The Stata Journal

The *Stata Journal* publishes reviewed papers together with shorter notes or comments, regular columns, book reviews, and other material of interest to Stata users. Examples of the types of papers include 1) expository papers that link the use of Stata commands or programs to associated principles, such as those that will serve as tutorials for users first encountering a new field of statistics or a major new technique; 2) papers that go "beyond the Stata manual" in explaining key features or uses of Stata that are of interest to intermediate or advanced users of Stata; 3) papers that discuss new commands or Stata programs of interest either to a wide spectrum of users (e.g., in data management or graphics) or to some large segment of Stata users (e.g., in survey statistics, survival analysis, panel analysis, or limited dependent variable modeling); 4) papers analyzing the statistical properties of new or existing estimators and tests in Stata; 5) papers that could be of interest or usefulness to researchers, especially in fields that are of practical importance but are not often included in texts or other journals, such as the use of Stata in managing datasets, especially large datasets, with advice from hard-won experience; and 6) papers of interest to those who teach, including Stata with topics such as extended examples of techniques and interpretation of results, simulations of statistical concepts, and overviews of subject areas.

The *Stata Journal* is indexed and abstracted by *CompuMath Citation Index*, *Current Contents/Social and Behavioral Sciences*, *RePEc: Research Papers in Economics*, *Science Citation Index Expanded* (also known as *SciSearch*), *Scopus*, and *Social Sciences Citation Index*.

For more information on the *Stata Journal*, including information for authors, see the webpage

http://www.stata-journal.com

Subscriptions are available from StataCorp, 4905 Lakeway Drive, College Station, Texas 77845, telephone 979-696-4600 or 800-STATA-PC, fax 979-696-4601, or online at

http://www.stata.com/bookstore/sj.html

Subscription rates listed below include both a printed and an electronic copy unless otherwise mentioned.

U.S. and Canada		Elsewhere	
Printed & electronic		**Printed & electronic**	
1-year subscription	$ 98	1-year subscription	$138
2-year subscription	$165	2-year subscription	$245
3-year subscription	$225	3-year subscription	$345
1-year student subscription	$ 75	1-year student subscription	$ 99
1-year institutional subscription	$245	1-year institutional subscription	$285
2-year institutional subscription	$445	2-year institutional subscription	$525
3-year institutional subscription	$645	3-year institutional subscription	$765
Electronic only		**Electronic only**	
1-year subscription	$ 75	1-year subscription	$ 75
2-year subscription	$125	2-year subscription	$125
3-year subscription	$165	3-year subscription	$165
1-year student subscription	$ 45	1-year student subscription	$ 45

Back issues of the *Stata Journal* may be ordered online at

http://www.stata.com/bookstore/sjj.html

Individual articles three or more years old may be accessed online without charge. More recent articles may be ordered online.

http://www.stata-journal.com/archives.html

The *Stata Journal* is published quarterly by the Stata Press, College Station, Texas, USA.

Address changes should be sent to the *Stata Journal*, StataCorp, 4905 Lakeway Drive, College Station, TX 77845, USA, or emailed to sj@stata.com.

The *Stata Journal* (ISSN 1536-867X) is a publication of Stata Press. Stata, Stata, Stata Press, Mata, Mata, and NetCourse are registered trademarks of StataCorp LP.

The Stata Journal (2014)
14, Number 1, pp. 4–21

sreweight: A Stata command to reweight survey data to external totals

Daniele Pacifico
Italian Department of the Treasury
Rome, Italy
daniele.pacifico@tesoro.it

Abstract. This article describes `sreweight`, a Stata command to reweight survey data to external aggregate totals.

Keywords: st0322, sreweight, survey reweighting, external totals, calibration of survey data

1 Introduction

Statistical agencies and research institutes often need to adjust survey weights so that estimates match known control totals for given variables. These adjustments could be important to reduce the bias of estimates, to increase their efficiency, or to achieve consistency when alignment with other sources is required. In this article, I examine the `sreweight` command, which implements the methodology proposed by Deville and Särndal (1992) for survey reweighting. In contrast to other user-written commands, `sreweight` allows for a large number of distance functions, something that can be helpful when convergence is difficult to achieve in practice. Moreover, for all distance functions that require numerical methods, `sreweight` implements the algorithm proposed by Creedy (2003), which provides extremely rapid convergence. Finally, `sreweight` has several flexible options that can be helpful when calibration to external totals is difficult to achieve independently from the chosen function.[1] Section 2 contains the theoretical background for survey reweighting. Section 3 describes the `sreweight` command. Section 4 presents two empirical applications based on survey data.

2 An overview of the calibration method

Let us consider a survey of N individuals and K individual-level variables, such as income, sex, and age. We collect these variables for the generic respondent i in the following vector: $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iK})'$. If we define the survey weight with the vector $\boldsymbol{s} = (s_1, s_2, \ldots, s_i, \ldots, s_N)$, the estimated $1 \times K$ vector of totals is given by

$$\widehat{\boldsymbol{t}} = \sum_{i=1}^{N} s_i \boldsymbol{x}_i \tag{1}$$

1. Other user-provided commands perform the same type of reweighting on the basis of Deville and Särndal's (1992) article, but they lack `sreweight`'s characteristics listed above.

If external information is available on the real population totals for these K variables, it is possible to compute a new vector of weights—$\boldsymbol{w} = (w_1, w_2, \ldots, w_i, \ldots, w_N)$—that is as close as possible to the original weights and that respects the following calibrating conditions,

$$\boldsymbol{t} = \sum_{i=1}^{N} w_i \boldsymbol{x}_i \tag{2}$$

where $\boldsymbol{t}$ is the $1 \times K$ vector of true totals. Indeed, if we denote the distance between the original and the new weights with the function $G(s_i, w_i)$, the new weights can be obtained by minimizing the following Lagrangian function with respect to $\boldsymbol{w}$,

$$L = \sum_{i=1}^{N} G(s_i, w_i) + \sum_{k=1}^{K} \lambda_k \left(t_k - \sum_{i=1}^{N} w_i x_{ik} \right) \tag{3}$$

where $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_K]'$ are the Lagrange multipliers. Clearly, the solution of the minimization problem depends on the properties of the chosen distance function. Here we follow Creedy (2003) and require the function $G(s_i, w_i)$ to respect the following convenient properties:

1. The first derivative of $G(s_i, w_i)$ with respect to w_i can be defined as a function of the ratio between the new and the original weights:

$$\frac{\partial G(s_i, w_i)}{\partial w_i} = g\left(\frac{w_i}{s_i}\right)$$

2. The inverse of the first derivative of $G(s_i, w_i)$ exists and can be obtained explicitly.

If these properties hold, then the N first-order conditions for the problem in (3) are

$$g\left(\frac{w_i}{s_i}\right) - \boldsymbol{x}_i^{'}\boldsymbol{\lambda} = 0 \quad i = 1, 2, \ldots, N$$

and the new weights can be obtained as

$$w_i = s_i g^{-1}(\boldsymbol{x}_i^{'}\boldsymbol{\lambda}) \quad i = 1, 2, \ldots, N \tag{4}$$

given a solution for the Lagrange multipliers.

The Lagrange multipliers can be obtained through an iterative procedure after some algebraic manipulation of (4). In particular, let us substitute (4) into (2),

$$\boldsymbol{t} = \sum_{i=1}^{N} w_i \boldsymbol{x}_i = \sum_{i=1}^{N} s_i g^{-1}\left(\boldsymbol{x}_i^{'}\boldsymbol{\lambda}\right) \boldsymbol{x}_i$$

and then subtract (1) from both sides:

$$\boldsymbol{t} - \widehat{\boldsymbol{t}} = \sum_{i=1}^{N} s_i g^{-1}\left(\boldsymbol{x}_i^{'}\boldsymbol{\lambda}\right) \boldsymbol{x}_i - \sum_{i=1}^{N} s_i \boldsymbol{x}_i = \sum_{i=1}^{N} s_i \left\{ g^{-1}\left(\boldsymbol{x}_i^{'}\boldsymbol{\lambda}\right) - 1 \right\} \boldsymbol{x}_i \tag{5}$$

If we define $\boldsymbol{a} = \boldsymbol{t} - \widehat{\boldsymbol{t}}$, (5) can be rewritten as

$$f(\boldsymbol{\lambda}) = \boldsymbol{a} - \sum_{i=1}^{N} s_i \left\{ g^{-1}\left(\boldsymbol{x}_i'\boldsymbol{\lambda}\right) - 1 \right\} \boldsymbol{x}_i = \mathbf{0} \tag{6}$$

The root of this function can be computed by means of Newton's method, which involves the following iterative algorithm,

$$\boldsymbol{\lambda}^{(t+1)} = \boldsymbol{\lambda}^{(t)} - \left\{ \frac{\partial f(\boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}} \right\}^{-1} f(\boldsymbol{\lambda}) \tag{7}$$

where $f(\boldsymbol{\lambda})$ is evaluated with the previous value of the Lagrange multipliers, $\boldsymbol{\lambda}^{(t)}$. The generic s, k element of the Hessian matrix that enters the previous recursion—$\{\partial f(\boldsymbol{\lambda})\}/\{\partial \boldsymbol{\lambda}\}$—can be computed from (6),

$$\begin{aligned} \frac{\partial f_s(\boldsymbol{\lambda})}{\partial \lambda_k} &= -\sum_{i=1}^{N} s_i \boldsymbol{x}_{is} \frac{\partial g^{-1}\left(\boldsymbol{x}_i'\boldsymbol{\lambda}\right)}{\partial \lambda_k} \\ &= -\sum_{i=1}^{N} s_i \boldsymbol{x}_{is} \boldsymbol{x}_{ik} \frac{\partial g^{-1}\left(\boldsymbol{x}_i'\boldsymbol{\lambda}\right)}{\partial \left(\boldsymbol{x}_i'\boldsymbol{\lambda}\right)} \end{aligned}$$

where $s = 1, 2, \ldots, k, \ldots, K$. Given a set of initial values for $\boldsymbol{\lambda}$, the recursion in (7) can be repeatedly evaluated until convergence. The following considers some of the most common distance functions and provides the related equations for the empirical implementation of the recursive algorithm explained in this section.

2.1 The chi-squared distance

The chi-squared distance function is one of the most popular choices in the applied literature. The main reason is that the minimization problem in (3) has an explicit solution that can be obtained immediately without the iterative procedure outlined above. The chi-squared distance function is given by

$$G(s, w) = \frac{1}{2} \sum_{i=1}^{N} \frac{(w_i - s_i)^2}{s_i} \tag{8}$$

Hence, substituting this function into (3), we obtain

$$L = \frac{1}{2} \sum_{i=1}^{N} \frac{(w_i - s_i)^2}{s_i} + \sum_{k=1}^{K} \lambda_k \left(t_k - \sum_{i=1}^{N} w_i x_{ik} \right)$$

and the first-order conditions are

$$\begin{aligned}\frac{\partial L}{\partial w_i} = \left(\frac{w_i}{s_i} - 1\right) - \sum_{k=1}^{K} \lambda_k x_{ik} &= 0 \\ \left(\frac{w_i}{s_i} - 1\right) - \boldsymbol{x}_i^{'}\boldsymbol{\lambda} &= 0 \\ s_i\left(1 + \boldsymbol{x}_i^{'}\boldsymbol{\lambda}\right) &= w_i \end{aligned} \tag{9}$$

for $i = 1, 2, \ldots, N$. To obtain the Lagrangian multipliers, we premultiply (9) by $\boldsymbol{x}_i$, rearrange, and sum over all N:

$$\sum_{i=1}^{N} s_i x_i \boldsymbol{x}_i^{'}\boldsymbol{\lambda} = \sum_{i=1}^{N} w_i \boldsymbol{x}_i - \sum_{i=1}^{N} s_i \boldsymbol{x}_i \tag{10}$$

Then given the conditions in (1) and (2), (10) can be rewritten as

$$\begin{aligned}\left(\sum_{i=1}^{N} s_i x_i \boldsymbol{x}_i^{'}\right)\boldsymbol{\lambda} &= \boldsymbol{t} - \widehat{\boldsymbol{t}} \\ \boldsymbol{\lambda} &= \left(\sum_{i=1}^{N} s_i x_i \boldsymbol{x}_i^{'}\right)^{-1}\left(\boldsymbol{t} - \widehat{\boldsymbol{t}}\right)\end{aligned} \tag{11}$$

Finally, substituting (11) into (9) gives the new weights for the N observations.

2.2 Alternative distance functions

The main limitation of the chi-squared distance is that no constraints are placed on the size of the adjustment of the survey weights, with the possibility that some of the calibrated weights become negative after the adjustment.

In addressing this, the literature proposes that alternative functions incorporate constraints on the size of the adjustment. However, for these functions, a closed-form solution is no longer available, and the iterative procedure explained above has to be used. The following table reports three distance functions coded into `sreweight` that force the new weight to be strictly positive. Table 1 also shows the equations needed to update the recursion during the iterative procedure, which are the functions $g(\boldsymbol{x}_i^{'}\boldsymbol{\lambda})$, $g^{-1}(\boldsymbol{x}_i^{'}\boldsymbol{\lambda})$, and $\{\partial g^{-1}(\boldsymbol{x}_i^{'}\boldsymbol{\lambda})\}/\{\partial(\boldsymbol{x}_i^{'}\boldsymbol{\lambda})\}$.

Table 1. Different distance functions

Type	$G(s_i, w_i)$	$g(\frac{w_i}{s_i})$	$g^{-1}(\boldsymbol{x}_i'\boldsymbol{\lambda})$	$\frac{\partial g^{-1}(\boldsymbol{x}_i'\boldsymbol{\lambda})}{\partial(\boldsymbol{x}_i'\boldsymbol{\lambda})}$
`a`	$2\left(\sqrt{w_i}-\sqrt{s_i}\right)^2$	$2\left\{1-\left(\frac{w_i}{s_i}\right)^{-\frac{1}{2}}\right\}$	$\left(1-\frac{\boldsymbol{x}_i'\boldsymbol{\lambda}}{2}\right)^2$	$\left(1-\frac{\boldsymbol{x}_i'\boldsymbol{\lambda}}{2}\right)^3$
`b`	$-s_i\ln\left(\frac{w_i}{s_i}\right)+w_i-s_i$	$1-\left(\frac{w_i}{s_i}\right)^{-1}$	$\left(1-\boldsymbol{x}_i'\boldsymbol{\lambda}\right)^{-1}$	$\left(1-\boldsymbol{x}_i'\boldsymbol{\lambda}\right)^{-2}$
`c`	$w_i\ln\left(\frac{w_i}{s_i}\right)-w_i+s_i$	$\ln\left(\frac{w_i}{s_i}\right)$	$\exp\left(\boldsymbol{x}_i'\boldsymbol{\lambda}\right)$	$\exp\left(\boldsymbol{x}_i'\boldsymbol{\lambda}\right)$

The DS distance function

The `sreweight` command also allows for the following distance function,

$$G(s_i,w_i)=\left(r_U-\frac{w_i}{s_i}\right)\log\left(\frac{r_U-\frac{w_i}{s_i}}{r_U-1}\right)+\left(\frac{w_i}{s_i}-r_L\right)\log\left(\frac{\frac{w_i}{s_i}-r_L}{1-r_L}\right)+\frac{r_U-r_L}{\alpha}s_i$$

where $r_L<1<r_U$ are positive constants, and $\alpha=\{r_U-r_L\}/\{(1-r_L)(r_U-1)\}$. This function was proposed by Deville and Särndal (1992) because—in contrast to the other functions outlined above—the calibrated weights are kept within a known range set by the user; that is, $r_L s_i<w_i<r_U s_i$.

The related elements of $G(s_i,w_i)$ that are needed to update the recursion with the Deville and Särndal (DS) distance function are

$$\begin{aligned}
g\left(\frac{w_i}{s_i}\right)&=\frac{1}{\alpha}\left\{\log\left(\frac{\frac{w_i}{s_i}-r_L}{1-r_L}\right)-\log\left(\frac{r_U-\frac{w_i}{s_i}}{r_U-1}\right)\right\}\\
g^{-1}\left(\boldsymbol{x}_i'\boldsymbol{\lambda}\right)&=\frac{r_L(r_U-1)+r_U(1-r_L)\exp(\alpha\boldsymbol{x}_i'\boldsymbol{\lambda})}{(r_U-1)+(1-r_L)\exp(\alpha\boldsymbol{x}_i'\boldsymbol{\lambda})}\\
\frac{\partial g^{-1}\left(\boldsymbol{x}_i'\boldsymbol{\lambda}\right)}{\partial(\boldsymbol{x}_i'\boldsymbol{\lambda})}&=g^{-1}\left(\boldsymbol{x}_i'\boldsymbol{\lambda}\right)\left\{r_U-g^{-1}\left(\boldsymbol{x}_i'\boldsymbol{\lambda}\right)\right\}\frac{(1-r_L)\,\alpha\exp\left(\alpha\boldsymbol{x}_i'\boldsymbol{\lambda}\right)}{(r_U-1)+(1-r_L)\exp\left(\alpha\boldsymbol{x}_i'\boldsymbol{\lambda}\right)}
\end{aligned}$$

The modified chi-squared distance function

Deville and Särndal (1992) also propose a modification of the chi-squared distance. Similar to the DS case, the modification constrains the new weights to be within the range $r_L<w_i/s_i<r_U$. Whereas this function does not allow for an explicit solution, it proves to be very stable and ensures convergence even when calibration to external totals is difficult in practice.

To show how the iterative procedure works for this function, we consider the derivative of the chi-squared function in (8),

$$g\left(\frac{w_i}{s_i}\right) = \frac{w_i}{s_i} - 1$$

which implies that $g^{-1}(\boldsymbol{x}_i'\boldsymbol{\lambda}) = w_i/s_i = 1 + \boldsymbol{x}_i'\boldsymbol{\lambda}$. Therefore, if at some iterations, $s_i(1 + \boldsymbol{x}_i'\boldsymbol{\lambda})$ is outside the required range, the updated weight, w_i, can be simply set to the relevant limit (that is, $s_i r_L$ or $s_i r_U$) until the algorithm achieves convergence.[2]

3 The sreweight command

3.1 Syntax

The generic syntax for `sreweight` is

`sreweight` *varlist* [*if*] [*in*], `sweight(`*varname*`)` `nweight(`*newvar*`)`
`total(`*matrix*`)` `dfunction(`*name*`)` [`svalues(`*matrix*`)` `tolerance(#)` `niter(#)`
`ntries(#)` `upbound(#)` `lowbound(#)` `rbounds(#)` `rlowbound(# #)`
`rupbound(# #)`]

where *varlist* includes the calibration variables.

3.2 Options

`sweight(`*varname*`)` specifies a numeric variable to be used for the original survey weights. `sweight()` is required.

`nweight(`*newvar*`)` contains the name of the variable to be created with the new weights. `nweight()` is required.

`total(`*matrix*`)` contains a $1 \times K$ matrix with the user-provided totals, which is ordered as the variables in *varlist*. The arguments must be inserted in the same order as the K calibrating variables in *varlist*. `total()` is required.

`dfunction(`*name*`)` specifies the distance function to be used when computing the new weights. The allowed functions are those introduced in section 2, which are the chi-squared (`chi2`), the modified chi-squared (`mchi2`), Deville and Särndal's (1992) function (`ds`), and the functions we defined as type A (`a`), type B (`b`), and type C (`c`). Note that for all functions but the chi-squared, `sreweight` works with the recursion outlined in the previous section. `dfunction()` is required.

2. In this case, given that $g^{-1}(\boldsymbol{x}_i'\boldsymbol{\lambda}) = 1 + \boldsymbol{x}_i'\boldsymbol{\lambda}$, the derivative $\{\partial g^{-1}(\boldsymbol{x}_i'\boldsymbol{\lambda})\}/\{\partial(\boldsymbol{x}_i'\boldsymbol{\lambda})\}$ has to be set to 0 for that observation because $1 + \boldsymbol{x}_i'\boldsymbol{\lambda}$ equals a constant (that is, r_U or r_L).

`svalues(`*matrix*`)` specifies user-provided starting values. Starting values must be put in a Stata $1 \times K$ matrix following the same order as the variables in *varlist*. The default is a vector with the Lagrange multipliers obtained from the chi-squared distance function.

`tolerance(#)` specifies the tolerance level that enters the iterative algorithm to declare convergence. The default is `tolerance(0.000001)`.[3]

`niter(#)` specifies the maximum number of iterations. The default is `niter(50)`.

`ntries(#)` specifies the maximum number of "tries" when the algorithm does not achieve convergence within the maximum number of iterations. This option can be useful when the external totals are significantly different from the survey totals. In such situations, the algorithm automatically restarts with new random starting values up to # times. The default is `ntries(0)`.[4]

`upbound(#)` specifies the upper bound of the ratio between the new and the original weights when using either the modified chi-squared or the DS distance function. The default is `upbound(4)`. Note that this value must be bigger than 1.

`lowbound(#)` specifies the lower bound of the ratio between the new and the original weights when using either the modified chi-squared or the DS distance function. The default is `lowbound(0.2)`. Note that this value must be between 0 and 1.

`rbounds(#)` is relevant only for the modified chi-squared and the DS functions when the `ntries()` option is effective. In this case, if the recursion does not achieve convergence, the algorithm restarts with both a new set of starting values and a new set of random bounds. The allowed values for this option are `0` (no random bounds) and `1` (allow for random bounds). The default is `rbounds(0)`.

`rlowbound(# #)` and `rupbound(# #)` are relevant options only for the modified chi-squared and the DS distance functions when the options `ntries()` and `rbounds()` are both effective. In this case, the two values in `rlowbound()` (or `rupbound()`) define the support of the uniform distribution from which the new lower (or upper) bound is drawn.[5] When the `rbounds()` option is effective, the default values for these options are `rlowbound(0.1 0.7)` and `rupbound(1.5 6)`.

3. `sreweight` uses a double criterion to assess convergence. The first is that the difference between the estimated and external totals must be lower than the tolerance level. The second is that from one iteration to the other, the percentage variations of the estimated distance between the new and the original weights must be lower than the tolerance level for each observation in the sample.
4. New starting values are obtained from a random perturbation of the Lagrange multipliers obtained with the chi-squared distance function. The perturbation for each multiplier is drawn from a uniform distribution with range -1 to 1.
5. Hence, if the user sets `rlowbound(0.2 0.8)`, the new lower bound will be drawn from a uniform distribution with support $0.2 - 0.8$.

4 Empirical applications

For our empirical exercise, we use data from the Second National Health and Nutrition Examination Survey (NHANES II). Two different applications are presented: One focuses on the problem of survey reweighing, and the other focuses on the problem of variance estimation in complex survey data.

4.1 Reweighing NHANES II

In this section, we show how to use `sreweight` for the calibration of NHANES II to a set of known totals related to a hypothetical distribution of the population by gender and age.

First, the database is loaded, and the command `svyset` is used to declare the survey design. Second, the calibrating variables—`sex` and `age1`–`age6`—are created, and the command `svy: total` is used to estimate the totals of interest. Finally, a vector of hypothetical population totals is created:[6]

```
. use http://www.stata-press.com/data/r13/nhanes2
. quietly svyset psu [pw=finalwgt], strata(strata)
. generate byte ones=1
. quietly tab agegr, gen(age)
. quietly recode sex 2=0
. label drop sex
. quietly svy: total ones, over(sex)
. matrix sex =e(b)
. quietly svy: total ones, over(agegr)
. matrix age =e(b)
. matrix Tot=[sex[1,2]*1.01 \ age[1,1]*0.99 \ age[1,2]*1.01 \ age[1,3]*0.98 \
> age[1,4]*1.019 \ age[1,5]*0.95 \  e(N_pop)]
```

Once the vector with the new totals is created, `sreweight` can be repeatedly used to compute the new vectors of weights, each based on a different distance function:

```
. local dfset "chi2 a b c ds mchi2"
. foreach df of local dfset {
  2. quietly sreweight sex age1-age5 ones, sweight(finalwgt) nweight(type_`df´)
> total(Tot) dfunction(`df´) niter(40)
  3. }
```

As explained in section 2, both the `ds` and the `mchi2` functions allow users to set bounds for the ratio w/s. In the previous lines, no upper or lower bounds are specified, and `sreweight` uses the default bounds (4 and 0.2, respectively). Clearly, these broad limits can be progressively modified toward 1 so long as a solution is available. In this specific database, for example, an upper bound of 1.1 and a lower bound of 0.94 do not

6. As the code shows, the original estimate of the population size is also included in the last row of the vector (`e(N_pop)`). Hence, only five out of the six new totals related to the population size by age group need to be included in the vector.

lead to convergence for either functions. When convergence is not achieved because of the tight intervals, the `rlowbound()` and `rupbound()` options can be used to avoid a time-consuming search of the closest values around 1 for the two bounds:

```
. set seed 987456123
. sreweight sex age1-age5 ones, sweight(finalwgt) nweight(type_1ds) total(Tot)
> niter(40) dfunction(ds) upbound(1.1) lowbound(0.94) ntries(50) rbounds(1)
> rupbound(1.15 1.20) rlowbound(0.94 0.99)
Iteration 1
  (output omitted)
Iteration 40 Not Converged within the maximum number of iterations, the
> algorithm now tries with new random bounds up to 50 times:
try number 1 current bounds .978 - 1.169
  (output omitted)
try number 17 current bounds .95 - 1.17
Converged, new starting values saved in the return list
Survey and calibrated totals
```

Variable	Original	New
sex	56159480	56721075
age1	32857697	32529120
age2	23935443	24174797
age3	19725067	19330566
age4	19584095	19956193
age5	15639265	14857302
ones	117157513	117157513

```
Note: type-ds distance function used
Current bounds: upper=1.1704 - lower=.9495
. set seed 987456123
. sreweight sex age1-age5 ones, sweight(finalwgt) nweight(type_1mchi2) total(Tot)
> niter(40) dfunction(mchi2) upbound(1.1) lowbound(0.94) ntries(50)
> rupbound(1.15 1.20) rlowbound(0.94 0.99)
Note: with this distance function, rbounds() is automatically set to 1 if the
> argument of ntries() is >0
Iteration 1
  (output omitted)
Iteration 40 Not Converged within the maximum number of iterations, the
> algorithm now tries with new random bounds up to 50 times:
try number 1 current bounds .978 - 1.169
  (output omitted)
try number 17 current bounds .95 - 1.17
Converged, new starting values saved in the return list
  (output omitted)
```

The following figures show, respectively, the kernel densities of the original and calibrated weights, and the densities of the ratios w/s for all the distance functions used in the previous lines:

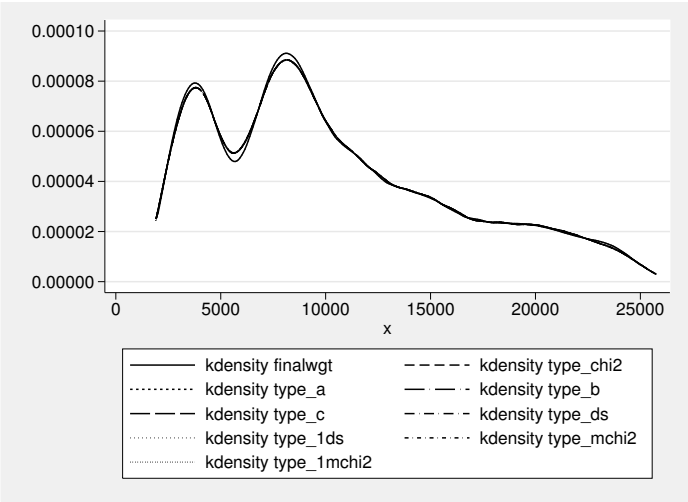

Figure 1. Nine overlapping kernel densities of original and calibrated weights

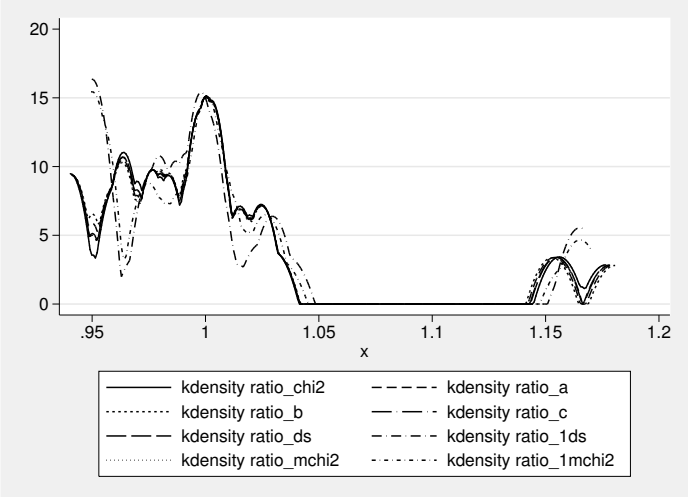

Figure 2. Densities of ratios w/s

From figure 1, we see that all the distance functions produce very similar results and that the densities of the calibrated weights closely resemble the densities of the original

weights. However, as figure 2 shows, reducing the allowed range of the ratio w/s may place several weights in the range limits, in particular the lower limit, given the chosen bounds and the type of data. This result was also found in Creedy (2003) and suggests some carefulness when constraining the maximum deviation of the calibrated weights from the survey weights.

When the known totals highly differ from the estimated totals, the `ntries()` and the `rbounds()` options can be useful, as can the possibility to choose among several distance functions, given that not all of them may achieve convergence. Let us redefine the 1×7 matrix with the known totals using values that are quite different from the estimated totals:

```
. matrix Tot=[sex[1,2]*1.5 \ age[1,1]*0.6 \ age[1,2]*1.5 \ age[1,3]*0.4 \
> age[1,4]*1.5 \ age[1,5]*1.5 \  e(N_pop)]
```

The calibrated weights can be recomputed for all distance functions using `ntries()` in the loop so as to automatically search for better starting values in case of no convergence:

```
. local dfset "chi2 a b c ds mchi2"
. foreach df of local dfset {
  2.  set seed 987456123
  3.  display " "
  4.  display in white "Type_`df´:"
  5.  sreweight sex age1-age5 ones, sweight(finalwgt) nweight(type_`df´)
> total(Tot) dfunction(`df´) niter(40) ntries(35)
  6. }

Type_chi2:
New weights obtained from the chi2 distance function are negative, try with
> other distance functions

Type_a:
Iteration 1
  (output omitted)
Iteration 7 - Converged
Survey and calibrated totals
```

Variable	Original	New
sex	56159480	84239220
age1	32857697	19714618
age2	23935443	35903165
age3	19725067	7890027
age4	19584095	29376142
age5	15639265	23458898
ones	117157513	117157513

```
Note: type-a distance function used

Type_b:
Iteration 1
  (output omitted)
```

```
Iteration 40 Not Converged within the maximum number of iterations, the
> algorithm now tries with new starting values up to 35 times:
try number 1
```
(*output omitted*)
```
try number 34
Converged, new starting values saved in the return list
```
(*output omitted*)

```
Type_c:
Iteration 1
```
(*output omitted*)
```
Iteration 40 Not Converged within the maximum number of iterations, the
> algorithm now tries with new starting values up to 35 times:
try number 1
```
(*output omitted*)
```
try number 35
Not Converged within the maximum number of tries. Try to increase the number of
>  maximum tries and/or the number of maximum iterations

Type_ds:
Iteration 1
```
(*output omitted*)
```
Iteration 40 Not Converged within the maximum number of iterations, the
> algorithm now tries with new starting values up to 35 times:
try number 1
```
(*output omitted*)
```
try number 35
Not Converged within the maximum number of tries. Try to: activate the
> rbounds() option, increase the number of maximum tries or the number of
> maximum iterations

Type_mchi2:
Note: with this distance function, rbounds() is automatically set to 1 if the
> argument of ntries() is >0
Iteration 1
```
(*output omitted*)
```
Iteration 40 Not Converged within the maximum number of iterations, the
> algorithm now tries with new random bounds up to 35 times:
try number 1 current bounds .554 - 3.171
```
(*output omitted*)
```
try number 25 current bounds .106 - 2.575
Converged, new starting values saved in the return list
```
(*output omitted*)

As can be seen, the new totals require such an adjustment of the original weights that some of the new weights become negative with the chi-squared function. Moreover, the algorithm does not achieve convergence within the selected number of iterations for all the functions except the `a` distance. However, using the option `ntries()`, we find a solution for the `b` and the `mchi2` functions after 34 and 25 tries, respectively.

Given the random bounds found for the `mchi2` function (2.575 and 0.106), an attempt can be made to see whether another set of bounds closer to 1 exists. To this end, the options `rlowbound()` and `rupbound()` can be used to avoid a manual search:

```
. set seed 987456123
. sreweight sex age1-age5 ones, sweight(finalwgt) nweight(type_1mchi2)
> total(Tot) dfunction(mchi2) upbound(2) lowbound(0.2) rupbound(2 2.5)
> rlowbound(0.14 0.2) ntries(35)
Note: with this distance function, rbounds() is automatically set to 1 if the
> argument of ntries() is >0
Iteration 1
```

(*output omitted*)

```
Iteration 50 Not Converged within the maximum number of iterations, the
> algorithm now tries with new random bounds up to 35 times:
try number 1 current bounds .185 - 2.186
```

(*output omitted*)

```
try number 18 current bounds .15 - 2.281
Converged, new starting values saved in the return list
```

(*output omitted*)

What is the performance of each distance function when the new totals are so much different from the original estimates? The next two figures show the densities of the various vectors of weights and the densities of the ratio w/s:

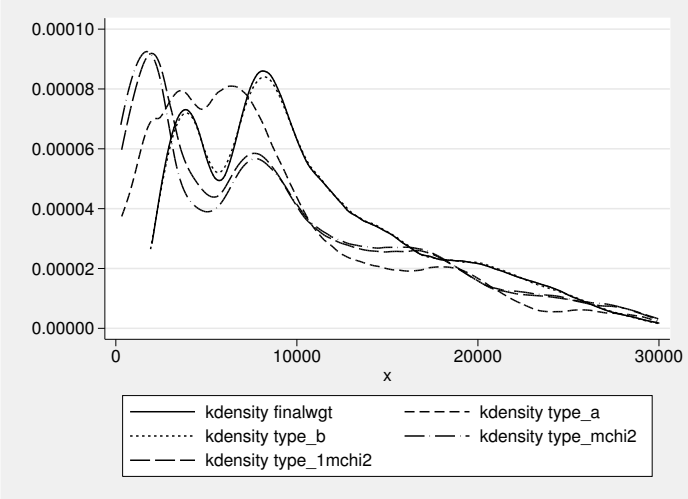

Figure 3. Densities of vectors of weights

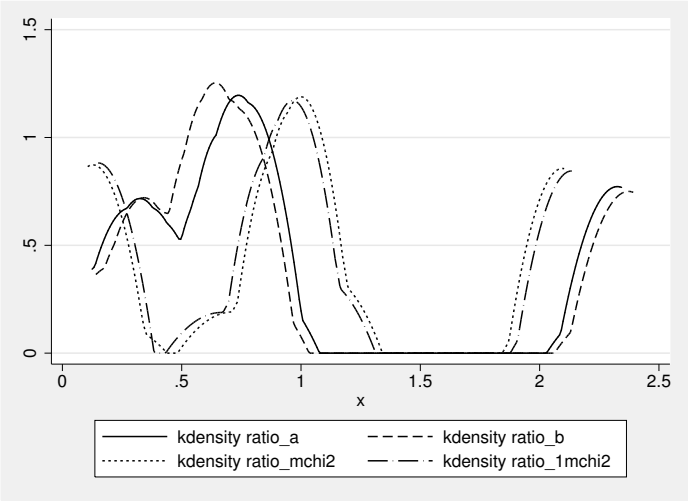

Figure 4. Densities of ratio w/s

The differences between the various distance functions are more evident if compared with the previous exercise. In particular, there is a clear difference between the results produced with the `a` and the `b` functions—those that do not place constraints on the adjustment of the original weights—and the results obtained with the `mchi2` distance.

Specifically, the `mchi2` function produces several weights that are very close to the limits of the distribution and, at the same time, a number of new weights that are similar to the original weights.[7] On the contrary, the functions that set no constraints to the adjustment rescale most of the original weights toward the bottom of the distribution without placing as many at the outer limit. However, this is partially compensated by a number of new weights that are relatively bigger than the original weights.

To summarize: When the difference between the new and the original totals is rather large, the functions that constrain the adjustments of the original weights allow preserving several of the original values, but at the cost of placing most of the adjustments at the bounds of the distribution. However, when no bounds are imposed, the adjustment is more general, with several new weights shifted toward the left (or the right, depending on the data) of the distribution, even though not so close to the outer limit.

Unfortunately, a one-size-fits-all solution does not exist for the selection of the optimal function, and as explained in Deville and Särndal (1992) and Särndal (2007), the choice should be mainly driven by practical issues rather than theoretical principles.[8] Typical discriminating factors are, for example, the presence of negative or implausible

7. This can be seen in figure 4, where the density of the ratio w/s is higher around 1 and around the selected bounds.

8. Deville and Särndal (1992) have demonstrated that different distance functions have a minor effect only on the variance of the calibration estimator and that under very mild conditions, they are all asymptotically equivalent.

high weights, the possibility that the algorithm does not achieve convergence—either because of the number of calibrating variables or the type of adjustment required to the original weights—and the degree of modification of the population relationships between calibrating and other target variables.[9]

4.2 The use of reweight for variance estimation in complex surveys

In Stata, the problem of variance estimation in complex surveys has been analyzed in Kolenikov (2010). As explained in that article, several variance estimation methods can be used with survey data depending on the information available. For example, the possibility to identify the stratum and the primary sampling units allows for estimating the variances of interest through a Taylor series expansion (the delta method). However, when such information is not available, other methods must be used (one of the most flexible is the balanced bootstrap with internal scaling).[10]

However, when the original weights are poststratified (or adjusted for nonresponse), the replicate weights created with the balanced bootstrap with internal scaling must be processed accordingly to account for the same adjustment that was used with the original weights. The command outlined in Kolenikov (2010)—`bsweights`—can perform the balance bootstrap with internal scaling, and `sreweight` can be easily used as a wrapper to perform the calibration adjustments of the replicate weights.

Let us suppose that the original weights of NHAMES II were poststratified with respect to the age group and the gender. Hence, each vector of replicate weights must be calibrated after the internal scaling so that the estimated population size by gender and age group is the same as that obtained with the original weights. To replicate the data used in Kolenikov (2010), we first collapse some of the strata of NHAMES II and recode the primary sampling units; then we again `svyset` the data and store the totals of interest in a 1×7 matrix:

```
. generate cstrata = floor(sqrt(2*strata-1))
. egen upsu = group(strata psu)
. quietly svyset upsu [pw=finalwgt], strata(cstrata)
. quietly svy: total ones, over(sex)
. matrix sex =e(b)
. quietly svy: total ones, over(agegr)
. matrix age =e(b)
. matrix Tot=[sex[1,2] \ age[1,1] \ age[1,2] \ age[1,3] \ age[1,4] \ age[1,5] \
> age[1,6]]
```

9. Given our application, a target variable of interest could be the individual weight, which is strongly related to the gender and the age group.
10. Internal scaling is important to rectify the bias of a naïve bootstrap scheme and can be achieved by the replicate weights. See Kolenikov (2010) for more details about this procedure.

The following lines define the calibration program that allows `bsweights` to use `sreweight` internally,

```
. program Calreweight
  1. args wvar
  2. tempvar app
  3. sreweight sex age1-age6, sweight(`wvar´) nweight(app) total(Tot)
> dfunction($df) ntries(50)
  4. replace `wvar´=app
  5. end
```

where it is worth noting that the command is general enough to allow for different distance functions in the reweighting procedure. Finally, we create the replicate weights for each distance function coded in `sreweight` and run the estimation command—a logistic regression—using the `bs4rw` command for each set of replicate weights:

```
. local dfset "a b c ds chi2"
. foreach df of local dfset {
  2.  global df="`df´"
  3.  bsweights bw`df´, n(-2) reps(120) dots balanced calibrate(Calreweight @)
> seed(10101) replace
  4.  bs4rw, rweights(bw`df´1-bw`df´120): logistic highbp height weight sex
> age1-age5 [pw=finalwgt]
  5.                                                          }
Balancing within strata:
.......
Rescaling weights
..................................................    50
..................................................   100
..................
(running logistic on estimation sample)
BS4Rweights replications (120)
----+--- 1 ---+--- 2 ---+--- 3 ---+--- 4 ---+--- 5
..................................................    50
..................................................   100
..................
Logistic regression                               Number of obs      =     10351
```

(*output omitted*)

Clearly, each distance function has an impact on the estimated standard errors. To compare such estimates, we recompute the same standard errors with the delta method, which is the default estimation procedure for survey data in Stata:

```
. svy: logistic highbp height weight female age1-age5
(running logistic on estimation sample)

Survey: Logistic regression

Number of strata   =         7                  Number of obs      =      10351
Number of PSUs     =        62                  Population size    =  117157513
                                                Design df          =         55
                                                F(   8,     48)    =     160.13
                                                Prob > F           =     0.0000

------------------------------------------------------------------------------
             |             Linearized
      highbp | Odds Ratio   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      height |   .9638611   .0050313    -7.05   0.000     .9538306     .973997
      weight |   1.053076   .0029406    18.52   0.000       1.0472    1.058986
      female |   .6137008   .0379283    -7.90   0.000     .5422095    .6946183
        age1 |   .1096952   .0102944   -23.55   0.000     .0908886    .1323933
        age2 |   .1794773    .018754   -16.44   0.000     .1455678    .2212859
        age3 |    .275236   .0308778   -11.50   0.000     .2198185    .3446246
        age4 |   .5544411   .0585061    -5.59   0.000     .4487601    .6850094
        age5 |    .694758   .0670287    -3.77   0.000     .5726174    .8429515
       _cons |   29.89828   23.18172     4.38   0.000     6.321514    141.4071
------------------------------------------------------------------------------
```

Then following Kolenikov (2010), the percentage variations of the standard errors obtained with `bsweights` and `bs4rw` for each distance function are compared with those obtained with the delta method:[11]

	Type of distance function				
Variable	χ^2	A	B	C	DS
`height`	5.006%	5.001%	4.999%	5.002%	5.002%
`weight`	−1.233%	−1.236%	−1.239%	−1.236%	−1.236%
`sex`	2.075%	2.072%	2.071%	2.073%	2.073%
`age1`	3.381%	3.410%	3.420%	3.400%	3.399%
`age2`	3.392%	3.410%	3.417%	3.404%	3.403%
`age3`	−3.056%	−3.046%	−3.042%	−3.049%	−3.050%
`age4`	3.362%	3.365%	3.365%	3.364%	3.364%
`age5`	0.529%	0.557%	0.566%	0.547%	0.546%

As the results show, the highest deviation is about 5%; therefore, the inference conclusions stay unchanged. Moreover, reading the results by row, we can see that the standard errors produced by the different distance functions are almost the same. Hence, we can conclude that in this database, the choice of the distance function does not have a significant impact on the variance estimation.

11. Results for the `mchi2` distance are not reported, because they are the same as those obtained with the `chi2` function.

5 References

Creedy, J. 2003. Survey reweighting for tax microsimulation modelling. Treasury Working Paper Series 03/17, New Zealand Treasury.

Deville, J.-C., and C.-E. Särndal. 1992. Calibration estimators in survey sampling. *Journal of the American Statistical Association* 87: 376–382.

Kolenikov, S. 2010. Resampling variance estimation for complex survey data. *Stata Journal* 10: 165–199.

Särndal, C.-E. 2007. The calibration approach in survey theory and practice. *Survey Methodology* 33: 99–119.

About the author

Daniele Pacifico works with the Italian Department of the Treasury in Rome, Italy, and is a member of the Centre for the Analysis of Public Policies in Modena, Italy.