
# SAMPLE DELIVERABLE 20

# SOFTWARE FOR THE POOLED INDICATORS

| | |
|---|---|
| Grant agreement No: | SSH - CT - 2007 – 217565 |
| Project Acronym: | SAMPLE |
| Project Full title: | Small Area Methods for Poverty and Living Conditions Estimates |
| Funding Scheme: | Collaborative Project - Small or medium scale focused research project |
| Deliverable n. | 20 |
| Deliverable name: | Software for the pooled estimators |
| WP no.: | WP 1.3 |
| Lead beneficiary: | 2 |
| Nature: | Report |
| Dissemination level: PU | PU |
| Due delivery date from Annex I: | 31 October 2010 |
| Actual delivery date: | 10 November 2010 |
| Project co-ordinator name: | Mrs. Monica Pratesi |
| Title: | Associate Professor of Statistics - University of Pisa |
| Organization: | Department of Statistics and Mathematics Applied to Economics of the University of Pisa (UNIPI-DSMAE) |
| Tel: | +39-050-2216252, +39-050-2216492 |
| Fax: | +39-050-2216375 |
| E-mail: | coordinator@sample-project.eu |
| Project website address: | www.sample-project.eu |

# SOFTWARE FOR THE POOLED INDICATORS

| | |
|---|---|
| Project Acronym: | SAMPLE |
| Project Full title: | Small Area Methods for Poverty and Living Conditions Estimates |
| Project/Contract No: | EU-FP7-SSH-2007-1 |
| Grant agreement No | 217565 |
| Work Package 1: | New indicators and models for inequality and poverty with attention to social exclusion, vulnerability and deprivation |
| Document title: | Software for the pooled estimators |
| Date: | 10 November, 2010 |
| Type of document: | Report |
| Status | Final |
| Editors: | Achille Lemmi, Francesca Gagliardi, Giulio Tarditi |
| Authors | Achille Lemmi (lemmi@unisi.it), CRIDIRE - UNISI<br>Vijay Verma (verma@unisi.it), CRIDIRE - UNISI<br>Gianni Betti (betti2@unisi.it), CRIDIRE - UNISI<br>Laura Neri (neri@unisi.it), CRIDIRE - UNISI<br>Francesca Gagliardi (gagliardi10@unisi.it), CRIDIRE – UNISI<br>Giulio Tarditi (tarditi2@unisi.it), CRIDIRE – UNISI<br>Caterina Ferretti (ferretti@ds.unifi.it), CRIDIRE - UNIFI |

# Contents

# SOFTWARE FOR THE POOLED INDICATORS

## 1. Introduction

The present document is meant to be a users' guide for the SAS and R programs developed for cumulated estimates. The theory, the logic and also results from cumulation are reported in our Deliverable D19 Pooled estimates of indicators.

The programs produce the average estimate over two consecutive waves and the corresponding variance. As example the estimation is done for Italy, using the EU-SILC data for 2007 and 2008.

The poverty measure used is the Head Count Ratio (HCR).

The estimate of the measure produced is:

$$(est1+est2)/2 \qquad (1)$$

where est1 and est2 are the estimate of the HCR in the cross-sectional dataset, one for 2007 and the other for 2008.

As shown in D19, the estimate of the variance comes from the following formula:

$$v_A = \frac{1}{2}.\left(\frac{V_1+V_2}{2}\right).\left(1+b.\left(\frac{n}{n_H}\right)\right) \qquad (2)$$

where

$V_1$ and $V_2$ are the variances estimated from the two cross-sectional samples,

$$b = \left(\frac{c_1}{v}\right) = \left(\frac{a-p^2}{p-p^2}\right)$$

*a* is the persistent poverty rate over the two years,

*n* is the overlap between the cross-sectional samples,

$n_H$ is the harmonic mean of the cross-sectional sample sizes n1 and n2.

Given that in EU-SILC data the cross-sectional datasets cannot be linked, correlation between the two waves cannot be computed directly.

So, we computed it through the longitudinal dataset with the following assumptions.

$b = \left( \frac{a - p^2}{p^2 - p} \right)$ where *a* is persistent poverty rate of the panel (e.g. 2007-2008 for Italy) and p is the average of the estimates done in the longitudinal dataset for each of the two years (e.g. 2007-2008 for Italy).

$n_h = \frac{2 \cdot n_1 \cdot n_2}{n_1 + n_2}$ where n1 and n2 are the sample size of the cross sectional dataset.

$n = 0.75 \cdot \min(n_1, n_2) \cdot \frac{m}{\min(m_1, m_2)}$ where m is the sample size of the panel (e.g. 2007-2008 for Italy) and m1 and m2 are the sample size of the longitudinal dataset in the two years.

## 2. Description of the programs

Both the programs are divided in three parts.

The first part concerns the longitudinal estimates needed as described above.

The second part concerns the cross-sectional estimates as described above.

The third part, very shortly, just put the result together and create the final required cumulated estimates.

Below, follows a detailed description of the SAS program, the R one has exactly the same logic.

### 2.1 First part of the program

In this part first is created a proper longitudinal dataset. So the original H, D and R longitudinal dataset are used. For a description of the construction of the longitudinal dataset , see Annex 1.

Once the longitudinal dataset is created, the estimates required from it are constructed; so the persistent poverty rate of the panel (e.g. 2007-2008 for Italy), the estimates of the HCR done in the longitudinal dataset for each of the two years (e.g. 2007-2008 for Italy).

Inside the macro **`%sub_ciclo`** these estimates are done using the corresponding macros:

**`%stat2`** : HCR Poverty Line as 60% of the median, long. file, year 2007;

**`%stat3`** : HCR Poverty Line as 60% of the median, long. file, year 2008;

**`%long3`** : Continuous poverty panel two waves (2007-2008).

The results are saved in a permanent dataset.

### 2.2 Second part of the program

This part has to be run twice, once for the cross-sectional estimates 2007 and the second for 2008.

The main macro is **`%sub_ciclo`** inside which all the estimates are done. The proper number of PSUs of the dataset has to be specified.

Inside it three main macros are present:

**`%stat1`** : HCR Poverty Line as 60% of the median, cross-sectional file;

**`%perc_bound`** : This macro allows to calculate any kind of weighted percentile of a distribution.

The value that it calculates is the linear interpolation of the percentile. This mean that if any real value of the distribution lies in this percentile, a value between the two nearest values - above and below the percentile is interpolated. It needs information from individual datasets (PID).

The SAS program doesn't calculate this value in such a manner. In fact if any real value of the distribution lies in this percentile, the SAS takes the nearest real value above the percentile.

We highly recommend this procedure for the estimation of percentiles.

**`%jrr`** : This macro implements the Jackkinfe Repeated Replications (JRR) methodology.

It implements the so called JRR variable (see Annex 2 for details). In order to estimate the standard errors, the measures are estimated inside the replications. In fact, inside this macro there is the computation of the "**`%stat1`**".

Inside the replications we also reallocate the weights. Once a PSU is deleted, its weights are assigned to the other PSUs in the same stratum of the PSU deleted, so that the total sum of the weights doesn't change (see formula [4] in Annex 2 for details).

**`%Kish`** : This macro calculates the quantity Kish_Jrr described in Annex 2 as the correlation between the weights and the statistic implemented.

The results are saved in two permanent datasets, one for 2007 and one for 2008.

### 3. SAS code

```
*********************************************************;
*This part is for the longitudinal estimates needed for cumulation.
It first create a proper longitudinal dataset;
*********************************************************;
data d; set silc08l.d08l (keep= db010 db020 db030 db040 db090 );
where db020 in ('IT')and db010 in (2007, 2008);
udb_d=1;
run;
data d; set d;
where db090 ne 0; run;
proc sort data= d; by db010 db020 db030; run;
data h; set silc08l.h08l (keep= hb010 hb020 hb030 hx090 hx050 hx040);
rename hb010=db010;
rename hb020=db020;
rename hb030=db030;
where hb020 in ('IT') and hb010 in (2007, 2008);
udb_h=1;
run;
proc sort data= h; by db010 db020 db030; run;
data all; merge d h;by db010 db020 db030; run;
data all2; set all; where udb_h ne .; run;
data r; set silc08l.r08l (keep= rb010 rb020 rb030 rb040 rb060 rb062
rx010 rx020);
rename rb010=db010;
rename rb020=db020;
rename rb040=db030;
where rb020 in ('IT') and rb010 in (2007, 2008);
udb_r=1;
run;
proc sort data= r; by db010 db020 db030; run;
proc sort data= all2; by db010 db020 db030; run;
data all_file; merge all2 r; by db010 db020 db030; run;
data all_file2; set all_file; where udb_h ne .; run;
proc sort data=all_file2; by db010 db020 rb030; run;
data b; set all_file2; d=dif(rb030); run;
data c (keep=db010 db020 rb030 double); set b; where d eq 0;
double=1;run;
proc sort data=c; by db010 db020 rb030; run;
data d; merge c all_file2; by db010 db020 rb030; run;
data e; set d; where double eq 1; run;
data f; set e; where rb060 ne 0; run;
data g; set d; where double eq .; run;
data r_long_nodouble; set f g; run;
data r07(keep=db020 rb030 r07); set r_long_nodouble ; where db010 eq
2007;r07=1; run;
data r08 (keep=db020 rb030 db030 r08); set r_long_nodouble ; where
db010 eq 2008;r08=1; run;
proc sort data=r07; by db020 rb030; run;
proc sort data=r08; by db020 rb030; run;
data ab; merge r07 r08; by db020 rb030; run;
data ab1 (keep=db020 rb030 r78); set ab;
r78=r07+r08; run;
proc sort data=ab1; by db020 rb030; run;
proc sort data=r_long_nodouble; by db020 rb030; run;
data operational_file; merge ab1 r_long_nodouble; by db020 rb030; run;
proc sort data=operational_file; by db010 db020 rb030; run;
```

```
data file07; set operational_file; where db010 eq 2007; run;
data file08; set operational_file; where db010 eq 2008; run;
data silc_it.longitudinal_file; set operational_file; run;

*********************************HCR P.L.as 60% median, long. file,
year 2007;
%macro stat2;
data input_bound;set working;wj=w0; where db010 eq 2007;run;
%perc_bound(50);
data working1;merge input_bound output_bound;by
country;line=0.60*y_perc; wj=ws;
if eqinc gt line then z=0;if eqinc le line then z=1;run;
proc sort data=working1; by country; run;
proc univariate data=working1 noprint;output out=est mean=est;var
z;weight wj;by country;run;
data jrr;merge working1 est;by country;y=z-est;subpop_i=1;run;
title 'stat1';
%mend;
*********************************HCR P.L.as 60% median, long. file,
year 2008;
%macro stat3;
data input_bound;set working;wj=w0; where db010 eq 2008;run;
%perc_bound(50);
data working1;merge input_bound output_bound;by
country;line=0.60*y_perc; wj=ws;
if eqinc gt line then z=0;if eqinc le line then z=1;run;
proc sort data=working1; by country; run;
proc univariate data=working1 noprint;output out=est mean=est;var
z;weight wj;by country;run;
data jrr;merge working1 est;by country;y=z-est;subpop_i=1;run;
title 'stat1';
%mend;
*********************************Continuous poverty panel two waves
(2007-2008);
%macro long3;
data input_bound;set working;wj=w0;where db010 eq 2007;run;
%perc_bound(50);
data output_bound1;set output_bound;
mediana1=y_perc; db010=2007; run;
data input_bound;set working;wj=w0;where db010 eq 2008;run;
%perc_bound(50);
data output_bound2;set output_bound;
mediana2=y_perc; db010=2008; run;
data working1;merge working output_bound1 output_bound2; by country
db010;pov_l1=0.60*mediana1; pov_l2=0.60*mediana2;
run;
data year1(drop=db010); set working1 (keep=db010 pid eqinc pov_l1);
where db010 eq 2007;
rename eqinc=eqinc1;run;
data year2(drop=db010 pov_l1 mediana1); set working1 ; where db010 eq
2008;
rename eqinc=eqinc2;run;
proc sort data=year1; by pid; run;
proc sort data=year2; by pid; run;
data long_file; merge year1 year2; by pid;
if r78 eq . then pov1=.;
if r78 eq . then pov2=.;
if (eqinc1 gt pov_l1) and (r78 eq 2) then pov1=0;
if (eqinc1 le pov_l1) and (r78 eq 2) then pov1=1;
```

```
if (eqinc2 gt pov_l2) and (r78 eq 2) then pov2=0;
if (eqinc2 le pov_l2) and (r78 eq 2) then pov2=1;
if pov1*pov2 eq 1 then z=1;if pov1*pov2 eq 0 then z=0;subpop_i=1;
wj=wlong;
run;
data long_file2; set long_file; where r78 eq 2; run;
proc sort data=long_file2; by country; run;
proc means data=long_file2 noprint; output out=est mean=est;var
z;weight wj;by country;run;
data jrr;merge long_file2 est;by country;y=z-est;run;
%mend;
**************************Perc bound;
%macro perc_bound (perc);
data input;set input_bound;percent=&perc;run;
proc sort data=input;by eqinc;run;
proc iml;
use input;read all var {pid} into pid;read all var {wj} into wj;
read all var {eqinc} into eqinc;read all var {percent} into percent;
read all var {country} into country_vec;
num=nrow(pid); share=repeat(0,num);ratio=percent[1]/100;
tot=sum(wj);
 i=2; do while (i<num+ 1);
      share[i]=sum(wj[1:i])/tot;
        if (share[i-1] < ratio) & (share[i] >= ratio)
      then y_perc=eqinc[i-1]+(eqinc[i]-eqinc[i-1])*((ratio-share[i-
1])/(share[i]-share[i-1]));
  i=i+1;  end;
country=country_vec[1];
create output_bound var {y_perc country};append ;close
output_bound;quit;run;
%mend;

data r; set silc_it.longitudinal_file;
where r78=2;
country=1;
rename rb030=pid;
rename hx090=eqinc;
I1=0; if r78=2 and db010 eq 2007 then I1=1;
I2=0; if r78=2 and db010 eq 2008 then I2=1;
I3=0; if r78=2 then I3=1;
run;
data r1 (drop= I1--I3); set r; run;
data h0;input country ah psu stratum stat;cards;
0 0 0 0 0
;run;
data stat0;input est stat_se ;cards;
0 0
;run;
data kish0;input kish index;cards;
0 0
;run;
data final_output0; input subpopulation $1-74 est stat_se  kish ;
cards;
This_is_for_initialising_the_Subpopulation_considered_in_the_analysis_
____ 0 0 0
;run;
%macro sub_ciclo (sub_ciclo_start,sub_ciclo_end);
%do j=&sub_ciclo_start %to &sub_ciclo_end;
data help; set r (keep=db010 pid I&j);rename I&j=I;run;**nome file;
```

```
proc sort data=r1; by db010 pid; run;
proc sort data=help; by db010 pid; run;
data dati; merge r1 help; by db010 pid; run;
data working_pop_1;set dati;run;
data kish;set kish0;run;
data stat; set stat0; run;
data working_pop0;set working_pop_1;w0=db090;w_sub=w0*I;
w_long=rb060*I;
run;
proc sort data=working_pop0; by country db010; run;
proc univariate data=working_pop0 noprint;
output out=sum_w sum=sum_w sum=sum_w_sub sum=sum_w_long;var w0 w_sub
w_long;by country db010;run;***;
data sum_w;set sum_w (keep=sum_w sum_w_sub sum_w_long country
db010);run;
data working_pop; merge working_pop0 sum_w;by country db010;
w0=db090/sum_w; ws=w_sub/sum_w_sub;whij=w0;wlong=w_long/sum_w_long;
run;
data working;set working_pop;run;
%if (&j eq 1) %then %do; %stat2; %end;
%if (&j eq 2) %then %do; %stat3; %end;
%if (&j eq 3) %then %do; %long3; %end;
proc univariate data=jrr noprint; output out=est mean=est n=n;var z;
weight wj;where (I eq 1) and (subpop_i eq 1);by country;run;
data est; set est (keep= est n country);run;
data stat_act (keep= est n index);set est ;by country;index=&j;
data stat;set stat stat_act;run;
proc freq data=stat_act;table index;run;
data JRR1;set stat;
where (est ne 0) ;run;
data stat_output; set stat;where (est ne 0) and (stat_se ne 0); run;
data stat_sub&j (drop=index); set stat_output;
if &j eq 1 then Subpopulation='HCR 2007                    ';
if &j eq 2 then Subpopulation='HCR 2008';
if &j eq 3 then Subpopulation='HCR, two-years longitudinal';
run;
data final_output; set final_output stat_sub&j; where est ne 0; run;
%end;
%mend;
data final_output; set final_output0; run;
%sub_ciclo(1,3);

data silc_it.long_est; set final_output; run;  ****longitudinal
results needed;



*********************************************************************
****;
**This part is for the cross-sectional estimates needed.
All this part has to be run twice. Once for the cross-sectional
estimate 2007 and
the second for the cross-sectional estimate 2008;
*********************************************************************
****;
****************************** HCR P.L.as 60% median, cross
sectional file;
%macro stat1;
data input_bound;set working;wj=w0;run;
```

```
%perc_bound(50);
data working1;merge working output_bound;by country;line=0.60*y_perc;
wj=ws;
if eqinc gt line then z=0;if eqinc le line then z=1;run;
proc univariate data=working1 noprint;output out=est mean=est;var
z;weight wj;by country;run;
data jrr;merge working1 est;by country;y=z-est;subpop_i=1;run;
title 'stat1';
%mend;
**********************************************Perc bound;
%macro perc_bound (perc);
data input;set input_bound;percent=&perc;run;
proc sort data=input;by eqinc;run;
proc iml;
use input;read all var {pid} into pid;read all var {wj} into wj;
read all var {eqinc} into eqinc;read all var {percent} into percent;
read all var {country} into country_vec;
num=nrow(pid); share=repeat(0,num);ratio=percent[1]/100;
tot=sum(wj);
 i=2; do while (i<num+ 1);
      share[i]=sum(wj[1:i])/tot;
        if (share[i-1] < ratio) & (share[i] >= ratio)
      then y_perc=eqinc[i-1]+(eqinc[i]-eqinc[i-1])*((ratio-share[i-
1])/(share[i]-share[i-1]));
  i=i+1;  end;
country=country_vec[1];
create output_bound var {y_perc country};append ;close
output_bound;quit;run;
%mend;
**********************************************Jrr variable;
%macro jrr_var (local);
 %do i=1 %to &local;
proc univariate data=working3 noprint; output out=str_notuse
mean=stratum mean=w_i;
var stratum w_c;where psu eq &i;run;
data str_notuse;set str_notuse;pr=1;run;
data working4;merge working3 str_notuse;by stratum;run;
data working;set working4;where psu ne &i;
if pr eq 1  then w0=w0_old*w_h/(w_h-w_i);else w0=w0_old;
if pr eq 1  then ws=ws_old*w_h/(w_h-w_i);else ws=ws_old; run;
proc univariate data=working4 noprint;output out=info mean=country
mean=ah mean=psu mean=stratum;
var country ah psu stratum;where psu eq &i;run;
%if (&j eq 1) %then %do; %stat1; %end;
proc univariate data=jrr noprint; output out=est mean=est;var z;weight
wj;
where subpop_i eq 1;by country;run;
data est; set est (keep= est country);run;
data replicate; merge info est;by country;run;
data h;set h replicate;run;
%end;
%mend;
************************************Kish;
%macro Kish(local,index);
proc univariate data=kish_input noprint;output out=ymean mean=ybar;var
y;weight wj;by country;run;
proc univariate data=kish_input noprint;output out=wmean mean=wbar;var
wj;by country;run;
data work;merge kish_input ymean wmean;by country;
```

```
if &local eq 1 then zj_2=(y-ybar)**2;else zj_2=1;
wjz_2=(wj/wbar)*zj_2;wj_2z_2=((wj/wbar)**2)*zj_2;run;
proc univariate data=work noprint;output out=sums sum=wjz_2_sum
sum=wj_2z_2_sum n=n;
var wjz_2 wj_2z_2;by country;run;
data kish_output (keep=kish index);set
sums;se_srs=sqrt(wjz_2_sum/(n*(n-1)));index=&j;
se_wt=sqrt(wj_2z_2_sum/(n*(n-1)));kish=se_wt/se_srs;run;
%mend;

*****************************************************;
data r; set silc_it.file07_cs; *silc_it.file08_cs; ***change the
dataset here!!!!;
country=1;
rename rb030=pid;
rename hx090=eqinc;
wt_r=rb050;
I1=1;
run;
data r1 (drop= I1); set r; run;
data h0;input country ah psu stratum stat;cards;
0 0 0 0 0
;run;
data stat0;input est stat_se ;cards;
0 0
;run;
data kish0;input kish index;cards;
0 0
;run;
data final_output0; input subpopulation $1-100 est stat_se  kish ;
cards;
This_is_for_initialising_the_Subpopulation_considered_in_the_analysis_
_____________________________ 0 0 0
;run;
%macro sub_ciclo (sub_ciclo_start,sub_ciclo_end,psu);
%do j=&sub_ciclo_start %to &sub_ciclo_end;
data help; set r (keep=pid I&j);rename I&j=I;run;
proc sort data=r1; by pid; run;
proc sort data=help; by pid; run;
data dati; merge r1 help; by pid; run;
data working_pop_1;set dati;run;
data kish;set kish0;run;
data stat; set stat0; run;
data working_pop0;set working_pop_1;w0=wt_r;w_sub=w0*I;run;
proc univariate data=working_pop0 noprint;output out=sum_w sum=sum_w
sum=sum_w_sub;var w0 w_sub;by country;run;
data sum_w;set sum_w (keep=sum_w sum_w_sub country);run;
data working_pop; merge working_pop0 sum_w;by country;
w0=wt_r/sum_w; ws=w_sub/sum_w_sub;whij=w0; run;
proc sort data=working_pop;by stratum;run;
proc univariate data=working_pop noprint;output out=weight_str
sum=w_h;var w0;by stratum;run;
proc sort data=working_pop;by psu;run;
proc univariate data=working_pop noprint; output out=weight_notuse
sum=w_c;var w0;by psu;run;
proc sort data=working_pop;by stratum;run;
data working2;merge working_pop weight_str;by stratum;run;
proc sort data=working2;by psu;run;
```

```
data working3;merge working2 weight_notuse;by
psu;w0_old=w0;ws_old=ws;run;
data h;set h0;run; %jrr_var(&psu);
data h;set h;where country ne 0;run;
data h_var;set h;run;
data working;set working_pop;run;
%if (&j eq 1) %then %do; %stat1; %end;
%if (&j eq 1)  %then %do;
data kish_input; set jrr;where ws ne 0;wj=ws;run;
%kish(1,&j); %end;
data kish;set kish kish_output;where index gt 0;run;
proc univariate data=jrr noprint; output out=est mean=est n=n;var
z;weight wj;where (I eq 1) and (subpop_i eq 1);by country;run;
data est; set est (keep= est n country);run;
proc univariate data=h noprint; output out=jks mean= ah; var ah; by
stratum; run;
proc sort data=h;by country stratum;run;
proc univariate data=h noprint;output out=jkm sum= yhsum_stat;var
est;by country stratum;run;
proc sort data=jkm; by stratum; run;
proc sort data=jks; by stratum; run;
data jk; merge jkm jks; by stratum; run;
data jk; set jk; yh_stat=yhsum_stat/ah;run;
proc sort data=h; by stratum; run;
proc sort data=jk; by stratum; run;
data prova; merge h jk; by stratum;factor=(ah-1)/ah; run;
data jk2_0;set prova;statdif2_0=(est-yh_stat)**2;run;
proc sort data=jk2_0; by country; run;
proc univariate data=jk2_0 noprint; output out=mean mean=mean; var
statdif2_0; by country; run;
data jk2; merge jk2_0 mean; by
country;statdif2=statdif2_0;limit=6*mean;
if statdif2_0 gt limit then statdif2=limit;run;
proc univariate data=jk2 noprint;  output out=var_stat
sum= stat_v; var statdif2 ; weight factor;by country;run;
data se_stat; set var_stat;stat_se=stat_v**0.5;run;
data stat_act (keep= est stat_se n index);merge est se_stat;by
country;index=&j;
data stat;set stat stat_act;run;
proc freq data=stat_act;table index;run;
 data JRR1;set stat;
where (est ne 0) and (stat_se ne 0);run;
data stat_output0; merge stat kish; by index; run;
data stat_output; set stat_output0;where (est ne 0) and (stat_se ne
0); run;
data stat_sub&j (drop=index); set stat_output;
if &j eq 1 then Subpopulation='HCR     ';
run;
data final_output; set final_output stat_sub&j; where est ne 0 and
stat_se ne 0; run;
%end;
%mend;
data final_output; set final_output0; run;
%sub_ciclo(1,1,1020); ************choose appropriate numbers of PSUs
here (2008:1010, 2007:1020);

******************Save with the proper name of the dataset;
/*data silc_it.results_it07_cs; set final_output; run;*/
data silc_it.results_it08_cs; set final_output; run;
```

```
*************************************************************************
************;
***********RUN AFTER THAT LONGITUDUNAL AND CROSS-SECTIONAL ESTIMATES
HAVE BEEN RUN!!!
This part put together all the results and create the cumulated
estimates;
*************************************************************************
************;
data cs07; set silc_it.results_it07_cs (keep=est stat_se n);
rename est=est1;
rename stat_se=se1;
rename n=n1;
country=1;
run;
data cs08; set silc_it.results_it08_cs (keep=est stat_se n);
rename est=est2;
rename stat_se=se2;
rename n=n2;
country=1;
run;
data long2007 (keep=p1 m1 country); set silc_it.long_est; where
subpopulation eq 'HCR 2007';
rename est=p1;
m1=41489;
country=1;
run;
data long2008 (keep=p2 m2 country); set silc_it.long_est; where
subpopulation eq 'HCR 2008';
rename est=p2;
m2=37731;
country=1;
run;
data persistent_p (keep=a m country); set silc_it.long_est; where
subpopulation eq 'HCR, two-years longitudinal';
rename est=a;
rename n=m;
country=1;
run;
data est_needed; merge cs07 cs08 long2007 long2008 persistent_p; by
country; run;
data cumulated_est; set est_needed;
cumulated_est=(est1+est2)/2;
v=(se1**2+se2**2)/4;
nh=(2*n1*n2)/(n1+n2);
n=(min(of n1 n2))*(0.75*m/(min(of m1 m2)));
b=(a-0.25*(p1+p2)**2)/(0.5*(p1+p2)-0.25*(p1+p2)**2);
cumulated_v=v*(1+b*(n/nh));
cumulated_se=cumulated_v**0.5;
run;
data silc_it.cumulated_est; set cumulated_est; run;
```

## 4. Example of the output produced

| est2 | se2 | n2 | p1 | m1 | p2 | m2 | a | m | cumulated_est | v | nh | n | b | cumulated_v | cumulated_se |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.18668769 | 0.0042495 | 52433 | 0.19583404 | 41489 | 0.18620132 | 37731 | 0.13255242 | 35986 | 0.192477069 | 1.2761E-05 | 52601.9538 | 37506.0415 | 0.6216574 | 0.000018417 | 0.004291503 |

## 5. R code

```
####################
#Functions source
####################
###############f.perc (long)
                f.perc <- function (x,w,level=50) {

                num=length(x);
                share=rep(0,num); ratio=level/100;
                tot=sum (w); i=2;
                while (i<num+1) {
                        share[i]=sum(w[1:i])/tot
                        if (share[i-1]<ratio & share[i] >=ratio)
                        result=x[i-1]+(x[i]-x[i-1])*((ratio-share[i-1])/(share[i]-share[i-1]))
                        i=i+1;}
                return (result)}
###############f.percx (short)
                f.percx<-function(x,w) {
                result=weightedMedian(x,w)
                return(result)}
###############f.stat
                f.stat <- function (x,w) {
                        quantile        = f.perc(x,w)
                line            = 0.6*quantile
                        z               = ifelse (x>line,0,1)
                w.mean          = weighted.mean(z,w)
                y               = z - w.mean
                subpop_i                = 1
                result          = data.frame(line,z,w.mean,y,subpop=1)
                return(w.mean)
                }
###############f.statx
                f.statx <- function (x,w,obj) {
                        quantile        = f.perc(x,w)
                line            = 0.6*quantile
                        obj$z           = ifelse (x>line,0,1)
                obj$w.mean              = weighted.mean(obj$z,w)
                obj$y           = obj$z -  obj$w.mean
                subpop_i                = 1
                result          = obj
                return(result)
                }
###############f.long
                f.long<-function(obj) {
                obj.1           = subset(obj, obj$DB010==2007)
                obj.2           = subset(obj, obj$DB010==2008)
                obj.1$line1             = f.perc(obj.1$equinc,obj.1$w0)*0.6
                obj.2$line2             = f.perc(obj.2$equinc,obj.2$w0)*0.6
                obj.1$pov1              = ifelse(obj.1$line1<obj.1$equinc,0,1)
                obj.2$pov2              = ifelse(obj.2$line2<obj.2$equinc,0,1)
                obj.x           =
merge(obj.1[,c("pid","pov1","w_long")],obj.2[,c("pid","pov2","w_long")],by="pid")
                obj.x$pp                = obj.x$pov1*obj.x$pov2
                persistent              = weighted.mean(obj.x$pp,obj.x$w_long.y)
                return(persistent)
```

```
		}
###############f.JRR
		f.JRR <- function (obj, j.local) {
				f.stat <- function (x,w) {
				quantile	= f.perc(x,w)
				line		= 0.6*quantile
				z		= ifelse (x>line,0,1)
				w.mean		= weighted.mean(z,w)
				y		= z - w.mean
				subpop_i		= 1
				result		= data.frame(line,z,w.mean,y,subpop=1)
				return(w.mean)
				}
			archive = numeric()
		for (i in 1:j.local) {
			ex.psu.wp		= subset(obj, obj$psu==i)
	#ex.psu.wp
			new.country		= mean (ex.psu.wp$country)
			new.ah		= mean (ex.psu.wp$ah)
			new.str		= mean (ex.psu.wp$stratum)
			new.w_i		= mean (ex.psu.wp$w_c)
			small.wp		= subset(obj, obj$psu!=i)
	#small.wp
			#Corrective scaling
			small.wp$w0	= ifelse (small.wp$stratum==new.str,

					small.wp$w0_old*small.wp$w_h/(small.wp$w_h-
new.w_i),
					small.wp$w0_old)

			small.wp$ws	= ifelse (small.wp$stratum==new.str,
					small.wp$ws_old*small.wp$w_h/(small.wp$w_h-
new.w_i),
					small.wp$ws_old)
			info		=
data.frame(country=new.country,ah=new.ah,psu=i,info.str=new.str)
			input_stat		= small.wp[order(small.wp$equinc),]
			w.mean.z		= f.stat(input_stat$equinc ,input_stat$ws)
			info		= data.frame(info,HCR=w.mean.z)
			archive		= rbind(archive,info)
	#archive
			}
		return(archive)
		}
###############f.kish
		f.kish <- function (kish,j.kish) {
		n		= length(kish$y)
		kish$ybar		= weighted.mean(kish$y,kish$wj)
		kish$wbar		= mean(kish$wj)
		if (j.kish==1)	kish$zj_2	= (kish$y-kish$ybar)^2
		if (j.kish!=1)	kish$zj_2	= 1
		kish$wjz_2		= (kish$wj/kish$wbar)*kish$zj_2
		kish$wj_2z_2	= (kish$wj/kish$wbar)^2*kish$zj_2
		wjz_2_sum		= sum (kish$wjz_2)
		wj_2z_2_sum	= sum (kish$wj_2z_2)
		se_srs		= sqrt (wjz_2_sum/(n*(n-1)))
		se_wt		= sqrt (wj_2z_2_sum/(n*(n-1)))
		kish.out		= se_wt/se_srs
```

```
                return(kish.out)
                }
###############f.subciclo1
                f.subciclo1 <- function (obj)  {
        out     =c(0,0,0)
        for ( k in 1:3) {
        if (k==1) obj$I     = ifelse(obj$DB010==2007,1,0)
        if (k==2) obj$I     = ifelse(obj$DB010==2008,1,0)
        if (k==3) obj$I     = 1

        obj$whij  = obj$DB090       /sum(obj$DB090)
        obj$w0    = obj$DB090       /sum(obj$DB090)
        obj$w_sub       = obj$DB090*obj$I/sum(obj$DB090*obj$I)
        obj$w_long      = obj$RB060*obj$I/sum(obj$RB060*obj$I)

        obj     = obj [order(obj$equinc),]
        if (k==1) out[1]    = f.stat( obj$equinc[ obj$DB010==2007],obj$w0[ obj$DB010==2007])
        if (k==2) out[2]    = f.stat( obj$equinc[ obj$DB010==2008],obj$w0[ obj$DB010==2008])
        if (k==3) out[3]    = f.long( obj)
        }
        return (out)
}
###############f.subciclo2
                f.subciclo2 <- function (obj)  {

          obj$whij      = obj$DB090
          obj$w0 = obj$DB090        /sum(obj$DB090)
          obj$ws = obj$DB090*obj$I/sum(obj$DB090*obj$I)

        obj$w_h  = as.numeric(tapply (obj$w0,obj$stratum,sum)[obj$stratum]) #w_h
        obj$w_c  = as.numeric(tapply (obj$w0,obj$psu,sum)[obj$psu]) #w_c

        obj$w0_old      = obj$w0 #w0_old
        obj$ws_old      = obj$ws #ws_old

        local       = length(unique(obj$psu))-sum(is.na(unique(obj$psu))) #(number psu)
        h = h.out=f.JRR(obj, local) #h
        in.stat     = obj[order(obj$equinc),]
        JRR.tot     = f.statx(in.stat$equinc,in.stat$w0,in.stat) #JRR.tot

        kish        = subset(JRR.tot,JRR.tot$ws!=0) #kish_input
        kish$wj     = kish$ws
        kish.out    = f.kish(kish,1)

        EST         = weighted.mean (JRR.tot$z,JRR.tot$ws ) #EST
        jk          = data.frame(info.str=unique (h$info.str)) #jk
        jk$yh.sum       = as.numeric(tapply (h$HCR,h$info.str,sum))
        jk$ah.mean      = as.numeric(tapply (h$ah,h$info.str,mean))
        jk$yh_stat      = jk$yh.sum/jk$ah.mean
        prova       = merge (jk,h, by= "info.str") #prova
        prova$factor        = (prova$ah-1)/prova$ah

        jk2         = prova;
        jk2$statdif2_0= (jk2$HCR-jk2$yh_stat)^2 #jk2
        jk2$mean        = mean(jk2$statdif2_0)
        jk2$statdif2        =jk2$statdif2_0
        jk2$statdif2[jk2$statdif2>6*jk2$mean[1]]=6*jk2$mean[1]
        stat_v      = jk2$statdif2%*%jk2$factor
```

```
        stat_se    = stat_v^0.5
        stat.out   = data.frame(EST,stat_se=stat_se,kish.out) #stat.out

        return (stat.out)
}
#################################################################
#Data preparation
#################################################################
#####################
#Pre-programming
#####################
#rm(list=ls()); output=list()                                          #output list
timex<-proc.time()[3];                                          #time count
require(aroma.light)                                            #Package (weighted median)
#####################
#Import data
#####################
LD08 <- read.csv("/@ HD/silc08l_d08l.csv")                              #D Longitudinal 08
LH08 <- read.csv("/@ HD/silc08l_h08l.csv")                              #H Longitudinal 08
LR08 <- read.csv("/@ HD/silc08l_r08l.csv")                              #R Longitudinal 08
CX07 <- read.csv("/@ HD/file07_cs.csv")                                 #Cross Section 08
CX08 <- read.csv("/@ HD/file08_cs.csv")                                 #Cross Section 08
#####################
#Reduce data
#####################
#d dim(31171     6)
d=LD08                                                          #read d
d=d[,c("DB010","DB020","DB030","DB040","DB090")]                        #select columns
d=subset(d,DB020=="IT" & (DB010==2007 | DB010==2008))                   #select rows
d=subset(d,DB090!=0)                                            #avoid zero weight
d$udb_d=1;                                                      #dummy
#h dim(31171     7)
h=LH08                                                          #read h
h=h[,c("HB010","HB020","HB030","HX040","HX050","HX090")]                #select
columns
names(h)[1:3]<-names(d)[1:3]                                    #standardize names
h=subset(h,DB020=="IT" & (DB010==2007 | DB010==2008))                   #select rows
h$udb_h=1;                                                      #dummy
#r dim(79679     8)
r=LR08                                                          #select r
r=r[,c("RB010","RB020","RB040","RB030","RB060","RB062","RX010")]        #select
columns
names(r)[1:3]<-names(d)[1:3]                                    #standardize names
r=subset(r,DB020=="IT" & (DB010==2007 | DB010==2008))                   #select
columns
r$udb_r=1;                                                      #dummy
#####################
#Merge and exclude doubles
#####################
all=merge(d,h, by=c("DB010","DB020","DB030"))                           #merge d&h
all=subset (all,!is.na(all$udb_h))                              #avoid units in d but not in h
all=merge(all,r,by=c("DB010","DB020","DB030"))                          #merge (d&h)&r
all=all[order(all$DB010,all$DB020,all$RB030),]                  #order for duplicates
all$RB030 = as.integer(all$RB030)                               #avoid rounding up
all$diff=c(-9,diff(all$RB030))                                  #find difference
double=subset (all[,c("DB010","DB020","RB030")], all$diff==0)           #find duplicate
double$double=1                                                 #dummy for duplicate
```

```
all=merge(all,double,by=c("DB010","DB020","RB030"),all.x=TRUE)          #find equivalent
duplicate
no.double=subset (all, is.na(all$double))                               #exclude duplicate and
equivalent
si.double=subset (all, all$double==1)                                   #select duplicate and
equivalent
ok.double=subset (si.double, si.double[,"RB060"]!=0)                    #determine ok
duplicates
nk.double = rbind(no.double,ok.double)                                  #final output
#N.B nk.double == r_long_nodouble in SAS
#####################
#Both years dummy
#####################
r07=nk.double[nk.double$DB010==2008,c("DB020","RB030")]                 #find all units in
2007
r08=nk.double[nk.double$DB010==2007,c("DB020","RB030")]                 #find all units in
2008
r07=r07[order(r07$DB020,r07$RB030),]; r07$r07=1                         #order them
and create dummy
r08=r08[order(r08$DB020,r08$RB030),]; r08$r08=1                         #order them
and create dummy
r78= merge (r07,r08,by=c("DB020","RB030"),all=TRUE)                     #merge for overlap
r78$r78=r78$r07+r78$r08;                                          #dummy overlap
r78=r78[,c("DB020","RB030","r78")]                                      #reduce dataset for
merge
r78$r78=ifelse (is.na(r78$r78),0,1)                                     #codify as dummy
final.data=merge(nk.double,r78,by=c("DB020","RB030"))                   #incorporate
dummy in final
#N.B final.data == silc_it.longitudinal_file in SAS
#################################################################
#Sub-ciclo 1(longitudinal)
#################################################################
rl=subset (final.data, final.data$r78==1); rl$country=1
names(rl)[ which( names(rl) == "RB030" ) ] <- "pid"
names(rl)[ which( names(rl) == "HX090" ) ] <- "equinc"
RESULT01=f.subciclo1(rl)
#N.B Result01 == long.est in SAS
#################################################################
#Sub-ciclo 2 (cross section)
#################################################################
rc = CX07; rc$l=rc$country=1
names(rc)[ which( names(rc) == "RB030" ) ] <- "pid"
names(rc)[ which( names(rc) == "HX090" ) ] <- "equinc"
names(rc)[ which( names(rc) == "RB050" ) ] <- "wt_r"
final.cross07 <- f.subciclo2(rc)
rc = CX08;rc$l=rc$country=1
names(rc)[ which( names(rc) == "RB030" ) ] <- "pid"
names(rc)[ which( names(rc) == "HX090" ) ] <- "equinc"
names(rc)[ which( names(rc) == "RB050" ) ] <- "wt_r"
final.cross08 <- f.subciclo2(rc)
#################################################################
#Conclusion
#################################################################
#N.B.RESULT01   = long_est.sas7bdat
#N.B.final.cross07 = results_it07_cs.sas7bdat
#N.B.final.cross08 = results_it08_cs.sas7bdat
#####################
#Save Synthetic Database (Data preparation)
```

```
#####################
#save(final.data             , file="step2.final.data.RData")
#save(conclusion                      , file="step2.conclusion.RData")
conclusion= rbind(RESULT01, final.cross07, final.cross08)
###################################
#Time check
###################################
tot.time<-diff(c(timex,proc.time()[3]));tot.time
```

**Annex 1**

**Note on constructing the longitudinal dataset**

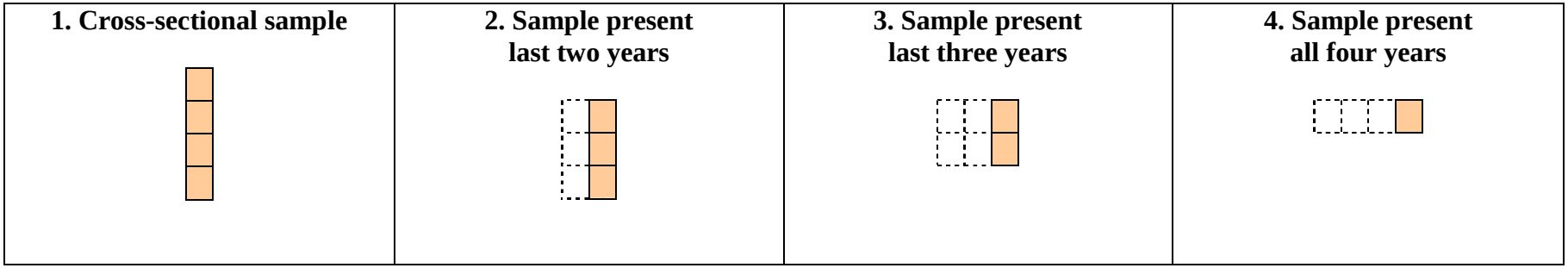
The longitudinal samples are identified on the basis of *continuous presence of individual persons in the survey* for the specified number of most recent years, for two most recent years for the 2-year longitudinal sample; three most recent years for the 3-year longitudinal sample, etc.

An “expansion” of the longitudinal sample base is required to ensure inclusion of *whole* households with all their members, as required for computation and analysis of income variables. The final set of units for inclusion in the computation can be identified in terms of the above as follows.

- Household level variables (H) for the set of households corresponding to each of the longitudinal individual sets as defined above, i.e. for households containing at least one longitudinal person.
- Person-level variables (R) for the set of all persons in each of the above-defined sets of households.
- Adult-level variables (P) for the set of adults in those households; similarly for the subset of variables for selected respondents in those household, if required.

See Figure 1.

**Figure 1. On the construction of the longitudinal files**




Steps:

- From R file identify people present last 2 (3) years = $R^*_{2(3)}$
- Identify households to which these people belong = $H^*_{2(3)}$
- Take all members of these households.

**Note on constructing the longitudinal weights**

Longitudinal weight variables RB062-RB064 provide the basis for use with longitudinal samples of durations, respectively, 2, 3 and 4 years.

The actual weights to be used for various types of variables in the above sets can be constructed as follows for the sample base defined in Section 2.

- To begin with, the weight variable for longitudinal persons is provided by the relevant longitudinal weight RB062 (or RB063, or RB064). In principle this variable should already exist in the UDB, but some procedure has to be used to construct it in cases it is missing in the data. Note that as defined here (and as defined for the EU-SILC weighting procedure) some longitudinal persons may be 'non-sample persons'. [1]

- Also to begin with, the weight variables are taken as zero for persons in the above sample but who lack the required continuous presence of the "true" longitudinal persons for the period concerned. These are persons who live in the same household as one or more longitudinal persons as defined above. Note that some of these persons may be 'sample persons' but who are not longitudinal; others are non-sample persons who are now co-residents with a longitudinal person.

- If remain people that are present 2 (3) years with no longitudinal and base weights, then the average of the weights of the first point, over all persons belonging to the sample base defined in Section 2 above, is assigned to them.

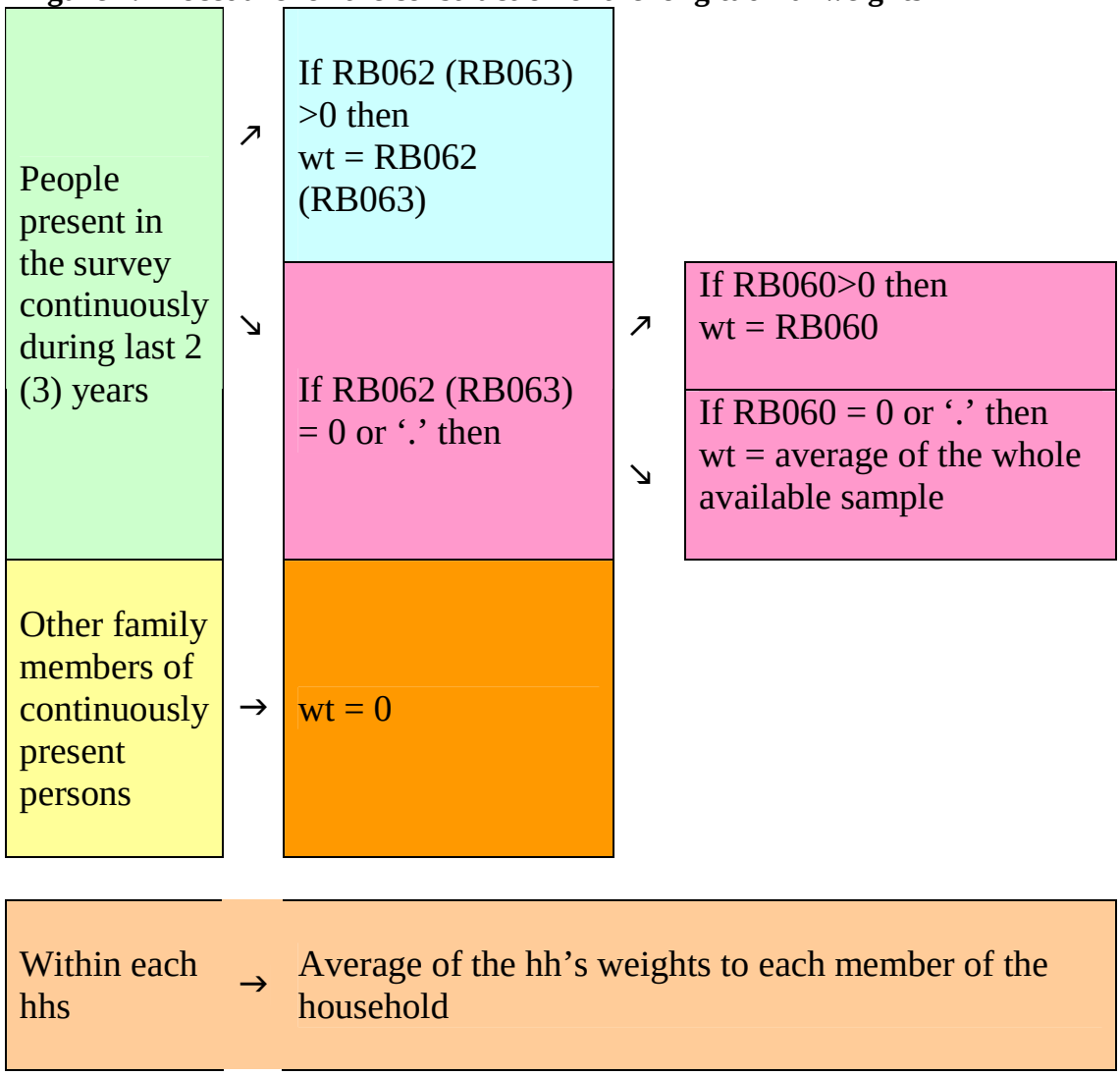
- Finally, this average weight is assigned to every person in the household, replacing the original longitudinal weights. This weight applies unchanged to the person concerned in any personal file such as R or P in which the person appears and also to the household file H.

- The above is done separately for the longitudinal sample of each duration – 2, 3 and 4 years.

See Figure 2.

[1] If the longitudinal weight variables are not yet available in a country data set, we may use the person's most recent (i.e. last wave) *base weight* (RB060) as an approximation.
For the full cross-sectional sample, the appropriate weights to use are the cross-sectional weights DB090, RB050, PB040 or PB060 if applicable, depending on the type of unit being considered.
Also see Figure 2.

**Figure 2. Procedure for the construction of the longitudinal weights**



## Annex 2

## Technical note on JRR procedure for estimating variance and design effect

### Jackknife Repeated Replication (JRR) for variance estimation

The Jackknife Repeated Replication (JRR) is one of a class of methods for estimating sampling errors from comparisons among sample replications which are generated through repeated sampling of the same parent sample. Each replication needs to be a representative sample in itself and to reflect the full complexity of the parent sample.

The basic model of the JRR may be summarised as follows. Consider a design in which two or more primary units have been selected independently from each stratum in the population. Within each primary sampling unit (PSU), sub-sampling of any complexity may be involved, including weighting of the ultimate units. In the standard version, each JRR replication can be formed by eliminating one sample PSU from a particular stratum at a time, and increasing the weights of the remaining sample PSUs in that stratum appropriately so as to obtain an alternative but equally valid estimate to that obtained from the full sample.

Briefly, the standard JRR involves the following.

Let $y$ be a full-sample estimate of any complexity, and $y_{(hi)}$ be the estimate produced using the same procedure after eliminating primary unit i in the stratum h and increasing the weight of the remaining $(a_h - 1)$ units in the stratum by an appropriate factor $g_h$. Let $y_{(h)}$ be the simple average of the $y_{(hi)}$ over $a_h$ values of i in h. The variance of $y$ is then estimated as:

$$\text{var}(y) = \sum_h \left[ (1 - f_h) \cdot \frac{a_h - 1}{a_h} \cdot \sum_i \left( y_{(hi)} - y_{(h)} \right)^2 \right]. \qquad [1]$$

A possible variation may be replacing $y_{(h)}$, the simple average of the $y_{(hi)}$ over the $a_h$ replication created from h, by the full-sample estimate of $y$.

$$\text{var}(y) = \sum_h \left[ (1 - f_h) \cdot \frac{a_h - 1}{a_h} \cdot \sum_i \left( y_{(hi)} - y \right)^2 \right]. \qquad [2]$$

Concerning the re-weighting the of units in a stratum after dropping one unit, normally the factor $g_h$ is taken as:

$$g_h = \frac{a_h}{a_h - 1}. \qquad [3]$$

However, a different form of $g_h$ can be used for practical reasons:

$$g_h = \frac{w_h}{w_h - w_{hi}} \qquad [4]$$

where $w_h = \sum_i w_{hi}$ , $w_{hi} = \sum_j w_{hij}$ , the sum of sample weights of ultimate units j in the primary selection i. This form retains the total weight of the included sample cases unchanged across the replications created – the same total as that for the full-sample. With sample weights scaled such that their sum is equal (or proportional) to some external more reliable population total, population aggregates from the sample can be estimated more effectively, often with the same precision as proportions or means.

**Design effect**

Design effect (deft) is estimated by the ratio of actual standard error (se) of a statistic under the given sample design, to standard error (se_srs) under a simple random sample of same size. The aims of this section is to outline the procedure for estimating design effect, under the JRR approach.

The approach involves decomposition of the design effect into components each of which can be separately estimated. The required components are

(1) the effect of sample weights on variance, and

(2) the effect of clustering stratification and other aspects of the design.

In fact, the identification of the effect of weighting is in itself of substantive interest apart from its usefulness for the above purpose.

A question of great practical interest is the following. How does the weighting affect variances? There are effects in both directions:

(i) Calibration weights and other weighting correlated with the survey variables can reduce, not only bias, but also variances. (Optimal allocation in stratified samples and the corresponding weighting involved is an obvious example.)

(ii) On the other hand, very often weighting is determined on the basis of 'external' factors (e.g., need to over-sample small regions; compensation for high non-response in certain areas due to the performance of particular

interviewers, etc.). Such weighting, essentially uncorrelated with survey variables, results in increased variance.

Generally, the second of the above effects is found to predominate in practice. That is, usually the net effect of weighing is to inflate variances[2].

Effect of clustering, stratification and aspects other than sample weighting

Estimate of variance of a weighted element sample can be obtained by "randomising" the sample and applying the ordinary stratified multistage variance estimation formula to it.

A randomised sample is created from the actual sample by completely randomising the position of individual elements (households, persons) within the sample structure. In principle, this creates a random element sample, which is not subject to clustering or stratification effects, and differs from a true simple random sample simply because of the presence of unequal weights. Random grouping of the elements can be formed to serve as clusters and strata in the variance estimation with affecting the expected results.

The standard error estimated from such a randomised sample using JRR is termed (se_rnd). On this basis, the effect clustering, stratification and any factors other than weighting can be estimated as:

$$\text{Effect of clustering and stratification} = \left( \frac{se}{se\_rnd} \right)$$

The full design effect is obtained by multiplying the above ratio by an estimate of the effect of weighting on the standard error.

Effect of sample weights

The effect of weighting can be estimated as follows under the JRR approach.

For a ratio r=(y/x), with

$y = \Sigma w_i . y_i$ , $x = \Sigma w_i . x_i$ , and $z_i = y_i - r.x_i$

the effect of weighting is given by

$$(Kish\_Jrr)^2 = \left[ \frac{n}{\sum w_i} \right] . \frac{\sum w_i^2 \cdot z_i^2}{\sum w_i \cdot z_i^2}$$

[2] Proper weighting should of course reduce mean squared error, by controlling bias even if there is some increase in variance.

The above has been named "Kish_Jrr" as it is based on the original formulation proposed by Kish (*Survey Sampling*, 1965) an approximate estimate of the effect of essentially "random" weights on variance:

$$(Kish\_Factor)^2 = D_w{}^2 = n \cdot \frac{\sum w_j{}^2}{\left(\sum w_j\right)^2} = \left(\frac{n}{\sum w_j}\right) \cdot \frac{\sum w_j{}^2}{\sum w_j} = 1 + cv^2\left(w_j\right).$$

In previous research it has been empirically demonstrated, at least for a wide variety of measures, the expression "Kish_Jrr" is *also valid for more statistics more complex than simple ratios*, such as various measures of poverty and income inequality.[3]

Design effect

Finally, the design effect is estimated as the product of the components defined above:

$$deft = \left(\frac{se}{se\_rnd}\right) \cdot \left(Kish\_Jrr\right)$$

[3] Any complex statistic may be expressed as a ratio, but involving unknown parameters themselves subject to sampling variability. Application of the above equation amounts to ignoring the variability of the parameters involved.