Euredit WP5.1 Internal Report No.1

DIS Software Documentation

Nargis Rahman - Office for National Statistics
Geoff Morgan - NAG
Date April 11, 2002

Contents

1 Description of donor imputation system.

The basic principle underlying the DIS is to search and use a single donor for all the missing variables of a recipient record. The method searches for a donor using a set of matching variables which are related to the missing variable(s) of the recipient record. The matching variables are used to calculate a statistical distance between recipient and donor records.

A donor is selected based on a statistical distance function. The donor is the one with minimum distance. If at the end of this stage a donor has not been found for a recipient, then the categories of each matching variable are collapsed and the search is repeated. If missing values are still present for recipient records then non-significant matching variables are removed in turn until only one matching variable remains.

There are two main stages in the implementation of DIS and these are:

searching and establishing a pool of suitable donors;

selection of the donor.

Several possibilities exist when more than one donor is available for a recipient. The simplest is to just use the first donor in the list, or one can randomly choose a donor from the available list. Multiple use of donors can be reduced by incorporating a penalty function for each use, see for example, (Yar, 1998).

To summarise, the donor imputation search algorithm is given by,

1. search for the donor using a set of matching variables;
2. search for the donor using the same set of matching variables but with collapsed categories of the variables;
3. remove non-significant matching variables one at a time and search for the donor as in steps 1-2.

As soon as a donor with minimum statistical distance has been found, the search process will be stopped. In the search algorithm, progression from a lower level to a higher level will take place only if a donor with minimum statistical distance has not been found.

2 Description of GeDaM

GeDaM is a general algorithm for finding the best match of a recipient record with another record in the data matrix. For each record in a search range GeDaM computes a score from a potential donor record to the recipient record as

$$\sum_{i=1}^{p} \omega_i d_i / \sum_{i=1}^{p} \omega_i$$

where d_i is a measure of the distance for the ith matching variable between the recipient record and the potential donor record. Each imputation variable has p matching variables. The value of p may be different for each imputation variable. Each matching variable also has a weight ω_i. Six types of distances are currently included, three for continuous variables and three for categorical variables. The program looks for the records that have the smallest score provided that the values of interest are present. The output is in the form of a list of imputed values which feeds into the ApplyEdits program to obtain a data set with imputed values.

3 Available options

3.1 Distances for continuous variables

For contiuous variables there is a choice of three distances. They are,

1. Euclidean distance;
2. Manhattan distance;
3. Regression distance (predictive mean matching).

In the following definitions y^r represents a matching variable from the recipient record and y^d represents a matching variable from a potential donor record. The Euclidean distance between y^r and y^d is given by $\mathrm{d} = (y^r - y^d)$ and the Manhattan distance is given by $\mathrm{d} = |y^r - y^d|$. The regression distance obtains predictions from the regression model built using the matching variables as covariates. At present only a linear model is available. Predictions are obtained for non-missing and missing variables. The prediction for each missing variable is compared with the predictions for the non-missing variables to find a match. The imputed value is then the true value from the matched record.

In each case the difference between the values (its absolute value for Manhattan distance) is multiplied by a weight and a scaling factor (or regression coefficient). An appropriate scaling factor is the reciprical of the range. For Euclidean distance the square root of the sum of squared values for all variables using this distance is added to the score. For the regression distance the absolute value of the sum of distances is added to the score.

3.2 Distances for categorical variables

For categorical variables there is a choice of three distances. They are,

1. Simple 0/1 matching;
2. Scaled rank difference;
3. User defined.

In the case of user defined distance for a variable with r categories, an r by r matrix containing the distances for any possible pair of categories has to be supplied.

4 Specifications

4.1 Specifying and ordering matching variables

For each imputation variable the weights ω_i can be used to select and give importance to variables when matching. The least significant variable will have the smallest weight. The program will try and find a donor using all specified matching variables, but if this is not possible then the variable with the smallest weight is dropped and the search for a donor is repeated. This procedure continues until one matching variable remains. A weight of zero means that that particular variable is not used in the matching process. If all of the weights are set to zero then no imputation is carried out for that imputation variable. A single donor to impute all imputation variables in a record can also be specified. This is achieved by defining a common set of matching variables for all imputation variables.

4.2 Merging categories of matching variables

An option is available to merge categories of categorical matching variables. This is done by selecting the user defined distance option for categorical variables. For a variable with r categories, the user needs to supply an r by r matrix containing the distances for any possible pair of categories. Ideally

the matrix should have a zero diagonal as a category should always match with itself. Categories that are to be merged should be assigned equal weights. The matrix should be set up so that the program searches for a donor using the unmerged categories first and if a match is not found then uses the merged categories.

Example

Suppose a variable has categories 1, 2, 3 and 4 and categories 1, 2 and 3 are to be merged. The following distance matrix could be used,

$$\text{distance matrix} = \begin{pmatrix} 0 & 1 & 1 & 4 \\ 2 & 0 & 2 & 4 \\ 3 & 3 & 0 & 4 \\ 4 & 4 & 4 & 0 \end{pmatrix}.$$

So for this variable, if a recipient record has value 1 the program will search for a donor with value 1. If one does not exist then a donor with values 2 or 3 is acceptable. That is, the program will merge categories 1, 2 and 3 if it is necessary to do so. This procedure is also applied to the other values.

4.3 Search options

Selection of imputation variables

In normal use the user can set the program to carry out imputation for all missing variables for all records. An option is also available that allows the user to specify specific records and variables for imputation.

Search range

An option is available for specifying the records to be searched when searching for a donor. Either all records in a given range are searched or the user can carry out regional matching where the program will only search records that have the same value of a categorical variable.

Matching options

The user can specify if a donor record should have non-missing values for all imputation variables. The alternative is that the program will sequentially search for a donor record for each imputation variable. If there is more than one donor record available, there are two options. The user can either select to use the first donor in the list or to randomly choose a donor from the list. The user can also specify the size of the list.

5 The options file

All specifications are declared in the options file. The structure of the options file is described below.

// **Options file**
// **Input File**
The data file
// **Number of Variables**
The number of variables in the data file
// **Maximum number of categories**
The maximum number of categories for any categorical variable
// **Input Delineator 0 = space, 1 = tab, 2 = comma**
How numbers are separated in the data file
// **Missing values**

Values to be treated as missing values for categorical and continuous variables. Note that in the case of tab or comma separator any blank field or non-numeric value is also considered to be missing.
// **Header: 1 = yes, 0 = no**
Whether or not the data file has a header row
// **Search range**
There are three items, the first indicates if are range is to be given (1) or regional matching (0). I range is selected the first and last records to be searched are given. If regional matching is selected the variable and the total number of observations are given.
// **Style type**
0 - indicates that the best matches are to be returned
1 - indicates that imputation results are to be returned using first occurrence of best match
2 - indicates that imputation results are to be returned using randomly selected record from best match.
(Note -1 lists all scores, used for debug purposes)
// **Maximum best matches**
The value of s.
// **Number of searches**
Either the value 0 to indicate all missing values in all records or the number of target records, followed by for each record
{
// **Observation**
The target record
// **Search**
0 - indicates all missing values
>0 then the following must be given
// **Variables**
List of variable
}
// **Match all**
The value 1 if a single matching record should be used for all missing variables, otherwise the value 0.
{
If Match all equals 1 then
// **Search Weights**
The values of the wi for all the variables.
}
// **Minimum weight**
The minimum acceptable value of the sum of weights for non-missing variables in a target record.
// **Variable Information**

The following information is required for each variable. For continuous variables

// **Variable number/name**
{
If Match all equals 0 then
// **Search Weights**
The values of the wi for the variable.
}
// **Distance type**
0 - Euclidean
1 - Manhattan
2 - Regression
// **Values**
0 - continuous
>0 number of categories

// **Scale**
The value of the scale factor or regression coefficient
// **End of variable**

For categorical variables

// **Variable number/name**
// **Search Weights**
The values of the wi for the variable.
// **Distance type**
1 - Simple matching
2 - Rank difference
3 - User defined
// **Values: 0 = continuous, >0 num of cat.**
Number of categories
// **Categories**
List of r categories.

If user defined distances are required then the following is needed.

// **User distances**
The r by r matrix with target categories as rows and search categories as columns.
// **End of variable**

// **End of file**

6 The ApplyEdits program

The ApplyEdits program takes the results from GeDaM and applies them to a data file. The results from GeDaM are given in the following format:

record number, variable number, old value, new value.

The ApplyEdits program allows the user to specify the output format of the edited data set. The ApplyEdits program is controlled by an options file. The format of the options file is given below.

// **Options file**
// **Input File**
The data file
// **Output File**
The edited data file
// **Edit File**
The file containing the changes, e.g. the output from GeDam.
// **Number of Observations**
The number of records in data file.
// **Number of Variables**
The number of variables in the data file.
// **Input Delineator 0 = space, 1 = tab, 2 = comma**
How numbers are separated in the data file
// **Header: 1 = yes, 0 = no**
Whether or not the data file has a header row
// **Method**
Indicates if values replace by new values (0) or old values (1)
{if Method = 1, old values, then

// **Missing values**
Values used as missing value indicators for categorical and continuous variables.
}
// **Output Delineator**
As for input delineator.
// **Output Missing value**
Gives missing value indicator used in the output file.
0 - blank,
1 - ".",
2 - "*",
Otherwise the number, e.g., -9
// **Width and dps**
The field width used in the output file and the number of decimal places for real numbers. If width is set to zero standard formatting is used.
// **End of file**

7 Outputs given by the programs

7.1 GeDaM outputs

The GeDaM program provides a list of all imputation results. It does not update the data file with imputation results, this is done via the ApplyEdits program. The form of the output is specified in the options file. The program will either return the s best matches, where s is specified by the user, for each imputation variable or the program will return the imputation results for each variable. Note, the first output method is for debug purposes. In order to update the data file with the imputation results the second output method must be used. With the second method the imputation results are provided in a list with the following format:

record number, variable number, old value, new value.

The file containing this list then feeds into the ApplyEdits program. Note, the inputs to GeDaM are the options file and the data file.

7.2 ApplyEdits output

This program produces a data file which is identical to the original data file but with the missing values replaced by the imputation results. The program requires the following inputs: options file, original data file, output from GeDaM.

8 How to run the programs

Both GeDaM and ApplyEdits are currently run through the dos command prompt:

GeDaM < gedamoptionsfile.txt > outputfile.txt

ApplyEdits < applyoptionsfile.txt.

Note, the name of the output file from the ApplyEdits program is specified in the options file.

References

Yar, M. (1998). The development of the donor imputation system (DIS). Technical report, Office for National Statistics.

A GeDaM example options file

```
// Options file
// Input File
c:\euredit\imputation\test1b.d
// Number of Variables
7
// Maximum number of categories
10
// Input Delineator 0 = space, 1 = tab, 2 = comma
1
// Missing values
-999 -999.0
// Header: 1 = yes, 0 = no
1
// Search range
1 1 15
//Style type
1
// Maximum best matches
10
// Number of searches
0
//Match all
0
// Variable Information
// Variable 1
//Search Weights
0.0 0.0 0.0 0.0 0.0 0.0 0.0
// Distance type
0
// Values: 0 = continuous, >0 num of cat.
0
// Scale
1.0
// End of variable
// Variable 2
//Search Weights
0.0 0.0 1.0 1.0 0.0 0.0 0.0
// Distance type
0
// Values: 0 = continuous, >0 num of cat.
2
//Categories
1 2
// End of variable
// Variable 3
//Search Weights
0.0 1.0 0.0 1.0 0.0 0.0 0.0
// Distance type
1
// Values: 0 = continuous, >0 num of cat.
3
//Categories
```

1 2 3
// End of variable
// Variable 4
//Search Weights
0.0 1.0 1.0 0.0 0.0 0.0 0.0
// Distance type
0
// Values: 0 = continuous, >0 num of cat.
0
// Scale
0.1
// End of variable
// Variable 5
//Search Weights
0.0 1.0 1.0 1.0 0.0 0.0 0.0
// Distance type
0
// Values: 0 = continuous, >0 num of cat.
0
// Scale
0.02
// End of variable
// Variable 6
//Search Weights
0.0 1.0 1.0 0.0 0.0 0.0 0.0
// Distance type
2
// Values: 0 = continuous, >0 num of cat.
4
//Categories
1 2 3 4
// User distances
0 1 1 5
2 0 1 4
3 1 0 3
10 1 1 0
// End of variable
// Variable 7
//Search Weights
0.0 0.0 0.0 0.0 0.0 0.0 0.0
// Distance type
0
// Values: 0 = continuous, >0 num of cat.
2
//Categories
1 2
// End of variable
// End of file

B ApplyEdits example options file

```
// Options file
// Input File
c:\euredit\imputation\test1b.d
// Output File
c:\euredit\imputation\test1b.out
// Edit File
c:\euredit\imputation\test1bedits.txt
// Number of Observations
15
// Number of Variables
7
// Input Delineator (0 = space, 1 = tab, 2 = comma)
1
//Header (1 = yes, 0 = no)
1
//Method (0 = new values, 1 = old values)
0
// Output Delineator
1
//Output Missing value (0 = blank, 1= ., 2 = *, else number)
1
//Width and dps
4 1
//End of file
```