Algorithms for correcting some obvious inconsistencies and rounding errors in business survey data

08

Sander Scholtus

Discussion paper (08015)

Statistics Netherlands

The Hague/Heerlen, 2008

Explanation of symbols

.	= data not available
*	= provisional figure
x	= publication prohibited (confidential figure)
–	= nil or less than half of unit concerned
–	= (between two figures) inclusive
0 (0,0)	= less than half of unit concerned
blank	= not applicable
2005-2006	= 2005 to 2006 inclusive
2005/2006	= average of 2005 up to and including 2006
2005/'06	= crop year, financial year, school year etc. beginning in 2005 and ending in 2006
2003/'04–2005/'06	= crop year, financial year, etc. 2003/'04 to 2005/'06 inclusive

Due to rounding, some totals may not correspond with the sum of the separate figures.

Publisher
Statistics Netherlands
Henri Faasdreef 312
2492 JP The Hague

Prepress
Statistics Netherlands - Facility Services

Cover
TelDesign, Rotterdam

Information
Telephone .. +31 88 570 70 70
Telefax .. +31 70 337 59 94
Via contact form: www.cbs.nl/information

Where to order
E-mail: verkoop@cbs.nl
Telefax .. +31 45 570 62 68

Internet
www.cbs.nl

ISSN: 1572-0314

6008308015 X-10

Summary

At Statistics Netherlands selective editing is used for the data of the structural business statistics. Records containing potentially influential errors are edited manually, while the non-critical records are edited automatically by an advanced software package called SLICE. Prior to this several types of obvious inconsistencies are detected and corrected deductively. The rationale of this step is that it would be a waste of resources if these errors were to be corrected by hand. Moreover, these obvious inconsistencies are in fact systematic errors that can be corrected more accurately by a separate, specific algorithm than by general editing software such as SLICE. This paper describes two additional types of frequently occurring obvious inconsistencies and gives simple algorithms to detect and correct them. It also contains a heuristic method for resolving rounding 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.

Keywords: structural business statistics, editing, obvious inconsistencies, sign errors, cumulation errors, rounding errors

1 Introduction

It is well known that data collected in a survey or register contain 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 these 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

$$profit = turnover - costs$$

and

$$number\ of\ employees\ (in\ persons) \geq number\ of\ employees\ (in\ fte).$$

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

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 respondents. This type of error can occur when a respondent misunderstands or misreads a survey question, e.g. by reporting financial amounts in Euros rather than the requested multiples of 10^3 Euros. A fault in the data processing software might also introduce a systematic error. Since it is reported consistently by different respondents, an undiscovered systematic error leads to biased aggregates. Once detected, a systematic error can be corrected (semi-)deductively because the underlying error mechanism is known. Random errors on the other hand occur by accident, e.g. when a 1 on a survey form is read as a 7 during data processing. Because of their random nature, no deductive method can typically be applied to correct errors of this type.

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 the software package *SLICE* was developed at Statistics Netherlands. SLICE uses an advanced error localisation algorithm based on a generalisation of the *Fellegi-Holt paradigm* (Fellegi & 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 contain only random errors. A description of the error localisation algorithm implemented in SLICE can be found in de Waal & 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 that do not receive a sufficiently high plausibility indicator are edited manually. In addition to this the largest companies are always edited manually, since they are likely to have a substantial influence on aggregates.

Selective editing leads to a more efficient editing process than traditional editing (where every record is corrected by hand), since part of the data stream is not reviewed by subject-matter specialists anymore. However, Fellegi-Holt based algorithms for automatic error localisation are not considered suitable for editing records that contain either influential or systematic errors. Moreover, in practice the error localisation problem becomes too complicated if a record violates too large a number of edit rules. To preserve the quality of the editing process, only records that contain a limited number of non-influential 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. An example of an obvious error occurs when a respondent reports the values of some items in a sum, but leaves the corresponding total blank. This error can be corrected deductively by calculating and filling in the total value.

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. Solving the error localisation problem becomes easier once obvious inconsistencies have been removed, since the number of violated edit rules becomes smaller.

Moreover, 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. 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 we leave the error in to be detected by SLICE, all we can hope for is that the general algorithm will detect and correct the error the same way the simple algorithm would have done, but at the cost of higher computational effort.

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. The purpose of this paper is to present algorithms for detecting and correcting further obvious inconsistencies. These error types were discovered through an inspection of data collected in the past. Section 3 deals with so-called sign errors. Cumulation errors are the subject of section 4. Finally, section 5 presents a heuristic method for correcting rounding errors. Rounding errors are not obvious inconsistencies in the true sense of the word (they are random, not systematic), but the efficiency of the editing process is expected to increase if these errors are also treated separately.

Due to item non-response, the unedited data contain a substantial number of missing values. It is assumed throughout this paper that these missing values have been temporarily replaced by 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 imputed zeros should be replaced by missing values again, to be imputed by a valid method later.

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 was already mentioned in section 1: the amounts on the survey form are sometimes reported in Euros instead of multiples of 1,000 Euros. This is 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.

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 wrongfully 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 was also mentioned in section 1: some respondents report component items of a sum but leave the corresponding total blank. When this is detected, the total value is calculated and filled in.

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. We refer to these balance variables by $x_0, x_1, \ldots, x_{n-1}$. A final balance amount x_n called the *pre-tax 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 returns item and a costs 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 returns item and $x_{k,c}$ the costs item. Rules of this form are referred to as *internal sums*.

We will use a general description of the profit-and-loss account, in which the returns and costs are not necessarily asked for every balance variable in the survey. To keep the notation simple we assume that the balance variables are arranged such that only $x_0, \ldots, x_m$ are split into returns and costs, for some $m \in \{0, 1, \ldots, n-1\}$. Thus, the following set of edit rules is used:

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

In this notation the 0^{th} 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 returns 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 $m = n - 1 = 3$. (That is, all balance variables are split into returns and costs here.) Table 1 also displays four example records that are inconsistent. The first three example records have been constructed for this paper 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 more realistic values.)

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

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	operating surplus	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	pre-tax results	195	610	−140	221

In example (a) two edit rules are violated: the external sum and the internal sum with $k = 1$. Interestingly, the profit-and-loss account can be made fully consistent with all edit rules by 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 is valid. 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 quite common. We will refer to them as *sign errors* and *interchanged returns and costs*. For the sake of brevity we also use the term *sign error* to refer to both types. Sign errors and interchanged returns and costs are closely related and should therefore be searched for by one detection algorithm. We will now formulate such an algorithm, 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

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	operating surplus	**−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	pre-tax results	195	610	−140	221

and/or interchanging returns items and costs items, this is indeed the way the record should be corrected.

It should be noted that the 0^{th} returns and costs items 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. E.g. *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 0^{th} internal sum, other edit rules might be violated. When detecting sign errors we therefore introduce the constraint that we are not allowed to interchange $x_{0,r}$ and $x_{0,c}$. (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 returns and costs 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:

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

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 immediately tells us 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 we are not allowed to interchange $x_{0,r}$ and $x_{0,c}$, no variable t_0 is present in (3.4).

Example. By way of illustration we set up (3.4) for example (c) from Table 1:

$$\begin{cases} 300 s_0 &= -300 \\ 100 s_1 &= 100 t_1 \\ 40 s_2 &= -40 t_2 \\ 20 s_3 &= 20 t_3 \\ -140 s_4 &= 300 s_0 + 100 s_1 + 40 s_2 + 20 s_3 \\ (s_0, s_1, s_2, s_3, s_4; t_1, t_2, t_3) &\in \{-1, 1\}^5 \times \{-1, 1\}^3 \end{cases} \tag{3.5}$$

This system has the following (unique) solution:

$$(s_0, s_1, s_2, s_3, s_4; t_1, t_2, t_3) = (-1, 1, 1, 1, 1; 1, -1, 1).$$

This solution tells us 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 results block 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? We derive a sufficient condition for uniqueness in appendix A at the end of this paper. It appears that this condition is usually satisfied; in the data that we examined the condition holds for over 95 percent of all records.

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, we write (3.4) in matrix notation to shorten the expressions. Define the $(m+2)\times(n+1)$-matrix $\mathbf{U}$ by

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

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

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

Note that the bottom row of $\mathbf{V}$ consists entirely of zeros. We also 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:

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

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[1] 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 we will see presently that this formulation can be adapted easily to accommodate possible rounding errors present in the data.

To reformulate the problem, we introduce the following binary variables:

$$\begin{aligned} \sigma_j &= \tfrac{1-s_j}{2}, && j\in\{0,1,\ldots,n\}, \\ \tau_k &= \tfrac{1-t_k}{2}, && k\in\{1,\ldots,m\}. \end{aligned}$$

Finding a solution to (3.6) may now be re-stated as follows:

$$\begin{aligned} &\text{minimise } \textstyle\sum_{j=0}^{n}\sigma_j+\sum_{k=1}^{m}\tau_k \\ &\text{such that:} \\ &\mathbf{U}(\mathbf{1}-2\boldsymbol{\sigma})-\mathbf{V}(\mathbf{1}-2\boldsymbol{\tau})=\mathbf{0} \\ &\sigma_0,\ldots,\sigma_n,\tau_1,\ldots,\tau_m\in\{0,1\}, \end{aligned} \tag{3.7}$$

[1] The actual number of possible vectors is $2^{m+n+1}-1$, if it has been established beforehand that the profit-and-loss account is inconsistent.

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 (unweighted) 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 the smallest possible difference. For instance, a reported total value is just one or two units smaller or larger than the sum of the reported item values. We refer to these inconsistencies as *rounding errors* if the absolute difference is no larger than two units.

In the profit-and-loss account, rounding errors can occur in two ways. Firstly the pre-tax results may differ slightly from the sum of the balance amounts (external sum),

$$x_0 + \cdots + x_{n-1} \neq x_n, \text{ but } -2 \leq x_0 + \cdots + x_{n-1} - x_n \leq 2,$$

and secondly a balance amount may just disagree with the difference between the reported returns and costs items (internal sum),

$$x_{k,r} - x_{k,c} \neq x_k, \text{ but } -2 \leq x_{k,r} - x_{k,c} - x_k \leq 2,$$

for some $k \in \{0, 1, \ldots, m\}$.

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.

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:

$$\begin{aligned} &\text{minimise } \textstyle\sum_{j=0}^{n} \sigma_j + \sum_{k=1}^{m} \tau_k \\ &\text{such that:} \\ &\mathbf{U}(\mathbf{1} - 2\boldsymbol{\sigma}) - \mathbf{V}(\mathbf{1} - 2\boldsymbol{\tau}) \geq -\mathbf{2} \\ &\mathbf{U}(\mathbf{1} - 2\boldsymbol{\sigma}) - \mathbf{V}(\mathbf{1} - 2\boldsymbol{\tau}) \leq \mathbf{2} \\ &\sigma_0, \ldots, \sigma_n, \tau_1, \ldots, \tau_m \in \{0, 1\}, \end{aligned} \tag{3.8}$$

where $\mathbf{2}$ is a vector of 2's and the rest of the notation is used as before.

Example. If (3.8) is set up for example (d) from Table 1, we find the following solution:

$$(\sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_4; \tau_1, \tau_2, \tau_3) = (0, 1, 1, 0, 0; 0, 0, 0).$$

Recalling that $\sigma_j = 1$ if and only if $s_j = -1$ (and a similar expression for τ_k and t_k), we conclude that 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 gets rid of 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, we need a separate method. This problem will be discussed in section 5. $\triangle$

3.5 Summary

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

1. Determine the matrices $\mathbf{U}$ and $\mathbf{V}$ 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 Cumulation errors

4.1 Definition and basic correction method

Table 3 shows three example records that have an inconsistent profit-and-loss account. (A description of the profit-and-loss account was given in section 3.1.) Each record contains a variation on a type of error that we will call a *cumulation error*. It appears that these respondents have reported intermediate versions of the final balance amount instead of correct values for the separate balances. This error occurs persistently in examples (a) and (b). Example (c) is slightly more complex. Here, most of the balance amounts are correct but a cumulation error occurs for one of them. To complicate matters further, this respondent has also interchanged the values of financial revenues and expenditure.

Table 3. Three example profit-and-loss accounts based on data from the structural business statistics 2001.

variable	*full name*	*(a)*	*(b)*	*(c)*
$x_{0,r}$	operating returns	6,700	8,300	6,900
$x_{0,c}$	operating costs	5,650	5,400	6,150
x_0	operating results	1,050	2,900	750
$x_{1,r}$	financial revenues	0	0	0
$x_{1,c}$	financial expenditure	0	150	40
x_1	operating surplus	1,050	2,750	790
$x_{2,r}$	provisions rescinded	0	0	0
$x_{2,c}$	provisions added	0	30	0
x_2	balance of provisions	1,050	2,720	0
$x_{3,r}$	exceptional income	0	0	0
$x_{3,c}$	exceptional expenses	0	110	0
x_3	exceptional result	1,050	2,610	0
x_4	pre-tax results	1,050	2,610	790

We will now describe a method to detect and correct cumulation errors in the profit-and-loss account automatically. We use some of the notation that was introduced in section 3.1. To avoid complications we assume throughout this section that $m = n - 1$, i.e. all balance amounts are accompanied by a returns item and a costs item.

Consider the variables $x_{k,r}$, $x_{k,c}$ and x_k, for some $k \in \{1, \ldots, n-1\}$. Suppose that the data in a given record do not satisfy the external sum (3.1), nor the

Table 4. Correction of cumulation errors occurring in example (b) of Table 3.

variable	*full name*	*input*	*step 1*	*step 2*	*output*
$x_{0,r}$	operating returns	8,300	8,300	8,300	8,300
$x_{0,c}$	operating costs	5,400	5,400	5,400	5,400
x_0	operating results	2,900	2,900	2,900	2,900
$x_{1,r}$	financial revenues	0	0	0	0
$x_{1,c}$	financial expenditure	150	150	150	150
x_1	operating surplus	2,750	**−150**	−150	−150
$x_{2,r}$	provisions rescinded	0	0	0	0
$x_{2,c}$	provisions added	30	30	30	30
x_2	balance of provisions	2,720	2,720	**−30**	−30
$x_{3,r}$	exceptional income	0	0	0	0
$x_{3,c}$	exceptional expenses	110	110	110	110
x_3	exceptional result	2,610	2,610	2,610	**−110**
x_4	pre-tax results	2,610	2,610	2,610	2,610

k^{th} internal sum (3.2), but the reported amounts do satisfy

$$x_k = x_{k-1} + x_{k,r} - x_{k,c}. \tag{4.1}$$

If this is true, then we say that the record contains a cumulation error. The error may be corrected by replacing the original values of $x_{k,r}$, $x_{k,c}$ and x_k by

$$x'_{k,r} = x_{k,r}, \qquad x'_{k,c} = x_{k,c}, \qquad x'_k = x_k - x_{k-1}, \tag{4.2}$$

since from (4.1) it is obvious that $x'_k = x'_{k,r} - x'_{k,c}$.

By letting k run from 1 to $n-1$, all cumulation errors in the profit-and-loss account may be found. Note that for the method to work correctly, the *original* value of x_{k-1} should be used in (4.1) and (4.2) rather than x'_{k-1}.

Example. Table 4 shows the result when this method is applied to example (b) of Table 3. In the first step we take $k = 1$ and obtain for (4.1):

$$2,750 = 2,900 + 0 - 150.$$

Since this equality holds true, a cumulation error is detected. The error is corrected by taking

$$x'_{1,r} = 0, \qquad x'_{1,c} = 150, \qquad x'_1 = 2,750 - 2,900 = -150.$$

The other columns of Table 4 are obtained analogously. △

4.2 Allowing for sign errors

It is not uncommon for a record to contain a cumulation error as well as a sign error (e.g. example (c) in Table 3). The method described in section 4.1 can be modified to detect these errors as well. We assume that the record at hand does not satisfy the external sum, nor the k^{th} internal sum, and that $x_{k-1} \neq 0$. Instead of (4.1), it should be checked whether

$$\lambda x_k = x_{k-1} + \mu\,(x_{k,r} - x_{k,c}) \tag{4.3}$$

holds for any combination $(\lambda, \mu) \in \{-1, 1\}^2$. This actually yields four equalities. If our record satisfies any of these four equalities, it is deduced that the record contains a cumulation error. The case $\lambda = -1$ then corresponds to a cumulation error in conjunction with a sign error in x_k, and the case $\mu = -1$ corresponds to a cumulation error in conjunction with interchanged values of $x_{k,r}$ and $x_{k,c}$. Note that the case $\lambda = \mu = 1$ coincides with (4.1) above.

If an error is detected through (4.3), it may then be corrected deductively by replacing the original values of $x_{k,r}$, $x_{k,c}$ and x_k by the new values

$$\begin{array}{rcl} x'_{k,r} & = & \frac{1+\mu}{2} x_{k,r} + \frac{1-\mu}{2} x_{k,c}, \\ x'_{k,c} & = & \frac{1-\mu}{2} x_{k,r} + \frac{1+\mu}{2} x_{k,c}, \\ x'_k & = & \lambda x_k - x_{k-1}. \end{array} \tag{4.4}$$

Note that $x'_{k,r} = x_{k,r}$ if $\mu = 1$ and $x'_{k,r} = x_{k,c}$ if $\mu = -1$, and similar expressions for $x'_{k,c}$. It follows from (4.3) that $x'_k = x'_{k,r} - x'_{k,c}$.

Example. In example (c) of Table 3 we find that

$$x_1 = 790 = 750 - (-40) = x_0 - (x_{1,r} - x_{1,c})\,,$$

i.e. (4.3) holds for $\lambda = 1, \mu = -1$. Using (4.4) we obtain the new values $x'_{1,r} = x_{1,c} = 40$, $x'_{1,c} = x_{1,r} = 0$ and $x'_1 = x_1 - x_0 = 40$. This correction removes both the cumulation error and the sign error. △

We proceed to show that, under the natural assumptions that $x_{k-1} \neq 0$ and $x_k \neq x_{k,r} - x_{k,c}$, the concept of a cumulation error as suggested by (4.3) is well-defined, because (4.3) can only hold for at most one combination $(\lambda, \mu) \in \{-1, 1\}^2$. To proof this, we write (4.3) as

$$\lambda a = c + \mu b, \tag{4.5}$$

where a, b, c are constants and it is assumed that $c \neq 0$ and $a \neq b$. First of all if it happens that $b = 0$, (4.5) reduces to $\lambda a = c$, which clearly has at most one solution for λ.[2] A similar argument deals with the case $a = 0$. (Note that the

[2] Strictly speaking the value of μ may be chosen freely and there is no unique solution in this sense. Since $x_{k,r} = x_{k,c}$, (4.4) yields the same $x'_{k,r}$ and $x'_{k,c}$ for both choices of μ, so these *are* determined uniquely. The point is that the value of μ is not very interesting in this degenerate case.

case $a = b = 0$ is excluded by our assumptions.) Therefore we assume from now on that $a \neq 0$ and $b \neq 0$. Suppose there are two different combinations of $\lambda \in \{-1, 1\}$ and $\mu \in \{-1, 1\}$ such that (4.5) holds, say (λ_1, μ_1) and (λ_2, μ_2). It follows that

$$\lambda_1 a - \mu_1 b = \lambda_2 a - \mu_2 b,$$

or, equivalently,

$$(\lambda_1 - \lambda_2)\, a = (\mu_1 - \mu_2)\, b. \tag{4.6}$$

Since a and b are non-zero and since at least one of $\lambda_1 - \lambda_2$ and $\mu_1 - \mu_2$ is non-zero, it follows that both these terms must be non-zero, that is: $\lambda_1 = -\lambda_2$ and $\mu_1 = -\mu_2$. Thus (4.6) reduces to

$$\lambda_1 a = \mu_1 b.$$

This means that either $a = b$ or $a = -b$. The former is prohibited by our assumptions, so we find that $a = -b$ and $\lambda_1 = -\mu_1$. However, if $a = -b$ then (4.5) can be written as

$$(\lambda + \mu)\, a = c.$$

Plugging in our supposed solution $\lambda_1 = -\mu_1$ thus yields $c = 0$. This flatly contradicts our assumption that c is non-zero. Hence there exists at most one combination $(\lambda, \mu) \in \{-1, 1\}^2$ such that (4.5) holds.

4.3 Allowing for rounding errors

As with sign errors, it often happens that cumulation errors occur in conjunction with rounding errors. Thus the deductive correction method of section 4.2 will be more effective if it takes these rounding errors into account. This can be achieved quite straightforwardly.

As a generalisation of (4.3), we can check whether there exists any combination $(\lambda, \mu) \in \{-1, 1\}^2$ such that

$$-2 \leq \lambda x_k - x_{k-1} - \mu\,(x_{k,r} - x_{k,c}) \leq 2 \tag{4.7}$$

holds. If this is indeed the case, the record under scrutiny contains a cumulation error. If λ or μ equals -1 it also contains a sign error, and if the expression between the two inequality signs is non-zero it also contains a rounding error. In either case, the cumulation error may be corrected deductively by taking the new values from (4.4). This correction also gets rid of the sign error, as before. It does not remove the rounding error, however, but it does make the rounding error visible so that it may removed by another method. Deriving such a method will be the subject of section 5.

4.4 Summary

The following plan summarises the correction method for cumulation errors. The input consists of a record $\mathbf{x}$ for which $-2 \leq x_0 + \cdots + x_{n-1} - x_n \leq 2$ does not hold (i.e. the external sum is violated and this can not be explained as a rounding error).

1. Define $x'_0 = x_0$, $x'_{0,r} = x_{0,r}$, $x'_{0,c} = x_{0,c}$ and $x'_n = x_n$. Let $k = 1$.
2. If the record satisfies $-2 \leq x_k - (x_{k,r} - x_{k,c}) \leq 2$ and/or if $x_{k-1} = 0$: define $x'_k = x_k$, $x'_{k,r} = x_{k,r}$ and $x'_{k,c} = x_{k,c}$, and continue to step 4. Otherwise: continue to step 3.
3. (a) Check whether there exists any combination $(\lambda, \mu) \in \{-1, 1\}^2$ such that (4.7) holds. If not: continue to step 4.
   (b) Compute x'_k, $x'_{k,r}$ and $x'_{k,c}$ from (4.4), for (λ, μ) found in step 3a.
4. Let $k = k + 1$. If $k = n$: stop. Otherwise: return to step 2.

The output consists of a record $\mathbf{x}'$ that does not contain any cumulation errors.

This method requires that $m = n - 1$, i.e. all balance amounts in the profit-and-loss account are accompanied by a returns item and a costs item.

5 Rounding errors

5.1 Introduction

In the previous sections we briefly touched on the subject of very small inconsistencies with respect to balance edit rules, e.g. the situation that a total value is just one unit smaller or larger than the sum of the values of the component items. We call such inconsistencies rounding errors, because they may be caused by values being rounded off to multiples of $1,000$ Euros. 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, the additivity is destroyed: $3 + 8 \neq 10$. Several algorithms for consistent rounding are available from the literature (see e.g. Bakker (1997) and Salazar-González et al. (2004)), but obviously very few respondents are aware of these methods or indeed inclined to use them while filling in a questionnaire.

Rounding errors may also be introduced when correcting uniform $1,000$-errors, as the following example demonstrates. The third column of Table 5 shows (part of) a record that contains a uniform $1,000$-error. This error is detected through the use of the reference data shown in the second column. To correct the error, all original values are divided by $1,000$ and rounded to the nearest integer. Note that the original (erroneous) values conform to the edit rule $x_1 + x_2 = x_3$, whereas the corrected values do not. Again, this problem can be avoided by using a controlled rounding method after dividing by $1,000$.

Table 5. Correcting a uniform $1,000$*-error, introducing a rounding error.*

variable	*reference data*	*original data*	*corrected data*
x_1	$3,331$	$3,148,249$	$3,148$
x_2	709	$936,442$	936
x_3	$4,040$	$4,084,691$	$4,085$

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 we 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 these violations. Thus a record containing many rounding errors and very few 'real' errors might not be suitable for automatic editing by means of SLICE 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.

In the remainder of this section we describe a heuristic method to resolve rounding errors in business survey data. We call this method 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 of 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, we need some results from matrix algebra to explain why it works. The necessary background will be summarized in section 5.2.

5.2 Something about matrices

5.2.1 General concepts

The vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ are called *linearly independent* if there do not exist numbers $\lambda_1, \ldots, \lambda_k$, not all equal to 0, such that $\lambda_1\mathbf{v}_1 + \cdots + \lambda_k\mathbf{v}_k = \mathbf{0}$. Otherwise, the vectors are called *linearly dependent*.

It can be shown that for any matrix the maximum number of linearly independent columns and the maximum number of linearly independent rows are equal. This common number is referred to as the *rank* of the matrix. If all columns or rows are linearly independent, the matrix is said to be of full rank.

A linear system of m equations in n unknowns may be expressed as $\mathbf{Ax} = \mathbf{b}$, where the $m \times n$-matrix $\mathbf{A}$ and the m-vector $\mathbf{b}$ are known and the n-vector $\mathbf{x}$ is to be found. When $m = n$, $\mathbf{A}$ is called a *square matrix*. If $\mathbf{A}$ is square and there exists a so-called *inverse matrix* $\mathbf{A}^{-1}$ such that the unique solution to $\mathbf{Ax} = \mathbf{b}$ is given by $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$, irrespective of the choice of $\mathbf{b}$, then $\mathbf{A}$ is called *invertible*. A non-invertible square matrix is called *singular*. A necessary and sufficient condition for a square matrix to be invertible is that it be of full rank.

To every square matrix $\mathbf{A}$ there is attached a number called the *determinant*, denoted by $\det \mathbf{A}$. The determinant of a 1×1-matrix is simply the value of its single element. The determinant of an $n \times n$-matrix ($n > 1$) may be defined as a weighted sum of smaller determinants:

$$\det \mathbf{A} = \sum_{i=1}^{n} (-1)^{i+j} a_{ij} \det \mathbf{C}_{ij}, \tag{5.1}$$

where $\mathbf{C}_{ij}$ denotes the $(n-1) \times (n-1)$-matrix found by removing the i^{th} row and j^{th} column of $\mathbf{A}$. Note that in (5.1) a fixed column is chosen for removal and the sum runs over all rows of $\mathbf{A}$. The outcome of this sum appears to depend on the choice of the j^{th} column, but it can be shown that this is not the case. Alternatively, the determinant may be calculated by choosing a fixed row

and summing over all columns analogously to (5.1); this also produces the same value. Once this has been established, it is easy to see that $\det \mathbf{A}' = \det \mathbf{A}$, where $\mathbf{A}'$ denotes the *transpose* of $\mathbf{A}$. The following is a very useful matrix property: a square matrix is invertible (that is: of full rank) if and only if its determinant is non-zero.

The coefficient $(-1)^{i+j} \det \mathbf{C}_{ij}$ in (5.1) is called the *cofactor* of the element a_{ij}. By taking the transpose of the matrix of cofactors of elements of $\mathbf{A}$, we obtain the so-called *adjoint matrix* of $\mathbf{A}$, denoted by $\mathbf{A}^\dagger$: $\left(\mathbf{A}^\dagger\right)_{ij} = (-1)^{i+j} \det \mathbf{C}_{ji}$. It can be shown that for any invertible matrix $\mathbf{A}$,

$$\mathbf{A}^{-1} = \frac{1}{\det \mathbf{A}} \mathbf{A}^\dagger. \tag{5.2}$$

Cramer's Rule is a theorem named after the Swiss mathematician Gabriel Cramer (1704-1752), which goes as follows. Let $\mathbf{A} = [a_{ij}]$ be an invertible $n \times n$-matrix. The unique solution $\mathbf{x} = [x_1, \ldots, x_n]'$ to the system $\mathbf{Ax} = \mathbf{b}$ is given by:

$$x_k = \frac{\det \mathbf{B}_k}{\det \mathbf{A}}, \qquad k = 1, \ldots, n, \tag{5.3}$$

where $\mathbf{B}_k$ denotes the matrix found by replacing the k^{th} column of $\mathbf{A}$ by $\mathbf{b} = [b_1, \ldots, b_n]'$. Cramer's Rule is actually another way of expressing (5.2).

To prove Cramer's Rule, we remark that $\mathbf{Ax} = \mathbf{b}$ can be written as

$$x_1 \mathbf{a}_1 + \cdots + x_k \mathbf{a}_k - \mathbf{b} + \cdots + x_n \mathbf{a}_n = \mathbf{0},$$

where $\mathbf{a}_1, \ldots, \mathbf{a}_n$ denote the columns of $\mathbf{A}$. Thus $\mathbf{a}_1, \ldots, x_k \mathbf{a}_k - \mathbf{b}, \ldots, \mathbf{a}_n$ are linearly dependent vectors, and the determinant of the matrix found by replacing the k^{th} column of $\mathbf{A}$ by the vector $x_k \mathbf{a}_k - \mathbf{b}$ must be 0. Hence we deduce from (5.1) that

$$\begin{aligned} 0 &= \sum_{i=1}^{n} (-1)^{i+k} \left(x_k a_{ik} - b_i\right) \det \mathbf{C}_{ik} \\ &= x_k \sum_{i=1}^{n} (-1)^{i+k} a_{ik} \det \mathbf{C}_{ik} - \sum_{i=1}^{n} (-1)^{i+k} b_i \det \mathbf{C}_{ik} \\ &= x_k \det \mathbf{A} - \det \mathbf{B}_k. \end{aligned}$$

Since $\mathbf{A}$ is invertible, $\det \mathbf{A} \neq 0$. Cramer's Rule now follows.

Proofs of all other observations made in this subsection may be found in most introductory books on linear algebra, such as Fraleigh & Beauregard (1995) and Harville (1997).

5.2.2 Unimodular and totally unimodular matrices

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

A 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. We stress that a totally unimodular matrix need not itself be square.

A stronger version of Property 5.1 may be proved for the submatrices of a totally unimodular matrix.

Property 5.2 *Let* $\mathbf{B}$ *be a square submatrix of a totally unimodular matrix. If* $\mathbf{B}$ *is invertible, all elements of* $\mathbf{B}^{-1}$ *are* 0, 1 *or* -1.

Proof. We use the adjoint matrix $\mathbf{B}^{\dagger}$. Since $|\det \mathbf{B}| = 1$ and all cofactors are equal to 0, 1 or -1, the property follows immediately from equation (5.2). □

Appendix B contains some further results on total unimodularity that may be used to determine whether a given matrix is totally unimodular without computing determinants.

5.3 The scapegoat algorithm

5.3.1 Basic idea

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

$$\mathbf{Rx} = \mathbf{a}, \tag{5.4}$$

where each row of the $r \times v$-matrix $\mathbf{R}$ 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 i^{th} row of $\mathbf{R}$ 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. In this paper we take $\delta = 2$.

Similarly, the edit rules that take the form of a linear inequality can be written as

$$\mathbf{Qx} \geq \mathbf{b}, \tag{5.5}$$

where each edit rule is defined by a row of the $q \times v$-matrix $\mathbf{Q}$ together with a constant from $\mathbf{b} = [b_1, \ldots, b_q]'$. We assume until section 5.3.4 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.[3] 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 $r \times v$-matrix $\mathbf{R}$ satisfies $r \leq v$ and $\text{rank}(\mathbf{R}) = r$, 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 $\mathbf{R}$ is a totally unimodular matrix. So far we have found that matrices of balance edit rules used for structural business statistics at Statistics Netherlands are always of this type. A similar observation is made in 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 $\mathbf{R}$ for which $|\mathbf{r}_i'\mathbf{x} - a_i| > 2$ are removed from the matrix and the associated constants are removed from $\mathbf{a}$. We denote the resulting $r_0 \times v$-matrix by $\mathbf{R}_0$ and the resulting r_0-vector of constants by $\mathbf{a}_0$. It may happen that the record satisfies the remaining balance edit rules $\mathbf{R}_0\mathbf{x} = \mathbf{a}_0$, because it does not contain any rounding errors. In that case the algorithm stops here.

It is easy to see that if $\mathbf{R}$ satisfies the assumptions above, then so does $\mathbf{R}_0$. Hence $\text{rank}(\mathbf{R}_0) = r_0$ and $\mathbf{R}_0$ 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 in Fraleigh & Beauregard (1995, §2.2), or alternatively by performing a QR-decomposition with column pivoting, as discussed in Golub & van Loan (1996, §5.4). (How these methods work is irrelevant for our present purpose.) Since we want the choice of scapegoat variables and hence of columns to vary between records, a random permutation of columns is performed beforehand, yielding $\tilde{\mathbf{R}}_0$. The variables of $\mathbf{x}$ are permuted accordingly to yield $\tilde{\mathbf{x}}$.

Next, $\tilde{\mathbf{R}}_0$ is partitioned into two submatrices $\mathbf{R}_1$ and $\mathbf{R}_2$. The first of these is an $r_0 \times r_0$-matrix that contains the linearly independent columns, the second is an $r_0 \times (v - r_0)$-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 $\mathbf{R}_1$ and $\mathbf{R}_2$, respectively. Thus

$$\tilde{\mathbf{R}}_0\tilde{\mathbf{x}} = \mathbf{a}_0 \text{ becomes } [\mathbf{R}_1\ \mathbf{R}_2] \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \mathbf{a}_0.$$

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

[3] The name "scapegoat algorithm" was coined by Léander Kuijvenhoven (Statistics Netherlands).

original record and we are left with the system

$$\mathbf{R}_1\mathbf{x}_1 = \mathbf{a}_0 - \mathbf{R}_2\mathbf{x}_2 \equiv \mathbf{c}, \tag{5.6}$$

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

By construction the square matrix $\mathbf{R}_1$ is of full rank and therefore invertible. Thus (5.6) has the unique solution $\hat{\mathbf{x}}_1 = \mathbf{R}_1^{-1}\mathbf{c}$. 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 $\mathbf{R}_0\mathbf{x} = \mathbf{a}_0$. Note however that this is not possible without slightly changing the value of at least one variable from $\mathbf{x}_2$ too.

If $\mathbf{R}$ happens to be a totally unimodular matrix, this problem does not occur. In that case the determinant of $\mathbf{R}_1$ is equal to -1 or 1 and we know from Property 5.1 that $\hat{\mathbf{x}}_1$ is always integer-valued. In the remainder of this paper we assume that $\mathbf{R}$ is indeed totally unimodular.

5.3.2 An example

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

$$\left.\begin{aligned} x_1 + x_2 &= x_3 \\ x_2 &= x_4 \\ x_5 + x_6 + x_7 &= x_8 \\ x_3 + x_8 &= x_9 \\ x_9 - x_{10} &= x_{11} \end{aligned}\right\} \tag{5.7}$$

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

$$\mathbf{R} = \begin{bmatrix} 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 \end{bmatrix}. \tag{5.8}$$

Thus $\mathbf{a} = \mathbf{0}$ here. It is easily established that $\text{rank}(\mathbf{R}) = 5$. Moreover, we demonstrate in appendix B that $\mathbf{R}$ is totally unimodular.

We are given the following record that is inconsistent with respect to (5.7):

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 $\mathbf{R}_0$ is identical to $\mathbf{R}$.

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

$$\begin{array}{llllll} 1 \to 11, & 2 \to 8, & 3 \to 2, & 4 \to 5, & 5 \to 10, & 6 \to 9, \\ 7 \to 7, & 8 \to 1, & 9 \to 4, & 10 \to 3, & 11 \to 6. & \end{array}$$

This yields the following result:

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

It so happens that the first five columns of $\tilde{\mathbf{R}}$ are linearly independent. We obtain:

$$\mathbf{R}_1 = \begin{bmatrix} 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 1 & 0 \end{bmatrix}, \qquad \mathbf{R}_2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The scapegoat variables are those that correspond with the columns of $\mathbf{R}_1$, that is to say x_8, x_3, x_{10}, x_9 and x_4. For the remaining variables we fill in the original values from the record to calculate the constant vector $\mathbf{c}$:

$$\mathbf{c} = -\mathbf{R}_2 \begin{bmatrix} x_{11} \\ x_7 \\ x_2 \\ x_6 \\ x_5 \\ x_1 \end{bmatrix} = - \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -13 \\ 8 \\ 4 \\ 1 \\ 3 \\ 12 \end{bmatrix} = \begin{bmatrix} -16 \\ -4 \\ -12 \\ 0 \\ -13 \end{bmatrix}.$$

We obtain the following system in $\mathbf{x}_1$:

$$\mathbf{R}_1 \mathbf{x}_1 = \begin{bmatrix} 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_8 \\ x_3 \\ x_{10} \\ x_9 \\ x_4 \end{bmatrix} = \begin{bmatrix} -16 \\ -4 \\ -12 \\ 0 \\ -13 \end{bmatrix} = \mathbf{c}. \tag{5.9}$$

Rather than explicitly computing the inverse matrix of $\mathbf{R}_1$, we observe by re-arranging the rows and columns of $\mathbf{R}_1$ (and also the corresponding elements of

$\mathbf{c}$ and $\mathbf{x}_1$) that (5.9) is equivalent to:

$$\begin{bmatrix} -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 1 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_{10} \\ x_4 \\ x_9 \\ x_8 \\ x_3 \end{bmatrix} = \begin{bmatrix} -13 \\ -4 \\ 0 \\ -12 \\ -16 \end{bmatrix}. \tag{5.10}$$

(5.10) has the so-called upper triangular form and the solution may be read off by applying back substitution (Golub & van Loan (1996, §3.1) or Harville (1997, §11.8)). Working from back to front we find: $\hat{x}_3 = 16$, $\hat{x}_8 = 12$, $\hat{x}_9 = 28$, $\hat{x}_4 = 4$ and $\hat{x}_{10} = 41$.

When the original values of the variables in $\mathbf{x}_1$ are replaced by these new values, the record becomes consistent with respect to (5.7):

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

We remark that in this example it was not necessary to change the value of every variable in $\mathbf{x}_1$. In particular, x_4 and x_{10} have retained their original values.

5.3.3 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 what we would hope for. We will now derive an upper bound on the size of the adjustments, under the assumption that $\mathbf{R}$ is totally unimodular.

The *maximum norm* of a vector $\mathbf{v} = [v_1, \ldots, v_n]'$ is defined as

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

Associated to this vector norm is a matrix norm (Stoer & Bulirsch (2002, §4.4)):

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

with $\mathbf{A} = [a_{ij}]$ any $m \times n$-matrix. It holds that

$$|\mathbf{A}\mathbf{v}|_\infty \leq ||\mathbf{A}||_\infty |\mathbf{v}|_\infty, \tag{5.11}$$

for every $m \times n$-matrix $\mathbf{A}$ and every n-vector $\mathbf{v}$.[4]

[4] To see this, note that

$$|\mathbf{A}\mathbf{v}|_\infty = \max_{i=1,\ldots,m} \left| \sum_{j=1}^{n} a_{ij} v_j \right| \leq \max_{i=1,\ldots,m} \sum_{j=1}^{n} |a_{ij}| \cdot |v_j| \leq ||\mathbf{A}||_\infty |\mathbf{v}|_\infty.$$

We now turn to the scapegoat algorithm. By construction $\hat{\mathbf{x}}_1$ satisfies $\mathbf{R}_1\hat{\mathbf{x}}_1 = \mathbf{c}$. The original vector $\mathbf{x}_1$ satisfies $\mathbf{R}_1\mathbf{x}_1 = \mathbf{c}^*$, with $\mathbf{c}^* \neq \mathbf{c}$. Thus

$$\hat{\mathbf{x}}_1 - \mathbf{x}_1 = \mathbf{R}_1^{-1}\left(\mathbf{c} - \mathbf{c}^*\right). \tag{5.12}$$

It follows from (5.11) and (5.12) that

$$|\hat{\mathbf{x}}_1 - \mathbf{x}_1|_\infty \leq ||\mathbf{R}_1^{-1}||_\infty |\mathbf{c} - \mathbf{c}^*|_\infty \leq r_0 |\mathbf{c} - \mathbf{c}^*|_\infty, \tag{5.13}$$

where the last inequality is found by observing that Property 5.2 implies

$$||\mathbf{R}_1^{-1}||_\infty = \max_{i=1,\ldots,r_0} \sum_{j=1}^{r_0} |\left(\mathbf{R}_1^{-1}\right)_{ij}| \leq r_0.$$

We write $\hat{\mathbf{x}} = [\hat{\mathbf{x}}_1', \mathbf{x}_2']'$. By

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

we see that

$$|\mathbf{c} - \mathbf{c}^*|_\infty = |\mathbf{R}_0\mathbf{x} - \mathbf{a}_0|_\infty \in \{1, 2\}.$$

Plugging this into (5.13) we find that

$$|\hat{\mathbf{x}}_1 - \mathbf{x}_1|_\infty \leq 2r_0. \tag{5.14}$$

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 (5.14) is proportional to the order of $\mathbf{R}_1$ suggests that we should expect ever larger adjustments as the number of balance edit rules increases, which is somewhat worrying. In practice we find much smaller adjustments than $2r_0$, though. E.g. in the example from section 5.3.2 the maximal absolute difference according to (5.14) equals 10. In actual fact no value was changed by more than one unit.

Example. It is possible to construct a pathological example for which upper bound (5.14) becomes exact. Let the balance edit rules be given by $\mathbf{R}\mathbf{x} = \mathbf{0}$, with $\mathbf{R}$ the following $r \times (r+1)$-matrix:

$$\mathbf{R} = \begin{bmatrix}
1 & 1 & 1 & \cdots & 1 & 1 & 0 \\
0 & 1 & 0 & \cdots & 0 & 0 & 1 \\
0 & 0 & 1 & \cdots & 0 & 0 & 1 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & \cdots & 1 & 0 & 1 \\
0 & 0 & 0 & \cdots & 0 & 1 & -1
\end{bmatrix}.$$

Let the inconsistent record be

$$\mathbf{x} = \begin{bmatrix} -4(r-1)+2 \\ 4 \\ \vdots \\ 4 \\ 0 \\ -2 \end{bmatrix}.$$

Note that $\mathbf{R}\mathbf{x} = [-2, 2, \ldots, 2]'$, so all edit rules are violated and all violations qualify as rounding errors. If $x_1, x_2, \ldots, x_r$ are chosen as scapegoat variables, the matrix $\mathbf{R}_1$ consists of the first r columns of $\mathbf{R}$. It is easy to see that

$$\mathbf{R}_1^{-1} = \begin{bmatrix} 1 & -1 & -1 & \cdots & -1 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}.$$

Moreover, $\mathbf{c} = -\mathbf{R}_2 x_{r+1} = [0, 2, \ldots, 2, -2]'$, and

$$\mathbf{R}_1^{-1}\mathbf{c} = \begin{bmatrix} 1 & -1 & -1 & \cdots & -1 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \\ \vdots \\ 2 \\ -2 \end{bmatrix} = \begin{bmatrix} -2(r-3) \\ 2 \\ \vdots \\ 2 \\ -2 \end{bmatrix}.$$

The adjusted record is therefore

$$\hat{\mathbf{x}} = \begin{bmatrix} -2(r-3) \\ 2 \\ \vdots \\ 2 \\ -2 \\ -2 \end{bmatrix},$$

which, as the reader may verify, indeed satisfies all edit rules. Note that

$$\hat{x}_1 - x_1 = -2(r-3) + 4(r-1) - 2 = 2r,$$

which is the upper bound given by (5.14). △

We remark that even in this contrived example most adjustments are quite small: apart from x_1 no scapegoat variable is adjusted by more than two units. A more interesting view on the size of the adjustments may therefore 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 (5.12), we see that

$$|\,(\hat{\mathbf{x}}_1 - \mathbf{x}_1)_i\,| = \left| \sum_{j=1}^{r_0} \left(\mathbf{R}_1^{-1}\right)_{ij} (\mathbf{c} - \mathbf{c}^*)_j \right| \leq \sum_{j=1}^{r_0} |\,\left(\mathbf{R}_1^{-1}\right)_{ij}\,| \cdot |\,(\mathbf{c} - \mathbf{c}^*)_j\,|.$$

Using again that $|\mathbf{c} - \mathbf{c}^*|_\infty \leq 2$, we find

$$\frac{1}{r_0} \sum_{i=1}^{r_0} |\,(\hat{\mathbf{x}}_1 - \mathbf{x}_1)_i\,| \leq \frac{2}{r_0} \sum_{i=1}^{r_0} \sum_{j=1}^{r_0} |\,\left(\mathbf{R}_1^{-1}\right)_{ij}\,| \equiv 2\gamma(\mathbf{R}_1), \tag{5.15}$$

where $\gamma(\mathbf{R}_1) = \frac{1}{r_0} \sum_{i=1}^{r_0} \sum_{j=1}^{r_0} |\,\left(\mathbf{R}_1^{-1}\right)_{ij}\,|$. This gives an upper bound on the average adjustment size that is easy to evaluate.

Suppose that a set of balance edit rules (5.4) is given. Restricting ourselves to the case $r_0 = r$, we can compute $\gamma(\mathbf{R}_1)$ for various invertible $r \times r$-submatrices of $\mathbf{R}$ to assess the magnitude of the upper bound in (5.15). It is shown at the end of appendix B that there exist exactly $\det(\mathbf{R}\mathbf{R}')$ of these submatrices. In practice, this number is very large and it is infeasible to compute $\gamma(\mathbf{R}_1)$ for all matrices $\mathbf{R}_1$. In that case we can take a random sample of reasonable size, by repeatedly performing the part of the scapegoat algorithm that constructs an invertible submatrix.

Example. For the 5×11-matrix from section 5.3.2, $\det(\mathbf{R}\mathbf{R}') = 121$, so $\mathbf{R}$ has 121 invertible 5×5-submatrices. Since this number is not too large, we have evaluated $\gamma(\mathbf{R}_1)$ for all these matrices. The mean value of $\gamma(\mathbf{R}_1)$ turns out to be about 1.68, with a standard deviation of 0.39. Thus according to (5.15) the average adjustment size is bounded on average by 3.4, which is still somewhat higher than what we found in our example. △

In section 5.4 we look at the adjustments that occur in a real-world example. These turn out to be quite small.

We remark that for records that violate $\mathbf{R}_0\mathbf{x} = \mathbf{a}_0$ by no more than one unit per edit rule, the factor 2 in (5.14) and (5.15) may be replaced by 1 to obtain more accurate upper bounds.

5.3.4 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 non-negative. The scapegoat algorithm as described above does not take this into account. A non-negative variable might therefore be changed by the algorithm from 0 to -1, resulting in a new violation of an edit rule. We will now extend the algorithm to prevent this.

Suppose that in addition to the balance edit rules (5.4), the data also have to satisfy the inequalities (5.5). For a given record, we call a variable *critical* if it occurs in an inequality that (almost) becomes an exact equality when the current values of the survey variables are filled in. That is to say:

$$x_j \text{ is a critical variable } \Leftrightarrow q_{ij} \neq 0 \text{ and } 0 \leq \mathbf{q}_i'\mathbf{x} - b_i \leq \epsilon_i \text{ for some } i, \quad (5.16)$$

where $\mathbf{q}_i'$ denotes the i^{th} row of $\mathbf{Q}$ and ϵ_i marks the margin we choose for the i^{th} restriction. As a particular case, x_j is called critical if it must be non-negative and currently has a value between 0 and $\epsilon_{i(j)}$, with $i(j)$ the index of the row in $\mathbf{Q}$ corresponding to the non-negativity constraint for x_j. To prevent the violation of edit rules in (5.5), 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 $\mathbf{R}_0$), two separate permutations should be performed for the non-critical and the critical variables. The permuted columns associated with the non-critical 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 non-critical variables, provided the record contains a sufficient number of non-critical 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. We can build in a check at the end of the algorithm that rejects the solution if a new violation of an edit rule from (5.5) is detected. It then seems advantageous to let the record be processed again, because a different permutation of columns may yield a feasible solution after all. 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, in our opinion not too much effort should be put into this, because these parameters only affect a limited number of records. In the real-world example discussed in section 5.4 we found only a handful of infeasible solutions when $\epsilon_i = 0$ for all i.

5.3.5 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 the start of the editing process we prefer to leave this inconsistency as it is, to be resolved by either a subject-matter specialist or SLICE.

This can be achieved by removing the balance edit rules concerning these exceptional variables from $\mathbf{R}$. The variables should not be removed from $\mathbf{x}$ however, as they may also occur in edit rules in (5.5). (E.g. the *number of employees* times a constant is used to maximise the *total labour costs*.) The values of the exceptional variables therefore play a rôle in determining the critical variables. Note that it is not necessary to remove the exceptional variables from $\mathbf{x}$ anyway, as the columns in the new version of $\mathbf{R}$ that correspond with these variables contain only zeros.

5.3.6 Summary

The following plan summarises the scapegoat algorithm. The input consists of an inconsistent record $\mathbf{x}$ (v variables), a set of r balance edit rules $\mathbf{Rx} = \mathbf{a}$, a set of q inequalities $\mathbf{Qx} \geq \mathbf{b}$ and parameters ϵ_i ($i = 1, \ldots, q$) and K. Edit rules concerning exceptional variables (as described in section 5.3.5) have been removed from $\mathbf{Rx} = \mathbf{a}$ beforehand.

1. (a) Remove all edit rules for which $|\mathbf{r}_i'\mathbf{x} - a_i| > 2$. The remaining system is denoted as $\mathbf{R}_0\mathbf{x} = \mathbf{a}_0$. The number of rows in $\mathbf{R}_0$ is called r_0. If $\mathbf{R}_0\mathbf{x} = \mathbf{a}_0$ holds: stop.

   (b) Determine the critical variables according to (5.16).

2. (a) Perform random permutations of the critical and non-critical variables separately. Then permute the corresponding columns of $\mathbf{R}_0$ the same way. Put the non-critical variables and their columns before the critical variables and their columns.

   (b) Determine the r_0 leftmost linearly independent columns in the permuted matrix $\tilde{\mathbf{R}}_0$. Together, these columns are a unimodular matrix $\mathbf{R}_1$ and the associated variables form a vector $\mathbf{x}_1$ of scapegoat variables. The remaining columns are a matrix $\mathbf{R}_2$ and the associated variables form a vector $\mathbf{x}_2$.

   (c) Fix the values of $\mathbf{x}_2$ from the record and compute $\mathbf{c} = \mathbf{a}_0 - \mathbf{R}_2\mathbf{x}_2$.

3. Solve the system $\mathbf{R}_1\mathbf{x}_1 = \mathbf{c}$.

4. Replace the values of $\mathbf{x}_1$ by the solution just found. If the resulting record does not violate any new edit rules from $\mathbf{Qx} \geq \mathbf{b}$, we are done. If it does,

return to step 2a, unless this has been the K^{th} attempt. In that case the record is not adjusted.

In this description it is assumed that $\mathbf{R}$ is totally unimodular. Methods to test this assumption in practice are given in appendix B.

5.4 A real-world application

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 92 represent non-negativity constraints. After exclusion of edit rules that affect exceptional variables, 26 balance edit rules remain. The resulting 26×97-matrix $\mathbf{R}$ is totally unimodular, as can be determined very quickly[5] using the reduction method described in appendix B.

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

Table 6. 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

Table 6 summarises the results of applying the scapegoat algorithm. No new violations of inequalities were found. In fact, Table 6 shows that the adjusted data happen to satisfy four additional inequalities.

According to (5.14) the size of the adjustments made by the algorithm is theoretically bounded by $2 \times 26 = 52$, which is rather high. A random sample of $10,000$ invertible 26×26-submatrices of $\mathbf{R}$ was drawn to evaluate (5.15). The sample mean of $\gamma(\mathbf{R}_1)$ is about 1.89, with a standard deviation of 0.27. Thus the average adjustment size is bounded on average by 3.8. We remark that

[5] Note that it would be practically impossible to determine whether $\mathbf{R}$ is totally unimodular just by computing all the relevant determinants.

this value is only marginally higher than the one obtained for the much smaller restriction matrix from section 5.3.2.

Table 7 displays the adjustment sizes that were actually found for the wholesale data. These turn out to be very reasonable.

Table 7. Distribution of the adjustments (in absolute value).

magnitude	frequency
1	11,953
2	1,426
3	134
4	12
5	4
6	2

References

Bakker, T. (1997). Rounding in Tables. Report, Statistics Netherlands.

Camion, P. (1965). Characterization of Totally Unimodular Matrices. *Proceedings of the American Mathematical Society*, 16, 1068–1073.

de Jong, A. (2002). Uni-Edit: Standardized Processing of Structural Business Statistics in The Netherlands. Paper presented at the UN/ECE Work Session on Statistical Data Editing, 27-29 May 2002, Helsinki, Finland.

de Waal, T. (2002). Algorithms for Automatic Error Localisation and Modification. Paper presented at the UN/ECE Work Session on Statistical Data Editing, 27-29 May 2002, Helsinki, Finland.

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

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

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

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

Fraleigh, J. B. & Beauregard, R. A. (1995). *Linear Algebra*. Addison-Wesley, third edition.

Golub, G. H. & van Loan, C. F. (1996). *Matrix Computations*. The Johns Hopkins University Press, third edition.

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

Heller, I. & Tompkins, C. B. (1956). An Extension of a Theorem of Dantzig's. In H. W. Kuhn and A. W. Tucker (Ed.), *Linear Inequalities and Related Systems* (pp. 247–254). Princeton University Press, New Jersey.

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

Raghavachari, M. (1976). A Constructive Method to Recognize the Total Unimodularity of a Matrix. *Zeitschrift für Operations Research*, 20, 59–61.

Salazar-González, J. J., Lowthian, P., Young, C., Merola, G., Bond, S., & Brown, D. (2004). Getting the Best Results in Controlled Rounding with the

Least Effort. In J. Domingo-Ferrer and V. Torra (Ed.), *Privacy in Statistical Databases* (pp. 58–72). Springer, Berlin.

Stoer, J. & Bulirsch, R. (2002). *Introduction to Numerical Analysis*. Springer, New York, third edition.

Tamir, A. (1976). On Totally Unimodular Matrices. *Networks*, 6, 373–382.

A Uniqueness of the solution to (3.4)

In this appendix to section 3 we derive a sufficient condition for (3.4) to have at most one solution. We begin by slightly simplifying the notation. Instead of (3.4) we look at the following system:

$$\begin{cases} a_0 s_0 &= b_0 \\ a_1 s_1 &= b_1 t_1 \\ &\vdots \\ a_m s_m &= b_m t_m \\ a_n s_n &= a_0 s_0 + a_1 s_1 + \cdots + a_{n-1} s_{n-1} \\ (s_0, \ldots, s_n; t_1, \ldots, t_m) &\in \{-1, 1\}^{n+1} \times \{-1, 1\}^m \end{cases} \tag{A.1}$$

We now proof the following Lemma.

Lemma A.1 *Let $n \geq 2$ and $0 \leq m \leq n-1$. Suppose that $a_0, a_1, \ldots, a_n$ are all non-zero. If there do not exist $\lambda_0, \lambda_1, \ldots, \lambda_{n-1} \in \{-1, 0, 1\}$, not all equal to zero, such that $\lambda_0 a_0 + \lambda_1 a_1 + \cdots + \lambda_{n-1} a_{n-1} = 0$, then (A.1) has at most one solution.*

Proof. We proceed by supposing that there exist two different solutions $(s'_0, \ldots, s'_n; t'_1, \ldots, t'_m)$ and $(s''_0, \ldots, s''_n; t''_1, \ldots, t''_m)$, and show that this would lead to a contradiction. Observe that $s'_0 = s''_0$, because of the first equation in (A.1).

First suppose that $s'_n = s''_n$. Since the two solutions are different, there exists at least one $j \in \{1, \ldots, n-1\}$ such that $s'_j \neq s''_j$. By the last equation in (A.1), it holds that

$$\begin{aligned} 0 &= a_0 \left(s'_0 - s''_0\right) + a_1 \left(s'_1 - s''_1\right) + \cdots + a_{n-1} \left(s'_{n-1} - s''_{n-1}\right) \\ &= a_1 \left(s'_1 - s''_1\right) + \cdots + a_{n-1} \left(s'_{n-1} - s''_{n-1}\right), \end{aligned}$$

and hence that $\lambda_1 a_1 + \cdots + \lambda_{n-1} a_{n-1} = 0$, where $\lambda_j = (s'_j - s''_j)/2$ for $j = 1, \ldots, n-1$. Since $\lambda_j \in \{-1, 0, 1\}$ and since at least one λ_j is non-zero, this would contradict our assumption.

Apparently it must hold that $s'_n = -s''_n$. Again using the last equation in (A.1), we find:

$$\begin{aligned} 0 &= a_0 \left(s'_0 + s''_0\right) + a_1 \left(s'_1 + s''_1\right) + \cdots + a_{n-1} \left(s'_{n-1} + s''_{n-1}\right) \\ &= 2a_0 + a_1 \left(s'_1 + s''_1\right) + \cdots + a_{n-1} \left(s'_{n-1} + s''_{n-1}\right) \end{aligned}$$

Thus, taking $\mu_j = (s'_j + s''_j)/2$ for $j = 1, \ldots, n-1$ and $\mu_0 = 1$, it holds that $\mu_0 a_0 + \mu_1 a_1 + \cdots + \mu_{n-1} a_{n-1} = 0$. Again we would find a contradiction, since $\mu_j \in \{-1, 0, 1\}$ and μ_0 is certainly non-zero.

We conclude that (A.1) does not have two different solutions. In other words: if (A.1) has a solution, it is unique. □

Translated back to the profit-and-loss account, Lemma A.1 states that system (3.4) has a unique solution (provided that it has any solution at all) unless there exists some simple linear relation between the amounts $x_0, x_1, \ldots, x_{n-1}$, e.g. if two of these values happen to be equal. Obviously no such linear relation exists by design, so most records will satisfy this condition.

It should be noted that Lemma A.1 establishes a condition that is sufficient for uniqueness, but not necessary. This can be seen from the following example with $n = 2$ and $m = 1$. The reader may verify that the system

$$\begin{cases} s_0 &= 1 \\ -s_1 &= t_1 \\ 2s_2 &= s_0 - s_1 \\ (s_0, s_1, s_2; t_1) \in \{-1, 1\}^3 \times \{-1, 1\} \end{cases}$$

has the unique solution $(s_0, s_1, s_2; t_1) = (1, -1, 1; 1)$. This system is just (A.1) with $a_0 = 1$, $a_1 = -1$, $a_2 = 2$, $b_0 = 1$ and $b_1 = 1$. The condition from Lemma A.1 is not satisfied, since the equation $\lambda_0 - \lambda_1 = 0$ clearly has two non-trivial solutions for $\lambda_0, \lambda_1 \in \{-1, 0, 1\}$.

In Lemma A.1 we have made the assumption that $a_0, a_1, \ldots, a_n$ are all non-zero. In the context of the profit-and-loss account this means that $x_0, x_1, \ldots, x_n$ have to be non-zero. For records that occur in practice this is not always true, because not all items on the profit-and-loss account necessarily apply to all firms. If some $a_j = 0$, it is clear that (A.1) does not have a unique solution, because the value of s_j may then be chosen freely. (And if $1 \leq j \leq m$ the same holds for t_j.) On the other hand, put in the context of finding sign errors the value of s_j would then be irrelevant, since it corresponds to a zero-valued balance amount. Hence it seems reasonable to say that the sign error is uniquely defined in this situation, provided that the values of s_j and t_j that correspond to *non-zero* balance amounts are uniquely determined by (A.1). In the next Theorem we allow $a_1, \ldots, a_{n-1}$ to be zero-valued. However, we still assume that $a_0 \neq 0$ and $a_n \neq 0$, because these assumptions are used explicitly in the proof of Lemma A.1.

Theorem A.2 *Let $n \geq 1$ and $0 \leq m \leq n-1$. Suppose that, for a given record, $a_0 \neq 0$ and $a_n \neq 0$. If the equation $\lambda_0 a_0 + \lambda_1 a_1 + \cdots + \lambda_{n-1} a_{n-1} = 0$ does not have any solution $\lambda_0, \lambda_1, \ldots, \lambda_{n-1} \in \{-1, 0, 1\}$ for which at least one term $\lambda_j a_j \neq 0$, then if (A.1) has a solution, the sign error in the record is uniquely determined by it.*

Proof. The case $n = 1, m = 0$ is not covered by Lemma A.1. It can be shown in a straightforward manner that for this case (A.1) has at most one solution, provided that $a_0 \neq 0$ and $a_1 \neq 0$. (We omit this proof here.)

Hence we assume that $n \geq 2$. We also assume that $|a_k| = |b_k|$, because (A.1) has no solution otherwise. By striking out all equations from (A.1) for which both sides are zero, and by deleting all terms that are zero from the last equation, we obtain a reduced system of equations. In this reduced system all coefficients are non-zero. If at least one value from the original set $\{a_1, \ldots, a_{n-1}\}$ is non-zero, then we may apply Lemma A.1 to the reduced system and find that this system allows at most one solution. Otherwise, the reduced system corresponds to the case $n = 1, m = 0$, and we have already seen that this also means that there is at most one solution.

But for the purpose of detecting sign errors, the solution to the reduced system is all we are interested in, because the deleted variables all have zero values. Hence if the record contains a sign error, it is uniquely determined. □

Translating this back to system (3.4), we now know that the sign error is uniquely determined if

- $x_0 \neq 0$ and $x_n \neq 0$;
- the equation $\lambda_0 x_0 + \lambda_1 x_1 + \cdots + \lambda_{n-1} x_{n-1} = 0$ does not have any solution $\lambda_0, \lambda_1, \ldots, \lambda_{n-1} \in \{-1, 0, 1\}$ for which at least one term $\lambda_j x_j \neq 0$.

While testing these conditions using data collected for the wholesale structural business statistics 2001, we found that close to 97% of all records satisfied both.

Example. For the purpose of illustration we apply this result to an example record. The questionnaire for structural business statistics was redesigned at Statistics Netherlands in 2006, leading to (among many other things) a new structure for the profit-and-loss account as shown in Table 8. The edit rules are given by (3.3) with $n = 5$ and $m = 1$. The third column of Table 8 displays an example record. Setting up (3.4) for this record we find:

$$\begin{cases} 200s_0 &= 200 \\ 0s_1 &= 0t_1 \\ 140s_5 &= 200s_0 + 0s_1 + 0s_2 + 60s_3 + 0s_4 \\ (s_0, s_1, s_2, s_3, s_4, s_5; t_1) \in \{-1, 1\}^6 \times \{-1, 1\} \end{cases}$$

By striking out all the zero terms, this system can be reduced to:

$$\begin{cases} 200s_0 &= 200 \\ 140s_5 &= 200s_0 + 60s_3 \\ (s_0, s_3, s_5) \in \{-1, 1\}^3 \end{cases} \tag{A.2}$$

The equation $200\lambda_0 + 60\lambda_3 = 0$ clearly has no non-trivial solution for $\lambda_0, \lambda_3 \in \{-1, 0, 1\}$. Thus, according to Lemma A.1, there is a unique solution to (A.2). The solution is in fact given by $(s_0, s_3, s_5) = (1, -1, 1)$. This may be expanded to a solution of the original system by choosing the remaining variables arbitrarily. Regardless of how this is done, there is only one interpretation as a sign error: the value of x_3 should be changed from 60 to -60. △

Table 8. Structure of the profit-and-loss account in the structural business statistics from 2006 onward, with an example record.

variable	*full name*	*example*	*corrected*
$x_{0,r}$	operating returns	$2,100$	$2,100$
$x_{0,c}$	operating costs	$1,900$	$1,900$
x_0	operating results	200	200
$x_{1,r}$	provisions rescinded	0	0
$x_{1,c}$	provisions added	0	0
x_1	balance of provisions	0	0
x_2	book profit/loss	0	0
x_3	operating surplus	60	**−60**
x_4	exceptional result	0	0
x_5	pre-tax results	140	140

B On totally unimodular matrices

B.1 Recognising the total unimodularity of a matrix

It is infeasible to determine whether a given matrix is totally unimodular by applying the definition, unless the matrix is very small. In particular, the number of square submatrices of a matrix $\mathbf{R}$ from (5.4) that arises in practice is exceedingly large. E.g. if there are 100 variables that should conform to 20 balance edit rules (a realistic situation), the number of determinants to be computed equals $\sum_{k=2}^{20} \binom{100}{k}\binom{20}{k} \approx 3 \times 10^{22}$. (On the other hand, we are done as soon as we find one determinant that lies outside the range $\{-1, 0, 1\}$.) The purpose of this section is to provide feasible methods by which the assumption that $\mathbf{R}$ is totally unimodular may be tested in practice.

We begin by making the following observation which, although easy to prove, is not readily found in the literature.

Property B.1 *Let* $\mathbf{A}$ *be a matrix containing only elements* 0, 1 *and* -1 *that has the following form, possibly after a permutation of columns:*

$$\mathbf{A} = \left[\begin{array}{cc} \mathbf{B} & \mathbf{C} \end{array} \right],$$

where each column of $\mathbf{B}$ *contains at most one non-zero element. Then* $\mathbf{A}$ *is totally unimodular if and only if* $\mathbf{C}$ *is totally unimodular.*

Proof. We prove that if $\mathbf{C}$ is totally unimodular it follows that $\mathbf{A}$ is totally unimodular, the other implication being trivial. To do this we must show that the determinant of every square submatrix of $\mathbf{A}$ is equal to 0, 1 or -1. The proof works by induction on the order of the submatrix. The statement is clearly true for all 1×1-submatrices. Suppose that the statement holds for all $(k-1)\times(k-1)$-submatrices of $\mathbf{A}$ (with $k > 1$) and let $\mathbf{A}_k$ be any $k \times k$-submatrix of $\mathbf{A}$.

We may assume that $\mathbf{A}_k$ is invertible and also that it contains at least one column from $\mathbf{B}$, since otherwise there is nothing to prove. Let the j^{th} column of $\mathbf{A}_k$ come from $\mathbf{B}$. Since $\mathbf{A}_k$ is invertible, this j^{th} column must contain a non-zero element, say in the i^{th} row. If we denote this non-zero element by a_{ij}, we know from (5.1) that

$$\det \mathbf{A}_k = (-1)^{i+j} a_{ij} \det \mathbf{A}_{k-1},$$

where $\mathbf{A}_{k-1}$ is the $(k-1) \times (k-1)$-matrix found by removing the i^{th} row and the j^{th} column of $\mathbf{A}_k$. Since $|a_{ij}| = 1$ this means that $|\det \mathbf{A}_k| = |\det \mathbf{A}_{k-1}|$. It follows from the invertibility of $\mathbf{A}_k$ and from the induction hypothesis that $|\det \mathbf{A}_k| = 1$. □

In the proof of the next property we use the fact that $\mathbf{A}$ is totally unimodular if and only if $\mathbf{A}'$ is totally unimodular.

Property B.2 *Let* $\mathbf{A}$ *be a matrix containing only elements* 0, 1 *and* -1 *that has the following form, possibly after a permutation of rows:*

$$\mathbf{A} = \begin{bmatrix} \mathbf{B} \\ \mathbf{C} \end{bmatrix},$$

where each row of $\mathbf{B}$ *contains at most one non-zero element. Then* $\mathbf{A}$ *is totally unimodular if and only if* $\mathbf{C}$ *is totally unimodular.*

Proof. By Property B.1 we know that $\mathbf{A}'$ is totally unimodular if and only if $\mathbf{C}'$ is totally unimodular. □

These two properties sometimes allow us to determine whether a given matrix $\mathbf{A}$ is totally unimodular by considering a much smaller matrix. Instead of $\mathbf{A}$ it suffices, by Property B.1, to consider the submatrix $\mathbf{A}_1$ that consists of all columns of $\mathbf{A}$ containing two or more non-zero elements. Similarly, instead of $\mathbf{A}_1$ it suffices, by Property B.2, to consider the submatrix $\mathbf{A}_2$ that consists of all rows of $\mathbf{A}_1$ containing two or more non-zero elements. Next, it can happen that one or more columns of $\mathbf{A}_2$ contain less than two non-zero elements, so we may again apply Property B.1 and consider the submatrix $\mathbf{A}_3$ found by deleting these columns from $\mathbf{A}_2$. And it may then be possible to consider a further submatrix by another application of Property B.2. This iterative process may be continued until we either come across a matrix of which all columns and all rows contain two or more non-zero elements, or a matrix that is clearly totally unimodular (or clearly not).

Example. As an illustration we apply this *reduction method* to the 5×11-matrix defined in (5.8). By removing all columns with less than two non-zero elements, we obtain the following matrix:

$$\begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Next, we remove all rows containing less than two non-zero elements to find:

$$\begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & 1 & -1 \end{bmatrix}.$$

One more application of Property B.1 reduces this matrix to:

$$\begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$

This 2×1-matrix is clearly totally unimodular, and we immediately know that the original matrix is also totally unimodular. Note that we have obtained this result without computing a single determinant. △

In this example the reduction method is very successful, because the matrix under scrutiny happens to be sparse. In general we also need other methods to determine whether a given matrix is totally unimodular, without having to resort to computing determinants. The next Theorem completely describes the case where all columns contain no more than two non-zero elements. The proof can be found in Heller & Tompkins (1956).

Theorem B.3 (Hoffman, Gale) *Let* $\mathbf{A}$ *be a matrix containing only elements* 0, 1 *and* -1, *that satisfies these two conditions:*

1. *Each column contains at most two non-zero elements.*
2. *The rows of* $\mathbf{A}$ *may be partitioned into two subsets* M_1 *and* M_2, *such that the following holds for each column containing two non-zero elements: if the two elements have the same sign their rows are in different subsets and if the two elements have different signs their rows are in the same subset.*

Then $\mathbf{A}$ *is totally unimodular. Moreover, every totally unimodular matrix that satisfies the first condition also satisfies the second condition.*

Example. The following 4×4-matrix is totally unimodular:

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}.$$

This follows from Theorem B.3 by taking e.g. $M_1 = \{1, 2\}$ and $M_2 = \{3, 4\}$. Note that the reduction method has no effect here. △

As the formulation of the last part of Theorem B.3 suggests, there are totally unimodular matrices that do not satisfy these two conditions, because they contain a column of three or more non-zero elements. One example of such a matrix is:

$$\mathbf{M} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}.$$

Several characterisations of total unimodularity that are valid for all matrices may be found in Camion (1965) and Tamir (1976). These characterisations are insightful, but not easy to check in practice. A different approach is used by Raghavachari (1976). His result (Theorem B.4 below) relates the total unimodularity of one matrix to that of several smaller matrices, thus providing a recursive algorithm for determining whether a given matrix is totally unimodular.

Let $\mathbf{A} = [a_{ij}]$ be an $m \times n$-matrix containing only elements 0, 1 and -1. Suppose that $\mathbf{a}_j$, the j^{th} column of $\mathbf{A}$, contains $k > 2$ non-zero elements, and suppose that $a_{ij} \neq 0$. We obtain a matrix in which the j^{th} column has exactly one non-zero element, by adding the i^{th} row of $\mathbf{A}$ to all rows that have an element in $\mathbf{a}_j$ with value $-a_{ij}$ and by subtracting it from all other rows that have an element in $\mathbf{a}_j$ with value a_{ij}. Next, we delete the j^{th} column of this matrix to obtain an $m \times (n-1)$-matrix that we call $\mathbf{B}_i$. Working from the j^{th} column of $\mathbf{A}$, such a matrix can be constructed for every i with $a_{ij} \neq 0$. Consider the set $\mathcal{A}_j$ of these matrices.

Theorem B.4 (Raghavachari) *Let* $\mathbf{A}$ *be an* $m \times n$*-matrix containing only elements* 0, 1 *and* -1. $\mathbf{A}$ *is totally unimodular if and only if every* $m \times (n-1)$*-matrix in* $\mathcal{A}_j = \{\mathbf{B}_{i_1}, \ldots, \mathbf{B}_{i_k}\}$ *is totally unimodular.*

The proof of this Theorem can be found in Raghavachari (1976).

Example. For the purpose of illustration we apply Theorem B.4 to demonstrate the total unimodularity of $\mathbf{M}$. Working from the first column, the reader may verify that $\mathcal{A}_1$ consists of the following 3×2-matrices:

$$\mathbf{B}_1 = \begin{bmatrix} 1 & 1 \\ 0 & -1 \\ -1 & 0 \end{bmatrix}, \qquad \mathbf{B}_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ -1 & 1 \end{bmatrix}, \qquad \mathbf{B}_3 = \begin{bmatrix} 1 & 0 \\ 1 & -1 \\ 0 & 1 \end{bmatrix}.$$

These matrices are all totally unimodular, as may be seen by applying either the reduction method or Theorem B.3. It follows from Theorem B.4 that $\mathbf{M}$ is also totally unimodular. △

In general, Raghavachari's method produces a set of $m \times (n-1)$-matrices for which neither the reduction method nor Theorem B.3 can be applied. In that case we can apply Raghavachari's method for each of these matrices separately. Many recursions may be needed before we find a final set of smaller matrices of which it is immediately clear whether they are totally unimodular or not. Since the number of matrices in this set is exponential in the number of recursions, Raghavachari's method is infeasible for large matrices.

B.2 Invertible submatrices

In section 5.3.3 we mentioned the following:

Property B.5 *Let* $m \leq n$ *and let* $\mathbf{A}$ *be a totally unimodular* $m \times n$*-matrix. The number of invertible* $m \times m$*-submatrices of* $\mathbf{A}$ *equals* $\det(\mathbf{A}\mathbf{A}')$.

We will now prove this.

The proof uses the *Cauchy-Binet formula* (also known as the *Binet-Cauchy formula*). This formula states that, for an arbitrary $m \times n$-matrix $\mathbf{A}$ and an arbitrary $n \times m$-matrix $\mathbf{B}$, the following holds:

$$\det(\mathbf{AB}) = \sum_S \det(\mathbf{A}_{\bullet S}) \det(\mathbf{B}_{S\bullet}), \tag{B.3}$$

where the sum is over all $S \subset \{1, \ldots, n\}$ with $\#S = m$, $\mathbf{A}_{\bullet S}$ denotes the submatrix of $\mathbf{A}$ found by choosing the columns in S and $\mathbf{B}_{S\bullet}$ denotes the submatrix of $\mathbf{B}$ found by choosing the rows in S. A proof of the Cauchy-Binet formula may be found in Harville (1997, §13.8).

In particular, it follows from (B.3) that

$$\det(\mathbf{AA}') = \sum_S \det(\mathbf{A}_{\bullet S}) \det(\mathbf{A}'_{S\bullet}) = \sum_S \left(\det(\mathbf{A}_{\bullet S})\right)^2.$$

Now suppose that $\mathbf{A}$ is totally unimodular. Observe that

$$\left(\det(\mathbf{A}_{\bullet S})\right)^2 = \begin{cases} 0 & \text{if } \mathbf{A}_{\bullet S} \text{ is singular} \\ 1 & \text{if } \mathbf{A}_{\bullet S} \text{ is invertible} \end{cases}$$

So the number of invertible $m \times m$-submatrices of $\mathbf{A}$ is equal to $\det(\mathbf{AA}')$.