the column vector **Y**:

$$\mathbf{Y} = \begin{pmatrix} y_{1C} \\ y_{2C} \\ \vdots \\ y_{nC} \end{pmatrix}$$

If a model variance variable v has been included in the definition of the estimator being processed, or a weight variable was included in the specification of the imputation estimators, these are taken into account in the diagonal matrix **D**:

$$\mathbf{D} = \begin{pmatrix} \frac{w_{1C}}{v_{1T_{m+1}}^{p_{m+1}}} & 0 & 0 & \cdots & 0 \\ 0 & \frac{w_{2C}}{v_{2T_{m+1}}^{p_{m+1}}} & 0 & \cdots & 0 \\ 0 & 0 & \frac{w_{3C}}{v_{3T_{m+1}}^{p_{m+1}}} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \frac{w_{nC}}{v_{nT_{m+1}}^{p_{m+1}}} \end{pmatrix}$$

Note that if no weight variable w is specified, then the value of the weight variable defaults to w_{hC}=1 for each eligible record. Similarly, if no model variance variable is specified, then $v_{hT_{m+1}}^{p_{m+1}} = 1$ for each eligible record. The column vector of the regression coefficients is denoted by:

$$\hat{\beta} = \begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \hat{\beta}_2 \\ \vdots \\ \hat{\beta}_m \end{pmatrix} \text{ if an intercept is required, } \hat{\beta} = \begin{pmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \\ \hat{\beta}_3 \\ \vdots \\ \hat{\beta}_m \end{pmatrix} \text{ if no intercept is required}$$

The values of the regression coefficients are obtained by solving the linear system

$$(\mathbf{X'DX})\hat{\beta} = \mathbf{X'DY}$$

The solution is provided by:

$$\hat{\beta} = \left(\mathbf{X'DX}\right)^{-1}\mathbf{X'DY}$$

Exclusion from the Calculation of Parameters

When parameters are required by algorithms, they are calculated each time Proc Estimator is executed before any imputation is actually carried out. All algorithms specified for the procedure then use these calculated parameters; the parameter values do not change during the execution of the procedure to reflect newly-imputed values. In addition to choosing a weight variable, the user may influence the parameters by choosing to include or exclude three types of values when the parameters are calculated. These three types of values are:

- Fields to Exclude(FTE) fields,
- fields which have already been imputed and
- all variables on other records which are identified by the user on the current or historical data SAS datasets

In addition to user-specified exclusions, Banff may automatically exclude some records so that the parameters of an algorithm are based on the same records. Each of these four types of exclusion is discussed in turn.

The Field to Exclude (FTE) code which may be used by the Donor Imputation procedure to exclude values from being donated may also be used to exclude records from contributing to the parameters in algorithms. FTE fields are not extreme enough to require imputation, but are sufficiently unusual that the user may wish to omit them from the calculation of the parameters. Excluding FTEs refers only to FTE codes in the field to be imputed or in the auxiliary variables; FTE codes in any other fields are ignored. See Outlier Detection (Section 4) for a description of how FTE fields are identified.

When a parameter is calculated, there may be fields which have already been imputed by the Donor Imputation procedure or by another imputation estimator. The user may choose to exclude such fields from contributing to the parameters. Excluding these fields ensures that only original respondent data are used for the parameters. Some users view this as an important advantage; others prefer to base the parameters on as many records as possible. A disadvantage of excluding imputed data is that relying only on data which originally passed the edits may distort the parameter values. For example, suppose that a high percentage of large units passed the edits while a low percentage of small units passed. If some imputation has been done, and if it has preserved the relationship of large and small units, then using all available data should produce means closer to the overall "true" means. Excluding the imputed fields in this case would produce means which are closer to the behaviour of the larger units. Note that values in fields imputed by the Deterministic Imputation procedure are treated as original data.

The user may also specify an “exclusion variable” in the current and/or historical data SAS datasets which flags entire records that should be excluded from contributing to the parameters. The exclusion variable remains in effect for the entire execution of the procedure. The user can specify the same exclusion variable or a different one for the next application of the procedure.

In addition to the exclusions specified by the user, Banff may automatically exclude some individual fields from contributing to the parameters so that the means in the numerator and denominator of certain estimator functions, or the regression parameters in a linear regression, are based on the same set of records. This further reduces the number of respondent values which are available to contribute to the parameter calculation.

<u>Criteria for Calculation of Parameters</u>
If the algorithm involves parameters, the user must specify a required number and percentage of eligible records which must be available for the parameters to be calculated. The number and percentage are calculated after all exclusions have been performed, so the expected number of exclusions must be taken into account when specifying the criteria. If the criteria are not satisfied, Banff will not proceed with the calculation of parameters and will print a message to that effect.

<u>Calculation of Imputation Estimators – First Execution of Proc Estimator</u>
Consider the following example.

Suppose the user has decided that for the first application of Proc Estimator that they would like to use an unweighted linear regression estimator of the form: $y_{iC} = \beta_0 + \beta_1 x_{iC}$. Note that this is the same as the system-defined algorithm CURREG. If this estimator were not available as a system-defined algorithm, the user would have to define it as follows:

Algorithm:	CURREG
Algorithm Type:	LR
Algorithm Status:	LR1
Format:	Intercept, aux1(c)
Description:	A simple linear regression based on one independent variable from the current data table

The user proceeds with the specification of this estimator. Note that no weight variable is specified and that an exclusion variable is present in the current data file. A random error term will be added to the imputed value.

		Exclude Outliers (Y/N):	Y
Field to be Imputed:	y	Exclude Imputed (Y/N):	Y
Algorithm:	CURREG	Exclude Records From	
Auxiliary Variable:	x	Current File Where:	z = 0
Random Error (Y/N):	Y	Accept Negative(Y/N):	N

Current File – Before Imputation

Ident	w	x		y		z	EXCL
R01	10	99	FTE	4		200	
R02	10	4		- 1	FTI	150	
R03	10	4		7		250	
R04	10	-2		9	IDN	200	
R05	10	2		5		0	E
R06	10	3	FTI	8		100	
R07	5	6		12		500	
R09	5	6		14		600	
R10	5	-1		- 1	FTI	500	

Suppose that the variable y is to be imputed by the CURREG algorithm with x as the auxiliary variable. The user has chosen to exclude FTEs, to exclude imputed values and also wants to exclude from the calculation of parameters any records with z=0 in the data file. Also, the user has chosen not to accept negative data. So, several records will not contribute to the calculation of the parameters, in this case the regression coefficients. Because the calculation of the coefficients in this linear regression depend on both variables x and y of the current file, only the records containing valid data for these two fields will be retained.

In this example, record R01 cannot be used because x has an FTE flag. The value of y in record R02 is one of the fields which requires imputation, so record R02 will not be used in the calculation of the parameters. Record R03 contributes to the parameters. Record R04 cannot be used for two reasons: first, it has a field (y) which was imputed by donor imputation but the user wants to exclude imputed values from the parameter calculations, and second, the value of x is negative but the user has chosen to consider negative values invalid. The value of z in record R05 satisfies the exclusion condition (exclusion variable EXCL='E') and so both the x and y values of that record are also excluded from the parameters. The variable x has FTI status in Record R06, so R06 will not contribute to the parameters. Records R07 and R09 contribute to the parameters. Record R10 requires imputation for y so it cannot be used in the parameter calculations. Also note that the variable y cannot be imputed for R10 in this step because the algorithm requires the auxiliary variable x to have a valid value available but the user has chosen to reject negative values.

With the regression model specified, the method of least squares solutions for the regression coefficients are:

$$\hat{\beta}_0 = \bar{y}_C - \hat{\beta}_1 \bar{x}_C \ , \quad \hat{\beta}_1 = \frac{\sum (x_{iC} - \bar{x}_C)(y_{iC} - \bar{y}_C)}{\sum (x_{iC} - \bar{x}_C)^2} \ , \qquad \hat{\varepsilon}_i = y_{JC} - \hat{\beta}_0 - \hat{\beta}_1 x_{JC}$$

The unweighted current means for y (the variable to be imputed) and x (the auxiliary variable) are calculated over the three eligible records as follows.

$$\bar{y}_C = \frac{7+12+14}{3} = \frac{33}{3} = 11$$
$$\bar{x}_C = \frac{4+6+6}{3} = \frac{16}{3} = 5\tfrac{1}{3}$$

Then, the parameters are calculated to be $\hat{\beta}_0 = -5$ and $\hat{\beta}_1 = 3$. Suppose that Banff had randomly chosen R07 as the record which will donate its residual to R02. Then the residual for R02 is:

$$\hat{\varepsilon}_{R02} = \hat{\varepsilon}_{R07} = 12 - (-5) - (3 \times 6) = -1$$

Finally, the value of y to be imputed for record R02 using the CURREG algorithm with a random error term would be:

$$\hat{y}_{R02C} = \hat{\beta}_0 + \hat{\beta}_1 x_{R02C} + \hat{\varepsilon}_{R02} = -5 + (3 \times 4) - 1 = 6$$

Calculation of Imputation Estimators – Second Execution of Proc Estimator, First Estimator

Now suppose that the user wants to specify the imputation of y with the pre-defined Difference Trend (DIFTREND) algorithm in a second application of Proc Estimator. The user has chosen to use a weight variable, but not to add a random error term to the imputed value. An auxiliary variable is not required for this algorithm. Records with z = 0 in the current data table are excluded. As well, the user has chosen to exclude outliers, but has decided to allow imputed data to contribute to the means. Also, the user has chosen to accept negative values as valid values. Note that the y value of record R02 has been imputed by the CURREG algorithm in the first execution of the procedure. The code ILR1 (imputation by CURREG) indicates the method used for imputation.

Field to be Imputed:	y	Exclude Imputed (Y/N):	N
Algorithm:	DIFTREND	Exclude Records From	
Weight Variable:	w	Current File Where:	z = 0
Random Error (Y/N):	N	Accept Negative (Y/N):	Y
Exclude Outliers (Y/N):	Y		

Current File								Historical File				
Ident	w	x		y		z	EXCL	Ident	w	x	y	z
R01	10	99	FTE	4		200		R01	8	3	6	125
R02	10	4		6	ILR1	150		R02	8	2	7	100
R03	10	4		7		250						
R04	10	-2		9	IDN	200		R04	8	2	4	0
R05	10	2		5		0	E	R05	8	1	4	150
R06	10	3	FTI	8		100		R06	8	2	7	300
R07	5	6		12		500		R07	4	7	12	200
								R08	4	4	8	175
R09	5	6		14		600		R09	4	5	10	200
R10	5	-1		-1	FTI	500		R10	4	-7	14	250

Here there are two means to calculate: y in the current data table and y in the historical data table. For this algorithm, if a record cannot contribute to one mean, then it cannot contribute to the other because means in the numerator and denominator must be based on the same records.

In this case, record R01 is used because the FTE is not in one of the variables for which a mean is required. The user has chosen to accept imputed data in the means, so R02 can be included because its current file value for y was imputed in the previous execution of Proc Estimator. R03 is not used because there is no corresponding historical record and the numerator and denominator must be based on the same records. R04 can be used because the user has allowed imputed data to be used in the calculation of parameters, and the value for y was imputed by the donor imputation procedure before this procedure was initiated. Note that the value of z for R04 in the historical table is z = 0, but the exclusion condition only specifies z=0 in the current data table. R05 is not used because the value of z in the current file satisfies the exclusion condition (exclusion variable EXCL=’E’). Record R08 is not used because there is no corresponding current record. Records R06, R07 and R09 are used. The record R10 is not used because of the FTI (Field to Impute) flag in y.

The current and historical means for y, the variable to be imputed, using the weight variable w, are calculated over the five eligible records as follows.

$$\bar{y}_C = \frac{(10\times 4)+(10\times 6)+(10\times 9)+(10\times 8)+(5\times 12)+(5\times 14)}{10+10+10+10+5+5} = \frac{400}{50} = 8$$

$$\bar{y}_H = \frac{(8\times 6)+(8\times 7)+(8\times 4)+(8\times 7)+(4\times 12)+(4\times 10)}{8+8+8+8+4+4} = \frac{280}{40} = 7$$

The value of y to be imputed for record R10 using the Difference Trend estimator would be:

$$\hat{y}_{R10C} = \frac{\bar{y}_C}{\bar{y}_H} y_{R10H} = \frac{8}{7}\times 14 = 16\,.$$

Calculation of Imputation Estimators – Second Execution of Proc Estimator, Second Estimator
Now suppose that in the second execution of Proc Estimator that the user also wants to impute for another variable, this time performing the imputation of x using the Current Ratio (CURRATIO) algorithm, which is included in the group of algorithms already defined by the system. The user must be aware that all estimators in a single execution of Proc Estimator must share the same exclusion condition. Since this is not a new execution of the procedure, the user must use the same exclusion condition as for the imputation of y with the DIFTREND estimator. So once again any records with z = 0 in the current data table will be excluded. The user has decided to again include a weight variable and to accept negative values. A random error term will be added to the imputed value. The user has also decided to exclude outliers, but will allow previously imputed values to contribute to the parameters. Note that the y value of record R10 is imputed by the Difference Trend estimator in the second (current) execution of Proc Estimator. The code IDT (Imputation by Difference Trend) indicates the method used for imputation.

Field to be Imputed:	x	Exclude Outliers (Y/N):	Y
Algorithm:	CURRATIO	Exclude Imputed (Y/N):	N
Auxiliary Variable:	y	Exclude Records From	
Weight Variable:	w	Current File Where:	z = 0
Random Error (Y/N):	Y	Accept Negative (Y/N):	Y

Current File

Ident	w	x		y		z	EXCL
R01	10	99	FTE	4		200	
R02	10	4		6	ILR1	150	
R03	10	4		7		250	
R04	10	-2		9	IDN	200	
R05	10	2		5		0	E
R06	10	3	FTI	8		100	
R07	5	6		12		500	
R09	5	6		14		600	
R10	5	-1		16	IDT	500	

Historical File

Ident	w	x	y	z
R01	8	3	6	125
R02	8	2	7	100
R04	8	2	4	0
R05	8	1	4	150
R06	8	2	7	300
R07	4	7	12	200
R08	4	4	8	175
R09	4	5	10	200
R10	4	-7	14	250

Here there are two means to calculate: x in the current data table and y in the current data table. Similar to the previous example, if a record cannot contribute to one mean, then it cannot contribute to the other because means in the numerator and denominator must be based on the same records.

Record R01 cannot be used in the means calculation because it has an FTE status in the x variable. Because the user has chosen to allow imputed values to contribute to the parameters in this step and the y value for record R02 was imputed in the previous execution of Proc Estimator, R02 is used. R03 is used. R04 can be used because its current value for y was previously imputed by the donor imputation procedure. Record R05 cannot be used since it has a value z=0 in the current file, which satisfies the exclusion condition (exclusion variable EXCL='E'). Record R06 is not used because its current value for x is being imputed and has FTI status. Record R08 is not used because there is no corresponding current record. R07 and R09 are used. The record R10 is not used because its current value for y is imputed during the active (second) execution of the procedure. That is, all parameter values are calculated at the beginning of the execution of the procedure, before any imputation actually takes place. Only values imputed in previous executions of the procedure are taken into account. The parameters are not recalculated to take into account data that have been imputed during the active execution of the procedure.

The current means for x, the variable to be imputed, and y, the auxiliary variable, using the weight variable w, are calculated over the five eligible records as follows:

$$\bar{x}_C = \frac{(10\times 4)+(10\times 4)+(10\times -2)+(5\times 6)+(5\times 6)}{10+10+10+5+5} = \frac{120}{40} = 3$$

$$\bar{y}_C = \frac{(10\times 6)+(10\times 7)+(10\times 9)+(5\times 12)+(5\times 14)}{10+10+10+5+5} = \frac{350}{40} = 8.75$$

Suppose that Banff had randomly chosen R03 as the record which will donate its residual to R06. Then the residual for R06 is:

$$\hat{\varepsilon}_{R06} = \hat{\varepsilon}_{R03} = x_{R03C} - \frac{\bar{x}_C}{\bar{y}_C} y_{R03C} = 4 - \frac{3}{8.75}\times 7 = 1.6$$

Finally, the value of x to be imputed for record R06 using the Current Ratio estimator with a random error term would be:

$$x_{R06C} = \frac{\bar{x}_C}{\bar{y}_C} y_{R06C} + \hat{\varepsilon}_{R06} = \frac{3}{8.75}\times 8 + 1.6 = 4.34$$

9. PROC PRORATE – PRORATING

Purpose

This procedure will prorate each record according to an equality edit in an attempt to ensure that the components of a sum add up to the desired total within each record. It incorporates features which allow it to discriminate between previously imputed and original data and therefore can be applied as a post-imputation procedure, as well as a stand-alone treatment. For each case where a summation does not match the total, the equation in question is submitted one of the two available prorating algorithms in order to rake the components to match the total. It is then submitted to a rounding algorithm to ensure the variables involved are output with the proper number of decimals.

Description of the Method

The user supplies a collection of equality edits to be prorated, contained within an edit group. Only addition operators may be used within the summation. Edit rules are allowed to be nested (i.e. dependent) within the edit group, but they can not be independent. Variables may only appear once on the left-hand side (in a summation) and subsequently once only on the right-hand side (as a fixed total). Example:

Edit Group1
tot1 + tot2 = grandtotal
a + b + c = tot1
d + e + f = tot2

The user may specify the edits in whatever order they want, as long as they are logical, since Banff contains a parser which determines the hierarchy. Note that for the edit group above, the second and third edits form a subset of the first edit, as "tot1" and "tot2" are subtotals which sum to the overall total "grandtotal".

The edits are applied in the hierarchical order to each record within the data group and if the equalities are not satisfied, the prorating and rounding algorithms are executed. When prorating elements of an equation, the total to the right of the equal sign will not be adjusted.

The user can specify whether only previously imputed variables should be eligible for prorating, only original data, or both imputed and original. This feature is implemented by reading the status codes of the variables involved as found in the input field status SAS dataset, which is provided by the user or is taken from a previously-executed Banff procedure. Any variable with a status code beginning with "I" (unique exception: "IDE" for deterministic imputation) will be considered an imputed variable. A status code of any other kind, or the non-presence of the variable in the input field status dataset will be taken to indicate the variable is original data. Variables not eligible for prorating will be removed from the summation and the total adjusted accordingly before prorating begins. Similarly, zero values will also be removed at this point as they cannot possibly be modified by the process.

The user also has the option of specifying weights for every component involved in a summation. In this way, the relative amount of change of each component due to prorating may be controlled, change being inversely proportional to the weight given. Weights are applied at the component level; i.e., the same weight applies across all records for that component.

Syntax Edit Checks

Before prorating begins, a series of edit checks are imposed by the system to ensure that vital syntax rules are not violated. Examples of this include verifying the hierarchical structure of the edits and the positivity of weights. If such an error is found, prorating will not be run and an appropriate error message will be produced instead.

Prorating Algorithms

In the event of a summation not being equal to the total, e.g., $x_1 + x_2 + \cdots + x_n \neq y$, a prorating algorithm will be applied. There are two prorating algorithms available to the user, the **basic** method and the **scaling** method.

When all components to be adjusted and their corresponding total have the same sign, the basic and scaling methods will give equivalent results when all else is equal, i.e., same parameter values, options, weights, etc.

Basic Method Algorithm

With the basic method, the adjusted value for each of the m_h eligible variables x_{hi} in an edit equation h (h=1,…,r) is calculated as:

$$x'_{hi} = x_{hi} + \frac{x_{hi}}{w_{hi}} \times \frac{y_h - \sum_{j=1}^{m_h} x_{hj}}{\sum_{j=1}^{m_h} \left(x_{hj}/w_{hj}\right)} \qquad i = 1, 2, 3, ..., m$$

where y_h is the fixed total for the edit equation h and the w_{hi} is the weight associated with x_{hi}.

Unweighted basic method prorating is simply a special case of weighted prorating where all w_{hi}=1:

$$x'_{hi} = \frac{y_h}{\sum_{j=1}^{m_h} x_{hj}} x_{hi}$$

With the basic method, it is possible that data values could change sign (i.e., from positive to negative, and vice-versa) unless this situation is prevented from occurring through proper specification of the Lower Ratio bound parameter, which is discussed below. Also, if the user prevents this change of sign from occurring and there is a mixture of positive and negative values, it is possible that even though the overall adjustment to the components is required to be positive (or negative) for the sum of the components to equal the fixed total, there could be some components that have positive adjustments made to them while others have negative adjustments made, depending on their signs and the sign of the total.

Scaling Method Algorithm

First, the scaling factor k for an edit equation h (h=1,…,r) is calculated across the m_h variables eligible to be prorated x_{hi} as:

$$k_h = \frac{\sum_{i=1}^{m_h} x_{hi} - y_h}{\sum_{i=1}^{m_h} | x_{hi}/w_{hi} |} \qquad h = 1, 2, 3, ..., r$$

where y_h is the fixed total for the edit equation h and the w_{hi} are the weights associated with the x_{hi}.

When -1=< k_h <=1, the prorated value of x_{hi} for the edit equation h is given as follows:

$$\begin{aligned} x'_{hi} &= (1 - k_h/w_{hi}) x_{hi} \quad \text{if } x_{hi} \rangle 0 \\ x'_{hi} &= (1 + k_h/w_{hi}) x_{hi} \quad \text{if } x_{hi} \langle 0 \end{aligned} \qquad i = 1, 2, 3, ..., m$$

It is very extreme that k>1 or k<-1 happens, for example in a bizarre case such as when the prescribed total y is a positive value, but all the x_i are negative. This indicates a problem that should be identified and handled during an editing process before prorating. If the calculation shows k>1 or k<-1, prorating for that record will not be executed, and a warning message given instead.

Under the scaling method, it is not possible for the sign of the component to change; negative values can never become positive, and vice versa. Also, if the net change to the components is positive (negative), then all changes to the prorated components will be positive (negative).

Rounding Algorithm

After the selected prorating algorithm has been applied to an equation, it is possible that many extra decimal places were needed to enable the summation to equal the total. Therefore it is also necessary to round the data involved to the number of decimal places desired by the user, and yet still ensure that the newly-created equality is not broken. The rounding algorithm adjusts all fields to the correct number of decimal places and ensures the summation still equals the total.

For the equation h:

Step 1. Round the x'_{hi} to d+1 decimal places; d=number of decimal places specified by the user.

Step 2. For i=1, round x'_{h1} up or down to x''_{h1} (with d decimals).

Step 3. For i>1, round u_i:

$$u_{hi} = x'_{hi} + \sum_{j=1}^{i-1} \left[x'_{hj} - x''_{hj} \right]$$

to the final prorated value x''_{h1}.

When the user specifies d=number of decimals places required, three conditions must always be met:

- d must be less than or equal to the number of decimals places existing in the data table
- d must be greater than or equal to the number of decimals in the total
- d must be within the interval [0,9].

The default value for d is zero.

After all the x_{hi} have been prorated and rounded for the first equation h=1, the procedure continues on in the hierarchical order for each of the other r-1 equations, taking into account the newly prorated values for any of the x_{hi} which may serve as the fixed total y_h in subsequent equations.

<u>Example</u>
Consider the following data submitted to prorating using the scaling method. Using Edit Group 1 (same as above), suppose the user requests all prorated variables be rounded to zero decimal places, and in addition specifies weights as well as the types of variables to impute as follows:

Edit Group 1	**Component Weights**	**Variables to Prorate:**
tot1 + tot2 = grandtotal	1,1	All
a + b + c = tot1	1,1,2	Imputed data only
d + e + f = tot2	1,1,1	All

For instance, variable "c" (weight =2) would have a relative prorating adjustment (if necessary) HALF as great as those for variables "a" (weight = 1) and "b" (weight = 1) due to its weight being twice as large.

The user has also specified that for the first edit specified, all variables are eligible for prorating; whereas for the second edit only variables specified as containing previously imputed data are eligible, and for the third edit once again all variables are eligible. These statuses are detailed below in the user's input field status table. The data to be prorated also appear below. The user has specified that negative data is valid.

Input dataset:

Key Value	**A**	**B**	**C**	**D**	**E**	**F**	**TOT1**	**TOT2**	**GRANDTOTAL**
REC001	6	4	-4	10	9	9	12	24	31

Input field status dataset:

Key Value	**FIELDID**	**STATUS**
REC001	A	IDN
REC001	C	IDT
REC001	E	IMP

With this input field status dataset the user has identified all variables that have been previously imputed. In Banff, all status codes beginning with "I" (except "IDE" for deterministic imputation) will be considered to indicate previously imputed data. Prorating will assume that any other variable contains original data.

Both “tot1" and “tot2" are eligible for prorating according to the variables to prorate specification and the input field status dataset, and also both have equal weights (=1), therefore the Prorating procedure calculates the scaling value k over the m=2 eligible components as:

$$k = \left(\sum_{j=1}^{2} x_j - y \right) \Big/ \left(\sum_{j=1}^{2} | x_j / w_j | \right) = (12 + 24 - 31)/(12 + 24) = 5/36 .$$

Then, the prorated values for TOT1 and TOT2 are calculated as:

$$TOT1' = (1 - k / w_{TOT1})(TOT1) = (1 - 5/36)(12) = 10.33$$
$$TOT2' = (1 - k / w_{TOT2})(TOT2) = (1 - 5/36)(24) = 20.67$$

The rounding algorithm is applied next, producing:

TOT1	**TOT2**	**GRANDTOTAL**
10	21	31

Continuing with the next level in the hierarchical order, prorating of the second and third edits gives us:

Key Value	**A**	**B**	**C**	**TOT1**	**D**	**E**	**F**	**TOT2**
REC001	9	4	-3	10	7.5	6.75	6.75	21

So due to weighting, variable “a” increased by 50% of its original value, which was twice as much as variable “c”, which increased by only 25% of its original value. Variable “b” was <u>not</u> previously imputed and therefore ineligible for prorating for this particular record, and so did not contribute to the calculation of the scaling factor k. All three of the variables d, e, and f contributed to the calculation of the scaling factor k in the third prorating edit since the user specified that all variables were eligible to be prorated.

Finally, submitting these two equations to the rounding algorithm results in:

Key Value	**A**	**B**	**C**	**TOT1**	**D**	**E**	**F**	**TOT2**
REC001	9	4	-3	10	8	6	7	21

<u>Order of Edit Variables</u>
Although the user may specify the edits in any order, various ways of ordering the variables in an edit may produce different results after applying the prorating algorithm. Sometimes the excess of the total can not be evenly distributed among the summation variables, and only some of them receive a share of this difference. In this case, a change in variable order may result in a change to which variables receive this distributed difference. For example, using the edit x + y = z, the distributed difference is 1 for the following sum to be prorated: 1 + 1 = 3. In this example, if the parameter DECIMAL=0, only the first variable x will receive the difference, and the prorated sum will become 2 + 1 = 3. If this edit is changed to y + x = z, the variable y will be increased by one. Using this example again, if the first variable has a higher weight than the second variable, the second variable will receive the difference, i.e. change is inversely proportional to the weight given, and the prorated sum will become 1 + 2 = 3. In conclusion, the order of the variables may have an impact on the prorating results.

<u>Bound Limitations</u>
Another option the user has control over is the constraint on relative change of the variables. This is denoted in the form of **Upper Ratio** and **Lower Ratio** bounds on change. For example, if the

user does not want the final value of any variable to increase by any more than 25% from its original value after prorating and rounding have successfully completed, they would specify an Upper Ratio bound = 1.25. It is possible for a variable to be prorated to a zero value if the Lower Ratio bound is set to 0. If the basic method is being applied and the user does not want any variables to change sign as a result of the prorating process, this could be achieved by setting the Lower Ratio bound greater than or equal to 0. By default, the Lower Ratio bound when applying the scaling method cannot be less than zero, in order to avoid values changing their sign. These bounds on change are applied after the rounding algorithm is complete. If any fields are found to exceed the upper or lower bounds on change at this point, prorating will stop processing all further data for that particular record, and skip to the next. An appropriate message will be output to advise the user that a bound limit has been exceeded, mentioning the specific variable in violation.

With reference to the previous example, if the user had assigned an Upper Ratio bound = 1.25, after rounding was completed the procedure would have found that variable “a” (prorated from 6 to 9) was more than 125% of its original value. Therefore the entire record REC001 would have been dropped from further analysis, along with a message detailing how prorating was unsuccessful for that particular record due to the violation of a Ratio bound.

10. PROC MASSIMPUTATION – MASS IMPUTATION

Purpose

For operational reasons, in some surveys, detailed information is collected only for a subsample (or second phase sample) of units selected randomly from a large first phase sample. Classical estimates based on the subsample require the derivation of subsampling weights. The derivation of such weights can be quite complex. An alternative technique is known as 'mass imputation' where a complete rectangular file is created for the entire first phase sample units by donor imputing the missing information for the nonsampled units, after the editing and imputation for the second phase sample units has been completed. A procedure has been developed in Banff to facilitate such an operation, using the nearest-neighbour approach. In a typical edit and imputation scenario, the objective is to determine whether a record contains incorrect, missing, inconsistent or outlying responses; the pattern of failure is assumed to be different for each record. In the case of mass imputation, however, the records which require imputation are known and the fields to be imputed are both known and identical for all records. Again, it is assumed that the set of core information collected from the entire sample and the extra items collected from the sub-sample have already been edited and imputed (possibly outside of Banff) and that no consistency edits need to be applied, either to the individual sections or between the two sections.

Matching Fields

Since there is basically no data to which to apply edits for the recipients, the concept of using edits within edit groups for system-identification of fields in error in Banff is not applicable. It follows that the determination of matching fields by the system cannot be performed. However, the user may specify matching fields to be used in finding donors. The matching fields are specified as parameters of the program.

The data for these user-specified matching fields must be in the input dataset. If it occurs that a matching field value is missing for a recipient, that matching field cannot be used for that record since there is no valid value to be used in the distance calculation. In the case of no valid matching fields being available, the user can have the system randomly select a donor for a recipient.

If the user has specified matching fields, the system will transform the matching field values for those donor and recipient records where there are valid values available. This is the same as in the Donor Imputation procedure; refer to section 7.3.

In the case where the user has not specified any matching fields, for each recipient the system will automatically select a random donor.

Other Parameters

As for the Donor Imputation procedure, the user may specify the minimum percentage and number of donor records that must be available for imputation to proceed. These parameters have default values if not specified by the user. The user may also limit the number of times a donor is used, which is unlimited by default.

Description of the Method
The actual execution of the program is very similar to that for the Donor Imputation procedure; refer to section 7.4. There is one significant difference. This is that post-imputation edits are not required. In fact, the procedure will simply impute into the recipient record whatever data is in the input data table from the closest donor record, or from the first randomly selected donor record in the case of no valid matching fields being available. Thus it is very important that the user provide "clean" data for the donor records.

No status file is created by Proc Massimputation. However, a massimputation status may be required by the user in order to calculate variance due to imputation. There is an option for the Banff Processor users to add the status IMAS to the global status file for values imputed by Proc Massimputation. For users of Banff in SAS, the status IMAS must be added manually. Contact Banff support if you need the SAS code to generate the IMAS status from the Proc Massimputation run.

11. REFERENCES

Banff Support Team (2006). Specifying Edits for Processing Negative Values with Banff. Statistics Canada Technical Report.

Banff Support Team (2017). Banff 2.07 Procedures User Guide. Statistics Canada Technical Report.

Banff Support Team (2017). Banff 2.07 Processor User Guide. Statistics Canada Technical Report.

Banff Support Team (2014). Banff 2.06 Tutorial. Statistics Canada Technical Report.

Banff Support Team (2016). Banff FAQ. Intranet page: \\fld6filer\Team0167\Public\Generalized Systems\Banff\FAQ\FAQ_en_fr.pdf.

Chernikova, N.V. (1964). Algorithm for finding a general formula for the nonnegative solutions of a system of linear equations. **U.S.S.R. Computational Mathematics and Mathematical Physics 4**, 151-158.

Chernikova, N.V. (1965). Algorithm for finding a general formula for the nonnegative solution of a system of linear inequalities. **U.S.S.R. Computational Mathematics and Mathematical Physics 5**, 228-233.

Fellegi, I.P., and Holt D. (1976). A systematic approach to automatic edit and imputation. **Journal of the American Statistical Association 71**, 17-35.

Friedman, J.H., Bentley, J.L. and Finkel, R.A. (1977). An algorithm for finding best matches in logarithmic expected time. **ACM Transaction on Mathematical Software 3**, 209-226.

GEIS Support Team (1991). Functional Description of the Generalized Edit and Imputation System. (Revised October 2000). Statistics Canada Technical Report.

Giles, P. (1989). Analysis of edits in a generalized edit and imputation system. Statistics Canada, Methodology Branch Working Paper No. SSMD-89-004E.

Hidiroglou, M.A. and Berthelot, J.-M. (1986). Statistical editing and imputation for periodic business surveys. **Survey Methodology 12**, 73-83.

Kovar, J.G., MacMillan, J. and Whitridge, P. (1988). Overview and strategy for the Generalized Edit and Imputation System. (Updated February 1991). Statistics Canada, Methodology Branch Working Paper No. BSMD-88-007E/F.

Morabito, J. and Shields, M. (1992). Generalized Edit and Imputation System Applications User's Guide. Statistics Canada Technical Report.

Rousseeuw, Peter J. and M. Leroy, Annick (2003). Robust Regression and Outlier Detection. Wiley, New Jersey.

Rubin, D.S. (1973). Vertex generation in cardinality constrained linear programs. **Operations Research 23**, 555-565.

Sande, G. (1978). An algorithm for the fields to impute problems of numerical and coded data. Statistics Canada Technical Report.

Sande, G. (1979). Numerical Edit and Imputation. Presented at the 42nd International Statistical Institute Meeting, Manila, Philippines.

Schiopu-Kratina, I. and Kovar, J.G. (1989). Use of Chernikova's algorithm in the Generalized Edit and Imputation System. Statistics Canada, Methodology Branch Working Paper No. BSMD-89-001E.

Appendix A – Calculation of Medians and Quartiles

Medians

The **median** divides a set of observations into two groups, each containing 50% of the observations. Half of the values are less than the median and half are greater.

If the number of observations is odd, the median is the middle observation. With n observations, the median is in the $\frac{(n+1)}{2}$ position of the ordered observations.

If the number of observations is even, the median is the average of the two central observations. With n observations, the median is the average of the observations in the $\frac{n}{2}$ and the $\frac{n}{2}+1$ positions.

Examples of Medians

	Sorted Observations						n	Median	
set 1:	1	2	<u>6</u>	7	9		5	6	
set 2:	42	<u>59</u>	59				3	59	
set 3:	3	<u>6</u>	<u>9</u>	10			4	7.5	= (6+9)/2
set 4:	2	10	<u>10</u>	<u>10</u>	12	500	6	10	=(10+10)/2

Quartiles

The **first quartile** divides the observations into two groups so that 25% of the observations are less than the value of the first quartile and 75% are greater. To find the first quartile, calculate .25 * (n+1) where n is the number of observations. If this is an integer, the first quartile is equal to the observation in that rank. If this is not an integer but is a value in the form w.d, then the first quartile is between the w th and the (w+1)th observations. The exact value is found by taking (1-.d) times the w th observation plus .d times the (w+1)th observation.

The **third quartile** divides the observations into two groups so that 75% of the observations are less than the value of the third quartile and 25% are greater. To find the third quartile, calculate .75 * (n+1) where n is the number of observations. If this is an integer, the third quartile is equal to the observation in that rank. If this is not an integer but is a value such as w.d, then the third quartile is between the w th and the (w+1)th observations. The exact value is found by taking (1-.d) times the w th observation plus .d times the (w+1)th observation.

Examples of Quartiles
Suppose there are 8 sorted observations: 1, 3, 6, 7, 10, 11, 12, 18. To calculate the first quartile, find the value of .25 * (n+1), which is .25 * 9 = 2.25 for this example. This indicates that the first quartile is one quarter of the way between the second and third ordered observations. Banff calculates the first quartile as .75 of the second ordered observation plus .25 of the third ordered observation. In this case, the first quartile is (.75*3) + (.25*6) = 3.75. It makes sense that the first quartile is nearer the second observation of 3 than the third observation of 6 because it was calculated to be only one quarter of the way between the two observations.

Similarly, to find the third quartile, calculate .75 * (n+1), which is .75 * 9 = 6.75 in this example. Therefore, the third quartile is taken as three quarters of the way between the sixth and the seventh ordered observations. In this example, the third quartile would be .25 times the sixth observation plus .75 times the seventh, or (.25*11) + (.75*12) = 11.75. Again, it makes sense that the third quartile is closer to the seventh observation of 12 than to the sixth observation of 11 because it was calculated that the third quartile was three quarters of the way between the sixth and seventh observations.

The median of this group of observations is 8.5, the average of the fourth and fifth observations.

Appendix B – Pre-Defined Algorithms in Banff

Algorithm:	AUXTREND
Equation:	$\hat{y}_{iC} = \frac{x_{iC}}{x_{iH}} y_{iH}$
Type:	EF
Status:	AT
Format:	aux1(c,v) * fieldid(h,v) / aux1(h,v)
Description:	The value from the previous survey for the same unit, with a trend adjustment calculated from an auxiliary variable, is imputed

Algorithm:	AUXTREND2
Equation:	$\hat{y}_{iC} = \frac{y_{iH}}{2}\left(\frac{u_{iC}}{u_{iH}} + \frac{v_{iC}}{v_{iH}}\right)$
Type:	EF
Status:	AT2
Format:	fieldid(h,v) / 2 * (aux1(c,v)/aux1(h,v) + aux2(c,v)/aux2(h,v))
Description:	An average of two AUXTRENDs is imputed

Algorithm:	CURAUX
Equation:	$\hat{y}_{iC} = x_{iC}$
Type:	EF
Status:	CA
Format:	aux1(c,v)
Description:	The current value of a proxy variable for the same unit is imputed

Algorithm:	CURAUXMEAN
Equation:	$\hat{y}_{iC} = \bar{x}_{C}$
Type:	EF
Status:	CAM
Format:	aux1(c,a)
Description:	The current average of a proxy variable is imputed

Algorithm:	CURMEAN
Equation:	$\hat{y}_{iC} = \bar{y}_{C}$
Type:	EF
Status:	CM
Format:	fieldid(c,a)
Description:	The mean value of all (user-defined) respondents for the current survey is imputed

Algorithm:	CURRATIO
Equation:	$\hat{y}_{iC} = \frac{\bar{y}_C}{\bar{x}_C} x_{iC}$
Type:	EF
Status:	CR
Format:	fieldid(c,a) * aux1(c,v) / aux1(c,a)
Description:	A ratio estimate, using values of all (user-defined) respondents from the current survey is imputed

Algorithm:	CURRATIO2
Equation:	$\hat{y}_{iC} = \frac{\bar{y}_C}{2}\left(\frac{u_{iC}}{\bar{u}_C} + \frac{v_{iC}}{\bar{v}_C}\right)$
Type:	EF
Status:	CR2
Format:	fieldid(c,a)/2 * (aux1(c,v)/aux1(c,a) + aux2(c,v)/aux2(c,a))
Description:	An average of two CURRATIOs is imputed

Algorithm:	CURREG
Equation:	$\hat{y}_{iC} = \hat{\beta}_0 + \hat{\beta}_1 x_{iC}$
Type:	LR
Status:	LR1
Format:	intercept, aux1(c)
Description:	A simple linear regression based on one independent variable from the current data table

Algorithm:	CURREG_E2
Equation:	$\hat{y}_{iC} = \hat{\beta}_0 + \hat{\beta}_1 x_{iC} + \hat{\beta}_2 x_{iC}^2$
Type:	LR
Status:	LRE
Format:	intercept, aux1(c), aux1(c)ϖ2
Description:	A regression based on the value and the squared value of a variable from the current data table

Algorithm:	CURREG2
Equation:	$\hat{y}_{iC} = \hat{\beta}_0 + \hat{\beta}_1 u_{iC} + \hat{\beta}_2 v_{iC}$
Type:	LR
Status:	LR2
Format:	intercept, aux1(c), aux2(c)
Description:	A linear regression based on two independent variables from the current data table

Algorithm:	CURREG3
Equation:	$\hat{y}_{iC} = \hat{\beta}_0 + \hat{\beta}_1 u_{iC} + \hat{\beta}_2 v_{iC} + \hat{\beta}_3 w_{iC}$
Type:	LR
Status:	LR3
Format:	intercept, aux1(c), aux2(c), aux3(c)
Description:	A linear regression based on three independent variables from the current data table

Algorithm:	CURSUM2
Equation:	$\hat{y}_{iC} = u_{iC} + v_{iC}$
Type:	EF
Status:	SM2
Format:	aux1 + aux2
Description:	The sum of two auxiliary variables from the current data table

Algorithm:	CURSUM3
Equation:	$\hat{y}_{iC} = u_{iC} + v_{iC} + w_{iC}$
Type:	EF
Status:	SM3
Format:	aux1 + aux2 + aux3
Description:	The sum of three auxiliary variables from the current data table

Algorithm:	CURSUM4
Equation:	$\hat{y}_{iC} = u_{iC} + v_{iC} + w_{iC} + z_{iC}$
Type:	EF
Status:	SM4
Format:	aux1 + aux2 + aux3 + aux4
Description:	The sum of four auxiliary variables from the current data table

Algorithm:	DIFTREND
Equation:	$\hat{y}_{iC} = \frac{\bar{y}_C}{\bar{y}_H} y_{iH}$
Type:	EF
Status:	DT
Format:	fieldid(c,a) * fieldid(h,v) / fieldid(h,a)
Description:	The value from the previous survey for the same unit, with a trend adjustment calculated from the difference of reported values for the variable, is imputed

Algorithm:	HISTREG
Equation:	$\hat{y}_{iC} = \hat{\beta}_0 + \hat{\beta}_1 y_{iH}$
Type:	LR
Status:	HLR
Format:	intercept, fieldid(h)
Description:	A linear regression based on the historical value of the variable to impute

Algorithm:	PREAUX
Equation:	$\hat{y}_{iC} = x_{iH}$
Type:	EF
Status:	PA
Format:	aux1(h,v)
Description:	The historical value of a proxy variable for the same unit

Algorithm:	PREAUXMEAN
Equation:	$\hat{y}_{iC} = \bar{x}_{H}$
Type:	EF
Status:	PAM
Format:	aux1(h,a)
Description:	The historical average of a proxy variable for the same unit is imputed

Algorithm:	PREMEAN
Equation:	$\hat{y}_{iC} = \bar{y}_{H}$
Type:	EF
Status:	PM
Format:	fieldid(h,a)
Description:	The mean value from the previous survey of all (user-defined) respondents is imputed

Algorithm:	PREVALUE
Equation:	$\hat{y}_{iC} = y_{iH}$
Type:	EF
Status:	PV
Format:	fieldid(h,v)
Description:	The value from the previous survey for the same unit is imputed