

TUTORIAL: SAMPLING, WEIGHTING AND ESTIMATION DAY 3

Stefan Zins, Matthias Sand
and Jan-Philipp Kolb

GESIS - Leibniz Institute
for the Social Sciences

GESIS Summer School

Cologne, Germany

August 26th, 2015

First order inclusion probability under SRSWOR

$$\pi_k = \frac{n}{N}$$

```
N <- nrow(bm)
n <- 180
pik <- rep(n/N, N)
```

First order inclusion probability under SRSWOR

$$\pi_k = \frac{n}{N}$$

```
N <- nrow(bm)
n <- 180
pik <- rep(n/N,N)
```

First order inclusion probability under StrRS

$$\pi_{hk} = \frac{n_h}{N_h}$$

```
Nh <- table(bm$Province)
nh <- c(15,20,35,22,18,22,7,13,20)
pihk <- data.frame(nh/Nh)
names(pihk)[1] <- "Province"
pihk.long <- merge(pihk,bm,by = "Province")
```

First order inclusion probability under SRSWOR

$$\pi_k = \frac{n}{N}$$

```
N <- nrow(bm)
n <- 180
pik <- rep(n/N,N)
```

First order inclusion probability under StrRS

$$\pi_{hk} = \frac{n_h}{N_h}$$

```
Nh <- table(bm$Province)
nh <- c(15,20,35,22,18,22,7,13,20)
pihk <- data.frame(nh/Nh)
names(pihk)[1] <- "Province"
pihk.long <- merge(pihk,bm,by = "Province")
```

- Or with the [sampling](#) package

```
pihk.sa <- inclusionprobastrata(bm$Province,nh)
```

First order inclusion probability under π_{ps}

$$\pi_k = \frac{x_k}{\sum_{l=1}^N x_l} * n$$

```
phi <- bm$Tot04/sum(bm$Tot04)
```

```
pik <- phi*n
```

```
head(pik)
```

```
[1] 0.2443 7.9021 0.2066 0.2714 0.1776 0.6403
```

First order inclusion probability under π_{ps}

$$\pi_k = \frac{x_k}{\sum_{l=1}^N x_l} * n$$

```
phi <- bm$Tot04/sum(bm$Tot04)
pik <- phi*n
head(pik)
```

```
[1] 0.2443 7.9021 0.2066 0.2714 0.1776 0.6403
```

- ⇒ Sampling frequencies > 1
- ⇒ With replacement → average frequency of element k in sample
- ⇒ Without replacement → each element can only be chosen once

π ps and the `sampling` package

```
pik <- inclusionprobabilities(bm$Tot04,n)
head(pik)
```

```
[1] 0.2885 1.0000 0.2439 0.3205 0.2097 0.7561
```

```
table(pik == 1)
```

```
FALSE  TRUE
  562    27
```

π ps and the `sampling` package

```
pik <- inclusionprobabilities(bm$Tot04,n)
head(pik)
```

```
[1] 0.2885 1.0000 0.2439 0.3205 0.2097 0.7561
```

```
table(pik == 1)
```

```
FALSE  TRUE
  562    27
```

⇒ The command `inclusionprobabilites` displays the first order inclusion probability → not the average sampling frequencies

⇒ It sets every frequency > 1 to 1 and redistributes the "rest", because:

π ps and the `sampling` package

```
pik <- inclusionprobabilities(bm$Tot04,n)
head(pik)
```

```
[1] 0.2885 1.0000 0.2439 0.3205 0.2097 0.7561
```

```
table(pik == 1)
```

```
FALSE  TRUE
  562    27
```

- ⇒ The command `inclusionprobabilites` displays the first order inclusion probability → not the average sampling frequencies
- ⇒ It sets every frequency > 1 to 1 and redistributes the "rest", because:
- ⇒ $0 < \pi_k \leq 1$

The Horvitz-Thompson-Estimator

$$\hat{t}_{HT} = \sum_{k=1}^n d_k * y_k = \sum_{k=1}^n \frac{1}{\pi_k} y_k$$

```
s <- strata(bm,"Province",nh, "srswor")  
samp <- getdata(bm,s)  
dk <- 1/samp$Prob  
sum(dk*samp$Men04)  
  
[1] 5681894
```

The Horvitz-Thompson-Estimator

$$\hat{t}_{HT} = \sum_{k=1}^n d_k * y_k = \sum_{k=1}^n \frac{1}{\pi_k} y_k$$

```
s <- strata(bm,"Province",nh, "srswor")  
samp <- getdata(bm,s)  
dk <- 1/samp$Prob  
sum(dk*samp$Men04)
```

```
[1] 5681894
```

```
sum(bm$Men04)
```

```
[1] 5097709
```

The **sampling** package also offers a function for the HT-estimator

```
HTstrata(y = samp$Men04,pik = samp$Prob,strata = samp$Province)
```

```
[1] 5681894
```

Scaling design weights to sample size

- Typically the sum of design weights adds up to the population size ($\sum_{k=1}^n d_k = N$)
 - It is common to use scaled weights (d_k^*) that add up to the sample size and have a mean of 1 ($\sum_{k=1}^n d_k^* = n$; $d_k^* = n * \frac{d_k}{\sum_{l=1}^n d_l}$)
- ⇒ Advantage: you see if an element has a higher or lower probability to be included in your sample

```
dk.sc <- nrow(samp)*dk/sum(dk)  
sum(dk.sc*samp$Men04)/nrow(samp)
```

```
[1] 9647
```

Truncating weights

- Truncating the weights within specific borders to avoid high and/ or negative weights
- MSE argument: trade-off between biased results and lower variance
- Algorithm should rescale the weights to its previous sum

```
trunc.bounds<-function(di,bound){
+   n<-sum(di)
+   nopt<-di
+   i<-0
+   s<-which(di<=0|di<bound[1]|di>bound[2])
+   while(i<n){
+     if(length(s)!=0){
+       s1<-which(nopt<=0)
+       s2<-which(nopt<bound[1])
+       s3<-which(nopt>bound[2])
+       nopt[s1]<-bound[1]
+       nopt[s2]<-bound[1]
+       nopt[s3]<-bound[2]
+       su<-length(s1)*bound[1]+length(s2)*bound[1]+length(s3)*bound[2]
+       ge<-(n-su)*nopt[-s]/sum(nopt[-s])
+       nopt[-s]<-ge
+       s<-which(nopt<=0|nopt<bound[1]|nopt>bound[2])}
+     if(length(s)!=0){
+       i<- i+1
+       fi<-i}
+     else {
+       fi<-i+1
+       i<-n
+     }
+   }
+   cat(" number of iterations ",fi,"\n",
+       "number of truncated weights ",length(which(nopt%in%bound)), "\n",
+       "minimal value",sum(di^2/nopt))
+   return(nopt)
+ }
```

Truncating weights (example)

```
wei <- runif(5000,0.5,7)  
table(wei>=6)
```

```
FALSE  TRUE  
4223    777
```

```
bounds <- c(0,6)  
wei.trunc <- trunc.bounds(wei,bounds)
```

```
number of iterations  18  
number of truncated weights  925  
minimal value 18908
```

```
table(wei.trunc==6)
```

```
FALSE  TRUE  
4075    925
```

Variance of the HT-Estimator

$$V_{SYG}(\hat{t}_{HT}) = \sum_{\substack{k=1 \\ k < l}}^N \sum_{l=1}^N (\pi_k * \pi_l - \pi_{kl}) * \left(\frac{y_k}{\pi_k} - \frac{y_l}{\pi_l} \right)^2$$

```
bm <- bm[-2,]  
pik <- inclusionprobabilities(bm$Tot03,30)  
IPk11 <- UPsampfordpi2(pik)  
IPk12 <- UPsystematicpi2(pik)  
SIGMA.samp <- IPk11 - pik%*%t(pik)  
SIGMA.syst <- IPk12 - pik%*%t(pik)
```


Variance of the HT-Estimator

```
var.HT.tot.samp <- t(bm$Tot04/pik
+                    )%*%SIGMA.samp%*(bm$Tot04/pik)
var.HT.tot.sys<- t(bm$Tot04/pik
+                  )%*%SIGMA.syst%*(bm$Tot04/pik)
var.HT.tot.samp

[,1]
[1,] 92926611

var.HT.tot.sys

[,1]
[1,] 113132108
```

Variance of the HT-Estimator

```
var.HT.tot.samp <- t(bm$Tot04/pik
+                    )%*%SIGMA.samp%*(bm$Tot04/pik)
var.HT.tot.sys<- t(bm$Tot04/pik
+                  )%*%SIGMA.syst%*(bm$Tot04/pik)
var.HT.tot.samp

[,1]
[1,] 92926611

var.HT.tot.sys

[,1]
[1,] 113132108
```

- Same design weights
 - Different sampling algorithm
- ⇒ Different Variance!

The HT-Estimator: Variance Estimation

$$\hat{V}_{SYG}(\hat{t}_{HT}) = \sum_{\substack{k=1 \\ k < l}}^n \sum_{l=1}^n \frac{\pi_k * \pi_l - \pi_{kl}}{\pi_{kl}} * \left(\frac{y_k}{\pi_k} - \frac{y_l}{\pi_l} \right)^2$$

```
s <- UPsystematic(pik)
samp <- getdata(bm,s)
SIGMA.s <- SIGMA.syst[s==1,s==1]
SIGMA.s.tilde <- SIGMA.s/IPk12[s==1,s==1]
var.hat.HT.tot.syst <- t(samp$Tot04/pik[s==1]
+                         )%*%SIGMA.s.tilde%*%(samp$Tot04/pik[s==1])
var.hat.HT.tot.syst

      [,1]
[1,] 7.403e+13
```

Design-based approach

$$deff = \frac{Var_c(\hat{t})}{Var_{srs}(\hat{t})}$$

```
var.tot.srs <- var(bm$Tot04)/30*(1-30/nrow(bm))*nrow(bm)^2  
deff <- var.HT.tot.samp/var.tot.srs  
deff
```

```
      [,1]  
[1,] 1.881e-05
```

- Although the sample has been drawn with unequal probabilities, the design effect is below 1
- ⇒ Highly correlated variable has been used to calculate the inclusion probabilities

Model-based approach

$$\hat{deff} = \hat{deff}_p * \hat{deff}_c = n \frac{\sum_{h=1}^I d_h^2 n_h}{(\sum_{h=1}^I d_h n_h)^2} * (1 + (\bar{b} - 1)\rho)$$

n_h is the number of units per cluster; \bar{b} is the average cluster size; ρ reflects the Intraclass Correlation Coefficient (ICC)

Model-based approach

$$\hat{deff} = \hat{deff}_p * \hat{deff}_c = n \frac{\sum_{h=1}^I d_h^2 n_h}{(\sum_{h=1}^I d_h n_h)^2} * (1 + (\bar{b} - 1)\rho)$$

n_h is the number of units per cluster; \bar{b} is the average cluster size; ρ reflects the Intraclass Correlation Coefficient (ICC)

⇒ No cluster/ stratified random sample

⇒ $deff_p$ captures the design effect due to unequal inclusion probabilities

```
deff_p <- sum(30*(1/pik[s==1])^2)/sum(1/pik[s==1])^2
deff_p
```

```
[1] 2.258
```

Model-based approach

$$\hat{deff} = \hat{deff}_p * \hat{deff}_c = n \frac{\sum_{h=1}^I d_h^2 n_h}{(\sum_{h=1}^I d_h n_h)^2} * (1 + (\bar{b} - 1)\rho)$$

n_h is the number of units per cluster; \bar{b} is the average cluster size; ρ reflects the Intraclass Correlation Coefficient (ICC)

⇒ No cluster/ stratified random sample

⇒ $deff_p$ captures the design effect due to unequal inclusion probabilities

```
deff_p <- sum(30*(1/pik[s==1])^2)/sum(1/pik[s==1])^2
deff_p
```

```
[1] 2.258
```

Effective sample size

$$n_{eff} = \frac{n}{deff}$$

```
30/deff
```

```
[,1]
```

```
[1,] 1594905
```

Poststratification

Using the my.pop data set of day 1

```
s <- srswor(100,10000)
samp <- my.pop[s==1,]
genXedu.s <- data.frame(table(samp$gender,samp$education))
genXedu.s[,3] <- genXedu.s[,3]/sum(genXedu.s[,3])
genXedu.pop <- data.frame(table(my.pop$gender,my.pop$education))
genXedu.pop[,3] <- genXedu.pop[,3]/sum(genXedu.pop[,3])
adj.w <- data.frame(genXedu.pop[,3]/genXedu.s[,3])
adj.w[,1]
```

```
[1] 0.7626 1.2745 0.8618 0.7132 2.0489 0.8720 1.8350 1.2467
```

```
samp$to.merge <- paste(samp$gender,samp$education)
adj.w$to.merge <- paste(genXedu.s$Var1,genXedu.s$Var2)
adjusted <- merge(adj.w,samp,by="to.merge")
```


Raking/ Iterative Proportional Fitting

Generating the necessary data frames

```
age <- rep(as.character(1:6),times=10)
edu <- rep(as.character(1:5),each=6,times=2)
gender <- rep(c("m","w"),each=30)
freq <- sample(100,60,replace=T)
# Synthetic sample distribution
samp <- data.frame(age,edu,gender,freq)
samp[,4] <- samp[,4]/sum(samp[,4])
freq2 <- sample(100,60,replace=T)
# in population
master <- data.frame(age,edu,gender,freq2)
master[,4] <- master[,4]/sum(master[,4])
masageXedu <- aggregate(master[,4],
+                         list(age=master[,1],edu=master[,2]),sum)
masageXgen <- aggregate(master[,4],
+                         list(age=master[,1],gender=master[,3]),sum)
maseduXgen <- aggregate(master[,4],
+                         list(edu=master[,2],gender=master[,3]),sum)
```

Raking/ Iterative Proportional Fitting

```
w_0 <- rep(1,times=nrow(samp))
times <- 0
while(times <= 1000){
+   #AgexEducation
+   saxe <- aggregate(samp[,4]*w_0,list(age=samp[,1],edu=samp[,2]),sum)
+   w_1 <- masageXedu[,3]/saxe[,3]
+   w_1 <- rep(w_1,times=(nrow(samp)/nrow(saxe)))
+   #AgexGender
+   saxg <- aggregate(samp[,4]*w_0*w_1,list(age=samp[,1],gen=samp[,3]),sum)
+   w_2 <- masageXgen[,3]/saxg[,3]
+   w_2 <- c(rep(w_2[1:(length(w_2)/2)],times=(nrow(samp)/nrow(saxg))),
+           rep(w_2[(length(w_2)/2+1):length(w_2)],times=(nrow(samp)/nrow(saxg))))
+   #EducationxGender
+   sexg <- aggregate(samp[,4]*w_0*w_1*w_2,list(edu=samp[,2],gen=samp[,3]),sum)
+   w_3 <- maseduXgen[,3]/sexg[,3]
+   w_3 <- rep(w_3,each=(nrow(samp)/nrow(sexg)))
+   #w4
+   w_4 <- w_0*w_1*w_2*w_3
+   if(max(abs(w_0-w_4))>0.05)
+       {w_0<-w_4
+         times<-times+1}
+   else {break}
+   cat("iteration",times,"\n")
+ }

iteration 1
iteration 2
iteration 3
iteration 4
iteration 5

samp$weight<-w_0*samp[,4]
```

Raking/ Iterative Proportional Fitting

```
sampeduXgen<-aggregate(samp[,5],list(edu=samp[,2],gen=samp[,3]),sum)
```

sampeduXgen

	edu	gen	x
1	1	m	0.09474
2	2	m	0.09211
3	3	m	0.09836
4	4	m	0.09309
5	5	m	0.10395
6	1	w	0.10000
7	2	w	0.10724
8	3	w	0.12204
9	4	w	0.10395
10	5	w	0.08454

maseduXgen

	edu	gender	x
1	1	m	0.09474
2	2	m	0.09211
3	3	m	0.09836
4	4	m	0.09309
5	5	m	0.10395
6	1	w	0.10000
7	2	w	0.10724
8	3	w	0.12204
9	4	w	0.10395
10	5	w	0.08454

- The survey package provides a large range of applications for complex survey samples
- Typically, the first step is to define a survey object with the `svydesign()` command

Simple survey object (single stage)

```
library(Matrix)
library(survey)
bm$pik1 <- inclusionprobabilities(bm$Tot03,100)
s <- UPmaxentropy(bm$pik1)
samp <- getdata(bm,s)
IPk1 <- UPmaxentropypi2(bm$pik1)
surv.obj <- svydesign(id=~1,fpc = samp$pik1,
+                   data = samp,pps = ppsmat(IPk1[s==1,s==1]),
+                   variance = "YG")
```

```
surv.obj <- svydesign(id=~1,fpc = samp$pk1,  
+                   data = samp,pps = ppsmat(IPk1[s==1,s==1]),  
+                   variance = "YG")
```

- `id` specifies the identifier of PSU and SSU; `id= ~0` or `id=~1` stipulates a single stage sampling
 - For multi-stage samples the `id` argument should always specify a formula with the cluster-identifier at each stage
 - `fpc` should be used for the finite population correction
- ⇒ Either as the total population size of each stratum or as a fraction of the total population that has been sampled

```
surv.obj <- svydesign(id=~1,fpc = samp$pik,  
+                   data = samp,pps = ppsmat(IPk1[s==1,s==1]),  
+                   variance = "YG")
```

- `data` reflects the data set for which the design object should be defined
 - `pps` should be used to define the design information that should be used; usually the second order probability of inclusion
- ⇒ `ppsmat()` is a wrapper for the joint inclusion probabilities of the HT-Estimator
- With `variance` you can specify whether you use the Yates-Grundy- or the HT-Estimator
- ⇒ For fixed sample sizes, "YG" should be used

Important commands

<code>svytotal</code>	returns the estimated total of a variable and its standard error (+ <i>deff</i>)
<code>svymean</code>	returns the estimated mean of a variable and its standard error (+ <i>deff</i>)
<code>svyquantile</code>	Computes quantiles for data from complex surveys
<code>svyvar</code>	Computes variances for data from complex surveys
<code>weights</code>	Returns the (design) weights of a survey object
<code>calibrate</code>	Calibration of a data set (uses the GREG-Estimator)
...	...

```
svytotal(~Tot04,surv.obj)
```

```
      total    SE  
Tot04 9957554 3770
```

Example: changing the "YG" argument with fixed sample sizes

```
surv.obj2 <- svydesign(id=~1,fpc = samp$pik,  
+                   data = samp,pps = ppsmat(IPk1[s==1,s==1])  
+                   ,variance = "HT")  
svytotal(~Tot04,surv.obj2)
```

total SE

Tot04 9957554 NA

Example: changing the "YG" argument with fixed sample sizes

```
surv.obj2 <- svydesign(id=~1,fpc = samp$pik,  
+                   data = samp,pps = ppsmat(IPk1[s==1,s==1])  
+                   ,variance = "HT")  
svytotal(~Tot04,surv.obj2)
```

```
total SE  
Tot04 9957554 NA
```

⇒ The variance estimator under HT varies more than the one of the YG estimator

Example: changing the "YG" argument with fixed sample sizes

```
surv.obj2 <- svydesign(id=~1,fpc = samp$pik,  
+                   data = samp,pps = ppsmat(IPk1[s==1,s==1])  
+                   ,variance = "HT")  
svytotal(~Tot04,surv.obj2)
```

total SE

Tot04 9957554 NA

- ⇒ The variance estimator under HT varies more than the one of the YG estimator
- ⇒ In this case the estimator is negative

Calibrating the sample

The GREG-Estimator within the survey package

```
svymean(~averageincome, surv.obj)
```

```
      mean  SE  
averageincome 25515 455
```

```
mean(bm$averageincome)
```

```
[1] 25095
```

- Using the variable Men03 and Arrondiss as auxiliary information
- Calculating the population Total

```
lm1 <- lm(averageincome ~Men03+Arrondiss, data=bm)  
pop.tot <- colSums(model.matrix(lm1))
```

Calibrating the sample

- Calculation the calibration weights with the `calibrate()` command

```
surv.obj3 <- svydesign(id=~0,fpc = samp$pik1,  
+                     data = samp,pps = "brewer")
```

- The `calibrate` function cannot be applied to the previous object of class "ppsmat"
- ⇒ Therefore we use the brewer approximation for joint inclusion probabilities

Calibrating the sample

- Calculation the calibration weights with the `calibrate()` command

```
surv.obj3 <- svydesign(id=~0,fpc = samp$pik1,  
+                    data = samp,pps = "brewer")
```

- The `calibrate` function cannot be applied to the previous object of class "ppsmat"
- ⇒ Therefore we use the brewer approximation for joint inclusion probabilities
- ⇒ Seems odd, since only first order inclusion probabilities are used for calibration

Calibrating the sample

```
g_i <- calibrate(surv.obj3, formula = ~Men03+Arrondiss  
+               , population=pop.tot, calfun="linear")
```

- The names of the variables that are used for the calibration have to be identical for your survey object and your population
- `formula` specifies the calibration model
- With `calfun` you can choose between a linear model (GREG) or a raking approach
- For only one calibration variable, `calibrate` produces the same weights as a poststratification

Calibration results

SAMPLE

```
svymean(~averageincome,g_i)

              mean  SE
averageincome 25185 391

svytotal(~Men03,g_i)

              total SE
Men03 4853501    0

svytotal(~Arrondiss,g_i)

              total SE
Arrondiss 27744    0
```

POPULATION

```
mean(bm$averageincome)

[1] 25095

sum(bm$Men03)

[1] 4853501

sum(bm$Arrondiss)

[1] 27744
```

Multi-stage samples

Loading the `api` data set and generating a *multi-stage* data frame

```
data(api)
score <- by(apiclus1, apiclus1$cds,
+           function(x) rnorm(x$api.stu, mean = x$api00, sd = sqrt(x$api00))
```

- the `by` command is similar to `tapply` and creates a list of normal distributed test scores for each school

```
l <- 50
nh <- 60
apiclus1$fpc <- inclusionprobabilities(apiclus1$enroll, l)
```

- We draw a sample of $l = 50$ schools (PSUs) proportional to the number of enrolled pupils

Sampling the cluster

```
cs <- UPmaxentropy(apiclus1$fpc)
cs.dat <- apiclus1[cs==1,]
```

Sampling within a school

```
score.cs.dat <- score[as.character(cs.dat$cds)]
score.samp <- lapply(score.cs.dat,function(x)x[sample(length(x),nh)])
names(score.samp) <- names(score.cs.dat)
data.s <- data.frame(score = unlist(score.samp),
+                   cds=rep(names(score.samp),
+                   times=sapply(score.samp,length)))
```

Merging the data sets/ Second stage inclusion probabilities

```
DATA.s <- merge(cs.dat,data.s,by="cds")  
DATA.s$id <- 1:nrow(DATA.s)  
DATA.s$fpc2 <- nh/DATA.s$enroll
```

- ⇒ Full sample of 50 PSUs with 60 SSUs each
- ⇒ *Self-weighting* approach

Merging the data sets/ Second stage inclusion probabilities

```
DATA.s <- merge(cs.dat, data.s, by="cds")
DATA.s$id <- 1:nrow(DATA.s)
DATA.s$fpc2 <- nh/DATA.s$enroll
```

- ⇒ Full sample of 50 PSUs with 60 SSUs each
- ⇒ *Self-weighting* approach

Specifying the survey object

```
mul.surv <- svydesign(id=~cds+id, fpc = ~fpc+fpc2,
+                   data=DATA.s, pps="brewer")
svymean(~api00, mul.surv)
```

```
      mean      SE
api00  625 12.4
```

```
table(weights(mul.surv))
```

```
33.3914893617021
      2820
```

```
35.1 35.7833333333333
      60      60
```

```
36.35
      60
```

DESIGN EFFECTS/ ONE-STAGE CLUSTER SAMPLE

- 1 Draw a sample of $I = 20$ FULL arondissements proportional to the population of 2003
 \Rightarrow Use the maximum entropy algorithm
- 2 Calculate the design effect by the model based approach

MODEL BASED APPROACH

$$\hat{deff} = \hat{deff}_p * \hat{deff}_c = n \frac{\sum_{h=1}^I d_h^2 n_h}{(\sum_{h=1}^I d_h n_h)^2} * (1 + (\bar{b} - 1)\rho)$$

$$\hat{\rho}^{AOV} = \frac{MSB - MSW}{MSB + (K - 1)MSW}$$

$$MSB = \frac{SSB}{I - 1}; \quad MSW = \frac{SSW}{n - I}; \quad K = \frac{1}{I - 1} \left(n - \sum_{h=1}^I \frac{n_h^2}{n} \right)$$

DESIGN EFFECTS/ ONE-STAGE CLUSTER SAMPLE

R-CODE MSW/ MSB

```
### MSW and MSB  
SS <- anova(lm(DATA.c$Tot04~DATA.c$Arrondiss))  
MSB <- SS$`Mean Sq`[1]  
MSW <- SS$`Mean Sq`[2]
```

- 3 Calculate deff under the design based approach
- 4 Calculate the effective sample size

THE SURVEY PACKAGE/ MULTI-STAGE SAMPLING

- 1 Create the normal distributed income for each individual in 2004 within each commune of the belgianmunicipalities data set (use the square root of the average income as the standard deviation)
- 2 Draw a sample of $I = 50$ communes (use the INS as identifier) proportional to the population of 2003
- 3 Draw a sample of $n_h = 80$ individuals within each commune
- 4 Estimate the mean income of your sample (design weighted)
- 5 Poststratify your data set to the number of men in 2003