Sampling and Estimation

Day 5: Resampling Methods for Variance Estimation

Stefan Zins[1] and Matthias Sand[2]

September 8, 2015

[1] Stefan.Zins@gesis.org
[2] Matthias.Sand@gesis.org

Idea: draw repeatedly (sub-)samples from the sample in order to build the sampling distribution of the estimator.

Estimate the variance as variability of the estimates from the resamples.

Methods of interest

- Balanced repeated replication
- Jackknife
- Bootstrap

Some remarks:

- If it works, one does neither need the π_{kl}'s nor the derivatives (i.e. the linearized variable) of the estimator.
  However they are difficult to adapt to complex sampling designs, especially to designs with unequal probability sampling.
- They can be computational intense.

The Balanced Repeated Replication Method

gesis Leibniz Institute for the Social Sciences

Balanced repeated replication (BRR) is (only) suitable for stratified samples. Half samples are build by selecting half the elements from each stratum.

- In the original version of the method there are only two elements per stratum.
- $\hat{\theta}_{(r)}$ is the estimate of the r-th half sample.
- For H strata there are 2^H possible half samples, instead of applying them all we use a balanced selection $R \ll 2^H$ via Hadamard matrices $\mathbf{\Lambda}$.

The Balanced Repeated Replication Method

gesis Leibniz Institute for the Social Sciences

A Hadamar matrices is a square matrix whose entries are either +1 or -1 and whose rows and columns are mutually orthogonal.

$$\mathbf{\Lambda} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

The rows of a $R \times R$ Hadamard matrix denote the half samples and the columns the strata, where $H + 1 \leqslant R \leqslant H + 4$, thus the half samples are drawn mutually independent. λ_{rh} is the element in $\mathbf{\Lambda}$ in the r-th row and the h-column. If $\lambda_{rh} = 1$ the first half of stratum h is in the r-th replicate and for $\lambda_{rh} = -1$ the second.

BRR Replicate Weights I

For $n_h > 2$ one possibility is to divide the PSU's in stratum h randomly into two disjoint sets s_{h_1} and s_{h_2} of sizes $n_{h_1} = \lfloor n_h/2 \rfloor$ and $n_{h_2} = n_h - n_{h_1}$, respectively.

For $n_h > 2$ one possibility is to divide the PSU's in stratum h randomly into two disjoint sets $\mathcal{s}_{h_1}$ and $\mathcal{s}_{h_2}$ of sizes $n_{h_1} = \lfloor n_h/2 \rfloor$ and $n_{h_2} = n_h - n_{h_1}$, respectively. The weights of the r-th replicate have to be adjusted. For $k \in \mathcal{s}_{h_1}$ we have:

$$d_{k\,(r)} := \begin{cases} d_k \left[1 + \left(\dfrac{n_{h_2} \left(1 - \frac{n_h}{N_h} \right)}{n_{h_1}} \right)^{1/2} \right] & \text{if } \lambda_{rh} = 1 \\ d_k \left[1 - \left(\dfrac{n_{h_1} \left(1 - \frac{n_h}{N_h} \right)}{n_{h_2}} \right)^{1/2} \right] & \text{if } \lambda_{rh} = -1 \end{cases} .$$

For $n_h > 2$ one possibility is to divide the PSU's in stratum h randomly into two disjoint sets $\mathcal{s}_{h_1}$ and $\mathcal{s}_{h_2}$ of sizes $n_{h_1} = \lfloor n_h/2 \rfloor$ and $n_{h_2} = n_h - n_{h_1}$, respectively. The weights of the r-th replicate have to be adjusted. For $k \in \mathcal{s}_{h_1}$ we have:

$$d_{k\,(r)} := \begin{cases} d_k \left[1 + \left(\dfrac{n_{h_2} \left(1 - \frac{n_h}{N_h} \right)}{n_{h_1}} \right)^{1/2} \right] & \text{if } \lambda_{rh} = 1 \\ d_k \left[1 - \left(\dfrac{n_{h_1} \left(1 - \frac{n_h}{N_h} \right)}{n_{h_2}} \right)^{1/2} \right] & \text{if } \lambda_{rh} = -1 \end{cases} .$$

The standard BRR variance estimator is given by

$$\widehat{\mathrm{V}} \left(\hat{\theta} \right)_{\mathrm{BRR}} = \frac{1}{R} \sum_{r=1}^{R} \left(\hat{\theta}_{(r)} - \hat{\theta} \right)^2 .$$

BRR Replicate Weights II

There are alternative forms of BRR, that try to cope with the possible instability of the BRR variance estimator. Fay (1989) suggests a scheme which makes the weighting milder by choosing a factor $0 < \epsilon \leqslant 1$ The resulting replicated weight for $k \in s_{h_1}$ is

$$d_{k\,(r)} := \begin{cases} d_k \left[1 + \varepsilon \left(\dfrac{n_{h_2} \cdot \left(1 - \frac{n_h}{N_h} \right)}{n_{h_1}} \right)^{1/2} \right], & \lambda_{rh} = 1, \\ d_k \left[1 - \varepsilon \left(\dfrac{n_{h_1} \cdot \left(1 - \frac{n_h}{N_h} \right)}{n_{h_2}} \right)^{1/2} \right], & \lambda_{rh} = -1. \end{cases} \tag{1}$$

The resulting variance estimator is defined by

$$\hat{\mathrm{V}} \left(\hat{\theta} \right)_{\mathrm{GBRR}} = \frac{1}{R\varepsilon^2} \cdot \sum_{r=1}^{R} \left(\hat{\theta}_{(r),\varepsilon} - \hat{\theta} \right)^2,$$

Setting $\epsilon = 1$ recovers weights from before.

Originally, the Jackknife method was introduced for estimating the bias of an estimator. If $\hat{\theta}$ is our estimator then let

$$\hat{\theta}_{-k}$$

be its corresponding value based on sample $s \backslash \{k\}$.

The replicate values $\hat{\theta}_{(k)} = n\hat{\theta} - (n-1)\hat{\theta}_{-k}$ can be used to estimate the variance of $\hat{\theta}$. This so called (delete-1) jackknife variance estimator is

$$\begin{aligned}\hat{\mathrm{V}}\left(\hat{\theta}\right)_{\mathrm{d1JK}} &= \frac{1}{n(n-1)} \sum_{k \in s} \left(\hat{\theta}_{(k)} - \sum_{l \in s} \frac{\hat{\theta}_{(k)}}{n} \right)^2 \\ &= \frac{n-1}{n} \sum_{k \in s} \left(\hat{\theta}_{-k} - \sum_{l \in s} \frac{\hat{\theta}_{-l}}{n} \right)^2 ,\end{aligned}$$

[Shao and Tu, 1995].

Also the jackknife can be implemented with replicated weights. For a stratified sample if the k-th element in stratum h is omitted, then the weights d_l for element l in stratum g also have to be recalculated:

$$d_{(l,-k)} = \begin{cases} d_l & \text{if } g \neq h \\ d_l \dfrac{n_h}{n_h - 1} & \text{if } g = h,\ k \neq l \\ 0 & \text{if } g = h,\ k = l \end{cases}$$

$d_{(l,-k)}$ is weight of element $l \in \mathcal{U}_g$ if element $k \in \mathcal{U}_h$ is deleted. Then the delete-one jackknife estimate of variance is given by

$$\hat{\mathrm{V}}\left(\hat{\theta}\right)_{\text{d1JK,StrRS}} = \sum_{h=1}^{H} \frac{(n_h - 1)(1 - \frac{n_h}{N_h})}{n_h} \sum_{k \in \mathcal{s}_h} \left(\hat{\theta}_{h,-k} - \sum_{l \in \mathcal{s}_h} \frac{\hat{\theta}_{h,-l}}{n_h} \right)^2 ,$$

where $\hat{\theta}_{h,-k}$ is the value of estimator $\hat{\theta}$ calculated without the $k \in \mathcal{s}_h$ using the adjusted weights.

Advantages and Disadvantages of the Jackknife

gesis Leibniz Institute for the Social Sciences

Very good for *smooth* estimators.

Biased for the variance estimation of quantiles (e.g. median) and similar non-smooth statistics.

Specialized procedures are needed for complex designs, e.g. unequal probability sampling.

Huge effort in case of large samples sizes.

A possible application to non-smooth estimators can be achieved by delete-a-group jackknifes.

Here the sample is divided into G groups, which can be done randomly [Shao and Tu, 1995, p. 195] or systematically [Kott, 2001].

Theoretical (parametric) bootstrap: (Model based) The distribution of the variable of interest is estimated.

Theoretical (parametric) bootstrap: (Model based) The distribution of the variable of interest is estimated.

Monte-Carlo or empirical bootstrap: Random selection of size n with replacement from the original sample.

The Bootstrap Method I

Theoretical (parametric) bootstrap: (Model based) The distribution of the variable of interest is estimated.

Monte-Carlo or empirical bootstrap: Random selection of size n with replacement from the original sample.

Special adaptions are needed in complex surveys

The method may have problem in case of large sampling fractions and sampling without replacement.

If F is the unknown distribution of your variable interest the bootstrap variance estimator results by substituting F in the theoretical formula of the variance define as

$$\mathrm{V}\left(\hat{\theta}\right) = \int \left[\hat{\theta} - \int \hat{\theta} d \prod_{k=1}^{n} F(y_k)\right]^2 d \prod_{k=1}^{n} F\left(y_k\right)$$

by $\hat{F}$:

$$\begin{aligned} \hat{\mathrm{V}}\left(\hat{\theta}\right)_{\text{boot}} &= \int \left[\hat{\theta} - \int \hat{\theta} d \prod_{k=1}^{n} \hat{F}(y_k)\right]^2 d \prod_{k=1}^{n} \hat{F}\left(y_k\right) \\ &= \mathrm{V}\left(\left[\hat{\theta}_{(r)} | \{y_k | k \in \mathcal{s}\}\right]\right) , \end{aligned}$$

where $\hat{\theta}_{(r)}$ is the value of $\hat{\theta}$ based on a sample drawn from $\hat{F}$, denoted as the *bootstrap* sample. $\mathrm{V}\left([.|\{y_k|k \in \mathcal{s}\}]\right)$ is the conditional variance given the original sample data $\{y_k|k \in \mathcal{s}\}$.

The theoretical bootstrap is often not practical to apply and a Monte-Carlo method is used to approximate it. B resamples of size n are drawn by SRSWR from the original sample. Estimator $\hat{\theta}$ is calculated on these resamples. $\hat{\theta}_{(r)}$ denotes the value of $\hat{\theta}$ based on the r-rh resample. Monte Carlo bootstrap variance estimator is given by

$$\hat{\mathrm{V}}\left(\hat{\theta}\right)_{\mathrm{MCboot}} = \frac{1}{B-1}\sum_{r=1}^{B}\left(\hat{\theta}_{(r)} - \frac{1}{B}\sum_{b=1}^{B}\hat{\theta}_{(b)}\right)^2 .$$

Due to the law of large numbers is $\hat{\mathrm{V}}\left(\hat{\theta}\right)_{\mathrm{boot}} = \lim_{B\to\infty}\hat{\mathrm{V}}\left(\hat{\theta}\right)_{\mathrm{MCboot}}$.

Bootstrap Confidence Intervals

Another application of the Monte Carlo bootstrap is to compute confidence intervals.

Another application of the Monte Carlo bootstrap is to compute confidence intervals.

Percentile Bootstrap

$$CI_{1-\alpha} = \left[\hat{\theta}_{r,\alpha/2}\,;\hat{\theta}_{r,1-\alpha/2}\right]$$

where $\hat{\theta}_{r,1-\alpha/2}$ denotes the $1-\alpha/2$ percentile of the bootstrapped coefficients $\mathcal{B} = \{\hat{\theta}_{(r)})|r = 1, , \ldots, B\}$.

Another application of the Monte Carlo bootstrap is to compute confidence intervals.

Percentile Bootstrap

$$CI_{1-\alpha} = \left[\hat{\theta}_{r,\alpha/2}\,;\hat{\theta}_{r,1-\alpha/2}\right]$$

where $\hat{\theta}_{r,1-\alpha/2}$ denotes the $1-\alpha/2$ percentile of the bootstrapped coefficients $\mathcal{B} = \{\hat{\theta}_{(r)})|r = 1, , \ldots, B\}$.

Studentized Bootstrap

$$CI_{1-\alpha} = \left[\hat{\theta} - t_{r,(1-\alpha/2)}\sqrt{\hat{\mathrm{V}}\left(\hat{\theta}\right)_{\mathrm{MCboot}}}; \hat{\theta} - t_{r,(\alpha/2)}\sqrt{\hat{\mathrm{V}}\left(\hat{\theta}\right)_{\mathrm{MCboot}}}\right]$$

where $t_{r,(1-\alpha/2)}$ denotes the $1-\alpha/2$ percentile of the bootstrapped Student's t-test $t_{(r)} = \frac{(\hat{\theta}_{(r)}-\hat{\theta})}{\sqrt{\hat{\mathrm{V}}(\hat{\theta})_{\mathrm{MCboot}}}}$.

Another application of the Monte Carlo bootstrap is to compute confidence intervals.

Percentile Bootstrap

$$CI_{1-\alpha} = \left[\hat{\theta}_{r,\alpha/2}\,;\hat{\theta}_{r,1-\alpha/2}\right]$$

where $\hat{\theta}_{r,1-\alpha/2}$ denotes the $1-\alpha/2$ percentile of the bootstrapped coefficients $\mathcal{B} = \{\hat{\theta}_{(r)})|r = 1, , \ldots, B\}$.

Studentized Bootstrap

$$CI_{1-\alpha} = \left[\hat{\theta} - t_{r,(1-\alpha/2)}\sqrt{\hat{\mathrm{V}}\left(\hat{\theta}\right)_{\text{MCboot}}}; \hat{\theta} - t_{r,(\alpha/2)}\sqrt{\hat{\mathrm{V}}\left(\hat{\theta}\right)_{\text{MCboot}}}\right]$$

where $t_{r,(1-\alpha/2)}$ denotes the $1-\alpha/2$ percentile of the bootstrapped Student's t-test $t_{(r)} = \frac{(\hat{\theta}_{(r)}-\hat{\theta})}{\sqrt{\hat{\mathrm{V}}(\hat{\theta})_{\text{MCboot}}}}$.

The usage of bootstrap confidence intervals is also the reason why usually odd numbers of bootstrap replicates are used, like $B = 49$ or $B = 499$.

For multistage designs the bootstrap is applied at the first stage of the sampling design and when a PSU is drawn with replacement from the original sample all units of the following stages are also included in the bootstrap replicate [Wolter, 2007, p. 211].

A further proposal of a particular bootstrap is the so called rescaling bootstrap. The approach can be described as follows: From the n_{h_l} PSUs of the original sample in the h-th stratum m_{h_l} PSUs are drawn with replacement. The survey weight of SSU k in the i-th PSU in the h-th stratum is for each resample adjusted to:

$$d_{k,(r)} = \left[\left(1 - \left(\frac{m_{h_l}}{n_{h_l} - 1} \right)^{1/2} \right) + \left(\frac{m_{h_l}}{n_{h_l} - 1} \right)^{1/2} \left(\frac{n_{h_l}}{m_{h_l}} \right) r_{hi} \right] d_k,$$

where r_{hi} describes the number of times a certain PSU appears in the resample.

A further proposal of a particular bootstrap is the so called rescaling bootstrap. The approach can be described as follows: From the n_{h_l} PSUs of the original sample in the h-th stratum m_{h_l} PSUs are drawn with replacement. The survey weight of SSU k in the i-th PSU in the h-th stratum is for each resample adjusted to:

$$d_{k,(r)} = \left[\left(1 - \left(\frac{m_{h_l}}{n_{h_l} - 1} \right)^{1/2} \right) + \left(\frac{m_{h_l}}{n_{h_l} - 1} \right)^{1/2} \left(\frac{n_{h_l}}{m_{h_l}} \right) r_{hi} \right] d_k ,$$

where r_{hi} describes the number of times a certain PSU appears in the resample. The value of m_{h_l} has to be determined. A choice with only a little, if any, loss in efficiency is $m_{h_l} = (n_{h_l} - 1)$. Then the calculation of bootstrap weights reduces to:

$$d_{k,(r)} = d_k \frac{n_{h_l}}{(n_{h_l} - 1)} r_{hi} \ .$$

A function to select a simple stratified random sample

```
strat.sample <- function(sind,n.h){
  N.h   <- table(sind)[names(n.h)]
  N     <- length(sind)
  sam   <- mapply(function(x,y)sample(x,y)
                  ,x=N.h,y=n.h
                  ,SIMPLIFY = FALSE)
  sam   <- mapply(function(x,y)x[y]
                  ,x=split(1:N,sind)[names(n.h)],y=sam
                  ,SIMPLIFY = FALSE)
  as.numeric(1:N%in%unlist(sam))
}
```

First we prepare the data set to select a two-stage sample

```
library(survey)
# 1. stage: districts are selected by a simple StrRS; 2. stage: schools are
# selected from sampled PSUs by a SRS

data(api)
## Make a stratification variable for the districts by grouping them by their
## mean value of 'api99'
apipop_d <- apipop[!duplicated(apipop$dnum), ]
api99.d <- tapply(apipop$api99, apipop$dnum, mean)
apipop_d$sind <- cut(api99.d, quantile(api99.d), include.lowest = T)

## make a new data set with design relevant information
apipop. <- merge(apipop, apipop_d[, c("sind", "dnum")], by = "dnum")

# the finite population sizes of the PSU's and SSU's
apipop.$fpc1 <- table(apipop_d$sind)[as.character(apipop.$sind)]
apipop.$fpc2 <- table(apipop.$dnum)[as.character(apipop.$dnum)]

# model for the a later calibration
pop.lm <- lm(api00 ~ sind:(stype + api99) - 1 + sind, data = apipop.)
pop.totals <- colSums(model.matrix(pop.lm))
```

Now we select the sample

```
n_I  <- 40 #select 40 PSU's
#proportional allocation of the 1. stage sample size
n_hI <- round(table(apipop_d$sind)/length(apipop_d$sind)*n_I)
##1. stage sample
s_I  <- strat.sample(sind=apipop_d$sind,n.h=n_hI)==1
#the sample of PSU's
apiclus1.str  <- apipop.[apipop.$dnum%in%apipop_d[s_I,"dnum"],]

#number of sampled SSU's in PSU's
N_i <- n_i <- table(apiclus1.str$dnum)
n_i[N_i<3]   <- N_i[N_i<3] #if less than 3 schools take all
n_i[N_i>=3]  <- 2          #and if their are more sample 2

##2. stage sample
s_i <- strat.sample(apiclus1.str$dnum,n.h=n_i)==1
#the final sample of schools
apiclus2.str <- apipop.[apipop.$snum%in%apiclus1.str[s_i,"snum"],]

#define the 'svydesign' object and calibrate the weights
dclus2.str   <-  svydesign(id = ~dnum + snum, strata=~sind
                           ,fpc = ~fpc1 + fpc2, data = apiclus2.str)
cal.dclus2.str <- calibrate(dclus2.str,stage=0,population = pop.totals
                            ,formula = ~ sind:(stype+api99)+sind-1)
```

Resampling weights based on the our design weights

```
#BRR  weights
dclus2.Brr    <- as.svrepdesign(dclus2.str, type="BRR")
# BRR with Fay's weights
dclus2.FayBrr <- as.svrepdesign(dclus2.str, type="Fay",fay.rho=0.3)
# Jackknife for stratified designs
dclus2.Jkn    <- as.svrepdesign(dclus2.str, type="JKn")

#standard MC boostrap
dclus2.Boot <- as.svrepdesign(dclus2.str, type="bootstrap",replicates=99)

#MC sub-boostrap
dclus2.Subboot <- as.svrepdesign(dclus2.str, type="subbootstrap",replicates=99)

#all off the above resampling methods will produce warnings, because of the
#multistage design

##MC multistage boostrap after Preston (2009).
dclus2.Mrbboot<-as.svrepdesign(dclus2.str, type="mrbbootstrap",replicates=99)

#a matirx of the replicate weights
dclus2.Mrbboot$repweights
```

The variance estimates obtained from the resampling methods and the direct method.

TABLE: Estimated Variances for Different Resampling Methods

Estimator	api00
$\hat{\mathrm{V}}(\overline{y}_\pi)_1$	579.81
$\hat{\mathrm{V}}(\overline{y}_\pi)_{\mathrm{BRR}}$	552.58
$\hat{\mathrm{V}}(\overline{y}_\pi)_{\mathrm{GBRR}}$	465.08
$\hat{\mathrm{V}}(\overline{y}_\pi)_{\mathrm{d1JK,StrRS}}$	634.30
$\hat{\mathrm{V}}(\overline{y}_\pi)_{\mathrm{MCboot}}$	452.41
$\hat{\mathrm{V}}(\overline{y}_\pi)_{\mathrm{subMCboot}}$	682.16
$\hat{\mathrm{V}}(\overline{y}_\pi)_{\mathrm{mrbMCboot}}$	747.59

If you want to use calibrate weights with resampling, then every set of replicate weights (based on the design weights!) has to be calibrated.

```
svymean(~api99,cal.dclus2.str)

##          mean SE
## api99 631.91  0

mrbclus2<-as.svrepdesign(cal.dclus2.str, type="mrb",replicates=99)
#does not re-calibrate after each resample!
svymean(~api99,mrbclus2)

##          mean     SE
## api99 631.91 26.211
```

Run a simulation to see which method is best to estimate the variance.

```
#Be carefull with the numer of runs 'R', this takes a while!
Ests <-
c( "Pest_d" ,"Vest_d" ,"Vest_BRR" ,"Vest_GBRR" ,"Vest_d1JK.StrRS","Vest_MCboot"
   ,"Vest_subMCboot" ,"Vest_mrbMCboot")
R<-1000
simOUT <- vapply(1:R,function(x){
  #1. stage
  s_I           <- strat.sample(sind=apipop_d$sind,n.h=n_hI)==1
  apiclus1.str  <- apipop.[apipop.$dnum%in%apipop_d[s_I,"dnum"],]

  N_i <- n_i    <- table(apiclus1.str$dnum)
  n_i[N_i<3]    <- N_i[N_i<3]
  n_i[N_i>=3]   <- 2

  #2. stage
  s_i <- strat.sample(apiclus1.str$dnum,n.h=n_i)==1
  apiclus2.str <- apipop.[apipop.$snum%in%apiclus1.str[s_i,"snum"],]

  #define the 'svydesign' object
  dclus2.str     <- svydesign( id = ~dnum + snum, strata=~sind,fpc = ~fpc1 + fpc2, data = apiclus2.str)
  dclus2.Brr     <- as.svrepdesign(dclus2.str, type="BRR")
  dclus2.FayBrr  <- as.svrepdesign(dclus2.str, type="Fay",fay.rho=0.3)
  dclus2.Jkn     <- as.svrepdesign(dclus2.str, type="JKn")
  dclus2.Boot    <- as.svrepdesign(dclus2.str, type="bootstrap",replicates=99)
  dclus2.Subboot <- as.svrepdesign(dclus2.str, type="subbootstrap",replicates=99)
  dclus2.Mrbboot <- as.svrepdesign(dclus2.str, type="mrbbootstrap",replicates=99)

  c("Pest_"=svymean(~api00,dclus2.str),"Vest_d"=SE(svymean(~api00,dclus2.str))^2
  ,"Vest_BRR"=SE(svymean(~api00,dclus2.Brr))^2,"Vest_GBRR"=SE(svymean(~api00,dclus2.FayBrr))^2
  ,"Vest_d1JK.StrRS"=SE(svymean(~api00,dclus2.Jkn))^2,"Vest_MCboot"=SE(svymean(~api00,dclus2.Boot))^2
  ,"Vest_subMCboot"=SE(svymean(~api00,dclus2.Subboot))^2,"Vest_mrbMCboot"=SE(svymean(~api00,dclus2.Mrbboot))^2)
 }
 ,FUN.VALUE = array(0,c(1,length(Ests)))
)
dimnames(simOUT) <- list(stat="mean.api00",est=Ests ,sim=1:R)
```

Resampling with Imputation and Calibration

Resampling can be used for variance estimation if a sample has been imputed for item non-response and calibrated as a treatment for unit non-response. The procedure can be described as follows:

Resampling with Imputation and Calibration

gesis Leibniz Institute for the Social Sciences

Resampling can be used for variance estimation if a sample has been imputed for item non-response and calibrated as a treatment for unit non-response. The procedure can be described as follows:

1. resample your imputed sample

Resampling with Imputation and Calibration

gesis Leibniz Institute for the Social Sciences

Resampling can be used for variance estimation if a sample has been imputed for item non-response and calibrated as a treatment for unit non-response. The procedure can be described as follows:

1. resample your imputed sample
2. impute the resample using the same imputation routing that was used to impute the original sample now based on the resample

Resampling with Imputation and Calibration

gesis Leibniz Institute for the Social Sciences

Resampling can be used for variance estimation if a sample has been imputed for item non-response and calibrated as a treatment for unit non-response. The procedure can be described as follows:

1. resample your imputed sample
2. impute the resample using the same imputation routing that was used to impute the original sample now based on the resample
3. calibrate the replicate weights with the same calibration method used to calibrate the original sample

Resampling with Imputation and Calibration

gesis Leibniz Institute for the Social Sciences

Resampling can be used for variance estimation if a sample has been imputed for item non-response and calibrated as a treatment for unit non-response. The procedure can be described as follows:

1. resample your imputed sample
2. impute the resample using the same imputation routing that was used to impute the original sample now based on the resample
3. calibrate the replicate weights with the same calibration method used to calibrate the original sample
4. calculate your estimator

Resampling with Imputation and Calibration

gesis Leibniz Institute for the Social Sciences

Resampling can be used for variance estimation if a sample has been imputed for item non-response and calibrated as a treatment for unit non-response. The procedure can be described as follows:

1. resample your imputed sample
2. impute the resample using the same imputation routing that was used to impute the original sample now based on the resample
3. calibrate the replicate weights with the same calibration method used to calibrate the original sample
4. calculate your estimator
5. repeat this process for each resample

Resampling with Imputation and Calibration

gesis Leibniz Institute for the Social Sciences

Resampling can be used for variance estimation if a sample has been imputed for item non-response and calibrated as a treatment for unit non-response. The procedure can be described as follows:

1. resample your imputed sample
2. impute the resample using the same imputation routing that was used to impute the original sample now based on the resample
3. calibrate the replicate weights with the same calibration method used to calibrate the original sample
4. calculate your estimator
5. repeat this process for each resample

Then you use the variance estimator that is specific to the resampling method you applied.

Literature I


R.E. Fay.
Theory and Application of Replicate Weighting for Variance Calculations.
Survey Research Methods Section: ASA, 1989.


P.S. Kott
The delete-a-group Jackknife.
Journal of Official Statistics, 2001.


J. Shao, D. Tu.
The Jackknife and Bootstrap.
Springer, 1995.


J. Preston.
Rescaled bootstrap for stratified multistage sampling.
Survey Methodology, 2009.