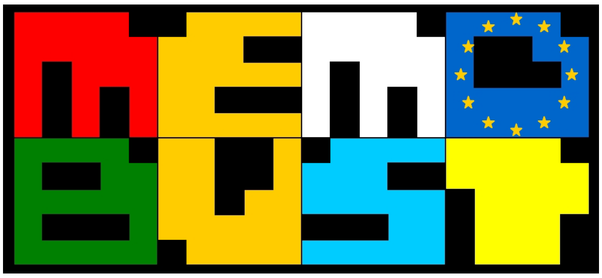

This module is part of the

Memobust Handbook

on Methodology of Modern Business Statistics

26 March 2014

Method: Automatic Editing

Contents

General section

1. Summary

The goal of automatic editing is to accurately detect and treat errors and missing values in a data file in a fully automated manner, i.e., without human intervention. Methods for automatic editing have been investigated at statistical institutes since the 1960s (Nordbotten, 1963). In practice, automatic editing usually implies that the data are made consistent with respect to a set of predefined constraints: the so-called *edit rules* or *edits*. The data file is checked record by record. If a record fails one or more edit rules, the method produces a list of fields that can be imputed so that all rules are satisfied.

In this module, we focus on automatic editing based on the (generalised) Fellegi-Holt paradigm. This means that the smallest (weighted) number of fields is determined which will allow the record to be imputed consistently. Designating the fields to be imputed is called error localisation. In practice, error localisation by applying the Fellegi-Holt paradigm often requires dedicated software, due to the computational complexity of the problem.

Although the imputation of new values for erroneous fields is often seen as a part of automatic editing, we do not discuss this here, because the topic of imputation is broad and interesting enough to merit a separate description. We refer to the theme module ‘Imputation’ and its associated method modules for a treatment of imputation in general and various imputation methods.

2. General description of the method

2.1 Introduction to automatic editing

For efficiency reasons, it can be desirable to edit at least part of a data file by means of automatic methods (see “Statistical Data Editing – Main Module”). Assuming that all systematic errors with a known structural cause have already been treated using methods for deductive editing (see the method module “Statistical Data Editing – Deductive Editing”), the task remains to also detect and treat random errors. In the literature on data editing, the problem of identifying the erroneous values in a record containing only random errors is known as the *error localisation problem*. Compared to detecting systematic errors, solving the error localisation problem is usually more difficult and requires complex methodology.

Broadly speaking, there are two approaches to solving the error localisation problem. The first approach uses outlier detection techniques in combination with an implicit or explicit statistical model for the data under consideration. Records corresponding to data points that do not fit the model well are supposed to contain errors, and within such a record, the values that contribute most to the ‘outlyingness’ of that record are identified as erroneous; see, e.g., Little and Smith (1987) and Ghosh-Dastidar and Schafer (2003). This approach appears to be mainly suitable for editing low-dimensional data (data sets containing a small number of variables). Moreover, if there are edit rules that define consistency constraints for the variables in the data set, these cannot be used under this approach. In particular, the edited data will not necessarily satisfy the edit rules. For these reasons, this approach is not ideal for automatic editing in business surveys at statistical offices, where one typically encounters data sets with many variables and many edit rules. In fact, it is seldom used in this context.

In the remainder of this module, we shall focus on the second approach. Under this approach, a set of edit rules is defined for the data set. A record is called *consistent* – and is considered to be error-free – if it satisfies all edit rules. For inconsistent records, the erroneous values are identified by solving a mathematical optimisation problem.

The remainder of this section is organised as follows. Section 2.2 considers edit rules. In Section 2.3, the error localisation problem is formulated as a mathematical optimisation problem. Sections 2.4 and 2.5 describe techniques for solving this optimisation problem.

2.2 Edit rules

Edit rules are introduced in a more general context in "Statistical Data Editing – Main Module". Here, we focus on aspects of edit rules that are relevant to automatic editing in particular.

A record of data can be represented as a vector of fields or variables: $x = (x_1, x_2, \ldots, x_n)$. The set of values that can be taken by variable x_i is called its domain. Examples of variables and domains are *size class* with domain $\{\text{'small'}, \text{'medium'}, \text{'large'}\}$, *number of employees* with domain $\{0,1,2,\ldots\}$, and *profit* with domain $(-\infty, \infty)$.

Edit rules indicate conditions that should be satisfied[1] by the values of single variables or combinations of variables in a record. For the purpose of automatic editing, all edit rules must be checkable per record, and may therefore not depend on values in fields of other records. However, they may contain parameters based on external sources (for instance, quantiles of univariate distributions in a reference data set that has been edited previously), provided that these parameters are set prior to the start of the editing process.

For automatic editing of numerical data, it is convenient to assume that all edit rules are written as linear relationships such as

$$Turnover \geq 0$$

or

$$Profit + Costs = Turnover.$$

The general form of a linear edit rule for a record $(x_1, x_2, \ldots, x_n)$ is as follows:

$$a_{j1}x_1 + \cdots + a_{jn}x_n + b_j \geq 0 \tag{1}$$

or

$$a_{j1}x_1 + \cdots + a_{jn}x_n + b_j = 0, \tag{2}$$

---

[1] Edit rules of this type are sometimes called 'validity rules'. In some applications, edit rules are specified instead in the form of 'conflict rules', which means that they indicate conditions that are satisfied by invalid combinations of values. For instance, an edit rule stating that the variable *turnover* should be non-negative can be written either as the validity rule '*turnover* ≥ 0' or as the conflict rule '*turnover* < 0'. Clearly, both formulations are equivalent. The choice of validity or conflict rules should not lead to difficulties, provided that one of the forms is used consistently.

where j numbers the edit rules, a_{ji} are numerical coefficients and b_j are numerical constants. It should be noted that a *ratio edit* – i.e., a bivariate edit rule of the form

$$x_1 / x_2 \geq a ,$$

where a denotes a numerical constant and x_1 and x_2 are constrained to be non-negative – can also be expressed as a linear edit rule. Namely, the ratio edit can be rewritten as

$$x_1 - ax_2 \geq 0 .$$

For categorical data, an edit rule can identify as admissible any combination of values from the domains of the categorical variables. Categorical edit rules are often written in if-then form, for example:

if *Gender* = 'male' **then** *Pregnant* = 'no'.

Finally, mixed data and mixed edit rules, containing both categorical and numerical variables also occur in practice. Mixed edit rules are also often written in if-then form. For example:

if *Size Class* = 'small' **then** *Number of Employees* < 10.

For automatic processing, it can be convenient to require that the if-part of a mixed edit only contains categorical variables, while the then-part only contains numerical variables. The above-mentioned example is written in this form. Many types of mixed edits can be rewritten in this simple form, although this may require the introduction of auxiliary variables; see De Waal (2005).

In the remainder of this module, we focus on numerical data and linear edits, because these are most common to business surveys. We refer to De Waal et al. (2011) for a discussion of automatic editing of categorical or mixed data. A numerical variable x_i is said to be *involved* in an edit rule of the form (1) or (2) if it holds that $a_{ji} \neq 0$. Clearly, whether a record fails or satisfies an edit rule only depends on the values of the variables that are involved in that edit rule.

In manual editing, subject-matter specialists often distinguish between *hard* and *soft* edit rules. As mentioned in "Statistical Data Editing – Main Module", hard edit rules are rules that must hold by definition, while soft edit rules only indicate whether a value, or value combination, is suspicious. A soft edit rule can occasionally be failed by unlikely values that are in fact correct.

In nearly[2] all methods for automatic editing, no distinction can be made between hard and soft edit rules: all rules are treated as hard edit rules. Thus, in automatic error localisation, all records that fail one or more edit rules are viewed as certainly inconsistent. Hence, formulating edits for the purpose of automatic editing should be done with care (Di Zio et al., 2005). If too many soft edit rules are defined, or soft edit rules that are too strict, there is a danger of *overediting*: the unjustified adaptation of correct values. On the other hand, if too few edit rules are defined, or soft edit rules that are not strict enough, then certain errors might be left in the data after automatic editing.

---

[2] In fact, to our best knowledge, all methods for automatic editing that are currently in use at statistical offices do not distinguish between hard and soft edit rules. The method of Freund and Hartley (1967) uses soft edit rules, but it has the important drawback that it cannot handle hard edit rules; hence, it is not recommended to be used in practice. Scholtus (2013) has described a method that incorporates both hard and soft edit rules, but at the time of writing, this method remains to be tested in practice.

2.3 *The error localisation problem*

For a given record and a collection of edit rules, it is straightforward to verify which values in the record are missing and whether any of the edit rules are failed. However, given that some of the edit rules are failed, determining which values in the record are actually causing the edit failures is much less straightforward. On the one hand, most edit rules involve more than one variable, and on the other hand, most variables are involved in more than one edit rule.

In order to solve the error localisation problem automatically, one has to choose a guiding principle for finding errors. The most commonly used guiding principle for error localisation is the so-called *Fellegi-Holt paradigm*, first formulated by Fellegi and Holt (1976). According to this paradigm, one should minimise the number of observed values that have to be adjusted in order to satisfy all edit rules. This paradigm is often used in a generalised form, for which each variable is given a *reliability weight* $w_i \geq 0$. A high value of w_i indicates that the variable x_i is expected to contain few errors. The generalised Fellegi-Holt paradigm now states that one should search for a subset of the variables E with the following two properties:

- The variables x_i ($i \in E$) can be imputed with values that, together with the observed values of the other variables in the record, satisfy all edit rules.
- Among all subsets that satisfy the first property, E has the smallest value of $\sum_{i \in E} w_i$.

The original Fellegi-Holt paradigm is recovered from this more general form by taking all reliability weights equal, for instance all equal to 1.

A distinctive feature of the (generalised) Fellegi-Holt paradigm is that it does not take the size of the differences between the original and imputed values into account in any way. In fact, the method of Fellegi and Holt only provides a list of variables that can be imputed to satisfy all edit rules, but it does not provide the actual values to impute. These have to be determined in a separate step. This might seem like a drawback, but it actually has the advantage that an appropriate imputation method can be chosen independently of the method used for error localisation. Methods for imputation are discussed in the topic “Imputation”.

Some authors have suggested other guiding principles for error localisation that do look at the size of the adaptations. Casado Valera et al. (1996) proposed to minimise the sum of the squared differences between the observed values and the adjusted values, under the restriction that all edit rules are satisfied by the adjusted values. This leads to a quadratic optimisation problem, which can be solved using standard software. A different formulation of the error localisation problem as a quadratic optimisation problem was proposed by Freund and Hartley (1967).

To illustrate the difference between these principles, we consider a very small example. Suppose that there are two edit rules:

$$Turnover = Profit + Costs,$$

$$Turnover \geq 0,$$

and suppose that we are presented with the following inconsistent record:

$$(Turnover, Profit, Costs) = (-30, 10, 20).$$

Under the Fellegi-Holt paradigm (in its original form, without reliability weights), the optimal solution is to adjust only the value of *Turnover*, because both edits can be satisfied without changing the values of the other variables. After imputation, this certainly yields

(*Turnover*, *Profit*, *Costs*) = (**30**, 10, 20),

because the value to impute for *Turnover* is uniquely determined by the edits in this example.

On the other hand, if we minimise the unweighted sum of the squared differences between observed and adjusted values, the optimal solution changes all values:

(*Turnover*, *Profit*, *Costs*) = (**0**, **–5**, **5**).

This happens because, under this minimisation criterion, it is optimal to distribute the total adjustment required by the edit rules over as many different variables as possible.

Assuming that errors occur with a low probability and in isolated values, the Fellegi-Holt paradigm appears to be a sensible choice, because it distorts as few of the observed values as possible. Methods that try to distribute the total adjustment over many different variables, such as the quadratic minimisation approach, are less suitable in this context. However, the latter type of method can be useful in the context of micro- or macro-integration, where many small inconsistencies in data from different sources have to be resolved, while preserving patterns that occur in the original data as much as possible. We refer to the topics “Micro-Fusion” (in particular the method module “Micro-Fusion – Reconciling Conflicting Microdata”) and “Macro-Integration” for these subjects.

In order to solve the error localisation problem according to the Fellegi-Holt paradigm, we have to find the smallest subset of the variables that can be imputed so that all edit rules become satisfied. Several methods have been proposed for this. Section 2.4 presents the original method of Fellegi and Holt (1976) for numerical data. Section 2.5 briefly mentions several other methods. These sections contain material that is somewhat more technical than the rest of this module.

2.4 Solving the error localisation problem: the method of Fellegi and Holt

For a given record that fails certain edit rules, we want to determine the smallest subset of the variables that can be imputed so that all edit failures are resolved. A naïve way to solve this problem might proceed as follows: “It is clear that a subset of the variables E can only be a feasible solution to the error localisation problem if every failed edit rule involves at least one variable in E, i.e., if the failed edit rules are ‘covered’ by these variables. Therefore, let us choose the smallest set of variables with this property.” Unfortunately, although ‘covering’ the original failed edits is a necessary condition for a set of variables to be a feasible solution to the error localisation problem, it is not a sufficient condition in general. We will demonstrate this by means of a small example.

Consider the following two numerical edit rules: $x_1 \geq x_2$ and $x_2 \geq x_3$. The unedited record $(x_1, x_2, x_3) = (4,5,6)$ fails both edits. Since the variable x_2 is involved in both edit rules – that is to say, the failed edits are ‘covered’ by x_2 –, we might try to obtain consistency with respect to the edit rules by changing only the value of x_2. This turns out to be impossible, because the imputed value would have to satisfy $4 \geq x_2$ and $x_2 \geq 6$.

Fellegi and Holt (1976) showed that, in order to determine whether a set of variables can be imputed to satisfy all edits simultaneously, it is necessary to derive so-called *implied edits* from the original set of edits. An implied edit is an edit rule that can be derived from the original edit rules by logical reasoning. For numerical data, the number of implied edits that can be derived from even a single original edit is actually infinite; e.g., if $x_1 \geq x_2$ is an edit rule, then so is $\lambda x_1 \geq \lambda x_2$ for any $\lambda > 0$. Fortunately, for the purpose of solving the error localisation problem, it is not necessary to derive all possible implied edits from the original set of edits, but only the so-called *essentially new implied edits* (see below). By adding the essentially new implied edits to the original set of edit rules, one obtains a so-called *complete set of edits*. For a complete set of edit rules, it does hold that any subset of the variables which 'covers' all failed edit rules is a feasible solution to the error localisation problem.

In the example above, the complete set of edits consists of the two original edit rules and the (only) essentially new implied edit $x_1 \geq x_3$. The latter edit rule is also failed and it does not involve the variable x_2, which shows that imputing only x_2 does not solve the error localisation problem. On the other hand, the three failed edits are 'covered' by $\{x_1, x_3\}$, and it is easy to see that imputing new values for x_1 and x_3 is indeed a feasible solution to the error localisation problem. In fact, imputing any combination of values with $x_1 \geq 5$ and $5 \geq x_3$ leads to a consistent record in this example.

In general, for a given set of edit rules of the forms (1) and (2), essentially new implied edits are constructed by selecting one of the variables, say x_g, as a so-called *generating variable*. We consider all pairs of edit rules that involve the generating variable, i.e., all pairs (s,t) with $a_{sg} \neq 0$ and $a_{tg} \neq 0$. If one of the edits, say edit s, is an equality, then we may solve this equality for x_g:

$$x_g = \frac{-1}{a_{sg}} \left(a_{s1} x_1 + \cdots + a_{s,g-1} x_{g-1} + a_{s,g+1} x_{g+1} + \cdots + a_{sn} x_n + b_s \right).$$

An implied edit is now obtained from the pair (s,t) by substituting this expression for x_g in edit rule t. This new edit rule is an essentially new implied edit, unless it happens to be identical to an existing edit rule, in which case it is redundant.[3]

If both edits are inequalities, then we apply a technique called *Fourier-Motzkin elimination* (Williams, 1986; De Waal et al., 2011). First, we check whether the coefficients a_{sg} and a_{tg} have opposite signs, i.e., whether $a_{sg} a_{tg} < 0$. If this is not the case, then this pair does not contribute an essentially new implied edit. Hence, we may assume without loss of generality that $a_{sg} < 0$ and $a_{tg} > 0$. This means that edit rule s can be written as an upper bound on x_g, given the values of the other variables:

$$x_g \leq \frac{-1}{a_{sg}} \left(a_{s1} x_1 + \cdots + a_{s,g-1} x_{g-1} + a_{s,g+1} x_{g+1} + \cdots + a_{sn} x_n + b_s \right).$$

---

[3] To give an example of a redundant edit, suppose that we already have the edit rule '$x \geq 3$' and we derive a new edit rule stating that '$2x \geq 6$'. Since the second edit rule is identical to the first one after simplification, it does not provide any new information and is therefore redundant.

Similarly, edit rule t can be written as a lower bound on x_g:

$$x_g \geq \frac{-1}{a_{tg}}\left(a_{t1}x_1 + \cdots + a_{t,g-1}x_{g-1} + a_{t,g+1}x_{g+1} + \cdots + a_{tn}x_n + b_t\right).$$

Combining the two bounds and removing x_g, we obtain the implicit condition

$$\frac{-1}{a_{sg}}\left(a_{s1}x_1 + \cdots + a_{s,g-1}x_{g-1} + a_{s,g+1}x_{g+1} + \cdots + a_{sn}x_n + b_s\right)$$
$$\geq \frac{-1}{a_{tg}}\left(a_{t1}x_1 + \cdots + a_{t,g-1}x_{g-1} + a_{t,g+1}x_{g+1} + \cdots + a_{tn}x_n + b_t\right)$$

which can be written in the general form (1) as

$$a_1^* x_1 + \cdots + a_n^* x_n + b^* \geq 0,$$

with $a_i^* = a_{tg}a_{si} - a_{sg}a_{ti}$ ($i = 1,\ldots,n$) and $b^* = a_{tg}b_s - a_{sg}b_t$. This is an essentially new implied edit that is derived from the pair of inequality edits (s,t), unless it happens to be redundant (see footnote 3).

It should be noted that, both for equalities and inequalities, the essentially new implied edit generated by this procedure does not involve the generating variable (i.e., the coefficient $a_g^* = 0$). This is in fact the defining property that makes an implied edit 'essentially new': it adds information to the existing edit rules by eliminating one of the variables.

According to the method of Fellegi and Holt (1976), a complete set of edits may be constructed by repeatedly applying the above-mentioned procedure of generating essentially new implied edits, using all variables in turn as generating variables, until no more (non-redundant) new edits can be derived. At that point, a complete set of edits has been generated.

Having obtained a complete set of edits, one can solve the error localisation problem for any given record in the following manner:

- Select all edits from the complete set of edits that are failed by the original record.
- Find the smallest (weighted) subset of the variables with the property that each selected (original or implied) edit involves at least one of them.

The first step amounts to evaluating the edits for a given record. The second step entails solving a set-covering problem, which is a well-known mathematical problem for which standard algorithms are available (see, e.g., Nemhauser and Wolsey, 1988). We shall work out a small example with the Fellegi-Holt method in Section 4.

A crucial element of the Fellegi-Holt method is the fact that a complete set of edits is 'sufficiently large' to reduce the error localisation problem to a set-covering problem. For a proof of this fact, see Fellegi and Holt (1976). For an explanation of what is meant by 'sufficiently large' from the viewpoint of logic, see Boskovitz et al. (2005).

The method discussed in this section works for numerical variables, but an analogous method exists for categorical variables. The only difference lies in the procedure for generating essentially new implied edits. We refer to Fellegi and Holt (1976) and De Waal et al. (2011) for a description of the Fellegi-Holt method for categorical data.

2.5 *Solving the error localisation problem: other methods*

An important drawback of the method of Fellegi and Holt discussed in Section 2.4 is that the complete set of edits can be extremely large, especially with numerical data. In many practical applications, generating a complete set of edits is simply not technically feasible.[4] For this reason, other algorithms have been developed that solve the error localisation problem without generating a complete set of edits. We can distinguish several classes of such algorithms.

Algorithms based on vertex generation

It is known from the literature that the optimal solution to the error localisation problem for a given record always corresponds with one of the vertices of an appropriately defined polyhedron; see, e.g., Theorem 3.1 in De Waal et al. (2011). Hence, in principle, the error localisation problem can be solved by generating all vertices of that polyhedron and identifying the optimal one. This approach has been elaborated in several error localisation algorithms. See, among others, Sande (1978), Kovar and Whitridge (1990), Fillion and Schiopu-Kratina (1993), Todaro (1999), and De Waal (2003). Tools for automatic editing that use algorithms based on vertex generation include GEIS (Kovar and Whitridge, 1990), Banff (Banff Support Team, 2008), CherryPi (De Waal, 1996), and AGGIES (Todaro, 1999).

Branch-and-bound algorithm

De Waal and Quere (2003) describe how the error localisation problem may be solved by means of a branch-and-bound algorithm. For a record containing n numerical variables, there are 2^n potential solutions, since each variable is either fixed to its original value or imputed. Basically, the branch-and-bound algorithm systematically considers all potential solutions and checks which of these are feasible. In order to do this, the algorithm generates relevant essentially new implied edits 'on the fly', but it does not construct a complete set of edits. Finally, the algorithm selects the feasible solution with the smallest sum of reliability weights. A similar branch-and-bound algorithm can be used for categorical or mixed data. We refer to De Waal and Quere (2003), De Waal (2003), and De Waal et al. (2011) for more details. Tools for automatic editing that use the branch-and-bound algorithm include SLICE (De Waal, 2005) and the R package `editrules` (De Jonge and Van der Loo, 2011).

Algorithms based on cutting planes

With this approach, to solve the error localisation problem for a given record, one starts by finding the minimal subset of the variables that 'covers' all original edit rules that are failed. As we have seen above, this solution may be infeasible. In that case, the algorithm generates new constraints, so-called

---

[4] One exception occurs when all edit rules are ratio edits: it can be shown that, for a data set with n variables, the complete set of edits contains at most $n(n-1)/2$ non-redundant ratio edits. Thus, for ratio edits, the Fellegi-Holt method is usually feasible; see Winkler and Draper (1997).

cutting planes, and adds these to the original set of edit rules. Next, a minimal covering set of variables is determined for the new problem. Again, this solution may be infeasible, in which case more cutting planes need to be generated. In this iterative manner, the algorithm continues until it finds a feasible solution to the error localisation problem. For more details, we refer to Garfinkel et al. (1988), Ragsdale and McKeown (1996), and De Waal et al. (2011).

Algorithms for mixed integer programming

Finally, it is also possible to formulate the error localisation problem according to the Fellegi-Holt paradigm as a mixed integer programming problem; see, e.g., Riera-Ledesma and Salazar-González (2003). This type of problem can be solved by commercially available solvers.

De Waal and Coutinho (2005) compared the performance of several different algorithms for error localisation. They did not find a strong preference for one particular algorithm. Note that ‘performance’ here refers simply to computational efficiency. All of the above algorithms try to solve the same error localisation problem and hence, in theory, should find the same solution.[5]

3. Preparatory phase

The method discussed in this module is only considered appropriate for identifying random errors. Therefore, it is important to treat systematic errors, such as unit of measurement errors, before applying this method. Methods for detecting and treating systematic errors are discussed in the method module “Statistical Data Editing – Deductive Editing”.

In addition, automatic editing is usually applied in combination with a form of selective editing: the most influential errors are edited manually by subject-matter experts, while the other, non-influential errors are resolved automatically. Selective editing and manual editing are discussed in the theme module “Statistical Data Editing – Selective Editing” and the method module “Statistical Data Editing – Manual Editing”, respectively. We refer to “Statistical Data Editing – Main Module” for a discussion on how to combine different editing methods into one editing process. See also Pannekoek and De Waal (2005) for suggestions on how to set up an automatic editing strategy in practice.

4. Examples – not tool specific

To illustrate the method of Fellegi and Holt discussed in Section 2.4, we work out an example based on Fellegi and Holt (1976). In this example, there are four numerical variables. We do not use different reliability weights. The original set of edit rules consists of two edits:

$$x_1 - x_2 + x_3 + x_4 \geq 0 \tag{3}$$

and

$$-x_1 + 2x_2 - 3x_3 \geq 0\,. \tag{4}$$

---

[5] In practice, the error localisation problem according to the Fellegi-Holt paradigm may have several equivalent optimal solutions, particularly if many variables have the same reliability weight. When this occurs, different implementations of these algorithms may differ in the way they choose between equivalent solutions.

By a repeated application of Fourier-Motzkin elimination, it is possible to derive the following essentially new implied edits from (3) and (4):

$$x_2 - 2x_3 + x_4 \geq 0, \tag{5}$$

$$x_1 - x_3 + 2x_4 \geq 0, \tag{6}$$

and

$$2x_1 - x_2 + 3x_4 \geq 0. \tag{7}$$

It is not possible to generate more essentially new implied edits from (3)–(7), so these five edit rules together constitute a complete set of edits. This means that we can now solve the error localisation problem for any record by solving an appropriate set-covering problem.

Consider the record $(x_1, x_2, x_3, x_4) = (3,4,6,1)$. By checking the edit rules (3)–(7), it is seen that this record fails edits (4), (5), and (6). Thus, in order to solve the error localisation problem, we have to find the minimal subset of variables that 'covers' these three edit rules. By inspection, we see that the variable x_3 is involved in edit rules (4), (5), and (6). Thus, in this example, x_3 can be imputed to satisfy all the edit rules. Since $\{x_3\}$ is the only single-variable set with this property, changing the value of x_3 is in fact the optimal solution to the error localisation problem for this record. [Note that the single-variable sets $\{x_1\}$ and $\{x_2\}$ cover the original failed edit (4), but not the implied failed edits (5) and (6).] A consistent record can be obtained by imputing, for instance, the value $x_3 = 1$.

5. Examples – tool specific

The R package `editrules`, which can be downloaded for free at http://cran.r-project.org, contains an implementation of the branch-and-bound algorithm of De Waal and Quere (2003). To illustrate the use of `editrules` for automatic editing, we work out the example from Section 4 in R code.[6]

First, we load the package:

```
> library(editrules)
```

Next, we create an object of type "editmatrix" containing the two original edit rules:

```
> E <- editmatrix(c("x1-x2+x3+x4 >= 0", "-x1+2*x2-3*x3 >= 0"))
```

We also have to read in the record that we want to edit as a data frame:

```
> x <- data.frame(x1 = 3, x2 = 4, x3 = 6, x4 = 1)
```

Now, the error localisation problem is solved to optimality by giving the following command:

```
> le <- localizeErrors(E, x)
```

This command runs the branch-and-bound algorithm to solve the error localisation problem and stores the results in a new object called `le`. The results can be inspected by calling attributes of this object.

---

[6] Version 2.5 of the `editrules` package was used to run the code in this example.

```
> le$status
  weight degeneracy user system elapsed maxDurationExceeded
1      1          1 0.05      0    0.13               FALSE
```

The attribute `le$status` contains background information on the performance of the algorithm. In this example, an optimal solution has been found with the sum of the reliability weights equal to 1 (as can be seen in the column `weight`). Since we have not specified the reliability weights in this example, R has used the default choice: all weights equal to 1. Other reliability weights can be specified by providing the function `localizeErrors` with an optional argument `weight`. The entry '1' in the column `degeneracy` in `le$status` shows that the optimal solution is unique.

To see which variables have to be changed according to the optimal solution, we inspect the attribute `le$adapt`.

```
> le$adapt
     x1    x2   x3    x4
1 FALSE FALSE TRUE FALSE
```

This command prints a boolean data frame with the value 'TRUE' for variables that have to be changed, and the value 'FALSE' for the other variables. In this example, the optimal solution is to change only the value of variable x_3. This solution is identical to the one found in Section 4 by applying the method of Fellegi and Holt.

We refer to De Jonge and Van der Loo (2011) for more details on the `editrules` package.

6. Glossary

For definitions of terms used in this module, please refer to the separate "Glossary" provided as part of the handbook.

7. References

Banff Support Team (2008), Functional Description of the Banff System for Edit and Imputation. Technical Report, Statistics Canada.

Boskovitz, A., Goré, R., and Wong, P. (2005), Data Editing and Logic. Working Paper, UN/ECE Work Session on Statistical Data Editing, Ottawa.

Casado Valero, C., Del Castillo Cuervo-Arango, F., Mateo Ayerra, J., and De Santos Ballesteros, A. (1996), Quantitative Data Editing: Quadratic Programming Method. Presented at the COMPSTAT 1996 Conference, Barcelona.

De Jonge, E. and van der Loo, M. (2011), Manipulation of Linear Edits and Error Localization with the Editrules Package. Discussion Paper 201120, Statistics Netherlands, The Hague.

De Waal, T. (1996), CherryPi: a Computer Program for Automatic Edit and Imputation. Working Paper, UN/ECE Work Session on Statistical Data Editing, Voorburg.

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

De Waal, T. (2005), SLICE 1.5: a Software Framework for Automatic Edit and Imputation. Working Paper, UN/ECE Work Session on Statistical Data Editing, Ottawa.

De Waal, T. and Coutinho, W. (2005), Automatic Editing for Business Surveys: an Assessment for Selected Algorithms. *International Statistical Review* **73**, 73–102.

De Waal, T., Pannekoek, J., and Scholtus, S. (2011), *Handbook of Statistical Data Editing and Imputation*. John Wiley & Sons, New Jersey.

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

Di Zio, M., Guarnera, U., and Luzi, O. (2005), Improving the Effectiveness of a Probabilistic Editing Strategy for Business Data. Report, ISTAT, Rome.

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

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

Freund, R. J. and Hartley, H. O. (1967), A Procedure for Automatic Data Editing. *Journal of the American Statistical Association* **62**, 341–352.

Garfinkel, R. S., Kunnathur, A. S., and Liepins, G. E. (1988), Error Localization for Erroneous Data: Continuous Data, Linear Constraints. *SIAM Journal on Scientific and Statistical Computing* **9**, 922–931.

Ghosh-Dastidar, B. and Schafer, J. L. (2003), Multiple Edit/Multiple Imputation for Multivariate Continuous Data. *Journal of the American Statistical Association* **98**, 807–817.

Hoogland, J. and Smit, R. (2008), Selective Automatic Editing of Mixed Mode Questionnaires for Structural Business Statistics. Working Paper, UN/ECE Work Session on Statistical Data Editing, Vienna.

Kovar, J. and Whitridge, P. (1990), Generalized Edit and Imputation System; Overview and Applications. *Revista Brasileira de Estadistica* **51**, 85–100.

Little, R. J. A. and Smith, P. J. (1987), Editing and Imputation of Quantitative Survey Data. *Journal of the American Statistical Association* **82**, 58–68.

Nemhauser, G. L. and Wolsey, L. A. (1988), *Integer and Combinatorial Optimization*. John Wiley & Sons, New York.

Nordbotten, S. (1963), Automatic Editing of Individual Statistical Observations. In: *Conference of European Statisticians Statistical Standards and Studies No. 2*, United Nations, New York.

Pannekoek, J. and de Waal, T. (2005), Automatic Edit and Imputation for Business Surveys: the Dutch Contribution to the EUREDIT Project. *Journal of Official Statistics* **21**, 257–286.

Ragsdale, C. T. and McKeown, P. G. (1996), On Solving the Continuous Data Editing Problem. *Computers & Operations Research* **23**, 263–273.

Riera-Ledesma, J. and Salazar-González, J. J. (2003), New Algorithms for the Editing and Imputation Problem. Working Paper, UN/ECE Work Session on Statistical Data Editing, Madrid.

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

Scholtus, S. (2013), Automatic Editing with Hard and Soft Edits. *Survey Methodology* **39**, 59–89.

Todero, T. A. (1999), Overview and Evaluation of the AGGIES Automated Edit and Imputation System. Working Paper, UN/ECE Work Session on Statistical Data Editing, Rome.

Williams, H. P. (1986), Fourier's Method of Linear Programming and Its Dual. *The American Mathematical Monthly* **93**, 681–695.

Winkler, W. E. and Draper, L. A. (1997), The SPEER Edit System. In: *Statistical Data Editing*, Volume 2: *Methods and Techniques*, United Nations, Geneva.

Specific section

8. Purpose of the method

Localising errors in microdata without human intervention

9. Recommended use of the method

1. The method should be used for error localisation in microdata containing only random errors. Any systematic errors that may occur in the original microdata have to be resolved beforehand, using deductive editing methods (see the method module “Statistical Data Editing – Deductive Editing”).
2. If it is known beforehand that certain variables contain more errors than others, then this information should be included in the form of reliability weights (see item 14).
3. The quality of the error localisation strongly depends on the specification of the edit rules. The set of edit rules should be sufficiently powerful to detect the majority of errors, but not so strict that the method results in overedited data.

10. Possible disadvantages of the method

1. In general, it is not possible to construct a set of edit rules that always leads to the correct solution. Thus, the edited data may still contain some errors, although the edited records are consistent with the edit rules. For this reason, automatic editing should not be applied to crucial records, e.g., records belonging to very large businesses. In addition, the quality of automatic editing is lower for records that contain many errors. Both disadvantages can be circumvented by always using automatic editing in combination with a form of selective editing. We refer to “Statistical Data Editing – Main Module” for a discussion on how to incorporate automatic editing in an overall editing strategy.

11. Variants of the method

1. The original method of Fellegi and Holt as described in Section 2.4.
2. Other methods as described in Section 2.5. These methods find the same solution as the original method of Fellegi and Holt, but they use different search algorithms. Examples include:
   - 2.1 Algorithms based on vertex generation;
   - 2.2 Algorithms based on branch-and-bound;
   - 2.3 Algorithms based on cutting planes;
   - 2.4 Algorithms based on (mixed) integer programming.

12. Input data

1. A data set containing unedited microdata.

13. Logical preconditions

1. Missing values
    1. Allowed; they will be considered as erroneously missing, i.e., available for imputation.
2. Erroneous values
    1. Allowed; in fact, the object of this method is to decide which values in a record are erroneous.
    2. It is assumed that the data contain only random errors; systematic errors should be removed beforehand by means of deductive editing.
3. Other quality related preconditions
    1. n/a
4. Other types of preconditions
    1. It is assumed that all edit rules may be interpreted as hard edit rules.

14. Tuning parameters

1. A collection of edit rules for the microdata at hand.
2. A set of reliability weights may be provided for the variables in the data set. By default, all reliability weights are equal to 1.
3. A maximum number of variables to impute may be set to reduce the computational workload. The error localisation problem will not be solved for records that cannot be imputed consistently by changing at most the specified maximum number of variables.

15. Recommended use of the individual variants of the method

1. For variant 1 (the original method of Fellegi and Holt), most of the work lies in the generation of a complete set of edits. Once this complete set is available, the error localisation problem can be solved for any record in a straightforward manner. If the complete set of edits is too large to be generated, this variant of the method cannot be used.
2. For the other variants, the work lies in solving a separate error localisation problem for each individual record. In this case, it is usually necessary to specify a maximum number of variables to impute (see item 14), unless the data set contains few variables (say less than 10).

16. Output data

1. For each record in the microdata, the method attempts to yield a list of variables that can be imputed to obtain a consistent record with respect to the edit rules. For some records, the method may not return such a list, because it could not find a feasible solution to the error localisation problem.

17. Properties of the output data

1. For each record for which the method returns a solution, the variables listed in the solution can be imputed so that the resulting record is consistent with respect to the edit rules. Moreover, they constitute the smallest (weighted) set of variables that has this property.
2. The original values of the variables that are listed for imputation have to be considered as erroneous in all further processing. The natural next step is to impute new values for these variables by means of some imputation method. It should be noted that the imputation step is not a part of the error localisation method itself.
3. For some records, the method may not find a solution. These records have to be processed interactively by subject-matter experts (see the method module "Statistical Data Editing – Manual Editing").

18. Unit of input data suitable for the method

Incremental processing by record

19. User interaction - not tool specific

1. Ideally, there is no user interaction other than setting parameters and reading in input data at the beginning, and processing output data at the end.

20. Logging indicators

1. The number of records for which the method found/did not find a solution.
2. The computing time per record.

21. Quality indicators of the output data

1. The quality of automatic editing can be assessed in a simulation study. This requires a data set that has been interactively edited by experts to a point where the edited data may be considered error-free. In the simulation study, the original data are edited again using automatic editing. The quality of automatic editing may then be measured in terms of the similarity of the automatically edited data to the interactively edited data.

22. Actual use of the method

1. The method is used at Statistics Netherlands in the production process for structural business statistics. This application uses the tool SLICE, which contains an implementation of the branch-and-bound algorithm of De Waal and Quere (2003). See Hoogland and Smit (2008) for more details.

Interconnections with other modules

23. Themes that refer explicitly to this module

1. Micro-Fusion – Main Module
2. Statistical Data Editing – Main Module

3. Statistical Data Editing – Selective Editing
4. Imputation – Main Module
5. Macro-Integration – Main Module

24. Related methods described in other modules

1. Micro-Fusion – Reconciling Conflicting Microdata
2. Statistical Data Editing – Deductive Editing
3. Statistical Data Editing – Manual Editing

25. Mathematical techniques used by the method described in this module

1. Fourier-Motzkin elimination

26. GSBPM phases where the method described in this module is used

1. GSBPM Sub-process 5.3: Review, validate and edit

27. Tools that implement the method described in this module

1. GEIS
2. Banff
3. CherryPi
4. AGGIES
5. SLICE
6. R package `editrules`

28. Process step performed by the method

Statistical data editing

Administrative section

29. Module code

Statistical Data Editing-M-Automatic Editing

30. Version history

Version	Date	Description of changes	Author	Institute
0.1	20-12-2011	first version	Sander Scholtus	CBS (Netherlands)
0.2	20-04-2012	improvements based on Swedish review	Sander Scholtus	CBS (Netherlands)
0.2.1	16-07-2013	adjusted to new template; minor improvements	Sander Scholtus	CBS (Netherlands)
0.3	04-09-2013	minor improvements	Sander Scholtus	CBS (Netherlands)
0.3.1	09-09-2013	preliminary release		
1.0	26-03-2014	final version within the Memobust project		

31. Template version and print date

Template version used	1.0 p 4 d.d. 22-11-2012
Print date	21-3-2014 18:11