The Stata Journal (2021)
21, Number 1, pp. 206–219 DOI: 10.1177/1536867X211000025

kmr: A command to correct survey weights for unit nonresponse using groups' response rates

Ercio Muñoz
Stone Center on Socio-Economic Inequality at CUNY Graduate Center
New York, NY
emunozsaavedra@gc.cuny.edu

Salvatore Morelli
University of Roma Tre,
Rome, Italy, and
Stone Center on Socio-Economic Inequality at CUNY Graduate Center
New York, NY
smorelli@gc.cuny.edu

Abstract. In this article, we describe `kmr`, a command to estimate a microcompliance function using groups' nonresponse rates (Korinek, Mistiaen, and Ravallion, 2007, *Journal of Econometrics* 136: 213–235), which can be used to correct survey weights for unit nonresponse. We illustrate the use of `kmr` with an empirical example using the current population survey and state-level nonresponse rates.

Keywords: st0634, kmr, sample surveys, selective unit nonresponse bias, survey reweighting

1 Introduction

Unit nonresponse rates in household socioeconomic surveys have been increasing over the last decades (Meyer, Mok, and Sullivan 2015). Unit nonresponse is problematic for the measurement of inequality and poverty when response is not random, especially when it is related to the variable of interest.

There is evidence that household income systematically affects survey response. Using the current population survey (CPS) of the United States, Bollinger et al. (2019) show that nonresponse increases in the tails of the income distribution. This empirical evidence rejects the ignorability assumption (the fact that nonresponse is random within some arbitrary subgroup of the population). Moreover, they show that approximately one-third to one-half of the difference in inequality measures between the survey and administrative data (tax records) is accounted for by nonresponse.

Korinek, Mistiaen, and Ravallion (2007, 2006) show how the latent income effect on compliance can be consistently estimated with the available data on average response rates by groups (for example, geographic areas) and the measured distribution of income across them. This strategy has been recently used with data of several countries (see Hlasny and Verme [2018a,b] and Hlasny [2020]). In this article, we present `kmr`, a new command in Stata to implement this method. We also illustrate its use with an empirical example using the 2018 CPS data and state-level response rates.

This article is organized as follows. In section 2, we describe the methodology. In section 3, we describe the `kmr` command. In section 4, we illustrate the use of the command with an empirical example, and in section 5 we conclude.

2 Methodology

As described in Korinek, Mistiaen, and Ravallion (2006, 2007), the proposed method has two main advantages: First, it does not assume that within the smallest subgroup the decision to respond is independent of income (ignorability assumption). Second, it relies only on the survey data and does not require any external information.

Here we sketch how the estimator is derived. We start by assuming that the probability of response denoted by $P(D_\epsilon = 1)$, where D_ϵ is an indicator function equal to 1 when the household ϵ responds, depends on a K-vector $\mathbf{X}_\epsilon$ [that is, $P(D_\epsilon = 1) = f(\mathbf{X}_\epsilon)$]. We observe the response rate for J groups together with the values of $\mathbf{X}$ for all the respondents, and the respondents can be divided into I patterns according to the observed values of $\mathbf{X}$ indexed by $i \in I$.

For a given group $j \in J$, the mass of respondents with a given value of $\mathbf{X}$ belonging to pattern i denoted by m^1_{ij} can be defined as

$$m^1_{ij} = \int_0^{m_{ij}} D_{ij\epsilon} d\epsilon$$

where m_{ij} is the total (unobserved) number of households with a value of $\mathbf{X}$ belonging to pattern i in group j. The expected value of m^1_{ij} is given by

$$E(m^1_{ij}) = m_{ij} P(D_{ij} = 1) = m_{ij} P_i$$

where the last equality comes from the fact that the probability of response for a given value of $\mathbf{X}$ is the same across the J groups. Then, we can construct a moment condition for group j as

$$E\left(\sum_i \frac{m^1_{ij}}{P_i}\right) = \sum_i m_{ij} = m_j$$

where the right-hand side corresponds to the observed total mass of sampled households in group j. To complete the moment condition, we need to assume a functional form for P_i, which we assume to be a logistic function such that

$$P_i = P(D_{ij\epsilon} = 1 | \mathbf{X}_i, \boldsymbol{\theta}) = \frac{e^{\mathbf{X}_i' \boldsymbol{\theta}}}{1 + e^{\mathbf{X}_i' \boldsymbol{\theta}}}$$

where $\boldsymbol{\theta}$ is a K-vector of parameters. Having set up the population moment condition for group j, we can define its respective sample moment condition as

$$\psi_j(\boldsymbol{\theta}) = \sum_i \frac{m^1_{ij}}{P_i} - m_j$$

Finally, the estimator is constructed by stacking the J sample moment conditions into $\Psi(\boldsymbol{\theta})$ to get an estimator for $\boldsymbol{\theta}$ of the form

$$\widehat{\boldsymbol{\theta}} = \text{argmin}_{\boldsymbol{\theta}} \Psi(\boldsymbol{\theta})'\mathbf{W}^{-1}\Psi(\boldsymbol{\theta})$$

where $\mathbf{W}$ is a positive-definite weighting matrix. The $J \times J$ weighting matrix has off-diagonal elements equal to zero because of the assumption of independence of the response decisions of all households between the J groups. It is assumed that the variance of $\psi_j(\boldsymbol{\theta})$ for each group j is proportional to the mass of the sampled household population m_j, with a factor of proportionality σ^2 that can be ignored for the estimation.

The variance of the estimator $\widehat{\boldsymbol{\theta}}$ can be computed as

$$\widehat{\text{Var}}\left(\widehat{\boldsymbol{\theta}}\right) = \widehat{\sigma}^2 \left(\frac{\partial \Psi(\boldsymbol{\theta})'}{\partial \boldsymbol{\theta}} \mathbf{W}^{-1} \frac{\partial \Psi(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right)^{-1}$$

with

$$\frac{\partial \psi_j(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -\sum_i \frac{m_{ij}^1}{P_i^2}\frac{\partial P_i}{\partial \boldsymbol{\theta}} = -\sum_i \frac{m_{ij}^1 X_i}{e^{\mathbf{X}_i'\boldsymbol{\theta}}}$$

Alternatively, the variance can be computed using a bootstrap by randomly sampling J groups with replacement and applying the estimator to each sample. After a given number of repetitions, the bootstrapped variance is computed as the average squared deviation of the bootstrapped estimates from the original estimate. This method is computationally intensive because it needs to solve the minimization problem again for each bootstrapped sample. Nevertheless, it can be easily implemented by using the commands `bsample` and `simulate`, as shown in the empirical example.

3 The kmr command

3.1 Syntax

The syntax of the `kmr` command is

`kmr` [*varlist*] [*if*] [*in*] `, groups(`*varname*`)` `interview(`*varname*`)` `nonresponse(`*varname*`)` [`noconstant` `sweights(`*varname*`)` `generate(`*newvar*`)` `graph(`*varname*`)` `technique(`*string*`)` `delta(`*#*`)` `start(`*matrix*`)` `difficult` `maxiter(`*#*`)`]

varlist includes the determinants of the response rate.

3.2 Options

`groups(`*varname*`)` is required and specifies a categorical variable representing the group identifiers (these are state identifiers in Korinek, Mistiaen, and Ravallion [2006; 2007]). This variable can be a numeric or string variable.

`interview(`*varname*`)` is required and specifies the number of interviews obtained for each group.

`nonresponse(`*varname*`)` is required and specifies the number of nonresponses obtained for each group.

`nonconstant` suppresses the constant term.

`sweights(`*varname*`)` specifies the survey weights to be corrected and generates a new variable with the subscript `_c`. The new variable contains corrected survey weights that are generated by multiplying the weights provided by the user to the inverse of the estimated probability of response. Ideally, the user would use weights before any unit nonresponse correction only. Unfortunately, these are not generally available in the public use files of standard survey data. Hence, users should be aware that the corrected weights will likely overestimate the total population if the weights used in the `sweights()` option already have a form of unit nonresponse correction. To avoid this problem, users can easily construct and use a new set of uncorrected weights, as done in Korinek, Mistiaen, and Ravallion (2006, 2007) and as shown in the empirical example below.

`generate(`*newvar*`)` specifies the name of a new variable to be created containing the predicted probability of response. In addition, two other variables with the same name plus the subscripts `_upper` and `_lower` are created. They contain, respectively, the upper and lower bounds of a 95% confidence interval for the predicted value.

`graph(`*varname*`)` generates a line graph of the predicted probability of response against *varname*.

`technique(`*string*`)` specifies the algorithm to use in the minimization problem. The default is `technique(nr)` (modified Newton–Raphson). The alternatives are `dfp` (Davidon–Fletcher–Powell), `bfgs` (Broyden–Fletcher–Golfarb–Shanno), `bhhh` (Berndt–Hall–Hall–Hausman), and `nm` (Nelder–Mead[1]).

`delta(`*#*`)` specifies the value of the delta to be used for building the simplex required by `technique(nm)`. The default is `delta(0.1)`.

`start(`*matrix*`)` specifies the row vector with initial values for the parameters to start the algorithm. The default initial values are set to a vector of zeros.

`difficult` specifies that the criterion function is likely to be difficult to maximize because of nonconcave regions. The option `difficult` specifies that a different stepping algorithm be used in nonconcave regions (a mixture of steepest descent and Newton).

`maxiter(`*#*`)` sets the maximum number of iterations to be performed before the maximization is stopped. The default is `maxiter(100)`.

1. This is the only nongradient algorithm among the options. It is the algorithm used by MATLAB's `fminsearch` command, which Korinek, Mistiaen, and Ravallion (2007) applies in the code made available from the authors.

3.3 Stored results

`kmr` stores the following in `e()`:

Scalars	
`e(n)`	number of observations
`e(ngroups)`	number of groups
`e(aic)`	Akaike information criteria
`e(schwarz)`	Schwarz information criteria
`e(value)`	value of the function
`e(sigmavalue)`	value of sigma
Macros	
`e(cmd)`	`kmr`
`e(cmdline)`	command as typed
`e(title)`	title in estimation output
`e(algorithm)`	algorithm used
`e(properties)`	`b V`
Matrices	
`e(b)`	coefficient vector
`e(V)`	variance matrix of the estimates
Functions	
`e(sample)`	marks estimation sample

In addition, the command optionally generates four new variables: the predicted probability of compliance, the upper and lower values of its 95% confidence interval, and corrected survey weights.

3.4 Dependency of kmr

`kmr` depends on the Mata function `mm_collapse()`, which is part of the `moremata` package (Jann 2005). If it is not already installed, you can install it by typing `ssc install moremata`.

4 Empirical example

To illustrate the use of `kmr`, we use the 2018 CPS data downloaded from IPUMS (Flood et al. 2018) merged to the number of interviews and type A nonresponses (interviewer finds the household's address but obtains no interviews) obtained from the NBER CPS Supplements website.[2] We estimate the compliance function using the specification

$$P_i = \frac{e^{\theta_0+\theta_1 \log(y_i)}}{1+e^{\theta_0+\theta_1 \log(y_i)}}$$

where y_i corresponds to log of total household gross income per capita in current dollars.[3]

We begin by loading the dataset and looking at the state-level geographical variation in nonresponse rates in the United States. We can use the community-contributed command `maptile` (Stepner 2015) to show these rates in a map:

2. https://www.nber.org/data/current-population-survey-data.html.
3. Gross income is factor income plus all public and private monetary transfers.

```
. use cps2018
. preserve
. generate nonresponse = 100*typea/(typea+interview)
. collapse nonresponse, by(statefip)
. rename statefip statefips
. maptile nonresponse, geo(state) geoid(statefips) fcolor(Greys2)
> legdecimals(1) nquantiles(10)
. restore
```

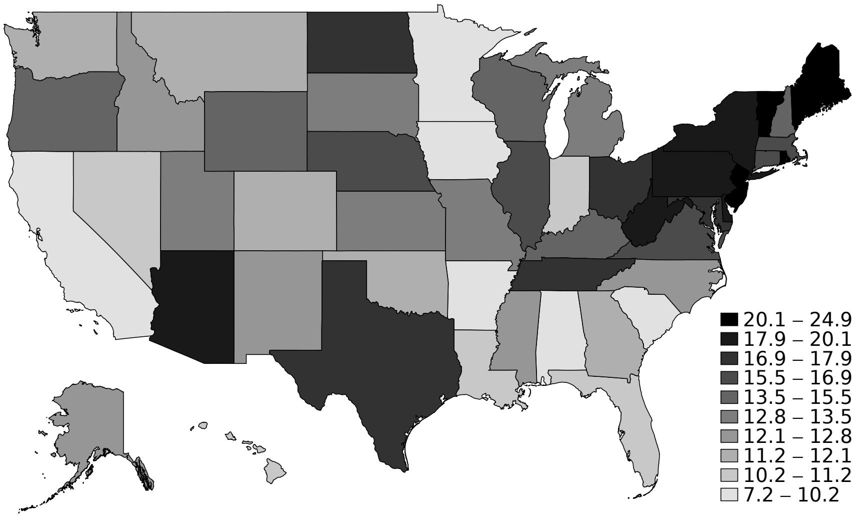

Figure 1. Nonresponse rates

Now, we can create the regressors and two sets of weights that have no correction for nonresponse. The first set corresponds to using the raw data (in other words, no weights or weights equal 1). The second set assumes equal weights within states ("grossed-up" weights by state), in which weights are constructed by dividing the population of each state (as derived by summing the official CPS weights) by the number of respondents:[4]

```
. generate ly = log(hhincome_pc)
. generate ly2 = ly^2
. by statefip: egen state_population = sum(asecwth)
. generate weights_1 = 1
. generate weights_2 = state_population/interview
```

4. Alternatively, we can construct uncorrected weights by dividing the sum of respondents and non-respondents by the number of respondents, but this would make little difference.

We can estimate the probability of response as a function of the log of total household gross income per capita, produce a line graph of it together with its 95% confidence interval, and generate a set of corrected weights called `weights_1_c`:[5]

```
. matrix init = -.9,12

. kmr ly, groups(statefip) interview(interview) nonresponse(typea) generate(P)
> sweights(weights_1) start(init)
Iteration 0:   f(p) =  636.90409  (not concave)
Iteration 1:   f(p) =  242.52283  (not concave)
Iteration 2:   f(p) =  155.56547  (not concave)
Iteration 3:   f(p) =  151.69864
Iteration 4:   f(p) =  151.41533
Iteration 5:   f(p) =  151.23525  (not concave)
Iteration 6:   f(p) =  151.23214  (not concave)
Iteration 7:   f(p) =  151.23182
Iteration 8:   f(p) =  151.23105
Iteration 9:   f(p) =  151.23105

Compliance function                               Number of obs      =    66899
                                                  AIC                =    59.44
Number of groups   =   51                         Schwarz            =  56.8224
------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          ly |  -.9866157   .2845149    -3.47   0.001    -1.544255   -.4289768
       _cons |   12.16841    3.11691     3.90   0.000      6.05938    18.27744
------------------------------------------------------------------------------

. ereturn list

scalars:
               e(value) =  151.2310513467116
          e(sigmavalue) =  4.8230144407221
                 e(aic) =  59.43614197528939
             e(schwarz) =  56.82243633640928
                   e(n) =  66899
             e(ngroups) =  51

macros:
             e(cmdline) : "kmr ly, groups(statefip) i(interview) n(typea) gen.."
               e(title) : "Compliance function estimate using group´s respons.."
                 e(cmd) : "kmr"
           e(technique) : "nr"
          e(properties) : "b V"

matrices:
                   e(b) :  1 x 2
                   e(V) :  2 x 2

functions:
              e(sample)

. sort ly
```

5. Note that we correct the unitary weights. To compute total population or total income, we should multiply the corrected weights by the ratio between the country population and the sampled households (interviews + nonresponses).

```
. line P P_upper P_lower ly, ytitle("Probability of response")
> xtitle("Log(income per capita)") lpattern("1" "-" "-")
> lcolor("black" "black" "black")
> legend(order(1 "Point estimate" 2 "95% CI")) scheme(s1color)
```

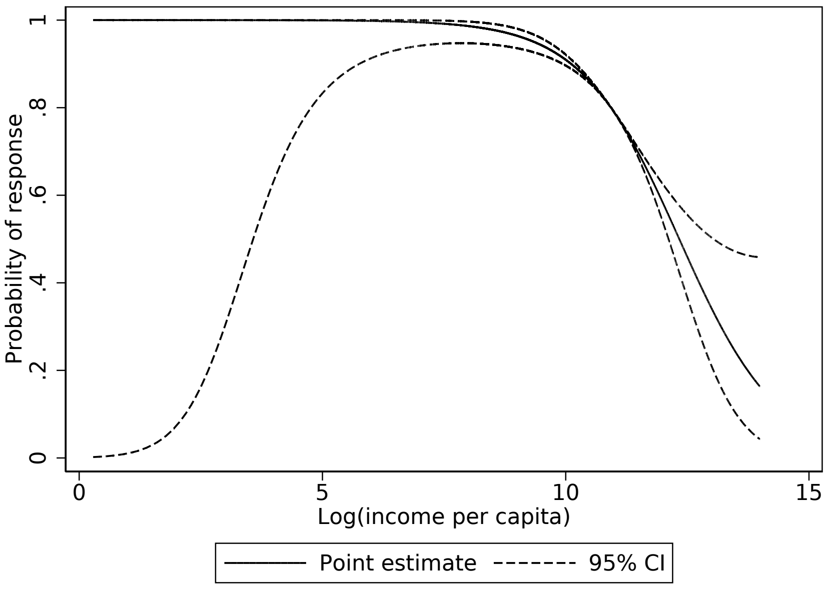

Figure 2. Compliance function

Now, we try a different specification that adds the squared log of income per capita as a second regressor. This will help to capture the fact that high nonresponse rates may occur in both tails of the income distribution, not just among rich households, as documented in Bollinger et al. (2019).

The inclusion of the log of income squared does capture some nonlinearity of compliance with respect to income. However, the estimates appear to be less precisely estimated (two of the coefficient's p-values are below 5% confidence level). Moreover, the Akaike criterion suggests that the linear specification is preferable.

```
. kmr ly ly2, groups(statefip) interview(interview) nonresponse(typea)
> generate(P2) difficult
Iteration 0:   f(p) =  39738.763
Iteration 1:   f(p) =   386.1303  (not concave)
Iteration 2:   f(p) =  181.12184
Iteration 3:   f(p) =   168.3632  (not concave)
Iteration 4:   f(p) =  164.02531
Iteration 5:   f(p) =  162.20711
Iteration 6:   f(p) =   160.9006  (not concave)
Iteration 7:   f(p) =   154.5554  (not concave)
Iteration 8:   f(p) =  153.88426
Iteration 9:   f(p) =  151.28687
Iteration 10:  f(p) =  150.52936
Iteration 11:  f(p) =  150.34972
Iteration 12:  f(p) =  150.34425
Iteration 13:  f(p) =  149.57514  (not concave)
Iteration 14:  f(p) =  149.41316  (not concave)
Iteration 15:  f(p) =  149.29277  (not concave)
Iteration 16:  f(p) =  149.29043
Iteration 17:  f(p) =  149.24785
Iteration 18:  f(p) =  149.24672
Iteration 19:  f(p) =  149.24614
Iteration 20:  f(p) =  149.24614

Compliance function                               Number of obs      =    66899
                                                  AIC                =    60.76
Number of groups   =   51                         Schwarz            =  58.0582
------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          ly |    1.65373      .9866     1.68   0.094    -.2799707     3.58743
         ly2 |  -.1191812   .0467899    -2.55   0.011    -.2108877   -.0274747
       _cons |  -2.320068   5.306235    -0.44   0.662     -12.7201    8.079962
------------------------------------------------------------------------------

. generate weights_1_c2 = weights_1/P2

. ereturn list

scalars:
               e(value) =  149.2461430060583
          e(sigmavalue) =  4.84797679101055
                 e(aic) =  60.76233511215851
             e(schwarz) =  58.05817197816284
                   e(n) =  66899
             e(ngroups) =  51

macros:
             e(cmdline) : "kmr ly ly2, groups(statefip) i(interview) n(typea).."
               e(title) : "Compliance function estimate using group´s respons.."
                 e(cmd) : "kmr"
           e(technique) : "nr"
          e(properties) : "b V"

matrices:
                   e(b) :  1 x 3
                   e(V) :  3 x 3

functions:
             e(sample)
```

```
. line P2 P2_upper P2_lower ly, ytitle("Probability of response")
>  xtitle("Log(income per capita)") lpattern("1" "-" "-")
>  lcolor("black" "black" "black")
>  legend(order(1 "Point estimate" 2 "95% CI")) scheme(s1color)
```

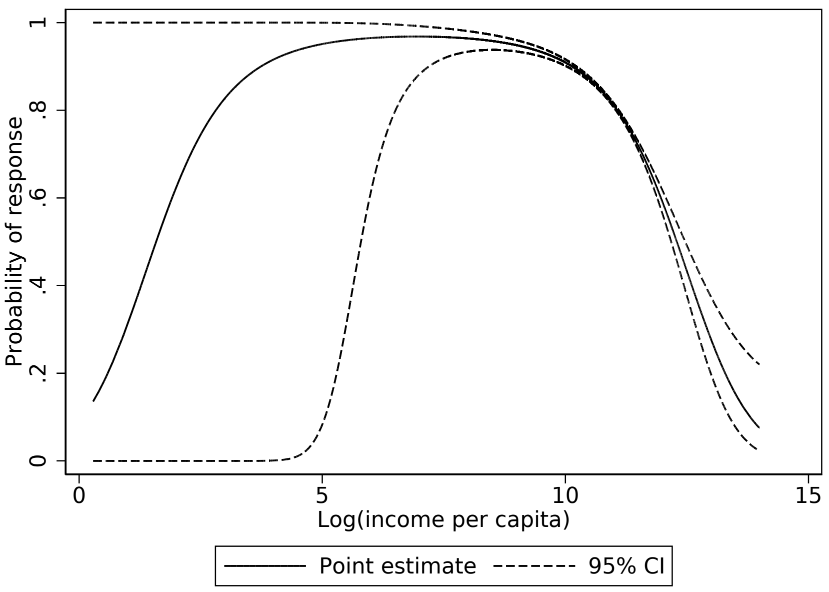

Figure 3. Compliance function quadratic on log(income)

As we mention at the end of section 2, we can also compute the standard errors using bootstrap. We do so by defining a small program called `kmrboot` that resamples states with replacement and estimates the compliance function for each new sample. This program is then called 1,000 times by the command `simulate`, which stores the estimated coefficients in each repetition.

```
. * Estimating the compliance function and storing the coefficients
. matrix init = -.9,12
. quietly kmr ly, groups(statefip) interview(interview) nonresponse(typea)
> start(init)
. matrix b = e(b)
. * Resampling clusters with replacement
. capture program drop kmrboot

. program define kmrboot, rclass
  1.  preserve
  2.   bsample, cluster(statefip) idcluster(newstatefip)
  3.   quietly kmr ly, groups(newstatefip) interview(interview)
> nonresponse(typea) start(b)
  4.   return scalar ly = e(b)[1,1]
  5.   return scalar _cons = e(b)[1,2]
  6.  restore
  7. end

. * Repeat the resampling 1,000 times
. preserve

. simulate ly = r(ly) _cons = r(_cons), reps(1000) seed(1) nodots: kmrboot

      command:  kmrboot
           ly:  r(ly)
        _cons:  r(_cons)

. * Analyze the results
. bstat, stat(b) n(1000)

Bootstrap results                               Number of obs     =      1,000
                                                Replications      =      1,000

------------------------------------------------------------------------------
             |   Observed   Bootstrap                         Normal-based
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          ly |  -.9866154   .3232632    -3.05   0.002      -1.6202   -.3530312
       _cons |   12.16841   3.450381     3.53   0.000     5.405785    18.93103
------------------------------------------------------------------------------

. * Store confidence intervals
. matrix lb = r(table)[5,1..2]

. matrix ub = r(table)[6,1..2]

. restore
```

From the results of the bootstrap exercise, we find a level of uncertainty surrounding the parameter estimates that is similar to the standard errors previously reported, although the confidence interval is slightly wider.

Finally, we can use the community-contributed command `fastgini` (Sajaia 2007) to compute the Gini coefficients using the following alternatives: CPS sample weights, raw data, "grossed-up" by state, and the `kmr` corrected weights according to the specification that uses log of income:

```
. matrix ginis = J(5,3,.)

. quietly fastgini hhincome_pc [w = asecwth]

. matrix ginis[1,1] = r(gini)

. quietly fastgini hhincome_pc [w = weights_1]

. matrix ginis[2,1] = r(gini)

. quietly fastgini hhincome_pc [w = weights_2]

. matrix ginis[3,1] = r(gini)

. quietly fastgini hhincome_pc [w = weights_1_c]

. matrix ginis[4,1] = r(gini)
. quietly fastgini hhincome_pc [w = 1/P_lower]
. matrix ginis[4,2] = r(gini)
. quietly fastgini hhincome_pc [w = 1/P_upper]
. matrix ginis[4,3] = r(gini)
. * Create probability of response according to bootstrap results
. generate p_boot_upper = invlogit(ly*ub[1,1]+ub[1,2])
. generate p_boot_lower = invlogit(ly*lb[1,1]+lb[1,2])
. matrix ginis[5,1] = ginis[4,1]
. quietly fastgini hhincome_pc [w = 1/p_boot_lower]
. matrix ginis[5,2] = r(gini)
. quietly fastgini hhincome_pc [w = 1/p_boot_upper]
. matrix ginis[5,3] = r(gini)
. matrix colnames ginis = "Point" "Lower" "Upper"
. matrix rownames ginis = "ASEC" "Raw" "Grossed-up" "kmr" "kmrboot"
. matlist ginis

             |     Point      Lower      Upper
-------------+---------------------------------
        ASEC |  .4652877          .          .
         Raw |  .4652385          .          .
  Grossed-up |  .4645361          .          .
         kmr |  .5051071   .5988177   .4799021
     kmrboot |  .5051071   .5779054   .4652385
```

In our exercise, we derive a range of values for our corrected Gini coefficient using the point estimates of the compliance function and its upper and lower bounds derived from the 95% confidence interval (row `kmr` and columns `Point`, `Lower`, and `Upper`, respectively). We then do the same for the range of values of the compliance function derived from the bootstrap exercise (row `kmrboot`). We prefer this "range of values" approach to the use of standard errors obtained via the `fastgini` command alone because the latter would leave out a substantial source of uncertainty originating from the adjustments of the original weights.[6]

We can summarize the results as follows. The use of sample weights does not significantly change the Gini coefficient compared with the use of unweighted data (comparing the two rows, `ASEC` and `Raw`); the proposed method of weight adjustment increases the estimated Gini coefficient by at least 8.6%, going from an uncorrected Gini of 0.465 to 0.505 (comparing the two rows, `ASEC` and `kmr`). The uncertainty associated with the estimation of the compliance function is nonnegligible, and it does not change significantly when the standard errors are computed using the bootstrap method (comparing the two columns `Lower` and `Upper`).

6. For instance, using the jackknife procedure without considering the uncertainty associated with the estimated weights, we get a much narrower confidence interval for the Gini coefficient (0.496; 0.514).

5 Conclusion

Unit nonresponse in household surveys could lead to biases in inequality and poverty measurement. The typical methods to correct survey weights for unit nonresponse assume ignorability within some arbitrary subgroup of the population, which recent empirical evidence suggests may not hold in the case of household survey data.

In this article, we presented the `kmr` command, which is designed to implement the econometric method introduced by Korinek, Mistiaen, and Ravallion (2007) to estimate a survey compliance function using group level nonresponse rates, allowing us to relax the ignorability assumption.

6 Acknowledgments

We thank Carolyn Fisher for comments on the draft and Anton Korinek for providing a MATLAB code with the dataset used in their article, which greatly facilitated this project. We are also grateful for the insightful and critical comments received by two anonymous referees, which pushed us to improve the structure of the article.

7 Programs and supplemental materials

To install a snapshot of the corresponding software files as they existed at the time of publication of this article, type

```
. net sj 21-1
. net install st0634      (to install program files, if available)
. net get st0634          (to install ancillary files, if available)
```

8 References

Bollinger, C. R., B. T. Hirsch, C. M. Hokayem, and J. P. Ziliak. 2019. Trouble in the Tails? What We Know about Earnings Nonresponse 30 Years after Lillard, Smith, and Welch. *Journal of Political Economy* 127: 2143–2185. https://doi.org/10.1086/701807.

Flood, S., M. King, R. Rodgers, S. Ruggles, and R. Warren. 2018. Integrated public use microdata series, current population survey: Version 6.0 [dataset]. Minneapolis, MN: IPUMS. https://doi.org/10.18128/D030.V6.0.

Hlasny, V. 2020. Nonresponse bias in inequality measurement: Cross-country analysis using Luxembourg Income Study surveys. *Social Science Quarterly* 101: 712–731. https://doi.org/10.1111/ssqu.12762.

Hlasny, V., and P. Verme. 2018a. Top incomes and the measurement of inequality in Egypt. *World Bank Economic Review* 32: 428–455. https://doi.org/10.1093/wber/lhw031.

———. 2018b. Top incomes and inequality measurement: A comparative analysis of correction methods using the EU SILC data. *Econometrics* 6: 30. https://doi.org/10.3390/econometrics6020030.

Jann, B. 2005. moremata: Stata module (Mata) to provide various functions. Statistical Software Components S455001, Department of Economics, Boston College. https://ideas.repec.org/c/boc/bocode/s455001.html.

Korinek, A., J. A. Mistiaen, and M. Ravallion. 2006. Survey nonresponse and the distribution of income. *Journal of Economic Inequality* 4: 33–55. https://doi.org/10.1007/s10888-005-1089-4.

———. 2007. An econometric method of correcting for unit nonresponse bias in surveys. *Journal of Econometrics* 136: 213–235. https://doi.org/10.1016/j.jeconom.2006.03.001.

Meyer, B. D., W. K. C. Mok, and J. X. Sullivan. 2015. Household surveys in crisis. *Journal of Economic Perspectives* 29: 199–226. https://doi.org/10.1257/jep.29.4.199.

Sajaia, Z. 2007. fastgini: Stata module to calculate Gini coefficient with jackknife standard errors. Statistical Software Components S456814, Department of Economics, Boston College. https://ideas.repec.org/c/boc/bocode/s456814.html.

Stepner, M. 2015. maptile: Stata module to map a variable. Statistical Software Components S457986, Department of Economics, Boston College. https://ideas.repec.org/c/boc/bocode/s457986.html.

About the authors

Ercio Muñoz is a PhD candidate in economics and a research associate at the Stone Center on Socio-Economic Inequality at the Graduate Center of City University of New York (CUNY).

Salvatore Morelli is an assistant professor in public economics at the University of Roma Tre, Law Department, Rome, Italy, and a core faculty at the Stone Center on Socio-Economic Inequality at the Graduate Center of City University of New York (CUNY).