# Using Stata for the Rapid Consumption Approach

It is straight-forward to implement the rapid consumption approach in Stata. After explaining the required data structure, three examples are given to impute the deliberately absent data. The first example makes use of two user-written high-level Stata functions requiring only a few lines of code. The second example prepares and evaluates the imputation manually, while using user-written Stata classes perform the model selection and imputation. The third example does not require any user-written code and relies only on Stata functions and packages, allowing the user to modify any aspect of the imputation. We recommend the code of the first example for high-level users, while the second example gives more flexibility but requires more advanced Stata knowledge. The third example is provided as a reference for users to modify the model selection and imputation. All examples are using the same data with the same parameters. However, results can differ slightly due to the use of Monte Carlo methods.

The input file is KEN-example.dta, which includes a dataset based on Kenya’s household survey. Note that the dataset has been reduced to allow quicker run-time. For testing purposes, the full consumption aggregate is included.[[1]](#footnote-1)

The dataset includes the following variables:

|  |  |
| --- | --- |
| Variable name | Description |
| hhid | Unique household ID |
| strata | Strata |
| urban | Urban/rural status [1,0] |
| cluster | Cluster |
| weight | Sampling weight |
| hhmod | Module assigned to household |
| hhsize | Number of household members |
| ccons | Deflated per-capita full consumption for reference and testing |
| xfcons[X] | Deflated per-capita food consumption for module X |
| xnfcons[X] | Deflated per-capita non-food consumption for module X |
| mcat\_\* | Set of categorical explanatory variables for model |
| mcon\_\* | Set of continuous explanatory variables for model |

## Example 1

The first example makes use of two user-written Stata functions rcs\_estimate and rcs\_test, each defined in the respective ado file, which needs to be available in Stata’s ado-paths. The first function runs the model selection and the imputation:

### rcs\_estimate , hhid("hhid") hhsize("hhsize") strata("urban") weight("weight") cluster("cluster") hhmod("hhmod") xfcons("xfcons") xnfcons("xnfcons") mcon("mcon\_\*") mcat("mcat\_\*") rcscons("xcons") nmi(`nmi')

The parameters hhid, hhsize, strata, weight, cluster and hhmod are defining the variables from the dataset. The variable defined by hhmod needs to contain the module assigned to each household. The parameters xfcons and xnfcons are defining the variables containing the module food and non-food consumption. Using the stub xfcons as in the example, the dataset needs to include xfcons0 for the food core consumption, fcons1 for the first optional module and so on. The two parameters mcon and mcat include the variable lists for continuous and categorical co-variates to be considered in the structural model. Note that these parameters are not defining the stub of the variable names as for xfcons and xfncons but are lists of variables (explaining the \* in the definition). Finally, the function needs to know the name for the variable to be created to contain the imputed consumption data, captured in rcscons, as well as the number of imputations in nmi, which should usually be at least 20. The function will add the variable named in rcscons to the dataset, and transform the dataset to reflect multiple imputations.

Assuming the output variable in rcscons is called xcons, a simple mean(xcons) will only show missing values, as the mi prefix needs to be used:

### mi estimate: mean xcons [pweight=weight\*hhsize]

The value of the first imputation can also be directly accessed in the variable \_1\_xcons, the second in \_2\_xcons and so on. Instead of directly accessing the imputations, however, it is recommended to use the mi prefix commands instead. For example, the poverty rate can be calculated by first generating the poverty status for each household and imputation (assuming a fictious poverty line of 1.2), and then taking the mean over all households and imputations:

### mi passive: gen poor = xcons < 1.2

### mi estimate: mean poor [pweight=weight\*hhsize]

If the full consumption is also available, the function rcs\_test can be used to calculate the bias for each possible poverty rate:

### rcs\_test , hhsize("hhsize") weight("weight") rcscons("xcons") fullcons("ccons")

The first three parameters are as in the previous command, while fullcons is used to define the variable containing the full consumption. The result using the mean of the absolute deviation confirms the accuracy of the estimates based on the rapid consumption methodology:

### Mean estimation Number of obs = 99

### --------------------------------------------------------------

### | Mean Std. Err. [95% Conf. Interval]

### -------------+------------------------------------------------

### dfgt0 | .014876 .0008257 .0132374 .0165146

### dfgt1 | .0078283 .0005253 .0067859 .0088707

### --------------------------------------------------------------

## Example 2

### Model Selection and Imputation

In the second example, we are making use of Stata classes to implement the model selection and the imputation. First, we define the variable list of the continuous and categorical co-variates in corresponding local macros mcon and mcat. Note that the categorical co-variates need to use the i. operator:

unab mcon : mcon\_\*

fvunab mcat : i.mcat\_\*

Next, we instantiate an object from the class RCS\_estimator and run its method .prepare to prepare the dataset followed by the model selection method .select\_model:

capture classutil drop .re

.re = .RCS\_estimator.new

.re.prepare , hhid("hhid") weight("weight") hhmod("hhmod") cluster("cluster") xfcons("xfcons") xnfcons("xnfcons") nmi(`nmi')

.re.select\_model hhsize urban `mcon' `mcat', model("`model'") logmodel("`logmodel'") method("forward aicc")

The parameters for the preparation step are the same as in Example 1. The model selection is parameterized at a lower level allowing the user to provide the names of the co-variates directly. The two parameters model and logmodel allow the user to pass the definition of the structural model for the non-transformed and log-transformed consumption values. This can be helpful if the user wants to specify the model manually. In the example, those parameters are empty and, thus, the model selection algorithm is run. The method parameter defines how the best model is selected using the vselect Stata package. The parameters can be ‘forward bic’, ‘forward aic’, ‘forward aicc’, or ‘best’ to use the Furnival-Wilson leaps-and-bounds algorithm to explore all possible subsets.

Once the model is defined, the imputation can take place, followed by aggregating the imputed module consumption to total consumption:

.re.est\_mi\_2cel

gen xcons = .

mi register passive xcons

quiet: mi passive: replace xcons = 0

foreach v of varlist xfcons? xnfcons? {

quiet: mi passive: replace xcons = xcons + `v'

}

\*cleaning

mi register imputed xcons

mi update

Note that the aggregation needs to make use of the mi passive prefix.

### Testing

We had included the full consumption in the dataset even though this variable will usually not be available if the dataset is collected using the rapid approach. Using the full consumption aggregate, we can calculate FGT0 and FGT1 from the multiple imputations and compare to the correct reference values. We calculate those indicators for poverty lines resulting in FGT0 of 1 percent, 2 percent, … , 99 percent.

merge 1:1 hhid using "`sf'", assert(match) nogen

\* calculate FGT for all possible poverty lines

\_pctile ccons [pweight=weight\*hhsize], nq(100)

quiet forvalues i = 1/100 {

local pline`i' = r(r`i')

}

gen t\_fgt0 = .

gen t\_fgt1 = .

mi register passive t\_fgt0 t\_fgt1

quiet forvalues i = 1/100 {

\*for reference

gen r\_fgt0\_i`i' = ccons < `pline`i''

gen r\_fgt1\_i`i' = max(`pline`i'' - ccons,0) / `pline`i''

\*for estimates

mi passive: replace t\_fgt0 = xcons < `pline`i''

mi passive: replace t\_fgt1 = max(`pline`i'' - xcons,0) / `pline`i''

\*shortcut to avoid mi collapse

egen x\_fgt0\_i`i' = rowmean(\_\*\_t\_fgt0)

egen x\_fgt1\_i`i' = rowmean(\_\*\_t\_fgt1)

}

mi unset

keep r\_fgt\* x\_fgt\* weight hhsize

gen id = 1

collapse (mean) r\_fgt\* x\_fgt\* [pweight=weight\*hhsize], by(id)

reshape long r\_fgt0\_i x\_fgt0\_i r\_fgt1\_i x\_fgt1\_i, i(id) j(p)

label var p "Percentile Poverty Line"

ren \*\_i \*

drop id

order p r\_fgt0 x\_fgt0 r\_fgt1 x\_fgt1

\*calculate absolute differences

forvalues i = 0/1 {

label var r\_fgt`i' "FGT`i' Reference"

label var x\_fgt`i' "FGT`i' RCS"

gen zfgt`i' = r\_fgt`i'-x\_fgt`i'

gen dfgt`i' = abs(zfgt`i')

label var dfgt`i' "Absolute difference for FGT`i'"

}

mean dfgt\*

graph twoway (line zfgt0 p) (line zfgt1 p)

Note that the code is directly accessing the imputed variables for convenience. The result using the mean of the absolute deviation confirms the accuracy of the estimates based on the rapid consumption methodology:

Mean estimation Number of obs = 99

--------------------------------------------------------------

| Mean Std. Err. [95% Conf. Interval]

-------------+------------------------------------------------

dfgt0 | .0152737 .0008642 .0135587 .0169887

dfgt1 | .0080434 .0005435 .0069649 .0091218

--------------------------------------------------------------

## Example 3

### Data preparation

In a first step, we need to calculate quartiles for core consumption. The quartiles are included in the estimation model rather than the core consumption directly, as it would require making linear assumptions.

foreach v of var xfcons0 xnfcons0 {

xtile p`v' = `v' [pweight=weight] , n(4)

label var p`v' "Quartiles for `: var label `v''"

}

### Model Selection

The model is selected using a forward-step algorithm. We use the package vselect (ssc install vselect). As left-hand-side (lhs) variable, we use collected log consumption. This includes the core module as well as the assigned module. Note that this consumption aggregate is incomplete (and, thus, not comparable) as it depends on the assigned module to the household. In this step for model selection, however, we only determine the model, while the estimation of consumption is done in a later step.

\*calculate all collected consumption in log space

egen tcons = rowtotal(xfcons\* xnfcons\*)

gen ltcons = log(tcons)

replace ltcons = log(.01) if missing(ltcons)

\*prepare variable lists

unab mcon : mcon\_\*

fvunab mcat : i.mcat\_\*

\*estimate and select best model

xi: vselect ltcons hhsize urban `mcon' `mcat' [pweight=weight], forward aicc fix(i.hhmod)

local model = "`r(predlist)'"

\*output regression

reg ltcons `model' i.hhmod [pweight=weight]

\*add quartiles from core consumption to model

local model = "`model' i.pxfcons0 i.pxnfcons0"

drop tcons ltcons

tempfile fh

save "`fh'", replace

For model selection, we exclude the quartiles of food and non-food consumption, as the lhs includes core consumption to train the model. Therefore, we add the quartiles after selecting the model.

### Estimation Preparation

The estimation of missing consumption values is performed at the module-level. Thus, we reshape the dataset so that each optional module (including non-assigned modules) for each household is a record. In the case of 4 optional food and non-food modules, a household will, thus, be represented on 8 rows.

ren (xfcons0 xnfcons0) (fcore nfcore)

ren (xfcons? xnfcons?) (y1? y0?)

qui reshape long y0 y1, i(hhid) j(imod)

qui reshape long y, i(hhid imod) j(food)

\*remember 0 consumption

gen y\_0 = y==0 if !missing(y)

\*log and regularize for zero consumption

replace y = .01 if y<=0

replace y = log(y)

\*conditional step in estimation skipped if almost all hh have module consumption >0

bysort food imod: egen ny\_0 = mean(y\_0)

replace y\_0 = 0 if ny\_0 < 0.01

drop ny\_0

We use log-space for consumption, and use an indicator whether consumption for this module had been zero. In the case that almost all households assigned to a module have non-zero module consumption, it is difficult statistically to estimate the probability of zero module consumption for non-assigned households as the first step in the estimation procedure. If at least 99 percent of households have non-zero module consumption, we will later skip the first estimation step by assigning non-zero consumption to all non-assigned households.

### Two-Step Estimation

The two-step estimation first estimates whether the module has non-zero consumption using a logit regression. In a second step, the consumption of the module is estimates with an OLS regression. We use the framework of multiple imputations to avoid truncating the consumption distribution. This means that we will draw multiple times point estimates, which include the error component drawn from the error distribution.

mi set wide

mi register imputed y y\_0

mi register regular imod food

mi register regular hh\* cluster strata mcon\* \_I\* pxfcons0 pxnfcons0

mi impute monotone (logit, augment) y\_0 (reg, cond(if y\_0==0)) y = `model', add(`nmi') by(imod food)

\*transform into household-level dataset and out of log-space

keep hhid y y\_0 \_\* imod food fcore nfcore

mi xeq: replace y = exp(y)

\*reshape back to the hh-level

mi xeq: replace y = 0 if y\_0==1

drop y\_0

mi reshape wide y, i(hhid imod) j(food)

mi rename y0 xnfcons

mi rename y1 xfcons

mi reshape wide xfcons xnfcons, i(hhid) j(imod)

mi ren fcore xfcons0

mi ren nfcore xnfcons0

gen xcons = .

mi register passive xcons

quiet: mi passive: replace xcons = 0

foreach v of varlist xfcons? xnfcons? {

quiet: mi passive: replace xcons = xcons + `v'

}

\*cleaning

keep hhid xcons \_\*xcons \_mi\*

mi register imputed xcons

mi update

tempfile fh\_est

save "`fh\_est'", replace

As the estimation functions also estimates consumption for zero consumption modules, they are manually masked with zeros after the estimation. We also reshape the dataset back to the household level and calculate total estimated consumption for the household.

### Testing

We had included the full consumption in the dataset even though this variable will usually not be available if the dataset is collected using the rapid consumption methodology. The testing is the same as in Example 2. The result using the mean of the absolute deviation confirms the accuracy of the estimates based on the rapid consumption methodology:

Mean estimation Number of obs = 99

--------------------------------------------------------------

| Mean Std. Err. [95% Conf. Interval]

-------------+------------------------------------------------

dfgt0 | .0146524 .0008437 .012978 .0163267

dfgt1 | .0081166 .0005066 .0071113 .0091219

-------------------------------------------------------------

## Full Code

### Example 1

\*estimate optional consumption using a reduced dataset from Kenya KIHBS 2005/6

ma drop all

set more off

set maxiter 100

\*number of imputations

local nmi = 20

\*load data file

local sf = "KEN-Example.dta"

use "`sf'", clear

\*run rapid consumption and then (as we have full consumption) test it

rcs\_estimate , hhid("hhid") hhsize("hhsize") strata("urban") weight("weight") hhmod("hhmod") cluster("cluster") xfcons("xfcons") xnfcons("xnfcons") mcon("mcon\_\*") mcat("mcat\_\*") rcscons("xcons") nmi(`nmi')

mi estimate: mean xcons [pweight=weight\*hhsize]

\*calculate poverty rate, using fictious poverty line

mi passive: gen poor = xcons < 1.2

mi estimate: mean poor [pweight=weight\*hhsize]

\*test imputations, using full consumption

rcs\_test , hhsize("hhsize") weight("weight") rcscons("xcons") fullcons("ccons")

### Example 2

### \*estimate optional consumption using a reduced dataset from Kenya KIHBS 2005/6

### ma drop all

### set more off

### set maxiter 100

### \*load data file

### local sf = "KEN-Example.dta"

### \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### \*find best model in log space of all collected consumption \*

### \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### \*number of imputations (should 100 for final results)

### local nmi = 10

### use "`sf'", clear

### \*prepare variable lists

### unab mcon : mcon\_\*

### fvunab mcat : i.mcat\_\*

### \*create class for model selection and estimation

### capture classutil drop .re

### .re = .RCS\_estimator.new

### .re.prepare , hhid("hhid") weight("weight") hhmod("hhmod") cluster("cluster") xfcons("xfcons") xnfcons("xnfcons") nmi(`nmi')

### .re.select\_model hhsize urban `mcon' `mcat', model("`model'") logmodel("`logmodel'") method("forward aicc")

### \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### \*run estimation \*

### \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### .re.est\_mi\_2cel

### gen xcons = .

### mi register passive xcons

### quiet: mi passive: replace xcons = 0

### foreach v of varlist xfcons? xnfcons? {

### quiet: mi passive: replace xcons = xcons + `v'

### }

### \*cleaning

### mi register imputed xcons

### mi update

### tempfile fh\_est

### save "`fh\_est'", replace

### \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### \* test results by comparing to full consumption \*

### \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### use "`fh\_est'", clear

### merge 1:1 hhid using "`sf'", assert(match) nogen

### \* calculate FGT for all possible poverty lines

### \_pctile ccons [pweight=weight\*hhsize], nq(100)

### quiet forvalues i = 1/100 {

### local pline`i' = r(r`i')

### }

### gen t\_fgt0 = .

### gen t\_fgt1 = .

### mi register passive t\_fgt0 t\_fgt1

### quiet forvalues i = 1/100 {

### \*for reference

### gen r\_fgt0\_i`i' = ccons < `pline`i''

### gen r\_fgt1\_i`i' = max(`pline`i'' - ccons,0) / `pline`i''

### \*for estimates

### mi passive: replace t\_fgt0 = xcons < `pline`i''

### mi passive: replace t\_fgt1 = max(`pline`i'' - xcons,0) / `pline`i''

### \*shortcut to avoid mi collapse

### egen x\_fgt0\_i`i' = rowmean(\_\*\_t\_fgt0)

### egen x\_fgt1\_i`i' = rowmean(\_\*\_t\_fgt1)

### }

### mi unset

### keep r\_fgt\* x\_fgt\* weight hhsize

### gen id = 1

### collapse (mean) r\_fgt\* x\_fgt\* [pweight=weight\*hhsize], by(id)

### reshape long r\_fgt0\_i x\_fgt0\_i r\_fgt1\_i x\_fgt1\_i, i(id) j(p)

### label var p "Percentile Poverty Line"

### ren \*\_i \*

### drop id

### order p r\_fgt0 x\_fgt0 r\_fgt1 x\_fgt1

### \*calculate absolute differences

### forvalues i = 0/1 {

### label var r\_fgt`i' "FGT`i' Reference"

### label var x\_fgt`i' "FGT`i' RCS"

### gen zfgt`i' = r\_fgt`i'-x\_fgt`i'

### gen dfgt`i' = abs(zfgt`i')

### label var dfgt`i' "Absolute difference for FGT`i'"

### }

### mean dfgt\*

### graph twoway (line zfgt0 p) (line zfgt1 p)

### Example 3

\*estimate optional consumption using a reduced dataset from Kenya KIHBS 2005/6

clear all

ma drop all

set more off

\*number of imputations (should 100 for final results)

local nmi = 20

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*load dataset and prepare per-capita variables, quartiles and transform to logs \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*load data file

local sf = "KEN-Example.dta"

use "`sf'", clear

\*create quartiles for consumption

foreach v of var xfcons0 xnfcons0 {

xtile p`v' = `v' [pweight=weight] , n(4)

label var p`v' "Quartiles for `: var label `v''"

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*find best model in log space of all collected consumption \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*calculate all collected consumption in log space

egen tcons = rowtotal(xfcons\* xnfcons\*)

gen ltcons = log(tcons)

replace ltcons = log(.01) if missing(ltcons)

\*prepare variable lists

unab mcon : mcon\_\*

fvunab mcat : i.mcat\_\*

\*estimate and select best model

xi: vselect ltcons hhsize urban `mcon' `mcat' [pweight=weight], forward aicc fix(i.hhmod)

local model = "`r(predlist)'"

\*output regression

reg ltcons `model' i.hhmod [pweight=weight]

\*add quartiles from core consumption to model

local model = "`model' i.pxfcons0 i.pxnfcons0"

drop tcons ltcons

tempfile fh

save "`fh'", replace

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* prepare dataset for estimation with two-step log estimation \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

use "`fh'", clear

ren (xfcons0 xnfcons0) (fcore nfcore)

ren (xfcons? xnfcons?) (y1? y0?)

qui reshape long y0 y1, i(hhid) j(imod)

qui reshape long y, i(hhid imod) j(food)

\*remember 0 consumption

gen y\_0 = y==0 if !missing(y)

\*log and regularize for zero consumption

replace y = .01 if y<=0

replace y = log(y)

\*conditional step in estimation skipped if almost all hh have module consumption >0

bysort food imod: egen ny\_0 = mean(y\_0)

replace y\_0 = 0 if ny\_0 < 0.01

drop ny\_0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* run estimation with two-step log estimation with multiple imputations \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

mi set wide

mi register imputed y y\_0

mi register regular imod food

mi register regular hh\* cluster strata mcon\* \_I\* pxfcons0 pxnfcons0

mi impute monotone (logit, augment) y\_0 (reg, cond(if y\_0==0)) y = `model', add(`nmi') by(imod food)

\*transform into household-level dataset and out of log-space

keep hhid y y\_0 \_\* imod food fcore nfcore

mi xeq: replace y = exp(y)

\*reshape back to the hh-level

mi xeq: replace y = 0 if y\_0==1

drop y\_0

mi reshape wide y, i(hhid imod) j(food)

mi rename y0 xnfcons

mi rename y1 xfcons

mi reshape wide xfcons xnfcons, i(hhid) j(imod)

mi ren fcore xfcons0

mi ren nfcore xnfcons0

gen xcons = .

mi register passive xcons

quiet: mi passive: replace xcons = 0

foreach v of varlist xfcons? xnfcons? {

quiet: mi passive: replace xcons = xcons + `v'

}

\*cleaning

keep hhid xcons \_\*xcons \_mi\*

mi register imputed xcons

mi update

tempfile fh\_est

save "`fh\_est'", replace

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* test results by comparing to full consumption \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

use "`fh\_est'", clear

merge 1:1 hhid using "`fh'", assert(match) nogen

\* calculate FGT for all possible poverty lines

\_pctile ccons [pweight=weight\*hhsize], nq(100)

quiet forvalues i = 1/100 {

local pline`i' = r(r`i')

}

gen t\_fgt0 = .

gen t\_fgt1 = .

mi register passive t\_fgt0 t\_fgt1

quiet forvalues i = 1/100 {

\*for reference

gen r\_fgt0\_i`i' = ccons < `pline`i''

gen r\_fgt1\_i`i' = max(`pline`i'' - ccons,0) / `pline`i''

\*for estimates

mi passive: replace t\_fgt0 = xcons < `pline`i''

mi passive: replace t\_fgt1 = max(`pline`i'' - xcons,0) / `pline`i''

\*shortcut to avoid mi collapse

egen x\_fgt0\_i`i' = rowmean(\_\*\_t\_fgt0)

egen x\_fgt1\_i`i' = rowmean(\_\*\_t\_fgt1)

}

mi unset

keep r\_fgt\* x\_fgt\* weight hhsize

gen id = 1

collapse (mean) r\_fgt\* x\_fgt\* [pweight=weight\*hhsize], by(id)

reshape long r\_fgt0\_i x\_fgt0\_i r\_fgt1\_i x\_fgt1\_i, i(id) j(p)

label var p "Percentile Poverty Line"

ren \*\_i \*

drop id

order p r\_fgt0 x\_fgt0 r\_fgt1 x\_fgt1

\*calculate absolute differences

forvalues i = 0/1 {

label var r\_fgt`i' "FGT`i' Reference"

label var x\_fgt`i' "FGT`i' RCS"

gen dfgt`i' = abs(r\_fgt`i'-x\_fgt`i')

label var dfgt`i' "Absolute difference for FGT`i'"

}

mean dfgt\*

1. For the purpose of illustration, the dataset was reduced by randomly selecting every second cluster. In addition, consumption items with less than 0.1 percent consumption share were also dropped, given the extensive original list of items in the survey, which slows down calculations. [↑](#footnote-ref-1)