Journal of Official Statistics, Vol. 27, No. 3, 2011, pp. 467–490

Algorithms for Correcting Sign Errors and Rounding Errors in Business Survey Data

Sander Scholtus[1]

Selective editing is often used for the data of structural business surveys. Records containing potentially influential errors are edited manually, whereas the other, noncritical records can be edited automatically. At Statistics Netherlands, the automatic editing is performed by an advanced software package called SLICE. Prior to this several types of obvious inconsistencies are detected and corrected deductively. This article describes two additional types of frequently occurring obvious inconsistencies, sign errors and rounding errors. Simple algorithms are given that detect and correct these errors. Correction of these errors in a separate step will increase the efficiency of the subsequent editing process, because more records will be eligible (and suitable) for automatic editing. By way of illustration, the algorithms are applied to real data from the Dutch structural business survey.

Key words: Structural business statistics; automatic editing; deductive correction; sign errors; rounding errors.

1. Introduction

It is well-known that data collected in a survey or register contains errors. In the case of a survey, these errors may be introduced when the respondent fills in the questionnaire or during the processing of survey forms at the statistical office. It is important to resolve the errors by editing the data, because figures based on erroneous data may be biased or logically inconsistent. For the structural business statistics, all survey variables are quantitative and many (linear) relationships between them can be formulated. Thus, a set of constraints called *edit rules* is established. Two examples of edit rules are

$$\textit{profit} = \textit{turnover} - \textit{costs}$$

and

$$\textit{number of employees}\,(\text{in persons}) \geq \textit{number of employees}\,(\text{in full time equivalent (fte)})$$

If the data in a particular record violates an edit rule, the record is found to be inconsistent and it is deduced that some variable(s) must be in error.

[1] Statistics Netherlands, P.O. Box 24500, 2490 HA The Hague, The Netherlands. Email: sshs@cbs.nl

Acknowledgments: The author wants to thank Jacco Daalmans, Bram Duyx, Jeroen Pannekoek, Caren Tempelman and Ton de Waal at Statistics Netherlands for many helpful contributions and suggestions. He also thanks the Associate Editor and four anonymous referees for their comments on the first draft, which have greatly improved this article. The views expressed in this article are those of the author and do not necessarily reflect the policies of Statistics Netherlands.

A distinction is often made between *systematic errors* and *random errors*. According to Eurostat (2007, §3.3.1), an error is systematic if it is reported consistently over time by different respondents. This type of error occurs when respondents consistently misunderstand a survey question, e.g. by reporting financial amounts in Euros rather than the requested multiples of 1,000 Euros (this example is called a *unity measure error*, cf. Di Zio et al. 2005). A structural fault in the data processing system might also introduce systematic errors. Since it is reported consistently by a number of respondents, an undiscovered systematic error can lead to biased aggregates. Once identified, a systematic error can be corrected deductively, because the underlying error mechanism is assumed to be known. Random errors on the other hand do not have a structural cause. An example of a random error occurs when a particular "1" on a particular survey form is accidentally keyed in as a "7" during data processing.

At Statistics Netherlands, *selective editing* is used to clean the data collected for structural business statistics (De Jong 2002). This means that only records containing potentially influential errors are edited manually by subject-matter specialists, whereas the remaining records are edited automatically. For the latter step, many statistical institutes have implemented error localisation algorithms based on a generalisation of the Fellegi-Holt paradigm (Fellegi and Holt 1976), which states that the smallest possible (weighted) number of variables should be labelled erroneous such that the record can be made consistent with every edit rule. This paradigm is based on the assumption that the data contains only random errors.

Examples of software packages for automatic editing based on the Fellegi-Holt paradigm are: GEIS (see Kovar and Withridge 1990) and its successor Banff (see Banff Support Team 2003), SPEER (see Winkler and Draper 1997), DISCRETE (see Winkler and Petkunas 1997) and AGGIES (see Todaro 1999). At Statistics Netherlands, the software package SLICE was developed for automatic editing. SLICE also uses an error localisation algorithm based on the Fellegi-Holt paradigm; a description of this algorithm can be found in De Waal and Quere (2003) and De Waal (2003).

A *plausibility indicator* is calculated for each record to assess whether it may contain influential errors and should be edited manually (Hoogland 2006). The plausibility indicator is calibrated such that all records that receive a score above a certain threshold are deemed suitable for automatic editing. Only the records with the lowest scores on the plausibility indicator are edited manually. In addition to this, the data of very large companies is always edited manually, since it is considered impossible to construct meaningful aggregates unless this part of the data set is error-free.

Selective editing leads to a more efficient editing process than traditional editing (where every record is edited by hand), because part of the data stream is not reviewed by subject-matter specialists any more. However, Fellegi-Holt-based algorithms for automatic error localisation are not considered suitable for editing records that contain either influential or systematic errors. In particular, the correction of systematic errors often requires changing more variables than the Fellegi-Holt paradigm suggests. For instance, a unity measure error affects all financial variables on the survey form, but it leads to few violated edit rules. Furthermore, in practice the automatic error localisation problem becomes too complicated if many variables contain erroneous values and/or if many edit rules are violated (De Waal and Quere 2003). For the Dutch structural business survey, because of the large number of edit rules involved, the error localisation problem tends to be too complex to solve with SLICE if more than, say about 15 variables have to be changed.

To preserve the quality of the statistical output, only records that contain a limited number of noninfluential random errors should be edited automatically. Ideally, the plausibility indicator filters out all records containing influential errors or too many inconsistencies. Prior to this, several types of *obvious errors* can be detected and resolved automatically in a separate step. A systematic error is called obvious if it can be detected “easily”, i.e. by applying some basic, specific search algorithm. Obvious errors are easy to correct, because the underlying cause of the error is detectable. An example of such an error is the unity measure error, which can be detected by comparing the reported amounts with reference values (see Section 2).

It is useful to detect and correct obvious inconsistencies as early as possible in the editing process, since it is a waste of resources if subject-matter specialists have to deal with them. When obvious inconsistencies are corrected in a separate step, before the plausibility indicator is calculated, the efficiency of the selective editing process increases because more records will be eligible for automatic editing. Moreover, solving the error localisation problem becomes easier once obvious inconsistencies have been removed, since the number of violated edit rules becomes smaller.

Furthermore, since obvious inconsistencies are systematic errors, they can be corrected more accurately by a specific, deductive algorithm than by a general error localisation algorithm based on the Fellegi-Holt paradigm. The deductive algorithm uses knowledge of the underlying cause of the error, so that the corrected values are true values, assuming that the error has been detected correctly. By contrast, the Fellegi-Holt-based algorithm does not use this knowledge, and the values returned by this algorithm are consistent with respect to the edit rules but are not necessarily true values. Hence, if a certain type of systematic error is expected to occur commonly and if a specific, reliable search routine is available to detect and correct it, it makes sense to apply this routine rather than to rely on the general algorithm used by SLICE. After all, if the error is left in the data to be resolved by SLICE, at best the general algorithm will detect and correct the error the same way the simple algorithm would have done, but at a much higher computational cost.

The currently implemented editing process for the structural business statistics at Statistics Netherlands contains a step during which three obvious systematic errors are treated. Section 2 provides a brief description of this step. Other obvious inconsistencies have been discovered by comparing raw and manually edited data from past cycles of the structural business survey. This study has resulted in several new deductive correction methods (Scholtus 2008; 2009).

The purpose of this article is to present new algorithms for the detection and correction of two types of errors. Section 3 deals with so-called sign errors and interchanged revenues and costs. Section 4 describes a heuristic method for correcting rounding errors. Rounding errors are not obvious inconsistencies in the true sense of the word (they can be considered as random errors), but the efficiency of the editing process is expected to increase if these errors are also treated separately. Section 5 presents some results of an application of the two algorithms to real-world data. Finally, a few concluding remarks follow in Section 6.

Due to item nonresponse, the unedited data contains a substantial number of missing values. The algorithms described in this article assume that these missing values have been temporarily replaced with zeros. This is merely a precondition for determining which edit rules are violated and which are satisfied, and should not be considered a full imputation. When the obvious inconsistencies have been corrected, all placeholder zeros should be replaced by missing values again, to be imputed by a valid method later. Clearly, this requires that placeholder zeros can be distinguished from actually reported zeros.

2. Current Approach at Statistics Netherlands

The currently implemented editing process for structural business statistics at Statistics Netherlands contains a step in which three kinds of obvious systematic errors are detected. These errors are treated deductively before any other correction is made in the data of the processed survey forms.

The first of these obvious inconsistencies is the unity measure error from Section 1: the amounts on the survey form are sometimes reported in Euros instead of in 1,000 Euros. This particular unity measure error is also referred to as a *uniform* 1,000-*error*. It is important to detect this error because otherwise publication figures of all financial items will be overestimated. Depending on which auxiliary information is available, two methods are used to detect uniform 1,000-errors. If the respondent is present in the VAT register, the amount of turnover in the register is compared to the reported turnover in the survey. For the other respondents, the amount of reported turnover per reported number of employees (in fte) is compared to its median in the edited data of the previous year. If a large discrepancy is found by either method, all financial amounts reported by the respondent are divided by 1,000. This is how uniform 1,000-errors are currently detected at Statistics Netherlands. Different methods are suggested by Di Zio et al. (2005) and Al-Hamad et al. (2008).

The second obvious inconsistency occurs when a respondent adds a redundant minus sign to a reported value. This sometimes happens with variables that have to be subtracted, even though there already is a printed minus sign on the survey form. As a result, the value of the variable becomes incorrectly negative after data processing. The resulting inconsistency can be detected and corrected easily: the reported amount is simply replaced by its absolute value.

The third and final obvious inconsistency occurs when respondents report component items of a sum but leave the corresponding total blank. When this is detected, the total value is calculated from the reported items and filled in automatically.

3. Sign Errors

3.1. The Profit-and-loss Account

The *profit-and-loss account* is a part of the questionnaire used for structural business statistics where the respondent has to fill in a number of balance amounts. These balance variables are denoted by $x_0, x_1, \ldots, x_{n-1}$. A final balance amount x_n called the *pretax results* is found by adding up the other balance variables. That is, the data should conform to the following edit rule:

$$x_0 + x_1 + \cdots + x_{n-1} = x_n \tag{3.1}$$

Rule (3.1) is sometimes referred to as the *external sum*. A balance variable is defined as the difference between a revenue item and a cost item. If these items are also asked in the questionnaire, the following edit rule should hold:

$$x_{k,r} - x_{k,c} = x_k \tag{3.2}$$

where $x_{k,r}$ denotes the revenue item and $x_{k,c}$ the cost item. Rules of this form are referred to as *internal sums*.

A statistical office may decide not to ask the revenues and costs for every balance variable in the survey, to limit the burden on respondents. To keep the notation simple but sufficiently general, it is assumed that the balance variables are arranged such that only $x_0, x_1, \ldots, x_m$ are split into revenues and costs, for some $m \in \{0, 1, \ldots, n-1\}$. Thus, the following set of edit rules is used:

$$\begin{cases} x_0 = x_{0,r} - x_{0,c} \\ \quad\vdots \\ x_m = x_{m,r} - x_{m,c} \\ x_n = x_0 + x_1 + \cdots + x_{n-1} \end{cases} \tag{3.3}$$

In this notation the 0th balance variable x_0 stands for *operating results*, and $x_{0,r}$ and $x_{0,c}$ represent *operating returns* and *operating costs*, respectively.

3.2. Sign Errors and Interchanged Revenues and Costs

Table 1 displays the structure of the profit-and-loss account from the structural business statistics questionnaire that was used at Statistics Netherlands until 2005. The associated edit rules are given by (3.3), with $n = 4$ and $m = n - 1 = 3$. Table 1 also displays four example records that are inconsistent. The first three example records have been

Table 1. Structure of the profit-and-loss account in the structural business statistics until 2005, with four example records, (a) – (d)

Variable	Full name	(a)	(b)	(c)	(d)
$x_{0,r}$	Operating returns	2,100	5,100	3,250	5,726
$x_{0,c}$	Operating costs	1,950	4,650	3,550	5,449
x_0	Operating results	150	450	300	276
$x_{1,r}$	Financial revenues	0	0	110	17
$x_{1,c}$	Financial expenditure	10	130	10	26
x_1	Financial result	10	130	100	10
$x_{2,r}$	Provisions rescinded	20	20	50	0
$x_{2,c}$	Provisions added	5	0	90	46
x_2	Balance of provisions	15	20	40	46
$x_{3,r}$	Exceptional income	50	15	30	0
$x_{3,c}$	Exceptional expenses	10	25	10	0
x_3	Exceptional result	40	10	20	0
x_4	Pretax results	195	610	− 140	221

constructed for this article with nice "round" amounts to improve readability, but the types of inconsistencies present were taken from actual records from the structural business statistics of 2001. The fourth example record contains realistic values.

In Example (a) two edit rules are violated: the external sum and the internal sum with $k = 1$. In this case, the profit-and-loss account can be made fully consistent with all edit rules by just changing the value of x_1 from 10 to -10 (see Table 2). This is the natural way to obtain a consistent profit-and-loss account here, since any other explanation would require more variables to be changed. Moreover, it is quite conceivable that the minus sign in x_1 was left out by the respondent or "lost" during data processing.

Two internal sums are violated in example (b), but the external sum holds. The natural way to obtain a consistent profit-and-loss account here is by interchanging the values of $x_{1,r}$ and $x_{1,c}$, and also of $x_{3,r}$ and $x_{3,c}$ (see Table 2). By treating the inconsistencies this way, full use is made of the amounts actually filled in by the respondent and no imputation of synthetic values is necessary.

The two types of errors found in Examples (a) and (b) are referred to as *sign errors* and *interchanged revenues and costs*, respectively. For the sake of brevity, the term sign error is also used to refer to both types. In an evaluation study at Statistics Netherlands, which compared raw data to manually edited data, it was found that a substantial number of respondents made these errors. Moreover, it was found that these errors were often not correctly identified by SLICE and hence resolved in a different way during automatic editing. In particular, it is very difficult to handle interchanged revenues and costs correctly by means of the Fellegi-Holt paradigm, because this requires changing two variables where it would actually suffice to change one. Therefore, it seems advantageous to add a separate detection step for sign errors at the beginning of the automatic editing process.

Sign errors and interchanged revenues and costs are closely related and should therefore be searched for by one detection algorithm. In the remainder of this section such an algorithm is formulated, working from the assumption that if an inconsistent record can be made to satisfy all edit rules in (3.3) by only changing signs of balance variables and/or

Table 2. Corrected versions of the example records from Table 1. Changes are shown in boldface

Variable	Full name	(a)	(b)	(c)	(d)
$x_{0,r}$	Operating returns	2,100	5,100	3,250	5,726
$x_{0,c}$	Operating costs	1,950	4,650	3,550	5,449
x_0	Operating results	150	450	**−300**	276
$x_{1,r}$	Financial revenues	0	**130**	110	17
$x_{1,c}$	Financial expenditure	10	**0**	10	26
x_1	Financial result	**−10**	130	100	**−10**
$x_{2,r}$	Provisions rescinded	20	20	**90**	0
$x_{2,c}$	Provisions added	5	0	**50**	46
x_2	Balance of provisions	15	20	40	**−46**
$x_{3,r}$	Exceptional income	50	**25**	30	0
$x_{3,c}$	Exceptional expenses	10	**15**	10	0
x_3	Exceptional result	40	10	20	0
x_4	Pretax results	195	610	−140	221

interchanging revenue items and cost items, this is indeed the way the record should be corrected.

It should be noted that *operating returns* ($x_{0,r}$) and *operating costs* ($x_{0,c}$) differ from the other variables in the profit-and-loss account in the sense that they are also present in other edit rules, connecting them to items from other parts of the survey. For instance, *operating costs* should equal the sum of *total labour costs*, *total machine costs*, etc. If $x_{0,r}$ and $x_{0,c}$ were interchanged to suit the 0th internal sum, other edit rules might be violated. It is therefore not allowed to interchange $x_{0,r}$ and $x_{0,c}$ when detecting sign errors. Because of the way the questionnaire is designed, it seems highly unlikely that any respondent would mix up these two amounts anyway.

As stated above, a record contains a sign error if it satisfies the following two conditions:

- at least one edit rule in (3.3) is violated;
- it is possible to satisfy (3.3) by only changing the signs of balance amounts and/or interchanging revenue and cost items other than $x_{0,r}$ and $x_{0,c}$.

An equivalent way of formulating this is to say that an inconsistent record contains a sign error if the following set of equations has a solution:

$$\begin{cases} x_0 s_0 = x_{0,r} - x_{0,c} \\ x_1 s_1 = (x_{1,r} - x_{1,c}) t_1 \\ \qquad \vdots \\ x_m s_m = (x_{m,r} - x_{m,c}) t_m \\ x_n s_n \ = x_0 s_0 + x_1 s_1 + \cdots + x_{n-1} s_{n-1} \\ (s_0, \ldots, s_n; t_1, \ldots, t_m) \in \{-1, 1\}^{n+m+1} \end{cases} \tag{3.4}$$

Note that in (3.4) the x's are used as known constants rather than unknown variables. Thus, a different set of equations in $(s_0, \ldots, s_n; t_1, \ldots, t_m)$ is found for each record.

Moreover, once a solution to (3.4) has been found, it is immediately clear how to obtain a consistent profit-and-loss account: if $s_j = -1$ then the sign of x_j must be changed, and if $t_k = -1$ then the values of $x_{k,r}$ and $x_{k,c}$ must be interchanged. It is easy to see that the resulting record satisfies all edit rules (3.3). Since $x_{0,r}$ and $x_{0,c}$ may not be interchanged, no variable t_0 is present in (3.4).

Example Consider (3.4) for Example (c) from Table 1:

$$\begin{cases} 300 s_0 = -300 \\ 100 s_1 = 100 t_1 \\ 40 s_2 = -40 t_2 \\ 20 s_3 = 20 t_3 \\ -140 s_4 = 300 s_0 + 100 s_1 + 40 s_2 + 20 s_3 \\ (s_0, s_1, s_2, s_3, s_4; t_1, t_2, t_3) \in \{-1, 1\}^8 \end{cases} \tag{3.5}$$

This system has the (unique) solution $(s_0 = -1, s_1 = 1, s_2 = 1, s_3 = 1, s_4 = 1; t_1 = 1, t_2 = -1, t_3 = 1)$. This solution shows that the value of x_0 should be changed from 300 to -300 and that the values of $x_{2,r}$ and $x_{2,c}$ should be interchanged. This correction indeed yields a fully consistent profit-and-loss account with respect to (3.3), as can be seen in Table 2. ▲

An important question is: does system (3.4) always have a unique solution? Scholtus (2008) derives the following sufficient condition for uniqueness: if $x_0 \neq 0$, $x_n \neq 0$, and if the equation

$$\lambda_0 x_0 + \lambda_1 x_1 + \cdots + \lambda_{n-1} x_{n-1} = 0$$

does not have any solution $(\lambda_0, \lambda_1, \ldots, \lambda_{n-1}) \in \{-1, 0, 1\}^n$ for which at least one term $\lambda_j x_j \neq 0$, then an inconsistency in the record can be resolved by changing signs and/or interchanging revenues and costs in at most one way. It appears that this condition is usually satisfied; in the data examined at Statistics Netherlands, the condition holds for over 95 per cent of all records. In the rare case that system (3.4) has more than one solution, it makes sense to assume that most of the original values were reported correctly by the respondent and therefore choose the solution with the smallest number of -1's among s_j and t_k. This assumption will indeed be made in the next subsection.

3.3. A Binary Linear Programming Problem

Detecting a sign error in a given record is equivalent to solving the corresponding system (3.4). Therefore, all that is needed to implement the detection of sign errors is a systematic method to solve this system. Before addressing this point, it is convenient to write (3.4) in matrix notation to shorten the expressions. Define the $(m+2) \times (n+1)$-matrix U by

$$U = \begin{bmatrix} x_0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 \\ 0 & x_1 & \cdots & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & x_m & 0 & \cdots & 0 & 0 \\ x_0 & x_1 & \cdots & x_m & x_{m+1} & \cdots & x_{n-1} & -x_n \end{bmatrix}$$

and define the $(m+2) \times (m+1)$-matrix V by

$$V = \begin{bmatrix} x_{0,r} - x_{0,c} & 0 & \cdots & 0 \\ 0 & x_{1,r} - x_{1,c} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & x_{m,r} - x_{m,c} \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

Note that the bottom row of V consists entirely of zeros. Moreover, define $\mathbf{s} = (s_0, s_1, \ldots, s_n)'$ and $\mathbf{t} = (1, t_1, \ldots, t_m)'$. Using this notation, (3.4) can be rewritten as:

$$\begin{cases} U\mathbf{s} - V\mathbf{t} = \mathbf{0}, \\ \mathbf{s} \in \{-1, 1\}^{n+1}, \\ \mathbf{t} \in \{1\} \times \{-1, 1\}^m, \end{cases} \tag{3.6}$$

where $\mathbf{0}$ denotes the $(m+2)$-vector of zeros.

The least sophisticated way of finding a solution to (3.6) would be to simply try all possible vectors $\mathbf{s}$ and $\mathbf{t}$. Since m and n are small in this situation, the number of possibilities is not very large and this approach is actually quite feasible. However, it is also possible to reformulate the problem as a so-called binary linear programming problem. This has the advantage that standard software may be used to implement the method. Moreover, it will be seen presently that this formulation can be adapted easily to accommodate possible rounding errors present in the data.

The following binary variables are introduced to reformulate the problem:

$$\sigma_j = \frac{1 - s_j}{2}, \quad j \in \{0, 1, \ldots, n\}$$

$$\tau_k = \frac{1 - t_k}{2}, \quad k \in \{1, \ldots, m\}$$

Finding an optimal solution to (3.6) may be restated as follows:

$$\begin{aligned} &\min \sum_{j=0}^{n} \sigma_j + \sum_{k=1}^{m} \tau_k \\ &\text{such that :} \\ &U(\mathbf{1} - 2\boldsymbol{\sigma}) - V(\mathbf{1} - 2\boldsymbol{\tau}) = \mathbf{0}, \\ &\boldsymbol{\sigma} \in \{0, 1\}^{n+1}, \boldsymbol{\tau} \in \{0\} \times \{0, 1\}^m, \end{aligned} \tag{3.7}$$

where $\mathbf{1}$ is a vector of ones, $\boldsymbol{\sigma} = (\sigma_0, \sigma_1, \ldots, \sigma_n)'$ and $\boldsymbol{\tau} = (0, \tau_1, \ldots, \tau_m)'$.

Observe that in this formulation the number of variables s_j and t_k that are equal to -1 is minimised, i.e., the solution is searched for that results in the smallest number of changes being made in the record. Obviously, if a unique solution to (3.6) exists, then this is also the solution to (3.7). The binary linear programming problem may be solved by applying a standard branch and bound algorithm. Since n and m are small, very little computation time is needed to find the solution.

3.4. *Allowing for Rounding Errors*

It often happens that balance edit rules are violated by a very small difference. For instance, a reported total value is just one or two units smaller or larger than the sum of the reported item values. These inconsistencies are called *rounding errors* if the absolute difference is no larger than δ units. In the examples in this article, δ is chosen equal to 2. In the profit-and-loss account, rounding errors can occur in two ways. Firstly the pretax results may differ slightly from the sum of the balance amounts (a rounding error in the external sum), and secondly a balance amount may just disagree with the difference between the reported revenue and cost items (a rounding error in an internal sum).

Rounding errors often occur in conjunction with other errors. In particular, a record might contain a sign error that is obscured by a rounding error. Column (d) in Table 1 shows an example of such a record. If the method described in the previous subsection is applied directly, the sign error will not be detected.

Fortunately, the binary linear programming problem (3.7) can be adapted to take the possibility of rounding errors into account. This leads to the following problem:

$$\begin{aligned}
&\min \sum\nolimits_{j=0}^{n} \sigma_j + \sum\nolimits_{k=1}^{m} \tau_k \\
&\text{such that :} \\
&-\boldsymbol{\delta} \leq U(\mathbf{1} - 2\boldsymbol{\sigma}) - V(\mathbf{1} - 2\boldsymbol{\tau}) \leq \boldsymbol{\delta}, \\
&\boldsymbol{\sigma} \in \{0,1\}^{n+1}, \boldsymbol{\tau} \in \{0\} \times \{0,1\}^{m},
\end{aligned} \tag{3.8}$$

where $\boldsymbol{\delta}$ is a vector of δ's and the rest of the notation is obtained as before. Problem (3.7) is obtained by taking $\delta = 0$, i.e., by assuming that no rounding errors occur.

Example If (3.8) is set up for Example (d) from Table 1, with $\delta = 2$, the following solution is found: $(\sigma_0 = 0, \sigma_1 = 1, \sigma_2 = 1, \sigma_3 = 0, \sigma_4 = 0; \tau_1 = 0, \tau_2 = 0, \tau_3 = 0)$. Recalling that $\sigma_j = 1$ if and only if $s_j = -1$ (and a similar expression for τ_k and t_k), the sign error may be removed by changing the signs of both x_1 and x_2. As can be seen in Table 2, this correction indeed eliminates the sign error. It does not lead to a fully consistent profit-and-loss account, however, because there are rounding errors left in the data. To remove these, a separate method is needed. This problem will be discussed in Section 4. ▲

3.5. Summary

The following plan summarises the correction method for sign errors and interchanged revenues and costs. The input consists of a record that does not satisfy (3.3) and a choice for δ.

1. Determine the matrices U and V and set up the binary linear programming problem (3.8).
2. Solve (3.8). If no solution is possible, then the record does not contain a sign error. If a solution is found, continue.
3. Replace x_j by $-x_j$ for every $\sigma_j = 1$ and interchange $x_{k,r}$ and $x_{k,c}$ for every $\tau_k = 1$.

If Step 3 is performed, the resulting record satisfies (3.3) barring possible rounding errors.

4. Rounding Errors

4.1. Introduction

It was mentioned in the previous section that very small inconsistencies with respect to balance edit rules often occur, e.g. a total value is just one unit smaller or larger than the sum of the component items. Such inconsistencies are called rounding errors, because they may be caused by values being rounded off to multiples of 1,000. It is not straightforward to obtain a so-called *consistent rounding*, i.e., to make sure that the rounded-off values have the same relation as the original values. For example, if the terms of the sum $2.7 + 7.6 = 10.3$ are rounded off to natural numbers the ordinary way, then the additivity is destroyed: $3 + 8 \neq 10$. Several algorithms for consistent rounding are available in the literature; see e.g. Salazar-González et al. (2004). Obviously, very few respondents are even aware of these methods, let alone inclined to use them while filling in a questionnaire.

By their nature, rounding errors have virtually no influence on aggregates, and in this sense the choice of method to correct them is unimportant. However, as mentioned in Section 1, the complexity of the automatic error localisation problem in SLICE increases rapidly as the number of violated edit rules becomes larger, irrespective of the magnitude of the violations. Thus, a record containing many rounding errors and few "real" errors might not be suitable for automatic editing by means of a Fellegi-Holt-based approach and might have to be edited manually. This is clearly a waste of resources. It is therefore advantageous to resolve all rounding errors in the early stages of the editing process, for instance immediately after the correction of obvious inconsistencies. Given the uninfluential nature of rounding errors, it might seem like a good approach to not correct them at all during automatic editing. Using SLICE, the only way to achieve this is by replacing each balance edit rule by two inequality edit rules that bound the difference of the total amount and its items between, say, -2 and 2. Unfortunately, this would make the automatic editing much more computationally demanding, because the error localisation algorithm of SLICE can handle equalities more efficiently than inequalities (De Waal and Quere 2003). On balance, it is actually more efficient to handle rounding errors in a separate step.

In the remainder of this section, a heuristic method is described to resolve rounding errors in business survey data. This method is called a heuristic method because it does not return a solution that is "optimal" in some sense, e.g. that the number of changed variables or the total change in values is minimised. The rationale for using such a method is that the adaptations needed to resolve rounding errors are very small, and that it is therefore not necessary to use a sophisticated and potentially time-consuming search algorithm.

Although the idea behind the method is quite simple, some results from matrix algebra are needed to explain why it works. The necessary background will be briefly summarised in Section 4.2.

4.2. *Matrix Theory*

Recall that Cramer's Rule is a theorem named after the Swiss mathematician Gabriel Cramer (1704–1752) which states the following. Let $A = [a_{ij}]$ be an invertible $p \times p$-matrix. The unique solution $\mathbf{x} = [x_1, \ldots, x_p]'$ to the system $A\mathbf{x} = \mathbf{b}$ is given by:

$$x_k = \frac{\det B_k}{\det A}, \quad k = 1, \ldots, p,$$

where B_k denotes the matrix found by replacing the kth column of A by $\mathbf{b}$. An alternative way of formulating this, is that for any invertible matrix A,

$$A^{-1} = \frac{1}{\det A} \mathrm{A}^{\dagger}, \tag{4.1}$$

where $A^{\dagger}$ denotes the *adjoint matrix* of A. The adjoint matrix is found by transposing the matrix of cofactors: $(A^{\dagger})_{ji} = (-1)^{i+j}\det C_{ij}$, where C_{ij} is the matrix A with the ith row and the jth column removed. For a proof of (4.1), see e.g., Harville (1997, Section 13.5).

A square matrix is called *unimodular* if its determinant is equal to 1 or -1. The following property is an immediate consequence of Cramer's Rule.

Property 4.1 If A is an integer-valued unimodular matrix and $\mathbf{b}$ is an integer-valued vector, then the solution to the system $A\mathbf{x} = \mathbf{b}$ is also integer-valued.

A (not necessarily square) matrix for which the determinant of every square submatrix is equal to 0, 1 or -1 is called *totally unimodular*. That is to say, every square submatrix of a totally unimodular matrix is either singular or unimodular. Clearly, in order to be totally unimodular, a matrix must have all elements equal to 0, 1 or -1. A stronger version of Property 4.1 can be proved for the submatrices of a totally unimodular matrix.

Property 4.2 Let B be a square submatrix of a totally unimodular matrix. If B is invertible, all elements of B^{-1} are in $\{-1, 0, 1\}$.

Proof This is easily seen using the adjoint matrix $B^{\dagger}$. Since $|\det B| = 1$ and all cofactors are equal to 0, 1 or -1, the property follows immediately from Equation (4.1). ■

Verifying whether a matrix is totally unimodular by directly applying the definition is usually impossible – unless the matrix happens to be very small – because the number of determinants to evaluate is simply too high. Scholtus (2008) lists some results on total unimodularity that may be used in practice to determine whether a given matrix is totally unimodular without computing determinants.

4.3. *The Scapegoat Algorithm*

4.3.1. Basic Idea

When the survey variables are denoted by the vector $\mathbf{x} = [x_1, \ldots, x_p]'$, the balance edit rules can be written as a linear system

$$R\mathbf{x} = \mathbf{a} \tag{4.2}$$

where each row of the $r \times p$-matrix R defines an edit rule and each column corresponds to a survey variable. The vector $\mathbf{a} = [a_1, \ldots, a_r]'$ contains any constant terms that occur in the edit rules. Denoting the ith row of R by $\mathbf{r}_i'$, an edit rule is violated when $|\mathbf{r}_i'\mathbf{x} - a_i| > 0$. The inconsistency is called a rounding error when $0 < |\mathbf{r}_i'\mathbf{x} - a_i| \leq \delta$, where $\delta > 0$ is small. Similarly, the edit rules that take the form of a linear inequality can be written as

$$Q\mathbf{x} \geq \mathbf{b} \tag{4.3}$$

where each edit rule is defined by a row of the $q \times p$-matrix Q together with a constant from $\mathbf{b} = [b_1, \ldots, b_q]'$. It is assumed until Section 4.3.4 that only balance edit rules are given.

The idea behind the heuristic method is as follows. For each record containing rounding errors, a set of variables is selected beforehand. Next, the rounding errors are resolved by only adjusting the values of these selected variables. Hence, the name *scapegoat algorithm* seems appropriate. The name "scapegoat algorithm" was coined by Léander Kuijvenhoven (Statistics Netherlands).

In fact, the algorithm performs the selection in such a way that exactly one choice of values exists for the selected variables such that all rounding errors are resolved. Different variables are selected for each record to minimise the effect of the adaptations on aggregates.

It is assumed that the $r \times p$-matrix R satisfies $r \leq p$ and $\operatorname{rank}(R) = r$, that is: the number of variables should be at least as large as the number of restrictions and no redundant restrictions may be present. Clearly, these are very mild assumptions. Additionally, the scapegoat algorithm becomes simpler if R is a totally unimodular matrix. At Statistics Netherlands, it was found that matrices of balance edit rules used for structural business statistics are always of this type. A similar observation is made by De Waal (2002, §3.4.1).

An inconsistent record $\mathbf{x}$ is given, possibly containing both rounding errors and other errors. In the first step of the scapegoat algorithm, all rows of R for which $|\mathbf{r}_i'\mathbf{x} - a_i| > \delta$ are removed from the matrix and the associated constants are removed from $\mathbf{a}$. The resulting $r_0 \times p$-matrix is denoted by R_0 and the resulting r_0-vector of constants by $\mathbf{a}_0$. It may happen that the record satisfies the remaining balance edit rules $R_0\mathbf{x} = \mathbf{a}_0$. In that case, the algorithm stops here.

It is easy to see that if R satisfies the assumptions above, then so does R_0. Hence $\operatorname{rank}(R_0) = r_0$ and R_0 has r_0 linearly independent columns. The r_0 leftmost linearly independent columns may be found by putting the matrix in row echelon form through Gaussian elimination, as described by Fraleigh and Beauregard (1995, §2.2), or alternatively by performing a QR-decomposition with column pivoting, as discussed by Golub and Van Loan (1996, §5.4). (How these methods work is irrelevant for the present purpose.) Since the choice of scapegoat variables and hence of columns should vary between records, a random permutation of columns is performed beforehand, yielding $\tilde{R}_0$. The variables of $\mathbf{x}$ are permuted accordingly to yield $\tilde{\mathbf{x}}$.

Next, $\tilde{R}_0$ is partitioned into two submatrices R_1 and R_2. The first of these is an $r_0 \times r_0$-matrix that contains the leftmost linearly independent columns of $\tilde{R}_0$, the second is an $r_0 \times (p - r_0)$-matrix containing all other columns. The vector $\tilde{\mathbf{x}}$ is also partitioned into subvectors $\mathbf{x}_1$ and $\mathbf{x}_2$, containing the variables associated with the columns of R_1 and R_2, respectively. Thus

$$\tilde{R}_0\tilde{\mathbf{x}} = \mathbf{a}_0 \quad \text{becomes} \quad \begin{bmatrix} R_1 & R_2 \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \mathbf{a}_0$$

At this point, the variables from $\mathbf{x}_1$ are selected as scapegoat variables and the variables from $\mathbf{x}_2$ remain fixed. Therefore the values of $\mathbf{x}_2$ are filled in from the original record to obtain the following system:

$$R_1\mathbf{x}_1 = \mathbf{a}_0 - R_2\mathbf{x}_2 \equiv \mathbf{c} \tag{4.4}$$

where $\mathbf{c}$ is a vector of known constants.

By construction, the square matrix R_1 is of full rank and therefore invertible. Thus (4.4) has the unique solution $\hat{\mathbf{x}}_1 = R_1^{-1}\mathbf{c}$. In general, this solution might contain fractional values, whereas most business survey variables are restricted to be integer-valued. If this is the case, a controlled rounding algorithm similar to the one described in Salazar-González et al. (2004) can be applied to the values of $[\hat{\mathbf{x}}_1', \mathbf{x}_2']'$ to obtain an integer-valued solution to

$R_0\mathbf{x} = \mathbf{a}_0$. Note however that this is not possible without slightly changing the value of at least one variable from $\mathbf{x}_2$ too.

If R happens to be a totally unimodular matrix, this problem does not occur. In that case $\det R_1$ is equal to -1 or 1, and Property 4.1 says that $\hat{\mathbf{x}}_1$ is always integer-valued. In the remainder of this article, it is assumed that R is indeed totally unimodular.

4.3.2. An Example

To illustrate the scapegoat algorithm, a small-scale example now follows. Suppose a data set contains records of eleven variables $x_1, \ldots, x_{11}$ that should conform to the following five balance edit rules:

$$\left.\begin{aligned} x_1 + x_2 &= x_3 \\ x_2 &= x_4 \\ x_5 + x_6 + x_7 &= x_8 \\ x_3 + x_8 &= x_9 \\ x_9 - x_{10} &= x_{11} \end{aligned}\right\} \tag{4.5}$$

These edit rules may be written as $R\mathbf{x} = \mathbf{0}$, with $\mathbf{x} = [x_1, \ldots, x_{11}]'$ and

$$R = \begin{bmatrix} 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 \end{bmatrix} \tag{4.6}$$

Thus $\mathbf{a} = \mathbf{0}$ here. It is easily established that rank $(R) = 5$. Moreover, it can be seen that R is totally unimodular by repeatedly applying the following property: a matrix containing only elements from $\{-1, 0, 1\}$ is totally unimodular, if and only if the submatrix found by removing all columns or rows with less than two non-zero elements is totally unimodular (see Scholtus 2008 for a proof).

The following record is inconsistent with respect to (4.5):

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}
12	4	15	4	3	1	8	11	27	41	−13

This record violates all edit rules, except for $x_2 = x_4$. In each instance, the violation is small enough to qualify as a rounding error. Thus in this example R_0 is identical to R.

A random permutation is applied to the elements of $\mathbf{x}$ and the columns of R. Suppose that the permutation is given by

$$1 \rightarrow 11, \quad 2 \rightarrow 8, \quad 3 \rightarrow 2, \quad 4 \rightarrow 5, \quad 5 \rightarrow 10, \quad 6 \rightarrow 9, \quad 7 \rightarrow 7, \quad 8 \rightarrow 1, \quad 9 \rightarrow 4,$$
$$10 \rightarrow 3, \quad 11 \rightarrow 6.$$

This yields the following result:

$$\tilde{R} = \begin{bmatrix} 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

It so happens that the first five columns of $\tilde{R}$ are linearly independent. Thus R_1 consists of the first five columns of $\tilde{R}$, and R_2 consists of the remaining six columns. The scapegoat variables are those that correspond to the columns of R_1, that is to say x_8, x_3, x_{10}, x_9 and x_4. The original values from the record are filled in for the non-scapegoat variables to calculate the constant vector $\mathbf{c}$:

$$\mathbf{c} = -R_2 \begin{bmatrix} x_{11} \\ x_7 \\ x_2 \\ x_6 \\ x_5 \\ x_1 \end{bmatrix} = - \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -13 \\ 8 \\ 4 \\ 1 \\ 3 \\ 12 \end{bmatrix} = \begin{bmatrix} -16 \\ -4 \\ -12 \\ 0 \\ -13 \end{bmatrix}$$

Thus, the following system in $\mathbf{x}_1$ is obtained:

$$R_1\mathbf{x}_1 = \begin{bmatrix} 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_8 \\ x_3 \\ x_{10} \\ x_9 \\ x_4 \end{bmatrix} = \begin{bmatrix} -16 \\ -4 \\ -12 \\ 0 \\ -13 \end{bmatrix} = \mathbf{c}$$

Solving this system yields: $\hat{x}_3 = 16$, $\hat{x}_8 = 12$, $\hat{x}_9 = 28$, $\hat{x}_4 = 4$ and $\hat{x}_{10} = 41$. When the original values of the variables in $\mathbf{x}_1$ are replaced by these values, the record becomes consistent with respect to (4.5):

x_1	x_2	$\hat{x}_3$	$\hat{x}_4$	x_5	x_6	x_7	$\hat{x}_8$	$\hat{x}_9$	$\hat{x}_{10}$	x_{11}
12	4	16	4	3	1	8	12	28	41	−13

Observe that in this example it was not necessary to change the value of every scapegoat variable. In particular, x_4 and x_{10} have retained their original values.

4.3.3. On the Size of the Adjustments

The solution vector $\hat{\mathbf{x}}_1$ is constructed by the scapegoat algorithm without any explicit use of the original vector $\mathbf{x}_1$. Therefore, it is not completely trivial that the adjusted values remain close to the original values, which is obviously desirable. In order to demonstrate this property, two upper bounds on the size of the adjustments are now derived, under the assumption that R is totally unimodular.

Recall that the *maximum norm* of a vector $\mathbf{v} = [v_1, \ldots, v_p]'$ is defined as

$$|\mathbf{v}|_\infty = \max_{j=1,\ldots,p} |v_j|$$

The associated matrix norm is (cf. Stoer and Bulirsch 2002, §4.4):

$$||A||_\infty = \max_{i=1,\ldots,m} \sum_{j=1}^{p} |a_{ij}|$$

with $A = [a_{ij}]$ any $m \times p$-matrix. It is easily shown that

$$|A\mathbf{v}|_\infty \leq ||A||_\infty |\mathbf{v}|_\infty \tag{4.7}$$

for every $m \times p$-matrix A and every p-vector $\mathbf{v}$.

Turning to the scapegoat algorithm, it holds by construction that $R_1\hat{\mathbf{x}}_1 = \mathbf{c}$. The original vector $\mathbf{x}_1$ satisfies $R_1\mathbf{x}_1 = \mathbf{c}^*$, with $\mathbf{c}^* \neq \mathbf{c}$. Thus

$$\hat{\mathbf{x}}_1 - \mathbf{x}_1 = R_1^{-1}(\mathbf{c} - \mathbf{c}^*) \tag{4.8}$$

It follows from (4.7) and (4.8) that

$$|\hat{\mathbf{x}}_1 - \mathbf{x}_1|_\infty \leq ||R_1^{-1}||_\infty |\mathbf{c} - \mathbf{c}^*|_\infty \leq r_0 |\mathbf{c} - \mathbf{c}^*|_\infty \tag{4.9}$$

where the last inequality is found by observing that Property 4.2 implies

$$||R_1^{-1}||_\infty = \max_{i=1,\ldots,r_0} \sum_{j=1}^{r_0} \left| \left(R_1^{-1}\right)_{ij} \right| \leq r_0$$

Writing $\hat{\mathbf{x}} = [\hat{\mathbf{x}}_1', \mathbf{x}_2']'$ and observing that

$$\begin{aligned} \mathbf{c} - \mathbf{c}^* &= R_1\hat{\mathbf{x}}_1 - R_1\mathbf{x}_1 \\ &= R_1\hat{\mathbf{x}}_1 + R_2\mathbf{x}_2 - \mathbf{a}_0 - (R_1\mathbf{x}_1 + R_2\mathbf{x}_2 - \mathbf{a}_0) \\ &= \tilde{R}_0\hat{\mathbf{x}} - \mathbf{a}_0 - (R_0\mathbf{x} - \mathbf{a}_0) \\ &= -(R_0\mathbf{x} - \mathbf{a}_0) \end{aligned}$$

it is seen that $|\mathbf{c} - \mathbf{c}^*|_\infty = |R_0\mathbf{x} - \mathbf{a}_0|_\infty = \delta_{\max}$, where $\delta_{\max} \leq \delta$ is the magnitude of the largest rounding error that occurs for this particular record. Plugging this into (4.9) yields

$$|\hat{\mathbf{x}}_1 - \mathbf{x}_1|_\infty \leq r_0\delta_{\max} \tag{4.10}$$

This upper bound on the maximum difference between elements of $\hat{\mathbf{x}}_1$ and $\mathbf{x}_1$ shows that the solution found by the scapegoat algorithm cannot be arbitrarily far from the original record. The fact that (4.10) is proportional to the order of R_1 suggests that ever larger adjustments should be expected as the number of balance edit rules increases, which is somewhat worrying. However, in practice much smaller adjustments than $r_0\delta_{\max}$ are found. For instance, in the example from Section 4.3.2 the maximal absolute difference according to (4.10) equals 5, but actually no value was changed by more than one unit.

Nevertheless, it is possible to construct a pathological example for which the upper bound (4.10) becomes exact. Scholtus (2008) provides such an example for the case $\delta_{\max} = 2$, which can easily be generalised to work for any value of $\delta_{\max}$.

In practice, a more interesting view on the size of the adjustments may be provided by the quantity

$$\frac{1}{r_0}\sum_{i=1}^{r_0} |(\hat{\mathbf{x}}_1 - \mathbf{x}_1)_i|$$

which measures the *average* size of the adjustments, rather than the maximum. Starting from (4.8), it is seen that

$$|(\hat{\mathbf{x}}_1 - \mathbf{x}_1)_i| = \left|\sum_{j=1}^{r_0} \left(R_1^{-1}\right)_{ij}(\mathbf{c} - \mathbf{c}^*)_j\right| \le \sum_{j=1}^{r_0} \left|\left(R_1^{-1}\right)_{ij}\right|\left|(\mathbf{c} - \mathbf{c}^*)_j\right|$$

Using again that $|\mathbf{c} - \mathbf{c}^*|_\infty = \delta_{\max}$ yields

$$\frac{1}{r_0}\sum_{i=1}^{r_0} |(\hat{\mathbf{x}}_1 - \mathbf{x}_1)_i| \le \frac{\delta_{\max}}{r_0}\sum_{i=1}^{r_0}\sum_{j=1}^{r_0} \left|\left(R_1^{-1}\right)_{ij}\right| \equiv \gamma(R_1)\delta_{\max} \tag{4.11}$$

where $\gamma(R_1) = (1/r_0)\sum_{i=1}^{r_0}\sum_{j=1}^{r_0}\left|\left(R_1^{-1}\right)_{ij}\right|$.

This upper bound on the average adjustment size can be evaluated before the scapegoat algorithm is applied to an actual data set. Namely, suppose that a set of balance edit rules (4.2) is given. Restricting oneself to the case $r_0 = r$, $\gamma(R_1)$ can be computed for various invertible $r \times r$-submatrices of R to assess the magnitude of the upper bound in (4.11). It can be shown (see Scholtus 2008) that there exist exactly $\det(RR')$ of these submatrices. In practice, this number is very large and it is infeasible to compute $\gamma(R_1)$ for all matrices R_1. In that case, a random sample of reasonable size can be taken, by repeatedly performing the part of the scapegoat algorithm that constructs R_1.

Example For the 5×11-matrix from Section 4.3.2, $\det(RR') = 121$, so R has 121 invertible 5×5-submatrices. Since this number is not too large, it is possible to evaluate $\gamma(R_1)$ for all these matrices. The mean value of $\gamma(R_1)$ turns out to be 1.68, with a standard deviation of 0.39. Since $\delta_{\max} = 1$ in this example, according to (4.11) the average adjustment size is bounded on average by 1.68. ▲

Section 4.4 examines the adjustments in a real-world example. These turn out to be quite small.

4.3.4. Critical Variables

In addition to balance edit rules, business survey variables usually have to satisfy a large number of edit rules that take the form of linear inequalities. For instance, it is very common that most variables are restricted to be nonnegative. The scapegoat algorithm as described above does not take this into account. A nonnegative variable might therefore be changed by the algorithm from 0 to -1, resulting in a new violation of an edit rule. The present section extends the algorithm to prevent this.

Suppose that in addition to the balance edit rules (4.2), the data also has to satisfy the inequalities (4.3). For a given record, a variable will be called *critical* if it occurs in an inequality that (almost) holds with exact equality when the current values of the survey variables are filled in:

$$x_j \text{ is a critical variable iff } 0 \leq \mathbf{q}_i'\mathbf{x} - b_i \leq \varepsilon_i \text{ for some } i \text{ with } q_{ij} \neq 0 \qquad (4.12)$$

where $\mathbf{q}_i'$ denotes the ith row of Q and ε_i marks the margin chosen for the ith restriction. As a particular case, x_j is called critical if it must be nonnegative and currently has a value between 0 and $\varepsilon_{i(j)}$, with $i(j)$ the index of the row in Q corresponding to the nonnegativity constraint for x_j. To prevent the violation of edit rules in (4.3), no critical variable should be selected for change during the execution of the scapegoat algorithm.

A way to achieve this works as follows. Rather than randomly permuting all variables (and all columns of R_0), two separate permutations should be performed for the noncritical and the critical variables. The permuted columns associated with the noncritical variables are then placed to the left of the columns associated with the critical variables. This ensures that linearly independent columns are found among those that are associated with noncritical variables, provided the record contains a sufficient number of noncritical variables. In practice, this is typically the case, because the number of survey variables is much larger than the number of balance edit rules.

If a record contains many critical variables, some of these might still be selected as scapegoat variables. This is not necessarily a problem, because usually not all scapegoat variables are changed by the algorithm. This is, in fact, the reason why the critical variables are also randomly permuted: it is unimportant whether a solution to (4.4) contains critical variables, provided that no inequality edit rules are violated as a result. It is therefore sufficient to build in a check at the end of the algorithm that rejects the solution if a new violation of an edit rule from (4.3) is detected. If this does happen, it seems advantageous to let the record be processed again, because a different permutation of columns may yield a feasible solution. To prevent the algorithm from getting stuck, the number of attempts should be maximised by a preset constant K. If no feasible solution has been found after K attempts, the record remains untreated.

Good values of ε_i and K have to be determined in practice. However, not too much effort should be put into this, because these parameters only affect a limited number of records. In the real-world example to be discussed in Section 4.4, only a handful of infeasible solutions were found before the improvements of the current section were included in the algorithm.

4.3.5. Exceptional Variables

In practice, the data may contain some variables that should not be changed by the scapegoat algorithm at all. An example of such a variable in the structural business statistics is *number of employees*. This variable occurs in a balance edit rule that is often inconsistent because of a very small violation, but this violation cannot be the result of inconsistent rounding; this variable is asked as a number, not as a multiple of 1,000 Euros. Moreover, the impact of changing the *number of employees* to suit the balance edit rule can be considerable, particularly for very small companies. Therefore, at this stage it seems preferable to leave the inconsistency as it is, to be resolved later by either a subject-matter specialist or SLICE.

This can be achieved by removing the balance edit rules concerning these exceptional variables from R. The variables themselves should not be removed from $\mathbf{x}$, however, as they may also occur in edit rules in (4.3). (For instance, *number of employees* times a constant is used to maximise the *total labour costs*.) The values of the exceptional variables therefore play a role in determining the critical variables. Note that it is not necessary to remove the exceptional variables from $\mathbf{x}$ anyway, because the columns that correspond with these variables contain only zeros in the new version of R.

4.3.6. Summary

The following plan summarises the scapegoat algorithm. The input consists of an inconsistent record $\mathbf{x}$ with p variables, a set of r balance edit rules $R\mathbf{x} = \mathbf{a}$, a set of q inequalities $Q\mathbf{x} \geq \mathbf{b}$ and parameters δ, $\varepsilon_i\,(i = 1, \ldots, q)$ and K. Edit rules concerning exceptional variables (as described in Section 4.3.5) have been removed from $R\mathbf{x} = \mathbf{a}$ beforehand.

1. Remove all edit rules for which $|\mathbf{r}_i'\mathbf{x} - a_i| > \delta$. The remaining system is denoted as $R_0\mathbf{x} = \mathbf{a}_0$. The number of rows in R_0 is called r_0. If $R_0\mathbf{x} = \mathbf{a}_0$ holds: stop. Otherwise: determine the critical variables according to (4.12).
2.
   (a) Perform random permutations of the critical and noncritical variables separately. Then permute the corresponding columns of R_0 the same way. Put the noncritical variables and their columns before the critical variables and their columns.
   (b) Determine the r_0 leftmost linearly independent columns in the permuted matrix $\tilde{R}_0$. Together, these columns are a unimodular matrix R_1 and the associated variables form a vector $\mathbf{x}_1$ of scapegoat variables. The remaining columns are a matrix R_2 and the associated variables form a vector $\mathbf{x}_2$.
   (c) Fix the values of $\mathbf{x}_2$ from the record and compute $\mathbf{c} = \mathbf{a}_0 - R_2\mathbf{x}_2$.
3. Solve the system $R_1\mathbf{x}_1 = \mathbf{c}$.
4. Replace the values of $\mathbf{x}_1$ by the solution just found. If the resulting record does not violate any other edit rule from $Q\mathbf{x} \geq \mathbf{b}$, the algorithm outputs the adjusted record and terminates. Otherwise, return to step 2a, unless this has been the Kth attempt. In that case, the record is not adjusted.

In this description, it is assumed that R is totally unimodular.

4.4. A Real-world Application

In Section 5, results will be discussed of an application of the two algorithms from this study to a large data set from the Dutch structural business statistics. These results focus on the impact of the algorithms on the efficiency of the editing process. In the current subsection some earlier test results are presented that focus more on technical aspects of the scapegoat algorithm.

The scapegoat algorithm has been tested using data from the wholesale structural business statistics of 2001. There are 4,725 records containing 97 variables each. These variables should conform to a set of 28 balance edit rules and 120 inequalities, of which

Table 3. Results of applying the scapegoat algorithm to the wholesale data

Number of records	4,725
Number of variables per record	97
Number of adjusted records	3,176
Number of adjusted variables	13,531
Number of violated edit rules (before)	34,379
Balance edit rules	26,791
Inequalities	7,588
Number of violated edit rules (after)	23,054
Balance edit rules	15,470
Inequalities	7,584

92 represent nonnegativity constraints. After exclusion of edit rules that affect exceptional variables, 26 balance edit rules remain. The resulting 26×97-matrix R is totally unimodular, as can be determined very quickly using the method of removing columns and rows mentioned in Section 4.3.2. Note that it would be practically impossible to determine whether a matrix of this size is totally unimodular just by computing all the relevant determinants.

An implementation of the algorithm in *S-Plus* was used to treat the data. The parameters used were: $\delta = 2$, $\varepsilon_i = 2\,(i = 1, \ldots, 120)$ and $K = 10$. The total computation time on an ordinary desktop PC was less than three minutes.

Table 3 summarises the results of applying the scapegoat algorithm. No new violations of inequalities were found. In fact, the adjusted data happens to satisfy four additional inequalities.

According to (4.10) the size of the adjustments made by the algorithm is theoretically bounded by $26 \times 2 = 52$, which is rather high. A random sample of 10,000 invertible 26×26-submatrices of R was drawn to evaluate (4.11). The sample mean of $\gamma(R_1)$ is 1.89, with a standard deviation of 0.27. Thus, the average adjustment size is bounded on average by $1.89 \times 2 \approx 3.8$. Note that this value of $\gamma(R_1)$ is only marginally higher than the one obtained for the much smaller restriction matrix from the example in Section 4.3.2.

Table 4 displays the adjustment sizes that were actually found for the wholesale data. These turn out to be very reasonable.

5. Application to the Dutch Structural Business Statistics of 2007

The algorithms from Sections 3 and 4 have been applied in an experiment using data from the Dutch structural business statistics of 2007. The data was collected by Statistics

Table 4. Distribution of the adjustments (in absolute value)

Magnitude	Frequency
1	11,953
2	1,426
3	134
4	12
5	4
6	2

Table 5. An example of structure of the profit-and-loss account in the structural business statistics survey of 2007, with interchanged revenues and costs

Variable	Full name	Original data	Corrected data
$x_{0,r}$	Operating returns	49,110	49,110
$x_{0,c}$	Operating costs	46,550	46,550
x_0	Operating results	2,560	2,560
$x_{1,r}$	Provisions rescinded	340	**0**
$x_{1,c}$	Provisions added	0	**340**
x_1	Balance of provisions	−340	−340
x_2	Book profit/loss	−90	−90
x_3	Financial result	30	30
x_4	Exceptional result	0	0
x_5	Pretax results	2,160	2,160

Netherlands from businesses in various sectors, including wholesale, construction and audiovisual services. The algorithms were run using the original, unedited data.

Table 5 displays the structure of the profit-and-loss account in the structural business survey that was used in 2007. This differs from the examples in Section 3, because the questionnaire was redesigned after 2005. The associated edit rules are given by (3.3), with $n = 5$ and $m = 1$.

The results of applying the algorithm that corrects sign errors and interchanged revenues and costs are displayed in Table 6. As can be seen, the fraction of profit-and-loss accounts requiring editing is low: almost 90 percent of the accounts are reported without error. This can be explained by the fact that in 2007 the majority of businesses reported by electronic questionnaire. Some of the edit rules were built into this questionnaire, so that respondents received a warning message if the reported amounts were inconsistent. In this way, many errors that would occur on a paper questionnaire could be avoided during electronic data collection (Giesen 2007).

On the other hand, among the profit-and-loss accounts that do require editing, the fraction of accounts containing sign errors is substantial: about one in five. This means that, as far as the profit-and-loss account is concerned, using the algorithm of Section 3 substantially reduces the amount of work remaining for either manual editing or automatic editing by SLICE. It should also be mentioned that the majority of errors corrected by the algorithm were in fact interchanged revenues and costs. As noted in Section 3, this type of error is difficult to handle using an automatic editing method based on the Fellegi-Holt paradigm.

The scapegoat algorithm was applied to the data using the same parameter settings as in the real-world application of Section 4.4. The results are displayed in Table 7. Please note

Table 6. Results of applying the correction algorithm for sign errors to the 2007 data

Total number of profit-and-loss accounts	17,258	
Without inconsistencies	15,465	(89.6%)
With inconsistencies	1,793	(10.4%)
Corrected by the algorithm	392	(21.9%)

Table 7. Results of applying the scapegoat algorithm to the 2007 data

Total number of records	17,297	
Without inconsistencies	11,183	(64.6%)
With inconsistencies	6,114	(35.4%)
Corrected by the algorithm	1,295	(21.2%)
Number of violated balance edit rules (before)	11,584	
Number of violated balance edit rules (after)	10,113	
Number of resolved violations	1,471	(12.7%)

that the total number of records and the number of records with/without inconsistencies are not comparable to the corresponding numbers in Table 6, because both records with an empty profit-and-loss account and inconsistencies outside the profit-and-loss account are not counted in Table 6.

Again, the large percentage of records without inconsistencies is due to the use of electronic data collection. This can be seen from the marked difference between the relative number of inconsistencies in the survey data of 2001 (shown in Table 3) and 2007 (shown in Table 7). In 2001, all data collection was still done through paper questionnaires.

Of the records that do contain inconsistencies, about one in five contains at least one rounding error. Moreover, the scapegoat algorithm succeeds in resolving 1,471 of the 11,584 violations of balance edits in the original data set, i.e. about one in eight. This entails a substantial reduction of the amount of editing that remains to be done either manually or automatically by SLICE.

The figures in Table 7 were obtained by applying the scapegoat algorithm directly to the unedited data. In practice, it would be better to first correct sign errors and then rounding errors, because an application of the algorithm from Section 3 may reveal "hidden" rounding errors; see Example (d) in Tables 1 and 2.

6. Conclusion

The main purpose of this study has been to discuss the use of deductive methods for correcting obvious inconsistencies in business survey data, i.e. inconsistencies where the underlying error mechanism can be recognised easily. In particular, algorithms have been described that detect and correct two types of inconsistencies that occur in data collected for the Dutch structural business statistics: sign errors and rounding errors. Other errors are discussed by Scholtus (2008; 2009).

Deductive algorithms are intended to be applied at the beginning of the data editing process, before manual editing and regular automatic editing with SLICE take place. In this way, the efficiency of the editing process is expected to increase, because more records will be eligible for automatic editing. In particular, the presence of obvious errors and rounding errors may cause a record to be submitted to manual editing, because the automatic error localisation problem is too difficult to solve, when in fact the record can be handled automatically by SLICE once the obvious errors have been removed. Moreover, in many cases the use of a deductive algorithm for obvious errors also increases the quality of the edited data, because systematic errors are often handled incorrectly by SLICE. This is for example true for sign errors.

Often, the presence of errors with a structural cause in survey data signifies that many respondents find it difficult to answer correctly because of a certain aspect of the questionnaire design. An alternative way to handle systematic errors is, therefore, to try to prevent them during data collection, e.g. by improving the wording of questions or the design of answer boxes, or by adding explanatory notes, or – in the case of electronic data collection – by means of warning messages. If this approach succeeds in removing the underlying cause of the systematic error, then the need for a deductive correction algorithm vanishes.

Nevertheless, in practice, deductive correction methods can still play an important part in the editing process. Firstly, it is not always possible to prevent systematic errors through an improved form of data collection. For instance, unity measure errors have been observed for many years in data collected by statistical offices but, so far, no conclusive method has been found to stop respondents from making this type of error. Secondly, changes in the questionnaire design are costly, because the new questionnaire has to be extensively tested, and they can adversely influence the comparability of statistics over time. Hence, they should not be implemented too often. By contrast, implementing a deductive correction algorithm is cheap and straightforward. If a new systematic error is discovered in the data, it can therefore be advantageous to correct this error deductively at first, until a major revision of the data collection strategy is due.

The algorithms for correcting sign errors and rounding errors described in this article have been implemented as part of the R package *deducorrect*, which is available for download from the Comprehensive R Archive Network (http://cran.r-project.org). See Van der Loo et al. (2001) for more details.

7. References

Al-Hamad, A., Lewis, D., and Silva, P.L.N. (2008). Assessing the Performance of the Thousand Pounds Automatic Editing Procedure at the Office for National Statistics and the Need for An Alternative Approach. Working Paper, UN/ECE Work Session on Statistical Data Editing, Vienna, Austria.

Banff Support Team (2003). Functional Description of the Banff System for Edit and Imputation. Technical Report, Statistics Canada.

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

De Waal, T. (2002). Algorithms for Automatic Error Localisation and Modification. Working Paper, UN/ECE Work Session on Statistical Data Editing, Helsinki, Finland.

De Waal, T. (2003). A Simple Branching Scheme for Solving the Error Localisation Problem. Discussion Paper 03010, Statistics Netherlands.

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

Di Zio, M., Guarnera, U., and Luzi, O. (2005). Editing Systematic Unity Measure Errors through Mixture Modelling. Survey Methodology, 31, 53–63.

Eurostat (2007). Recommended Practices for Editing and Imputation in Cross-Sectional Business Surveys. Manual prepared by ISTAT, Statistics Netherlands, and SFSO.

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

Fraleigh, J.B. and Beauregard, R.A. (1995). Linear Algebra (Third Edition). Reading, MA: Addison-Wesley.

Giesen, D. (2007). Does Mode Matter? First Results of the Comparison of the Response Burden and Data Quality of a Paper Business Survey and an Electronic Business Survey. Paper presented at QUEST 2007, 24–26 April, Ottawa, Canada.

Golub, G.H. and Van Loan, C.F. (1996). Matrix Computations (Third Edition). Baltimore: The Johns Hopkins University Press.

Harville, D.A. (1997). Matrix Algebra from a Statistician's Perspective. New York: Springer.

Hoogland, J. (2006). Selective Editing using Plausibility Indicators and SLICE. Statistical Data Editing, Volume No. 3, Impact on Data Quality. New York and Geneva: United Nations, 106–130.

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

Salazar-González, J.J., Lowthian, P., Young, C., Merola, G., Bond, S., and Brown, D. (2004). Getting the Best Results in Controlled Rounding with the Least Effort. Privacy in Statistical Databases, J. Domingo-Ferrer and V. Torra (eds). Berlin: Springer, 58–72.

Scholtus, S. (2008). Algorithms for Correcting Some Obvious Inconsistencies and Rounding Errors in Business Survey Data. Discussion Paper 08015, Statistics Netherlands, The Hague.

Scholtus, S. (2009). Automatic Correction of Simple Typing Errors in Numerical Data with Balance Edits. Discussion Paper 09046, Statistics Netherlands, The Hague.

Stoer, J. and Bulirsch, R. (2002). Introduction to Numerical Analysis (Third Edition). New York: Springer.

Todaro, 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, Italy.

Van der Loo, M., De Jonge, E., and Scholtus, S. (2011). Correction of Rounding, Typing, and Sign Errors with the Deducorrect Package. Discussion Paper 201119, Statistics Netherlands, The Hague.

Winkler, W.E. and Draper, L.R. (1997). The SPEER Edit System. Statistical Data Editing, Volume No. 2, Methods and Techniques. New York and Geneva: United Nations, 51–55.

Winkler, W.E. and Petkunas, T.F. (1997). The DISCRETE Edit System. Statistical Data Editing, Volume No. 2, Methods and Techniques. New York and Geneva: United Nations, 55–62.

Received July 2009
Revised January 2011