kmr: A Command to Correct Survey Weights for Unit Nonresponse using Group's Response Rates

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

Salvatore Morelli
Stone Center on Socio-Economic Inequality and CUNY Graduate Center
New York, USA
smorelli@gc.cuny.edu

February 24, 2019

Abstract

This article describes **kmr**, a Stata command to estimate a micro compliance function using group's nonresponse rates (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: *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 et al. (2015)), which is problematic for measurement of inequality and/or poverty when response is not random and specially when it is related to the variable we are interested in.

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

Korinek et al. (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 paper presents `kmr`, a new command in Stata to implement this method, and illustrate its use with an empirical example using the CPS of the year 2018 and state-level response rates.

The paper is organized as follows. Section 2 describes the methodology. Section 3 describes the `kmr` command. Finally, in Section 4 we illustrate the use of the command with the empirical example and Section 5 concludes.

2 Methodology

As described in Korinek et al. (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 an K-vector X_ϵ (i.e. $P(D_\epsilon = 1) = f(X_\epsilon)$) and we observe the response rate for J groups together with the values of X for all the respondents.
For a given group $j \in J$, the mass of respondents with a given value i of X denoted by m^1_{ij} can be defined as:

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

where m_{ij} is the total (unobserved) number of households with income i in group j. The expected value of m^1_{ij} is given by:

$$E[m^1_{ij}] = m_{ij} P(D_{ij} = 1) = m_{ij} P_i \tag{2}$$

where the last equality comes from the fact that the probability of response for a given value of X is the same across groups. Then we can construct a moment condition for group j as follows:

$$E[\sum_i \frac{m_{ij}^1}{P_i}] = \sum_i m_{ij} \tag{3}$$

where the right hand side corresponds to the observed total mass of sampled households. 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:

$$P_i = P(D_{ij\epsilon} = 1 | X_i, \theta) = \frac{e^{X_i'\theta}}{1 + e^{X_i'\theta}} \tag{4}$$

where θ is a K-vector of parameters.
Having set up the population moment condition for region j, we can define its respective sample moment condition as:

$$\psi_j(\theta) = \sum_i \frac{m_{ij}^1}{P_i} - m_j \tag{5}$$

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

$$\hat{\theta} = argmin_\theta \Psi(\theta)' W^{-1} \Psi(\theta) \tag{6}$$

where 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 $\psi_j(\theta)$ 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 $\hat{\theta}$ can be computed as follows:

$$\hat{Var}(\hat{\theta}) = \hat{\sigma}^2 [\frac{\partial \Psi(\theta)'}{\partial \theta} W^{-1} \frac{\partial \Psi(\theta)}{\partial \theta}]^{-1} \tag{7}$$

with

$$\frac{\partial \psi_j(\theta)}{\partial \theta} = -\sum_i \frac{m_{ij}^1}{P_i^2} \frac{\partial P_i}{\partial \theta} = -\sum_i \frac{m_{ij}^1 X_i}{e^{X_i \theta}} \tag{8}$$

Alternatively, the variance can be computed using 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 bootstraped sample.

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(`*newvarname*`)` `graph(`*varname*`)` `technique(`*string*`)` `delta(`#`)`
`start(`#`)` `difficult` `maxiter(`#`)`]

where *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 et al. (2006, 2007)). This variable can be a numerical 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 non-responses obtained for each group.

`nonconstant` supress constant term.

`sweights(`*varname*`)` specifies the survey weights to be corrected and generates a new variable with the same name plus an extension "_c" containing corrected survey weights that are generated by multiplying the weights provided by the user times the inverse of the estimated probability of response. Note that ideally they should correspond to weights before any unit nonresponse correction, which are not typically available. If the user provides weights after unit nonresponse correction, then the corrected weights will have an additional correction that will over-estimate the total population.

`generate(`*newvarname*`)` 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 an extension "_upper" and "_lower" are created, they contain 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 "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(#)` value of delta to be used for building the simplex required by the technique “nm”. The default `delta` is set to 0.1.

`start(#)` 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 `maxiter` is set to 100.

3.3 Returned values

`kmr` saves the following in e():

Scalars	
`e(aic)`	Akaike information criteria
`e(schwarz)`	Schwarz information criteria
`e(value)`	value of the function
`e(sigmavalue)`	value of sigma
`e(n)`	number of observations
`e(ngroups)`	number of groups
Macros	
`e(cmdline)`	command line
`e(title)`	command title
`e(cmd)`	command name
`e(algorithm)`	algorithm used
`e(properties)`	properties
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.

4 Empirical Example

To illustrate the use of the command, we use data from the Current Population Survey of the year 2018 downloaded from IPUMS (Flood et al. 2018) merged to the number of interviews and type A nonresponses obtained from the NBER CPS Supplements

1. The only non-gradient algorithm among the options. It is the algorithm used by Matlab’s fminsearch command that Korinek et al. (2007) apply in the code made available from the authors.

website[2]. We estimate the compliance function using the following specification:

$$P_i = \frac{e^{\theta_0+\theta_1 log(y_i)}}{1+e^{\theta_0+\theta_1 log(y_i)}} \tag{9}$$

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

We begin by loading the data set and looking at the state-level geographical variation in nonresponse rates in the United States. We can use the user-written command `maptile` to show these rates in a map:

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

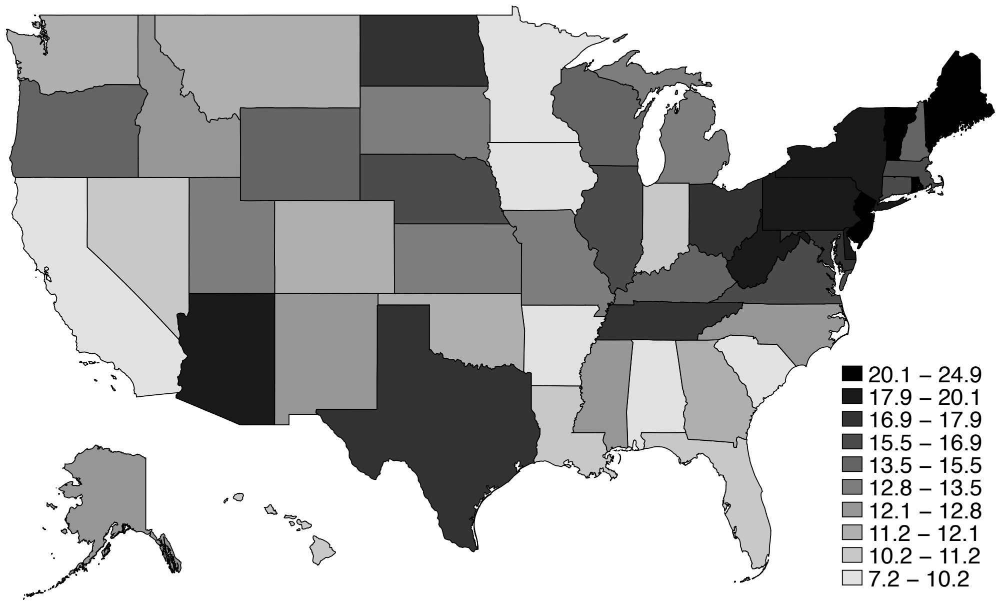

Figure 1: Nonresponse rates

Now we can create the regressors and two sets of uncorrected weights, one that corresponds to using the raw data (in other words, no weights) and one that assumes equal weights within states ("grossed-up" weights by state):

2. https://www.nber.org/data/current-population-survey-data.html

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

Now we can estimate the probability of response as a function of the log of total household gross income per capita, do a line graph of it together with its 95% confidence interval, and generate a set of corrected weights called "weights_1_c":[3]

```
. mat b = -.9,12
. kmr ly, groups(statefip) i(interview) n(typea) gen(P) sweights(weights_1) start(b)
Iteration 0:   f(p) =  636.90409  (not concave)
Iteration 1:   f(p) =  242.52283  (not concave)
Iteration 2:   f(p) =  155.56969  (not concave)
Iteration 3:   f(p) =  151.70047  (not concave)
Iteration 4:   f(p) =  151.42287
Iteration 5:   f(p) =  151.40201
Iteration 6:   f(p) =  151.23491
Iteration 7:   f(p) =  151.23105
Iteration 8:   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 |  -.9866137   .2845151    -3.47   0.001    -1.544253   -.4289743
       _cons |   12.16839   3.116912     3.90   0.000     6.059354    18.27743
------------------------------------------------------------------------------

. ereturn list
scalars:
              e(value) =  151.2310513468423
         e(sigmavalue) =  4.82301315461889
                e(aic) =  59.43614197533347
            e(schwarz) =  56.82243633645336
                  e(n) =  66899
            e(ngroups) =  51
macros:
            e(cmdline) : "kmr ly, groups(statefip) i(interview) n(typea) gen(P) "
              e(title) : "Compliance function estimate using group´s response rates"
                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
```

3. Note that we correct the unitary weights. If we want to compute total population or total income, then we should multiply these weights by the ratio of the country population over sampled households (interviews+nonresponses).

```
. line P P_upper P_lower ly, ytitle("Probability of response") ///
>  xtitle("Log(income per capita)") legend(off)
```

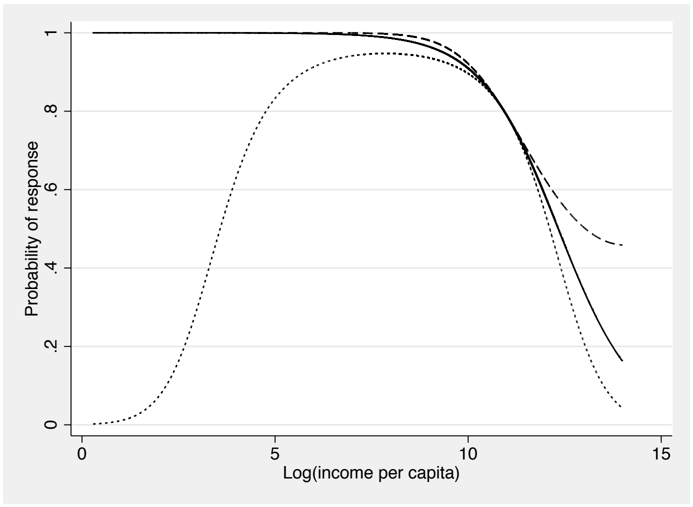

Figure 2: Compliance function

Now we try a different specification that adds the squared log of income per capita as a second regressor. After the estimation we compute a new set of corrected weights and plot the corresponding compliance function:

```
. kmr ly ly2, groups(statefip) i(interview) n(typea) gen(P2) difficult
Iteration 0:   f(p) =  39738.763
Iteration 1:   f(p) =   386.1304  (not concave)
Iteration 2:   f(p) =  205.18188  (not concave)
Iteration 3:   f(p) =   182.5436  (not concave)
Iteration 4:   f(p) =  181.01963  (not concave)
Iteration 5:   f(p) =  179.19322
Iteration 6:   f(p) =  176.98751  (not concave)
Iteration 7:   f(p) =   165.7847  (not concave)
Iteration 8:   f(p) =  164.02896  (not concave)
Iteration 9:   f(p) =  156.14176  (not concave)
Iteration 10:  f(p) =  154.56365
Iteration 11:  f(p) =  151.48705  (not concave)
Iteration 12:  f(p) =  150.33863  (not concave)
Iteration 13:  f(p) =  150.15689
Iteration 14:  f(p) =  150.09895  (not concave)
Iteration 15:  f(p) =  149.63349
Iteration 16:  f(p) =  149.25675
Iteration 17:  f(p) =  149.24854
Iteration 18:  f(p) =  149.24667
Iteration 19:  f(p) =  149.24663  (not concave)
Iteration 20:  f(p) =  149.24624
Iteration 21:  f(p) =  149.24614
Iteration 22:  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 |   1.653716   .9866511     1.68   0.094    -.2800843    3.587517
         ly2 |  -.1191805    .046792    -2.55   0.011    -.2108912   -.0274699
       _cons |  -2.320002   5.306531    -0.44   0.662    -12.72061    8.080607
------------------------------------------------------------------------------

. gen weights_1_c2 = weights_1/P2

. ereturn list

scalars:
              e(value) =  149.2461430043642
         e(sigmavalue) =  4.847973878218152
                e(aic) =  60.76233511157962
            e(schwarz) =  58.05817197758395
                  e(n) =  66899
            e(ngroups) =  51

macros:
            e(cmdline) : "kmr ly ly2, groups(statefip) i(interview) n(typea) gen(P2) "
              e(title) : "Compliance function estimate using group´s response rates"
                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)") legend(off)
```

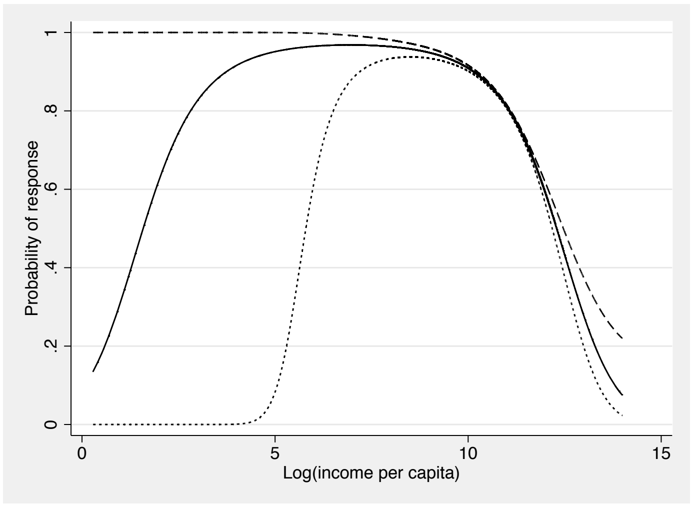

Figure 3: Compliance function quadratic on log(income)

Finally, we can use the user-written command `fastgini` to compute the Gini coefficients using the different alternatives: CPS sample weights, raw data, “grossed-up” by state, and the corrected weights under two alternative specifications:

```
. fastgini hhincome_pc [w = asecwth], jk
(sampling weights assumed)

Gini coefficient                                        Number of obs =   66899

──────────────────────────────────────────────────────────────────────────────
 hhincome_pc │       Gini   Std. Err.       t    P>|t|     [95% Conf. Interval]
─────────────┼────────────────────────────────────────────────────────────────
             │   .4652877    .0023628   196.92   0.000      .4606566    .4699187
──────────────────────────────────────────────────────────────────────────────

. fastgini hhincome_pc [w = weights_1], jk
(sampling weights assumed)

Gini coefficient                                        Number of obs =   66899

──────────────────────────────────────────────────────────────────────────────
 hhincome_pc │       Gini   Std. Err.       t    P>|t|     [95% Conf. Interval]
─────────────┼────────────────────────────────────────────────────────────────
             │   .4652385    .0018378   253.15   0.000      .4616365    .4688405
──────────────────────────────────────────────────────────────────────────────

. fastgini hhincome_pc [w = weights_2], jk
(sampling weights assumed)

Gini coefficient                                        Number of obs =   66899

──────────────────────────────────────────────────────────────────────────────
 hhincome_pc │       Gini   Std. Err.       t    P>|t|     [95% Conf. Interval]
─────────────┼────────────────────────────────────────────────────────────────
             │   .4645361    .0020602   225.48   0.000      .4604981    .4685741
──────────────────────────────────────────────────────────────────────────────

. fastgini hhincome_pc [w = weights_1_c], jk
(sampling weights assumed)

Gini coefficient                                        Number of obs =   66899

──────────────────────────────────────────────────────────────────────────────
 hhincome_pc │       Gini   Std. Err.       t    P>|t|     [95% Conf. Interval]
─────────────┼────────────────────────────────────────────────────────────────
             │   .5051069    .0044074   114.61   0.000      .4964686    .5137451
──────────────────────────────────────────────────────────────────────────────

. fastgini hhincome_pc [w = weights_1_c2], jk
(sampling weights assumed)

Gini coefficient                                        Number of obs =   66899

──────────────────────────────────────────────────────────────────────────────
 hhincome_pc │       Gini   Std. Err.       t    P>|t|     [95% Conf. Interval]
─────────────┼────────────────────────────────────────────────────────────────
             │   .5346141    .0071982    74.27   0.000       .520506    .5487223
──────────────────────────────────────────────────────────────────────────────
```

In our empirical example for the year 2018, we find the following results: the use of sample weights do not change significantly the Gini coefficient compared to unweighted data; the proposed method increases the Gini coefficient at least 8.6% (from an uncorrected Gini of .465 to .505); the correction using the second specification inflates the weights in both tails increasing the measure of inequality much more than the first specification (a Gini coefficient of .53 versus .50); the second specification shows a compliance function more in line with Bollinger et al. (2019) findings of lower response in the tails, however the information criteria that we obtain from our estimation favors the first specification.

5 Concluding remarks

Unit nonresponse in household surveys could lead to biases in inequality and poverty measurement. The typical methods applied by statistical offices to adress this problem assume ignorability within some arbitrary subgroup of the population, which may be at odds with recent empirical evidence.

In this article, we presented the command `kmr`, which implements Korinek et al. (2007) econometric method to estimate a survey compliance function using group level nonresponse rates and allow us to relax the ignorability assumption.

6 Acknowledgments

We thank Anton Korinek for providing a Matlab code with the data set used in their paper, which greatly facilitated this project.

7 References

Bollinger, C. R., B. Hirsch, C. Hokayem, and J. P. Ziliak. 2019. Trouble in the Tails? What We Know about Earnings Nonresponse Thirty Years after Lillard, Smith, and Welch. *Journal of Political Economy, forthcoming* .

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.* .

Korinek, A., J. A. Mistiaen, and M. Ravallion. 2006. Survey Nonresponse and the Distribution of Income. *Journal of Economic Inequality* 4(1): 33–55.

———. 2007. An Econometric Method of Correcting for Unit Nonresponse Bias in Surveys. *Journal of Econometrics* 136: 213–235.

Meyer, B. D., W. K. C. Mok, and J. X. Sullivan. 2015. Household Surveys in Crisis. *Journal of Economic Perspectives* 29(4): 199–226. http://pubs.aeaweb.org/doi/10.1257/jep.29.4.199.

About the authors

Ercio Munoz is student in the Ph.D. program in Economics and Research Associate at the Stone Center on Socio-Economic Inequality at the Graduate Center in the City University of New York.

Salvatore Morelli is Research Assistant Professor, ARC Distinguished Fellow, and Core Faculty at Stone Center on Socio-Economic Inequality at the Graduate Center in the City University of New York.