ALGORITHMS FOR ADJUSTING SURVEY DATA THAT FAIL BALANCE EDITS

Richard Sigman and Dennis Wagner, Bureau of the Census
Richard Sigman, ESMPD, Room 3108-4, Washington, D.C. 20233-6200

1. Introduction

In economic surveys and censuses, it is common that when data for particular items do not "balance"—i.e., they fail to satisfy one or more additivity conditions—that these data are considered to be unusable. This paper discusses algorithms for adjusting such unusable data so that they are usable. The algorithms we describe can be used in data editing and imputation. We consider the following types of additivity conditions:

**Simple-one dimensional (1d) balancing:**

\[ y = x_1 + x_2 + \ldots + x_n, \]

**Nested-one dimensional (1d) balancing:**

\[
\begin{align*}
y_1 &= x_{11} + x_{12} + \ldots + x_{1k(1)} \\
y_2 &= x_{21} + x_{22} + \ldots + x_{2k(2)} \\
&\vdots \\
y_m &= x_{m1} + x_{m2} + \ldots + x_{mk(m)} \\
z &= y_1 + y_2 + \ldots + y_m
\end{align*}
\]

**Two dimensional (2d) balancing:**

\[
\begin{array}{cccc}
x_{11} & x_{12} & \ldots & x_{1a} & r_1 \\
x_{21} & x_{22} & \ldots & x_{2a} & r_2 \\
& \vdots & \ddots & \vdots & \vdots \\
x_{m1} & x_{m2} & \ldots & x_{ma} & r_m \\
c_1 & c_2 & \ldots & c_a & z
\end{array}
\]

where \( r_i = \text{sum of row } i, \ c_j = \text{sum of column } j, \) and \( z = \Sigma r_i = \Sigma c_j \) is fixed.

What motivated us to look at balancing algorithms was the Census Bureau's development of an editing-and-imputation subsystem, called Plain Vanilla (PV), for processing its economic censuses. PV is so named because it provides basic editing and imputation capabilities that one can augment with survey-specific computer code (i.e. toppings) to suit one's particular tastes. Sigman (1997) and internal memoranda (available from the authors) describe PV in more detail.

This paper describes the PV balancing module. Section 2 describes simple-1d balancing in general. Section 3 describes one particular algorithm—the trim and adjust algorithm—for adjusting data that fail simple-1d balancing. Sections 4 and 5 discuss nested-1d balancing and 2d balancing, respectively.

2. Algorithms for simple-1d balancing

A simple-1d balancing complex is one in which two or more details (denoted \( x_i \)) add to a single total (denoted \( y \)). We assume \( y \geq 0 \) and \( x_i \geq 0 \), with \( y = 0 \) or \( x_i = 0 \) indicating either a valid zero or an item nonresponse. The following are examples of simple-1d balancing complexes:

**Example 1.** \( 100 = PRCNT1 + PRCNT2 + PRCNT3 \)

**Example 2.** \( CM = CP + CR + CF + EE + CW \)

In the first example the total is fixed at 100, whereas, in the second example the total is not fixed but is provided by the respondent.

The PV development team identified several situations in which a simple-1d balancing condition can fail to be satisfied. Table 1 lists these situations and briefly describes the associated adjustment procedures developed by the PV development team. These are explained in more detail in Table 2 and below, using the following notation:

- \( y \) = unadjusted total,
- \( x_i \) = \( i \)th unadjusted detail,
- \( R \) = residual of unadjusted data = \( y - \Sigma x_i \),
- \( R' \) = residual of adjusted data = \( y' - \Sigma x_i' \),
- \( y' \) = adjusted total,
- \( x_i' \) = \( i \)th adjusted detail,
- \( c_i \) = category average for \( i \)th detail, and
- \( f_{ij}/f_{2j} \) = category average for ratio of field \( f_{ij} \) to \( f_{2j} \).

When two or more adjustment methods can be used in a particular balancing situation, the user specifies which one(s) are to be used (and in what order) by preparing (prior to data processing) a specification file, called the PV script file.

2.1. One-dimensional raking of details to a total

One-dimensional-raked details are given by

\[ x_i' = \frac{y}{\sum x_i} x_i. \]

When the unadjusted details are integers, adjusted details that are also integers and add to the total can be obtained by the following integer-rounding algorithm:

**Step 1.** Calculate all the \( x_i' \) to one decimal place.

**Step 2.** For \( i = 1 \), round \( x_i' \) up or down to \( x_i'' \) depending on whether \( x_i' \) is 0.5 or <0.5.

**Step 3.** For \( i > 1 \), round

\[ u_i = x_i'' + \sum_{j=1}^{i-1} [x_j' - x_j'']. \]

up or down to \( x_i'' \) depending on \( u_i \geq 0.5 \) or <0.5.

If \(|R|/y\) is small, say less than 0.05, it is the general practice of subject-matter experts to rake details to a total. This practice has a sound statistical basis in situations in which the error in reporting a detail, \( x_i \), occurs at random and the variance of the random error in reporting \( x_i \), denoted \( \text{var}(x_i) \), is proportional to \( x_i \). Then, using the
method of Lagrange multipliers, it can be shown that the raked details, $x_\cdot'$, minimize the chi-square "statistic"

$$\chi^2 = \sum_i \frac{(x_i' - x_i)^2}{\text{var}(x_i)}$$

subject to the constraint $y = \sum x_i$. (See Deming, 1943, Chapter 5.)

2.2. Detection of rounding errors

In economic surveys, respondents are sometimes asked to round certain items to the nearest thousand. If a total is rounded but the details are not, or vice versa, then the total will not equal the sum of the details. This type of rounding error can be detected and corrected using the following algorithm, which determines if rounding the total or the details can reduce the residual to less than $ky$. (A small value of $k$ is usually used, say $k=0.05$).

- If $y$ is not fixed and $(y/1000) - k\sum x_i \leq y' \leq (y/1000) + k\sum x_i$, set $y' = y/1000$, and rake the $x_i$ to $y'$;
- Else if $(\sum x_i/1000) - ky \leq y \leq (\sum x_i/1000) + ky$, then set $x_i = x_i/1000$ and rake the $x_i$ to $y$.

3. Trim-and-adjust algorithm

If details do not add to a total, the trim-and-adjust algorithm changes the total and/or one or more details to obtain additivity. The algorithm consists of two major activities: trimming controlled by intervals, followed by data adjustments controlled by weights. The "trim" part of the algorithm examines the value of each item to determine if it falls outside the item’s adjustment interval. Values outside the adjustment interval are made equal to the value of the closest interval endpoint. The "adjust" part of the algorithm has the following properties:

P1. Values are adjusted so that $y = \sum x_i$.

P2. If an item’s value is adjusted, the adjusted value does not fall outside the item’s adjustment interval.

P3. Items with large weights are adjusted before items with small weights.

P4. If two items have the same weight, the item with the larger data value is adjusted before the item with smaller data value.

P5. The number of items adjusted in a balance complex is minimized subject to the constraints that properties P1 through P4 are satisfied.

Section 3.1 below defines the algorithm’s inputs and outputs. Section 3.2 describes the algorithm’s calculations and Section 3.3 applies the algorithm to an example. Section 3.4 discusses sources and roles of the controlling parameters (intervals and weights) and proposes some areas for further research and enhancement.

3.1. Inputs and Outputs

The following are the inputs to the algorithm:

- $y = \text{unadjusted total}$, $N = \text{number of details}$,
- $[L_0, U_0]$ = adjustment interval for total,
- $x = [x_1, x_2, \ldots, x_N]$ = vector of unadjusted detail values,
- $L = [l_1, l_2, \ldots, l_N]$ = vector of lower bounds for details,
- $U = [u_1, u_2, \ldots, u_N]$ = vector of upper bounds for details, and
- $W_0, W = \text{weights for } y \text{ and } x$, respectively, where items with large weights are adjusted before items with small weights. For example, for the balancing complex $y = q_1 + q_2$ with $a_1 < q_1 < b_1$, the inputs are $x = [q_1, q_2]$, $L = [a_1, b_1]$, and $U = [b_1, a_1]$.

The following are the outputs from the algorithm:

- $y' = \text{adjusted total}$, $f_0 = \text{adjustment flag for } y$ (1 = yes, 0 = no),
- $x' = [x_1', x_2', \ldots, x_N']$ = vector of adjusted detail values,
- $f = [f_1, f_2, \ldots, f_N]$ = vector of adjustment flags for details, and
- $err = \text{error indicator } (err=0 \text{ for no error})$.

3.2. Calculations

Define $x_0 = y$ and let $i=0, 1, \ldots, N$ index both the inputted weights $W_0, W_1, \ldots, W_N$ and the data $x_0, x_1, \ldots, x_N$. Sort the (weight, data) pairs in descending order, with the weight as the primary key and the data as the secondary key. Let $j=1, 2, \ldots, N+1$ index the sorted (weight, data) pairs, denoted $(W_j^*, x_j^*)$. Then there exists an index function $i(j)$ that satisfies $W_j^* = W_{j_0}$, $j_0 = 1, 2, \ldots, N+1$, $x_j^* = x_{i(j)}$, $j_0 = 1, 2, \ldots, N+1$, $W_j^* \geq W_{j+1}^*$, and if $W_j^* = W_{j+1}^*$ then $x_j^* \geq x_{j+1}^*$. The purpose of this sorting operation is to prioritize the data for the data adjustments following interval-based trimming--i.e., data at the beginning of the sorted list are adjusted before data at the end of the sorted list.

The algorithm obtains adjusted values by completing the spreadsheet shown in Figure 1. The inputs define the $z_j$ column, and the $z_j^*$ and $f_j$ columns produce the outputs--i.e., adjusted values and flags. The following are the definition of the columns, moving from left to right across the spreadsheet:

$$z_j = \begin{cases} y & i(j) = 0 \\ -x_{i(j)} & i(j) = 1 \\ resid_1 = \sum_{j=1}^{N+1} z_j \end{cases}$$

If $resid_1 = 0$, then return with $y' = y$, $f_0 = 0$, $x' = x$, $f = [0, 0, \ldots, 0]$ and $err = 0$. Otherwise, proceed with the algorithm.

$$L_j^* = \begin{cases} L_0 & i(j) = 0 \\ -U_{i(j)} & i(j) = 1 \\ L_j^* = \sum_{j=1}^{N+1} L_j^* \end{cases}$$

$$U_j^* = \begin{cases} U_0 & i(j) = 0 \\ -L_{i(j)} & i(j) = 1 \\ U_j^* = \sum_{j=1}^{N+1} U_j^* \end{cases}$$

The algorithm will be able to successfully adjust the data only if $U_j^* \geq 0 \geq L_j^*$. If this condition is not satisfied, return with $err = 1$. Otherwise, trim the data as follows:
and calculate the trimmed residual,

\[ \text{resid}_2 = \sum_{j=1}^{N+1} z_j^* . \]

If \( \text{resid}_2 = 0 \), then return with

\[ x'_{0j} = -z_j^* , \quad f_i = \begin{cases} 1 & x'_i = x_i \\ 0 & x'_i = x_i \end{cases} \]

\( y' = x'_* \), \( f = f_0 \) and \( \text{err} = 0 \). Otherwise, proceed to adjust the data so that the trimmed residual is made to vanish:

\[ I = \begin{cases} 1 & \text{resid}_2 > 0 \\ 0 & \text{resid}_2 < 0 \end{cases} \]

\[ e_j^* = \begin{cases} \max(U_j^* - z_j^*, 0) & \text{resid}_2 > 0 \\ \min(L_j^* - z_j^*, 0) & \text{resid}_2 < 0 \end{cases} \]

\[ e_j = I e_j^* + (1 - I) e_j^* \]

\[ c_j = \begin{cases} 0 & j = 0 \\ |e_j^*| & j = 1 \\ c_{j-1} + |e_j^*| & j \geq 2 \end{cases} \]

(NOTE: \( c_0 \) is used only to define \( e_1 \).)

\[ e_j = \begin{cases} e_j^* & |\text{resid}_2| > c_j \text{ or } \text{resid}_2 \geq c_{j-1} \\ |e_j^*| & \text{resid}_2 < c_{j-1} \end{cases} \]

\[ c_{j-1} = \begin{cases} |e_j^*| & \text{resid}_2 \geq c_{j-1} \\ c_{j-1} - |\text{resid}_2| & \text{resid}_2 < c_{j-1} \end{cases} \]

\[ e_j^* = z_j^* + e_j \]

\[ f_j^* = \begin{cases} 1 & z_i' = z_i \\ 0 & z_i' = z_i \end{cases} \]

Return with \( x_{0j}' = -z_j', \quad f_{0j} = f_0^* , \quad y' = x'_* \), and \( \text{err} = 0 \).

3.3. Example

The entries in the spreadsheet in Figure 1 are the calculation results for the following inputs: \( y = 10 \), \( [L_0, U_0]=[8, 11], \quad x=[4, 3, 7], \quad L=[4, 0, 1], \quad U=[4, 5, 5], \quad W_0=1.3, \quad W=[1.2, 1.4, 1.1] \). The outputs are \( y' = 10 \), \( f_0 = 0 \), \( x'=[4, 1, 5] \), \( y'=-x'_* \), and \( \text{err} = 0 \).

3.4. Discussion

We define the following residuals:

The weights \( W_0, W_1, \ldots, W_n \), the lower bounds \( L_0, L_1, \ldots, L_n \), and the upper bounds \( U_0, U_1, \ldots, U_n \) control the trim-and-adjust algorithm. Items with high weights are adjusted before items with low weights. If two items have the same weight, the item with the larger value is changed before the item with the lower value. We considered using raking when items have equal weights but decided against this because changing the larger value results in a smaller average of the relative absolute change (when averaging over equally weighted items) compared to using raking.

It is useful to make the weights depend on editing actions that occur prior to calling the trim-and-adjust algorithm. For example, in PV if an item is fixed or if the user "goldplates" it—that is, specifies that the item cannot be changed—its weight is set to 0.0, whereas if an item contains an imputed value (produced by other PV editing actions) 100.00 is added to the user-supplied weight. In PV, it is not necessary, however, for the user to supply weights. If the user does not specify a weight for an item, its initial value is 1.0.

It is also useful to have the upper and lower adjustment bounds depend on editing actions that occur prior to calling the trim-and-adjust algorithm. In PV, we set an item’s upper and lower bounds equal to the value of the item, if the item is fixed or goldplated. If a user of PV does not supply a lower bound, the default is 0.0, and if an upper bound is not supplied, the default is \( \max(x, \sum x_i) \). The adjustment bounds for an item can also depend on the values of other items and/or on parameters determined from historical data. For example, in PV the following methods are available for defining the upper and lower adjustment bounds for item \( f_i \):

- **Method (1):** bound = constant,
- **Method (2):** bound = (constant) \( f_i \),
- **Method (3):** bound = (constant) \( f_i^{(0)} / f_i^{(10)} \) \( f_i \), and
- **Method (4):** bound = (constant) \( f_i^{(0)} / f_i^{(10)} \) \( f_i \),

where item \( f_i \) is either fixed or goldplated. If a user of PV does not supply a lower bound, the default is 0.0, and if an upper bound is not supplied, the default is \( \max(x, \sum x_i) \). The adjustment bounds for an item can also depend on the values of other items and/or on parameters determined from historical data.

4. Algorithm for nested-1d balancing

We will say that a set of data items constitutes a nested-1d balancing complex if in order for it to be balanced it must satisfy all of the following equations:

\[ y_i' = \sum_{j=1}^{m_i} x_{ij}', \quad i = 1, \ldots, m \]

\[ z' = \sum_{i=1}^{n} y_i' \]

where \( x_{ij}' \), \( j = 1, \ldots, m_i \), \( y_i' \), and \( z' \) are the values after editing of the values \( x_{ij} \), \( j = 1, \ldots, m_i \), \( y_i \), and \( z \), respectively. We will refer to the \( x_{ij} \) as sub-details, to the \( y_i \) as details, and to \( z \) as the grand total. If the equation for \( y_i' \) is satisfied for \( i = 1 \), we will say that the sub-details for detail \( i^* \) are balanced, and if equation for \( z' \) is satisfied we will say that the details are balanced.

We define the following residuals:
If all of the defined residuals are zero, the complex does not require editing because all the details and sub-details are in balance. Different balancing situations result from various combinations of non-zero residuals. Representing a nested-1d balancing complex as a two-colored network--see Figure 2, in which "solid" and "dashed" are the two colors--indicates the combinations of possible non-zero residuals. The network's nodes correspond to additivity conditions. Flows along the solid-line arcs correspond to the values of details, sub-details, and the grand total; and flows along the dotted-line arcs correspond to the values of the residuals. (See Cox, 1995, for additional details on using networks to represent additivity relationships.) Whether or not z, the grand total, is a fixed value (e.g., 100%) determines additional balancing situations.

Table 3 lists the nested-1d balancing situations. In the longer version of this paper, we include figures representing five subsets of these situations in which we have eliminated selected zero-value residuals from the networks. Since balancing corresponds to forcing flows in dotted-line arcs to zero, these network representations suggest items that should be changed and others that should remain unchanged when performing balancing.

The last column of Table 3 indicates the items to be changed in each balancing situation, using the following procedures:

4.1. Adjusting only xu

For each i such that yi ≠ ∑ xij, fix yi and use simple-1d balancing algorithms to adjust the xu.

4.2. Adjusting only yj

For yj ≠ ∑ xij, calculate yj' = ∑ xij.

4.3. Adjusting only z

Set z = ∑ yj.

4.4. Adjusting xij and yj, when all Rij = 0

The following are available options:

- If |Rij| ≤ z is small, use common-factor raking: 
  yi' = β yi, and xij' = β xij, where β = z / ∑ xij.
- Set complex unusable.
- Set detail-level NSK to Rij.
- Use historical data to impute all details and sub-details--first impute details (followed by raking), then impute sub-details (followed by raking).

4.5. Adjusting xij and yj, when some Rij ≠ 0

The following are available options:

- If |Rij| ≤ z is small and |Rij| < |Rij|, take the yj to z.
  Then use the simple-1d balancing with yj fixed to balance details that are not the sum of their sub-details.
- If |Rij| ≤ z is small and if |Rij| < |Rij|, then take the xij to z, and set yj' equal to the sum of the associated raked xij.
  - Set complex unusable.
- Execute the one-dimensional trim-and-adjust algorithm m*+1 times, where m* is the number of details with out-of-balance sub-details. The input data for one of the trim-and-adjust operations are z and the yj. The data for the other m* trim-and-adjust operations are the yj and xij values associated with the m* details that have out-of-balance sub-details. Perform the m*+1 trim-and-adjust operations in the order of increasing absolute value of the associated residuals. After each trim-and-adjust operation is performed, recalculate all the remaining order-determining residuals. Set weights to zero for items associated with completed trim-and-adjust operations.

5. Algorithm for 2d balancing

We will say that a set of data items constitutes a 2d balancing complex if in order for it be balanced it must satisfy all of the following equations:

\[
R_i = y_i - \sum_j x_{ij} = \text{residual for sub-details of detail i}
\]

\[
R_{CO} = z - \sum_i y_i = \text{residual for the sum of the details}
\]

\[
R_{CO} = z - \sum_j x_{ij} = \text{grand residual}
\]

where \(x_{ij}'\), \(r_{ij}'\), and \(c_j'\) are the values after-editing of the values \(x_{ij}\), \(r_{ij}\), and \(c_j\), respectively. We will refer to the \(x_{ij}\) as cell entries, to the \(r_{ij}\) as row sums, to the \(c_j\) as column sums, and to \(z\) as the grand total. In addition, we will use the symbols \(x_{ij}'\), \(r_{ij}'\), and \(c_j'\) to denote data based on \(x_{ij}\), \(r_{ij}\), and \(c_j\), respectively, that are in turn used to obtain \(x_{ij}'\), \(r_{ij}'\), and \(c_j'\), respectively. We make the following assumptions:

- \(z\), the grand total, is fixed--i.e., \(z = z\) -- and is a non-negative integer,
- Balance equations are strictly additive (i.e., cells cannot be subtracted),
- All rows contain the same number of cell entries,
- All columns contain the same number of cell entries,
- \(x_{ij}\), \(r_{ij}\), and \(c_j\) are non-negative, and
- \(x_{ij}'\), \(r_{ij}'\), and \(c_j'\) are non-negative integers.

The following procedure performs 2d balancing:

**Step 1.** Calculate:

\[R_{ROW} = z - \sum_i r_i = \text{residual for the row sums.}\]

If \(R_{ROW} \neq 0\), rake the row sums: \(r_i' = r_i (z / \Sigma r_i)\).

If \(R_{ROW} = 0\), then \(r_i = r_i'\).

**Step 2.** Calculate:

\[R_{COL} = z - \sum_j c_j = \text{residual for the column sums.}\]

If \(R_{COL} \neq 0\), rake the column sums: \(c_j' = c_j (z / \Sigma c_j)\).

If \(R_{COL} = 0\), then \(c_j = c_j'\).

**Step 3.** Calculate:

\[R_{(rew)} = r_i - \sum_j x_{ij} = \text{residual for cells in row i}\]
\[ R_j^{(col)} = c_j \sum_{i} x_{ij} \] for \( i = 1, 2, \ldots, m \), and \( j = 1, 2, \ldots, n \). If all of the \( R_j^{(row)} \) and \( R_j^{(col)} \) are zero, then \( x_{ij} = x_{ij} \). If any of the \( R_j^{(row)} \) or \( R_j^{(col)} \) are non-zero, calculate adjusted cell entries, denoted \( x'_{ij} \), by performing two-dimensional raking of the \( x_{ij} \), via the iterative-proportional-fitting (IPF) algorithm. The IPF algorithm alternates between raking cell entries to row sums and raking cell entries to column sums. Oh and Scheuren (1978) provide an extensive bibliography on the IPF algorithm and Bishop, Fienberg, and Holland (1975) provide additional discussion. We first attempt to achieve additivity by changing only non-zero cell entries. If this is impossible, we restore the non-zero \( x_{ij} \), change the cells containing zero to 1.0, and rerun the IPF algorithm. Fagan and Greenberg (1984) discuss the properties of two-dimensional raking when some of the cell entries are zero, and Fagan and Greenberg (1985) discuss methods in addition to two-dimensional raking for performing 2d balancing.

**Step 4.** If any of the \( x'_{ij} \), \( r'_i \), or \( c'_j \) are non-integer, use controlled rounding to obtain \( x''_{ij} \), \( r'_i \), and \( c''_j \). Controlled rounding changes each non-integer data value to either the integer immediately above the value or the integer immediately below the value in such a way that all additive relationships are preserved. (See Cox and Ernst (1982) and Causey, Cox, and Ernst (1985).) If all of the \( x''_{ij} \), \( r'_i \), or \( c'_j \) are integers, then \( x''_{ij} = x'_{ij} \), \( r'_i = r'_i \), and \( c''_j = c'_j \).

**Acknowledgments**

We gratefully acknowledge the contributions of the PV Balancing Mini-Team—consisting of Richard Graham, Barbara Lambert, Rich Sterner, and the authors—who identified the need for the above algorithms, proposed algorithm logic and imputation options, and developed PV script files to test the algorithms. Barbara Lambert programmed the PV balancing module, and Brian Greenberg and Jim Fagan contributed ideas and software for two-dimensional balancing.

**References**


Figure 1. Spreadsheet for Trim-and-Adjust Algorithm. (Cells entries are calculation results for § 3.3 example.)

<table>
<thead>
<tr>
<th>j</th>
<th>1(j)</th>
<th>( W^* )</th>
<th>( z_j )</th>
<th>( L_j^* )</th>
<th>( U_j^* )</th>
<th>( z_j^* )</th>
<th>( e_j^+ )</th>
<th>( e_j^- )</th>
<th>( e_j^* )</th>
<th>( c_j )</th>
<th>( e_j )</th>
<th>( z_j' )</th>
<th>( f_j^* )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1.4</td>
<td>-3</td>
<td>-5</td>
<td>0</td>
<td>-3</td>
<td>3</td>
<td>-2</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1.3</td>
<td>10</td>
<td>8</td>
<td>11</td>
<td>10</td>
<td>1</td>
<td>-2</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1.2</td>
<td>-4</td>
<td>-4</td>
<td>-4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>-4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>1.1</td>
<td>-7</td>
<td>-5</td>
<td>-5</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>8</td>
<td>0</td>
<td>-5</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>resid</th>
<th>( L^* )</th>
<th>( U^* )</th>
<th>resid</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4</td>
<td>-6</td>
<td>6</td>
<td>-2</td>
<td>0</td>
</tr>
</tbody>
</table>

580
Figure 2. Network representation of a nested-1d balancing complex.

Table 1. Simple-1d balancing--Situations

<table>
<thead>
<tr>
<th>Situation</th>
<th>PV Procedure (Table 2 describes words in bold)</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>balance condition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$y = \sum x_i$</td>
<td></td>
<td>$y' = 0$</td>
</tr>
<tr>
<td>$y=0$ and $\sum x_i = 0$ fixed</td>
<td>$\text{ZERO_SET}$ each $x'_i = 0.$</td>
<td></td>
</tr>
<tr>
<td>not fixed</td>
<td>$\text{YSUMX}$</td>
<td>$y' = \sum x_i, x'_i = x_i$</td>
</tr>
<tr>
<td>$y=0$ and $\sum x_i = 0$ fixed</td>
<td>Same as $y=0,$ $\sum x_i = 0,$ and $y' = \sum x_i$</td>
<td>$y' = 0.$</td>
</tr>
<tr>
<td>not fixed</td>
<td>$\text{ZERO_SET}$ $y' = 0.$</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Simple-1d-balancing--Adjustment methods

<table>
<thead>
<tr>
<th>Abbr</th>
<th>Description</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZERO_SET</td>
<td>Set to zero</td>
<td>$y' = y,$ $x'_i = 0,$ or $y' = 0,$ $x'_i = x_i$</td>
</tr>
<tr>
<td>YSUMX</td>
<td>Set $y$ to $\sum x_i$</td>
<td>$y' = \sum x_i, x'_i = x_i$</td>
</tr>
<tr>
<td>NSK</td>
<td>Not specified by kind</td>
<td>$y' = y,$ $x'_i = x_i,$ $&lt;\text{NSK}&gt;$</td>
</tr>
<tr>
<td>UNUSABLE</td>
<td>Mark data unusable</td>
<td>$y' = \text{&lt;unusable&gt;}, x'_i = \text{&lt;unusable&gt;}$</td>
</tr>
<tr>
<td>REPLACE-V</td>
<td>Impute using category-average ratio</td>
<td>$y' = y,$ $x'_i = [(x_i/y)_R] y$</td>
</tr>
<tr>
<td>REPLACE-H</td>
<td>Impute using historic data</td>
<td>$y' = y,$ $x'_i = (y/(x'_i/|x_i|))y$</td>
</tr>
<tr>
<td>ROUND</td>
<td>Round $y$ or $x_i$ by 1000 if $</td>
<td>R/y</td>
</tr>
<tr>
<td>RAKE</td>
<td>Rake details to total if $</td>
<td>R/y</td>
</tr>
<tr>
<td>TRIM</td>
<td>Trim-and-adjust algorithm</td>
<td>See § 3.</td>
</tr>
</tbody>
</table>

Table 3. Nested-1d balancing situations

<table>
<thead>
<tr>
<th>z</th>
<th>all fixed</th>
<th>$R^{\omega} = 0$</th>
<th>$R^{\omega} = 0$</th>
<th>$R^{\omega} = 0$</th>
<th>Adjust</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>none (balanced)</td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>n.a. (impossible)</td>
<td>no</td>
</tr>
<tr>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>n.a. (impossible)</td>
<td>no</td>
</tr>
<tr>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>$x_i$ (See § 4.1)</td>
<td>no</td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>$z$ (See § 4.3)</td>
<td>no</td>
</tr>
<tr>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>$x_i$ (See § 4.1)</td>
<td>no</td>
</tr>
<tr>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>$y_i$ (See § 4.2)</td>
<td>no</td>
</tr>
<tr>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>$x_{gi}, y_i$ (See § 4.5)</td>
<td>yes</td>
</tr>
<tr>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>none (balanced)</td>
</tr>
<tr>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>n.a. (impossible)</td>
<td>yes</td>
</tr>
<tr>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>n.a. (impossible)</td>
<td>yes</td>
</tr>
<tr>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>$x_i$ (See § 4.1)</td>
<td>yes</td>
</tr>
<tr>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>$y_i$ (See § 4.2)</td>
<td>yes</td>
</tr>
<tr>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>$x_{gi}, y_i$ (See § 4.5)</td>
<td>yes</td>
</tr>
<tr>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>$y_i$ (See § 4.2)</td>
<td>yes</td>
</tr>
<tr>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>$y_i$ (See § 4.5)</td>
<td>yes</td>
</tr>
</tbody>
</table>