ALGORITHMS FOR ADJUSTING SURVEY DATA THAT FAIL BALANCE EDITS[1]

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:

$$y = x_1 + x_2 + \dots + x_n,$$

Nested-one dimensional (1d) balancing:

$$y_1 = x_{11} + x_{12} + \dots + x_{1n(1)}$$
$$y_2 = x_{21} + x_{22} + \dots + x_{2n(2)}$$
$$\vdots$$
$$y_m = x_{m1} + x_{m2} + \dots + x_{mn(m)}$$
$$z = y_1 + y_2 + \dots + y_m$$

Two dimensional (2d) balancing:

x_{11}	x_{12}	...	x_{1n}	r_1
x_{21}	x_{22}	...	x_{2n}	r_2
...	...	...	...	...
x_{m1}	x_{m2}	...	x_{mn}	r_m
c_1	c_2	...	c_n	z

where r_i = sum of row i, c_j = 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 ones 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 = ith unadjusted detail,
R = residual of unadjusted data = $y - \sum x_i$
R' = residual of adjusted data = $y' - \sum x_i'$
y' = adjusted total,
x_i' = ith adjusted detail,
<NSK> = not-specified-by-kind variable
$y^{(h)}$ = historic value for total,
$x_i^{(h)}$ = historic value for ith detail, and
$(f_1/f_2)_{\bar{R}}$ = category average for ratio of field f_1 to f_2.

When two or more adjustment methods can be used in a particular balancing situation, the user specifies which one(s) are to 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 $x_i' = (y/\sum x_i)x_i$. 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_1' up or down to x_1'' depending on whether x_1' 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 $var(x_i)$, is proportional to x_i. Then, using the

[1] This paper reports the general results of research undertaken by Census Bureau staff. The views expressed are attributable to the authors and do not necessarily reflect those of the Census Bureau.

method of Lagrange multipliers, it can be shown that the raked details, x_i', minimize the chi-square "statistic"

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

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 $(y/1000)-(k\sum x_i)\leq \sum x_i \leq (y/1000)+(k\sum x_i)$, 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 = unadjusted total, N = number of details,

$[L_0, U_0]$ = adjustment interval for total,

$x = [x_1,x_2,...,x_N]$ = vector of unadjusted detail values,

$L = [L_1,L_2,...,L_N]$ = vector of lower bounds for details,

$U=[U_1,U_2,...,U_N]$ = vector of upper bounds for details, and

W_0, W = weights, for y 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 $a_i\leq q_i \leq b_i$, the inputs are $x=[q_1,-q_2]$, $L=[a_1,-b_2]$, and $U=[b_1,-a_2]$.

The following are the outputs from the algorithm:

y' = adjusted total, f_0 = adjustment flag for y (1=yes, 0=no),

$x' = [x_1',x_2',...,x_N']$ = vector of adjusted detail values,

$f=[f_1,f_2,...,f_N]$ = vector of adjustment flags for details, and

err = error indicator (err=0 for no error).

3.2. Calculations

Define $x_0 = y$ and let i=0,1,...,N index both the inputted weights $W_0,W_1,...,W_N$ and the data $x_0,x_1,...,x_N$. 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,...,N+1 index the sorted (weight,data) pairs, denoted (W_j^*,x_j^*). Then there exists an index function $i(j)$ that satisfies

$$W_j^* = W_{i(j)} \quad j=1,2,...N+1,$$
$$x_j^* = x_{i(j)} \quad j=1,2,...N+1,$$
$$W_1^* \geq W_2^* \geq ... \geq W_{N+1}^*$$

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:

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

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

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

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

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

$$z_j^* = \begin{cases} z_j & L_j^* \leq z_j \leq U_j^* \\ U_j^* & z_j > U_j^* \\ L_j^* & z_j < L_j^* \end{cases}$$

and calculate the trimmed residual,

$$resid_2 = \sum_{j=1}^{N+1} z_j^* \quad .$$

If $resid_2$=0, then return with

$$x'_{i(j)} = -z_j^* \qquad f_i = \begin{cases} 1 & x_i' \neq x_i \\ 0 & x_i' = x_i \end{cases}$$

$y'=-x_o'$, $f=f_o$ and err=0. Otherwise, proceed to adjust the data so that the trimmed residual is made to vanish:

$$I = \begin{cases} 1 & resid_2 > 0 \\ 0 & resid_2 < 0 \end{cases} \qquad \begin{aligned} e_j^+ &= \max(U_j^* - z_j^*, 0) \\ e_j^- &= \min(L_j^* - z_j^*, 0) \\ e_j^* &= I e_j^- + (1-I) e_j^+ \end{aligned}$$

$$c_j = \begin{cases} 0 & j=0 \\ |e_1^*| & j=1 \\ c_{j-1} + |e_j^*| & j \geq 2 \end{cases}$$

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

$$e_j = \begin{cases} e_j^* & |resid_2| > c_j \text{ or } |resid_2| = c_j > c_{j-1} \\ e_j^* \dfrac{|resid_2| - c_{j-1}}{|e_j^*|} & c_j > |resid_2| \geq c_{j-1} \\ 0 & c_{j-1} > |resid_2| \text{ or } |resid_2| = c_j = c_{j-1} \end{cases}$$

$$z_j' = z_j^* + e_j \qquad f_j^* = \begin{cases} 1 & z_i' \neq z_i \\ 0 & z_i' = z_i \end{cases}$$

Return with $x_{i(j)}' = -z_j'$, $f_{i(j)} = f_j^*$, $y'=-x_o'$, and 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], x=[4,3,7], L=[4,0,1], U=[4,5,5], W_0=1.3, W=[1.2, 1.4, 1.1]. The outputs are $y'=10$, f_0=0, x'=[4,1,5], f=[0,1,1], err=0.

3.4 Discussion

The weights (W_0, W_1, ..., W_N), the lower bounds (L_0, L_1, ..., L_N), and the upper bounds (U_0, U_1, ..., 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 $\max(y, \sum x_i)$. 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_1:

Method (1): bound = constant,
Method (2): bound = (constant)f_2,
Method (3): bound = (constant) $(f_1/f_2)_{\bar{R}}$ f_2, and
Method (4): bound = (constant) $(f_1^{(h)}/f_2^{(h)}) f_2$,

where item f_2 is either fixed or is not involved in the balancing complex. Method (2) can be used to ensure that the value outputted by the trim-and-adjust algorithm for f_1 satisfies the ratio edit $L_{12} < f_1/f_2 < U_{12}$.

The following are two areas for further research and enhancement of the trim-and-adjust algorithm:

- Developing a capability for "learning" the appropriate weights and adjustment bounds from results of analyst review of the algorithms outputs, and
- For simultaneously satisfying ratio and balance edits, compare Method (2) with the approach proposed by Draper and Winkler (1997).

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:

$$y'_i = \sum_{j=1}^{m_i} x'_{ij}, \ i=1,...,m \qquad z' = \sum_{i=1}^{m} y'_i$$

where x'_{ij}, j=1,2,...,m_i, y'_i, and z' are the values after-editing of the values x_{ij}, j=1,2,...,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=i*, 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:

$R_i^{(d)} = y_i - \sum_j x_{ij}$ = residual for sub-details of detail i

$R^{(D)} = z - \sum_i y_i$ = residual for the sum of the details

$R^{(g)} = z - \sum_{ij} x_{ij}$ = grand residual

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 x_{ij}.

For each i such that $y_i \neq \sum_j x_{ij}$, fix y_i and use simple-1d balancing algorithms to adjust the x_{ij}.

4.2. Adjusting only y_i.

For $y_i \neq \sum x_{ij}$, calculate $y_i' = \sum_j x_{ij}$.

4.3. Adjusting only z

Set $z = \sum y_i$.

4.4. Adjusting x_{ij} and y_i, when all $R_i^{(d)} = 0$.

The following are available options:

- If $|R^{(D)}|/z$ is small, use common-factor raking: $y_i' = \beta y_i$ and $x_{ij}' = \beta x_{ij}$, where $\beta = z / \sum_j x_{ij}$.
- Set complex unusable.
- Set detail-level NSK to $R^{(D)}$.
- 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 x_{ij} and y_i, when some $R_i^{(d)} \neq 0$.

The following are available options:

- If $|R^{(D)}|/z$ is small and $|R^{(D)}| < |R^{(g)}|$, rake the y_i to z . Then use the simple-1d balancing with y_i fixed to balance details that are not the sum of their sub-details.
- If $|R^{(g)}|/z$ is small and if $|R^{(g)}| < |R^{(D)}|$, then rake the x_{ij} to z, and set y'_i equal to the sum of the associated raked x_{ij}
- 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 y_i . The data for the other m^* trim-and-adjust operations are the y_i and x_{ij} 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:

$$r''_i = \sum_{j=1}^{n} x''_{ij} \quad i=1,\dots,m \qquad c''_j = \sum_{i=1}^{m} x''_{ij} \quad j=1,\dots,n \qquad z'' = \sum_{i=1}^{m} r''_i = \sum_{j=1}^{n} c''_j$$

where x''_{ij}, r''_i, c''_j, and z'' are the values after-editing of the values x_{ij} , r_i, c_j, and z, respectively. We will refer to the x_{ij} as *cell entries*, to the r_i 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'_i, and c'_j to denote data based on x_{ij} , r_i, and c_j, respectively, that are in turn used to obtain x''_{ij}, r''_i, 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_i, and c_j, are non-negative, and
- x''_{ij} , r''_i, and c''_j are non-negative integers.

The following procedure performs 2d balancing:

Step 1. Calculate

$R^{(ROW)} = z - \sum_i r_i$ = 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

$R^{(COL)} = z - \sum_j c_i$ = 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

$R_i^{(row)} = r'_i - \sum_j x_{ij}$ = residual for cells in row i

$$R_j^{(col)} = c'_j - \sum_i x_{ij} = \text{residual for cells in column j}$$

for $i=1,2,...,m$, and $j=1,2,...,n$. If all of the $R_i^{(row)}$ and $R_j^{(col)}$ are zero, then $x'_{ij} = x_{ij}$. If any of the $R_i^{(row)}$ 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 the 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 $c''_j = c'$.

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

Bishop, Y., Fienberg, S., and Holland, P. (1975). *Discrete Multivariate Analysis: Theory and Practice*, Cambridge, Mass.: MIT Press.

Causey, B.D., Cox, L.H., and Ernst, L.R. (1985). "Applications of Transportation Theory to Statistical Problems," *Journal of the American Statistical Association*, **80**, pp. 903-909.

Cox, L.H. (1995). "Protecting Confidentiality in Business Surveys," in *Business Survey Methods*, Cox, Binder, Chinnappa, Christianson, Colledge, and Kott, eds. New York: Wiley, pp. 443-476.

_______ and Lawrence R. Ernst (1982). Controlled Rounding, *Infor*, **20**, pp. 423-432.

Deming, W.E. (1943). *Statistical Adjustment of Data*, New York: Wiley.

Draper, L. and Winkler, W. (1997). "Balancing and Ratio Editing with the new SPEER System," to appear in *Proceedings of the Section on Survey Research Methods*, American Statistical Association.

Fagan, James T. and Greenberg, B. (1984). "Making Tables Additive in the Presence of Zeroes," *Proceedings of the Section on Survey Research Methods*, American Statistical Association, pp. 195-200.

_______________________ (1985). Algorithms for Making Tables Additive: Raking, Maximum Likelihood, and Minimum Chi-Square, *Statistical Research Division Report Series*: *Census/SRD/RR-85/15*, Washington, DC: U.S. Bureau of the Census.

Oh, H.L. and Scheuren, F.J. (1978). "Multivariate Raking Ratio Estimation in the 1973 Exact Match Study," *Proceedings of the Survey Research Methods Section*, American Statistical Association, pp. 716-722.

Sigman, R. (1997). "Development of a 'Plain Vanilla' System for Editing Economic Census Data," *UN-ECE Work Session on Statistical Data Editing, Working Paper no. 24*, Prague, October 14-17, 1997.

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

j	i(j)	W_j*	z_j	L_j*	U_j*	z_j*	e_j+	e_j-	e_j*	c_j	e_j	z_j'	f_j*
1	2	1.4	-3	-5	0	-3	3	-2	3	3	2	-1	1
2	0	1.3	10	8	11	10	1	-2	1	4	0	10	0
3	1	1.2	-4	-4	-4	-4	0	0	0	4	0	-4	0
4	3	1.1	-7	-5	-1	-5	4	0	4	8	0	-5	1

$resid_1$	L*	U*	$resid_2$	I
-4	-6	6	-2	0

$resid_3$
0

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

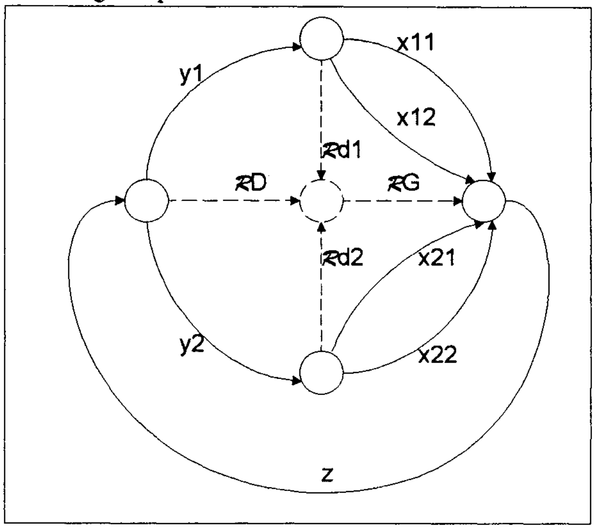

Table 1. Simple-1d balancing--Situations

Situation: balance condition	Situation: y	PV Procedure (Table 2 describes words in bold)
$y=\sum x_i$		No action taken. ($y'=y$, $x_i'=x_i$)
$y=0$ and $\sum x_i \neq 0$	fixed	**ZERO_SET** each $x_i'=0$.
	not fixed	**YSUMX**
$y \neq 0$ and $\sum x_i=0$	fixed	Same as $y \neq 0$, $\sum x_i \neq 0$, and $y \neq \sum x_i$
	not fixed	**ZERO_SET** $y'=0$.
$y \neq 0$, $\sum x_i \neq 0$, and $y \neq \sum x_i$		**RAKE** if $\lvert R \rvert / y$ is less than the specified (or default) raking tolerance. Otherwise, execute in script-specified sequence (stopping when an imbalance is no longer present) any script-specified **ROUND** (followed by **RAKE), REPLACE-H** (followed by **RAKE), REPLACE-V** (followed by **RAKE),** and **TRIM** if $\lvert R \rvert / y$ is less than the script-specified (or default) custom-trim tolerance. If an imbalance is still present, execute either **UNUSABLE** or **NSK** if one is specified in the script, and execute **TRIM** otherwise.

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

Abbr	Description	Definition
ZERO_SET	Set to zero	$y'=y$, $x_i'=0$; or $y'=0$, $x_i'=x_i$
YSUMX	Set y to $\sum x_i$,	$y'=\sum x_i$, $x_i'=x_i$
NSK	Not specified by kind	$y'=y$, $x_i'=x_i$, <NSK>=R
UNUSABLE	Mark data unusable	y' =<unusable> x_i' =<unusable>
REPLACE-V	Impute using category-average ratio	$y'=y$ $x_i' = [(x_i/y)_{\bar{R}}]\, y$
REPLACE-H	Impute using historic data	$y'=y$ $x_i'=(y/y^{(h)})x_i^{(h)}$
ROUND	Round y or x_i by 1000 if $\lvert R' \rvert / y'$ is less than a specified threshold	See § 2.2.
RAKE	Rake details to total if $\lvert R \rvert / y$ is less than a specified threshold	$y'=y$, $x_i'=(y/\sum x_i)x_i$ (Also, see § 2.1)
TRIM	Trim-and-adjust algorithm	See § 3.

Table 3. Nested-1d balancing situations

z fixed	all $R^{(d)}=0$	$R^{(D)}=0$	$R^{(g)}=0$	Adjust
no	yes	yes	yes	none (balanced)
no	yes	yes	no	n.a. (impossible)
no	yes	no	yes	n.a. (impossible)
no	no	yes	yes	x_{ij} (See § 4.1)
no	yes	no	no	z (See § 4.3)
no	no	yes	no	x_{ij} (See § 4.1)
no	no	no	yes	y_i (See § 4.2)
no	no	no	no	x_{ij}, y_i (See § 4.5)
yes	yes	yes	yes	none (balanced)
yes	yes	yes	no	n.a. (impossible)
yes	yes	no	yes	n.a. (impossible)
yes	no	yes	yes	x_{ij} (See § 4.1)
yes	yes	no	no	x_{ij}, y_i (See § 4.4)
yes	no	yes	no	x_{ij} (See § 4.1)
yes	no	no	yes	y_i (See § 4.2)
yes	no	no	no	x_{ij}, y_i (See § 4.5)