EUROPEAN CENTRAL BANK

EUROSYSTEM

HFCS User Database Documentation

User guide

July 2023

Contents

1 Technical information on the HFCS User Database (UDB) 4

1.1 Database naming convention 4

1.2 The data structure in which HFCS data are distributed 4

1.3 Available data formats 5

1.4 ID variables 6

1.5 Variable naming conventions 6

1.6 Flag variables 8

1.7 Special codes 10

2 Using the HFCS dataset: Multiple imputation, survey weights and replicate weights 11

2.1 Multiple imputation 11

2.2 Survey weights 13

2.3 Replicate weights and variance estimation 13

3 Using the HFCS with Stata 16

3.1 Importing the data to allow estimation in the mi framework 16

3.2 Setting up multiple imputation 18

3.3 Setting up new survey variables 20

3.4 Using standard estimation procedures 20

3.5 Including additional estimation procedures 21

3.6 Useful commands 22

3.7 Additional commands and instructions 23

4 Using HFCS with R 24

4.1 Importing the data 24

4.2 Memory Considerations 25

4.3 Data Wrangling 25

4.4 Multiple Imputation and estimation 27

5 The panel component in the HFCS 28

6 References 32

The purpose of this document is to provide a general brief overview and a user-oriented technical description to accompany the dissemination of the Eurosystem Household Finance and Consumption Survey (HFCS) micro dataset. Information is provided with a view to facilitating the use of the dataset. Technical information is also provided on the structure in which the data is distributed to users, the available data formats, the conventions used for naming the HFCS variables and the contents of flag variables. The document also provides practical guidance on how to start using the HFCS multiply imputed data set with Stata and R.

This document supplements the methodological reports of the HFCS which are published for each survey wave. These publications are available on the HFCN website and inform about the main features of the surveys and describe their contents.

1 Technical information on the HFCS User Database (UDB)

This chapter describes the data structure in the HFCS User Database, including variable naming conventions and flag variables.

1.1 Database naming convention

The naming of the HFCS User Database respects the following conventions.

All of the database names begin by “HFCS_UDB”. **The first and second numeral suffixes** correspond respectively to the wave the database relates to and the version number. **The third suffix** is the software format of the database.

For example, the database named HFCS_UDB_2_4_SAS includes the updated version of the wave 2014 (second wave) data in SAS, while HFCS_UDB_3_3_SAS includes the wave 2017 (third wave) data, and HFCS_UDB_4_0_SAS includes the wave 2021 (fourth wave) data.

1.2 The data structure in which HFCS data are distributed

The HFCS anonymised User Database is distributed **in a household-person relational data structure** for all waves.

The HFCS survey microdata are **multiply imputed,** meaning that five implicates are provided for each observation. Each of the datasets mentioned below, except the ones containing replicate weights, is provided in five separate files, one for each multiple imputation implicate (see chapters 2 - 4 for more information on working with multiply imputed datasets).

Box 1 List of files in the HFCS microdata

H1 – H5	Core household files
P1 – P5	Core personal files
HN1 – HN5	Non-core household files
PN1 – PN5	Non-core personal files
D1 – D5	Derived variables files
W	Replicate weights file

(1) Core variables

Personal and household-level core variables are presented in separate data files:

- The main data file with household level variables, denoted as the H-file.
- The personal questionnaire file, containing information collected at the personal level, is denoted as the P-file. Variables only collected for persons aged 16 and over are set to missing in records for persons younger than 16.

The files are linkable via the household identification variable.

(2) Non-core variables

Non-core variables are made available with separate non-core files named HN and PN.

(3) Derived household-level variables

A separate file is provided with standard aggregations at household-level (so-called derived variables), such as total net wealth, total assets, total liabilities, total gross income, etc. This file is named D.

(4) Replicate weights

A set of replicate weights is provided for variance estimation purposes (see more details in the replicate weights and variance estimation section). The replicate weight file has only one implicate in the UDB (IM0100=1).

1.3 Available data formats

The HFCS User Database is available in three different software formats:

- as SAS data files,
- as Stata data files,
- as ASCII files (text files, with comma as data separator, dot as decimal separator and double quotes as text separator, first text row as the header line with names of the variables).

All three data formats are again information equivalent, containing exactly the same information. The SAS and Stata files are accompanied by program code files containing labels for variables and for answer values of categorical variables. Files *labels_.csv* and *valuelabels_.csv* contain labels for processing with the ASCII files or for other potential user needs. Labels are identical to the variable names and answer category descriptions in the

HFCS variables catalogues. For Stata, which is case sensitive, it is recommended changing all the variable names to lower case.

```
rename *, lower
```

1.4 ID variables

The ID identification variable provides a unique identifier of each household and personal record. In addition, the HID household identifier is also provided in the personal files to provide a quick link between household and personal records in the “household-person relational” data format.

An option for linking is also to use directly the variables underlying the household and personal identification variables. These are country code (SA0100), household identification number (SA0010), implicate number (IM0100) and personal identification number (RA0010). For panel households, there are two additional ID variables as explained in chapter 5 of this document.

1.5 Variable naming conventions

Each core HFCS variable is assigned a unique variable name. The naming of variables follows the conventions outlined below.

The first character identifies the reference unit for which the variable was collected (household: H, all persons: R, persons 16 plus: P). Basic identifiers’ names start with an S (household identification number SA0010, country SA0100) and the multiple imputation implicate number with an I (IM0100). These variables retain their name in all files.

The second character identifies the questionnaire thematic section the variable belongs to:

A – Demographics

B – Real assets and their financing

C – Other liabilities / Credit constraints

D – Private businesses / Financial assets

E – Employment

F – Pensions and insurance policies

G – Income

H – Intergenerational transfers / gifts

I – Consumption

In cases where this does not apply, e.g. ID variables, weight variables or implicate IDs, the second character is purely conventional[1].

The third, fourth and fifth characters indicate the variable number. Variables are numbered in sequential order of appearance of their corresponding questions in the Euro area blueprint HFCS questionnaire.

The sixth character is for loops – situations when the question is asked more than once (for example the question on monthly payments asked for each non-collateralised loan in variables HC1001, HC1002, HC1003 …) The sixth character takes the value 1-9 to identify each loop, and is 0 if there is no loop and the question is asked only once.

The seventh character is optional and it is used either i) to identify items in a set for variables where several responses are grouped together into a set sharing a common base name (typically multiple choice questions allowing more than one possible answer) or ii) in the 'double loop' for mortgages on other properties to indicate one of the three mortgages with highest principal outstanding collateralised by one of the three most important real estate properties. In this loop the sixth character indicates the real estate property.

Box 2
Examples of variable names

Example of a standard variable: RA0300 [age], a personal register variable collected for all household members (R). Meaning of the variable name: section A (general and demographic information), number 030, 0 - no loop.

Example of a variable from a set of items: HI0400g [purpose of saving – old age provision (one of the items in the multiple choice question)]. Meaning of the variable name: household questionnaire variable (H), section I (consumption), number 040, 0 - no loop, item g in a set of responses.

Example of a variable from a set of items, with loops: HB1201a [loop for mortgages on household main residence, mortgage #1, first of possible multiple purposes why the mortgage was taken]. Meaning of the variable name: household questionnaire variable (H), section B (real assets), number 120, first loop (1), first item in a set of multiple responses for the first, second, … purpose of the loan (a).

[1] Another exception are pension variables in the 2017 and 2021 waves that start with a prefix 'PFA' and the fourth, fifth and sixth character indicate the variable number.

Similar naming conventions are also applied to the non-core variables, with the addition **of the non-core indication letter N on the second position** – example: HNB0810 Household main residence – year of construction, as non-core variable in the Real assets and their financing thematic section, supplementing in this case the core information on the household main residence. The non-core thematic section on payment habits has been assigned the letter J – example variable: HNJ2400 Number of credit card payments per month.

1.6 Flag variables

Each data variable is accompanied by its flag variable. Flag variables contain additional meta-information on the status of each individual observation (i.e. whether it was collected, estimated, imputed, reasons for missingness, etc.). Flag variables are named with the f prefix and each one is located in the data files next to the data variable to which it corresponds. Some flag values and their definitions have been modified between various survey waves (see Box 3).

Box 3
List of codes for flag variables in wave 4

Possible data values	Flags in wave 4 (2021)		Flags in wave 3 (2017)	
missing value	0	Not applicable (i.e. skipped due to routing)	0	Not applicable (i.e. skipped due to routing)
valid response	1	Recorded as collected, complete observation	1 11 13	Recorded as collected, complete observation Recoded from collected verbatim text response Value transformed from collected answers to the national questionnaire (output harmonisation)
valid response /missing value	71	Value taken from (statistical, fiscal or other) register	71	Value taken from (statistical, fiscal or other) register
-1, -2, missing value	1050	Not imputed, question was asked but respondent gave no answer, an incorrect/unreliable answer, or value not collected due to missing answer of a previous question, CAPI/interviewer error or to editing of parent variable	1050 1051 1052 1054 1057	Not imputed, originally Don't know Not imputed, originally No answer Not imputed, originally not collected due to missing answer to a previous question (i.e. originally missing due to higher order missing data) Not imputed, collected value deleted (considered incorrect or unreliable) or value not collected due to a CAPI or interviewer error Not imputed, value not collected due to a CAPI or interviewer error or to editing of parent variable
Valid response (IM0100 in 1 and 5)/missing value (IM0100 in 2,3,4)	1053	Not imputed, originally collected from a range or from brackets	1053	Not imputed, originally collected from a range or from brackets
missing value	2050	All observations (or a subset of observations) for the variable set as missing	2051	Missing, set as missing because data was not collected
missing value	2099	Missing, set as missing for anonymisation purposes (only available in the ESCB dataset)	2050	Missing, set as missing for anonymisation purposes (only available in the ESCB dataset)
valid response/missing value	3050	Edited	3050	Edited, set to modified value as considered incorrect or unreliable
valid response[2]	4050	Imputed, question was asked but respondent gave no answer, an incorrect/unreliable answer, or value not collected due to CAPI/interviewer error or to editing of parent variable	4050 4051 4054 4057	Imputed, originally Don't know Imputed, originally No answer Imputed, collected value deleted (considered incorrect or unreliable) Imputed, value not collected due to a CAPI or interviewer error or to editing of parent variable

[2] Exceptionally, flag 4050 can be used with missing values if values were imputed to missing.

valid response/missing value	4052	Imputed, question was not asked (higher order item nonresponse)	4052	Imputed, originally not collected due to missing answer to a previous question
valid response	4053	Imputed, originally collected from a range or from brackets	4053	Imputed from range or from brackets
valid response /missing value	5050	Estimated, originally not collected	12 5050	Answer could be determined without Estimated, originally not collected (question was not asked to the respondent)

Box 4
List of codes for flag variables in previous waves

Possible data values	Flag		Waves applied in
missing value	0	Not applicable (i.e. skipped due to routing)	1,2,3
valid response	1	Recorded as collected, complete observation	1,2,3
valid response	11	Collected from verbatim text response	2,3
valid response /missing value	12	Answer could be determined without asking the question (question was not asked to the respondent)	2,3
valid response	13	Value transformed from collected answers to national questionnaire (output harmonisation)	2,3
valid response /missing value	71	Value taken from (statistical, fiscal or other) register	2,3
-1	1050	Not imputed, originally Don't know	1,2,3
-2	1051	Not imputed, originally No answer	1,2,3
missing value	1052	Not imputed, originally not collected due to missing answer to a previous question (i.e. originally missing due to higher order missing data)	1,2,3
missing value	1053	Not imputed, originally collected from a range or from brackets	1,2,3
missing value	1054	Not imputed, collected value deleted (considered incorrect or unreliable) or value not collected due to a CAPI or interviewer error	1
missing value	1054	Not imputed, collected value deleted (considered incorrect or unreliable)	2,3
missing value	1055	Not imputed, value not collected due to a CAPI or interviewer error	2
missing value	1056	Not imputed, value missing due to editing of parent variable	2
missing value	1057	Not imputed, value not collected due to a CAPI or interviewer error or to editing of parent variable	3
missing value	2050	Missing, set as missing for anonymisation purposes (only available in the ESCB dataset)	1,2,3
missing value	2051	Missing, set as missing because data was not collected	1,2,3
valid response/missing value	3050	Edited, set to modified value as considered incorrect or unreliable	1,2,3
valid response	4050	Imputed, originally - Don't know	1,2,3
valid response	4051	Imputed, originally - No answer	1,2,3
valid response/missing value	4052	Imputed, originally not collected due to missing answer to a previous question (i.e., originally missing due to higher order missing data)	1,2,3
valid response	4053	Imputed, originally collected from a range or from brackets	1,2,3
valid response	4054	Imputed, collected value deleted (considered incorrect or unreliable) or value not collected due to a CAPI or interviewer error	1
valid response	4054	Imputed, collected value deleted (considered incorrect or unreliable)	2,3
valid response	4055	Imputed, value not collected due to a CAPI or interviewer error	2
valid response /missing value	4056	Imputed, value missing due to editing of parent variable	2
valid response /missing value	4057	Imputed, value not collected due to a CAPI or interviewer error or to editing of parent variable	3
valid response /missing value	5050	Estimated, originally not collected	1,2,3

1.7 Special codes

Various special codes may be used in the HFCS. In the user dataset, these codes may either be recoded to missing or provided as originally filled.

For most variables, special codes -1 and -2 can be used if the respondent does not provide an answer. Their meaning is as follows:

- -1: Don’t know
- -2: No answer

These codes are recoded to missing in the final dataset for all the numeric values. They are provided for string variables.

Other special codes that can be used for some specific variables are: -3; -4; -7; -8; -9; 99. Their meaning varies slightly across variables and can be found in the list of core variables or the HFCS questionnaire, in the HFCS website. These special codes are disseminated as originally provided.

2 Using the HFCS dataset: Multiple imputation, survey weights and replicate weights

2.1 Multiple imputation

HFCS UDB data are provided as imputed in relation to the household assets, liabilities and income variables. The HFCS provides multiply imputed values to cover for item non-response via stochastic imputation, meaning estimating missing observations conditional upon observed variables that can plausibly explain missingness. For each missing value, five imputed values are provided (thus giving rise to the same number of complete data sets). The reason why missing values are multiply imputed is that if the procedure were run just once (single imputation) without adding the appropriate random term, it would not take into account imputation uncertainty (the resulting variance could be underestimated). This would be a particular problem in cases of significant item non-response.

Imputed values are provided for observations that:

- have replies ‘No answer’ or ‘Do not know’
- are not collected due to a missing answer to a previous question
- are originally collected from a range
- are collected, but the collected value is deleted, because the answer is considered incorrect or unreliable or
- are not collected due to editing of the previous question.

The provision of imputed values enables the analysis of the data with complete-data methods. All imputed observations are flagged, and the nature of any originally missing values can be understood by examining the values of the correspondent flag variables. A minimum requirement of the HFCS data is that all missing observations of variables entering the computation of total household income, consumption and wealth are imputed. In addition, imputed values for some other variables are provided, but the selection of these variables is country specific.

A multiple imputation (MI) procedure yielding five values (‘implicates’) for each missing value is used in the HFCS[3]. Thus, the number of observations in the full data set is five times the actual number of respondents. MI offers two distinct advantages compared with singly-imputed data. First, because multiple imputation yields multiple outcomes from a random process, it supports more efficient estimation than singly-imputed data. Second, multiple imputation allows users to make straightforward estimates of the degree of uncertainty associated with the missing information (Rubin 1987).

Information from all five implicates should be used to generate best point estimates and best estimates of variances for parameters of interest. This is achieved by first analysing each of the five data sets by complete-data methods and then combining the results across implicates.

Suppose the interest lies in a point estimate of some parameter Y (e.g. mean, median, regression parameter) and that for each of the five imputed datasets we have obtained an estimate of Y (using standard complete-data methods), denoted $\hat{Y}$. The MI point estimate of Y, $\bar{Y}$, is the average of the five complete data estimates

$$\bar{Y} = \frac{1}{5}\sum_{i=1}^{5} \hat{Y}_i$$

The variance associated with this estimate $\bar{Y}$ has two components:

(i) the within imputation sampling variance W which is the average of the five complete-data variance estimates ($\hat{V}_i$):

$$W = \frac{1}{5}\sum_{i=1}^{5} \hat{V}_i$$

(ii) the between imputations variance B which reflects the variability due to imputation uncertainty and is the variance of the complete data point estimates:

$$B = \frac{1}{4}\sum_{i=1}^{5} \left(\hat{Y}_i - \bar{Y}\right)^2$$

The total variance is given by:

$$T = W + \frac{6}{5}B$$

3 Multiple imputation procedure is not applied in Finland and Italy (all waves), in France (second and third waves), Ireland (second wave), and Malta (third wave). In the second wave Hungary used multiple imputation for 33 variables and single imputation for the remaining imputed variables. However, to keep the structure of the data consistent, also the single imputed data sets have five implicates, but the imputed values of each implicate are identical.

For producing regression results, one should be cautious in the treatment of implicates, since many regression packages will treat each of the five implicates as an independent observation and correspondingly inflate the reported significance of results. On producing regressions with the repeated-imputation inference method (RII), see the discussion by Montalto and Sung (1996).

2.2 Survey weights

The HFCS data set has a variable on cross-sectional weights (HW0010) to compensate for unequal probability of the household being selected into the sample and differential unit non-response. In the construction of these weights account is also taken of the household composition and therefore the weight is the same for the household and for any of the household members. The weight HW0010 is available in two files: the D-file and H-file. For the person-level P-file, weight needs to be merged from either of the two household files.

Weights are adjusted for information on external sources using the calibration to margins method. The method consists of adjusting weights to permit extrapolating survey results and obtaining totals consistent with information known over the whole population from external sources (such as distribution by age and sex). The choice of calibration variables is country specific. The sum of weights over all households in the sample is an estimate of the total number of households in the population of each respective country (i.e. the weights reported are the inverse of the probability that a household is in the sample).

Further information on the HFCS weighting procedures is available in the Methodological report for the wave 2021, section 5.3.

Weights play a critical role in the interpretation of survey data. Taking into account weights is crucial in obtaining population totals, means, and shares from the data. However, there is some controversy on when weights should be used in regressions. Each user has to evaluate the situation given the objectives of the analysis at hand and users should think carefully about the effects of weights in their particular models. Useful references on these issues are provided by Cameron and Trivedi (2005), Deaton (1997) and Skinner et al. (1989).

2.3 Replicate weights and variance estimation

Variance estimation refers to computation of standard errors from the HFCS data acknowledging the complex sampling designs. The HFCS variance estimation methods are described in the Methodological report for the wave 2021, chapter 7.

In order to allow users to estimate variance for the HFCS, taking into account the sampling design employed in each country, countries provide **replicate weights** using a bootstrap replication method[4]. These are provided in a separate file (W-file). In the bootstrap procedure, a with-replacement sample of primary sampling units (PSU) from each stratum is selected. The number of PSUs per unit does not need to be constant. The number of replicates (bootstrap samples), as well as the number of PSUs sampled in each replicate, can be chosen by the analyst. The number of replicates in the HFCS data is 1000.

The variant of bootstrap for the HFCS is the rescaling bootstrap of Rao and Wu (1988), as further specified by Rao, Wu, and Yue (1992). It is applicable for one-stage samples and can be used as well in the case of a multi-stage samples drawn with a low sampling fraction in the first stage[5]. While – as all bootstrap methods – the rescaling bootstrap is computationally intensive and the resulting variance estimates may be less stable than with other methods (such as Jackknife and linearization), it provides consistent variance estimates in case of non-smooth statistics such as distribution quantiles.

The formula for the variance of an estimator $\hat{\theta}$ can be obtained by using the replicate weights as follows:

$$V(\hat{\theta}) = \frac{1}{H}\sum_{h}\left(\hat{\theta}_h - \bar{\theta}\right)^2,$$

where H is the number of replicates, h the index of the replicate sample and $\bar{\theta}$ the weighted estimate obtained in replicate sample h.

Since the final survey weights are adjusted for non-response, post-stratified or calibrated, the replicate weights are adjusted as well according to the same procedure (for example by running CALMAR calibration program with the same margins on each of the replicate weights). This can be considered as an additional rescaling factor. For instance, after drawing the sample and rescaling the weights, the weights are further rescaled to satisfy post-stratification or calibration constraints for each replicate. This is to ensure that the replicate estimates are close to be unbiased in each replicate sample.

For the HFCS statistical tables, standard errors have been computed for all countries and the euro area by looping over the 1,000 replicates for each

[4] Bootstrap was selected over other methods (e.g. Jackknife or balance repeated replication) because it allows analysts to select the number of replicates (in other methods the number of replicates is determined by the number of strata and/or number of PSUs). Besides, bootstrap samples are independently drawn across strata, so the replicate weights of different countries can be stacked and analysed as if they came from a single bootstrap procedure, thus allowing users to calculate a variance in the combined euro area dataset in a standard way.

[5] This is the case in several popular setups of stratified sampling. In addition, other sampling designs can be approximated by this setup.

of the five implicates and combining the results according to the formulas given above, using SAS software.

To compute standard errors, the data file and the replicate weights file must be first merged. Replicate weights file has only one implicate in the UDB (IM0100=1). The following example in SAS first combines the five D-files, available in SAS library UDBW4, and then merges the analysis variable and sampling weights from this file with the replicate weights file (W) with proc SQL.

```
Data work.DW4; set UDBW4.d1-UDBW4.d5; run;

proc sql;
	create table hfcs as
	select a.sa0100,a.sa0010,a.im0100,a.dn3001,a.hw0010,b.*
	from work.dw4 a , udbw4.w b
	where a.sa0100=b.sa0100 and a.sa0010=b.sa0010
	order by im0100;
quit;
```

3 Using the HFCS with Stata

This chapter proposes a step-by-step approach to working with the HFCS in Stata (starting in version 12.1), taking into account both the multiple imputation framework and the replicate weights, in order to provide correctly calculated standard errors.

As stated above, in order to use the information from all five implicates for parameters and variance estimation, each of the five data sets should be analysed separately and then the results should be combined across implicates.

To facilitate the imputation and the use of multiply imputed data, Stata provides the *mi* suit of commands. mi's estimation step encompasses both estimation on individual datasets and pooling in one procedure. The following instructions illustrate how the data can be imported in Stata to allow the use of mi estimation commands, also in combination with the svy command and the use of replicate weights. However, since working with the complete multiply imputed datasets and the replicate weights requires large memory (RAM) and the estimation processes become slow, users are advised to first get familiar with the data and test their analyses in one of the implicates, before using all implicates together.

3.1 Importing the data to allow estimation in the mi framework

The UDB (user database), as available from the ECB, can be downloaded in Stata format. We assign the global variable `$HFCSDATA` for the folder where the data are stored. The data consist of several .dta files containing the core variables: H1,…H5, P1,…P5, D1…D5, W, as well as files containing the non-core variables: HN1,…, PN1,…

```
global HFCSDATA "[...folder...]"
```

The questionnaire variables are provided in one format, with the household- and personal-level separate (h and p files for households and persons, respectively). The questionnaires files are provided in 5 versions, called implicates, which result from the process of multiple imputation. Each version is meant to be used as complete data (i.e. there is no item non-response), and the analysis carried out in the 5 implicates needs to be combined to provide the correct point estimates and standard errors.

3.1.1 Memory considerations

Users may need to allocate more memory than the Stata default. In Stata 12 and above, memory is managed dynamically by the software itself and does not need to be set. It still useful to set the maximum variables to 8000 (or higher) if the default is lower. It may also be useful to drop unneeded variables when loading the data. Because of the large memory requirements, it is also recommended that the commands below are first tried out with the data of a few countries only.

```
set maxvar 8000
```

A further possibility to reduce the memory requirements is to convert the replicate weights to float format, rather than their current double format. This halves memory requirement for the replicate weights, at the expense of a very limited loss in precision. This is done further below.

3.1.2 Preparing the data for MI import

Once the data are imported in Stata, the data need to be prepared to be used with Stata's mi tool. The first step is to correct some issues that interact badly with the mi tool: all string variables (except the ID) need to be recoded as numerical variables. The following routine converts such variables to numeric ones, giving the values a label corresponding to the original string.

```
/* this subroutine is used to encode string variables as numeric, saving the
strings in the label */
program encodestrings
	syntax varlist
	foreach var of varlist `varlist' {
		capture confirm numeric variable `var'
		if _rc {
			rename `var' `var'_string
			encode `var'_string, gen(`var')
			drop `var'_string
		}
	}
end
```

To save memory at the expense of a minimal amount of precision, we recast the replicate weights as floats instead of doubles.

```
/* we encode the string of the country in the w file, to be merged later on */
use "$HFCSDATA\w"
encodestrings sa0100
quietly recast float wr*, force
save w, replace
```

Stata's **mi** routines expect implicate number 0 to contain missing values. We create this implicate and set as missing all values in implicate 0 that vary over implicates. Implicate 0 is not used in the calculations; users are advised to use the information available in the flags rather than relying on Stata's **mi misstable** routine.

```
use "$HFCSDATA\h1"
replace im0100=0
append using "$HFCSDATA\h1" "$HFCSDATA\h2" "$HFCSDATA\h3" "$HFCSDATA\h4"
"$HFCSDATA\h5"

/* for some strange reason string variables do not play well with mi and need to
be encoded */
encodestrings sa0100 sb1000 hd030*

/* set as missing in im0100==0 all values varying, and also those whose flags set
them as imputed */
global IMPUTEDVARSH ""

foreach var of varlist hb* hc* hd* hg* hh* hi*  {
        capture confirm numeric variable `var'
        if !_rc {
                tempvar sd count
                quietly bysort sa0100 sa0010 : egen `sd'=sd(`var')
                quietly bysort sa0100 sa0010 : egen `count'=count(`var')
                quietly count if ( (`sd'>0 & `sd' <. ) | inrange(`count',1,5) |
(f`var'>4000 & f`var'<5000) ) & im0100==0
                if r(N)>0 global IMPUTEDVARSH "$IMPUTEDVARSH `var'"
                quietly replace `var'=. if ( (`sd'>0 & `sd' <. ) |
inrange(`count',1,5) | (f`var'>4000 & f`var'<5000) ) & im0100==0
                drop `sd' `count'
                disp ".", _continue
        }
}
/* some more housekeeping */
drop id
```

To save some more space, all other numeric variables can be converted to floats, saving approximately 35% of memory space at the expense of a bit of precision.

```
/* this converts all double variables as floats, saving 35% of memory */
foreach var of varlist hb* hc* hd* hg* hh* hi* {
        capture confirm double variable `var'
        if !_rc {
                quietly recast float `var', force
        }
}
```

3.2 Setting up multiple imputation

The data are now ready to be imported into mi.

```
/* import as multiply imputed data */
```

```
mi import flong, m(im0100) id(sa0100 sa0010) clear
```

Before the data can be converted, we need to register the relevant variables as “imputed”. We use the list of variables created above.

```
mi register imputed $IMPUTEDVARSH
```

Alternatively, it is possible to use an **`mi`** routine to detect variables varying across implicates and correcting the status of the variables.

```
mi varying
local unregistered `r(uvars_v)'
mi register imputed `unregistered'
```

The following command will inspect the data and report on imputed variables. In particular, the variables listed in the “unregistered super varying” line need to be inspected, to report possible issues with the setup of the data. Stata might reply with “**`type mismatch - r(109);`**”. This usually indicates that a string variable is left in the dataset and needs to be converted (see above for the **`encodestrings`** function).

```
mi varying
```

If previous steps were successful, the previous command should only report a few non-critical variables in the “unregistered varying” line (using data for the 2017 wave at the time of writing, 239 flag variables from FHB0800 to FHI0400l, and the implicate number variable IM0100). Stata will replace the varying values of these variables with the value found in the 0th implicate.

The data can now be converted to wide format (see the Stata documentation help mi styles for more information). This almost halves memory consumption. This would also be a good moment to save the data. Users are advised to drop variables not needed in their further analysis at this stage.

```
/* convert to wide style - memory preserving */
drop im0100
mi convert wide, clear
save h_wide
```

Once the data have been **`mi imported`**, a few tools are available to manage it. For example, **`mi append, mi merge, mi reshape`** to append, merge, and reshape data. The generic command **`mi xeq: cmd`** executes the command **`cmd`** on each implicate. This is particularly useful in flongsep data; in other cases it is better to use other commands, e.g. **`mi passive: generate lnprice=log(hb0800)`** to create a new variable.

3.3 Setting up new survey variables

Before the survey variables are set up, we need to merge the replicate weights file w. This requires the data to be in the **wide** format (see above).

```
merge 1:1 sa0100 sa0010 using w
drop _merge
```

To merge other imputed datasets, it is preferable to use the **mi merge** command. This step is memory intensive and may take some time to complete.

The survey environment is controlled with the **svyset** command. This command with the option **vce(bootstrap)** only works in Stata 11.1 and higher. The number of replicate weights can be adjusted by changing the variables in the **bsrweight** option, as running with 1000 replicate weights takes longer than running with 100 for testing purposes.

```
mi svyset [pw=hw0010], bsrweight(wr0001-wr1000) vce(bootstrap)
```

3.4 Using standard estimation procedures

Once the data are **mi svyset**, estimation commands are run as follows:

```
mi estimate: svy: mean hb0100
mi estimate: svy: proportion hb0300
mi estimate: svy: ratio hi0100 hb0100
mi estimate: svy: regress hb0100 hb0300
```

Stata will complain in versions 12.0 and 12.1 that "**vce(bootstrap) previously set by mi svyset is not allowed with mi estimate**". The use of the undocumented **vceok** option to mi estimate will allow the commands to run properly.

```
mi estimate, vceok: svy: mean hb0100
mi estimate, vceok: svy: proportion hb0300
mi estimate, vceok: svy: ratio hi0100 hb0100
mi estimate, vceok: svy: regress hb0100 hb0300
```

Stata will issue an error if the estimation sample varies across implicates: **estimation sample varies between m=1 and m=2; click here for details**. This is normal not unusual in the HFCS, since branch variables can be imputed, leading to "true" missing values in some implicates. The **esampvaryok** option of **mi estimate** allows the estimation to proceed.

```
mi estimate, vceok esampvaryok: svy: mean  hb0900, over(sa0100)
```

By adding the option **vartable** to **mi estimate**, it is possible to display the different components of variance (within and between imputation variance, relative increase in variance, fraction of missing information, relative efficiency).

3.5 Including additional estimation procedures

The **svy estimate** command only works with a limited number of commands. In particular, calculating a median or another quantile is not straightforward, and requires an intermediate Stata program. The **cmdok** option below is used to skip a check that the routine is not a standard mi estimation routine.

```
program define medianize, eclass properties(svyb mi)
/*******************************************************

MEDIANIZE - calculate a median

This Stata command estimates the median and can be combined
with the svy and mi commands, only with replicate weights.

Syntax: syntax varlist [weight] [if] [in] [, over(varname) Statistic(string)]

In order to combine with mi and svy, only one variable can
be given in varlist.

Statistic is any statistic accepted by tabstat.

*******************************************************/

    syntax varlist [aweight iweight pweight] [if] [in] [,over(varname)
Statistic(string)]

    marksample touse, novarlist zeroweight

    * iweight cannot be used with qreg
    if "`weight'"!="" local weight="aweight"

       * statistic by default - the median
    if "`statistic'"=="" local statistic="p50"

       /* calculate the statistic of interest with tabstat */
       tabstat `varlist' [`weight'`exp'] if `touse', s(`statistic')  by(`over')
save

       matrix _zz=r(Stat1)

       if _zz[1,1]==. matrix _zz=r(StatTotal)

       capture matrix drop _z

       /* construct the e(b) output and assign the correct colnames */
       local i=1
       local names
       while _zz[1,1] ~= . {
              if "`i'"=="1" matrix _z=_zz
              else matrix _z=_z,_zz

              /* replace spaces in the labels by underscores */
```

```
          local tempname `r(name`i')'
          local tempname : subinstr local tempname " " "_", all

          local names=`"`names' "`tempname'""'

          local ++i
          matrix _zz=r(Stat`i')
     }

     matrix colnames _z = `names'

     /* the sample used */
     tempvar one
     gen `one'= `touse'
     ereturn post _z, esample(`one')

     /* arguments required by svy and mi */
     ereturn local cmd medianize
     ereturn local title "Medianize `statistic'"

     quiet count if `touse'
     ereturn local N r(N)

     capture matrix drop _z _zz
end
```

The following commands show the use of **`medianize`**.

```
mi estimate: svy: medianize hb1701
mi estimate: svy: medianize hb1701, over(sa0100)
mi estimate: svy: medianize hb1701, over(sa0100) stat(p10)
```

(Do not forget to add the **`vceok`** and the **`esampvaryok`** in case of errors.)

3.6 Useful commands

The information specific to each household member (e.g. age, gender, education, personal income) are stored in the P files. To merge this information with the household-level data of the H and D files, it is necessary to convert the P file to a "wide" format. This is done with the simple Stata instruction, which needs to be run on each of the 5 implicates. The resulting file can then be merged with the corresponding H file:

```
use P1
gen tmp="_"+string(ra0010)
drop id
reshape wide r* p* f* , i(sa0100 sa0010) j(tmp) string
```

3.7 Additional commands and instructions

3.7.1 Fine-tuning the calculation of variance

```
svy, bsn(1.001001)
```

Adding this option to the **svy** command can correct the denominator used in the bootstrap variance formula. By default, in Stata the number B of replicate weights is used, whereas in the literature it is also possible to find $B - 1$. With 1000 replicate weights, the difference is marginal.

3.7.2 Speed improvements

Working with a dataset of around 90,000 households, 5 implicates, and 1,000 replicate weights can take a significant amount of time. Optimizing the use of the mi routines is thus helpful.

<u>Instructions that can be used in (almost) all cases</u>

```
mi estimate, noupdate:...
```

For commands that do not modify the data, the **noupdate** option skips a check in Stata, and allows commands to run slightly faster, especially on big and complex datasets like the HFCS.

<u>Instructions which affect the results</u>

The following few instructions can be helpful during exploratory work, but need to be rolled back when preparing the final material.

```
mi estimate, nimputations(2):...
```

This option only considers 2 implicates, and not all 5 of them. The point estimates and the standard errors are therefore not correct.

Another alternative is to use the **flongsep** style of multiply imputed data in Stata, which only loads one implicate in memory.

```
mi svyset [pw=hw0010], bsrweight(wr0001-wr0010) vce(bootstrap)
```

Changing the number of bootstrap weights reduces the number of computations that need to be made. The point estimates are correct, but the standard errors are not.

4 Using HFCS with R

This chapter proposes a simple approach to work with HFCS in R[6] (starting in version 3.6.5) which enables to manage both multiple imputation framework and replicate weights therefore allowing to correctly calculate standard errors.

4.1 Importing the data

The UDB (user database), as available from the ECB, can be downloaded in ASCII format. The HFCS data structure is described in chapter 1 of this document. The data consist of several household and personal level ASCII files containing the core variables: H1,…H5, P1,…P5, D1…D5, W, as well as files containing the non-core variables: HN1,…, PN1,…The files are provided in 5 versions, called implicates, which result from the process of multiple imputation. Each version is meant to be used as complete data (i.e. there is no item non-response), and the analysis carried out in the 5 implicates needs to be combined to provide the correct point estimates and standard errors.

When using **RStudio**, importing ASCII files can be easily done by clicking on the “Import dataset” button in the Workspace, choosing “From Text file…” and selecting the file to be imported. One of the main advantages of the feature is that datasets can be visualised before importation. However, this kind of routine might select automatically the format of the variables. For example, all numeric variables are considered by the system as double, which consumes resources in terms of memory.

Another option is to import manually the data with the following code:

```
H1= read.csv("path"[7], sep=',',header=TRUE, stringsAsFactors=FALSE, na.string =
c('','.'))
save(H1,file='H1.RData')
```

In order to speed up the process you can use the “nrow” and ‘’colClasses’’ argument in the previous command or subset the dataset in order to better manage the conversion of all the variables. Another option to import your dataset might be to use the function *fread* offered by the package ‘’data.table’’[8].

[6] The codes presented in this section are based on the following GitHub:https://github.com/pierre-lamarche/Household-Finance-and-Consumption-Survey.

[7]It is possible to avoid inserting the path by explicating before the working directory via setwd(“path”’).

[8] For some hints on how to handle large datasets in R: https://rpubs.com/msundar/large_data_analysis

Please note that once the data have been saved (see the last line of the code example), the dataset can be easily be reloaded in the R workspace by submitting the following code:

```
load('H1.RData')
```

4.2 Memory Considerations

R stands as a memory-consuming software that uses only RAM resources. Using the HFCS data with a 64-bit CPU would enable to keep the entire dataset in memory but those users with 32-bit configuration have to subset the data in order to be able to work in R.

Indeed, the total memory size needed to store the HFCS data might reach 2.9 GB in R and this might result in the warning message: “Error: cannot allocate vector of size xxx Kb”. Hence it is important to monitor the size of the objects and the memory used. For example, through the package ‘’pryr’’ you can run the following commands and control the memory used:

```
object_size ('H1')

object_size ('H1', 'D1') #to know how much memory they occupy together

mem_used()#return the total amount of memory (in megabytes) currently used by R
```

Note that If the computer disposes important quantities of RAM (up to 4 GB RAM), it is possible to increase the amount of memory used by R with the command:

```
memory.limit(size=)
```

As underlined above, reshaping the data and selecting the only relevant variables will be necessary for users with 32-bit configuration.

4.3 Data Wrangling

After loading the desired datasets, a range of different actions can be performed. For example the user might want to merge the five different implicates:

```
list_tab = c("H")

for (i in 1:length(list_tab)){
  for (k in 1:5){
    txt = paste("load('",list_tab[i],k,".RData')",sep="")
    #    print(txt)
    message(paste("Loading of table ",list_tab[i],", implicat ",k,sep=""))
    eval(parse(text=txt))
  }
}

H.complete<- rbind(H1,H2,H3,H4,H5)
```

In addition to this, it might be interesting to merge information from the different available datasets. In what follows a purely illustrative and recyclable routine is shown:

```
load('P1.RData')
load('H1.RData')

tab= H1[,c("SA0010","SA0100","HW0010", "IM0100", "HB0100")]
impl = merge(tab,P1,by=c("SA0010","SA0100", "IM0100" ))



#automatize the same command for the five implicates

list_tab = c("H","P")

for (i in 1:length(list_tab)){
  for (k in 1:5){
    txt = paste("load('",list_tab[i],k,".RData')",sep="")
    #    print(txt)
    message(paste("Loading of table ",list_tab[i],", implicat ",k,sep=""))
    eval(parse(text=txt))
  }
}

for (j in 1:5){
  txt = paste("tab=H",j,"[,c('SA0010','SA0100','HW0010','HB0100', 'IM0100' )] \n
rm(H",j,") \n imp",j,"=merge(P",j,",tab,by=c('SA0010','SA0100', 'IM0100')) \n
rm(P",j,")",sep="")
  message(paste("Merging household and personal information",j,sep=""))
  eval(parse(text=txt))
}
```

Indeed the user can use the above codes for selecting different variables or merging other datasets.

Another tool to select and manipulate variables/datasets is the ‘’dyplr’’ package which is easy to use, for example:

```
load('H1.Rdata')
imp1<-H1 %>% filter( SA0100=="ES")%>%
select("SA0010","SA0100","HW0010", "IM0100", "HB0100")
```

As before, the user might create five different implicates with this code and finally be able to compute any indicator with HFCS data.

4.4 Multiple Imputation and estimation

Once that the five implicates are created, the packages “survey” and “mitools” are needed in order to compute the estimators and their variance. Indeed, the survey package is useful to manage data coming from complex surveys whereas mitools is useful for multiple-imputed data. Moreover, the user will now need to load and save the replicate weights file (W) in order to fully consider the uncertainty caused by sampling.

Therefore, the suggested procedure would be as follow:

```
install.packages('survey')

load('W.RData')

repweg= select(W, "wr0001":"wr1000")

hfcs.design = svrepdesign(repweights=repweg, weights= ~HW0010,
data=imputationList(list(imp1,imp2,imp3,imp4,imp5)),scale=1,rscale=rep(1/999,1000
),mse=FALSE,type='other', combined.weights=TRUE)
```

The result will be a *svyimputationList* objejct which will be used for estimation. Note that such computation can require some time depending on the pc performances.

A range of different estimations can be accessed via survey package documentation, below few examples that also illustrate the use of ‘’mitools’’.

```
install.packages('mitools')

MIcombine(with(hfcs.design,svymean(~ HB0100)))

#depending on the variables included in your five implicates you might want to
compute also the following:

MIcombine(with(hfcs.design,svytotal(~DA3001)))

MIcombine(with(hfcs.design,svyratio(~DA3001,~DH0001)))
```

:

5 The panel component in the HFCS

In several countries, the data involves a panel component. This means that some household members interviewed in wave 2021 (fourth wave), wave 2017 (third wave) or wave 2014 (second wave) were also included in the net sample of the previous wave(s). For convenience, Box 5 summarises the countries with panel data (and having panel ID variables available) in the four HFCS waves.

Box 5
Countries having a panel component in the HFCS

Country	Waves
Belgium	1,2,3,4
Germany	1,2,3,4
Estonia	3,4
Ireland	4
Spain	1,2,3,4
France	3,4
Italy	1,2,3,4
Cyprus	1,2,3,4
Latvia	3
Lithuania	4
Malta	1,2,3,4
Netherlands	4
Slovakia	3,4
Finland	3,4

It should be noted that the use of HFCS data for panel analysis is still best considered being at an experimental phase. There are no specific longitudinal or panel weights available to be used for a proper analysis of panel data. Some countries have constructed panel weights for their data, but these are not included in the current version of the UDB. Moreover, the panel follow-up rules are not harmonised in the HFCS, with some countries following persons and some following households or addresses.

In order to link members of the panel from the two data sets, variables SA0110 “Past household id” and RA0020 “Past personal id” are included in the UDB data for all those countries that have delivered them. The variable SA0110 identifies households that belong to the panel component.

Household panel ID variable SA0110 is available only in the H-file and personal panel ID variable RA0020 only in the P-file. Therefore, the household and personal level variables need to be merged first before linking the data to the previous wave. Since the composition of households may change from one wave to the next and because the follow-up rules may differ across the countries, it is advisable to construct the data wave by linking both households and persons.

The SQL commands below give an example on how to do this for the 3rd wave data for the purpose of having household net wealth, household size and age of household members in the panel data file. The code is in SAS proc sql, but it can be easily adapted to generic SQL code.

```
/* List the countries with a panel component in the wave 2017 */
%let panelcountries=%str("EE","FR","LV","SK","BE","DE","ES","IT","CY","MT","FI","PL");

/* Prepare wave 2017 to have the required variables at household and personal level */
proc sql;
       create table w3_paneldata as select a.*,b.sa0110,c.ra0010,c.ra0020,c.ra0300
       from
              /* select analysis variables from the d-file */
              (select sa0100,sa0010,im0100,hw0010,
               dn3001 as dn3001_wave3,dh0001 as hhsize_wave3
                     from udbw3.d1
                     where sa0100 in (&panelcountries)) a
              /* link household id from h-file */
              inner join
              (select sa0100,sa0010,im0100,sa0110 from udbw3.h1) b
              on a.sa0100=b.sa0100 and a.sa0010=b.sa0010 and a.im0100=b.im0100
              /* link personal panel id and age from p-file */
              left join
              (select sa0100,sa0010,im0100,ra0010,ra0020,ra0300 from udbw3.p1) c
              on a.sa0100=c.sa0100 and a.sa0010=c.sa0010 and a.im0100=c.im0100
       order by sa0110 desc,sa0100,sa0010,ra0010,ra0020;/* Sort the data by ID variables */
       ;
quit;


/* Link the wave 2017 file to wave 2014 persons and households */
proc sql;
       create table w3w2_paneldata as
       select a.*,b.ra0010_w2,b.ra0300_w2,c.dn3001_w2,c.hhsize_wave2
        from
       (select * from w3_paneldata) a
       left join
       (select sa0100,sa0010,im0100,ra0010 as ra0010_w2,
         ra0300 as ra0300_w2 from udbw2.p1)  b
       on a.sa0100=b.sa0100 and a.sa0110=b.sa0010 and a.ra0020=b.ra0010_w2
        and a.im0100=b.im0100
       left join
       (select sa0100,sa0010 as sa0010_w2,im0100,
       dn3001 as dn3001_w2,dh0001 as hhsize_wave2 from udbw2.d1) c
       on a.sa0100=c.sa0100 and a.sa0110=c.sa0010_w2 and a.im0100=c.im0100
       order by sa0110 desc,sa0100,sa0010,ra0010,ra0020;/* Sort the data by ID variables */
       ;
quit;
```

An example using stata to link panel household between waves is given below. Since linking such households is done through the sa0110, which is specific to a specific wave, we need to create an id variable with a common name through waves. So, in order to merge the panel observations across two waves, for example across waves 3 and 4, we need to create a common id in the files of these two waves, sa0110_w3. Below is an example of linking the panel observations of wave 3 and wave 4 data for the countries BE and IE, using the first imputation.

```
** Before running the code below create the globals HFCSDATAw4,  HFCSDATAw3, and
outputdir to specify the directories for HFCS wave 4, HFCS wave 3 data and an
ouput directory respectively
*global HFCSDATAw4  " "
*global HFCSDATAw3 " "
*outputdir    "   "

use "$HFCSDATAw4\h1.dta", clear

** wave variable if data are to be appended

gen wave=4

** gen linking id for previous wave
clonevar sa0010_w3=sa0110
keep if sa0100=="BE" | sa0100=="IE"

* sort
sort sa0100 sa0010 sa0010_w3
save "$outputdir\h1_w4_BE_IE.dta", replace


*** rename/generate wave specific variables
gen hb0300_w4=hb0300
gen hb0900_w4=hb0900
gen hb1701_w4=hb1701


keep if sa0110<.
save "$outputdir\h1_w4_BE_IE_panel.dta", replace


** wave 3

use "$HFCSDATAw3\h1.dta", clear

** wave variable if data are to be appended
gen wave=3

** gen linking id for next wave
clonevar sa0010_w3=sa0010

** gen linking id for previous wave
clonevar sa0010_w2=sa0110

*** select variables to be merged and rename them
gen hb0300_w3=hb0300
gen hb0900_w3=hb0900
gen hb1701_w3=hb1701

* sort data
sort sa0100 sa0010 sa0010_w2

keep if sa0100=="BE" | sa0100=="IE"
save "$outputdir\h1_w3_BE_IE.dta", replace

** for linking with wave 2
keep if sa0110<.
save "$outputdir\h1_w3_BE_IE_panel.dta", replace


*******  merge waves 3 & 4
use "$outputdir\h1_w4_BE_IE_panel.dta", clear
```

```
merge 1:1 sa0100 sa0010_w3 using  "$outputdir\h1_w3_BE_IE.dta",
keepusing(hb0300_w3 hb0900_w3 hb1701_w3)
```

As previously mentioned, countries use different rules to follow initially sampled households. For example, in DE all household members of initially sampled households are followed. In that case, from an initially sampled household there may be more than one panel households in a future wave. In the data, this results in duplicate sa0110s. In that case one needs to use the merge m:1 stata command. Moreover, household members are followed even if they have skipped one wave and so in waves 3 and 4 there are households that have an sa0110 but can only be linked with households in wave 2 or 1 respectively. To make this link one needs to create a common id with both of the previous waves (for example, in wave 4, in addition to clonevar sa0010_w3=sa0110, also do: clonevar sa0010_w2=sa0110).

The following issues with the panel IDs are known in the UDB data:

- Linking can only be done at household level for Poland because variable RA0020 is not available in the wave 2017. In addition, SA0200 survey vintage is panel year and not the survey year (to fix this, change SA0200=2016 in the data).
- For Finland, there are 190 observations in the 3rd wave with IDs SA0110 (past household ID) which do not correspond to any household ID (SA0010) in the 2nd wave. These cases belong to the sample refreshment and should not have a panel id. Users can correct for this by treating such cases as non-panel households/individuals.
- For Germany and Cyprus, there are observations with IDs SA0110 (past household id) in wave 2 or wave 3 which do not correspond to any household ID in wave 1 or wave 2. This issue has not been resolved in the current version of the UDB.

It should be noted that the composition of “panel households” (whose ID is included in both survey waves) can change between the waves. The personal IDs of new household members (entries such as new-borns) have an empty value for RA0020, and IDs of members who have left the household do not have any value of RA0020 in the latter wave data.

6 References

Cameron, A. C., and P. K. Trivedi (2005), “Microeconometrics: Methods and Applications”, Cambridge University Press.

Deaton, A. (1997), “The Analysis of Household Surveys: A Microeconometric Approach to Development Policy”, Johns Hopkins University Press, pp. 67-73.

HFCN (2020), “The Household Finance and Consumption Survey: methodological report for the 2017 wave”, *Statistics Paper Series*, No 35, ECB

HFCN (2023), “The Household Finance and Consumption Survey: methodological report for the 2021 wave”, *Statistics Paper Series*, No 45, ECB

Montalto, C. P. and Sung, J. (1996), “Multiple Imputation in the 1992 Survey of Consumer Finances”, *Journal of Financial Counselling and Planning*, Vol. 7.

Rao, J. N. K., Wu, C. F. J., and Yue, K. (1992), “Some recent work on resampling methods for complex surveys”, *Survey Methodology*, Vol. 18, No. 2, pp. 209-217.

Rao, J. N. K., and Wu, C. F. J. (1988), “Resampling inference with complex survey data”, *Journal of the American Statistical Association*, Vol. 83, pp. 231–241.

Rubin, D. B. (1987), “Multiple Imputation for Nonresponse in Surveys”, John Wile and Sons.

Skinner, C.J., Holt, D., and Smith, T.M.F. (editors) (1989) “Analysis of Complex Surveys”, John Wiley and Sons, pp. 8-10, 154-157, and 286-287.