Public Use Files for EU-SILC

Matthias Templ
Statistics Austria & TU WIEN

Lydia Spies
DeStatis

Maxime Bergeaut
INSEE

Abstract

The simulation of synthetic EU-SILC data using synthetic data simulation methods are described and the corresponding code is embedded in this paper. Basically, the newest version of the R package **simPop** is used for simulating EU-SILC synthetic data. TODO: A wrapper function tailored specifically towards EU-SILC data for convenience and ease of use is presented, as well as detailed instructions for performing each of the four involved data generation steps separately. In addition, the generation of diagnostic plots for the simulated population data is illustrated and the data utility is discussed.

Keywords: R, synthetic data, simulation, survey statistics, EU-SILC.

1. Introduction

This contribution shows how the creation of synthetic EU-SILC data. The data simulation framework consists of four steps:

1. Setup of the household structure
2. Simulation of categorical variables
3. Simulation of (semi-)continuous variables
4. Splitting (semi-)continuous variables into components

Note that this paper does not motivate, describe or evaluate the statistical methodology of the framework. Instead it is focused on the R code to generate synthetic population data and produce diagnostic plots. For details on the statistical methodology, the reader is referred to Alfons, Kraft, Templ, and Filzmoser (2011).

The *European Union Statistics on Income and Living Conditions* (EU-SILC) is panel survey conducted in European countries and serves as data basis for the estimation social inclusion indicators in Europe. EU-SILC data are highly complex and contain detailed information on the income of the sampled individuals and households. More information on EU-SILC can be found in Eurostat (2004).

In Alfons *et al.* (2011), three methods for the simulation of the net income of the individuals in the population are proposed and analyzed:

MP Multinomial logistic regression models with random draws from the resulting categories. For the categories corresponding to the upper tail, the values are drawn from a (truncated) generalized Pareto distribution, for the other categories from a uniform distribution.

Table 1: Variables of the EU-SILC example data TODO: UPDATE THIS INFORMATION.

Variable	Name	Type	
Region	`db040`	Categorical	9 levels
Household size	`hsize`	Categorical	9 levels
Age	`age`	Categorical	
Gender	`rb090`	Categorical	2 levels
Economic status	`pl030`	Categorical	7 levels
Citizenship	`pb220a`	Categorical	3 levels
Personal net income	`netIncome`	Semi-continuous	
Employee cash or near cash income	`py010n`	Semi-continuous	
Cash benefits or losses from self-employment	`py050n`	Semi-continuous	
Unemployment benefits	`py090n`	Semi-continuous	
Old-age benefits	`py100n`	Semi-continuous	
Survivor's benefits	`py110n`	Semi-continuous	
Sickness benefits	`py120n`	Semi-continuous	
Disability benefits	`py130n`	Semi-continuous	
Education-related allowances	`py140n`	Semi-continuous	
Household sample weights	`db090`	Continuous	
Personal sample weights	`rb050`	Continuous	

TR Two-step regression models with trimming and random draws from the residuals.

TN Two-step regression models with trimming and random draws from a normal distribution.

The first two steps of the analysis, namely the simulation of the household structure and additional categorical variables, are performed in exactly the same manner for the three scenarios. While the simulation of the income components is carried out with the same parameter settings, the results of course depend on the simulated net income.

It is important to note that the original Austrian EU-SILC sample provided by Statistics Austria and used in Alfons *et al.* (2011) is confidential, hence the results presented there cannot be 1:1 reproduced in this vignette. Nevertheless, the code for such an analysis is presented here using the example data from the package, which has been synthetically generated itself. In fact, this example data set is a sample drawn from one of the populations generated in Alfons *et al.* (2011). However, the sample weights have been modified such that the size of the resulting populations is about 1% of the real Austrian population in order to keep the computation time low. Table 1 lists the variables of the example data used in the code examples.

With the following commands, the package and the example data are loaded. Furthermore, the numeric value stored in `seed` will be used as seed for the random number generator in the examples to make the results reproducible.

2. Data preprocessing

First ensure that your current working directory in R is set correctly. My path is e.g.

```
setwd("~/workspace/sga_sdc/silcPop")
```

Please change the path accordingly.

2.1. Read data and select variables

The data provided by the SILC group at Statistics Austria contains the merged Austrian SILC information in SPSS. To read the data it is convienient to install package **haven** first. Please set the file path accordingly.

```
library(haven)
getwd()

## [1] "/data/home/templ/workspace/sga_sdc/silcPop"

x <- read_spss("../../new_workfile.sav")
# x <- data.frame(read.spss("../../new_workfile.sav"))
```

Note that in some countries the data set is stored in four files, the personal, the household, the childrens and an register file. Please merge the latter three files to the personal file using the ID's in the data set.

Here is a code to see how this might work when you have separated files:

```
## please use appropriate names for the csv files
suf.d <- read.csv2(file = "children.csv", header =TRUE,sep=",")
suf.h <- read.csv2(file = "households.csv", header =TRUE,sep=",")
suf.p <- read.csv2(file = "persons.csv", header =TRUE,sep=",")
suf.r <- read.csv2(file = "register.csv", header =TRUE,sep=",")
## merge data (small modifications might be needed (lower/upper case, ...))
suf.pers <- merge(x=suf.r, y=suf.p, by.x="RB030", by.y="PB030", all.x=TRUE)
suf.pers$RB030 <- suf.pers$PB030
suf.hh <- merge(x=suf.d, y=suf.h, by.x="DB030", by.y="HB030", all.x=TRUE)
suf.hh$HB030 <- suf.hh$DB030
suf <- merge(x=suf.pers, y=suf.hh, by.x="RX030", by.y="HB030")
names(suf) <- tolower(names(suf))
names(suf.d) <- tolower(names(suf.d))
names(suf.h) <- tolower(names(suf.h))
names(suf.p) <- tolower(names(suf.p))
names(suf.r) <- tolower(names(suf.r))
```

Some member states have capitals in the variable names, others have lower case. To ensure lower case, type

```
names(x) <- tolower(names(x))
```

We extract the most important variables.

```
eusilc13 <- data.frame("db030"=x$db030, # HHid
                       "db040"=x$db040, # region
                       "rb030"=x$rb030, # Personid
                       "rb080"=x$rb080, # year of birth
                       "rb090"=factor(x$rb090), # sex
                       "pl031"=factor(x$pl031), # economic status
                       "pb220a"=factor(x$pb220a), # citizenship
                       ## personal income components:
                       "py010g"=x$py010g,
                       #"py020g"=x£py020g,
                       "py021g"=x$py021g,
                       "py050g"=x$py050g,
                       "py080g"=x$py080g,
                       "py090g"=x$py090g,
                       "py100g"=x$py100g,
                       "py110g"=x$py110g,
                       "py120g"=x$py120g,
                       "py130g"=x$py130g,
                       "py140g"=x$py140g,
                       ## household income components:
                       "hy040g"=x$hy040g,
                       "hy050g"=x$hy050g,
                       "hy060g"=x$hy060g,
                       "hy070g"=x$hy070g,
                       "hy080g"=x$hy080g,
                       #"hy081g"=x£hy081g,
                       "hy090g"=x$hy090g,
                       "hy100g"=x$hy100g,
                       "hy110g"=x$hy110g,
                       "hy120g"=x$hy120g,
                       "hy130g"=x$hy130g,
                       "hy140g"=x$hy140g,
                       #"hy170g"=x£hy170g,
                       "db090"=x$db090,  # household cross-sectional weight
                       "rb050"=x$rb050,  # personal cross-sectional weight
                       "pb190"=factor(x$pb190),  # maritial status
                       "pe040"=factor(x$pe040),  # highest isced level attained
                       "pl051"=factor(x$pl051),  # occupation
                       "pl111"=factor(x$pl111),  # NACE
                       "rb010"=factor(x$rb010),   # year, and now rest
                       "rb250"=factor(x$rb250),
                       "p119000"=factor(x$p119000),
                       "p038003f"=factor(x$p038003f),
                       "p118000i"=factor(x$p118000i),
                       "aktivi"=factor(x$aktivi),
                       "erwintensneu"=factor(x$erwintensneu)
)
```

For Austria, rename levels of region (`db040`):

```
eusilc13$db040 <- factor(eusilc13$db040,
                         labels= c("Burgenland", "Lower Austria", "Vienna",
```

```
                                "Carinthia", "Styria", "Upper Austria",
                                "Salzburg", "Tyrol", "Vorarlberg"),
                                                ordered = FALSE)
```

Just for a clearer view, arrange the data set according to `db030` (household ID)

```
eusilc13 <- eusilc13[order(eusilc13$db030),]
```

2.2. Adding some useful variables

Various useful helper functions are written and included in simPop. However, since they are only suited and specific for EU-SILC, they are not documented in the package. However, the application of these utility functions can be seen in the following.

In the EU-SILC data set, only the birth is given. We add the age of the persons

```
eusilc13$age <- getAge(data=eusilc13)
```

The gender is coded with 1's and 2's, we change it to *female* and *male* using the following helper function.

```
eusilc13$rb090 <- getGender(data=eusilc13)
```

The household size is only indirectly included in EU-SILC (throught the household ID), but we calculate and add it

```
eusilc13$hsize <- getHsize(data=eusilc13)
```

We slightly modify `pl031` (economic status)

```
eusilc13$pl031 <- getEcoStat(data=eusilc13,levels=levels(eusilc13$pl031))
```

Variable `pb220a` (citizenship) is too sparse. We make broader categories. Note that this is country-specific.

```
## modify pb220a
owncountry <- "AT"
EU <- c("BE","BG","CY","CZ","DE","DK","EL","ES","FI","FR","HU",
        "IT","LT","LU","LV","MT","NL","PL","PT","RO","SI","SE","SK","UK")
other <- c("CAN","CH","CSA","IS","MK","NAF","NME","NO",
           "OAF","OAS","OCE","OEU","OT","OTH","TR","USA","WAF")
x$pb220a <- factor(eusilc13$pb220a)
eusilc13$pb220a <- getCitizenship(data=eusilc13, owncountry=owncountry,
                                  EU=EU, other=other)
```

The household ID's has to be restructured.

```
## restructure HHID
eusilc13$db030 <- restructureHHid(eusilc13)
```

The NACE code should be a factor, as well as `hsize`

```
eusilc13$pl111 <- as.factor(eusilc13$pl111)
eusilc13$hsize <- as.factor(eusilc13$hsize)
```

It has 42 categories. We will reduce it to the main NACE categories.

```
levels(eusilc13$pl111) <-  c(rep("a", 3), rep("b-e", 35), rep("f", 3), rep("g", 3),
                             rep("h", 5), rep("i", 2), rep("j", 6), rep("k", 3),
                             rep("l-n", 15), "o", "p", rep("q", 3), rep("r-u", 10))
## only one-digit codes for pl051
eusilc13$pl051 <- as.numeric(as.character(eusilc13$pl051))
eusilc13$pl051 <- as.factor(trunc(eusilc13$pl051/10))
```

For simulation purposes, we need the summarized personal gross income as well as the summarized household incomes. Better to include the gross income variables instead of the net income variables. The equivalised disposable income is computed from the gross income variables. The net income variables are not compulsory and therefore some countries do not transmit them.

```
require(simPop)
eusilc13$pgrossIncome <- apply(eusilc13[, c("py010g","py021g","py050g","py080g",
                                            "py090g","py100g", "py110g","py120g",
                                            "py130g","py140g"
                                            )], 1, sum, na.rm=TRUE)
eusilc13$hgrossIncome <- apply(eusilc13[, c("hy040g","hy050g","hy060g","hy070g","hy080g",
                                            "hy090g","hy100g","hy110g")], 1, sum,
                               na.rm=TRUE)
eusilc13$hgrossminus <- apply(eusilc13[, c("hy120g","hy130g","hy140g")], 1, sum,
                              na.rm=TRUE)
```

For latter simulation use, we need the categorized incomes as well. Afterwards resampling of fractions conditional on gross income category and economic status is done. Therefore, the gross income categories need to be constructed first. With the function `getBreaks()`, default breakpoints based on quantiles are computed. In this example, the argument `upper` is set to `Inf` to avoid problems with different maximum values in the three synthetic populations, and the argument `equidist` is set to `FALSE` such that non-equidistant probabilities as described in Alfons *et al.* (2011) are used for the calculation of the quantiles.

```
breaks <- getBreaks(eusilc13$pgrossIncome, eusilc13$rb050,
    upper = Inf, equidist = FALSE)
eusilc13$pgrossIncomeCat <- getCat(eusilc13$pgrossIncome, breaks)
breaks <- getBreaks(eusilc13$hgrossIncome, eusilc13$rb050,
    upper = Inf, equidist = FALSE)
eusilc13$hgrossIncomeCat <- getCat(eusilc13$hgrossIncome, breaks)
breaks <- getBreaks(eusilc13$hgrossminus, eusilc13$rb050,
                    upper = Inf, equidist = FALSE)
eusilc13$hgrossminusCat <- getCat(eusilc13$hgrossminus, breaks)
```

Our data set is now prepared to start for the generation of synthetic data. The basic structure is

```
str(eusilc13)

## 'data.frame': 13250 obs. of 49 variables:
## $ db030 : num 1 1 2 2 2 2 2 3 3 3 ...
## $ db040 : Factor w/ 9 levels "Burgenland","Lower Austria",..: 6 6 4
4 4 4 4 3 3 3 ...
## $ rb030 : num 101 102 201 202 203 204 205 301 302 303 ...
## $ rb080 : num 1990 1933 1969 1974 1995 ...
## $ rb090 : Factor w/ 2 levels "male","female": 1 2 2 1 2 2 1 1 2 2
...
## $ pl031 : Factor w/ 11 levels "1","2","3","4",..: 1 7 7 5 5 NA NA 1
1 6 ...
## $ pb220a : Factor w/ 32 levels "","AT","BE","BG",..: 2 2 2 2 2 1 1
14 3 14 ...
## $ py010g : num 13556 0 0 0 1473 ...
## $ py021g : num 0 0 0 0 0 NA NA 0 0 0 ...
## $ py050g : num 0 0 0 0 0 NA NA 0 0 0 ...
## $ py080g : num 0 0 0 0 0 NA NA 0 0 0 ...
## $ py090g : num 0 0 0 0 8380 ...
## $ py100g : num 0 5284 0 0 0 ...
## $ py110g : num 0 4265 0 0 0 ...
## $ py120g : num 0 0 0 0 40 NA NA 0 0 0 ...
## $ py130g : num 0 0 22384 0 0 ...
## $ py140g : num 0 0 0 0 0 NA NA 0 0 0 ...
## $ hy040g : num 0 0 0 0 0 0 0 0 0 0 ...
## $ hy050g : num 0 0 9248 9248 9248 ...
## $ hy060g : num 0 0 0 0 0 0 0 0 0 0 ...
## $ hy070g : num 0 0 632 632 632 632 632 0 0 0 ...
## $ hy080g : num 0 0 0 0 0 0 0 0 0 0 ...
## $ hy090g : num 117 117 120 120 120 ...
## $ hy100g : num 0 0 0 0 0 0 0 0 0 0 ...
## $ hy110g : num 0 0 4164 4164 4164 ...
## $ hy120g : num 0 0 0 0 0 0 0 0 0 0 ...
## $ hy130g : num 0 0 0 0 0 0 0 720 720 720 ...
## $ hy140g : num 2364 2364 333 333 333 ...
## $ db090 : num 379 379 1051 1051 1051 ...
## $ rb050 : num 379 379 1051 1051 1051 ...
## $ pb190 : Factor w/ 5 levels "1","2","3","4",..: 1 4 2 2 1 NA NA 1 1
1 ...
## $ pe040 : Factor w/ 7 levels "0","1","2","3",..: 4 3 3 4 3 NA NA 6 4
4 ...
## $ pl051 : Factor w/ 10 levels "0","1","2","3",..: 8 7 10 8 6 NA NA 3
4 NA ...
## $ pl111 : Factor w/ 13 levels "a","b-e","f",..: 2 NA NA NA NA NA NA
7 2 NA ...
## $ rb010 : Factor w/ 1 level "2013": 1 1 1 1 1 1 1 1 1 1 ...
## $ rb250 : Factor w/ 2 levels "11","14": 1 1 1 1 1 NA NA 1 1 1 ...
## $ p119000 : Factor w/ 51 levels "-1","8","10",..: 12 7 7 11 8 NA NA
23 11 10 ...
```

```
## $ p038003f : Factor w/ 4 levels "-2","-1","1",..: 2 1 1 1 3 NA NA 3
3 1 ...
## $ p118000i : Factor w/ 11 levels "0","1","2","3",..: 3 2 NA NA NA NA
NA 10 7 2 ...
## $ aktivi : Factor w/ 13 levels "0","1","2","3",..: 7 1 NA NA NA NA
NA 13 13 1 ...
## $ erwintensneu : Factor w/ 4 levels "-1","1","2","3": 4 1 2 2 2 2 2
4 4 4 ...
## $ age : num 22 79 43 38 17 15 13 49 47 18 ...
## $ hsize : Factor w/ 8 levels "1","2","3","4",..: 2 2 5 5 5 5 5 3 3 3
...
## $ pgrossIncome : num 13556 9549 22384 0 9893 ...
## $ hgrossIncome : num 117 117 14164 14164 14164 ...
## $ hgrossminus : num 2364 2364 333 333 333 ...
## $ pgrossIncomeCat: Factor w/ 12 levels "0","(0,463]",..: 6 5 7 1 5 1
1 11 9 3 ...
## $ hgrossIncomeCat: Factor w/ 12 levels "0","(0,13.3]",..: 5 5 10 10
10 10 10 7 7 7 ...
## $ hgrossminusCat : Factor w/ 22 levels "[-1.87e+03,-1.57e+03)",..:
15 15 13 13 13 13 13 21 21 21 ...
```

3. Simulation of the population

The variable names of the example data set are used as default values in the function arguments. It would therefore mostly not be necessary to write them explicitely in the function calls, but in order to demonstrate how these arguments are used, the names of the involved variables are always supplied in the commands called.

First, it is important to check the platform since of parallel computing tasks – Windows cannot fork, so if Windows we use only one CPU, otherwise the number of CPU's are set to maximum of CPU's on your computer minus one.

```
## Number of CPU's is set to NULL
##  which means that maximum number of CPU's minus one is used
##  On my machine, this is 15
```

The first step of the analysis is to set up the basic household structure using the function `simStructure()`. Note that a variable named `"hsize"` giving the household sizes is generated automatically in this example, but the name of the corresponding variable in the sample data can also be specified as an argument. Furthermore, the argument `additional` specifies the variables that define the household structure in addition to the household size (in this case age and gender).

In case of large populations (like Germany or France), the division of the household weight (=`db090`) by a factor, e.g. 10, is needed as long you have too less memory on your computer. A factor of 10 would mean that a population of size $N/10$, with N the number of inhabitants in the corresponding country are simulated.

```
eusilc13$db090 <- eusilc13$db090/10
```

```
## [1] "/data/home/templ/workspace/sga_sdc/silcPop/eusilc13.RData"
```

3.1. Defining important variables for the simulation

We specify the input, an provide information on household ID, household size, strata and sampling weights.

```
require(simPop)
inp <- specifyInput(data=eusilc13,
                    hhid="db030",
                    hhsize="hsize",
                    strata="db040",
                    weight="db090")
inp

##
##  --------------
## survey sample of size 13250 x 50
##
##  Selected important variables:
##
##  household ID: db030
##  personal ID: pid
##  variable household size: hsize
##  sampling weight: db090
##  strata: db040
##  --------------
```

3.2. The basic household structure

The basic household strucure have to be defined:

```
eusilcP <- simStructure(inp, method="direct",
                        basicHHvars=c("age", "rb090") )
eusilcP

##
## --------------
## synthetic population  of size
##  836859 x 7
##
## build from a sample of size
## 13250 x 50
## --------------
##
## variables in the population:
## db030,hsize,db040,age,rb090,pid,weight
```

3.3. Simulation of categorical variables

Additional categorical variables are then simulated using the function `simCategorical()`. The argument `additional` specifies the variables to be simulated in this step (economic status and citizenship). Function argument `regModel` is used to specify the predictors and/or the regression model. Possible values are `basic` (only the basic structural variables are used as a predictor), `available` (all available predictors are used) or an object of class `formula` (specifying a formula). We choose `available`.

```
eusilcP <- simCategorical(eusilcP,
                          additional = c("pl031", "pb220a", "pb190",
                                         "pe040", "pl051", "pl111"),
                          regModel = rep("available", 6),
                          nr_cpus=cpus, MaxNWts=5000 )

## Simulating variable 'pl031'.
## we are running the following multinom-model:
## suppressWarnings(multinom(pl031 ~ age + rb090, weights=db090,
data=dataSample, trace=FALSE, maxit=500, MaxNWts=5000))
## Simulating variable 'pb220a'.
## we are running the following multinom-model:
## suppressWarnings(multinom(pb220a ~ age + rb090 + pl031,
weights=db090, data=dataSample, trace=FALSE, maxit=500, MaxNWts=5000))
## Simulating variable 'pb190'.
## we are running the following multinom-model:
## suppressWarnings(multinom(pb190 ~ age + rb090 + pl031 + pb220a,
weights=db090, data=dataSample, trace=FALSE, maxit=500, MaxNWts=5000))
## Simulating variable 'pe040'.
## we are running the following multinom-model:
## suppressWarnings(multinom(pe040 ~ age + rb090 + pl031 + pb220a +
pb190, weights=db090, data=dataSample, trace=FALSE, maxit=500,
MaxNWts=5000))
## Simulating variable 'pl051'.
## we are running the following multinom-model:
## suppressWarnings(multinom(pl051 ~ age + rb090 + pl031 + pb220a +
pb190 + pe040, weights=db090, data=dataSample, trace=FALSE, maxit=500,
MaxNWts=5000))
## Simulating variable 'pl111'.
## we are running the following multinom-model:
## suppressWarnings(multinom(pl111 ~ age + rb090 + pl031 + pb220a +
pb190 + pe040 + pl051, weights=db090, data=dataSample, trace=FALSE,
maxit=500, MaxNWts=5000))

eusilcP

##
## --------------
## synthetic population of size
## 836859 x 13
##
## build from a sample of size
## 13250 x 50
```

```
## --------------
##
## variables in the population:
##
db030,hsize,db040,age,rb090,pid,weight,pl031,pb220a,pb190,pe040,pl051,pl111
```

3.4. Simulation of continuous personal variables

Next, the function `simContinuous()` is used to simulate the net income according to a formula on predictors.

```
eusilcP <- simContinuous(eusilcP, additional = "pgrossIncome",
                         upper = 200000, equidist = TRUE, zeros=TRUE,
                         nr_cpus=cpus, MaxNWts=2000,
                         regModel = formula(~ age + pl031 + rb090 + pb220a) )

## running multinom with the following model:
## multinom(pgrossIncomeCat ~ age + pl031 + rb090 + pb220a,
weights=db090, data=dataSample, trace=FALSE, maxit=maxit,
MaxNWts=MaxNWts)
```

The gross income is now splitted to income components.

```
pcomponents <- c("py010g","py021g","py050g","py080g",
                 "py090g","py100g", "py110g","py120g",
                 "py130g","py140g")
eusilcP <- simComponents(eusilcP,
                          total = "pgrossIncome", components = pcomponents,
                          conditional = c("pl031"))
```

3.5. Simulation of continuous household income

We first simulate the totals

```
eusilcP <- simContinuous(eusilcP, additional = "hgrossIncome",
                         upper = 200000, equidist = TRUE, nr_cpus=cpus, MaxNWts=2000,
                         regModel = formula(~ db040 + hsize + pl031),
                         byHousehold = TRUE)

## running multinom with the following model:
## multinom(hgrossIncomeCat ~ db040 + hsize + pl031, weights=db090,
data=dataSample, trace=FALSE, maxit=maxit, MaxNWts=MaxNWts)

eusilcP <- simContinuous(eusilcP, additional = "hgrossminus",
                         upper = 200000, equidist = TRUE, nr_cpus=cpus, MaxNWts=2000,
                         regModel = formula(~ db040 + hsize + pl031),
                         byHousehold = TRUE)

## running multinom with the following model:
## multinom(hgrossminusCat ~ db040 + hsize + pl031, weights=db090,
data=dataSample, trace=FALSE, maxit=maxit, MaxNWts=MaxNWts)
```

and split to components

```
hcomponents <- c("hy040g","hy050g","hy060g","hy070g","hy080g",
                 "hy090g","hy100g","hy110g")
eusilcP <- simComponents(eusilcP,
                         total = "hgrossIncome", components = hcomponents,
                         conditional = c("hsize"))
hminuscomponents <- c("hy120g","hy130g","hy140g")
eusilcP <- simComponents(eusilcP,
                         total = "hgrossminus", components = hminuscomponents,
                         conditional = c("hsize"))
```

4. Extract final data

The final object has the following variables included

```
eusilcP


##
## --------------
## synthetic population of size
## 836859 x 40
##
## build from a sample of size
## 13250 x 50
## --------------
##
## variables in the population:
##
db030,hsize,db040,age,rb090,pid,weight,pl031,pb220a,pb190,pe040,pl051,pl111,pgrossIncomeCat,pgrossI
```

Extract the population

TODO: option to extract all variables without specification of every variable name.

```
vars <- c("db030","hsize","db040","age","rb090","pid","weight","pl031",
          "pb220a","pb190","pe040","pl051","pl111","pgrossIncomeCat",
          "pgrossIncome","py010g","py021g","py050g","py080g","py090g",
          "py100g","py110g","py120g","py130g","py140g","hgrossIncomeCat",
          "hgrossIncome","hgrossminusCat","hgrossminus","hy040g","hy050g",
          "hy060g","hy070g","hy080g","hy090g","hy100g","hy110g","hy120g",
          "hy130g","hy140g")
## extract population
pufpop <- pop(eusilcP, var=vars)
```

Draw a sample with the same number of households as in the original data set.

```
require(simFrame)
## table of number of households per region
tab <- as.numeric(table(eusilc13[!duplicated(eusilc13$db030), "db040"]))
```

```
## stratified group sampling, equal size
set.seed(23456)
puf <- draw(data.frame(pufpop),
             design = "db040",
             grouping = "db030",
             size = tab)
dim(puf)

## [1] 13513    41

colnames(puf)[which(colnames(puf) == ".weight")] <- "rb050"
puf$rb050 <- puf$rb050*10
puf$pb040 <- puf$rb050

puf$hb030 <- puf$db030
puf$px030 <- puf$db030
puf$rx030 <- puf$db030
puf$pb030 <- puf$pid
puf$rb030 <- puf$pid
puf$hx040 <- puf$hsize
puf$pb150 <- puf$rb090

## ---- age variables:
puf$rx020 <- as.numeric(as.character(puf$age))
puf$px020 <- as.numeric(as.character(puf$age))

## WHY THIS?
# ## ---- age difference:
# tab <- tableWt(suf[rx010-suf[rx020, weights=suf[,"rb050"])
# tab <- wtd.table(suf[rx010-suf[rx020, weights=suf[,"rb050"],type="table")
# p <- tab/sum(suf[,"rb050"])
# age.dif <- sample(x=c(0,1),size=dim(puf)[1],prob=p,replace=T)
# puf[rx010 <- puf[rx020 + age.dif

## ---- equivalized household size:
puf$hx050 <- eqSS("db030", "rx020", data=puf)
```

4.1. Create rest of variables using synthetic reconstruction

The rest of the variables can be preferable simulated using function `simCategorical` and `simContinuous`. However, to avoid for overfitting, the regression model should be specified. Expemlarely this is done for three variables:

```
eusilcP <-  simCategorical(eusilcP,
                 additional = c("rb250", "p119000", "p038003f",
                                "p118000i", "aktivi", "erwintensneu"),
                 regModel = rep("basic", 6),
                 nr_cpus=cpus )

## Simulating variable 'rb250'.
```

```
## we are running the following multinom-model:
## suppressWarnings(multinom(rb250 ~ age + rb090, weights=db090,
data=dataSample, trace=FALSE, maxit=500, MaxNWts=1500))
## Simulating variable 'p119000'.
## we are running the following multinom-model:
## suppressWarnings(multinom(p119000 ~ age + rb090, weights=db090,
data=dataSample, trace=FALSE, maxit=500, MaxNWts=1500))
## Simulating variable 'p038003f'.
## we are running the following multinom-model:
## suppressWarnings(multinom(p038003f ~ age + rb090, weights=db090,
data=dataSample, trace=FALSE, maxit=500, MaxNWts=1500))
## Simulating variable 'p118000i'.
## we are running the following multinom-model:
## suppressWarnings(multinom(p118000i ~ age + rb090, weights=db090,
data=dataSample, trace=FALSE, maxit=500, MaxNWts=1500))
## Simulating variable 'aktivi'.
## we are running the following multinom-model:
## suppressWarnings(multinom(aktivi ~ age + rb090, weights=db090,
data=dataSample, trace=FALSE, maxit=500, MaxNWts=1500))
## Simulating variable 'erwintensneu'.
## we are running the following multinom-model:
## suppressWarnings(multinom(erwintensneu ~ age + rb090, weights=db090,
data=dataSample, trace=FALSE, maxit=500, MaxNWts=1500))
```

References

Alfons A, Kraft S, Templ M, Filzmoser P (2011). "Simulation of close-to-reality population data for household surveys with application to EU-SILC." *Statistical Methods & Applications*, **20**(3), 383–407.

Eurostat (2004). "Description of Target Variables: Cross-sectional and Longitudinal." *EU-SILC 065/04*, Eurostat, Luxembourg.

Affiliation:

Matthias Templ
CSTAT - Computational Statistics
Institute of Statistics Mathematical Methods in Economics
Vienna University of Technology
Wiedner Hauptstr. 8-10
1040 Vienna, Austria
Tel. +43 1 58801 10715
e-mail: matthias.templ@tuwien.ac.at
&
Statistics Austria
Guglgasse 13

1110 Vienna, Austria
Lydia Spies

Destatis
Maxime Bergeaut

INSEE