WP. 39
ENGLISH ONLY

UNITED NATIONS STATISTICAL COMMISSION and ECONOMIC COMMISSION FOR EUROPE

CONFERENCE OF EUROPEAN STATISTICIANS

Work Session on Statistical Data Editing
(Ottawa, Canada, 16-18 May 2005)

Topic (iv): New and emerging methods, including automation through machine learning, imputation, evaluation of methods

SLICE 1.5: A SOFTWARE FRAMEWORK FOR AUTOMATIC EDIT AND IMPUTATION

Supporting Paper

Submitted by Statistics Netherlands[1]

ABSTRACT: SLICE is a software framework for automatic edit and imputation developed by Statistics Netherlands. This framework consists of several related modules. Recently version 1.5 of SLICE has been released for use at Statistics Netherlands. SLICE 1.5 can handle a mix of categorical, continuous, and integer-valued data. To localise and correct erroneous fields SLICE uses edit rules. These edit rules check whether the data of each individual respondent are consistent. The Blaise parser of SLICE 1.5 can rewrite edit rules in Blaise format to edit rules in SLICE format that can be processed by the other modules of SLICE 1.5. Based on the edit rules the *Cherry Pie* module can localise the erroneous fields. The fields localised as being erroneous are set to missing. These fields can, if desired, be imputed by the imputation module of SLICE 1.5, together with the missing values in the observed data. While imputing one generally does not take edit rules into account. After imputation edit rules may hence still be violated. The *AdaptValues* module of SLICE 1.5 can adjust the imputed values so that all edit rules become satisfied. In this paper we give an overview of the edit and imputation process by means of SLICE 1.5. Subsequently, we briefly discuss the Blaise parser, *Cherry Pie*, the imputation module, and the *AdaptValues* module.

KEYWORDS: automatic editing, categorical data, continuous data, edits, integer-valued data, imputation, numerical data

I. INTRODUCTION

1. SLICE is a software framework for automatic edit and imputation developed by Statistics Netherlands (De Waal, 2001a). Currently, SLICE 1.0 is used to process most of the structural annual business statistics at Statistics Netherlands (De Jong, 2002). The latest version is SLICE 1.5. In this paper we describe SLICE 1.5 in an easily accessible way. Technical aspects of the implemented algorithms are not discussed.

2. We note first that SLICE is a *software framework* consisting of several interrelated modules. SLICE 1.5 is not a computer program with a user interface. A user of SLICE has to develop a computer program with a user interface himself, and has to connect one or more modules of SLICE to this program. We have developed a software framework instead of a regular computer program to allow users to implement a (graphical) user interface according to their own wishes, and to let users decide for

[1] Prepared by Ton de Waal, (twal@cbs.nl).

themselves which modules of SLICE they want to use for their particular applications. A test program, SliceTest, has been developed that demonstrates the functionality of the SLICE framework.

3. SLICE 1.5 can process a mix of categorical, continuous, and integer-valued data. To localise and correct the errors in the data, SLICE uses edit rules, or *edits* for short. These edits check whether each record, *i.e.* the data on an individual respondent, is consistent or not. The edits check only the internal consistency of a record, not the consistency between different records. For instance, SLICE can check whether the profit and the costs of an enterprise sum up to its turnover, but not whether the turnovers of all records sum up to a value known from another data source.

4. We point out that automatic editing alone is generally not enough to obtain data of sufficient statistical quality. We feel, however, that the combined use of modern editing techniques leads to a reduction in processing time and a decrease in required resources in comparison to the traditional, interactive method, while preserving or often even increasing quality. At Statistics Netherlands we apply a combination of editing techniques to edit our structural annual business surveys. Using SLICE for automatic editing is only one step in a chain of related editing and imputation steps. Our approach consists of the following main steps (see De Jong, 2002; Van Velzen, 2005):

- application of selective editing to split the records in a critical stream and a non-critical stream (see Lawrence and McKenzie, 2000; Hedlin, 2003);
- editing of the data: the records in the critical stream are edited interactively, the record in the non-critical stream are edited and imputed automatically (using SLICE);
- validation of the publication figures by means of (graphical) macro-editing (Granquist, 1990).

During the selective editing step so-called plausibility indicators are used to split the records in a critical stream and a non-critical stream (see Hoogland, 2002 and 2005). Very suspicious or highly influential records are edited interactively. The remaining records are edited automatically, using SLICE. The final validation step consists of detecting the remaining outliers in the data, and comparing the publication figures based on the edited and imputed data to publication figures from a previous year. Influential errors that were not corrected during automatic (or interactive) editing can be detected during this final step.

5. The remainder of the paper is organised as follows. In Section II we describe the edit and imputation process when using SLICE. Section III discusses the data and edits that can be handled by modules of SLICE 1.5. Next we discuss various steps in the edit and imputation process for which SLICE 1.5 offers support. Section IV describes how erroneous fields are identified using the specified edits, Section V how missing values are imputed, and Section VI how imputed values are adjusted so that all edits become satisfied. Section VII ends the paper with a short discussion.

II. AUTOMATIC EDIT AND IMPUTATION BY MEANS OF SLICE

6. Figure 1 below sketches the usual steps of the automatic edit and imputation process when SLICE 1.5 is applied. In a first step SLICE 1.5 reads the specified edits. SLICE 1.5 can parse a large class of Blaise edits. The Blaise parser of SLICE 1.5 the Blaise edits into SLICE edits that SLICE 1.5 uses internally. Users of SLICE 1.5 may also specify their edits directly in the SLICE format. For most users it will, however, be easier to specify the edits in the Blaise format, and let the Blaise parser of SLICE 1.5 rewrite them into the SLICE format. The class of Blaise edits that can be handled by SLICE 1.5 is described in Section III. In order to rewrite Blaise edits into SLICE edits a complicated algorithm is used (Sluis and De Waal, 2004).

7. Once edits have been rewritten into the SLICE format, all records are read and processed separately. That is, after reading a record the erroneous fields are localised, better values are subsequently imputed, and finally the imputed values are adjusted so all edits become satisfied, before the next record is read and processed.

8. After reading a record the *Cherry Pie* module of SLICE 1.5 first checks by means of the specified edits whether the record contains internal inconsistencies. If this is the case, then *Cherry Pie* localises the fields that are the most likely ones to be erroneous. *Cherry Pie* localises these fields by assuming that the fewest possible (weighted) number of fields should be modified. In other words, *Cherry Pie* assumes that the (weighted) number of errors in the record is as small as possible. Here each field is assigned a weight: the so-called *reliability weight* of this field. This weight, which can be specified by the user, indicates how reliable the user considers the observed value of this field to be. A high weight indicates that the user considers the observed value to be reliable; a low weight indicates that the user considers the observed value to be not so reliable.

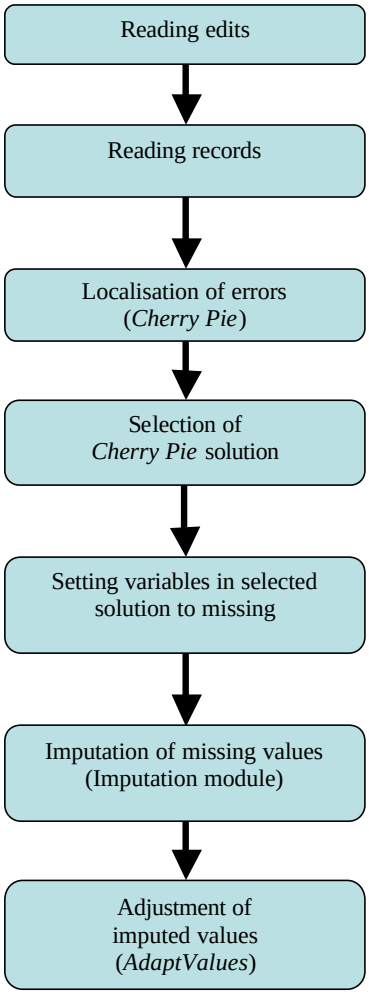

Figure 1. Overview of automatic edit and imputation by means of SLICE

9. Often there are several minimal sets of fields to "correct" a record. *Cherry Pie* determines all, up to a user-specified maximum number, these minimal sets of fields. These optimal ways to modify a record are the solutions to the so-called *error localisation problem*, *i.e.* the problem to localise the erroneous fields in a record. We also call these solutions the *Cherry Pie* solutions. In Section IV we describe the *Cherry Pie* module in some more detail.

10. From all *Cherry Pie* solutions that are generated one has to be selected. The fields involved in this solution are set to missing, and are later imputed. SLICE 1.5 offers little support to select one *Cherry Pie* solution from several *Cherry Pie* solutions, because selecting such a solution in a good way often requires specialist subject-matter knowledge. A very simple way to select a *Cherry Pie* solution that does not require any subject-matter knowledge at all is selecting the first *Cherry Pie* solution that is found. Another, more advanced, manner to select a *Cherry Pie* solution has been used in an evaluation study described by Pannekoek and De Waal (2005).

11. After the selection of a *Cherry Pie* solution the imputation step is carried out. In this step all missing values, both the values that were missing in the observed record as well as the values of the fields in the selected *Cherry Pie* solution that were set to missing, are estimated and filled in. To impute missing values the user can apply the imputation module of SLICE 1.5 or any other imputation package, such as the Missing Value Analysis module of SPSS (Hoogland and Pannekoek, 2000) and WAID (De Waal, 2001b). In Section V we describe the imputation module of SLICE 1.5.

12. It is very complex to take the edits into account while imputing missing values (Tempelman and Steerneman, 2004; Tempelman, 2005; Van den Eijkhof, De Waal and Pannekoek, 2005). Most imputation packages, including the imputation module of SLICE 1.5, do therefore not take the edits into account while imputing missing values. As a consequence, an imputed record may still fail the edits. In a last step the *AdaptValues* module of SLICE 1.5 therefore adjusts the imputed values as little as possible so that the final record does satisfy all edits. The not imputed values, *i.e.* the values that were not missing originally and that were also not identified as being erroneous, are not adjusted by this module. Section VI discusses the *AdaptValues* module of SLICE 1.5.

13. Above we have described the standard edit and imputation process when using SLICE 1.5. If desired, the user can deviate from this standard process. The steps in this process can namely also be carried out separately. For instance, one can apply only the *Cherry Pie* module to localise the erroneous fields in a data set. One can also apply only the imputation module to impute the missing values in a data set. Finally, one can apply only the *AdaptValues* module to adjust the imputed values in a data set so all edits become satisfied. These values may have been imputed by means of another software package.

14. In order to apply the *Cherry Pie* module or the *AdaptValues* module of SLICE 1.5, edits have to be specified. For the imputation module of SLICE 1.5 no edits have to be specified as edits are not taken into account during the imputation step. Conversely, in order to apply the imputation module of SLICE 1.5, imputation rules have to be specified. Such imputation rules do not have to be specified for the *Cherry Pie* module or the *AdaptValues* module of SLICE 1.5 as these modules do not use imputation rules.

III. DATA AND EDITS

15. SLICE 1.5 can handle a mix of categorical and numerical data. With respect to categorical data SLICE 1.5 does not distinguish between nominal and ordinal data. For each categorical variable v_i the user has to specify the domain D_i, *i.e.* the set of all possible values. If the observed value of a categorical variable is not an element of this domain, the observed value is considered erroneous by the *Cherry Pie* module.

16. With respect to numerical data SLICE 1.5 makes a distinction between continuous and integer-valued data. Integer-valued data are processed in a different manner by the *Cherry Pie* module and the *AdaptValues* module than continuous data (De Waal, 2005). A numerical variable may, unless the user has specified otherwise by means of the edits, attain a positive value, the value zero or a negative value. If the observed value of an integer-valued variable is not integer, the observed value is considered erroneous by the *Cherry Pie* module.

A. The edits of SLICE 1.5

17. The algorithms on which the *Cherry Pie* module and the *AdaptValues* module of SLICE 1.5 are based can directly handle edits of the following type

$$\text{IF } (v_1 \in F_1 \text{ AND } \ldots \text{ AND } v_m \in F_m) \text{ THEN } \textit{numerical condition}, \qquad (1)$$

where m denotes the number of categorical variables, the v_i (i=1,…,m) are the categorical variables, and the F_i (i=1,…,m) are subsets of the domains D_i (i=1,…,m) of the categorical variables. The numerical conditions are of the following type:

$a_{k1}x_1 + \ldots + a_{kn}x_n = b_k$

or

$a_{k1}x_1 + \ldots + a_{kn}x_n \geq b_k$.

where n denotes the number of numerical variables. In other words, a SLICE edit consists of a linear numerical condition that has to hold true if the categorical variables attain certain combinations of values.

18. Some purely categorical edits can directly be written as SLICE edits, namely edits of the following type:

IF ($v_1 \in F_1$ AND … AND $v_m \in F_m$) THEN Error.

Such an edit expresses that certain combinations of categorical values, given by the condition directly following the IF statement, are not allowed.

19. Also certain purely numerical edits can directly be formulated as SLICE edits, namely edits of the type:

IF ($v_1 \in D_1$ AND … AND $v_m \in D_m$) THEN *numerical condition*

Such a numerical condition has to hold true for all possible combinations of categorical values.

20. Edits that are not of type (1), cannot be handled directly by the *Cherry Pie* module or the *AdaptValues* module of SLICE 1.5. Many edits that are not of type (1) can, however, be rewritten by means of the Blaise parser of SLICE 1.5 into a number of edits that are of type (1). In this way a much larger class of (Blaise) edits can be handled than expressed by (1). The Blaise edits that can be processed by the *Cherry Pie* module and the *AdaptValues* module of SLICE 1.5, after rewriting them by means of the Blaise parser of SLICE 1.5, are described in the next subsection.

B. The Blaise edits that SLICE 1.5 can process

21. The Blaise parser of SLICE 1.5, and hence indirectly also the *Cherry Pie* module and the *AdaptValues* module of SLICE 1.5, can process the following class of Blaise edits:

```
IF Condition_1 THEN                                    (2)
        Condition_2
ELSE
        Condition_3
ENDIF
```

Here Condition_2 and Condition_3 may be empty or may themselves be of type (2). Condition_*i* (*i*=1,…,3) may also be of the following form:

Condition_*i*_1 AND/OR Condition_*i*_2 … AND/OR Condition_*i*_*k*. (3)

That is, Condition_*i* (*i*=1,…,3) may be a number of conditions connected by AND and OR operators.

22. In its turn, Condition_*i*_*j* (*i*=1,…,3; *j*=1,…,*k*) may itself be of type (3) or an *elementary expression*. Elementary expressions differ for categorical and numerical variables. An elementary expression for categorical variables involves only a single variable. An elementary expression for numerical variables may involve several variables simultaneously.

23. An elementary expression for a categorical variable v_r is given by

$v_r \in F_r$ (4)

or by

NOT ($v_r \in F_r$). (5)

That is, an elementary expression for a categorical variable expresses that its value is an element (expression (4)), or is not an element (expression (5)) of a certain subset of its domain.

Elementary expressions for a number of numerical variables x_1 to x_n are given by:

- $a_{k1}x_1 + \ldots + a_{kn}x_n \geq b_k$; (6)

- $a_{k1}x_1 + \ldots + a_{kn}x_n > b_k$; (7)
- $a_{k1}x_1 + \ldots + a_{kn}x_n = b_k$; (8)
- NOT ($a_{k1}x_1 + \ldots + a_{kn}x_n \geq b_k$); (9)
- NOT ($a_{k1}x_1 + \ldots + a_{kn}x_n > b_k$); (10)
- NOT ($a_{k1}x_1 + \ldots + a_{kn}x_n = b_k$). (11)

That is, an elementary expression for a number of numerical variables x_1 to x_n expresses that

- a certain linear combination of these variables should be greater than or equal to a certain constant (expression (6)), strictly greater than a certain constant (expression (7)), or equal to a certain constant (expression (8)),

or that

- a certain linear combination of these variables should not be greater than or equal to a certain constant (expression (9)), not strictly greater than a certain constant (expression (10)), or not equal to a certain constant (expression (11)).

24. A complex structure, such as

```
IF Condition_1 THEN
        IF Condition_2 THEN
                Condition_3
        ELSE
                Condition_4
        ENDIF
ELSE Condition_5
ENDIF
```

is split by the Blaise parser of SLICE 1.5 into three simpler edits:

```
IF Condition_1 AND Condition_2 THEN
        Condition_3
ENDIF

IF Condition_1 AND (NOT Condition_2) THEN
        Condition_4
ENDIF

IF (NOT Condition_1) THEN
        Condition_5
ENDIF
```

Subsequently, each of these three edits is further processed until edits are obtained that can be handled by the *Cherry Pie* module and the *AdaptValues* module of SLICE 1.5. During this parsing process auxiliary variables and edits may be introduced in the background. The user does not have to interfere with the construction of these auxiliary variables and edits. SLICE 1.5 take care of the construction and the further processing of these auxiliary variables and edits. For details we refer to Sluis and De Waal (2004).

25. Summarising, SLICE 1.5 can handle nested "IF THEN"-expressions consisting of several elementary categorical and numerical expressions connected by AND and OR operators in Blaise-format.

C. Examples of edits

26. Below we give some simple of examples of Blaise edits that can be parsed by the Blaise parser of SLICE 1.5. The processed edits can subsequently be processed by other modules of SLICE 1.5. As we already mentioned above, SLICE 1.5 can also parse complicated nested "IF THEN"-expressions.

- IF (*Gender = "Male"*) THEN (*Pregnant = "No"*)

- IF (*Tax on wages* > 0) THEN (*Number of employees* ≥ 1)
- (*Costs* > 0) OR (*Gender* = "*Male*") OR (*Married* = "*No*")
- (*X* ≥ 10) OR (*Y* ≥ 100)
- IF ((*Profit* = 0) AND (*Activity code* = "*Chemical Industry*"))
  THEN (*Number of employees* ≤ 1)
- IF ((*Tax on wages* > 0) AND (*Activity code* = "*Chemical Industry*"))
  THEN ((*Number of employees* ≥ 100) OR (*Size* ∈ {"*Medium*", "*Large*"}))

IV. AUTOMATIC ERROR LOCALISATION

27. Statistics Netherlands has devoted many resources to the development of an algorithm for automatic error localisation. An overview of our work in this respect is given by De Waal and Coutinho (2005) and De Waal (2003a). As a result of this work, an algorithm based on vertex generation (Sande, 1978, Schiopu-Kratina and Kovar, 1989, Fillion and Schiopu-Kratina, 1993, De Waal, 2003b) that was implemented in SLICE 1.0 has in SLICE 1.5 been replaced by a branch-and-bound algorithm (De Waal and Quere, 2003). In fact, SLICE 1.5 offers two methods, and corresponding modules, to localise errors. In this section we sketch both methods and modules.

A. Automatic error localisation by means of *Cherry Pie*

28. The algorithm of *Cherry Pie* has been developed to localise errors in records fully automatically. As we already described in Section II, to each variable a weight is assigned, the reliability weight of this variable. The reliability weight of a variable may assume different values for different records. For each record sets of variables are determined of which the values can be adjusted in such a way that all specified edits become satisfied. The optimal sets, *i.e.* the sets with the smallest sum of reliability weights, form the optimal solutions to the error localisation problem. Details on the algorithm of *Cherry Pie* are described in De Waal and Quere (2003) and De Waal (2003a).

29. After termination of *Cherry Pie*, for each record a list of optimal solutions is returned as output. Having such a list of optimal solutions gives us the opportunity to later select an optimal solution that is also best according to a secondary, statistical criterion. This criterion can, for instance, be based on the deviations of the values in the observed record from expected values (see Pannekoek and De Waal, 2005, for an application of such a criterion). The fields in the selected solution are assumed to be erroneous, and their values are set to missing. In a next step these fields, and the fields of which the value was missing in the observed record, may be imputed (see Figure 1 in Section II).

30. To edit data automatically an assumption on the (number of) errors in the data set has to be made. Without making an assumption on the errors in the data set, a computer would be unable to localise the errors. To localise the errors in a data set *Cherry Pie* in fact makes two assumptions. The first assumption is that all edits have to be satisfied. *Cherry Pie* cannot handle "soft" edits, *i.e.* edits that may be violated and are only used as a warning. Each record that does not satisfy all edits is hence considered to be incorrect, and has to be adjusted so it does satisfy all edits.

31. *Cherry Pie* also assumes that the well-known and often used principle of Fellegi and Holt (1976) holds true. The original version of this principle says that the number of errors in a record equals the minimal number of values that has to be changed so that all edits can be satisfied. In other words, the principle of Fellegi and Holt assumes that the fewest possible number of errors has been made while collecting the data and entering these data into the computer system at the statistical office.

32. The principle of Fellegi and Holt cannot be used to localise *systematic* errors, such as net values where gross values should have been answered. Systematic errors have to be detected by means of another system. Such a system will frequently have to rely on subject-matter knowledge that allows one to explain the occurrence of systematic errors. Only after the systematic errors have been corrected one

can apply the principle of Fellegi and Holt, and hence the *Cherry Pie* module of SLICE 1.5, with sufficient confidence. To illustrate the power of the principle of Fellegi and Holt we give an example.

33. *Example* 1: We assume that each record in our data set to be edited contains the values of the variables "profit", "turnover" and "cost" of an enterprise. We denote the values of these variables by *P* (profit), *T* (turnover), and *C* (cost). Logically, these values have to satisfy the following edit:

$$T = P + C. \quad (12)$$

This edit expresses that the profit plus the cost of an enterprise should be equal to the turnover of this enterprise. We assume that four other edits also have to be satisfied. The first two edits express that *T* and *C* should be non-negative:

$$T \geq 0 \quad (13)$$

and

$$C \geq 0. \quad (14)$$

The third edit expresses that the profit of an enterprise is at most 40% of the turnover, and the fourth edit that the cost of an enterprise is at least 50% of the turnover. In formulas we have:

$$P \leq 0{,}4 \times T \quad (15)$$

and

$$C \geq 0{,}5 \times T. \quad (16)$$

The above two edits are based on subject-matter knowledge in combination with statistical analyses, and will only be valid for certain classes of enterprises. This in contrast to, for instance, edit (12) that is strictly logical and has to hold true for every enterprise.

We assume that we encounter a record with the following values in the data set: $P = 755$, $T = 200$ and $C = 125$. This record violates edits (12) and (15). According to the principle of Fellegi and Holt we have to change the fewest possible number of values so that all edits become satisfied. In our example we therefore first try to change the value of only one variable. If this turns out to be impossible, we try to change two, or in the worst case all three, values.

It is clear that it is useless to change only *C* because (15) will then remain violated. We can try to change only *T*. To satisfy (12), *T* would have to become equal to 880, but then (16) would be violated. Finally, we try to change *P*. To satisfy (12), *P* would have to become equal 75. In this case all edits become satisfied. In other words, if we apply the principle of Fellegi and Holt, we conclude that the value of *P* is erroneous. In fact, in this specific case we can even conclude that *P* should be equal to 75. *Cherry Pie* does, however, not carry out an imputation step. Even in this case *Cherry Pie* only localises *P* as erroneous, and sets the value of *P* to missing.

Note that in this example manual editing would probably lead to the same result. Namely, considering the values of the record it seems likely that the respondent accidentally entered a "5" too many while filling in the value of *P*. ■

34. In *Cherry Pie* a generalised version of the principle of Fellegi and Holt is used. As mentioned before, to each field we assign its reliability weight. The larger the reliability weight of a variable, the less likely it is that the value of this field is erroneous. *Cherry Pie* does not try to adjust the fewest possible number of fields so that all edits become satisfied, but to minimise the sum of the reliability weights of the fields to be adjusted. In this way, the user of SLICE 1.5 can take into account that the value of one field is more reliable than the value of another field. Particularly for component variables and the corresponding total variable this is of importance: the total variable is often filled in more accurately than the corresponding component variables. We therefore often prefer changing the value of a component variable to changing the value of the total variable. This can be achieved by assigning a larger reliability weight to the total variable than to the separate component variables.

B. Error localisation in the case of many errors

35. *Cherry Pie* aims to find "optimal" solutions to the error localisation problem. For some, highly contaminated records the algorithm of *Cherry Pie* requires too much computing time or computer memory, however. For such records one can replace the computer intensive (generalised) principle of Fellegi and Holt by a theoretically speaking less good, but less computer intensive principle. This

principle, *the principle of minimum total change*, says that *all* values of an incorrect record may be changed so that all edits become satisfied but that the total change should be as small as possible. In Section VI we explain how the total change is measured. For the missing values arbitrary values are filled in before the principle of minimum total change is applied. After application of this principle we consider all fields for which the value in the new, synthetic record differs from the corresponding value in the observed record as being erroneous. The rest of the edit and imputation process is as usual: the fields that have been identified as missing or as being erroneous are imputed by means of a suitable imputation method and are later possibly adapted so that all edits become satisfied (see Figure 1 in Section II).

36. Application of the principle of minimum total change generally leads to more values being considered as erroneous as when the principle of Fellegi and Holt were applied. The total change in a record (after imputation and possible adjustment of the imputed values) is generally less than when the principle of Fellegi and Holt were applied. Both principles often lead to comparable results in terms of quality of the final data. The application of the principle of minimum total change is usually much faster than the application of the principle of Fellegi and Holt. For more information we refer to an evaluation study by Harte (2000) who has examined the principle of minimum total change in detail for numerical data, and has compared it to the (generalised) principle of Fellegi and Holt.

37. To apply the principle of minimum total change a module in SLICE 1.5 is available. From a technical point of view this module is a modified version of the *AdaptValues* module that we describe in Section VI. For details with respect to the algorithm for the application of the principle of minimum total change we refer to De Waal (2003a; Section 13.2).

38. *Example* 2: For the record of Example 1 and edits (12) to (16) the application of the principle of minimum total change as implemented in SLICE 1.5 leads to the same solution as *Cherry Pie*: again *P* is identified as being erroneous. In general, however, the solutions according to the principle of Fellegi and Holt and the principle of minimum total change will be different.

IV. IMPUTATION OF MISSING VALUES

39. The imputation module of SLICE 1.5 uses regression imputation for imputing the missing values. The edits are not taken into account by the imputation module. After the imputation step the edits can hence be violated. Regression imputation is much more suited for numerical variables than for categorical variables. In the rest of this section we therefore assume that only numerical variables are imputed by means of the imputation module of SLICE 1.5.

40. In order to impute a (numerical) variable x_j (j=1,…,n), a regression model is used. In principle, each other (numerical) variable may be used as predictor. For variable x_j (j=1,…,n) the full model, *i.e.* the model with all other numerical variables as predictors, is given by

$$\hat{x}_j = b_0 + b_1 x_1 + \ldots + b_{j-1} x_{j-1} + b_{j+1} x_{j+1} + \ldots + b_n x_n .$$

Here $\hat{x}_j$ is the imputed value, the x_k (k=1,…,j-1,j+1,…,n) are the values of the other numerical variables, and the b_k (k=0,…,j-1,j+1,…,n) the corresponding regression coefficients.

41. The user of the imputation module of SLICE 1.5 has the specify himself which predictors he wishes to include in the regression model. The user can, for instance, decide to impute the mean of x_j. In that case no predictor is used, and is the regression model given by

$$\hat{x}_j = b_0 ,$$

where b_0 is the mean of x_j.

42. If at most one predictor for a variable to be imputed is specified, then the imputation module of SLICE 1.5 can estimate the corresponding regression coefficient(s) (the constant b_0 and/or the regression coefficient of the predictor). To this end SLICE 1.5 uses a reference data set with complete records, which has to be specified by the user. This reference data set can be a complete data set from a

previous period, or can consist of the complete records in the data set to be imputed. If the user wishes to use more than one predictor for a variable to be imputed, he has to estimate the corresponding regression coefficients himself, for instance by means of a statistical package such as SPSS, and in enter these estimated regression coefficients in SLICE 1.5.

43. *Example* 3: In Example 1 the value of P was localised as being erroneous. Suppose we use the following imputation model for P

$$P = 10 + 0.5 \times T,$$

where the regression coefficients (10 and 0.5) either have been estimated by SLICE 1.5 using a reference data set, or have been estimated and specified by the user. We then impute then the value $10 + 0.5 \times 200 = 110$ for P. The imputed record, $P = 110$, $T = 200$ and $C = 125$, does not satisfy all edits. The value imputed for P hence has to be adjusted later in order to satisfy all edits. In Section VI we discuss the adjustment of imputed values.

44. An imputation model is specified in SLICE by means of imputation rules (De Waal and Wings, 1999). These imputation rules have an easy to understand form. In Example 3 the imputation rule for P would be given by

$$P = \texttt{Linear}(T),$$

if the user wants SLICE to calculate the coefficients, and by

$$P = \texttt{User}(10 + 0.5 \times T),$$

if the user wants to specify the coefficients himself.

VI. ADJUSTMENT OF IMPUTED VALUES

45. The algorithm of the *AdaptValues* module of SLICE 1.5 has been developed to adjust the imputed values in a record so that the adjusted record satisfies all specified edits. The *AdaptValues* module tries to minimise the total change in a record subject to the restriction that the final record satisfies all edits.

46. The *AdaptValues* module modifies only imputed values. If *Cherry Pie* or the principle of minimum total change (see Section IV, Subsection B) has been used to localise the erroneous values, and hence which values have to be imputed, it is guaranteed the *AdaptValues* module can find a solution. If *Cherry Pie* or the principle of minimum total change is not used, it is not guaranteed that the imputed values can be adjusted so that all edits become satisfied. If the imputed values cannot be adjusted so all edits become satisfied, the *AdaptValues* module can obviously not find a solution, which is then reported.

47. The *AdaptValues* module in fact applies the principle of minimum total change discussed in Section IV (Subsection B), with the extra condition that only the imputed values in a record may be adjusted. To measure the total change in a record, weights can be specified by the user. Such a weight indicates the loss when the value of the corresponding variable is changed. These weights may differ from the reliability weights. In SLICE 1.5, the total change for numerical data is measured by taking the sum of absolute differences between the observed values and the modified values, weighted by the specified weights. The total change for categorical data is measured by summing the weights for those categorical variables for which the observed value differs from the modified value. The default value of all weights equals 1.

48. Technical details on the algorithm of the *AdaptValues* module of SLICE 1.5 are described in Kartika (2001) and De Waal (2003a; Section 12.4). Below we illustrate the *AdaptValues* module by means of an example.

49. *Example* 4: In Example 3 we have imputed the value 110 for variable P. The record we then obtain does not yet satisfy the edits (12) to (16). We therefore use the *AdaptValues* module to adjust the imputed value. *AdaptValues* adjusts the imputed value as little as possible subject to the restriction that

the final, adjusted record satisfies all edits. In this case the solution is obvious: *P* has to be equal to 75 in order to satisfy all edits.

50. In Example 4 is the final value for *P* is easy to determine. In general the solution to the problem that the *AdaptValues* module is trying to solve is, however, not easy to find.

VII. DISCUSSION

51. In this paper we have given an overview of automatic edit and imputation by means of SLICE 1.5 SLICE 1.5 is quite generic software tool: it can handle a mix of categorical, continuous and integer-valued variables, as well as a large class of edits involving these variables. The modules can be used in combination or separately.

52. Statistics Netherlands continues working on the further development of SLICE. At the moment, attention is focussed on improving the computing time of the modules for localising errors automatically, especially the *Cherry Pie* module. By reducing the computing time (even) larger and more contaminated data set could be edited automatically by means of SLICE.

53. In the more distant future we hope to develop an imputation method that takes the edits into account, and implement this method into SLICE. The imputation step and the adjustment step could then be replaced by a single imputation step. We expect that this will lead to imputations of higher statistical quality. Methodological research on such imputation methods is currently carried out at Statistics Netherlands (Tempelman and Steerneman, 2004; Tempelman, 2005; Van den Eijkhof, De Waal and Pannekoek, 2005).

REFERENCES

De Jong, A. (2002), *Uni-Edit: Standardized Processing of Structural Business Statistics in the Netherlands.* UN/ECE Work Session on Statistical Data Editing, Helsinki.

De Waal, T. (2001a), *SLICE: Generalised Software for Statistical Data Editing*. In: Proceedings in Computational Statistics (ed. J.G. Bethlehem and P.G.M. of der Heijden), Physica-Verlag, New York, pp. 277-282.

De Waal, T. (2001b), WAID 4.1: A Computer Program for Imputation of Missing Values. *Research in Official Statistics 4*, pp. 53-70.

De Waal, T. (2003a), *Processing of Erroneous and Unsafe Data*. Ph.D. Thesis, Erasmus University, Rotterdam

De Waal, T. (2003b), Solving the Error Localization Problem by Means of Vertex Generation. *Survey Methodology 29*, pp. 71-79.

De Waal, T. (2005), Automatic Error Localisation for Categorical, Continuous and Integer Data. *Statistics and Operations Research Transactions*, forthcoming.

De Waal, T. and W. Coutinho (2005), Automatic Editing for Business Surveys: an Assessment for Selected Algorithms. *International Statistical Review*, forthcoming.

De Waal, T. and R. Quere (2003), A Fast and Simple Algorithm for Automatic Editing of Mixed Data. *Journal of Official Statistics 19*, pp. 383-402.

De Waal, T. and H. Wings (1999), *From CherryPi to SLICE*. UN/ECE Work Session on Statistical Data Editing, Rome.

Fellegi, I.P. and D. Holt (1976). A Systematic Approach to Automatic Edit and Imputation. *Journal of the American Statistical Association 71*, pp. 17-35.

Fillion, J.M. and I. Schiopu-Kratina (1993), *On the Use of Chernikova's Algorithm for Error Localization*. Report, Statistics Canada.

Granquist, L. (1990), A Review of Some Macro-Editing Methods for Rationalizing the Editing Process. *Proceedings of the Statistics Canada Symposium*, pp. 225-234.

Harte, P. (2000), *Testing Automatic Editing by Means of the Simplex Method* (in Dutch). Internal note, Statistics Netherlands, Voorburg.

Hedlin, D. (2003), Score Functions to Reduce Business Survey Editing at the U.K. Office for National Statistics. *Journal of Official Statistics 19*, pp. 177-199.

Hoogland, J. (2002), *Selective Editing by Means of Plausibility Indicators*. UN/ECE Work Session on Statistical Data Editing, Helsinki.

Hoogland, J. (2005), *Evaluation of Score Functions for Selective Editing of Annual Structural Business Statistics*. UN/ECE Work Session on Statistical Data Editing, Ottawa.

Hoogland, J. and J. Pannekoek (2000), *Evaluation of SPSS Missing Value Analysis 7.5*. Report (research paper 0014), Statistics Netherlands, Voorburg.

Kartika, W. (2001), *Consistent Imputation for Categorical and Numerical Data*. Report (research paper 0113), Statistics Netherlands, Voorburg.

Lawrence, D. and R. McKenzie (2000), The General Application of Significance Editing. *Journal of Official Statistics 16*, pp. 243-253.

Pannekoek, J. and T. De Waal (2005), Automatic Editing and Imputation for Business Surveys: The Dutch Contribution to the EUREDIT Project. *Journal of Official Statistics*, forthcoming.

Sande, G. (1978), *An Algorithm for the Fields to Impute Problems of Numerical and Coded Data.* Report, Statistics Canada.

Schiopu-Kratina, I. and J.G. Kovar (1989), *Use of Chernikova's Algorithm in the Generalized Edit and Imputation System*. Report, Statistics Canada.

Sluis, W. and T. De Waal (2004), *The Methodology of SLICE 1.5: An Algorithm for the Parsing Blaise Edits to SLICE Edits* (in Dutch). Internal note, Statistics Netherlands, Voorburg.

Tempelman, C. (2005), *Imputation of Data Subject to Balance and Inequality Restrictions Using the Truncated Normal Distribution*. UN/ECE Work Session on Statistical Data Editing, Ottawa.

Tempelman, C. and T. Steerneman (2004), *Imputation for Economic Data under Linear Restrictions*. Report (discussion paper 04002), Statistics Netherlands, Voorburg.

Van den Eijkhof, F., T. De Waal and J. Pannekoek (2005), *On the Imputation of Categorical Data Subject to Edit Restrictions Using Loglinear Models*. UN/ECE Work Session on Statistical Data Editing, Ottawa.

Van Velzen, J. (2005), *The Embedding of a Uniform Statistical Process*. UN/ECE Work Session on Statistical Data Editing, Ottawa.
