

Updates to the ipfraking ecosystem

Stanislav Kolenikov
Abt Associates
stas_kolenikovs@abtassoc.com

Abstract. ? introduced package `ipfraking` for weight calibration procedures known as iterative proportional fitting, or raking, of complex survey weights. This article briefly describes the original package, and adds updates to the core program, as well as a host of additional programs that are used to support the process of creating survey weights in the authors’ production code.

Keywords: st0001, survey, calibration, weights, raking

1 Introduction and background

Large scale social, behavioral and health data are often collected via complex survey designs that may involve some or all of stratification, multiple stages of selection and unequal probabilities of selection (??). In an ideal setting, varying probabilities of selection are accounted for by using the Horvitz-Thompson estimator of the totals (??), and the remaining sampling fluctuations can be further ironed out by post-stratification (?). However, on top of the planned differences in probabilities of obtaining a response from a sampled unit, non-response is a practical problem that has been growing more acute over the recent years (??). The analysis weights that are provided along with the public use microdata by data collecting agencies are designed to account for unequal probabilities of selection, non-response, and other factors affecting imbalance between the population and the sample, thus making the analyses conducted on such microdata generalizable to the target population.

Earlier, I introduced (?) a Stata package called `ipfraking` that implements calibration of survey weights to known control totals to ensure that the resulting weighted data are representative of the population of interest. The process of calibration is aimed at aligning the sample totals of the key variables with those known for the population as a whole.

For a given finite population \mathcal{U} of units indexed $i = 1, \dots, N$, the interests of survey statisticians often lie in estimating the population total of a variable Y

$$T[Y] = \sum_{i \in \mathcal{U}} Y_i \quad (1)$$

A sample \mathcal{S} of n units indexed by $j = 1, \dots, n$ is taken from \mathcal{U} . If the probability to select the i -th unit is known to be π_i , then the *probability weights*, or *design weights*, are given by the inverse probability of selection:

$$w_{1i} = \pi_i^{-1} \quad (2)$$

With these weights, an unbiased (design-based, non-parametric) estimator of the total (1) is (?)

$$t_1[y] = \sum_{j \in \mathcal{S}} \frac{y_j}{\pi_j} \equiv \sum_{j \in \mathcal{S}} w_{1j} y_j, \quad (3)$$

The subindex 1 indicates that the weights w_{1i} were used in obtaining this estimator. Probability weights protect the end user from potentially informative sampling designs, in which the probabilities of selection are correlated with outcomes, and the design-based methods generally ensure that inference can be generalized to the finite population even when the statistical models used by analysts and researchers are not specified correctly (??).

Often, survey statisticians have auxiliary information on the units in the frame, and such information can be included it at the sampling stage to create more efficient designs. Unequal probabilities of selection are then controlled with probability weights, implemented as `[pw=exp]` in Stata (and can be permanently affixed to the data set with `svyset` command).

In many situations, however, usable information is not available beforehand, and may only appear in the collected data. The census totals of the age and gender distribution of the population may exist, but age and gender of the sampled units is unknown until the survey measurement is taken on them. It is still possible to capitalize on this additional data by adjusting the weights in such a way that the reweighted data conforms to these known figures. The procedures to perform these reweighting steps are generally known as *weight calibration* (?????).

Suppose there are several (categorical) variables, referred to as *control variables*, that are available for both the population and the sample (age groups, race, gender, educational attainment, etc.). Weight calibration aims at adjusting the margins, or low level interactions, via an iterative optimization aimed at satisfying the *control totals* for the control variables $\mathbf{x} = (x_1, \dots, x_p)$:

$$\sum_{j \in \mathcal{S}} w_{3j} \mathbf{x}_j = T[\mathbf{X}_j] \quad (4)$$

where the right hand side is assumed to be known from a census or a higher quality survey. ? framed the problem of finding a suitable set of weights as that of constrained optimization with the control equations (4) serving as constraints, and optimization targeted at making the discrepancy between the design weights w_{1j} and calibrated weights w_{3j} as close as possible, in a suitable sense.

In package `ipfraking` (?), I implemented a popular calibration algorithm, known as *iterated proportional fitting*, or as *raking*, which consists of iterative updating (post-stratification) of each of the margins. (For an in-depth discussion of distinctions between raking and post-stratification, see ?.) Since 2014, the continuing code development resulted in additional features that this update documents.

2 Package description

Below, I provide full syntax, and list the new features in a dedicated section.

2.1 Syntax of ipfraking

```
ipfraking [if] [in] [weight] , ctotal(matname [matname ...]) [
  generate(newvarname) replace double iterate(#) tolerance(#)
  ctrltolerance(#) trace nodivergence trimhiabs(#) trimhirel(#)
  trimloabs(#) trimlorel(#) trimfrequency(once|sometimes|often) double
  meta nograph ]
```

Note that the weight statement [**pw**=varname] is required, and must contain the initial weights.

Required options

ctotal(matname [matname ...]) supplies the names of the matrices that contain the control totals, as well as meta-data about the variables to be used in calibration.

□ Technical note

The row and column names of the control total matrices (see [P] **matrix rownames**) should be formatted as follows.

- **rownames**: the name of the control variable
- **colnames**: the values the control variables takes
- **coleq**: the name of the variable for which total is computed; typically it is identically equal to 1.

See examples in Section ??.

□

generate(newvarname) contains the name of the new variable to contain the raked weights.

replace indicates that the weight variable supplied in the [**pw**=varname] expression should be overwritten with the new weights.

One and only one of generate() or replace must be specified.

Linear calibration

linear requests linear calibration of weights.

Options to control convergence

`tolerance(#)` defines convergence criteria (the change of weights from one iteration to next). The default is 10^{-6} .

`iterate(#)` specifies the maximum number of iterations. The default is 2000.

`nodivergence` overrides the check that the change in weights is greater at the current iteration than in the previous one, i.e., ignores this termination condition. It is generally recommended, especially in calibration with simultaneous trimming.

`ctrltolerance(#)` defines the criterion to assess the accuracy of the control totals. It does not impact iterations or convergence criteria, but rather only triggers alerts in the output. The default value is 10^{-6} .

`trace` requests a trace plot to be added.

Trimming options

`trimhiabs(#)` specifies the upper bound U on the greatest value of the raked weights. The weights that exceed this value will be trimmed down, so that $w_{3j} \leq U$ for every $j \in \mathcal{S}$.

`trimhirel(#)` specifies the upper bound u on the adjustment factor over the baseline weight. The weights that exceed the baseline times this value will be trimmed down, so that $w_{3j} \leq uw_{1j}$ for every $j \in \mathcal{S}$.

`trimloabs(#)` specifies the lower bound L on the smallest value of the raked weights. The weights that are smaller than this value will be increased, so that $w_{3j} \geq L$ for every $j \in \mathcal{S}$.

`trimlorel(#)` specifies the lower bound l on the adjustment factor over the baseline weight. The weights that are smaller than the baseline times this value will be increased, so that $w_{3j} \geq lw_{1j}$ for every $j \in \mathcal{S}$.

`trimfrequency(keyword)` specifies when the trimming operations are to be performed. The following keywords are recognized:

`often` means that trimming will be performed after each marginal adjustment.

`sometimes` means that trimming will be performed after a full set of variables has been used for post-stratification. This is the default behavior if any of the numeric trimming options above are specified.

`once` means that trimming will be performed after the raking process is declared to have converged.

The numeric trimming options `trimhiabs(#)`, `trimhirel(#)`, `trimloabs(#)`, `trimlorel(#)` can be specified in any combination, or entirely omitted to produce untrimmed weights. By default, there is no trimming.

Miscellaneous options

double specifies that the new variable named in **generate()** option should be generated as double type. See [D] **data types**.

meta puts information taken by **ipfraking** as inputs and produced throughout the process into characteristics stored with the variable specified in **generate()** option. See Section ??.

nograph omits the histogram of the calibrated weights, which can be used to speed up **ipfraking** (e.g., in replicate weight production).

2.2 New features of ipfraking

Since the first publication, the following features and options were added.

Reporting of results and errors by **ipfraking** was improved in several directions.

1. The discrepancy for the worst fitting category is now being reported.
2. The number of trimmed observations is reported.
3. If **ipfraking** determines that the categories do not match in the control totals received from **ctotals()** and those found in the data, a full listing of categories is provided, and the categories not found in one or the other are explicitly shown.

Linear calibration (Case 1 of ?) is provided with **linear** option. The weights are calculated analytically:

$$w_{j,\text{lin}} = w_{1j}(1 + \mathbf{x}'_j \lambda). \quad \lambda = \left(\sum_{j \in \mathcal{S}} w_{1j} \mathbf{x}_j \mathbf{x}'_j \right)^{-1} (T[\mathbf{X}_j] - t_1[y]) \quad (5)$$

This works very fast, but has an undesirable artefact of producing negative weights, as the range of weights is not controlled. (As raking works by multiplying the currents weights by positive factors, if the input weights are all positive, the output weights will be positive as well.) Negative weights are not allowed by the official **svy** commands or commands that work with **[pweights]**. In my experience, running linear weights first, pulling up the negative weights (**replace weight = 1 if weight <= 1**) and re-raking using the iterative proportional fitting “proper” runs faster than raking from scratch.

Option **meta** saves more information in characteristics of the calibrated weight variables.

(Continued on next page)

```

. capture drop rakedwgt3
. ipfraking [pw=finalwgt], gen( rakedwgt3 ) ///
>   cttotal( ACS2011_sex_age Census2011_region Census2011_race ) ///
>   trimhiabs(200000) trimloabs(2000) meta

Iteration 1, max rel difference of raked weights = 14.95826
Iteration 2, max rel difference of raked weights = .21474256
Iteration 3, max rel difference of raked weights = .02754514
Iteration 4, max rel difference of raked weights = .00511347
Iteration 5, max rel difference of raked weights = .00095888
Iteration 6, max rel difference of raked weights = .00018036
Iteration 7, max rel difference of raked weights = .00003391
Iteration 8, max rel difference of raked weights = 6.377e-06
Iteration 9, max rel difference of raked weights = 1.199e-06
Iteration 10, max rel difference of raked weights = 2.254e-07
The worst relative discrepancy of 3.0e-08 is observed for race == 3
Target value = 20053682; achieved value = 20053682
Trimmed due to the upper absolute limit: 5 weights.

Summary of the weight changes

```

	Mean	Std. dev.	Min	Max	CV
Orig weights	11318	7304	2000	79634	.6453
Raked weights	22055	18908	4033	200000	.8573
Adjust factor	2.1486		0.9220	18.9828	

```

. char li rakedwgt3[]
rakedwgt3[source]:      finalwgt
rakedwgt3[objfcn]:      2.25435521346e-07
rakedwgt3[maxctrl]:     3.00266822363e-08
rakedwgt3[converged]:   1
rakedwgt3[worstcat]:    3
rakedwgt3[worstvar]:    race
rakedwgt3[command]:     [pw=finalwgt], gen( rakedwgt3 ) cttotal( ACS2011_sex_age Census2011_region ..
rakedwgt3[trimloabs]:   trimloabs(2000)
rakedwgt3[trimhiabs]:   trimhiabs(200000)
rakedwgt3[trimfrequency]: sometimes
rakedwgt3[hash1]:       2347674164
rakedwgt3[mat3]:        Census2011_race
rakedwgt3[over3]:       race
rakedwgt3[totalof3]:    _one
rakedwgt3[Census2011_race]: 7.48567503861e-09
rakedwgt3[mat2]:        Census2011_region
rakedwgt3[over2]:       region
rakedwgt3[totalof2]:    _one
rakedwgt3[Census2011_region]:
3.00266822363e-08
rakedwgt3[mat1]:        ACS2011_sex_age
rakedwgt3[over1]:       sex_age
rakedwgt3[totalof1]:    _one
rakedwgt3[ACS2011_sex_age]: 4.13778410340e-09
rakedwgt3[note1]:       Raking controls used: ACS2011_sex_age Census2011_region Census2011_race
rakedwgt3[note0]:       1

```

The following characteristics are stored with the newly created weight variable (see [P] **char**).

<code>command</code>	The full command as typed by the user
<code>matrix name</code>	The relative matrix difference from the corresponding control total, see [D] functions
<code>trimhiabs, trimloabs,</code> <code>trimhirel, trimlorel,</code> <code>trimfrequency</code>	Corresponding trimming options, if specified
<code>maxctrl</code>	the greatest <code>mreldif</code> between the targets and the achieved weighted totals
<code>objfcn</code>	the value of the relative weight change at exit
<code>converged</code>	whether <code>ipfraking</code> exited due to convergence (1) vs. due to an increase in the objective function or reaching the limit on the number of iterations (0)
<code>source</code>	weight variable specified as the <code>[pw=]</code> input
<code>worstvar</code>	the variable in which the greatest discrepancy between the targets and the achieved weighted totals (<code>maxctrl</code>) was observed
<code>worstcat</code>	the category of the <code>worstvar</code> variable in which the greatest discrepancy was observed

For the control total matrices $\# = 1, 2, \dots$, the following meta-information is stored.

<code>mat#</code>	the name of the control total matrix
<code>totalof#</code>	the multiplier variable (matrix' <code>coleg</code>
<code>over#</code>	the margin associated with the matrix (i.e., the categories represented by the columns)

Also, `ipfraking` stores the notes regarding the control matrices used, and which of the margins did not match the control totals, if any. See [D] **notes**.

2.3 Excel reports on raked weights: `ipfraking_report`

```
ipfraking_report using filename , raked.weight(varname) [
    matrices(namelist) by(varlist) xls replace force ]
```

The utility command `ipfraking_report` produces a detailed report describing the raked weights, and places it into `filename.dta` file (or, if `xls` option is specified, both `filename.dta` and `filename.xls` files).

Along the way, `ipfraking_report` runs a regression of the log raking ratio w_{3j}/w_{1j} on the calibration variables. This regression is expected to have R^2 very close to 1, and the regression coefficients provide insights regarding which categories received greater vs. smaller adjustments.

(Continued on next page)

```
. ipfraking_report using rakedwtg3-report, raked_weight(rakedwtg3) replace by(_one)
Margin variable sex_age (total variable: _one; categories: 11 12 13 21 22 23).
Margin variable region (total variable: _one; categories: 1 2 3 4).
Margin variable race (total variable: _one; categories: 1 2 3).
Auxiliary variable _one (categories: 1).
```

```
file rakedwtg3-report.dta saved
```

Source	SS	df	MS	Number of obs	=	10,351
Model	2086.13859	10	208.613859	F(10, 10340)	>	99999.00
Residual	.78315703	10,340	.000075741	Prob > F	=	0.0000
				R-squared	=	0.9996
				Adj R-squared	=	0.9996
Total	2086.92175	10,350	.201634952	Root MSE	=	.0087

__000003	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
sex_age						
11	.0644365	.0002775	232.21	0.000	.0638925	.0649804
12	.4545577	.0003154	1441.25	0.000	.4539395	.455176
13	.6782466	.0002804	2418.71	0.000	.6776969	.6787963
22	.3966406	.0003049	1300.84	0.000	.3960429	.3972383
23	.7304392	.0002726	2679.97	0.000	.7299049	.7309734
region						
NE	-.4455127	.0002536	-1756.49	0.000	-.4460099	-.4450155
MW	-.4428144	.0002335	-1896.53	0.000	-.4432721	-.4423567
W	-.6672675	.0002407	-2772.21	0.000	-.6677393	-.6667957
race						
Black	.3360321	.0002848	1180.08	0.000	.3354739	.3365902
Other	1.613276	.0006303	2559.34	0.000	1.612041	1.614512
_cons	.5864801	.0002455	2388.48	0.000	.5859988	.5869614

```
Raking adjustments for sex_age variable:
```

```
the smallest was 1.798 for category 21 (21)
the greatest was 3.732 for category 23 (23)
```

```
Raking adjustments for region variable (1=NE, 2=MW, 3=S, 4=W):
```

```
the smallest was 0.922 for category 4 (W)
the greatest was 1.798 for category 3 (S)
```

```
Raking adjustments for race variable (1=white, 2=black, 3=other):
```

```
the smallest was 1.798 for category 1 (White)
the greatest was 9.023 for category 3 (Other)
```

Options of ipfraking_report

`raked_weight(varname)` specifies the name of the raked weight variable to create the report for. This is a required option.

`matrices(namelist)` specifies a list of matrices (formatted as the matrices supplied to `ctotal()` option of `ipfraking`) to produce weighting reports for. In particular, the variables and their categories are picked up from these matrices; and the control totals/proportions are compared to those defined by the weight being reported on.

`by(varlist)` specifies a list of additional variables for which the weights are to be tabulated in the raking weights report. The difference with the `matrices()` option is that

the control totals for these variables may not be known (or may not be relevant). In particular, `by(_one)`, where `_one` is identically one, will produce the overall report.

`xls` requests exporting the report to an Excel file.

`replace` specifies that the files produced by `ipfraking_report` (i.e., the `.dta` and the `.xls` file if `xls` option is specified) should be overwritten.

`force` requires that a variable that may be found repeatedly (between the calibration variables supplied originally to `ipfraking`, the variables found in the independent total `matrices()`, and the variables without the control totals provided in `by()` option) is processed every time it is encountered. (Otherwise, it is only processed once.)

Variables in the raking report

The following variables are saved in the raking report.

(Continued on next page)

Variable name	Definition
<code>Weight_Variable</code>	The name of the weight variable, <code>generate()</code>
<code>C.Total.Margin.Variable.Name</code>	The name of the control margin, <code>rowname</code> of the corresponding <code>ctotal()</code> matrix
<code>C.Total.Margin.Variable.Label</code>	The label of the control margin variable
<code>Variable_Class</code>	The role of the variable in the report: Raking margin: a variable used as a calibration margin (picked up automatically from the <code>ctotal()</code> matrix, provided <code>meta</code> option was specified) Other known target: supplied with <code>matrices()</code> option of <code>ipfraking_report</code> Auxiliary variable: additional variable supplied with <code>by()</code> option of <code>ipfraking_report</code>
<code>C.Total.Arg.Variable.Name</code>	The name of the multiplier variable
<code>C.Total.Arg.Variable.Label</code>	The label of the multiplier variable
<code>C.Total.Margin.Category.Number</code>	Numeric value of the control total category
<code>C.Total.Margin.Category.Label</code>	Label of the control total category
<code>Category.Total.Target</code>	The control total to be calibrated to (the specific entry in the <code>ctotal()</code> matrix)
<code>Category.Total.Prop</code>	Control total proportion (the ratio of the specific entry in the <code>ctotal()</code> matrix to the matrix total)
<code>Unweighted.Count</code>	Number of sample observations in the category
<code>Unweighted.Prop</code>	Unweighted proportion
<code>Unweighted.Prop.Discrep</code>	Difference <code>Unweighted.Prop</code> - <code>Category.Total.Prop</code>
<code>Category.Total.SRCWGT</code>	Weighted category total, with source weight
<code>Category.Prop.SRCWGT</code>	Weighted category proportion, with source weight
<code>Category.Total.Discrep.SRCWGT</code>	Difference <code>Category.Total.SRCWGT</code> - <code>Category.Total.Target</code>
<code>Category.Prop.Discrep.SRCWGT</code>	Difference <code>Category.Prop.SRCWGT</code> - <code>Category.Total.Prop</code>
<code>Category.RelDiff.SRCWGT</code>	<code>reldif(Category.Total.SRCWGT, Category.Total.Target)</code>
<code>Overall.Total.SRCWGT</code>	Sum of source weights
<code>Source</code>	The name of the matrix from which the totals were obtained
<code>Comment</code>	Placeholder for comments, to be entered during manual review

For each of the input weights (SRCWGT suffix), raked weights (RKDWGT suffix) and raking ratio (the ratio of raked and input weights, RKDRATIO suffix), the following summaries are provided.

Variable name	Definition
<code>Min_WEIGHT</code>	Min of source weights
<code>P25_WEIGHT</code>	25th percentile of source weights
<code>P50_WEIGHT</code>	Median of source weights
<code>P75_WEIGHT</code>	75th percentile of source weights
<code>Max_WEIGHT</code>	Max of source weights
<code>Mean_WEIGHT</code>	Mean of source weights
<code>SD_WEIGHT</code>	Standard deviation of source weights
<code>DEFF_WEIGHT</code>	Apparent UWE DEFF of source weights

Example

2.4 Collapsing weighting cells: `wgtcellcollapse`

2.5 Utility programs

The original package `ipfraking` provided two additional utility programs, `mat2do` and `xls2row`. An additional utility program was added to compute the design effects and margins of error, common tasks associated with describing survey weights. Specifically, the Transparency Initiative of the American Association for Public Opinion Research (?) requires that

For probability samples, the estimates of sampling error will be reported, and the discussion will state whether or not the reported margins of sampling error or statistical analyses have been adjusted for the design effect due to weighting, clustering, or other factors.

`whatsdeff weight_variable [if] [in] , [by(varlist)]`

The utility program `whatsdeff` calculates the apparent design effect due to unequal weighting, $DEFF_{UWE} = 1 + CV_w^2 = 1 + r(Var)/(r(mean))^2$ from `summarize weight_variable`. Additionally, it reports the effective sample size, $n/DEFF_{UWE}$, and also returns the margins of error for the sample proportions that estimate the population proportions of 10% and 50%.

```
. webuse nhanes2, clear
. whatsdeff finalwgt
```

Group	Min	Mean	Max	CV	DEFF	N	N eff
Overall	2000.00	11318.47	79634.00	0.6453	1.4164	10351	7307.97

```
. return list
scalars:
      r(N) = 10351
      r(MOE10) = .0068792766212984
      r(MOE50) = .0114654610354974
      r(Neff_Overall) = 7307.97435325364
```

```

r(DEFF_Overall) = 1.416397964696134
. whatsdeff finalwgt, by(sex)

```

Group	Min	Mean	Max	CV	DEFF	N	N eff
sex							
Male	2000.00	11426.14	79634.00	0.6578	1.4326	4915	3430.94
Female	2130.00	11221.12	61534.00	0.6333	1.4010	5436	3880.01
Overall	2000.00	11318.47	79634.00	0.6453	1.4164	10351	7307.97

```

. return list
scalars:
      r(N) = 10351
      r(MOE10) = .0068792766212984
      r(MOE50) = .0114654610354974
      r(Neff_Overall) = 7307.97435325364
      r(DEFF_Overall) = 1.416397964696134
      r(Neff_Female) = 3880.00710397866
      r(DEFF_Female) = 1.40102836266093
      r(Neff_Male) = 3430.938195872213
      r(DEFF_Male) = 1.432552765279559

```

3 Examples

3.1 Basic syntax and input requirements

In this very simple example, I shall demonstrate the basic mechanics of `ipfraking`, its input requirements and output. These examples are intended to only demonstrate the syntax and the output of `ipfraking`, and may or may not provide substantively meaningful results.

► Example 1

We shall work with the standard example of `svy` data, an excerpt from the NHANES II data set available from Stata Corp. website. We shall introduce some small changes to the data so that `ipfraking` will have some work to do.

```
. webuse nhanes2, clear
. generate byte _one = 1
.
. svy : total _one , over( sex, nolabel )
(running total on estimation sample)
Survey: Total estimation
Number of strata =      31      Number of obs   =      10351
Number of PSUs   =      62      Population size = 117157513
                                   Design df      =        31

      1: sex = 1
      2: sex = 2
```

	Over	Linearized		[95% Conf. Interval]	
		Total	Std. Err.		
_one	1	5.62e+07	1377465	5.34e+07	5.90e+07
	2	6.10e+07	1396159	5.82e+07	6.38e+07

```
. matrix NHANES2_sex = e(b)
. matrix rownames NHANES2_sex = sex
.
. svy : total _one , over( race, nolabel )
(running total on estimation sample)
Survey: Total estimation
Number of strata =      31      Number of obs   =      10351
Number of PSUs   =      62      Population size = 117157513
                                   Design df      =        31

      1: race = 1
      2: race = 2
      3: race = 3
```

(Continued on next page)

Over	Linearized			
	Total	Std. Err.	[95% Conf. Interval]	
_one	1	1.03e+08	2912042	9.71e+07 1.09e+08
	2	1.12e+07	1458814	8213964 1.42e+07
	3	2968728	1252160	414930.1 5522526

```
. matrix NHANES2_race = e(b)
. matrix rownames NHANES2_race = race
.
. matrix NHANES2_sex[1,1] = NHANES2_sex[1,1]*1.25
. matrix NHANES2_race[1,1] = NHANES2_race[1,1]*1.4
.
```

Let us now look at the matrices that will serve as an input to the raking procedure.

```
. matrix list NHANES2_sex, f(%12.0g)
NHANES2_sex[1,2]
      _one:      _one:
      1      2
sex  70199350  60998033
. matrix list NHANES2_race, f(%12.0g)
NHANES2_race[1,3]
      _one:      _one:      _one:
      1      2      3
race 144199368.6  11189236  2968728
```

These input matrices are organized as follows. Input matrices always have a single row, just as estimation results $e(b)$ do. The column names follow the naming conventions of $e(b)$, namely, the name of the variable for which the total is being computed (here, `_one`) and the numeric categories of the variable that was used in the `over` option (here, `sex`, with values 1 for males and 2 for females; and `race`, with values 1 for whites, 2 for blacks, and 3 for other). These values must be in an increasing order. Since that variable is not stored in the $e(b)$ per se, it needs to be added to this matrix, which is done in the form of the row name. The entries of the matrix are the totals that the weights in the categories of the control variables need to sum up to. In this example, they are scaled to be the population totals. Alternatively, these can be made to sum up to the sample size, as is done sometimes in public opinion research, or to 1, which is what `proportion` estimation command would produce.

The input requirements in terms of control totals are thus made as simple as possible. If a higher quality survey is available, all the survey statistician needs to do is to obtain the totals for the categories of the control variables using `svy: total ... , over(... , nolabel)` and save the name of that variable along with the matrix. Note that the `total` is computed with `over(... , nolabel)` suboption to suppress the otherwise informative labeling of the categories; `ipfraking` expects the numeric values of the categories as column names (see [P] `matrix rownames`). The name of the matrix itself is immaterial, but it is a good programming practice to have informative names

(?). Thus the names of the matrices in the examples generally follow the convention *data_source.variable*.

We are now ready to run `ipfraking` and see what it produces.

```
. ipfraking [pw=finalwgt], ctotal( NHANES2_sex NHANES2_race ) gen( rakedwgt1 )
Warning: the totals of the control matrices are different:
  Target 1 (NHANES2_sex) total      =      131197383
  Target 2 (NHANES2_race) total     =      158357332.6
Iteration 1, max rel difference of raked weights = .56227988
Iteration 2, max rel difference of raked weights = .00073288
Iteration 3, max rel difference of raked weights = 2.356e-07
Warning: the controls NHANES2_sex did not match
Summary of the weight changes
```

	Mean	Std. dev.	Min	Max	CV
Orig weights	11318	7304	2000	79634	.6453
Raked weights	15299	10274	1914	90831	.6716
Adjust factor	1.3490		0.8846	1.5614	

In this simple case with just two control variables and the control totals that are not very different from the existing sample totals, the procedure converged very quickly in three iterations. A diagnostic message was produced upfront by `ipfraking` informing about apparent differences in total population counts as obtained from the different control total matrices. As a result, the control totals for the variable that was adjusted first (`sex`) could not match the required control totals even after the weights converged in the sense of differing little between iterations. Both of these warnings are only produced when problems are encountered.

The summary table is always produced, and shows some relevant characteristics of the original weights w_{1j} , the raked weights w_{3j} , and the raking ratios w_{3j}/w_{1j} . As expected, the coefficient of variation went up from 0.645 to 0.672.

The graphic output produced by `ipfraking` is shown on Figure ???. Generally, we would want to inspect these graphs to see if there any unexpected patterns, such as highly outlying values, gaps in the distribution (here, there are only six distinct values of the adjustment factor corresponding to the 2×3 combinations of the control variables) or concentration near the limits of the weight range (as is typical for trimmed weights, see below in section ??). Also, these graphs may inform later trimming decisions: the trimming limits can be chosen to conform to the breaks in the distributions of the untrimmed raked weights.

Figure 1: Histograms of the raked weights and calibration ratios, Example 1.

Table 1: Control totals for the 2011 US population.

Group	Population
ACS 2011 1-year estimates, Table S0101	
Male, total	153,267,860
Ages 20–39	27.4%
Ages 40–59	27.5%
Ages 60+	17.3%
Female, total	158,324,057
Ages 20–39	26.0%
Ages 40–59	27.6%
Ages 60+	20.7%
US Census Bureau 2011 projections, Table NST-EST2011-01	
Northeast	55,521,598
Midwest	67,158,835
South	116,046,736
West	72,864,748
US Census Bureau 2011 projections, Table NC-EST2011-03	
White	243,470,497
Black	40,750,746
Other	27,370,674
Total	311,591,917

3.2 Preparing control matrices from scratch

In many situations, the control totals will be obtained from outside of Stata, and need to be prepared to work with `ipfraking`.

► Example 2

Suppose I wanted to calibrate the NHANES II data set to the latest control totals available from the US Census Bureau website. Using the tables S0101 from the 2011 American Community Survey 1-year estimates and NST-EST2011 from the US Census Bureau population projections, the latest available at the time of writing this paper, the figures displayed in Table ?? can be obtained.

Thus, we have information in the two-way age by sex table, as well as two additional margins. We shall need an additional sex-by-age group variable, and we shall try to make its values somewhat informative (e.g., the value 12 of the variable `sex_age` means the first group of sex and the second group of age):

```
. generate byte age_grp = 1 + (age>=40) + (age>=60) if !mi(age)
. generate sex_age = sex*10 + age_grp
```

With that, the matrices will have to be defined explicitly, and their labels need to be hand-coded, too (see [P] `matrix rownames`). Note that the US Census Bureau 2011

projections relate to the total population, while the target population of the study is the population age 20+. Assuming that the age structure is the same across regions and races, the control totals for region and race need to be rescaled to the adult population to avoid the warning messages. (More accurate figures can be obtained from ACS microdata which can be downloaded from the U.S. Census Bureau website.)

```
. matrix ACS2011_sex_age = ( ///
>   153267860*0.274, 153267860*0.275, 153267860*0.173, /// males
>   158324057*0.260, 158324057*0.276, 158324057*0.207 /// females
> )

. matrix colnames ACS2011_sex_age = 11 12 13 21 22 23
. matrix coleq   ACS2011_sex_age = _one
. matrix rownames ACS2011_sex_age = sex_age
. scalar ACS2011_total_pop = 311591917
. matrix ACS2011_adult_pop = ACS2011_sex_age * J(colsof(ACS2011_sex_age),1,1)
. matrix Census2011_region = ///
>   (55521598, 67158835, 116046736, 72864748 )
. matrix Census2011_region = Census2011_region * ACS2011_adult_pop / ACS2011_to
> tal_pop
. matrix colnames Census2011_region = 1 2 3 4
. matrix coleq   Census2011_region = _one
. matrix rownames Census2011_region = region
. matrix Census2011_race = ///
>   (243470497, 40750746, 27370674 )
```

(Continued on next page)

```
. matrix Census2011_race = Census2011_race * ACS2011_adult_pop / ACS2011_total_
> pop
. matrix colnames Census2011_race = 1 2 3
. matrix coleq    Census2011_race = _one
. matrix rownames Census2011_race = race
```

Let us check the matrix entries and labels once again before producing the weights. Note that the values of the control variable categories are given in an increasing order.

```
. matrix list ACS2011_sex_age, f(%10.0g)
ACS2011_sex_age[1,6]
      _one:      _one:      _one:      _one:      _one:      _one:
      11      12      13      21      22      23
sex_age 41995394 42148662 26515340 41164255 43697440 32773080
. matrix list Census2011_region, f(%10.0g)
Census2011_region[1,4]
      _one:      _one:      _one:      _one:
      1      2      3      4
region 40679030 49205289 85024007 53385843
. matrix list Census2011_race, f(%11.0g)
Census2011_race[1,3]
      _one:      _one:      _one:
      1      2      3
race 178383622 29856864.7 20053682.2
```

As the labels appear to be in place, let us run `ipfraking`:

```
. ipfraking [pw=finalwgt], gen( rakedwgt2 ) ///
>      ctotal( ACS2011_sex_age Census2011_region Census2011_race )

Iteration 1, max rel difference of raked weights = 14.95826
Iteration 2, max rel difference of raked weights = .19495004
Iteration 3, max rel difference of raked weights = .02204455
Iteration 4, max rel difference of raked weights = .00315355
Iteration 5, max rel difference of raked weights = .00043857
Iteration 6, max rel difference of raked weights = .00006061
Iteration 7, max rel difference of raked weights = 8.365e-06
Iteration 8, max rel difference of raked weights = 1.154e-06
Iteration 9, max rel difference of raked weights = 1.593e-07

Summary of the weight changes
```

	Mean	Std. dev.	Min	Max	CV
Orig weights	11318	7304	2000	79634	.6453
Raked weights	22055	19227	4050	338675	.8717
Adjust factor	2.1464		0.9264	18.3694	

The diagnostic plots for these weights are given in Figure ???. They do appear to have some outlying cases (which are not very clearly seen on these plots as they are single count observations with outlying weights), and we shall address them in the next section with trimming.

◀

Figure 2: Histograms of the raked weights and calibration ratios, Example 2.

3.3 Trimming options

As discussed in Section ?? above, if variability of the weights becomes excessive, the weights can be trimmed by restricting the extremes. Using `ipfraking` options, upper and/or lower limits can be defined for either the absolute values of the weights or the relative changes from the base weights. The frequency of the trimming operations can also be controlled. Trimming can be applied once to the final data (`trimfreq(once)`) at step ?? of Algorithm 2. Alternatively, trimming can be applied after every full cycle over variables at step ?? of Algorithm 2. Finally, trimming can be applied after each sub-iteration at step ?? of the algorithm.

► Example 3

Inspecting the histograms on Figure ??, it appears reasonable to restrict the upper tail of the raked weights. A more detailed investigation of the histogram reveals a somewhat greater concentration of the raked weights around the value of 160,000, and sparse bars beyond 200,000. This latter number will be used as the top cut-off point for trimming, and is provided as an input to `ipfraking` via option `trimhiabs`. Also, I specified the absolute lower bound of 2,000, which is the minimum of the original weights, but, as the output in the previous example suggested, the calibrated weights tend to run above 4,000, so specifying the lower limit as `trimloabs(2000)` may not really affect the calibration procedure.

```
. ipfraking [pw=finalwgt], gen( rakedwgt3 ) ///
>   ctotat( ACS2011_sex_age Census2011_region Census2011_race ) ///
>   trimhiabs( 200000 ) trimloabs( 2000 )

Iteration 1, max rel difference of raked weights = 14.95826
Iteration 2, max rel difference of raked weights = .21474256
Iteration 3, max rel difference of raked weights = .02754514
Iteration 4, max rel difference of raked weights = .00511347
Iteration 5, max rel difference of raked weights = .00095888
Iteration 6, max rel difference of raked weights = .00018036
Iteration 7, max rel difference of raked weights = .00003391
Iteration 8, max rel difference of raked weights = 6.377e-06
Iteration 9, max rel difference of raked weights = 1.199e-06
Iteration 10, max rel difference of raked weights = 2.254e-07

Summary of the weight changes
```

	Mean	Std. dev.	Min	Max	CV
Orig weights	11318	7304	2000	79634	.6453
Raked weights	22055	18908	4033	200000	.8573
Adjust factor	2.1486		0.9220	18.9828	

The resulting coefficient of variation of weights, 0.857, is slightly better than that with unrestricted range of weights, 0.872. The summary also shows that the weights were capped at 200,000, as requested.

Setting the absolute limits on the range of the raked weights is often very subjective. A somewhat better plan might be to set limits in terms of the range of the adjustment factors, as shown in the next example. The relative change in the weights can be bounded with `trimlorel()` and `trimhirel()` options. I also demonstrate here how to use the results of `summarize` to feed into `ipfraking`. While ensuring that accurate numbers are being carried over in the context of the code, the approach is fragile for interactive work: simply running the single line with the sole `ipfraking` command that refers to the `r()` return values may break down if `summarize` was not the immediately preceding command.

```
. sum finalwgt
  Variable |      Obs      Mean   Std. Dev.      Min      Max
-----|-----
  finalwgt |    10351   11318.47   7304.04     2000   79634
. ipfraking [pw=finalwgt], gen( rakedwgt4 ) ///
>   cttotal( ACS2011_sex_age Census2011_region Census2011_race ) ///
>   trimhiabs(`=2.5*r(max)`) trimloabs(`=r(min)`) trimhirel(6)
```

(Continued on next page)

```

Iteration 1, max rel difference of raked weights = 5
Iteration 2, max rel difference of raked weights = .25592859
Iteration 3, max rel difference of raked weights = .0626759
Iteration 4, max rel difference of raked weights = .0158786
Iteration 5, max rel difference of raked weights = .00299304
Iteration 6, max rel difference of raked weights = .00070812
Iteration 7, max rel difference of raked weights = .00016401
Iteration 8, max rel difference of raked weights = .00003734
Iteration 9, max rel difference of raked weights = 8.434e-06
Iteration 10, max rel difference of raked weights = 1.898e-06
Iteration 11, max rel difference of raked weights = 4.265e-07
Warning: the controls ACS2011_sex_age did not match
Warning: the controls Census2011_region did not match
Warning: the controls Census2011_race did not match

```

Summary of the weight changes

	Mean	Std. dev.	Min	Max	CV
Orig weights	11318	7304	2000	79634	.6453
Raked weights	21830	18115	4113	199085	.8298
Adjust factor	2.1323		0.8973	6.0000	

◀

Setting the trimming options too aggressively may lead to adverse consequences. First, it may bias the estimates, as discussed in Section ???. Second, as this example demonstrates, it can impede (statistical) convergence: the output contains multiple warnings about targets not being achieved within desired accuracy, while no problems were encountered without trimming.

3.4 Tracking convergence

Let us now look in more detail into the issue of trimming frequency, and demonstrate another diagnostic plot that can be produced by `ipfraking`.

► Example 4

We return to the first set of options of Example 3, and re-run the raking procedure.

```

. capture drop rakedwgt3
. ipfraking [pw=finalwgt], gen( rakedwgt3 ) ///
>   ctotat( ACS2011_sex_age Census2011_region Census2011_race ) ///
>   trimhiabs(200000) trimloabs(2000) trimfreq(sometimes) trace
Iteration 1, max rel difference of raked weights = 14.95826
(output omitted)
Iteration 10, max rel difference of raked weights = 2.254e-07

```

Summary of the weight changes

	Mean	Std. dev.	Min	Max	CV
Orig weights	11318	7304	2000	79634	.6453
Raked weights	22055	18908	4033	200000	.8573
Adjust factor	2.1486		0.9220	18.9828	

The option `trace` requests that trace plots be added to the diagnostic plots, as shown on Figure ?? . The trace plots are presented on the absolute scale and on the log scale. The exponentially declining discrepancy appears to be a general phenomenon. In other words, after the first few iterations, discrepancy between the currently weighted totals to the control totals roughly follows the rate of $\text{const} \times \alpha^k$ for some $\alpha < 1$, where k is the (outer cycle) iteration number. When convergence is very slow or the sample size is very large, this rule may be helpful in determining the number of iterations necessary to achieve the required accuracy, and hence the expected computing time. Zero cross-cells and collinearity between the control variables may make the convergence factor α close to 1 thus hampering convergence. This happens when the control variables have very similar meaning, such as age and grade of children: it is impossible to have children of age 8 in grade 10. Also, sets of interactions of categorical variables, such as interactions of age group and education along with age group and race, are guaranteed to produce zero cells in the cross-tabulation: it is impossible to have any observations in the cells defined say by (age under 40 interacted with higher education) on one margin against (age above 60 interacted with white race) on the other.

Figure 3: Diagnostic plots for Example 4.

While `trimfreq(sometimes)` is the default in presence of other trimming options, the behavior can be changed with explicit specification of trimming frequency. Note that slightly different weights will be produced that way.

```
. ipfraking [pw=finalwgt], gen( rakedwgt5 ) ///
>   cttotal( ACS2011_sex_age Census2011_region Census2011_race ) ///
>   trimhiabs(200000) trimloabs(2000) trimfreq(often) trace
```

Iteration 1, max rel difference of raked weights = 14.95826
Iteration 2, max rel difference of raked weights = .21613885
Iteration 3, max rel difference of raked weights = .02673316
Iteration 4, max rel difference of raked weights = .00480164
Iteration 5, max rel difference of raked weights = .00086195
Iteration 6, max rel difference of raked weights = .00015444
Iteration 7, max rel difference of raked weights = .00002762
Iteration 8, max rel difference of raked weights = 4.940e-06
Iteration 9, max rel difference of raked weights = 8.832e-07

Summary of the weight changes

	Mean	Std. dev.	Min	Max	CV
Orig weights	11318	7304	2000	79634	.6453
Raked weights	22055	18905	4033	200000	.8572
Adjust factor	2.1487		0.9220	18.9844	

```
. compare rakedwgt3 rakedwgt5
```

	count	minimum	difference average	maximum
rakedwgt3<rakedwgt5	3638	-15.27963	-1.226753	-.0128687
rakedwgt3=rakedwgt5	4			
rakedwgt3>rakedwgt5	6709	.0011514	.6652557	2471.578
jointly defined	10351	-15.27963	.0000264	2471.578

total	10351
-------	-------

In this example, trimming the weights after adjusting each of the margins led to fewer iterations. This may or may not translate to lower overall computing times as more computing is performed within each iteration.

◀

3.5 Metadata

The results of raking operations can be stored with the newly created weight variables for later review and reproduction of the results. Let us reproduce the example in the previous section adding all the metadata available:

► Example 5

```
. capture drop rakedwgt3
. ipfraking [pw=finalwgt], gen( rakedwgt3 ) ///
>   cttotal( ACS2011_sex_age Census2011_region Census2011_race ) ///
>   trimhiabs(200000) trimloabs(2000) meta
```

```
Iteration 1, max rel difference of raked weights = 14.95826
Iteration 2, max rel difference of raked weights = .21474256
Iteration 3, max rel difference of raked weights = .02754514
Iteration 4, max rel difference of raked weights = .00511347
Iteration 5, max rel difference of raked weights = .00095888
Iteration 6, max rel difference of raked weights = .00018036
Iteration 7, max rel difference of raked weights = .00003391
Iteration 8, max rel difference of raked weights = 6.377e-06
Iteration 9, max rel difference of raked weights = 1.199e-06
Iteration 10, max rel difference of raked weights = 2.254e-07
The worst relative discrepancy of 3.0e-08 is observed for race == 3
Target value = 20053682; achieved value = 20053682
Trimmed due to the upper absolute limit: 5 weights.
```

Summary of the weight changes

	Mean	Std. dev.	Min	Max	CV
Orig weights	11318	7304	2000	79634	.6453
Raked weights	22055	18908	4033	200000	.8573
Adjust factor	2.1486		0.9220	18.9828	

```
. char li rakedwgt3[]
rakedwgt3[source]:      finalwgt
rakedwgt3[objfcn]:      2.25435521346e-07
rakedwgt3[maxctrl]:     3.00266822363e-08
rakedwgt3[converged]:   1
rakedwgt3[worstcat]:    3
rakedwgt3[worstvar]:    race
rakedwgt3[command]:     [pw=finalwgt], gen( rakedwgt3 ) cttotal( ACS2011_sex_age Census2011_region ..
rakedwgt3[trimloabs]:    trimloabs(2000)
rakedwgt3[trimhiabs]:    trimhiabs(200000)
rakedwgt3[trimfrequency]: sometimes
rakedwgt3[hash1]:       2347674164
rakedwgt3[mat3]:        Census2011_race
rakedwgt3[over3]:       race
```

```

rakedwgt3[totalof3]:      _one
rakedwgt3[Census2011_race]: 7.48567503861e-09
rakedwgt3[mat2]:         Census2011_region
rakedwgt3[over2]:        region
rakedwgt3[totalof2]:      _one
rakedwgt3[Census2011_region]:
                           3.00266822363e-08
rakedwgt3[mat1]:         ACS2011_sex_age
rakedwgt3[over1]:        sex_age
rakedwgt3[totalof1]:      _one
rakedwgt3[ACS2011_sex_age]: 4.13778410340e-09
rakedwgt3[notel]:        Raking controls used: ACS2011_sex_age Census2011_region Census2011_race
rakedwgt3[note0]:        1

```

◀

The following characteristics are stored with the newly created weight variable (see [P] **char**).

command	The full command as typed by the user
matrix name	The relative matrix difference from the corresponding control total, see [D] functions
trimhiabs, trimloabs, trimhirel, trimlorel, trimfrequency	Corresponding trimming options, if specified
maxctrl	the greatest mreldif between the targets and the achieved weighted totals
objfcn	the value of the relative weight change D_k (??) at exit
converged	whether ipfraking exited due to convergence (1) vs. due to an increase in the objective function or reaching the limit on the number of iterations (0)

Also, **ipfraking** stores the notes regarding the control matrices used, and which of the margins did not match the control totals, if any. See [D] **notes**.

3.6 Replicate weights

As discussed in Section ??, one of the greater challenges of weight calibration is ensuring that variance estimates take into account the greater precision achieved by adjusting the sample towards the fixed population quantities. As estimating the variances using linearization is cumbersome, replicate variance estimation may be more attractive.

► Example 6

The simplest code for calibrated replicate weights is obtained by calling **ipfraking** from within **bsweights** (?) which can pass the name of a replicate weight variable to an arbitrary calibration routine. In this example, we shall use the same settings as in Section ?? and thus we shall have the calibrated weight **rakedwgt2** which was produced in that example as the main weight for which the bootstrap weights provide the measure of sampling variability.


```

. set seed 2013
. set rmsg on
r; t=0.00 14:50:44
. bsweights bsw , reps(310) n(-1) balanced dots ///
>   calibrate( ipfraking [pw=@], replace nograph meta ///
>   cttotal( ACS2011_sex_age Census2011_region Census2011_race ) )
Balancing within strata:
.....
Rescaling weights
..... 50
..... 100
..... 150
..... 200
..... 250
..... 300
.....
r; t=178.79 14:53:43
. forvalues k=1/310 {
2.   _dots `k' 0
3.   assert `': char bsw`k'[converged]` == 1
4.   assert `': char bsw`k'[maxctrl]` < 10*c(epsfloat)
5. }
..... 50
..... 100
..... 150
..... 200
..... 250
..... 300
.....r; t=0.32 14:53:43
. set rmsg off
. svyset [pw=rakedwgt2], vce(bootstrap) bsrw( bsw* ) dof( 31 )
    pweight: rakedwgt2
      VCE: bootstrap
      MSE: off
    bsrweight: bsw1 bsw2 bsw3 bsw4 bsw5 bsw6 bsw7 bsw8 bsw9 bsw10 bsw11 bsw12
(output omitted)
              bsw301 bsw302 bsw303 bsw304 bsw305 bsw306 bsw307 bsw308 bsw309
              bsw310
    Design df: 31
    Single unit: missing
    Strata 1: <one>
      SU 1: <observations>
      FPC 1: <zero>

```

The options of `bsweights` request 310 replicate weights (a multiple of 31 strata), resample one less PSU than available in a given stratum, and obtain the first-order balance within a stratum. With the 2 PSU/stratum design and these options, `bsweights` produces random half-samples of data. The at-character @ is a placeholder for the name of the replicate weight variable. For explanations of these and other options of `bsweights`, see ?. The procedure took about 3 minutes on a laptop computer, which can be considered moderately computationally intensive beyond interactive. A new option of `ipfraking` in the above code is `nograph` that suppresses the histograms. The additional asserts (?) following the bootstrap weight generation demonstrate how the minimal quality assurance can be done on the bootstrap weights in the weight produc-

A more compact set of weights can be developed based on the existing BRR weights and a slightly more explicit code cycling over the weight variables:

		BRR			
		Proportion	Std. Err.	[95% Conf. Interval]	
highbp	0	.8941859	.0067023	.8805165	.9078553
	1	.1058141	.0067023	.0921447	.1194835

(Continued on next page)

Summary of the weight changes					
	Mean	Std. dev.	Min	Max	CV
Orig weights	11318	7304	2000	79634	.6453
Raked weights	22055	19227	4050	338675	.8717
Adjust factor	2.1464		0.9264	18.3694	

```

. forvalues k=1/32 {
2.   quietly ipfraking [pw=brr_`k'], gen( brrc_`k' ) nograph ///
>   ctotal( ACS2011_sex_age Census2011_region Census2011_race )
3.   _dots `k' 0
4. }
.....
. svyset [pw=rakedwgt2], vce(brr) brrw( brrc* ) dof( 31 )
    pweight: rakedwgt2
      VCE: brr
      MSE: off
    brrweight: brrc_1 brrc_2 brrc_3 brrc_4 brrc_5 brrc_6 brrc_7 brrc_8 brrc_9
               brrc_10 brrc_11 brrc_12 brrc_13 brrc_14 brrc_15 brrc_16
               brrc_17 brrc_18 brrc_19 brrc_20 brrc_21 brrc_22 brrc_23
               brrc_24 brrc_25 brrc_26 brrc_27 brrc_28 brrc_29 brrc_30
               brrc_31 brrc_32
    Design df: 31
    Single unit: missing
      Strata 1: <one>
        SU 1: <observations>
        FPC 1: <zero>

. svy : proportion highbp
(running proportion on estimation sample)
BRR replications (32)
|-----|-----|-----|-----|-----|
| 1 | 2 | 3 | 4 | 5 |
|-----|-----|-----|-----|-----|
.....
Survey: Proportion estimation      Number of obs   =      10351
                                   Population size  =  228294169
                                   Replications      =       32
                                   Design df         =       31

```

		BRR		
		Proportion	Std. Err.	[95% Conf. Interval]
highbp	0	.8730544	.0081501	.8564323 .8896766
	1	.1269456	.0081501	.1103234 .1435677

The data can be analyzed with the standard `svy` prefix, and the standard errors will appropriately capture the efficiency gains from weight calibration. No additional action is required for the analyst or researcher.

◀

CAUTION: the input weights for the replicate weight calibration must be the probability replicate weights. The existing NHANES II weights have been adjusted for non-response and calibrated by the data provider, and are used above for demonstration purposes only.

4 Error messages and troubleshooting

4.1 Critical errors

The following critical errors will stop execution of `ipfraking`.

`pweight` is required

The `[pweight=...]` component of `ipfraking` syntax is required. Probability weights must be specified as inputs to `ipfraking`.

`ctotal()` is required

The `ctotal()` component of `ipfraking` syntax is required. Names of the matrices containing the control totals must be specified.

one and only one of `generate()` or `replace` must be specified

Either `generate()` option with the name of the new variable must be supplied to `ipfraking`, or `replace` to replace the variable specified in `[pw=...]` statement.

raking procedure appears diverging

The maximum relative difference of weights D_k has increased from the previous iteration. This may or may not indicate a problem. Re-run `ipfraking` with `nodivergence` option to override the warning.

cannot process matrix *matrix_name*

For whatever reason, `ipfraking` could not process this matrix. The matrix may not have been defined or the variables in this matrix cannot be found.

variable *varname* corresponding to the control matrix *matrix_name*
not found

The variables contained in row or column names of this matrix cannot be found.

varname1 and *varname2* variables are not compatible

When running `total varname1, over(varname2)`, an error was encountered. One of the variables may be a string variable or have missing values resulting in an empty estimation sample.

categories of *varname* do not match in the control *matrix_name*
and in the data (nolab option)

There was a mismatch in the categories of *varname* found in the data and in the control matrix *matrix_name*. This could happen for any of the following reasons: (i) there were more categories in one than in the other; (ii) the entries are in the wrong order in the control matrix; (iii) the labels in the control matrix do not correspond to the category values in the data set; (iv) the control matrix was obtained via `total varname2, over(varname)`, but `nolabel` suboption of `over()` was omitted, and the labels of the control matrix may include some unexpected text. Tabulate *varname*

without labels, and compare the results to the matrix listing of the *matrix.name*.

cannot compute controls for *matrix.name* over *varname* with the current weights

This is a generic error message that something bad happened while **ipfraking** was computing the totals for the current set of weights. This error message should generally be very rare, but as computing the totals may be the slowest operation of the iterative optimization process, stopping **ipfraking** with a *Ctrl+Break* combination or the *Break* GUI button may produce this error message.

trimhiabs|trimloabs|trimhirel|trimlorel must be a positive number

One or more of the trimming options are given as a non-positive number or a non-number.

trimhiabs must be greater than trimloabs

trimhirel must be greater than trimlorel

The trimming parameters are illogical (the lower bound is greater than the upper bound). Respecify the values of the trimming parameters.

4.2 Other errors and warnings

The following warning messages may be produced by **ipfraking**. The program will continue running, but you must double-check the results for potential problems.

the totals of the control matrices are different

The sum of values of the control matrices are different. These sums will be listed for review. Convergence is still possible, but some of the control total checks are likely to fail.

trimfrequency() option is specified without numeric settings; will be ignored

The option **trimfrequency()** was specified without any numeric trimming options. There is no way to interpret this, and **ipfraking** will proceed without trimming.

trimfrequency() option is specified incorrectly, assume default value (sometimes)

Something other than **often**, **sometimes** or **once** was supplied in **trimfrequency**, and the default value is being used instead.

raking procedure did not converge

The maximum number of iterations was reached, but weights never met the convergence criteria (see step ?? of Algorithm 2 in Section ??). The user may want to increase the number of iterations or relax convergence criteria.

the controls *matrix_name* did not match

After convergence of weights was declared, `ipfraking` checked again the control totals, and found that the results differed from the target for one or more of the control total matrices. Any of the following can cause this: (i) the sum of entries of this particular matrix differs from the others; (ii) the trimming options are too restrictive, and do not allow the weights to adjust enough; (iii) the problem may not have a solution due to incompatible control totals or a bad sample.

division by zero weighted total encountered with *matrix_name* control

The weights for a category of the control variable summed to zero. `ipfraking` will skip calibration over this variable and proceed to the next one.

missing values of *varname* encountered; convergence will be impaired

A control variable has missing values in the calibration sample. There is little way for `ipfraking` to figure out how to deal with the weights for the observations with missing values. The user would need either to restrict the sample to non-missing values of all control variables, to impute the missing values or to create a separate category for the missing values of a given control variable (which may lead to difficulties in defining valid population control totals for it).

Acknowledgements

The author is grateful to Ben Phillips, Andrew Burkey and Brady West, as well as the editor and an anonymous referee, who suggested additional functionality and provided helpful comments to improve the readability of this article. The opinions stated in this paper are of the author only, and do not represent the position of Abt SRBI.

5 References

- AAPOR. 2014. *AAPOR Terms and Conditions for Transparency Certification*. The American Association for Public Opinion Research. Available at http://www.aapor.org/AAPOR_Main/media/MainSiteFiles/TI-Terms-and-Conditions-10-4-17.pdf.
- Binder, D. A., and G. R. Roberts. 2003. Design-based and Model-based Methods for Estimating Model Parameters. In *Analysis of Survey Data*, ed. R. L. Chambers and C. J. Skinner, chap. 3. New York: John Wiley & Sons.
- Deville, J. C., and C. E. Särndal. 1992. Calibration Estimators in Survey Sampling. *Journal of the American Statistical Association* 87(418): 376–382.
- Deville, J. C., C. E. Särndal, and O. Sautory. 1993. Generalized Raking Procedures in Survey Sampling. *Journal of the American Statistical Association* 88(423): 1013–1020.
- Gould, W. 2003. Stata tip 3: How to be assertive. *Stata Journal* 3(4).

- Groves, R. M., D. A. Dillman, J. L. Eltinge, and R. J. A. Little. 2001. *Survey Nonresponse*. Wiley Series in Survey Methodology, Wiley-Interscience.
- Holt, D., and T. M. F. Smith. 1979. Post Stratification. *Journal of the Royal Statistical Society, Series A* 142(1): 33–46.
- Horvitz, D. G., and D. J. Thompson. 1952. A Generalization of Sampling Without Replacement From a Finite Universe. *Journal of the American Statistical Association* 47(260): 663–685.
- Kolenikov, S. 2010. Resampling inference with complex survey data. *The Stata Journal* 10: 165–199.
- . 2014. Calibrating survey data using iterative proportional fitting. *The Stata Journal* 14(1): 22–59.
- . 2016. Post-stratification or non-response adjustment? *Survey Practice* 9(3). Available at <http://www.surveypractice.org/index.php/SurveyPractice/article/view/315>.
- Korn, E. L., and B. I. Graubard. 1995. Analysis of Large Health Surveys: Accounting for the Sampling Design. *Journal of the Royal Statistical Society, Series A* 158(2): 263–295.
- . 1999. *Analysis of Health Surveys*. John Wiley and Sons.
- Kott, P. S. 2006. Using Calibration Weighting to Adjust for Nonresponse and Coverage Errors. *Survey Methodology* 32(2): 133–142.
- . 2009. Calibration Weighting: Combining Probability Samples and Linear Prediction Models. In *Sample Surveys: Inference and Analysis*, ed. D. Pfeffermann and C. R. Rao, vol. 29B of *Handbook of Statistics*, chap. 25. Oxford, UK: Elsevier.
- McConnell, S. 2004. *Code Complete: A Practical Handbook of Software Construction*. 2nd ed. Microsoft Press.
- Pew Research Center. 2012. Assessing the Representativeness of Public Opinion Surveys. Technical report, Pew Research Center for People and Press. Available at [http://www.people-press.org/files/legacy-pdf/Assessing the Representativeness of Public Opinion Surveys.pdf](http://www.people-press.org/files/legacy-pdf/Assessing%20the%20Representativeness%20of%20Public%20Opinion%20Surveys.pdf).
- Pfeffermann, D. 1993. The role of sampling weights when modeling survey data. *International Statistical Review* 61: 317–337.
- Särndal, C.-E. 2007. The calibration approach in survey theory and practice. *Survey Methodology* 33(2): 99–119.
- Thompson, M. E. 1997. *Theory of Sample Surveys*, vol. 74 of *Monographs on Statistics and Applied Probability*. New York: Chapman & Hall/CRC.

Appendix: Common notation

\mathcal{C}_k	Sec. ??	Calibration cell
D_k	(??)	Maximum relative difference of weights from iteration $k - 1$ to iteration k
δ_D	Sec. ??	Convergence criteria for D_k
δ_T	Sec. ??	Quality control criteria for control totals
i	Sec. ??	Subscript i usually applies to units in population
j	Sec. ??	Subscript j usually applies to units in sample
k	Sec. ??	The outer cycle iteration number
K	Sec. ??	The maximum number of the outer cycle iterations
l	Sec. ??	Relative limit on weights: all the weights will be made $\geq (l \times$ the input weight)
l		summation index, where I run out of other traditional integer letters
L	Sec. ??	Absolute limit on weights: all the weights will be made $\geq L$
n		Sample size; number of sampled units
N		Population size; number of units in population or frame
π_i		Probability of selection of unit i specified by the sampling design
\mathcal{S}	Sec. ??	Sample; set of sampled units
$T[y]$	(1)	Population-based total of variable $[y]$
$t_m[y]$	(3)	Sample-based weighted estimate of the total $T[y]$; subscript $m = 1, 2, 3$ indicates the type of weights used in computing the total
u	Sec. ??	Relative upper limit on weights: all the weights will be made $\leq (u \times$ the input weight)
U	Sec. ??	Absolute upper limit on weights: all the weights will be made $\leq U$
\mathcal{U}	Sec. ??	Universe or population; set of units in population
w_{1i}	(2)	Probability (design) weights; inverse probability of selection
w_{2j}	(??)	Post-stratified weights; random; depend on group sizes in sample; analytically computable
w_{3j}	Sec. ??	Calibrated (raked) weights; random; require iterative optimization
x_v	Sec. ??	the v -th calibration (control) variable, $v = 1, \dots, p$. The population total $T[x_v]$ is known.

About the author

Stanislav (Stas) Kolenikov is a Senior Survey Statistician at Abt SRBI. His research interests include applications of statistical methods in public opinion research, such as advanced sampling techniques, survey weighting, calibration, missing data imputation, and variance estimation. Besides survey statistics, Stas has extensive experience developing and applying statistical methods in social sciences, with focus on structural equation modeling and microeconometrics. He has been writing Stata programs since 1998 when Stata was version 5.