The Banff software version 2.05

This set of documents is intended to guide you in using the Banff SAS procedures. The options and statements for each one of the Banff procedures are explained. The guides also offer one or more examples of SAS code that you can copy and paste into your SAS session editor and run as is.

If you need assistance, please contact the support teams at ***banff***.

General Information

Overview

Banff is a system that offers methods of editing and imputing survey data in the form of SAS procedures. The procedures have been built in-house and behave exactly the same way as any other procedures of the SAS software.

The Banff procedures are available to methodologists responsible for the development of edit and imputation strategies for numerical continuous data. They are easy to use and present a standard interface that is well documented. They offer methods that are known and have been approved at Statistics Canada.

Banff is built from libraries of functions that implement the basic algorithms as well as a number of utilities. The communication with the SAS system is carried out with the use of the SAS/Toolkit functions. With its flexible and modular architecture, new procedures can be added with little effort to the Banff system, therefore making it easy to create new versions.

The software is fully supported by a team of programmers and methodologists who continuously work at improving the software and are ready to answer requests related to problems that users may encounter.

Banff is part of the Statistics Canada Generalized Systems Program and therefore is free of charge to Statistics Canada employees. Other organizations may purchase a copy or receive a free limited-time evaluation copy. Please send an e-mail to *banff* for more details.

Functions

- **PROC VERIFYEDITS:** Check Edits for consistency and redundancy; Generate Extremal Points and Implied Edits
- **PROC EDITSTATS:** Edit Summary Statistics Tables
- **PROC OUTLIER:** Detection of Outlier Values
- **PROC ERRORLOC:** Error Localization (Identification of fields to impute)
- **PROC DETERMINISTIC:** Deterministic Imputation
- **PROC DONORIMPUTATION:** Donor Imputation
- **PROC ESTIMATOR:** Estimator Imputation
- **PROC PRORATE:** Prorated adjustments
- **PROC MASSIMPUTATION:** Mass Imputation

Benefits

- Seamlessly integrated into SAS
- Available on MS Windows™ and UNIX

- Easy to use, easy to learn
- Well documented
- Known and approved methods
- Full methodology and systems development support

Installation

- Foundation software required: SAS 9
- Available on UNIX after a custom installation performed by the technical support team. (the C code must be re-compiled for each new UNIX platform)
- Installs on MS Windows™ by running the Banff setup program.

Banff news subscription service

For Statistics Canada users:
Users can register to receive updates via email regarding changes and improvements to the Banff software. This can be done by clicking on the **Subscribe** button on the News page of the Banff intranet site:

http://gensys/DesktopDefault.aspx?lang=en&tabid=384

Please note that the updates are also available in this section of the intranet site.

Documentation

A user guide is available for each procedure. For more information on the methodology, you can refer to the Banff Functional Description.

Support Team

The software is supported by a technical team in the System Engineering Division and by a methodology team in the Business Survey Methods Division. To contact us, please send an e-mail to *banff*.

The VERIFYEDITS Procedure

Overview

This 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 is used in this module; it is the edits themselves which are analyzed.

The procedure also generates all the extremal points, or vertices, of the feasible region described by the 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.

The procedure then 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.

Procedure Syntax

PROC VERIFYEDITS <*option(s)*>*;*

PROC VERIFYEDITS Statement

PROC VERIFYEDITS <*option(s)*>*;*

To do this	Use this option
Specify the edits	EDITS=
Specify the maximum number of implied edits to generate	IMPLY=
Specify the maximum cardinality of the shown extremal points	EXTREMAL=
Use the rules in the EDITS= option as specified	ACCEPTNEGATIVE
Add one positivity edit for each variable involved in the rules of the EDITS= option	REJECTNEGATIVE

Options

EDITS=*quoted string of all edits*

specifies the edit rules as linear equations. They must be enclosed in quotes and end with semi-colons. They are mandatory.

IMPLY=
MAXIMPLIEDEDITS=***positive integer***

number of implied edits to generate. This number is optional. If this option is not specified no implied edits will be generated.

EXTREMAL=
MAXCARDINALITY=***positive integer***

show all extremal points with cardinality less than or equal to this number. Here, **cardinality** refers to the number of non-zero coordinates of an extremal point. Optional. If not provided or zero, no extremal points will be generated. If the option ACCEPTNEGATIVE is specified, no extremal points will be generated. Indeed, the use of the algorithm (Chernikova) to calculate the extremal points does not permit that the variables take on negative values.

ACCEPTNEGATIVE

if this option is specified, the rules specified in the EDITS= option will be used as is.

Warning: Before using this option, please read the document Edits for Negative Values on specifying edits when processing negative numbers in Banff.

REJECTNEGATIVE

if this option is specified, for each variable involved in the rules of the EDITS= option, a positivity edit will be added.

NOTE: If neither ACCEPTNEGATIVE nor REJECTNEGATIVE is specified, the default value is REJECTNEGATIVE.

Details

- The edits will always be checked for consistency.
- The ACCEPTNEGATIVE and REJECTNEGATIVE options are mutually exclusive. Specifying both will cause the procedure to stop.

Example 1

Addition of one positivity edit for each variable involved in the rules of the EDITS= option

```
/* Implicit addition of positivity edits: REJECTNEGATIVE */
proc VERIFYEDITS
edits = "
HEN_LT20+HEN_GE20+HEN_OTH=HEN_TOT;
2*EGG_LAID<=HEN_GE20;
HEN_GE20<=4*EGG_LAID;
EGG_SOLD<=EGG_LAID;
EGG_VALU<=2.75*EGG_SOLD;
0.9*EGG_SOLD<=EGG_VALU;
fail:HEN_GE20>300000;"
imply = 50
extremal = 10
;
```

```
run;
```

Example 2

No positivity rules added

```
/* With rules specified, value of variables NETINC1, NETINC2
and NETINCYEAR could be negative. */
proc VERIFYEDITS
edits = "
NETINC1 + NETINC2 = NETINCYEAR;
INC1 - EXP1 = NETINC1;
INC2 - EXP2 = NETINC2;
INC1 >= 0; INC2 >= 0;
EXP1 >= 0; EXP2 >= 0;"
imply = 50
extremal = 10
;
run;
```

Notes

This document is a guide for the use of the procedure PROC VERIFYEDITS. For more information on the methodology, please see the *Banff Functional Description* document.

The EDITSTATS Procedure

Overview

This procedure applies a group of edits to a SAS dataset and determines if each observation passes, misses or fails each edit. Five tables summarizing the status codes are produced and may be used to fine-tune the group of edits or to evaluate the effects of imputation. To facilitate the manipulation of that information, five output data sets are built with the corresponding information and an additional output data set is also produced with the minimal set of edits built.

Procedure Syntax

PROC EDITSTATS <*option(s)*>*;*

BY *variable(s);*

To do this	Use this statement
Produce the tables for each BY group	BY

PROC EDITSTATS Statement

PROC EDITSTATS <*option(s)*>*;*

To do this	Use this option
Specify the input data set	DATA=
Specify the edits	EDITS=
Specify the output data set that contains the minimal set of edits	OUTREDUCEDEDITS=
Specify the output data set that contains the report corresponding to Table 1.1 from the methodological guide	OUTEDITSTATUS=
Specify the output data set that contains the report corresponding to Table 1.2 from the methodological guide	OUTKEDITSSTATUS=
Specify the output data set that contains the report corresponding to Table 1.3 from the methodological guide	OUTGLOBALSTATUS=
Specify the output data set that contains the report corresponding to Table 2.1 from the methodological guide	OUTEDITAPPLIC=
Specify the output data set that contains the report corresponding to Table 2.2 from the methodological guide	OUTVARSROLE=
Specify that negative values are valid values	ACCEPTNEGATIVE
Specify that negative values are invalid values	REJECTNEGATIVE

Options

DATA=*SAS-data-set*

specifies the input SAS data set. If BY variables are specified, the observations of this data set must be sorted by the values of those variables. If DATA= is omitted, the most recently created SAS data set is used.

EDITS=*quoted string of all edits*

specifies the edit rules as linear equations. They must be enclosed in quotes and end with semi-colons. They are mandatory.

OUTREDUCEDEDITS =*SAS-data-set*

specifies the output data set that contains the minimal set of edits. This SAS data set is optional. If it is not provided, a default output data set will be generated. Specify _NULL_ as the name for this data set if no data set is to be generated.

The variables in this SAS data set are:

EDITID	**Identification number of the edit belonging to the minimal set (useful as reference in OUTEDITSTATUS=)**
EDIT_EQUATION	The formulation of the edit

OUTEDITSTATUS =*SAS-data-set*

specifies the output data set that contains the report corresponding to Table 1.1 in the methodological guide (see Notes below). This SAS data set is optional. If it is not provided, a default output data set will be generated. Specify _NULL_ as the name for this data set if no data set is to be generated.

The variables in this SAS data set are:

EDITID	**Identification number of the edit belonging to the minimal set**
OBS_PASSED	The number of observations having passed the edit
OBS_MISSED	The number of observations having one or more missing variables involved in the edit
OBS_FAILED	The number of observations having failed the edit because of one or more non-missing values

OUTKEDITSSTATUS =*SAS-data-set*

specifies the output data set that contains the report corresponding to Table 1.2 in the methodological guide (see Notes below). This SAS data set is optional. If it is not provided, a default output data set will be generated. Specify _NULL_ as the name for this data set if no data set is to be generated.

The variables in this SAS data set are:

K_EDITS	**« k » cumulated edits belonging to the minimal set**
OBS_PASSED	The number of observations having passed « k » edits

OBS_MISSED	The number of observations having one or more missing variables involved in « k » edits
OBS_FAILED	The number of observations having failed « k » edits because of one or more non-missing values

OUTGLOBALSTATUS =***SAS-data-set***

specifies the output data set that contains the report corresponding to Table 1.3 in the methodological guide (see Notes below). This SAS data set is optional. If it is not provided, a default output data set will be generated. Specify _NULL_ as the name for this data set if no data set must be generated.

The variables in this SAS data set are:

OBS_PASSED	**The number of observations with PASS as overall status (i.e. having passed all the edits belonging to the minimal set)**
OBS_MISSED	The number of observations with MISS as overall status (i.e. having one or more missing variables involved in one or more edits belonging to the minimal set but without FAIL edit status for any rule)
OBS_ FAILED	The number of observations with FAIL as overall status (i.e. having at least one FAIL edit status for one edit belonging to the minimal set)
OBS_TOTAL	Total number of observations

OUTEDITAPPLIC =***SAS-data-set***

specifies the output data set that contains the report corresponding to Table 2.1 in the methodological guide (see Notes below). This SAS data set is optional. If it is not provided, a default output data set will be generated. Specify _NULL_ as the name for this data set if no data set is to be generated.

The variables in this SAS data set are:

FIELDID	**The name of the variable**
EDIT_APPLIC_PASSED	The number of times the variable has « inherited » a PASS status code given to observations for the edits involving the variable
EDIT_APPLIC_MISSED	The number of times the variable has « inherited » a MISS status code given to observations for the edits involving the variable
EDIT_APPLIC_FAILED	The number of times the variable has « inherited » a FAIL status code given to observations for the edits involving the variable
EDIT_APPLIC_NOTINVOLVED	(Number of edits not involving the variable) X (Number of observations)
EDITS_INVOLVED	The number of edits involving the variable

OUTVARSROLE =***SAS-data-set***

specifies the output data set that contains the report corresponding to Table 2.2 in the methodological guide (see Notes below). This SAS data set is optional. If it is not provided, a default output data set will be generated. Specify _NULL_ as the name for this data set if no data set is to be generated.

The variables in this SAS data set are:

FIELDID	The name of the variable
OBS_PASSED	The number of times a variable has an incidence on the PASS overall status code given to observations
OBS_MISSED	The number of times a variable has an incidence on the MISS overall status code given to observations
OBS_FAILED	The number of times a variable has an incidence on the FAIL overall status code given to observations
OBS_NOT_APPLICABLE	The number of times a variable has no incidence on the MISS or FAIL overall status code given to observations

ACCEPTNEGATIVE

if this option is specified, negative values will be considered valid values.

Warning: Before using this option, please read the document Edits for Negative Values on specifying edits when processing negative numbers in Banff.

REJECTNEGATIVE

if this option is specified, negative values will not be considered valid values. For each variable involved in the rules of the EDITS= option, a positivity edit will be added.

NOTE: If neither ACCEPTNEGATIVE nor REJECTNEGATIVE is specified, the default value is REJECTNEGATIVE.

BY Statement

BY *variable*-1 ... *variable-n*;

Required arguments

variable(s)

specifies the variable(s) that the procedure uses to form BY groups. The tables will be generated for each group independently. You can specify more than one variable. This statement is optional.

Details

- **The sorting of variables must be done in ascending order of the values.**
- **If BY variables are specified, the observations of the DATA= input data set must be sorted by the values of those variables.**
- If BY variables are specified, the BY variables will appear in the 5 output data sets that correspond to the 5 tables from the *Banff Functional Description*. The BY variables will not appear in the output data set OUTREDUCEDEDITS=.
- The ACCEPTNEGATIVE and REJECTNEGATIVE options are mutually exclusive. Specifying both will cause the procedure to stop.

Example

```
data data;
input x1 x2 x3 Group $1.;
cards;
4 3 2 A
-4 3 2 A
6 3 2 B
4 3 . A
6 3 . B
;
run;
proc sort data=data; by Group; run;
proc editstats
data=data
outreducededits=min_edits
outeditstatus=table11
outkeditsstatus=table12
outglobalstatus=table13
outeditapplic=table21
outvarsrole=table22
edits="
x1+1>=x2;
x1<=5;
x2>=x3;
x1+x2+x3<=9;"
acceptnegative
;
by Group;
run;
```

Notes

This document is a guide for the use of the procedure PROC EDITSTATS. For more information on the methodology, please see the *Banff Functional Description* document.

The OUTLIER Procedure

Overview

This procedure identifies outlying observations. Values of selected variables are compared across records rather than comparing fields within each individual record, as is done with the linear edit rules.

Procedure Syntax

PROC OUTLIER <*option(s)*>;

ID *variable;*

VAR *variable(s);*

WITH *variable(s);*

BY *variable(s);*

To do this	Use this statement
Identify the key variable of the input data set(s)	ID
Identify variables for which to find outliers	VAR
Identify (if needed by the method) historical or auxiliary variables which will be paired with VAR variables to find outliers	WITH
Obtain separate analysis on observations in BY groups	BY

PROC OUTLIER Statement

PROC OUTLIER <*option(s)*>;

To do this	Use this option
Specify the input SAS data set that holds variables for which outliers are found and may also hold the auxiliary variables depending on the method used	DATA=
Specify (if needed) the input SAS data set that holds the historical or auxiliary variables	HIST|AUX=
Name the output SAS data set that contains the status of the fields (FTE/FTI) and also the detailed status for the outliers (ODEL/ODER/ODIL/ODIR)	OUTSTATUS=
Specify the method to use to detect outlying observations	METHOD=
Specify the multiplier for imputation interval	MII=
Specify the multiplier for exclusion interval	MEI=
Specify the minimum distance multiplier	MDM=
Specify the exponent for historical or ratio method	EXPONENT=
Specify the minimum number of observations that have to exist in the DATA= input data set or in a BY group if BY variables are	MINOBS=

used	
Specify to add more information about the bounds of the intervals in the output SAS data set OUTSTATUS=	BOUNDSTAT
Specify that negative values are valid values	ACCEPTNEGATIVE
Specify that negative values are invalid values	REJECTNEGATIVE

Options

DATA=*SAS-data-set*

specifies the input SAS data set that holds the variables for which outliers are found. If DATA= is omitted, the most recently created SAS data set is used.

This data set will also hold the WITH variables when the historical or ratio method is specified (METHOD=H or METHOD=R) and no data set is specified with the option HIST|AUX=. (See METHOD= for more information).

If BY variables are specified, the observations of this data set must be sorted by the values of those variables.

HIST=*SAS-data-set*

specifies the input SAS data set that holds the historical or auxiliary variables which will be paired with the VAR variables if METHOD=H or R. Keywords HIST and AUX are interchangeable and refer to the same methodology (see option METHOD= for more information).

Only observations matching on "*key variable*" values with the input data set DATA= are kept.

If BY variables are specified, the observations of this data set must be sorted by the values of those variables.

OUTSTATUS=*SAS-data-set*

names the output SAS data set that contains a status of FTE or FTI for variables that were identified as outliers to exclude or to impute. This output data set also contains the field OUTSTATUS giving the detailed status of the outliers (ODEL, ODER, ODIL or ODIR). If a BY statement is specified, then this data set also contains the BY variables. If the option BOUNDSTAT is in effect, additional variables will be written in this output SAS data set (see option BOUNDSTAT). If you want the OUTSTATUS= data set to be permanent, specify a two-level name.

METHOD=*keyword*

specifies the method to use to detect outlier variables. Mandatory. For the current method: **CURRENT**, **CU** or **C**. For the historical or ratio method: **HISTORIC**, **HT**, **H**, **RATIO** or **R**.

NOTE: The methodology behind HISTORIC and RATIO is the same. Keywords HISTORIC and RATIO are interchangeable. This methodology can be used in the following situations:

- identify outlier values for a numeric variable in one input data set paired with the **same** numeric variable in **another** input data set.

- identify outlier values for a numeric variable in one input data set paired with **another** numeric variable in the **same** input data set.
- identify outlier values for a numeric variable in one input data set paired with **another** numeric variable in **another** input data set.

The value specified with the option METHOD= will be used to decide what combinations of METHOD=, HIST|AUX=, VAR and WITH values are valid. The following table shows the valid combinations.

	DATA=*SAS-data-set* (only one input data set)	**DATA=*SAS-data-set* HIST | AUX=*SAS-data-set* (two input data sets)**
Only VAR statement	**Case 1 :** **Current method and identifying outlier values for:** **-X_{data1}** **-Y_{data1}** PROC OUTLIER **DATA=data1** OUTSTATUS=status **METHOD=current** …. ; **VAR X Y**; ID IDNUM; RUN;	**Case 2 :** **Historical or Ratio method and identifying outlier values for:** **-X_{data1}** (trend analysis between X_{data1} and X_{data2}, i.e. the same variable in two input data sets) PROC OUTLIER **DATA=data1** **HIST=data2 /* or AUX=data2 */** OUTSTATUS=status **METHOD=historic /* or ratio */** …. ; **VAR X**; ID IDNUM; RUN;
VAR and WITH statements	**Case 3 :** **Historical or Ratio method and identifying outlier values for:** **-X_{data1}** (trend analysis between X_{data1} and Y_{data1}, i.e. between two variables in the same input data set) PROC OUTLIER **DATA=data1** OUTSTATUS=status **METHOD=ratio /* or historic */** …. ;	**Case 4 :** **Historical or Ratio method and identifying outlier values for:** **-X_{data1}** (trend analysis between X_{data1} and Y_{data2}, i.e. between two variables in two input data sets) **-Z_{data1}** (trend analysis between Z_{data1} and Z_{data2}, i.e. the same variable in two input data sets) PROC OUTLIER **DATA=data1** **AUX=data2 /* or HIST=data2 */** OUTSTATUS=status

	VAR X; **WITH Y**; ID IDNUM; RUN;	**METHOD=ratio /* or historic */** …. ; **VAR X Z**; **WITH Y Z**; ID IDNUM; RUN;

MII= ***positive real number***

specifies the multiplier for imputation interval. Optional. It must be greater than zero. MII must be greater than MEI, if MEI is specified.

MEI= ***positive real number***

specifies the multiplier for exclusion interval. Optional. It must be greater than zero. MEI must be lower than MII, if MII is specified.

MDM= ***real number***

specifies the minimum distance multiplier. Optional. It must be greater than or equal to zero. Default value is 0.05.

NOTE : The minimum distance multiplier specified will not be used when the value for the median is 0 for the current method or the value of the median of effects is 0 for the historical or ratio method (METHOD=H or METHOD=R). See the Banff Functional Description for more information on the effect definition and calculation.

EXPONENT=*real number between 0 and 1*

specifies the exponent for historical or ratio method (METHOD=H or METHOD=R). Optional. It must be between 0 and 1 inclusively. Default value is 0.

MINOBS=*positive integer greater than 3*

specifies the minimum number of observations that must exist on the DATA= data set or in the BY group being processed. Optional. This option applies to all methods. If not specified, the default value is 10.

BOUNDSTAT

specifies that the user wants more information about the bounds of the imputation and exclusion intervals in the output SAS data set OUTSTATUS=. This information is the same as the one written in the SAS log. When either the MII option or the MEI option is not specified, the values of the corresponding bounds are set to missing. Depending on the method specified with the option METHOD=, additional variables may appear on the data set. Optional.

For the current data method (**METHOD = C**), variables added to the output data set are:

CURRENT_VALUE	current value of the outlier variable (value read from the input data set

	DATA=)
IMP_BND_L	value of the upper bound of left imputation interval if option MII= has been specified (same as lower bound of left exclusion interval if MEI has been specified)
EXCL_BND_L	value of upper bound of left exclusion interval if option MEI= has been specified (same as lower bound of acceptance interval)
EXCL_BND_R	value of lower bound of right exclusion interval if option MEI= has been specified (same as upper bound of acceptance interval)
IMP_BND_R	value of lower bound of right imputation interval if option MII= has been specified (same as upper bound of right exclusion interval if MEI has been specified)

For the historical or ratio method (**METHOD = H or METHOD= R**), variables added to the output data set are:

CURRENT_VALUE	current value of the outlier variable (will be read from the input data set DATA=)
HIST_AUX	If WITH specified: name of the variable paired with the corresponding VAR variable If WITH not specified, this variable is not added in the output data set
HIST_AUX_VALUE	If WITH specified: value of the variable paired with the corresponding VAR variable (will be read from the HIST|AUX= data set if specified, from the DATA= data set otherwise) If WITH not specified: value of the variable having the same name as the VAR variable (will be read from the HIST|AUX= data set)
EFFECT	value used to compare with the interval boundaries
IMP_BND_L	value of the upper bound of left imputation interval if option MII= has been specified (same as lower bound of left exclusion interval if MEI has been specified)
EXCL_BND_L	value of upper bound of left exclusion interval if option MEI= has been specified (same as lower bound of acceptance interval)
EXCL_BND_R	value of lower bound of right exclusion interval if option MEI= has been specified (same as upper bound of acceptance interval)
IMP_BND_R	value of lower bound of right imputation interval if option MII= has been specified (same as upper bound of right exclusion interval if MEI has been specified)

ACCEPTNEGATIVE

if this option is specified, negative values will be considered valid values and will be used in calculations.

NOTE: This option is available for METHOD=CURRENT, CU or C only.

REJECTNEGATIVE

if this option is specified, negative values will not be used in calculations.

NOTE: If neither ACCEPTNEGATIVE nor REJECTNEGATIVE is specified, the default value is REJECTNEGATIVE.

ID Statement

ID *variable*;

Required argument

variable

specifies the variable that the procedure uses as the key variable of the input data set(s). It is mandatory. It must be a character variable and only one is allowed (i.e. no composite key).

VAR Statement

VAR *variable*-1 ... *variable*-n;

Required arguments

variables

The VAR statement lists variables for which to find outliers. The variables must be numeric and must exist in the input data set DATA=.

For the current data method, the VAR statement is optional. If not specified, **all numeric variables** of the DATA= data set other than the ones specified with the BY statement will be processed.

For the historical or ratio method (METHOD=H or METHOD=R), the VAR statement is mandatory. If the WITH statement is not specified when METHOD=H or R, the option HIST|AUX= is mandatory and the variables specified with the VAR statement must exist on the input data set HIST|AUX= (see option METHOD= and statement WITH).

WITH Statement

WITH *variable*-1 ... *variable*-n;

Required arguments

variables

The WITH statement lists variables to be paired one by one with the VAR variables and concerns only the historical or ratio method (METHOD=H or METHOD=R). Optional (see option METHOD= for more information).

If a WITH statement is specified, the number of variables in the list must be the same as the one in the VAR list and the variables must be numeric in the input data set from which they are read. If the option HIST|AUX= is specified, the WITH variables are read from the data set specified, otherwise they are read from the DATA= data set.

BY Statement

BY *variable*-1 ... *variable*-n;

Required arguments

variable(s)

specifies the variable(s) that the procedure uses to form BY groups. If the outlier method specified is the historical or ratio method (METHOD=H or METHOD=R) and option HIST|AUX= is specified, the BY variable(s) must exist in both the DATA= and the HIST|AUX= data sets. A BY statement can be used to obtain separate analysis on observations in groups defined by the BY variables. You can specify more than one variable. This statement is optional.

Details

- The sorting of variables must be done in ascending order of the values.
- **If BY variables are specified, the observations of the DATA= and HIST (or AUX)= input data sets must be sorted by the values of those variables.**
- If a BY statement is used, the BY variables will appear in the OUTSTATUS data set.
- At least one of MII and MEI must be specified.
- A variable cannot be specified in more than one ID, BY, VAR and WITH statements. These lists of variables are mutually exclusive.
- Missing values will not be processed.
- If the REJECTNEGATIVE option is in effect, the observations with negative values for variables involved in calculations will not be processed. A warning message will be entered in the log file with a counter for the number of observations dropped.
- The ACCEPTNEGATIVE and REJECTNEGATIVE options are mutually exclusive. Specifying both will cause the procedure to stop.
- If the value of MINOBS is less than 3, the procedure will stop with an error message.
- If the value of MINOBS is greater or equal to 3 and less than 10, a warning will be written to the log.
- If the number of valid observations is less than MINOBS, the procedure will not run.

Example 1

Detecting outliers using the current trend method with options boundstat and acceptnegative

```
options linesize=80 pagesize=32000;
data outlierdata (drop=n);
n = 1;
do while (n <= 30);
IDENT = 'R' || put (n, Z7.);
X01 = round (ranuni (1) * 20);
X02 = round (ranuni (1) * 20);
if mod(n,2)=0 then Prov=10; else Prov=11;
if mod(n,5)=0 then do; X01=-X01; X02=-X02; end;
output;
n = n + 1;
end;
run;
/* sort the data in order of the BY variable */
proc sort; by prov; run;
proc outlier
data=outlierdata
outstatus=outlierstatus
method=current
mii=1.5
mei=1.3
mdm=.05
boundstat
acceptnegative
;
id ident;
var X01 X02;
by prov;
run;
```

Example 2

Detecting outliers using the historical trend method (rejectnegative is the default value)

```
options linesize=80 pagesize=32000;
data outlierdata (drop=n);
n = 1;
do while (n <= 30);
IDENT = 'R' || put (n, Z7.);
X01 = round (ranuni (1) * 20);
X02 = round (ranuni (1) * 20);
if mod(n,2)=0 then Prov=10; else Prov=11;
if mod(n,5)=0 then do; X01=-X01; X02=-X02; end;
output;
```

```
n = n + 1;
end;
run;
/* sort the data in order of the BY variable */
proc sort; by prov; run;
data outlierhist (drop=n);
n = 1;
do while (n <= 30);
IDENT = 'R' || put (n, Z7.);
X01 = round (ranuni (1) * 30);
X02 = round (ranuni (1) * 30);
if mod(n,2)=0 then Prov=10; else Prov=11;
if mod(n,7)=0 then do; X01=-X01; X02=-X02; end;
output;
n = n + 1;
end;
run;
/* sort the data in order of the BY variable */
proc sort; by prov; run;
proc outlier
data=outlierdata
hist=outlierhist
outstatus=outlierstatus
method=historic
mii=1.5
mei=1.3
mdm=.05
;
id ident;
var X01 X02;
by prov;
run;
```

Notes

This document is a guide for the use of the procedure PROC OUTLIER. For more information on the methodology, please see the *Banff Functional Description* document.

The ERRORLOC Procedure

Overview

This procedure identifies the minimum number of variables which must be changed in each observation so that the observation can be made to pass all the edits. The variables which require imputation are identified, but no imputation takes place.

Procedure Syntax

PROC ERRORLOC *<option(s)>;*

ID *variable;*

BY *variable(s);*

To do this	Use this statement
Identify the key variable of the input data set	ID
Perform error localisation for each BY group	BY

PROC ERRORLOC Statement

PROC ERRORLOC *<option(s)>;*

To do this	Use this option
Specify the input data set	DATA=
Specify the output SAS data set that contains the status of the fields (FTI)	OUTSTATUS=
Specify the output SAS data set that contains the rejected observations	OUTREJECT=
Specify the edits	EDITS=
Specify the weights for the variables	WEIGHTS=
Specify the maximum cardinality	CARDINALITY=
Specify the maximum processing time allowed per observation	TIMEPEROBS=
Specify that negative values are valid values	ACCEPTNEGATIVE
Specify that negative values are invalid values	REJECTNEGATIVE

Options

DATA=*SAS-data-set*

specifies the input SAS data set. **The observations of this data set must be sorted by the values of the key variable specified in the ID statement**. **If BY variables are specified, the observations of this data set must also be sorted by the values of those variables. When sorting, the BY variables**

must be listed before the ID variable. If DATA= is omitted, the most recently created SAS data set is used.

OUTSTATUS=*SAS-data-set*

names the output data set that contains the variables that require imputation. The STATUS variable is set to FTI. The variables in this data set are: "*key variable*", FIELDID and STATUS. FIELDID is the variable whose values are the names of the variables for which a STATUS of FTI is assigned. If a BY statement is specified, then this data set also contains the BY variables. If you want the OUTSTATUS= data set to be permanent, specify a two-level name.

OUTREJECT=*SAS-data-set*

names the output data set that contains the observations for which error localisation could not be performed. The variables created in this data set are: "*key variable*" and NAME_ERROR. If a BY statement is specified, then this data set also contains the BY variables. If you want the OUTREJECT=data set to be permanent, specify a two-level name. The following table describes the two types of errors:

CARDINALITY EXCEEDED*	**The cardinality for this observation's solution exceeds the maximum cardinality specified in the CARDINALITY= option.**
TIME EXCEEDED*	The processing time for this observation exceeds the maximum time allowed per observation specified in the TIMEPEROBS= option.

*Any observations on which error localization could not be performed, because they exceeded the maximum allowable cardinality or time per observation, will be identified on the REJECTS file and will still remain on the original input data set. If the user plans to run additional Banff procedures on the data, the rejected observations should be removed with a user-defined program. Otherwise, they will be taken into account when other Banff procedures generate statistics on the data in the SAS log. The REJECTS can be added back later with another user-defined program if, for example, the user wants to keep a complete file or to impute them using Proc Massimputation.

EDITS=*quoted string of all edits*

specifies the edit rules as linear equations. They must be enclosed in quotes and end with semi-colons. They are mandatory. It is important to note that the EDITS are used to tell the procedure which variables from the DATA= dataset will be processed.

WEIGHTS=*quoted string of weights*

specifies the weights. Optional. If a weight is not specified for a variable, it will be set to 1. Weights are specified by a list of *variable=value* statements separated by semi-colons, where *variable* is a variable specified in the EDITS= option and *value* is a number **greater than 0.** Weights are used to exert some influence on the fields that are selected for imputation. Please refer to the *Banff Functional Description* for **more details on using weights.**

CARDINALITY=*positive number*

specifies the maximum cardinality. Optional. If not provided, the result will not be constrained by cardinality.

TIMEPEROBS=*positive number*

specifies the maximum processing time spent on each observation in seconds in order to find a solution. Optional. If not provided, the time limit will be set to 20 seconds per observation.

SEED=*positive integer*

this parameter is optional and is useful only in development when results need to be compared from one run to the next. The default value is a random number. If a negative value, 0 or a value exceeding the maximum acceptable by the platform is specified, it will be replaced by the default value.

ACCEPTNEGATIVE

if this option is specified, negative values will be considered valid values.

Warning: Before using this option, please read the document Edits for Negative Values on specifying edits when processing negative numbers in Banff.

REJECTNEGATIVE

if this option is specified, variables with negative values will be flagged as fields to impute (FTI). For each variable involved in the edits of the EDITS= option, a positivity edit will be added.

NOTE: If neither ACCEPTNEGATIVE nor REJECTNEGATIVE is specified, the default value is REJECTNEGATIVE.

ID Statement

ID *variable*;

Required argument

variable

specifies the variable that the procedure uses as the key variable of the input data set. It is mandatory. It must be a character variable and only one is allowed (i.e. no composite key).

BY Statement

BY *variable*-1 ... *variable-n*;

Required arguments

variable(s)

specifies the variable(s) that the procedure uses to form BY groups. Error localisation is performed on one observation at a time. So specifying BY variables will not influence the results. However, the BY variables will appear in the two output data sets. You can specify more than one variable. This statement is optional.

Details

- **The sorting of observations must be done in ascending order of the values of the variables.**
- **If BY variables are specified, the observations of the DATA= input data set must be sorted by the values of those variables as well as by the values of the ID variable. When sorting, the BY variables must be listed before the ID variable.**
- If BY variables are specified, the BY variables specified will appear in the OUTSTATUS= and OUTREJECT= data sets.
- The only variables that will be processed are the ones specified in the edits.
- It is not necessary to code any of the positivity edits.
- The coherence of the system of linear equations is verified. If the system is not coherent, the program stops with an error message.
- Observations with *a missing value for the key variable in the input data set DATA=* will not be processed. A warning message will be entered in the log file with a counter for the number of observations dropped.
- Variables with a MISSING value will always be marked as FTI.
- If the option REJECTNEGATIVE is in effect, variables with a negative value will always be marked as FTI.
- The procedure does not read any field statuses at input. So for example, if the OUTLIER procedure was run and it identified some fields as being FTI, the user will have to set the value of those fields to MISSING before running ERRORLOC in order for ERRORLOC to flag those fields as Fields to be Imputed.
- The ACCEPTNEGATIVE and REJECTNEGATIVE options are mutually exclusive. Specifying both will cause the procedure to stop.

Example

Option "rejectnegative" is the default value and the input data set is sorted by the BY variable ZONE listed before the ID variable IDENT (BY ZONE IDENT).

```
options ls=80 ps=80 nodate;
data example;
input IDENT $ X1 X2 ZONE $1.;
cards;
R03 10 40 B
R02 -4 49 A
R04 4 49 A
R01 16 49 A
R05 15 51 B
R07 -4 29 B
R06 30 70 B
;
run;
/* sort the data in order of the BY variable */
proc sort data=example; by ZONE IDENT;run;
proc errorloc
data=example
outstatus=outstatus
outreject=outreject
edits="x1>=-5; x1<=15; x2>=30; x1+x2<=50;"
weights="x1=1.5;"
```

```
cardinality=2
timeperobs=.1
;
id IDENT;
by ZONE;
run;
```

Note

This document is a guide for the use of the procedure PROC ERRORLOC. For more information on the methodology, please see the *Banff Functional Description* document.

The DETERMINISTIC Procedure

Overview

This procedure processes the data by applying the deterministic imputation method. This method supplies valid values to fields to impute in the case where only one possible imputed value will permit them to pass the set of edits.

Procedure Syntax

PROC DETERMINISTIC <*option(s)*>*;*

ID *variable;*

BY *variable(s)*

To do this	Use this statement
Identify the key variable of the input data set	ID
Perform imputation for each BY group	BY

PROC DETERMINISTIC Statement

PROC DETERMINISTIC <*option(s)*>*;*

To do this	Use this option
Specify the input data set	DATA=
Specify the SAS data set that contains the status of the fields (FTI) before imputation	INSTATUS=
Specify the SAS data set that contains the imputed data	OUT=
Specify the SAS data set that contains the status of the fields (IDE) after imputation	OUTSTATUS=
Specify the edits	EDITS=
Specify that negative values are valid values	ACCEPTNEGATIVE
Specify that negative values are invalid values	REJECTNEGATIVE

Options

DATA=*SAS-data-set*

specifies the input SAS data set. **The observations of this data set must be sorted by the values of the key variable specified on the ID statement**. **If BY variables are specified, the observations of this data set must also be sorted by the values of those variables. When sorting, the BY variables**

must be listed before the ID variable. If DATA= is omitted, the most recently created SAS data set is used.

INSTATUS=*SAS-data-set*

specifies the SAS data set that contains the status of the fields (FTI) before imputation. This SAS data set is mandatory. Three character variables must be in this data set: "*key variable*", FIELDID and STATUS. The "*key variable*" is the same as the one specified for the DATA= input data set and in the ID statement. FIELDID is the variable whose values are the names of the variables for which a STATUS of FTI is assigned.
If BY variables are specified: If all the BY variables specified are present on this data set, then the observations of this data set must be sorted by the values of those variables as well as by the values of the ID variable. When sorting, the BY variables must be listed before the ID variable.
If not all the BY variables specified are present on this data set, then the observations need only be sorted by the values of the ID variable.
Note: To increase the performance of the procedure, the BY variables should be in this data set.

OUT=*SAS-data-set*

names the output data set that contains the imputed data. Each observation contains the EDITS= variables where at least one variable has been imputed. If a BY statement is specified, then this data set also contains the BY variables. If you want the OUT= data set to be permanent, specify a two-level name.

OUTSTATUS=*SAS-data-set*

names the output data set that contains the variables that were successfully imputed, with status equal to IDE. The variables in this SAS data set are: "*key variable*", FIELDID and STATUS. If a BY statement is specified, then this data set also contains the BY variables. If you want the OUTSTATUS= data set to be permanent, specify a two-level name.

EDITS=*quoted string of all edits*

specifies the edit rules as linear equations. They must be enclosed in quotes and end with semi-colons. They are mandatory. It is important to note that the EDITS are used to tell the procedure which variables from the DATA= dataset will be processed.

ACCEPTNEGATIVE

if this option is specified, negative values will be considered valid values. Therefore it will be possible to impute a variable with a negative value.

Warning: Before using this option, please read the document Edits for Negative Values on specifying edits when processing negative numbers in Banff.

REJECTNEGATIVE

if this option is specified, it will not be possible to impute a variable with a negative value. For each variable involved in the rules of the EDITS= option, a positivity edit will be added. If an observation has a variable with a negative value without an FTI status code in the INSTATUS= dataset for that variable, the observation will be rejected.

Note: If neither ACCEPTNEGATIVE nor REJECTNEGATIVE is specified, the default value is REJECTNEGATIVE.

ID Statement

ID *variable*;

Required arguments

variable

specifies the variable that the procedure uses as the key variable of the input data set. It is mandatory. It must be a character variable and only one is allowed (no composite key).

BY Statement

BY *variable*-1 *... variable-n*;

Required arguments

variable(s)

specifies the variable(s) that the procedure uses to form BY groups. Deterministic imputation is performed on one observation at a time. So specifying BY variables will not influence the results. However, the BY variables will appear in the two output data sets. You can specify more than one variable. This statement is optional.

Details

- **The sorting of variables must be done in ascending order of the values.**
- **If BY variables are specified, the observations of the DATA= input data set must be sorted by the values of those variables, and by the values of the ID variable; when sorting the BY variables must be listed before the ID variable.**
- **If BY variables are specified: If all the BY variables specified are present on the INSTATUS= input data set, then the observations of this data set must be sorted by the values of those variables as well as by the values of the ID variable. When sorting, the BY variables must be listed before the ID variable.**
- **If not all the BY variables specified are present on the INSTATUS= input data set, then the observations need only be sorted by the ID variable.**
  Note: To increase the performance of the procedure, the BY variables should be in the INSTATUS= data set.
- If BY variables are specified, the BY variables specified will appear in the OUT= and OUTSTATUS= data sets.
- Observations with *a missing value for the key variable in the DATA= data set* will not be processed. A warning message will be entered in the log file with a counter for the number of observations dropped.
- Observations with a valid value for the key variable in the DATA= set but with *missing values in this data set* for one or more fieldids specified in the edit rules, *without the status* 'FTI' for these fieldids in the INSTATUS= data set will not be processed. A warning message will be entered in the log file with a counter for the number of observations dropped.

- If the REJECTNEGATIVE option is in effect, observations with a valid value for the key variable in the DATA= data set but with *negative values in this data set* for one or more fieldids specified in the edits *without the status* 'FTI' for these fieldids in the INSTATUS= data set will not be processed. A warning message will be entered in the log file with a counter for the number of observations dropped.
- If the option REJECTNEGATIVE is in effect, fields to impute will not be imputed with negative values.
- Observations with a *missing value for the key variable or a missing value for FIELDID in the INSTATUS= data set* will not be processed. A warning message will be entered in the log file with a counter for the number of observations dropped.
- The ACCEPTNEGATIVE and REJECTNEGATIVE options are mutually exclusive. Specifying both will cause the procedure to stop.

Example

```
data determdata;
infile cards;
input ident $ TOTAL Q1 Q2 Q3 Q4 staff prov;
cards;
REC01 500 100 125 125 150 2000 24
REC02 750 200 170 130 250 2500 24
REC03 400 80 90 100 130 1500 24
REC04 1000 150 250 350 250 3500 24
REC05 3000 500 500 1000 1000 5000 24
REC06 50 10 15 500 20 100 24
REC07 600 110 140 230 45 2400 30
REC08 900 175 999 999 300 3000 30
REC09 2500 400 555 600 5000 89 30
REC10 800 11 12 13 14 2800 30
REC11 -25 -10 -5 -5 -10 3000 30
;
data determstat;
infile cards;
input fieldid $ status $ ident $;
cards;
Q3 FTI REC06
Q4 FTI REC07
Q2 FTI REC08
Q3 FTI REC08
Q4 FTI REC09
staff FTI REC09
Q1 FTI REC10
Q2 FTI REC10
Q3 FTI REC10
Q4 FTI REC10
Q4 FTI REC11
;
proc DETERMINISTIC
data=determdata
instatus=determstat
out=outdata
outstatus=outstatus
edits="Q1 + Q2 + Q3 + Q4 - TOTAL = 0;"
acceptnegative;
id ident;
by prov;
run;
```

Notes

This document is a guide for the use of the procedure PROC DETERMINISTIC. For more information on the methodology, please see the *Banff Functional Description* document.

The DONORIMPUTATION Procedure

Overview

This procedure performs donor imputation using a nearest neighbour approach to find, for each observation requiring imputation, the valid observation that is most similar to it and that will allow the imputed recipient observation to pass the user specified post imputation edits.
An observation requiring imputation is called a recipient. A recipient is an observation for which at least one variable of the linear equations is marked as FTI in the INSTATUS dataset.
A donor is an observation which does not have any variables marked as FTI in the INSTATUS dataset for the variables specified in the linear equations. **The procedure does not verify that donors pass the edits.**

Procedure Syntax

PROC DONORIMPUTATION <*option(s)*>*;*

ID *variable;*

MUSTMATCH *variable(s);*

DATAEXCLVAR *variable;*

BY *variable(s);*

To do this	Use this statement
Identify the key variable of the input data set	ID
Specify user defined matching fields	MUSTMATCH
Identify the variable of the input data set used to exclude donors	DATAEXCLVAR
Perform imputation for each BY group	BY

PROC DONORIMPUTATION Statement

PROC DONORIMPUTATION <*option(s)*>;

To do this	Use this option
Specify the input data set	DATA=
Specify the SAS data set that contains the status of the fields (FTI) before imputation	INSTATUS=
Specify the SAS data set that contains the imputed data	OUT=
Specify the SAS data set that contains the status of the fields (IDN) after imputation	OUTSTATUS=
Specify the SAS data set that contains the mapping of donor/recipient identifiers	DONORMAP=
Specify the edits	EDITS=
Specify the post imputation edits	POSTEDITS=

Specify the minimum number of donors required to perform imputation	MINDONORS=
Specify the minimum percentage of donors required to perform imputation	PCENTDONORS=
Specify the maximum number of donors to try when looking for a donor	N=
Specify the exclusion or non exclusion of donors having imputed values for at least one field in the edits rules	ELIGDON=
Specify to use random selection of donors for recipients without matching fields	RANDOM
Specify to add MFS, MFU and MFB matching field status to OUTSTATUS	MATCHFIELDSTAT
Specify the root to the random number generator	SEED=
Specify that negative values are valid values	ACCEPTNEGATIVE
Specify that negative values are invalid values	REJECTNEGATIVE
Specify the maximum number of times a donor can be used	NLIMIT
Specify the multiplier for ratio limit, in order to limit the number of times a donor can be used	MRL

Options

DATA=*SAS-data-set*

specifies the input SAS data set. **If BY variables are specified, the observations of this data set must be sorted by the values of those variables.** If DATA= is omitted, the most recently created SAS data set is used.

INSTATUS=*SAS-data-set*

specifies the SAS data set that contains the status of the fields (FTI) before imputation. This SAS data set is mandatory. Three character variables must be in this data set: "*key variable*", FIELDID and STATUS. The "*key variable*" is the same as the one specified on the input SAS data set and on the ID statement. FIELDID is the variable whose values are the names of the variables for which a STATUS of FTI is assigned.
If BY variables are specified: If all the BY variables specified are present on the INSTATUS= input data set, then the observations of this data set must be sorted by the values of the BY variables.
Note: To increase the performance of the procedure, the BY variables should be in the INSTATUS= data set.

OUT=*SAS-data-set*

names the output data set that contains the imputed data. Each observation contains the EDITS= variables where at least one variable has been imputed. If a BY statement is specified, then this data set also contains the BY variables. If you want the OUT= data set to be permanent, specify a two-level name.

OUTSTATUS=*SAS-data-set*

names the output data set that contains the fieldids of the variables that were successfully imputed, with status equal to IDN. The variables in this SAS data set are: "*key variable*", FIELDID and STATUS. If a BY statement is specified, then this data set also contains the BY variables. If you want the OUTSTATUS= data set to be permanent, specify a two-level name.

DONORMAP=*SAS-data-set*

names the output data set that contains the identifiers of recipients that have been imputed along with their donor identifier and the number of donors tried before the recipient passed the postedits. This number will not be greater than option N unless some donors are found with the same distance from the recipient. This number will be zero if the donor was found randomly. If a BY statement is specified, then this data set also contains the BY variables. If you want the DONORMAP= data set to be permanent, specify a two-level name.

EDITS=*quoted string of all edits*

specifies the original edit rules as linear equations. They must be enclosed in quotes and end with semi-colons. They are mandatory. The original edits are used to find matching fields and to tell the procedure which variables from the DATA=dataset will be processed.

POSTEDITS=*quoted string of all edits*

specifies the post-imputation edit rules as linear equations. The post imputation edits determine if the imputed recipient has been successfully imputed or not. They should be enclosed in quotes and separated by semi-colons. They are optional. If they are not specified, the (original) EDITS will be used.

MINDONORS=*positive integer*

represents the minimum number of donors needed in the current BY group in order for imputation to be performed (on the current BY group). This number is optional and has a default of 30.

PCENTDONORS=*real number*

represents the minimum percentage of donors needed in the current BY group in order for imputation to be performed (on the current BY group). This number is optional and has a default of 30.0. Its value must be between 1 and 100 inclusively.

N=*positive integer*

represents the maximum number of donors to try to eventually find a suitable donor. This number is mandatory.

ELIGDON=*keyword*

used to include or exclude from all potential donors the donors with imputed values for at least one field of the edits. Acceptable keywords are **A** or **ANY** (to include all potential donors), **O** or **ORIGINAL** (to exclude donors with Ixxx fields, except IDE coming from deterministic imputation). Optional. If omitted, ORIGINAL is the default value.

RANDOM

if this option is specified then random selection of a donor will be applied to recipients **without** matching fields. If this option is not set, no imputation will be performed for recipients **without** matching fields.

Note: No matter what the value of this option is, only the nearest neighbour method will be applied for recipients with matching fields.

MATCHFIELDSTAT

if this option is specified, then the statuses of all matching fields will be added to the OUTSTATUS= dataset for each recipient. The status of a matching field can take on one of three possible values: MFS (system matching field), MFU (user-specified matching field), or MFB (matching field that is both system and user-specified). User-specified matching fields will be the ones specified, if any, on the MUSTMATCH statement.

SEED=*positive integer*

this parameter is optional and is useful only in development when results need to be compared from one run to the next. The default value is a random number. If a negative value, 0 or a value exceeding the maximum acceptable by the platform is specified, it will be replaced by the default value.

ACCEPTNEGATIVE

if this option is specified, negative values will be considered valid values. Therefore it will be possible to impute a variable with a negative value.

Warning: Before using this option, please read the document Edits for Negative Values on specifying edits when processing negative numbers in Banff.

REJECTNEGATIVE

if this option is specified, it will not be possible to impute a variable with a negative value. For each variable involved in the edits of the EDITS= option, a positivity edit will be added. If an observation has a variable with a negative value but without an FTI status code in the STATUS= dataset, the observation will be rejected.

NOTE: If neither ACCEPTNEGATIVE nor REJECTNEGATIVE is specified, the default value is REJECTNEGATIVE.

NLIMIT=*positive integer*

if this option is specified, it will limit the number of times a donor can be used. NLIMIT and MRL are optional parameters used to calculate DONORLIMIT. One or both can be specified. When both are specified, DONORLIMIT takes the rounded up maximum value between NLIMIT and the ratio using the MRL. If NLIMIT and MRL are omitted, the number of times a donor can be used is unlimited.

MRL=*positive real number*

if this option is specified, it will limit the number of times a donor can be used. The MRL (multiplier ratio limit) is multiplied by the ratio of the number of recipients to donors. NLIMIT and MRL are optional parameters used to calculate DONORLIMIT. One or both can be specified. When both are specified, DONORLIMIT takes the rounded up maximum value between NLIMIT and the ratio using the MRL. If NLIMIT and MRL are omitted, the number of times a donor can be used is unlimited.

ID Statement

ID *variable*;

Required argument

variable

specifies the variable that the procedure uses as the key variable of the DATA= input data set. It is mandatory. It must be a character variable and only one is allowed (no composite key).

MUSTMATCH Statement

MUSTMATCH *variable*-1 *... variable-n*;

Required arguments

variables

specifies the variable(s) that the procedure uses as "user matching fields". This statement is optional.

NOTE: The "user matching fields" are matching fields that are applied to ALL recipients. The system also finds "system matching fields" that are specific to each recipient even when this statement is not specified.

DATAEXCLVAR Statement

DATAEXCLVAR *variable*;

Required argument

variable

specifies the variable that the procedure uses to exclude observations from the pool of donors when reading the input data set DATA=. This statement is optional.

NOTE: This variable is a character variable that must be created prior to calling PROC DONORIMPUTATION and that must be added to the DATA= input SAS data set. The variable will take the value 'E' if a donor is to be excluded. Observations identified as recipients will never be excluded, no matter what value this variable takes. (See example)

BY Statement

BY *variable*-1 *... variable-n*;

Required arguments

Variable(s)

specifies the variable(s) that the procedure uses to form BY groups. Imputation will be performed on each group independently. You can specify more than one variable. This statement is optional.

Details

- **The sorting of variables must be done in ascending order of the values.**
- **If BY variables are specified, the observations of the DATA= input data set must be sorted by the values of those variables.**
- **If BY variables are specified: If all the BY variables specified are present on the INSTATUS= input data set, then the observations of this data set must be sorted by the values of the BY variables.**
  Note: To increase the performance of the procedure, the BY variables should be in the INSTATUS= data set.
- If BY variables are specified, the BY variables will appear in the OUT=, OUTSTATUS= and DONORMAP= data sets.
- Observations with *a missing value for the key variable in the input data set* DATA= will not be processed. A warning message will be entered in the log file with a counter for the number of observations dropped.
- Observations with a valid value for the key variable in the input data set DATA= but with *missing values in this data set* for one or more fieldids specified in the edit rules, *without the status* 'FTI' for these fieldids in the INSTATUS= data set, will not be processed. A warning message will be entered in the log file with a counter for the number of observations dropped.
- Observations with a valid value for the key variable in the input data set DATA= but with *missing values in this data set* for one or more fieldids specified in the MUSTMATCH statement, *without the status* 'FTI' for these fieldids in the INSTATUS= data set, will not be processed. A warning message will be entered in the log file with a counter for the number of observations dropped.
- If the REJECTNEGATIVE option is in effect, observations with a valid value for the key variable in the input data set DATA= but with *negative values in this data set* for one or more fieldids specified in the edit rules, *without the status* 'FTI' for these fieldids in the INSTATUS= data set, will not be processed. A warning message will be entered in the log file with a counter for the number of observations dropped.
- If the REJECTNEGATIVE option is in effect, observations with a valid value for the key variable in the input data set DATA= but with *negative values in this data set* for one or more fieldids specified in the MUSTMATCH statement, *without the status* 'FTI' for these fieldids in the INSTATUS= data set, will not be processed. A warning message will be entered in the log file with a counter for the number of observations dropped.
- If the option REJECTNEGATIVE is in effect, fields to impute will not be imputed with negative values.
- Observations with valid values in the input data set DATA= for the key variable and for the variables in the edit rules, and with one or more MUSTMATCH fields outside the edits marked with the status 'FTI' in the INSTATUS= data set will not be processed. These observations are called *mixed observations*. A warning message will be entered in the log file with a counter for the number of mixed observations dropped.
- Observations with a *missing value for the key variable or a missing value for FIELDID in the INSTATUS= data set* will not be processed. A warning message will be entered in the log file with a counter for the number of observations dropped.

- A donor having a fieldid with the status 'FTE' will not be used to impute the value of the same fieldid for a recipient.
- A variable cannot be specified in more than one ID, BY, EDITS, MUSTMATCH or DATAEXCLVAR statement. Except for the MUSTMATCH variables which can also be in the EDITS, these lists of variables are mutually exclusive.
- Variables with an "IDE" status in the INSTATUS data set are considered as having original values (not previously imputed).
- The ACCEPTNEGATIVE and REJECTNEGATIVE options are mutually exclusive. Specifying both will cause the procedure to stop.
- If the NLIMIT or the MRL option is in effect, details will be entered in the log file with regards to the ratio of donors that have reached $DONORLIMIT_j$ for each group.
- If the NLIMIT or MRL option is in effect, DONORLIMIT data will be added to the DONORMAP= dataset. When these parameters are omitted, the DONORLIMIT variable will remain empty in the DONORMAP= dataset.
- When limiting donors with the NLIMIT option, the number of remaining donors may end up being less than MINDONORS. In such a case, the procedure will continue and ignore MINDONORS which was validated at the beginning. The same applies for PCENTDONORS.

Example

```
/* create the data= data set */
data donordata;
infile cards;
input IDENT $ TOTAL Q1 Q2 Q3 Q4 STAFF PROV;
cards;
REC01 500 120 150 80 150 50 24
REC02 750 200 170 130 250 75 24
REC03 400 80 90 100 130 40 24
REC04 1000 150 250 350 250 100 24
REC05 1050 200 225 325 300 100 24
REC06 500 100 125 5000 130 45 24
REC07 400 80 90 100 15 40 30
REC08 950 150 999 999 200 90 30
REC09 1025 200 225 300 10 10 30
REC10 800 11 12 13 14 80 30
REC11 -25 -10 -5 -5 -10 3000 30
REC12 1000 150 250 350 250 100 30
REC13 999 200 225 325 300 100 30
REC14 -25 -10 -5 -10 -5 3000 30
;
/* create the exclusion variable */
data donordata;
set donordata;
if (TOTAL > 1000) then TOEXCL='E';
else TOEXCL='';
run;
/* create the instatus data set */
data donorstat;
infile cards;
input FIELDID $ STATUS $ IDENT $;
cards;
Q3 IPR REC01
Q4 FTE REC04
Q3 FTI REC06
Q2 FTI REC07
Q2 FTI REC08
```

```
Q3 FTI REC08
Q4 FTI REC09
STAFF FTI REC09
Q1 FTI REC10
Q2 FTI REC10
Q3 FTI REC10
Q4 FTI REC10
Q1 FTI REC11
Q2 FTI REC11
Q3 FTI REC11
Q4 FTI REC11
Q2 FTI REC13
Q3 FTI REC13
;
/* call the donorimputation procedure */
proc DONORIMPUTATION
data=donordata
instatus=donorstat
out=donorout
outstatus=outstat
donormap=donormap
edits="Q1 + Q2 + Q3 + Q4 - TOTAL = 0;"
postedits="Q1 + Q2 + Q3 + Q4 - TOTAL <= 0;"
mindonors=1
pcentdonors=1
n=1
nlimit=1
mrl=0.5
random
eligdon=original
matchfieldstat
acceptnegative
;
id IDENT;
mustmatch STAFF;
dataexclvar TOEXCL;
by prov;
run;
```

Note

This document is a guide for the use of the procedure PROC DONORIMPUTATION. For more information on the methodology, please see the *Banff Functional Description* document.

The ESTIMATOR Procedure

Overview

This procedure performs imputation by estimation. It imputes one variable at a time using a variety of imputation estimators. Users may choose from twenty (20) pre-defined imputation estimator algorithms that are hard-coded in the procedure, or may specify their own custom-defined algorithms (the terms "imputation estimators" and "algorithms" are interchangeable). There are two types of algorithms available: estimator functions and linear regression estimators. These algorithms may reference current and historical data.

Procedure Syntax

PROC ESTIMATOR <*option(s)*>*;*

ID *variable;*

DATAEXCLVAR *variable*

HISTEXCLVAR *variable;*

BY *variable(s);*

To do this	Use this statement
Identify the key variable of the input data sets	ID
Identify the variable of the current input data set used to exclude observations from the set of acceptable observation when calculating parameters	DATAEXCLVAR
Identify the variable of the historical input data set used to exclude observations from the set of acceptable observation when calculating parameters	HISTEXCLVAR
Perform imputation for each BY group	BY

PROC ESTIMATOR Statement

PROC ESTIMATOR <*option(s)*>;

To do this	Use this option
Specify the current input data set	DATA=
Specify the data set that contains the status of the current input data set variables before imputation	DATASTATUS=
Specify the historical input data set	HIST=
Specify the data set that contains the status of the historical input data set variables	HISTSTATUS=
Specify the data set that contains the imputed data	OUT=
Specify the data set that contains the status of the imputed	OUTSTATUS=

data	
Specify the data set that contains the random error report	OUTRANDOMERROR=
Specify the data set that contains the report on imputation statistics: estimator functions with or without parameter(s), and linear regression estimators	OUTESTPARMS=
Specify the data set that contains the report on calculation of averages: estimator functions (type EF with at least one parameter)	OUTESTEF=
Specify the data set that contains the report on calculation of « beta » coefficients: linear regression estimators (type LR)	OUTESTLR=
Specify the data set that contains the report on acceptable observations retained to calculate parameters by estimator: estimator functions with at least one parameter and linear regression estimator	OUTACCEPTABLE=
Specify the data set that contains the user defined algorithms	ALGORITHM=
Specify the data set that contains the estimator specifications	ESTIMATOR=
Specify the seed to the random number generator	SEED=
Specify to verify the specifications without doing imputation	VERIFYSPECS
Specify that negative values are valid values	ACCEPTNEGATIVE
Specify that negative values are invalid values	REJECTNEGATIVE

Options

DATA=*SAS-data-set*

specifies the current input SAS data set. **If BY variables are specified, the observations of this data set must be sorted by the values of those variables.** If DATA= is omitted, the most recently created SAS data set is used.

DATASTATUS=*SAS-data-set*

specifies the SAS data set that contains the status of the current input data set variables before imputation. This SAS data set is mandatory. Three character variables must be in this data set: "*key variable*", FIELDID and STATUS. The "*key variable*" is the same as the one in the input SAS data set and on the ID statement. FIELDID is the variable whose values are the names of the variables for which a STATUS of FTI, FTE or I* is assigned (* represents DN for donor imputation, DE for deterministic imputation, etc.).
If BY variables are specified: If all the BY variables specified are present on the DATASTATUS= input data set, then the observations of this data set must be sorted by the values of the BY variables.
Note: To increase the performance of the procedure, the BY variables should be in the DATASTATUS= data set.

HIST=*SAS-data-set*

specifies the historical input SAS data set. This SAS data set is mandatory only if historical variables are involved in one of the algorithms. Only observations that match on "*key variable*" with the DATA= data set are kept. **If BY variables are specified, the observations of this data set must be sorted by the values of those variables.**

HISTSTATUS=*SAS-data-set*

specifies the SAS data set that contains the status of the historical input data set variables before imputation. This SAS data set is optional. If it is not specified but the HIST= data set is specified then all historical values will be considered valid values.
If BY variables are specified: If all the BY variables specified are present on the HISTSTATUS= input data set, then the observations of this data set must be sorted by the values of the BY variables.
Note: To increase the performance of the procedure, the BY variables should be in the HISTSTATUS= data set.
See DATASTATUS= for more details on this data set.

OUT=*SAS-data-set*

specifies the data set that contains the imputed data. Each observation contains the FIELDID variables from the ESTIMATOR= data set that were imputed. If a BY statement is specified, then this data set also contains the BY variables. If you want the OUT= data set to be permanent, specify a two-level name.

OUTSTATUS=*SAS-data-set*

specifies the data set that contains the status of the imputed data. The status depends on the estimator specifications, but it will start with a leading "I". The variables in this SAS data set are: "*key variable*", FIELDID and STATUS. If a BY statement is specified, then this data set also contains the BY variables. If you want the OUTSTATUS= data set to be permanent, specify a two-level name.

OUTRANDOMERROR=*SAS-data-set*

specifies the data set that contains the random error report. This SAS data set is optional. If this option is not specified, no data set will be generated. If it is specified and no estimator adds a random error to the imputed variable (by setting the RANDOMERROR variable in the ESTIMATOR= data set to "Y"), an empty data set will be created. The variables in this SAS data set are:

ESTIMID	Identification number of the estimator
ALGORITHMNAME	The name of the algorithm
RECIPIENT	Key of the recipient
DONOR	Key of the donor
FIELDID	The name of the imputed variable
RESIDUAL	Original value of a donor - estimated value for the same observation
RANDOMERROR	If algorithm type is LR and variance is used then it is equal to: RESIDUAL* sqrt ((recipient variance ^ exponent) / (donor variance ^ exponent)) otherwise it is the same as RESIDUAL
ORIGINALVALUE	The reported value of the variable
IMPUTEDVALUE	The estimated value of the variable

OUTESTPARMS=*SAS-data-set*

specifies the output SAS data set that contains the report on imputation statistics by estimator (estimator functions with or without parameters, and linear regression estimators). This information is similar to the one written in the SAS log. This SAS data set is optional. If this option is not specified, no data set will be generated. The variables in this SAS data set are:

ESTIMID	Identification number for the estimator (begins with number 0) type: numeric
ALGORITHMNAME	The name of the algorithm
FIELDID	The name of the variable to impute
FTI	Number of imputations to do
IMP	Number of imputations done with success
DIVISIONBYZERO	Number of imputations failed because calculation implies a division by 0
NEGATIVE	Number of imputations discarded because imputed value is negative and option REJECTNEGATIVE is specified Note: This variable is not shown if option ACCEPTNEGATIVE is specified

OUTESTEF=*SAS-data-set*

specifies the output SAS data set that contains the report on the calculation of averages for estimator functions (type EF with at least one parameter). This information is similar to the one written in the SAS log. This SAS data set is optional. If this option is not specified, no data set will be generated. If it is specified and no estimator function (type EF) has at least one parameter, an empty data set will be created. The variables in this SAS data set are:

ESTIMID	Identification number of the estimator function (type EF with at least one parameter) type: numeric
ALGORITHMNAME	The name of the algorithm
FIELDID	The name of the variable for which an average is calculated
PERIOD	Current (C) or historical (H) period specific for the FIELDID variable type: character (length = 1)
AVERAGE_VALUE	Value of average of the variable
COUNT	Number of acceptable observations used to calculate the average The number of acceptable observations is the same in the calculation of all averages present in the formula of one estimator.

OUTESTLR=*SAS-data-set*

specifies the output SAS data set that contains the report on the calculation of « beta » coefficients for linear regression estimators (type LR). This information is similar to the one written in the SAS log. This SAS data set is optional. If this option is not specified, no data set will be generated. If it is specified and no linear regression is being performed (type LR), an empty data set will be created. The variables in this SAS data set are:

ESTIMID	Identification number of the linear regression estimator (type LR) type: numeric
ALGORITHMNAME	The name of the algorithm
FIELDID	The name of the variable or regressor for which a beta coefficient is

	calculated
EXPONENT	Regressor exponent
PERIOD	Current (C) or historical (H) period specific to the regressor type: character (length = 1)
BETA_VALUE	Value of « beta » coefficient associated with the regressor
COUNT	Number of acceptable observations used to calculate the « beta » coefficients The number of acceptable observations is the same for all « beta » coefficients present in the formula of one estimator identification

OUTACCEPTABLE=*SAS-data-set*

specifies the output SAS data set that contains the report on acceptable observations retained to calculate the parameters for each estimator given in the specifications (estimator functions with at least one parameter and linear regression estimators). This SAS data set is optional. If this option is not specified, no data set will be generated. If it is specified and no estimator function with at least one parameter (type EF) or no linear regression (type LR) is being performed, an empty data set will be created. The variables in this SAS data set are:

ESTIMID	Identification number for the estimator (type EF with at least one parameter and type LR) type: numeric
ALGORITHMNAME	The name of the algorithm
«key variable»	“Key” value of observation retained in parameter calculation for the estimator

ALGORITHM=*SAS-data-set*

specifies the SAS data set that contains the user defined algorithms. This SAS data set is optional. If an algorithm ALGORITHMNAME is the same as a predefined one, the predefined one will be replaced.

The following table describes the variables that must appear in the data set:

ALGORITHMNAME	The name of the algorithm. Mandatory
TYPE	The type of the algorithm 2 types are possible: "EF" for estimator function and "LR" for linear regression Mandatory
STATUS	A 1 to 3 character string that will be inserted in the OUTSTATUS= status variable when a variable is estimated by this algorithm. This string will be prefixed with "I" in the OUTSTATUS= data set. Mandatory
FORMULA	Holds the formula of the algorithm. The syntax of the formula is dependent on the type of the algorithm. Follow the links to see how to specify estimator function and linear regression. Only placeholders like AUX1 or FIELDID can be used in FORMULA. Mandatory
DESCRIPTION	A text to explain the purpose of the algorithm Optional

ESTIMATOR=*SAS-data-set*

specifies the SAS data set that contains the estimator specifications. This SAS data set is mandatory. **The estimators will be processed in the order in which they appear in the data set.**

The following table describes the variables of the data set:

ALGORITHMNAME	**The name of the algorithm. It can be a predefined algorithm or one found in the ALGORITHM= data set. Mandatory**
FIELDID	The name of the variable to be imputed. It must exist on the DATA= data set. Mandatory
AUXVARIABLES	A comma separated list of variable names These names are used to replace the placeholders in the FORMULA variable of the ALGORITHM= data set or in the formula of the pre-defined algorithms. They must exist in the DATA= or HIST= data sets. Note: The first variable name will replace AUX1 in the FORMULA variable, the second, AUX2 and so on.
WEIGHTVARIABLE	The name of the variable used as the weight variable It is optional if the algorithm calculates a parameter, otherwise it must be left blank. For "EF" algorithms, the variable must exist on the DATA= and/or the HIST= data set depending on the period of the average requested. For "LR" algorithms, the variable must exist on the DATA= data set.
VARIANCEVARIABLE	The name of the variable used as the variance variable It is optional if the algorithm is of type "LR", otherwise it must be left blank. It must exist on the DATA= or HIST= data set depending on the value of VARIANCEPERIOD. Note: This variable and the next two variables concern LR algorithms only. If VARIANCEVARIABLE is specified, the 2 others must be specified.
VARIANCEEXPONENT	A number indicating the power of the variance variable
VARIANCEPERIOD	The period of the variance variable Two periods are possible: "C" for current and "H" for historical
EXCLUDEIMPUTED	A flag indicating whether observations with variables with STATUS starting with "I*" (except "IDE") should be included in or excluded from the set of acceptable observations. 2 choices are possible: "Y" to exclude variables with status "I*" and "N" to include variables with status "I*". It is mandatory when an algorithm is computing a parameter. For algorithms not calculating parameters, it must be left blank.
EXCLUDEOUTLIERS	A flag indicating whether observations with variables with STATUS="FTE" should be included in or excluded from the set of acceptable observations. 2 choices are possible: "Y" to exclude variables with status "FTE" and "N" to include variables with status "FTE". It is mandatory when an algorithm is computing a parameter. For algorithms not calculating parameters, it must be left blank.

COUNTCRITERIA	A positive integer indicating the minimum number of acceptable observations needed in the current BY group in order for imputation to be performed (on the current BY group). It is mandatory when an algorithm is computing a parameter. For algorithms not calculating parameters, it must be left blank.
PERCENTCRITERIA	A real number between 0 and 100 (excluding 0 and 100) indicating the minimum percentage of acceptable observations needed in the current BY group in order for imputation to be performed (on the current BY group). It is mandatory when an algorithm is computing a parameter. For algorithms not calculating parameters, it must be left blank.
RANDOMERROR	A flag indicating whether a random error is added to the imputed variable. It is mandatory. 2 choices are possible: "Y" to add a random error and "N" to not add a random error. A warning will be printed if less than 5 observations are available for the random choice of the error.

SEED=*positive integer*

this parameter is optional and is useful only in development when results need to be compared from one run to the next. The default value is a random number. If a negative value, 0 or a value exceeding the maximum acceptable by the platform is specified, it will be replaced by the default value.

VERIFYSPECS

specifies that the user only wants to verify the specifications. The procedure verifies the specifications, it calculates parameters and it stops after printing them in the log window. Optional. If specified, the user may use the _NULL_ data set for the output data sets. All the input data sets still have to be specified.

ACCEPTNEGATIVE

if this option is specified, negative values will be considered valid values. They will be used in the calculation of parameters and to impute variables.

REJECTNEGATIVE

if this option is specified, negative values will not be used in the calculation of parameters. Variables to impute will not be imputed with negative values. If a variable involved in an estimator has a negative value and does not have a corresponding FTI flag on the STATUS= data set, the observation will be skipped.

NOTE: If neither ACCEPTNEGATIVE nor REJECTNEGATIVE is specified, the default value is REJECTNEGATIVE.

ID Statement

ID *variable*;

Required argument

variable

specifies the variable that the procedure uses as the key variable of the input data set. It is mandatory. It must be a character variable and only one is allowed (no composite key). DATA=, DATASTATUS=, HIST= and HISTSTATUS= data sets must have this variable.

DATAEXCLVAR Statement

DATAEXCLVAR *variable*;

Required argument

variable

specifies the character variable that the procedure uses to exclude current and the corresponding historical observations from the set of acceptable observations for computing parameters. When the value of the variable identified by DATAEXCLVAR= is 'E' the observation is not used to compute parameters. Even if the parameters to calculate involve only historical data (that's possible only in EF algorithms), the DATAEXCLVAR will be used to restrict the pool of acceptable records. This statement is optional.

NOTE : This character variable must be created prior to calling PROC ESTIMATOR and it must be added to the DATA= input SAS data set. (See example)

HISTEXCLVAR Statement

HISTEXCLVAR *variable*;

Required argument

variable

specifies the character variable that the procedure uses to exclude historical and the corresponding current observations from the set of acceptable observations for computing parameters. When the value of the variable identified by HISTEXCLVAR= is 'E' the observation is not used to compute parameters. Even if the parameters to calculate involve only current data, the HISTEXCLVAR will be used to restrict the pool of acceptable records. This statement is optional.

NOTE : This character variable must be created prior to calling PROC ESTIMATOR and it must be added to the HIST= input SAS data set. (See example)

BY Statement

BY *variable*-1 ... *variable-n*;

Required arguments

Variable(s)

specifies the variable(s) that the procedure uses to form BY groups. The variable(s) must exist on both the DATA= and the HIST= data sets. Imputation will be performed on each group independently. You can specify more than one variable. This statement is optional.

Predefined algorithms

A set of predefined algorithms has been provided. See the document on pre-defined algorithms.

Details

- **All sorting of variables must be done in ascending order.**
- **If BY variables are specified, the observations of the DATA= and HIST= data sets must be sorted by the values of those variables.**
- **If BY variables are specified then: If all the BY variables specified are present on the DATASTATUS= data set, then the data set must be sorted by the values of those BY variables.** Note: to increase the performance of the procedure, the BY variables should be on the DATASTATUS= data set.
- If BY variables are specified, the BY variables will appear in the OUT= and OUTSTATUS= data sets, and in the 5 output data sets described as “report”.
- The same applies to the HISTSTATUS= data set. The BY variables can be on either one, or both of the input status data sets.
- Observations with *a missing value for the key variable in the input data sets* will not be processed. A warning message will be entered in the log file with a counter for the number of observations dropped.
- Observations with a valid value for the key variable in the input data set but with *missing values in this data set* for one or more variables specified in the estimator specifications *without the status* 'FTI' for these fieldids in the STATUS= data set will not be processed. A warning message will be entered in the log file with a counter for the number of observations dropped.
- If the option REJECTNEGATIVE is in effect, observations with a valid value for the key variable in the input data set but with *negative values in this data set* for one or more variables specified in the estimator specifications *without the status* 'FTI' for these fieldids in the STATUS= data set will not be processed. A warning message will be entered in the log file with a counter for the number of observations dropped.
- If the option REJECTNEGATIVE is in effect, fields to impute will not be imputed with negative values.
- Observations with a *missing value for the key variable or a missing value for FIELDID in the STATUS= data set* will not be processed. A warning message will be entered in the log file with a counter for the number of observations dropped.
- Any weight variables with missing or negative values or with a status in the DATASTATUS or HISTSTATUS data sets will cause the procedure to stop.
- Any variance variables with missing, negative or zero (0) values or with a status in the DATASTATUS or HISTSTATUS data sets will cause the procedure to stop.
- The DATAEXCLVAR and HISTEXCLVAR statements can use the same variable name for the exclusion variable.

- A variable cannot be specified in more than one of the following statements: ID statement, BY statement, DATAECLVAR and HISTEXCLVAR statements and in the estimator specifications (weight variable, variance variable, field to impute, auxiliary variable, etc.). These lists of variables are mutually exclusive.
- Variables with an "IDE" status in the DATASTATUS data set are considered as having original values (not previously imputed).
- The ACCEPTNEGATIVE and REJECTNEGATIVE options are mutually exclusive. Specifying both will cause the procedure to stop.

Example 1

With option “acceptnegative” and report “outrandomerror”

```
data currentdata;
infile cards;
input ident $ x prov;
cards;
REC01 -500 24
REC02 750 24
REC03 -400 24
REC04 1000 24
REC05 1050 24
REC06 500 24
REC07 400 30
REC08 950 30
REC09 1025 30
REC10 -800 30
REC12 10000 30
REC13 500 24
REC14 750 24
REC15 400 24
REC16 -1000 24
REC17 1050 24
REC18 500 24
REC19 -400 30
REC20 950 30
REC21 -1025 30
REC22 800 30
REC23 10000 30
;
run;

/* Create the historical data set*/
data histdata;
infile cards;
input IDENT $ X prov;
cards;
REC01 500 24
REC02 -750 24
REC03 400 24
REC05 870 24
REC06 500 24
REC07 400 30
REC08 950 30
REC09 950 30
REC10 -800 30
REC11 800 30
REC12 500 24
```

```
REC13 750 24
REC14 -400 24
REC15 870 24
REC16 500 24
REC17 -400 30
REC18 950 30
REC19 950 30
REC20 800 30
REC21 800 30
;
run;

/* Create the status file for the current data set*/
/* This data set can also be created by running */
/* PROC ERRORLOC on the current data set with the */
/* edit x>0 */
data currstatus;
input ident $ fieldid $ status $;
cards;
REC01 x FTI
REC03 x FTI
REC10 x FTI
REC16 x FTI
REC19 x FTI
REC21 x FTI
;
run;

/* Create the status file for the historical data set*/
/* This data set can also be created by running */
/* PROC ERRORLOC on the historical data set with the */
/* edit x>0 */
data histstatus;
input ident $ fieldid $ status $;
cards;
REC02 X FTI
REC10 X FTI
REC14 X FTI
REC17 X FTI
;
run;

/* Create an exclusion variable called CURRDATAEXCL */
/* to be used with the DATAEXCLVAR statement */
/* of PROC ESTIMATOR */
data currentdata;
set currentdata;
if (x > 2000) then
CURRDATAEXCL='E';
else
CURRDATAEXCL='';
run;

/* Create an exclusion variable called HISTDATAEXCL */
/* to be used with the HISTEXCLVAR statement */
/* of PROC ESTIMATOR */
data histdata;
set histdata;
if (x > 1000) then
HISTDATAEXCL='E';
else
HISTDATAEXCL='';
run;
```

```
/* Here we define our own algorithm which redefines */
/* a predefined algorithm */
data algorithm;
length algorithmname $ 8;
length type $ 2;
length status $ 2;
length formula $ 30;
length formula $ 30;
algorithmname='prevalue';
type='ef';
status='vp';
formula='fieldid(h,v)';
description='Previous value is imputed.';
run;

/* Here we create the data set that contains the imputation */
/* strategy.  The estimators will be processed in the order in which */
/* they appear in the data set. */
data estimator;
length est_setid $ 4;
length algorithmname $ 8;
length excludeimputed $ 1;
length excludeimputed $ 1;
length randomerror $ 1;
length countcriteria 8;
length percentcriteria 8;
length weightvariable $ 15;
length variancevariable $ 15;
length varianceperiod $ 1;
length varianceexponent 8;
length fieldid $ 15;
length auxvariables $ 45;
est_setid='set1';
excludeimputed='y';
excludeoutliers='y';
randomerror='y';
algorithmname='prevalue';
fieldid='x';
output;
est_setid='set1';
excludeimputed='y';
excludeoutliers='y';
randomerror='y';
countcriteria=1;
percentcriteria=1;
algorithmname='curmean';
fieldid='x';
output;
est_setid='set2';
excludeimputed='n';
excludeoutliers='n';
randomerror='y';
countcriteria=1;
percentcriteria=1;
algorithmname='diftrend';
fieldid='x';
output;
est_setid='set2';
excludeimputed='y';
excludeoutliers='y';
randomerror='y';
countcriteria=1;
```

```
percentcriteria=1;
algorithmname='histreg';
fieldid='x';
output;
run;

/* Sort the data set and call PROC ESTIMATOR */
proc sort data=currentdata; by prov ident; run;
proc sort data=histdata; by prov ident; run;

proc estimator
data=currentdata
datastatus=currstatus
hist=histdata
algorithm=algorithm
estimator=estimator (where=(est_setid='set1'))
out=outdata
outstatus=outstatus
outrandomerror=outrandomerror
acceptnegative;
id ident;
dataexclvar CURRDATAEXCL;
histexclvar HISTDATAEXCL;
by prov;
run;
```

Example 2

With option “rejectnegative” (default value) and the 4 output reports “outestparms, outestef, outestlr and outacceptable”

```
proc estimator
data=currentdata
datastatus=currstatus
hist=histdata
algorithm=algorithm
estimator=estimator (where=(est_setid='set2'))
out=outdata
outstatus=outstatus
outrandomerror=outrandomerror
outestparms=outestparms
outestef=outestef
outestlr=outestlr
outacceptable=outacceptable
;
id ident;
dataexclvar CURRDATAEXCL;
histexclvar HISTDATAEXCL;
by prov;
run;
```

Note

This document is a guide for the use of the procedure PROC ESTIMATOR. For more information on the methodology, please see the *Banff Functional Description* document.

The PRORATE Procedure

Overview

This procedure processes the data by applying the prorating edit rules to the related values and raking them, with rounding, if needed. The raking will balance a summation by distributing the differential with the expected total across the summation components based on a specific weight associated with each of the components and only on components which might be changeable.

Procedure Syntax

PROC PRORATE *<option(s)>;*

ID *variable;*

BY *variable(s);*

To do this	Use this statement
Identify the key variable of the input data set	ID
Perform prorating for each BY group	BY

PROC PRORATE Statement

PROC PRORATE *<option(s)>*;

To do this	Use this option
Specify the input data set	DATA=
Specify the SAS data set that contains the status of the fields before prorating	INSTATUS=
Specify the SAS data set that contains the prorated data	OUT=
Specify the SAS data set that contains the status of the fields (IPR) after prorating	OUTSTATUS=
Specify the SAS data set that contains the rejected observations	OUTREJECT=
Specify the prorating edits	EDITS=
Specify the number of decimals used in the rounding algorithm	DECIMAL=
Specify the lower bound used to verify the prorating ratio	LOWERBOUND=
Specify the upper bound used to verify the prorating ratio	UPPERBOUND=
Specify the global variable modifier	MODIFIER=
Specify that the procedure only verify the EDITS	VERIFYEDITS
Specify the method of prorating	METHOD=
Specify that negative values are valid values	ACCEPTNEGATIVE
Specify that negative values are invalid values	REJECTNEGATIVE

Options

DATA=*SAS-data-set*

specifies the input SAS data set. **The observations of this data set must be sorted by the values of the key variable specified on the ID statement**. **If BY variables are specified, the observations of this data set must also be sorted by the values of those variables. When sorting, the BY variables must be listed before the ID variable**. If DATA= is omitted, the most recently created SAS data set is used.

NOTE: In the EDITS, the first letter of any MODIFIER (A, I, N, O) cannot be used as a variable name. The full word for the modifier however, can be used (i.e. ALWAYS is a valid variable name, but A is not).

INSTATUS=*SAS-data-set*

specifies the SAS data set that contains the status of the fields before prorating. Three character variables must be in this data set: "*key variable*", FIELDID and STATUS. The "*key variable*" is the same as the one specified for the input SAS data set with the ID statement. FIELDID is the variable whose values are the names of the variables for which a STATUS has been assigned by other imputation processes before prorating. A FIELDID is considered as imputed if the first letter of STATUS is 'I' (except for IDE). If INSTATUS= is omitted or is _NULL_, and at least one modifier is ORIGINAL or IMPUTED, an error will be generated and the procedure will stop.

If BY variables are specified: If all the BY variables specified are present on this data set, then the observations of this data set must be sorted by the values of those variables as well as by the values of the ID variable. When sorting, the BY variables must be listed before the ID variable.
If not all the BY variables specified are present on this data set, then the observations need only be sorted by the values of the ID variable.
Note: To increase the performance of the procedure, the BY variables should be in this data set.

If a variable has a status of FTI, its value will be processed as an original value.

OUT=*SAS-data-set*

names the output data set that contains the prorated observations. Each observation contains the EDITS= variables where at least one variable has been prorated. If you specify the VERIFYEDITS option, specify _NULL_ for this dataset, otherwise an empty dataset will be created. If a BY statement is specified, this data set also contains the BY variables. If you want the OUT= data set to be permanent, specify a two-level name.

OUTSTATUS=*SAS-data-set*

names the output data set that contains the fieldids of the variables that were successfully prorated, with status equal to IPR. The variables in this SAS data set are: "*key variable*", FIELDID and STATUS. If you specify the VERIFYEDITS option, specify _NULL_ for this dataset, otherwise an empty dataset will be created. If a BY statement is specified, then this data set also contains the BY variables. If you want the OUTSTATUS= data set to be permanent, specify a two-level name.

OUTREJECT=*SAS-data-set*

names the output data set that contains the observations for which prorating could not be performed. The variables created in this data set are: "*key variable*", NAME_ERROR, TOTAL_NAME, FIELDID and RATIO_ERROR. If you specify the VERIFYEDITS option, specify _NULL_ for this dataset, otherwise an empty dataset will be created. If a BY statement is specified, then this data set also contains the BY variables. If you want the OUTREJECT= data set to be permanent, specify a two-level name.

The following table describes the types of errors:

NAME_ERROR	Description
DECIMAL ERROR	The user has specified fewer decimal places than exist in the adjusted total.
SCALING VALUE K GREATER THAN +1	Only for “scaling method”. The acceptable interval for factor “k” is: -1 <= k <= +1 (Please refer to the Banff Functional Description for more information)
SCALING VALUE K LESS THAN -1	Only for “scaling method”. The acceptable interval for factor “k” is: -1 <= k <= +1 (Please refer to the Banff Functional Description for more information)
NOTHING TO PRORATE	No variables are left to prorate; they have all been eliminated because the modifier does not identify the variables as proratable (taking into account their status in INSTATUS= if the modifier is O or I) or their value is 0.
OUT OF BOUNDS	The rounded value divided by the original value is not within the interval defined by the bounds.
SUM OF PRORATABLE COLUMNS IS 0	The factor "k" cannot be calculated because the weighted sum of the proratable columns is 0. (Please refer to the Banff Functional Description for more information)
NEGATIVE VALUE IN DATA	A variable has a negative value and option REJECTNEGATIVE is specified.

For every type of error, the variable name of the total is reported under TOTAL_NAME. If the error is "*OUT OF BOUNDS* ", the variable name where it occurs is reported under FIELDID and the out of bounds ratio is reported under RATIO_ERROR.

EDITS=*quoted string of all edits*

specifies the prorating edit rules. They must be enclosed in quotes and end with semi-colons. They are mandatory. The edits may be entered in any order. Parts of sums and subtotals can be nested to an unlimited degree in leading to an overall fixed total. It is possible to specify weights for every field involved in a summation. In this way the relative amount of change of each field due to pro-rating may be controlled, change being inversely proportional to the weight given. Weights are applied at the variable level; i.e., the same weight applies across all records for that variable. The weights are positive numbers and are specified before the variable name. (See example). It is important to note that the EDITS are used to tell the procedure which variables from the DATA= dataset will be processed. Only one prorating group can be processed at a time.

DECIMAL=*positive integer*

specifies the number of decimals used in the rounding algorithm. Must be a positive integer between 0 and 9 inclusively. Optional. Default value is 0. The number of decimals specified must be equal to or greater than the actual number of decimals found on the total.

LOWERBOUND=*real number*

specifies the lower bound used to verify that the rounded value divided by the original value is within the interval. May be any value. Must be lower than the value specified as UPPERBOUND. Optional. If omitted, the lower limit is zero.

If the basic method of prorating is used, the lower bound may be any value if ACCEPTNEGATIVE option is specified, but must be greater than or equal to zero if REJECTNEGATIVE option is specified (see options METHOD= and ACCEPTNEGATIVE / REJECTNEGATIVE below). If the scaling method is used, the lower bound must be greater than or equal to zero whether ACCEPNEGATIVE or REJECTNEGATIVE is specified. The value of this limit must always be lower than the value specified as UPPERBOUND.

UPPERBOUND=*real number*

specifies the upper bound used to verify that the rounded value divided by the original value is within the interval. May be any value. Must be greater than the value specified as LOWERBOUND. Optional. If omitted, the upper limit is the highest numeric value possible (for the operating system).

If the basic method of prorating is used, the upper bound may be any value. If the scaling method is used, the upper bound must be greater than or equal to zero (see option METHOD= below). The value of this limit must always be greater than the value specified as LOWERBOUND.

MODIFIER=*keyword*

specifies the global modifier to use **if none is given for a variable in an edit**. Uppercase or lowercase. Optional. Default value is ALWAYS.

MODIFIER (uppercase or lowercase accepted)	Description
ALWAYS or A	Always change original and/or previously imputed data
IMPUTED or I	Change previously imputed data only
NEVER or N	Never change the data
ORIGINAL or O	Change only original data

VERIFYEDITS

specifies that the user only wants to verify the EDITS. The procedure stops after printing them in the log. Optional. If omitted, the procedure does not stop after the printing. If specified, the user may use the _NULL_ data set for the three output data sets and the INSTATUS= data set, **but not for the input DATA= dataset**.

METHOD=*keyword*

specifies the method the user wants to use for prorating. Uppercase or lowercase accepted. Optional. Default value is BASIC.

METHOD (uppercase or lowercase accepted)	Description
BASIC or B	Use the basic method of prorating. When this method is used, the signs of some variables may change during the prorating process.
SCALING or S	Use the "scaling method" of prorating. When this method is used, the signs of variables can never change, so that negative values will always remain negative, and positive values will always remain positive.

ACCEPTNEGATIVE

if this option is specified, negative values will be considered valid values. Therefore it will be possible to prorate a variable with a negative value.

REJECTNEGATIVE

if this option is specified, it will not be possible to prorate a variable with a negative value. An observation with a negative value for an edit variable will not be processed and will be added in the OUTREJECT= output data set. A warning message will also be entered in the log file with a counter for the number of observations dropped.

NOTE: If neither ACCEPTNEGATIVE nor REJECTNEGATIVE is specified, the default value is REJECTNEGATIVE.

ID Statement

ID *variable*;

Required argument

variable

specifies the variable that the procedure uses as the key variable of the input data set. It is mandatory. It must be a character variable and only one is allowed (i.e. no composite key).

BY Statement

BY *variable*-1 ... *variable-n*;

Required arguments

Variable(s)

specifies the variable(s) that the procedure uses to form BY groups. Prorating is performed on one observation at a time. So specifying BY variables will not influence the results. However, the BY variables will appear in the three output data sets. You can specify more than one variable. This statement is optional.

Details

- **The sorting of variables must be done in ascending order of the values.**
- **If BY variables are specified, the observations of the DATA= input data set must be sorted by the values of those variables, and by the values of the ID variable; when sorting the BY variables must be listed before the ID variable.**
- **If BY variables are specified: If all the BY variables specified are present on the INSTATUS= input data set, then the observations of this data set must be sorted by the values of those variables as well as by the values of the ID variable. When sorting, the BY variables must be listed before the ID variable.**
- **If not all the BY variables specified are present on the INSTATUS= input data set, then the observations need only be sorted by the ID variable.**
  Note: To increase the performance of the procedure, the BY variables should be in the INSTATUS= data set.
- If BY variables are specified, the BY variables specified will appear in the OUT=, OUTSTATUS= and OUTREJECT= data sets.
- Observations with *a missing value for the key variable in the input data set DATA*= will not be processed. A warning message will be entered in the log file with a counter for the number of observations dropped.
- Observations with *a missing value for the key variable or missing value for FIELDID in the input data set INSTATUS*= will not be processed. A warning message will be entered in the log file with a counter for the number of observations dropped.
- A missing value for an edit variable in the input data set DATA= will be processed as 0 and will reappear as missing in the output data set OUT=.
- Variables with an "IDE" status in the INSTATUS data set are considered as having original values (not previously imputed).

Example 1

Verify the prorating edits only.

```
options linesize=80 pagesize=32000;
%let proratingedits="
Q1:A+Q2:I+Q3:n+Q4:o=YEAR;
M1+M2+M3=Q1;
";
data proratedata(drop=n);
n = 1;
do while (n <= 10);
IDENT = 'R' || put (n, Z2.);
M1 = round ((10 + rannor (1)) / 3);
M2 = round ((10 + rannor (1)) / 3);
```

```
M3 = round ((10 + rannor (1)) / 3);
Q1 = round (10 + rannor (1) / 4);
Q2 = round (10 + rannor (1) / 4);
Q3 = round (10 + rannor (1) / 4);
Q4 = round (10 + rannor (1) / 4);
YEAR = 40;
output;
n = n + 1;
end;
run;
proc prorate
data=proratedata
edits=&proratingedits
verifyedits
;
id ident;
run;
```

Example 2

Perform prorating using the modifier specified in the edits, and for variables without a modifier, prorate only if the variable has been previously imputed (modifier = imputed). Reject negative data.

```
options linesize=80 pagesize=32000 nodate;
%let proratingedits="
Q1:A+Q2:I+Q3:n+Q4:o=YEAR;
M1+M2+M3=Q1;
";

data proratedata(drop=n);
n = 1;
do while (n <= 10);
IDENT = 'R' || put (n, Z2.);
M1 = round ((10 + rannor (1)) / 3);
M2 = round ((10 + rannor (1)) / 3);
M3 = round ((10 + rannor (1)) / 3);
Q1 = round (10 + rannor (1) / 4);
Q2 = round (10 + rannor (1) / 4);
Q3 = round (10 + rannor (1) / 4);
Q4 = round (10 + rannor (1) / 4);
YEAR = 40;
if mod(n,3)=0 then ZONE=1; else ZONE=2;
if mod(n,10)=0 then Q1=-Q1; /* negative data */
output;
n = n + 1;
end;
run;
data proratestatus;
infile datalines delimiter=',';
input ident $ Fieldid $ Status $;
datalines;
R01,M1,IDN
R01,M3,IDN
R02,M1,IPV
R05,M1,IDN
R05,Q3,IDN
R07,M3,IPV
R08,M1,IPV
```

```
R09,M2,IDN
R10,M1,IPV
R10,Q1,IPV
;
run;
proc sort data=proratedata; by zone ident; run;
proc prorate
data=proratedata
instatus=proratestatus
out=pro_outdata
outstatus=pro_outstatus
outreject=pro_rejected
edits=&proratingedits
method = scaling
decimal = 1
lowerbound = 0.1
upperbound = 1.1
modifier = imputed
rejectnegative
;
id ident;
by zone;
run;
```

Example 3

Compare handling of negative values (acceptnegative) by the two methods of prorating First run PROC PRORATE with the example below and then rerun by replacing method = BASIC with method = SCALING and setting lowerbound = 0.0. Note that there is no instatus= dataset, since all variables have the default modifier=A.

```
options linesize=80 pagesize=32000 nodate;
%let proratingedits="
Q1+4Q2+2Q3+Q4=YEAR;
";
data proratedata;
infile datalines delimiter=',';
input ident $ Q1 Q2 Q3 Q4 YEAR;
datalines;
R01,-25,42,27,25,40
R02,-25,44,20,20,40
R03,-18,44,15,5,40
R04,-18,42,15,5,40
R05,12,30,15,5,40
R06,12,20,10,5,40
run;
proc prorate
data=proratedata
out=pro_outdata
outstatus=pro_outstatus
outreject=pro_rejected
edits=&proratingedits
method = BASIC
decimal = 1
lowerbound = -100.0 /* set to 0.0 for method = SCALING */
upperbound = 100.0
modifier = always
acceptnegative
;
```

```
id ident;
run;
```

Notes

This document is a guide for the use of the procedure PROC PRORATE. For more information on the methodology, please see the *Banff Functional Description* document.

The MASSIMPUTATION Procedure

Overview

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. 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. Also, it is assumed that the set of core information collected from the entire sample and the extra items collected from the subsample have already been edited and imputed and that no consistency edits need to be applied, either to the individual sections or between two sections of the questionnaires.

This procedure performs massive imputation using a nearest neighbour approach to find a valid observation that is most similar to the one which needs imputation, without searching for "system matching fields" as PROC DONORIMPUTATION does, but using only the "user matching fields". In the case where no "user matching fields" are specified, a random selection of the donor will be performed. No post imputation edits are needed since the first nearest neighbour found will be kept for imputation.

An observation requiring imputation is called a recipient. A recipient is an observation for which all variables to impute are missing on the input data set.

A valid observation is called a donor. A donor does not have any missing values for variables defined as the ones to impute.

Procedure Syntax

PROC MASSIMPUTATION *<options(s)>;*

ID *variable;*

MUSTIMPUTE *variable(s);*

MUSTMATCH *variable(s);*

BY *variable(s);*

To do this	Use this statement
Identify the key variable of the input data set	ID
Specify field(s) to impute	MUSTIMPUTE
Specify user defined matching field(s)	MUSTMATCH
Perform imputation for each BY group	BY

PROC MASSIMPUTATION Statement

PROC MASSIMPUTATION <*option(s)*>;

To do this	Use this option
Specify the input data set	DATA=
Specify the SAS data set that contains the imputed data	OUT=
Specify the SAS data set that contains the mapping of donor/recipient identifiers	DONORMAP=
Specify the minimum number of donors required to perform imputation	MINDONORS=
Specify the minimum percentage of donors required to perform imputation	PCENTDONORS=
Specify to use random selection of donors for recipients without matching fields	RANDOM
Specify the root to the random number generator	SEED=
Specify that negative values are valid values	ACCEPTNEGATIVE
Specify that negative values are invalid values	REJECTNEGATIVE
Specify the maximum number of times a donor can be used	NLIMIT
Specify the multiplier for ratio limit, in order to limit the number of times a donor can be used	MRL

Options

DATA=*SAS-data-set*

specifies the input SAS data set. The variables required are the "*key variable*", and all the variables listed on the MUSTIMPUTE statement and on the MUSTMATCH statement. The "*key variable*" must be the same as the one specified in the ID statement. **If BY variables are specified, the observations of this data set must be sorted by the values of those variables.** If DATA= is omitted, the most recently created SAS data set is used.

OUT=*SAS-data-set*

names the output data set that contains the identifiers of recipients that have been imputed and the imputed values for the variables listed on the MUSTIMPUTE statement. If a BY statement is specified, then this data set also contains the BY variables. If you want the OUT= data set to be permanent, specify a two-level name.

DONORMAP=*SAS-data-set*

names the output data set that contains the identifiers of recipients that have been imputed along with their donor identifier and the number of donors tried. This number will be one (1) if the donor was found by the nearest neighbour method or zero (0) if it was found randomly. If a BY statement is specified, then this data set also contains the BY variables. If you want the DONORMAP= data set to be permanent, specify a two-level name.

MINDONORS=*positive integer*

represents the minimum number of donors needed in the current BY group in order for imputation to be performed (on the current BY group). This number is optional and has a default of 30.

PCENTDONORS=*real number*

represents the minimum percentage of donors needed in the current BY group in order for imputation to be performed (on the current BY group). This number is optional and has a default of 30.0.

RANDOM

optional. If this option is specified with the MUSTMATCH statement, then random selection will be applied to recipients with missing values for all MUSTMATCH fields (and/or all negative if the REJECTNEGATIVE option is in effect). If this option is specified without the MUSTMATCH statement, then random selection will be applied to all recipients, without using the nearest neighbour method. If this option is not specified but the MUSTMATCH statement is specified, no random selection will be applied to recipients with missing values (and/or negative if REJECTNEGATIVE is specified) for all MUSTMATCH fields.

SEED=*positive integer*

this parameter is optional and is useful only in development when results need to be compared from one run to the next. The default value is a random number. If a negative value, 0 or a value exceeding the maximum acceptable by the platform is specified, it will be replaced by the default value.

ACCEPTNEGATIVE

if this option is specified, negative values will be considered valid values. Therefore it will be possible to impute a variable with a negative value.

REJECTNEGATIVE

if this option is specified, negative values will not be considered valid and it will not be possible to impute a variable with a negative value.

NOTE: If neither ACCEPTNEGATIVE nor REJECTNEGATIVE is specified, the default value is REJECTNEGATIVE.

NLIMIT=*positive integer*

if this option is specified, it will limit the number of times a donor can be used. NLIMIT and MRL are optional parameters used to calculate DONORLIMIT. One or both can be specified. When both are speficied, DONORLIMIT takes the rounded up maximum value between NLIMIT and the ratio using the MRL. If NLIMIT and MRL are **omitted**, the number of times a donor can be used is unlimited.

MRL=*positive real number*

if this option is specified, it will limit the number of times a donor can be used. The MRL (multiplier ratio limit) is multiplied by the ratio of the number of recipients to donors. NLIMIT and MRL are optional parameters used to calculate DONORLIMIT. One or both can be specified. When both are specified, DONORLIMIT takes the rounded up maximum value between NLIMIT and the ratio using the MRL. If NLIMIT and MRL are omitted, the number of times a donor can be used is unlimited.

ID Statement

ID *variable*;

Required argument

variable

specifies the variable that the procedure uses as the "*key variable*" of the input data set. It is mandatory. It must be a character variable and only one is allowed (no composite key).

MUSTIMPUTE Statement

MUSTIMPUTE *variable*-1 ... *variable-n*;

Required arguments

variables

specifies the variable(s) to be imputed. The procedure uses these variables to define an observation as being a recipient or a potential donor on the input data set. To be a recipient, all the variables listed on the MUSTIMPUTE statement must be missing for an observation. To be a potential donor, all the variables of an observation listed on the MUSTIMPUTE statement must have non missing values (and non negative values if REJECTNEGATIVE is specified). This statement is mandatory.

MUSTMATCH Statement

MUSTMATCH *variable*-1 ... *variable-n*;

Required arguments

variables

specifies the variable(s) that the procedure uses as "user matching fields". This statement is optional. If no statement is given, option RANDOM must be specified and a donor will be selected randomly for each recipient.

NOTE: The "user matching fields" *variables are matching fields that apply to ALL recipients. The system does not search for "system matching fields" specific to each recipient.*

BY Statement

BY *variable*-1 ... *variable-n*;

Required arguments

Variable(s)

specifies the variable(s) that the procedure uses to form BY groups. Imputation will be performed on each group independently. You can specify more than one variable. This statement is optional.

Details

- **The sorting of variables must be done in ascending order of the values.**
- **If BY variables are specified, the observations of the DATA= input data set must be sorted by the values of those variables.**
- If BY variables are specified, the BY variables specified will appear in the OUT= and DONORMAP= data sets.
- Observations with *a missing value for the key variable in the input data set* will not be processed. A warning message will be entered in the log file with a counter for the number of observations dropped.
- Observations with a valid value for the key variable in the input data set but with *missing values for at least one variable but not all variables specified on the MUSTIMPUTE statement* will not be processed. A warning message will be entered in the log file with a counter for the number of observations dropped.
- Donor observations with a valid value for the key variable and *valid values for all variables specified on the MUSTIMPUTE statement but with at least one missing value for MUSTMATCH variables*, if this last statement is specified, will not be processed. A donor must have all valid values for the "user matching fields". A warning message will be entered in the log file with a counter for the number of observations dropped.
- If the REJECTNEGATIVE option is in effect, donor observations with a valid value for the key variable in the input data set but with *negative values for one or more variables specified on the MUSTIMPUTE or MUSTMATCH statements* will not be processed. A warning message will be entered in the log file with a counter for the number of observations dropped.
- A variable cannot be specified in more than one ID, BY, MUSTIMPUTE and MUSTMATCH statement. These lists of variables are mutually exclusive.
- The ACCEPTNEGATIVE and REJECTNEGATIVE options are mutually exclusive. Specifying both will cause the procedure to stop.
- If the NLIMIT or the MRL option is in effect, details will be entered in the log file with regards to the ratio of donors that have reached DONORLIMIT_j for each group.
- If the NLIMIT or MRL option is in effect, DONORLIMIT data will be added to the DONORMAP= dataset. When these parameters are omitted, the DONORLIMIT variable will remain empty in the DONORMAP= dataset.
- When limiting donors with the NLIMIT option, the number of remaining donors may end up being less than MINDONORS. In such a case, the procedure will continue and ignore MINDONORS which was validated at the beginning. The same applies for PCENTDONORS.

Example

```
data massimpdata;
infile cards;
input IDENT $ TOTAL Q1 Q2 Q3 Q4 STAFF;
cards;
REC01 500 100 125 125 150 1000
REC02 750 200 170 130 250 2000
```

```
REC03 400 80 90 100 130 1000
REC04 1000 150 250 350 250 2000
REC05 3000 500 500 1000 1000 1000
REC06 800 200 225 200 175 2000
REC07 600 125 150 175 150 1000
REC08 900 175 200 225 300 2000
REC09 2500 500 650 600 750 1000
REC10 800 150 175 225 250 2000
REC21 3000 -45 -50 75 -234 2000
REC11 575 . . . . 1000
REC12 850 . . . . 2000
REC13 375 . . . . 1000
REC14 1100 . . . . 2000
REC15 3100 . . . . 1000
REC16 750 . . . . 2000
REC17 675 . . . . 1000
REC18 875 . . . . 2000
REC19 4000 . . . . 1000
REC20 . . . . . 2000
;
/* sort the input data set by the STAFF and IDENT variables.*/

proc sort data=massimpdata;
by STAFF IDENT;
run;

proc massimputation
data=massimpdata
out=outdata
donormap=donormap
mindonors=1
pcentdonors=1
random
acceptnegative
;
id IDENT;
mustimpute Q1 Q2 Q3 Q4;
mustmatch TOTAL;
by STAFF;
run;
```

Note

This document is a guide for the use of the procedure PROC MASSIMPUTATION. For more information on the methodology, please see the *Banff Functional Description* document.

Writing Linear Edits

Rules

The rules for writing edits are as follows:

editlist:
edit
edit editlist

edit:
componentlist operator componentlist ;
modifier : componentlist operator componentlist ;

componentlist:
component
component + componentlist
component - componentlist

component:
- component
number
number * variable
variable
variable * number

modifier:
one of ACCEPTE | FAIL | PASS | REJET

operator:
one of > | >= | = | != | <= | <

variable:
letter followed by list of letters | digits | underscore

From these rules we can say that:

- An editlist is made up of one or more edits.
- An edit starts optionally with a modifier followed by a colon (:), followed by a componentlist, followed by an operator, followed by a componentlist and terminated by a semi-colon (;).
- A componentlist is made up of or one more components. The components are separated by a plus (+) or a minus (-) sign.
- A component starts optionally with a minus (-) sign followed by a number and/or a variable (they can be in any order). When a component has a number and a variable they must be separated by an asterisk (*).
- If not specified, the default modifier value is PASS.

Examples of editlist:

EDITLIST	NOTE

x1 + x2 = 3000 ;	Spaces can be inserted as desired
x1>=5;	Spaces can be eliminated
x1 +x2 + x3 = 1500;	
100 <= x3 - 3.5 * x4;	Constant can appear on left side of equation
FAIL: x1 != 400;	Inequality is permitted, but only with a FAIL modifier
PASS: x1 != 400;	Invalid editlist
PASS: x2 = 500;	Equality is permitted, but only with a PASS modifier
FAIL: x2 = 500;	Invalid editlist
Pass: x2 > 37;	Modifier can be specified in lower case

Editlist

An editlist can contain any number of edits.

Edit

An edit must be a linear equation. (e.g. no exponents to variables and no division operator)

Modifier

Valid values are ACCEPTE, FAIL, PASS and REJET. If no value is specified, the modifier will default to PASS.

Variable

The variable must be a valid variable name found on the input data set. In Banff an underscore (_) is not valid as the first character of a variable name.

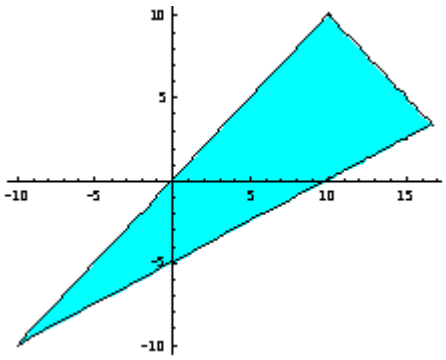
Note

If a coefficient in the EDITS is very close to zero, (for example in Windows, if the value is smaller than 1.19209290e-07) then the coefficient will be set to zero.

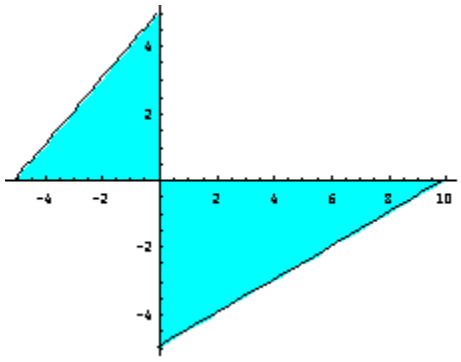
Specifying Edits for Processing Negative Numbers with Banff

Specifying edits for processing negative numbers in Banff is in most cases as straightforward as specifying edits for processing nonnegative numbers. However, the case where we require one variable to be within a certain percentage of another variable can present some difficulties. This document concentrates on treating this particular case.

Banff uses edit rules which are linear equalities or inequalities. A feasible region defined by such linear edits is convex. A feasible region in $\mathbf{R}^n$ is said to be convex if for all points A and B in the region and all t in [0,1], the point (1-t)A + tB is also in the region. In other words, a feasible region is convex if for all points A and B, the line segment connecting A and B is inside the region.



Convex Feasible Region

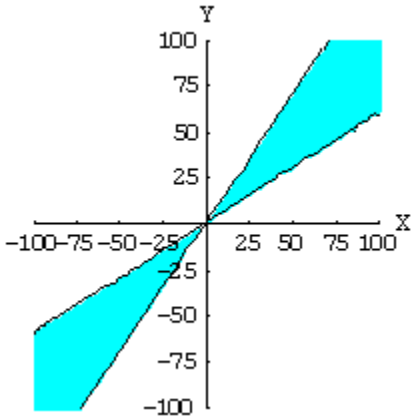


Non-Convex Feasible Region

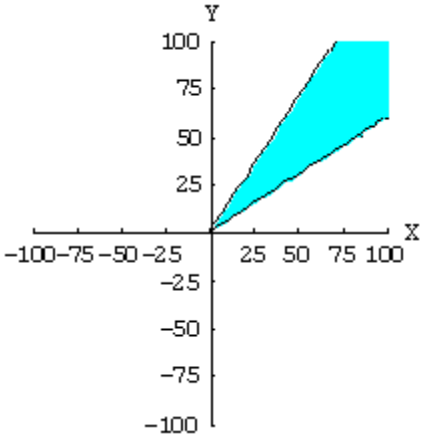
Problem 1: Specify edits for the condition "Variable Y is within 40% of variable X"

When negative values are considered valid values, this condition corresponds to the inequalities 0.6X<=Y<=1.4X, if X and Y are positive and 0.6X>=Y>=1.4X, if X and Y are negative, which define the first feasible region below. Clearly, this feasible region is

non-convex; therefore, it cannot be specified using linear edits. Note that the linear edits "0.6*X<=Y; Y<=1.4*X;" correspond to the second feasible region below. In this feasible region, all negative values are invalid – i.e. an observation such as (X,Y)=(-50,-50) will be deemed invalid by PROC ERRORLOC and identified for imputation and will in turn be imputed with positive values by PROC DONORIMPUTATON.

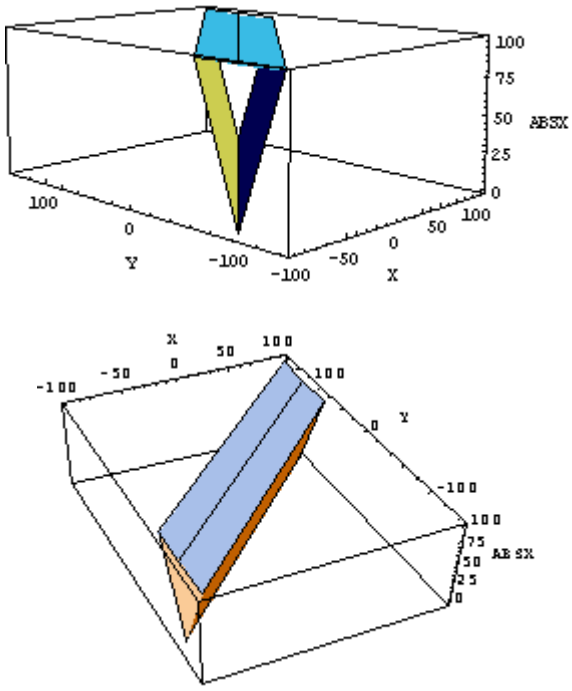


Feasible region defined by
0.6X<=Y<=1.4X, if X and Y are positive
0.6X>=Y>=1.4X, if X and Y are negative



Feasible region defined by
0.6X<=Y<=1.4X

Solution: One possible way to specify edits for the condition "Variable Y is within 40% of variable X" is to add a third variable, ABSX, to the dataset and define it as the absolute value of X. The linear edits can then be defined as "X-0.4*ABSX<=Y; Y<=X+0.4*ABSX; ABSX>=0;" The corresponding feasible region is convex and shown below.



Feasible region defined by X-0.4ABSX<=Y<=X+0.4ABSX

PROC ERRORLOC: In PROC ERRORLOC we are interested in identifying invalid values of X and Y, not ABSX. To ensure that only X and Y can be identified as invalid by PROC ERRORLOC, a large weight should be assigned to ABSX.

After running PROC ERRORLOC, update your output data set to include an observation for ABSX with status "FTI" for every record where X was flagged for imputation. This is necessary because in records where X is considered invalid, the value of ABSX should be treated as invalid by imputation procedures later on.

PROC DETERMINISTIC: If you are replacing your original edits "X-0.4*ABSX<=Y; Y<=X+0.4*ABSX; ABSX>=0;" with the equality edit "Y=X;" in PROC DETERMINISTIC, or if there are no records in which X requires imputation, the variable ABSX does not require any special treatment. After running PROC DETERMINISTIC, update your data and status datasets as usual. Then recompute ABSX and change its status to "IDE" in the status dataset for all records where X was imputed by PROC DETERMINISTIC to indicate that ABSX is now imputed and valid. Skip the next paragraph and go on to PROC DONORIMPUTATION.

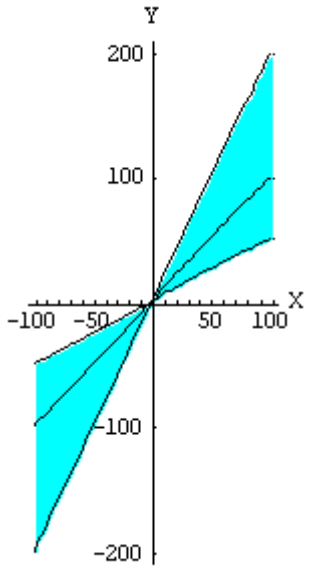
If you are using the edits "X-0.4*ABSX<=Y; Y<=X+0.4*ABSX; ABSX>=0;" and there are records in which X was imputed by PROC DETERMINISTIC, it is possible that the imputed value for X is too large or too small. This is because the relationship between X and ABSX is not specified by the edits and ABSX is unbounded. Consequently, in a record with (X,Y,ABSX)=(. ,50, .), 100 would be a valid value for ABSX, so imputing the value 30 to X does not contradict the edits (30-0.4*100<=50<=30+0.4*100) even though it does not satisfy our original condition that Y is within 40% of X. To ensure

that such values are detected, create a new dataset containing only the records where X was imputed by PROC DETERMINISTIC and calculate ABSX based on those values. **Rerun PROC ERRORLOC** with the edits "X-0.4*ABSX<=Y; Y<=X+0.4*ABSX; ABSX>=0;" and with large weights for ABSX and Y. In records where X was imputed values that were too large or too small by PROC DETERMINISTIC, X will be flagged for imputation by PROC ERRORLOC. In the above example, the corresponding record in the new dataset is (X,Y,ABSX)=(30,50,30) and it clearly fails the edits. Because Y and ABSX have large weights, PROC ERRORLOC will flag X for imputation. Finally, in all records where X is flagged for imputation by this run of PROC ERRORLOC, change the status of X from "IDE" back to "FTI" in the status dataset produced by PROC DETERMINISTIC. Update your original datasets as usual, recompute ABSX and change its status to "IDE" in the original status dataset for all records where X was imputed by PROC DETERMINISTIC to indicate that ABSX is now imputed and valid.

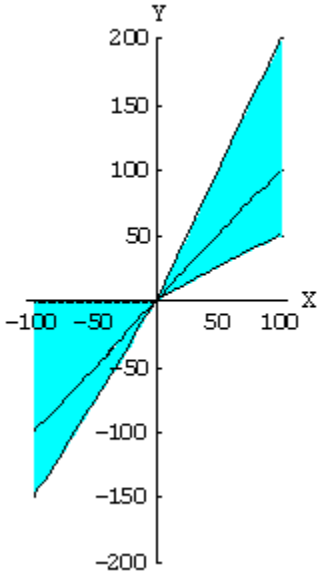
PROC DONORIMPUTATION: In PROC DONORIMPUTATION the linear edits "X-0.4*ABSX<=Y; Y<=X+0.4*ABSX; ABSX>=0;" can be used both as edits and post-edits. After running PROC DONORIMPUTATION, update your datasets as usual. Recomputing ABSX is not necessary because whenever X is imputed, the corresponding correct value for ABSX will be imputed using the same donor with valid values for X and ABSX.

Problem 2: Specify edits for the condition "Variable Y is no smaller in magnitude than 50% of variable X and no larger in magnitude than 200% of variable X".

This condition corresponds to the inequalities 0.5X<=Y<=2X, if X and Y are positive and 2X<=Y<=0.5X, if X and Y are negative. The corresponding feasible region is shown below. The line X=Y is also shown to make it easier to see that Y is allowed to be anywhere between twice as large in magnitude as X and half as large in magnitude as X.



Feasible region defined by
0.5X<=Y<=2X, if X and Y are positive
2X<=Y<=0.5X, if X and Y are negative



Feasible region defined by
X-0.5ABSX <= Y <= X+ABSX

This problem is similar to Problem 1, so the same general approach can be taken to solve it (please read the solution for Problem 1). Note that using the exact same solution as the solution for Problem 1, will produce the second feasible region shown above, which is not the region we need. Because the feasible region for this problem is not symmetric around the line X=Y as in Problem 1, this problem requires the addition of two variables to the dataset instead of one. Create and define the two variables UB and LB as

$$UB = \begin{cases} X, & \text{if } X \geq 0 \\ -0.5X, & \text{if } X < 0 \end{cases} \qquad LB = \begin{cases} 0.5X, & \text{if } X \geq 0 \\ -X, & \text{if } X < 0 \end{cases}$$

The linear edits for this problem are then "X-LB<=Y; Y<=X+UB; LB>=0; UB>=0;". These edits can be used in procedures ERRORLOC, DETERMINISTIC and DONORIMPUTATION as described in the solution for Problem 1 by replacing ABSX in that solution by LB and UB.

Problem 3: Specify edits for the condition "Variable Y is within 40% plus a constant of variable X"

Solution: This condition corresponds to the inequalities 0.6X–const<=Y<=1.4X+const, if X and Y are positive and 0.6X+const>=Y>=1.4X–const, if X and Y are negative. Note that const is assumed to be a positive constant. The feasible region for this problem is symmetric around the line X=Y, so the solution is as in Problem 1. Create the variable ABSX as in Problem 1 and use the linear edits "X–0.4*ABSX–const<=Y; Y<=X+0.4*ABSX+const; ABSX >= 0;" These edits can be used in procedures ERRORLOC, DETERMINISTIC and DONORIMPUTATION as described in the solution for Problem 1.

Problem 4: Specify edits for the condition "Variable Y is no smaller in magnitude than 60% of variable X less a constant and no larger in magnitude than 140% of variable X plus another constant"

Solution: This condition corresponds to the inequalities $0.6X - const_1 <= Y <= 1.4X + const_2$, if X and Y are positive and $0.6X + const_1 >= Y >= 1.4X - const_2$, if X and Y are negative. Note that both constants are assumed to be positive. If $const_1 \neq const_2$, then the feasible region for this problem is not symmetric around the line X=Y and so, the solution to this problem is similar to the solution for Problem 2. Create and define the two variables UB and LB as

$$UB = \begin{cases} 0.4X + const_2, & \text{if } X \geq 0 \\ -0.4X + const_1, & \text{if } X < 0 \end{cases} \qquad LB = \begin{cases} 0.4X + const_1, & \text{if } X \geq 0 \\ -0.4X + const_2, & \text{if } X < 0 \end{cases}$$

The linear edits for this problem are then "X-LB<=Y; Y<=X+UB; LB>=0; UB>=0;" These edits can be used in the procedures ERRORLOC, DETERMINISTIC and DONORIMPUTATON as described in the solution for Problem 1 by replacing ABSX in that solution by LB and UB.

Appendix

Problem 1: An example

Suppose our dataset is as shown below. We want the observations in it to satisfy the condition "Y is within 40% of X". We will first use PROC ERRORLOC to identify fields requiring imputation and then PROC DETERMINISTIC and PROC DONORIMPUTATION to impute values to these fields. Note that we will use the variable Z only in PROC DETERMINISTIC for the purpose of generating a deterministic solution for X. Here the variable Z simply represents any other variables that may be present in your dataset.

PROC ERRORLOC

First, compute the variable ABSX as the absolute value of X.

```
data indata;
input ID $ X Y Z;
cards;
R1 100 50 80
R2 100 50 30
R3 -100 -50 -90
R4 -100 -50 -40
R5 80 40 80
;
run;
data indata;
set indata;
ABSX=abs(X);
run;
```

Use the resulting dataset (shown below) as the input dataset to PROC ERRORLOC. Note the large weight assigned to ABSX in the PROC ERRORLOC options.

Obs	ID	X	Y	Z	ABSX
1	R1	100	50	80	100
2	R2	100	50	30	100
3	R3	-100	-50	-90	100
4	R4	-100	-50	-40	100
5	R5	80	40	80	80

```
PROC ERRORLOC
data=indata
outstatus=outstatus_errorloc1
outreject=outreject_errorloc1
edits="X-0.4*ABSX<=Y; Y<=X+0.4*ABSX; ABSX>=0; X>=-99999; Y>=-99999;"
weights="ABSX=999;"
acceptnegative
seed=9
;
id ID;
run;
```

The status dataset produced by PROC ERRORLOC is as follows:

Obs	ID	FIELDID	STATUS
1	R1	X	FTI
2	R2	X	FTI
3	R3	X	FTI
4	R4	X	FTI
5	R5	Y	FTI

Update this status dataset to include an observation for ABSX with status "FTI" whenever X is flagged for imputation by proc Errorloc.

```
data updateABSX;
set outstatus_errorloc1;
if fieldid="X";
fieldid="ABSX";
run;
data instatus_deterministic;
set outstatus_errorloc1 updateABSX;
by ID;
run;
proc sort;
by ID fieldid;
run;
```

The resulting status dataset is as follows:

Obs	ID	FIELDID	STATUS
1	R1	ABSX	FTI
2	R1	X	FTI
3	R2	ABSX	FTI
4	R2	X	FTI
5	R3	ABSX	FTI
6	R3	X	FTI
7	R4	ABSX	FTI
8	R4	X	FTI
9	R5	Y	FTI

PROC DETERMINISTIC

Run PROC DETERMINISTIC with the updated data and status datasets. Here we include the edit X=Z only to produce a deterministic solution for X for illustrative purposes. The variable Z represents any set variables in your dataset which in combination with the remaining edits in your edit set may determine that there is only a single value for X which will satisfy the edit set.

```
PROC DETERMINISTIC
data=indata
instatus=instatus_deterministic
out=outdata_deterministic
outstatus=outstatus_deterministic
edits="X-0.4*ABSX<=Y; Y<=X+0.4*ABSX; X=Z; ABSX>=0; X>=-99999; Y>=-
99999;"
acceptnegative
;
id ID;
run;
```

The data and status datasets produced by PROC DETERMINISTIC are shown as follows:

Obs	ID	Z	Y	X	ABSX
1	R1	80	50	80	100
2	R2	30	50	30	100
3	R3	-90	-50	-90	100
4	R4	-40	-50	-40	100

Obs	ID	FIELDID	STATUS
1	R1	X	IDE
2	R2	X	IDE
3	R3	X	IDE
4	R4	X	IDE

After running PROC DETERMINISTIC, update the original data and status datasets as usual and recompute ABSX as the absolute value of X.

```
data indata;
update indata outdata_deterministic;
by ID;
ABSX=abs(X);
run;
data instatus_donorimputation;
update instatus_deterministic outstatus_deterministic;
by ID fieldid;
run;
```

The updated data and status datasets are as follows:

Obs	ID	X	Y	Z	ABSX
1	R1	80	50	80	80

2	R2	30	50	30	30
3	R3	-90	-50	-90	90
4	R4	-40	-50	-40	40
5	R5	80	40	80	80

Obs	ID	FIELDID	STATUS
1	R1	ABSX	FTI
2	R1	X	IDE
3	R2	ABSX	FTI
4	R2	X	IDE
5	R3	ABSX	FTI
6	R3	X	IDE
7	R4	ABSX	FTI
8	R4	X	IDE
9	R5	Y	FTI

If we had replaced the edits "X-0.4*ABSX<=Y; Y<=X+0.4*ABSX; ABSX>=0;" by the edit "X=Y" in this run of PROC DETERMINISTIC or if there were no values imputed to X by PROC DETERMINISTIC, we would set the status of ABSX to "IDE" in the above status dataset for all observations where X was imputed and we would move on to PROC DONORIMPUTATON. But for illustrative purposes, we used the original edits and ensured that X was imputed by PROC DETERMINISTIC, so we continue by checking whether the imputed values of X satisfy our original edits.

Note that the values imputed to X in observations R1 and R4 satisfy the condition "Y is within 40% of X" but because the relationship between X and ABSX is not specified by the edits, the values imputed to X in observations R2 and R3 do not satisfy this condition. To check whether the imputed values of X satisfy our original edits, create a dataset which contains the subset of observations in which X was imputed a value by PROC DETERMINISTIC.

```
proc sql;
create table test
as select * from indata where id in
(select id from instatus_donorimputation where fieldid="X" and
status="IDE");
quit;
```

The resulting dataset contains observations R1 to R4.

Obs	ID	X	Y	Z	ABSX
1	R1	80	50	80	80
2	R2	30	50	30	30
3	R3	-90	-50	-90	90
4	R4	-40	-50	-40	40

Run PROC ERRORLOC on this dataset with the original edits. Note the large weights assigned to ABSX and Y.

```
PROC ERRORLOC
data=test
```

```
outstatus=outstatus_errorloc2
outreject=outreject_errorloc2
edits="X-0.4*ABSX<=Y; Y<=X+0.4*ABSX; ABSX>=0; X>=-99999; Y>=-99999;"
weights="ABSX=999; Y=999;"
acceptnegative
seed=9
;
id ID;
run;
```

The status dataset produced by PROC ERRORLOC indicates that the values imputed to X in observations R2 and R3 by PROC DETERMINISTIC do not satisfy the condition "Y is within 40% of X" as expected.

Obs	ID	FIELDID	STATUS
1	R2	X	FTI
2	R3	X	FTI

Update the original status dataset by setting the status of these fields back to "FTI". Then, for observations where X was imputed an acceptable value by PROC DETERMINISTIC, set the status of ABSX to "IDE" to indicate that ABSX has been successfully imputed along with X.

```
proc sql;
update instatus_donorimputation
set status="FTI"
where fieldid="X" and status="IDE" and id in
(select id from outstatus_errorloc2 where fieldid="X" and
status="FTI");
quit;
data updateABSXstat;
set instatus_donorimputation;
if fieldid="X" and status="IDE";
fieldid="ABSX";
run;
data instatus_donorimputation;
update instatus_donorimputation updateABSXstat;
by ID fieldid;
run;
```

The resulting status dataset is as follows:

Obs	ID	FIELDID	STATUS
1	R1	ABSX	IDE
2	R1	X	IDE
3	R2	ABSX	FTI
4	R2	X	FTI
5	R3	ABSX	FTI
6	R3	X	FTI
7	R4	ABSX	IDE
8	R4	X	IDE
9	R5	Y	FTI

PROC DONORIMPUTATION

Run PROC DONORIMPUTATION with the latest data and status datasets and the original edits.

```
PROC DONORIMPUTATION
data=indata
instatus=instatus_donorimputation
out=outdata_donorimputation
outstatus=outstatus_donorimputation
donormap=map
edits="X-0.4*ABSX<=Y; Y<=X+0.4*ABSX; ABSX>=0; X>=-99999; Y>=-99999;"
postedits="X-0.4*ABSX<=Y; Y<=X+0.4*ABSX; ABSX>=0; X>=-99999; Y>=-
99999;"
mindonors=1
pcentdonors=1
n=1
acceptnegative
;
id ID;
run;
```

The resulting data, status, and donor map datasets are as follows:

Obs	ID	ABSX	X	Y
1	R2	80	80	50
2	R3	40	-40	-50
3	R5	80	80	50

Obs	ID	FIELDID	STATUS
1	R2	ABSX	IDN
2	R2	X	IDN
3	R3	ABSX	IDN
4	R3	X	IDN
5	R5	Y	IDN

Obs	RECIPIENT	DONOR	NUMBER_OF_ATTEMPTS
1	R2	R1	1
2	R3	R4	1
3	R5	R1	1

Update the original data and status datasets as usual.

```
data indata;
update indata outdata_donorimputation;
by ID;
run;
proc print; run;
data instatus;
update instatus_donorimputation outstatus_donorimputation;
by ID fieldid;
run;
```

The final data and status datasets are as follows:

Obs	ID	X	Y	Z	ABSX
1	R1	80	50	80	80
2	R2	80	50	30	80
3	R3	-40	-50	-90	40
4	R4	-40	-50	-40	40
5	R5	80	50	80	80

Obs	ID	FIELDID	STATUS
1	R1	ABSX	IDE
2	R1	X	IDE
3	R2	ABSX	IDN
4	R2	X	IDN
5	R3	ABSX	IDN
6	R3	X	IDN
7	R4	ABSX	IDE
8	R4	X	IDE
9	R5	Y	IDN

Defining EF algorithms

An estimator function is a mathematical expression involving constants, current and/or historical values of some variables of the record, and current and/or historical averages of some variables, those averages being calculated from acceptable records (roughly speaking, an acceptable record is a record such that all variables for which an average must be calculated in the algorithm are available). Arithmetic operators include the addition (+), the subtraction (-), the multiplication (*), the division (/), the exponentiation (^) and the unary negation (-). Parenthesis can be used for changing the order of execution; otherwise the order is determined by standard rules: the unary negation has the highest priority, followed by the exponentiation, followed by the multiplication and division, followed by the addition and subtraction.

To define an Estimator Function (EF) algorithm formula, the user fills the variable FORMULA of the ALGORITHM= data set and the variables FIELDID and optionally AUXVARIABLES of the ESTIMATOR= data set. Spaces can be used everywhere in an expression except for breaking the name of a placeholder. No distinction is made between upper and lower case.

	Data sets		
	ALGORITHM=	**ESTIMATOR=**	
Example	**FORMULA**	**FIELDID**	**AUXVARIABLES**
y = 1	1	y	
$y = x^2$	aux1 ^ 2	y	x
y = curr average of y	fieldid (a)	y	
a = curr value of x1 + hist value of x2	aux1 (c, v) + aux2 (h, v)	a	x1, x2
b = (hist value of y + curr average of y) / 2	(aux1 (h) + aux1 (a)) / 2	b	y

In its general form, a placeholder is used with the one of the formats:
AUX1 (*period*, *aggregation*),
AUX1 (*aggregation*, *period*).

where *period* is:
C for current data set,
H for the historical data set.

and *aggregation* is
V for using the variable's value of the observation,
A using the average of the variable based on acceptable observations.

The default *period* is current and the default *aggregation* is the value. For example:
AUX1, AUX1 (C), AUX1 (V) and AUX1 (C, V) are equivalent to each other.
AUX1 (H) and AUX1 (v, H) are equivalent to each other.
AUX1 (A) and AUX1 (C, a) are equivalent to each other.

The time period and aggregation options are called placeholder attributes.

Exponents can be applied to constants, placeholders and expressions.

The exponent for a constant simply follows the constant:
3^4

The exponent for a placeholder follows the closing parenthesis of the placeholder attributes:
AUX1^3, AUX1 (C)^3, AUX1 (V)^3 and AUX1 (C, V)^3 each mean the same thing.

The exponent for an expression requires parenthesis enclosing the expression:
(AUX1 (H, V) + AUX2 (H, V))^2

In all cases, the exponent must be a constant (a number). A placeholder cannot be used as an exponent. Similarly, an expression cannot be used as an exponent, even if it is a constant expression. Thus x^y and x^(2+1) are not allowed.

Only placeholders FIELDID and AUXn are acceptable in the FORMULA variable. Placeholders are AUX1, AUX2, AUX3, and so on, starting with 1. We must use AUXn-1 in the algorithm if we use AUXn. The special placeholder FIELDID is a placeholder for the name of the variable to impute and thus this one won't need to be passed through the AUXVARIABLES variable of the ESTIMATOR data set. FIELDID (C, V) must never be used, it has to be imputed.

Variable names specified in the AUXVARIABLES variable are positional. The first one replaces placeholder AUX1, the second, AUX2, and so on. The number of variables in AUXVARIABLES must be equal to the number of different AUXn placeholders.

Any constants are numbers different from 0.

Defining LR algorithms

Regression imputation consists of imputing a variable y_{iC} by a linear regression like

$y_{iC} = B_0 + B_1 * X1_{iT1}^{P1} + B_2 * X2_{iT2}^{P2} + ... + B_m * Xm_{iTm}^{Pm}$

where Xm are m variables (called independent variables or regressors) specified by the user, the index i refers to the number of the record, the subscript C refers to the current data and the subscript Tj is either current or historical data. The superscripts P1, P2, …, Pm are numbers different from zero used as exponents. The Pj can be non-integer if Xj_{iTj} is positive. The B_0, B_1, ..., B_m are values not specified by the user but instead are calculated from the acceptable records. Note that B_0, which is the intercept in the regression line, is optional and can be omitted from the model.

To define a Linear Regression (LR) algorithm, the user fills the variable FORMULA of the ALGORITHM= data set and the variables FIELDID and optionally AUXVARIABLES of the ESTIMATOR= data set. Spaces can be used everywhere in an expression except for breaking the name of a placeholder. No distinction is made between upper and lower case.

	Data sets		
	ALGORITHM=	**ESTIMATOR=**	
Example	FORMULA	FIELDID	AUXVARIABLES
$y = B_1 * xC$	aux1	y	x
$t = B_0 + B_1 * x_H$	intercept, aux1(h)	t	x
$y = B_1 * a + B_2 * b + B_3 * c$	aux1, aux2, aux3	y	a, b, c
$q = B_1 * x + B_2 * x^2 + B_3 * x^3$	aux1, aux1^2, aux1^3	q	x

In its general form, a placeholder is used with the format:
AUX1 (*period*)

where *period* is:
C for current data set,
H for the historical data set.

The default *period* is current. Thus AUX1 and AUX1 (C) are equivalent to each other.

The time period option is called placeholder attribute.

The exponent for a placeholder follows the closing parenthesis of the placeholder attribute:
AUX1^3 and AUX1 (C)^3 each mean the same thing.

In all cases, the exponent must be a constant (a number). A placeholder cannot be used as an exponent. Similarly, an expression cannot be used as an exponent, even if it is a constant expression. Thus x^y and x^(2+1) are not allowed.

Only placeholders FIELDID and AUXn are acceptable in the FORMULA variable. Placeholders are AUX1, AUX2, AUX3, and so on, starting with 1. We must use AUXn-1 in the algorithm if we use AUXn. The special placeholder FIELDID is a placeholder for the name of the variable to impute and thus this one won't need to

be passed through the AUXVARIABLES variable of the ESTIMATOR data set. FIELDID (C) must never be used, it has to be imputed, the user must always write FIELDID (H).

A regressor cannot be repeated. The intercept and a variable with the same period and exponent should be specified once and once only. Commas delimit the different regressors. There is no order for specifying those elements except that auxiliary variables passed with placeholders should match the order in the AUXVARIABLES variable of the ESTIMATOR= data set.

Variable names specified in the AUXVARIABLES variable are positional. The first one replaces placeholder AUX1, the second, AUX2, and so on. The number of variables in AUXVARIABLES must be equal to the number of different AUXn placeholders.

Any constants are numbers different from 0.

Pre-defined algorithms tables

The **first line** gives the *name*, the *type* and the *status code* of the pre-defined algorithm.
The **second line** describes the *purpose* of the algorithm.
The **last one** shows the *formula*.

Estimator functions

AUXTREND : EF : IAT The value from the previous survey for the same unit, with a trend adjustment calculated from an auxiliary variable, is imputed. fieldid(h,v) * aux1(c,v) / aux1(h,v)
AUXTREND2 : EF : IAT2 An average of two AUXTRENDs is imputed. fieldid(h,v) / 2 * (aux1(c,v) / aux1(h,v) + aux2(c,v) / aux2(h,v))
CURAUX : EF : ICA The current value of a proxy variable for the same unit is imputed. aux1(c,v)
CURAUXMEAN : EF : ICAM The current average of a proxy variable is imputed. aux1(c,a)
CURMEAN : EF : ICM The mean value of all (user-defined) respondents for the current survey is imputed. fieldid(c,a)
CURRATIO : EF : ICR A ratio estimate, using values of all (user-defined) respondents from the current survey is imputed. fieldid(c,a) * aux1(c,v) / aux1(c,a)
CURRATIO2 : EF : ICR2 An average of two CURRATIOs is imputed. fieldid(c,a) / 2 * (aux1(c,v) / aux1(c,a) + aux2(c,v) / aux2(c,a))
CURSUM2 : EF : ISM2 The sum of two auxiliary variables from the current data table. aux1 + aux2
CURSUM3 : EF : ISM3 The sum of three auxiliary variables from the current data table. aux1 + aux2 + aux3
CURSUM4 : EF : ISM4 The sum of four auxiliary variables from the current data table. aux1 + aux2 + aux3 + aux4
DIFTREND : EF : IDT 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. fieldid(c,a) * fieldid(h,v) / fieldid(h,a)
PREAUX : EF : IPA The historical value of a proxy variable for the same unit. aux1(h,v)

PREAUXMEAN : EF : IPAM
The historical average of a proxy variable for the same unit is imputed.
aux1(h,a)

PREMEAN : EF : IPM
The mean value from the previous survey of all (user-defined) respondents is imputed.
fieldid(h,a)

PREVALUE : EF : IPV
The value from the previous survey for the same unit is imputed.
fieldid(h,v)

Estimation by linear regressions

CURREG : LR : ILR1
A simple linear regression based on one independent variable from the current data table.
intercept, aux1(c)

CURREG_E2 : LR : ILRE
A regression based on the value and the squared value of a variable from the current data table.
intercept, aux1(c), aux1(c)^2

CURREG2 : LR : ILR2
A linear regression based on two independent variables from the current data table.
intercept, aux1(c), aux2(c)

CURREG3 : LR : ILR3
A linear regression based on three independent variables from the current data table.
intercept, aux1(c), aux2(c), aux3(c)

HISTREG : LR : IHLR
A linear regression based on the historical value of the variable to impute.
intercept, fieldid(h)

Writing Prorating Edits

Rules

The rules for writing the prorating edits are as follows:

editlist:
edit
edit editlist

edit:
componentlist = variable ;

componentlist:
component
component + componentlist

component:
variable
variable : modifier
weight variable
weight variable : modifier

modifier:
one of A | I | N | O

variable:
letter followed by list of letters | digits | underscore

From these rules we can say that:

- An editlist is made up of one or more edits.
- An edit is made up of a componentlist followed by an equal sign (=), followed by a variable and terminated by a semi-colon (;).
- A componentlist is made up of one or more components. The components are separated by a plus (+) sign.
- A component is a variable that is optionally preceded by a weight and optionally followed by a colon (:), followed by a modifier.

Examples of editlist:

Editlist	NOTE
Total1 +Total2+Total3 = GrandTotal ;	Spaces can be inserted as desired
0.1Total1Part1 +.5 Total1Part2 + 1.2 Total1Part3 + 10 Total1Part4 = Total1;	Decimal values can have leading zeroes. Number can be integers.
Total2Part1 : a + Total2Part2 : i + Total2Part3 : o + Total2Part4 : n = Total2;	Use of modifiers. Modifiers can be specified in lower and/or uppercase.
1Total3Part1:a+2Total3Part2:i+3Total3Part3:o+4Total3Part4:n=Total3;	Weights and modifiers are being specified.

AVariable + AnotherVariable = AnotherGrandTotal;	

Editlist

An editlist can contain one or more edits.

Within an editslist, edits do not have to be sorted nor ordered.

There is no restriction on the number of edits in an editlist.

Edit

An edit must be a sum of components that add up to a total. Parts of sums and subtotals can be nested to an unlimited degree in leading to an overall fixed total.

Modifier

The modifier lets the user choose when or how a variable is modifiable.

Possible values are: A, I, N or O.

A: Always.
I: Imputed.
N: Never.
O: Original.

Variable

The variable must be a valid variable name found on the input data set. In Banff an underscore (_) is not valid as the first character of a variable name.

A variable cannot be A, I, N or O.

Weight

A weight is a positive number.

Examples of valid numbers are: 1, 0.12, 5, 999999999, 2e1.

Definition of Status Codes used in Banff

FTI: Field To Impute. This code identifies a field that needs imputation. This status code is created either by the Outlier procedure or by the Errorloc procedure. When created by the Outlier procedure, the code includes ODIL and ODIR. When created by the Errorloc procedure, it means that the field value does not pass the linear equations (see the definition of OUTSTATUS= in the Errorloc procedure User Guide).

FTE: Field To Exclude. This code identifies a field which might be excluded. This status code is created by the Outlier procedure and includes ODEL and ODER. This status code is used to exclude automatically or by option some observations (See the section Details in the Donorimputation procedure User Guide and the definition of the data set ESTIMATOR= in the Estimator procedure User Guide).

IDE: Field that has been imputed using the Deterministic imputation method. All imputation procedures will consider a field flagged as IDE as if it has not been imputed (i.e. the value is considered to be an original or reported value).

IDN: Field that has been imputed using the Donorimputation imputation method.

IPR: Field that has been prorated using the Prorate procedure.

IAT: Variable that has been imputed using the AUXTREND algorithm of the ESTIMATOR procedure.

IAT2: Variable that has been imputed using the AUXTREND2 algorithm of the ESTIMATOR procedure.

ICA: Variable that has been imputed using the CURAUX algorithm of the ESTIMATOR procedure.

ICAM: Variable that has been imputed using the CURAUXMEAN algorithm of the ESTIMATOR procedure.

ICUM: Variable that has been imputed using the CURMEAN algorithm of the ESTIMATOR procedure.

ICR: Variable that has been imputed using the CURRATIO algorithm of the ESTIMATOR procedure.

ICR2: Variable that has been imputed using the CURRATIO2 algorithm of the ESTIMATOR procedure.

IDT: Variable that has been imputed using the DIFTREND algorithm of the ESTIMATOR procedure.

IHLR: Variable that has been imputed using the HISTREG algorithm of the ESTIMATOR procedure.

ILR1: Variable that has been imputed using the CURREG algorithm of the ESTIMATOR procedure.

ILR2: Variable that has been imputed using the CURREG2 algorithm of the ESTIMATOR procedure.

ILR3: Variable that has been imputed using the CURREG3 algorithm of the ESTIMATOR procedure.

ILRE: Variable that has been imputed using the CURREG_E2 algorithm of the ESTIMATOR procedure.

IPA: Variable that has been imputed using the PREAUX algorithm of the ESTIMATOR procedure.

IPAM: Variable that has been imputed using the PREAUXMEAN algorithm of the ESTIMATOR procedure.

IPM: Variable that has been imputed using the PREMEAN algorithm of the ESTIMATOR procedure.

IPV: Variable that has been imputed using the PREVALUE algorithm of the ESTIMATOR procedure.

ISM2: Variable that has been imputed using the CURSUM2 algorithm of the ESTIMATOR procedure.

ISM3: Variable that has been imputed using the CURSUM3 algorithm of the ESTIMATOR procedure.

ISM4: Variable that has been imputed using the CURSUM4 algorithm of the ESTIMATOR procedure.

MFS: Field flagged **by the system** as a matching field for a given recipient. This code is generated by the Donorimputation procedure.

MFU: Field flagged **by the user** (MUSTMATCH=) as a matching field **for all recipients**. This code is generated by the Donorimputation procedure.

MFB: Field flagged **by the system** as a matching field for a given recipient and **by the user** (MUSTMATCH=) as a matching field **for all recipients**. This code is generated by the Donorimputation procedure.

These status codes must be specified in CAPITAL letters in order to be recognised by Banff. They will be generated in the OUTSTATUS= data set if the MATCHFIELDSTAT option has been specified when calling the Donorimputation procedure.

ODER: Outlier field, with values falling outside the exclusion interval, on the right (see the definition of MEI= in the Outlier procedure User Guide).

ODEL: Outlier field, with values falling outside the exclusion interval, on the left (see the definition of MEI= in the Outlier procedure User Guide)

ODIR: Outlier field, with values falling outside the imputation interval, on the right (see the definition of MII= in the Outlier procedure User Guide)

ODIL: Outlier field, with values falling outside the imputation interval, on the left (see the definition of MII= in the Outlier procedure User Guide)