

Simulation Tools for Small Area Estimation: Introducing the R-package **saeSim**

Sebastian Warnholz

Freie Universität Berlin

Timo Schmid

Freie Universität Berlin

Abstract

In this article the R-package **saeSim** is introduced. The package provides a simulation framework for studies in the field of small area estimation. As the demand for reliable regional estimates from sample surveys has been substantially grown, small area estimation provides statistical methods to produce such reliable predictions when the sample sizes in regions are small. Model- and design-based simulations are used to introduce new methods to the field. In this article a reproducible research approach towards the publication of simulation studies alongside articles and during research is promoted. The package aims to assist the researcher during the research process and as a vessel to publish new tools and studies.

Keywords: package, R, reproducible research, simulation, small area estimation.

1. Introduction

The demand for reliable small area statistics from sample surveys has been substantially grown over the last decades due to their use in public and private sectors. In this paper we present a framework for simulation studies inside the field of small area estimation. This tool might be useful for the prospective researcher or data analyst to provide reproducible research.

Reproducible research has become a widely discussed topic in many academic areas. In the field of statistics many mostly open-source tools like the R-language (R Core Team 2014) and L^AT_EX, dynamic reporting packages like knitr (Yihui 2013), sweave (Leisch 2002) and more recently rmarkdown (Allaire, McPherson, Xie, Wickham, Cheng, and Allen 2014), make the integration of text and source code for statistical analysis possible. Publishing source code and data alongside research results draws special attention to authoring the analysis. However, the requirements for source code are different from the written words in the article itself.

Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do. (Knuth 1992, p.99)

Next to the combination of text and source code, reproducible research aims that the full output of the academic research which is the paper combined with the full computational

environment like data and source code, is available. However, real data is often very sensitive and governed by strict confidentiality rules. Synthetic data generation mechanisms (Kolb 2013) can be used to provide safe data which is publicly available to enable the community to reproduce the analysis and results. Burgard, Kolb, and Münnich (2014) interpreted this as an open research philosophy. Such synthetic data sets can be used to test newly proposed statistical methods in a close-to-reality framework/ simulation. This may provide valuable insights about the quality of the introduced methods in a controlled environment under different scenarios. In general, statistical simulation studies are divided into two concepts:

- Design-based: The simulation study is based on true or synthetic data of a fixed population. Then, samples are selected repeatedly from the underlying finite population and different estimation methods are applied in each replication. The obtained estimates are compared to the true values of the population, for instance, in terms of relative bias (RB) or relative root mean squared error (RRMSE).
- Model-based: The simulation study uses data drawn from certain distributions. In each iteration, the population is generated from a model and a sample is selected according to a specific sampling scheme. The sample is used to estimate the quantity of interest and quality measures (like RB and RRMSE) are derived.

Further discussion regarding model- and design-based simulations is available in Münnich, Schürle, Bihler, Boonstra, Knotterus, Nieuwenbroek, Haslinger, Laaksonen, Eckmair, Quatember, Wagner, Renfer, Oetliker, and Wiegert (2003), Salvati, Chandra, Giovanna-Ranalli, and Chambers (2010) or Alfons, Templ, and Filzmoser (2010).

However, simulation studies are often presented very briefly in academic articles without providing enough details about the underlying structure of the study. Alfons *et al.* (2010) provide a framework (`simFrame`) which helps to conduct simulation studies in a reproducible environment. The R-package `simFrame` (Alfons *et al.* 2010) is an object-oriented tool for statistical simulation studies based on S4 classes. It includes a wide range of different features (like data generation, sampling schemes, outlier contamination mechanisms and missing values) to conduct simulation studies. `simFrame` was originally developed for simulations in the context of survey statistics but is now designed to be as general as possible (cf. Alfons *et al.* 2010).

Survey statistics are used, for example, in order to deliver specific indicators as a basis for economic and political decision processes. Especially regional or group-specific comparisons are of interest (cf. Schmid and Münnich 2014). Surveys which shall provide the sufficient data for these regional indicators, however, are generally designed for larger areas (NUTS 1-2 level). Hence, sample information on more detailed levels, like NUTS3, is hardly available so that classical estimation methods (direct estimators) may lead to high variances of the estimates (cf. Ghosh and Rao 1994). In this case, small area estimation methods may reveal highly improved results for the target estimates. Small area estimation has become more and more attractive over the last decade:

In 2002, small area estimation (SAE) was flourishing both in research and applications, but my own feeling then was that the topic has been more or less exhausted in terms of research and that it will just turn into a routine application in sample survey practice. As the past 9 years show, I was completely wrong; not only is the research in this area accelerating, but it now involves some of the best known statisticians... Pfeffermann (2013)

However, the empirical part is also often presented very briefly in the context of small area estimation. Thus, there is a need to have a suited framework to guarantee the reproducibility of analysis. To the best of our knowledge, there is not any R-package or framework adjusted for the special case of small area estimation which provides a simulation environment.

The aim of this article is to introduce a new R-package, *saeSim*, which supports the process of making simulation studies in the field of small area estimation reproducible. To be more precise, the package has three main objectives: First, provide tools for data generation which are reproducible. Second, unify the process of simulation studies. Third, make the source-code of simulation studies available, such that it supports the conducted research in a transparent manner.

The paper is organised as follows. In Section 2 we give a short introduction to small area estimation focusing mainly on unit-level (Battese, Harter, and Fuller 1988) and area-level models (Fay and Herriot 1979). Section 3 introduces a framework for simulation studies and how it is supported by the R-package *saeSim*. To illustrate some of the features of the package we present a case study in Section 4. We conclude the paper in Section 5 by summarising the main findings and by providing some avenues for further research.

2. Small area estimation

The objective of small area estimation is to produce reliable statistics (means, quantiles, proportions, etc.) for domains where little or no sampled units are available. Groups may be areas or other entities, for example defined by socio-economic characteristics. The demand for such estimators is rising as they are used for fund allocation, educational and health programs (Pfeffermann 2013). As direct estimation of such statistics are considered to be unreliable, methods in small area estimation try to improve the domain predictions by borrowing strength from neighbored or *similar* domains. This can be achieved by using additional information from census data or registers to assist the prediction for non-sampled domains or domains with little information. For the purpose of this article we will introduce two basic models frequently used in small area estimation, the unit-level model introduced by Battese *et al.* (1988) and the area-level model introduced by Fay and Herriot (1979).

The unit level model (Battese *et al.* 1988) can be expressed as:

$$\begin{aligned} y_{ij} &= x_{ij}^\top \beta + v_i + e_{ij} \\ v_i &\stackrel{iid}{\sim} N(0, \sigma_v^2) \\ e_{ij} &\stackrel{iid}{\sim} N(0, \sigma_e^2) \end{aligned}$$

where $i = 1, \dots, D$ and $j = 1, \dots, n_i$. The population of size N is divided into D non-overlapping small areas of sizes N_i and into n sampled and $N - n$ non-sampled units, denoted by s and r respectively. y_{ij} is the the dependent variable for domain i and unit j , and x_{ij} are the corresponding auxiliary information for that unit. Furthermore v and e are independent. Let $\hat{\beta}$ denote the best linear unbiased estimator (BLUE) of β and \hat{v}_i defines the linear unbiased predictor (BLUP) of v_i (cf. ? or ?). The empirical best linear unbiased predictor (EBLUP) of the mean in small area i in the Battese-Harter-Fuller model is then given by

$$\hat{\bar{y}}_i^{BHF} = N_i^{-1} \left\{ \sum_{j \in s_i} y_{ij} + \sum_{j \in r_i} (x_{ij}^\top \hat{\beta} + \hat{v}_i) \right\}. \quad (1)$$

Due to reasons of confidentiality unit-level information is not always available. Instead only aggregates, or rather the direct estimators may be supplied. However, these direct estimations are known to be unreliable, hence in such situations area-level models can be valuable. The area-level model introduced by Fay and Herriot (1979) is build on a sampling model:

$$y_i = \mu_i + e_i,$$

where y_i is a direct estimator of a statistic of interest μ_i for an area i . The sampling error e_i is assumed to be independent and normally distributed with known variances $\sigma_{e,i}^2$, i.e. $e_i | \mu_i \sim$

$N(0, \sigma_{e,i}^2)$. The model is modified with a linking model by assuming a linear relationship between the true area statistic μ_i and some auxiliary variables x_i :

$$\mu_i = x_i^\top \beta + v_i,$$

with $i = 1, \dots, D$. Note that x_i is a vector containing area-level (aggregated) information for P variables and β is a vector ($1 \times P$) of regression coefficients describing the (linear) relationship. The model errors v_i are assumed to be independent and normally distributed, i.e. $v_i \sim N(0, \sigma_v^2)$. Furthermore e_i and v_i are assumed to be independent. Combining the sampling and linking model leads to:

$$y_i = x_i^\top \beta + v_i + e_i. \quad (2)$$

Model 2 is effectively a random-intercept model where the distribution of the error term e_i is heterogeneous and known. The EBLUP of the small area mean in the Fay-Herriot (FH) model is given by

$$\hat{y}_i^{FH} = x_i^\top \hat{\beta} + \hat{v}_i. \quad (3)$$

3. A simulation framework

In this section we will present the simulation framework implemented in `saeSim`. The framework relies strongly on the idea to describe a simulation as a process of data manipulation. Independent of simulation studies, `dplyr` as well as `tidyr` strongly promote this idea by providing tools for cleaning and transforming data. In those frameworks every defined function takes a `data.frame` as input and also returns it modified. This leads to a natural connection between all defined functions as the result of one function can be directly passed to the next. The symbioses of these packages with the pipe operator (`%>%`) from the package `magrittr` (Bache and Wickham 2014) only emphasises the process of data manipulation.

In `saeSim` we want to extend this approach to simulation studies where we have the field of small area estimation in mind. The main focus lies on the description of a simulation as a process of data manipulation. Each step in this process can be defined as a self contained component (function) and thus easily replaces extended and most importantly reused. Before we go into any detail of the functionality of the package we will discuss this process behind simulation studies followed by how `saeSim` maps this process into R.

Simulation studies in small area estimation address three different levels, the population, the sample and data on aggregated level. Figure 1 illustrates these levels. In the left column the steps of data manipulation are described, the right column presents the function names to define those steps. The **population-level** defines the data on which a study is conducted and may be a true population, synthetic population data or randomly generated variates from a model. We see three different point of views to define a population. First *design-based*, which means that a simulation study is based on true or synthetic data of *one* population. Second a *semi-model-based* point of view, where only one population is drawn from a model and is fixed in the whole simulation study. And third, *model-based* studies which have changing random populations drawn from a model.

The scope of this article is not not to promote any of those viewpoints, but simply to identify the similarity in them. The *base* (first component in figure 1) of any simulation study is a data table, the question is, if this data is *fixed* or *random* over the course of the simulation. Or from a more technical point of view, is the data generation (the second step in figure 1) repeated in each simulation run or omitted?

Depending on the choice of a fixed or random population it is necessary to recompute the population domain-statistics like domain means and variances, or other statistics of interest (third component in figure 1).

The **sample-level** is when domain predictions are conducted for unit-level models. Independently of how the population is treated, fixed or random, this phase consists of two steps, first drawing a sample, and second conducting computations on the samples (fourth and fifth component in figure 1). Given a sample, design or model based small area methods are applied. Of interest are estimated parameters, which can be estimated model parameters or domain predictions as well as measures of uncertainty for the estimates.

As the sample-level is when unit-level models are applied, the **aggregate-level** is when area-level models are applied (the seventh and last component in figure 1). Area-level models in small area estimation typically only use information available for domains (in contrast to units). Thus, the question for simulation studies for area-level methods is, if the data is generated on unit-level and used after the aggregation (sixth component in figure 1) or if the data is generated directly on area-level, i.e. drawn from an area-level model. Depending on whether unit-level data and sampling are part of the simulation the aggregate-level follows the generation of the population or is based on the aggregated sample. Again, we do not promote a specific viewpoint but simply allow steps in the process of simulation to be omitted.

Depending on the topic of research, steps in this simulation framework can be more relevant than others or completely irrelevant. We see these steps more as a complete list of phases one can encounter, thus single components can be omitted if not relevant in specific applications. For example *data generation* is not relevant if you have population data, or the *sample-level* is not used, when the sample is directly drawn from the model.

From this considerations, **saeSim** maps the different steps into R. Two layers with separate responsibilities need to be discussed. The first is *how* different simulation components can be combined, and the second is *when* or in which order they are applied. Regarding the first, in **saeSim** we put a special emphasis on the interface of each component, which is to use functions which take a **data.frame** as argument and have a **data.frame** as return value. This definition of interfaces, the return value of one component is the input of the next, is used for all existing tools in **saeSim**.

The second column in figure 1 shows how the different steps in a simulation can be accessed. It is important to note that the functions in figure 1 control the process, the second layer, i.e. *when* components are applied. Each of these functions take a simulation setup object to be modified and a function with the discussed interface as arguments. Hence a simulation setup is a collection of functions to be applied in a certain sequence. Also the second-layer functions have a defined interface: a **sim_setup** as input to be modified and a **sim_setup** as output. Thus, components can be chained together using the *pipe operator* (`%>%`) from the package **magrittr**.

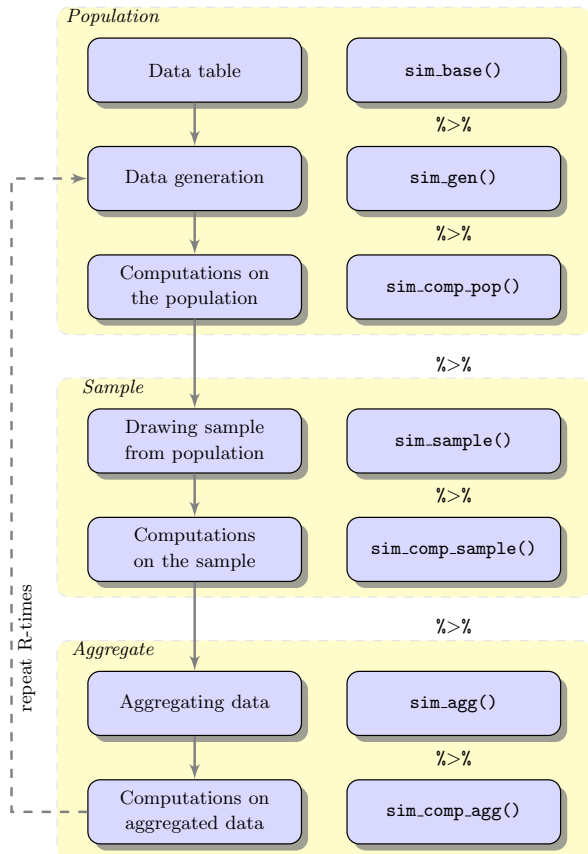


Figure 1: Process of simulation. Left column are the steps in a simulation. Right column are the corresponding function names to represent those steps in R

With `saeSim` we want to contribute tools for simulation studies in the field of small area estimation. We see the need for sharing tools for data generation and simulation amongst the scientific community and thus defined an interface for these tools as well as a platform to make them accessible. By defining the steps behind a simulation we hope to promote a reasonable way to communicate them alongside publications and during research. In the next section we present two case studies and introduce some of the main features provided by the package.

4. Case studies

We will present two case studies, one model-based simulation in section 4.1 and a design-based simulation in section 4.2. Before, we want to introduce some basic functionalities as the pipe operator (`%>%`) needs some explanation, if unfamiliar. The pipe operator is designed to make otherwise nested expressions more readable as a line can be read from left to right, instead of inside out (Bache and Wickham 2014). As a minimal example see the following lines which are equivalent:

```
> sum(1:10)
> 1:10 %>% sum
```

In `saeSim` we rely on this operator, although all functions can be used without, we strongly recommend to use it. The following example shows some of the aspects of the package:

```
> setup1 <- sim_base_lm() %>% sim_sample(sample_number(5))
> setup2 <- sim_base_lm() %>% sim_sample(sample_fraction(0.05))
```

Without knowing anything about the setup defined in `sim_base_lm` we can see that `setup1` and `setup2` only differ in the applied sampling scheme. `sim_sample` is responsible to control when a function is applied (after the population-level) and `sample_number(5)` and `sample_fraction(0.05)` define the explicit way of drawing samples. Braking the responsibility of each component into what is applied and when it is applied makes it possible to add new components to any step in the process. The composition of a simulation in that manner will focus on the definition of components and hide control structures. Any function can be passed to `sim_sample` which has a `data.frame` as input as well as return value. The only responsibility of that function is to draw a sample, which makes it easy to find, understand and reuse when published. The operator `%>%` is used to add new components to the setup.

4.1. Model-based simulation

In the following we want to show how to construct a simulation in a model-based setting. The goal is to estimate the domain predictions under a FH model. Components which are involved are *data generation* and *computing on aggregated data*. The first step is to generate the data under the model:

$$y_i = 100 + 2 \cdot x_i + v_i + e_i$$

where $x_i \stackrel{iid}{\sim} N(0, 4^2)$, $v_i \stackrel{iid}{\sim} N(0, 1)$ and $e_i \stackrel{indep}{\sim} N(0, \sigma_i^2)$ with $\sigma_i^2 = 0.1, 0.2, \dots, 4$ and $i = 1, \dots, 40$ as index for the domains. Also are x_i , v_i and e_i independent from each other. The area-level data for the simulation is generated in each Monte Carlo repetition.

In this case the *base-component* is a data table with an id variable named `idD` and constructed with the function `base_id`. Any random number generator in R can be used, however we have normally distributed variates, for which some predefined functions are available.


```
> library(saeSim)

> setup <- base_id(nDomains = 40, nUnits = 1) %>%
+   sim_gen_x(mean = 0, sd = 4) %>%
+   sim_gen_v(mean = 0, sd = 1)
> setup
```

	idD	x	v
1	1	-2.5058152	-0.1645236
2	2	0.7345733	-0.2533617
3	3	-3.3425144	0.6969634
4	4	6.3811232	0.5566632
5	5	1.3180311	-0.6887557
6	6	-3.2818735	-0.7074952

Note that if you print a simulation setup to the console, as in the above example, one simulation run is performed and only the first rows of the resulting data table are printed. This enables interactivity with the object itself, however it hides that the setup object is a collection of functions to be called. The error component e_i in the model has different variances which is not covered by a predefined function. Thus, as a *generator component* we define a function which takes a `data.frame` as input and returns it after adding a variable named `vardir` with the variances and the variable `e` with the generated random numbers:

```
> gen_e <- function(dat) {
+   dat$vardir <- seq(0.1, 4, length.out = nrow(dat))
+   dat$e <- rnorm(nrow(dat), sd = sqrt(dat$vardir))
+   dat
+ }
> setup <- setup %>% sim_gen(gen_e)
> setup
```

	idD	x	v	vardir	e
1	1	-2.2746749	-0.5059575	0.1	0.1344285
2	2	-0.5407145	1.3430388	0.2	-0.1067262
3	3	4.7123480	-0.2145794	0.3	0.5797550
4	4	-6.0942672	-0.1795565	0.4	0.5606229
5	5	2.3757848	-0.1001907	0.5	-0.4378710
6	6	1.3318015	0.7126663	0.6	1.7088396

The last step in data generation is to construct the response variable which will be named `y` and added to the data. Also we will add the *true* area statistic under the model to the data:

```
> setup <- setup %>%
+   sim_resp_eq(y = 100 + 2 * x + v + e) %>%
+   sim_comp_pop(comp_var(trueStat = y - e))
```

To add the area-level predictions from a Fay-Herriot model we need to define another component. The function takes a `data.frame` as input and returns the modified version. For the estimation of the EBLUP under the FH model we use the function `ebupFH` from the package `sae` (Molina and Marhuenda 2013). Finally we add `comp_FH` to the process:

```
> library(sae)
```

```

> comp_FH <- function(dat) {
+   modelFH <- eblupFH(y ~ x, vardir, data = dat)
+   dat$FH <- modelFH$eblup
+   dat
+ }
> setup <- setup %>% sim_comp_agg(comp_FH)
> setup

  idD      x      v vardir      e      y trueStat      FH
1   1 1.637607 0.7073107  0.1 0.1258998 104.10843 103.98253 104.06121
2   2 6.755493 1.0341077  0.2 -0.1822523 114.36284 114.54509 114.23876
3   3 6.346354 0.2234804  0.3 0.7253263 113.64151 112.91619 113.45213
4   4 -1.323631 -0.8787076 0.4 -0.4434978  96.03053  96.47403  96.45416
5   5 -9.140942  1.1629646 0.5 -0.4105563  82.47052  82.88108  82.46045
6   6  9.990646 -2.0001649 0.6 -0.7754272 117.20570 117.98113 118.08379

```

The object `setup` stores all necessary information to run one iteration of the simulation. In the following $R = 100$ repetitions are performed, the result is a list of `data.frames`. The function `rbind_all` from the package `dplyr` is used to combine the resulting list:

```

> library(dplyr)

> simResults <- sim(setup, R = 100) %>% rbind_all
> simResults %>% select(idD, idR, simName, trueStat, y, FH)

```

Source: local data frame [4,000 x 6]

	idD	idR	simName	trueStat	y	FH
1	1	1		107.87088	108.21065	108.30528
2	2	1		113.29033	114.13809	113.80733
3	3	1		105.88208	105.55180	105.63108
4	4	1		84.61171	84.36450	84.63272
5	5	1		103.68692	103.39260	103.42371
6	6	1		104.26130	103.97032	103.79131
7	7	1		97.85114	97.54439	97.43298
8	8	1		101.93187	101.66741	101.49500
9	9	1		92.51517	93.88300	93.67397
10	10	1		104.07055	103.37302	104.01407
...

An additional variable `idR` is automatically added as an ID-variable for the iteration as well as a variable `simName` to distinguish between scenarios, should there be more than one. At this time we do not provide further tools to process the resulting data. As it is a `data.frame` many packages are available in R. This point will be presented in more detail in the design-based study.

4.2. Design-based simulation

In the design-based simulation we want to illustrate the use of `saeSim` when starting from data of a population. For this purpose we use a synthetic population generated from Austrian EU-SILC (European Union Statistics on Income and Living Conditions) data. The data consists of 25 thousand households and does not represent the true population size of Austria. It is published alongside the R-package `simFrame` (Alfons *et al.* 2010) where it is used as an example data set. To keep this study as simple as possible, we further restrict the data for the main income holder and will only use some of the provided auxiliary information.


```
> data(eusilcP, package = "simFrame")
> simDat <- eusilcP %>%
+   mutate(agesq = age^2, eqIncome = as.numeric(eqIncome)) %>%
+   filter(main) %>%
+   select(region, eqIncome, age, agesq, gender)
> head(simDat)
```

	region	eqIncome	age	agesq	gender
1	Upper Austria	11128.45	25	625	male
2	Styria	19694.85	53	2809	male
3	Styria	5066.24	30	900	female
4	Upper Austria	31480.01	32	1024	male
5	Vienna	17813.40	77	5929	female
6	Lower Austria	13501.53	35	1225	male

Using this data as population we want to repeatedly draw samples from it. The sampling design is to draw 10 per cent from each region with simple random sampling. For each region the direct estimator for income as well as the EBLUP under the BHF model is computed. We use the function `ebLUPBHF` from the package `sae` for that. This function expects three data objects to produce the predictions. The sampled data, the population means of the auxiliary variables and the population sizes in each domain. Before we can begin to construct the simulation setup we store these data tables as attributes to the population data. There are other options but setting attributes to the processed data is supported in the package and presents a very flexible form of processing data on different aggregation levels.

```
> attr(simDat, "popMeans") <- group_by(simDat, region) %>%
+   summarise(age = mean(age),
+             agesq = mean(agesq),
+             genderFemale = mean(as.integer(gender) - 1),
+             trueStat = mean(eqIncome))
> attr(simDat, "popMeans")
```

Source: local data frame [9 x 5]

	region	age	agesq	genderFemale	trueStat
1	Burgenland	54.50063	3269.677	0.3366708	22005.42
2	Lower Austria	51.95259	3009.934	0.3777874	19813.37
3	Vienna	46.98310	2486.448	0.4662797	20395.84
4	Carinthia	51.81428	2995.735	0.3540337	19486.18
5	Styria	50.64087	2886.845	0.3573538	19335.39
6	Upper Austria	50.18644	2795.804	0.3443871	20517.29
7	Salzburg	51.44943	2965.268	0.4189108	19890.33
8	Tyrol	51.76707	2995.451	0.3975648	19350.89
9	Vorarlberg	49.06904	2697.382	0.3583756	22156.12

```
> attr(simDat, "popN") <- group_by(simDat, region) %>% summarise(N = n())
> attr(simDat, "popN")
```

Source: local data frame [9 x 2]

	region	N
1	Burgenland	799
2	Lower Austria	4619

```

3      Vienna 5857
4      Carinthia 1723
5      Styria 3386
6 Upper Austria 4071
7      Salzburg 1671
8      Tyrol 1889
9      Vorarlberg 985

```

Before we come to the estimation, the first step is to add a sampling scheme. As stated earlier, the starting point of a simulation setup is to provide a `data.frame` as *base-component* which, in this case, is the population data. Then the sampling component is added, in which we define to draw 10 per cent of the observations from each domain with simple random sampling.

```

> setup <- simDat %>%
+   sim_sample(sample_fraction(0.1, groupVars = "region"))
> setup

```

```

      region eqIncome age agesq gender
1 Burgenland 23572.28  67  4489  male
2 Burgenland 24056.37  26   676  male
3 Burgenland 21613.73  46  2116  male
4 Burgenland 11750.30  40  1600  male
5 Burgenland  9664.08  80  6400 female
6 Burgenland 19369.91  63  3969 female

```

What now needs to be done, is to define the components which add the desired estimates to the data. Here we will compute the mean of income in each domain as the direct estimator and the EBLUP under the BHF model. Although this could be done in one step, we will separate the two computations to illustrate how to combine several estimations and define each component independent of one another. This focus on the definition of each component and meeting the convention of the defined interface is the intended approach. It will automatically organise the simulation and each component is arranged using the simulation framework. Hence, we will define two functions, one for adding the direct estimates and one for adding the EBLUP.

```

> comp_direct <- function(dat) {
+   attr(dat, "sampleMean") <-
+     dat %>% group_by(region) %>% summarise(direct = mean(eqIncome))
+   dat
+ }
> comp_BHF <- function(dat) {
+   popMeans <- select(attr(dat, "popMeans"), -trueStat)
+   modelBHF <-
+     eblupBHF(eqIncome ~ age + agesq + gender, region, meanxpop = popMeans,
+               popnsize = attr(dat, "popN"), data = dat)
+   attr(dat, "BHF") <- modelBHF$eblup
+   dat
+ }

```

Another positive aspect of the above definitions is, that each step is relatively small and the purpose clearly defined, which will make them easy to understand and reuse. Finally the simulation results are combined in an *aggregation-component*, possibly followed by the application of area-level models but omitted here. The result of this aggregation step will be a `data.frame` with one row for each region.

```
> agg_results <- function(dat) {
+   cbind(attr(dat, "sampleMean"),
+         BHF = attr(dat, "BHF")$eblup,
+         trueStat = attr(dat, "popMeans")$trueStat)
+ }
```

To combine the simulation setup and the defined components we arrange them using the function `sim_comp_sample` to ensure that the direct estimator and the EBLUP are computed on the sampled data, and `sim_agg` to add the above aggregation step.

```
> setup <- setup %>%
+   sim_comp_sample(comp_BHF) %>%
+   sim_comp_sample(comp_direct) %>%
+   sim_agg(agg_results)
> setup
```

	region	direct	BHF	trueStat
1	Burgenland	21933.95	21255.53	22005.42
2	Lower Austria	18885.08	19036.29	19813.37
3	Vienna	20734.08	20720.48	20395.84
4	Carinthia	19211.55	19377.21	19486.18
5	Styria	18704.75	18955.96	19335.39
6	Upper Austria	20636.94	20504.37	20517.29

To repeat the simulation $R = 50$ times the simulation setup can be passed to the function `sim`, the resulting list is directly combined using `rbind_all`.

```
> simResults <- setup %>% sim(R = 50) %>% rbind_all
> simResults
```

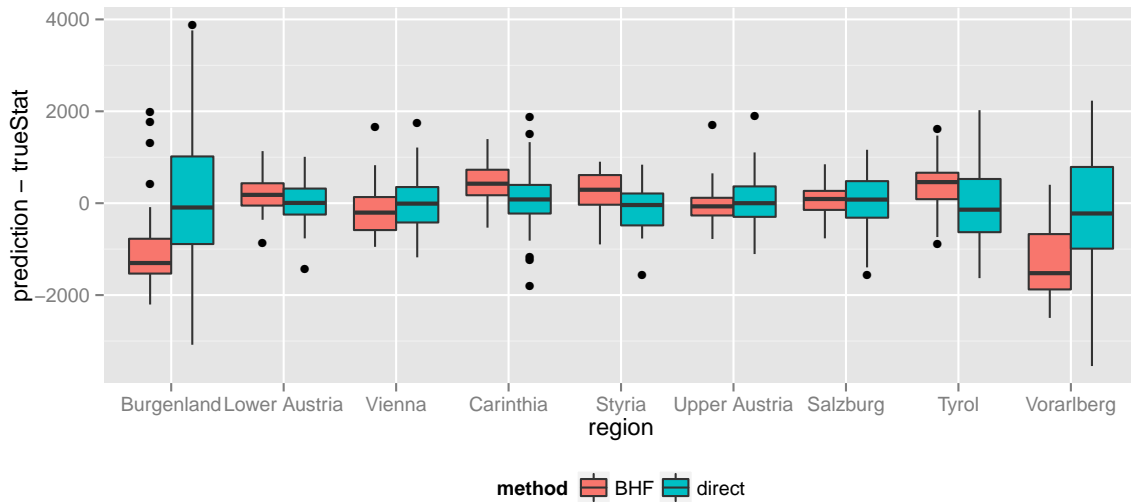
Source: local data frame [450 x 6]

	region	direct	BHF	trueStat	idR	simName
1	Burgenland	23083.61	20800.15	22005.42	1	
2	Lower Austria	20414.91	20264.56	19813.37	1	
3	Vienna	20756.78	20422.85	20395.84	1	
4	Carinthia	19581.64	19912.43	19486.18	1	
5	Styria	19443.11	19757.42	19335.39	1	
6	Upper Austria	20417.38	20356.38	20517.29	1	
7	Salzburg	19900.44	20009.83	19890.33	1	
8	Tyrol	18493.29	19540.25	19350.89	1	
9	Vorarlberg	21166.20	20444.35	22156.12	1	
10	Burgenland	21340.42	20510.92	22005.42	2	
..

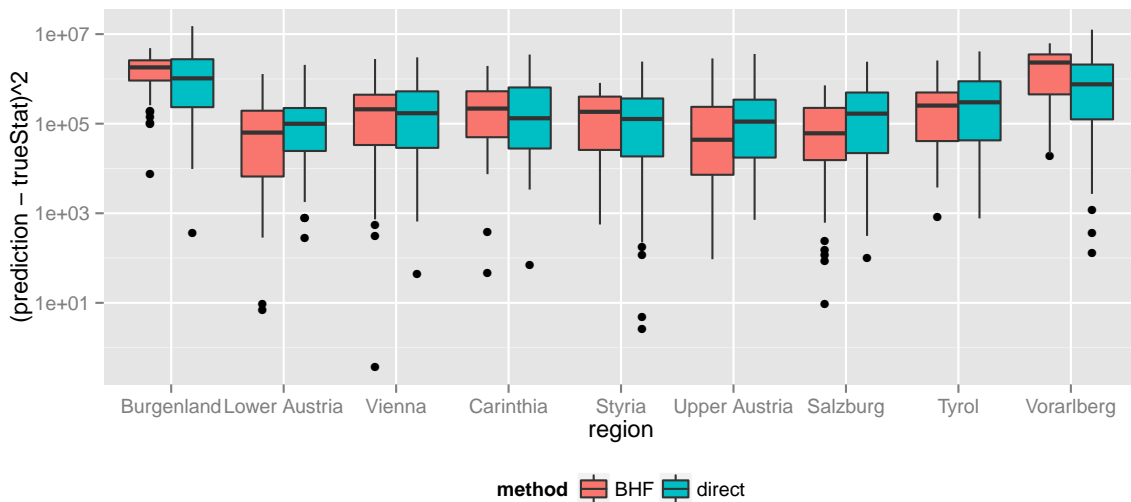
To further process the simulation results we present two graphics, using the package `ggplot2` (Wickham 2009) and `reshape2` (Wickham 2007) for further reshaping of the data.

```
> library(reshape2)
> ggDat <- melt(simResults,
+             id.vars = c("region", "trueStat"),
+             measure.vars = c("BHF", "direct"),
+             variable.name = "method",
+             value.name = "prediction")
```

```
> library(ggplot2)
> ggplot(ggDat, aes(x = region, y = prediction - trueStat, fill = method)) +
+   geom_boxplot() + theme(legend.position = "bottom")
```



```
> ggplot(ggDat, aes(x = region, y = (prediction - trueStat)^2, fill = method)) +
+   geom_boxplot() + scale_y_log10() + theme(legend.position = "bottom")
```



5. Outlook

With *saeSim* we have three objectives, making simulation tools available and reusable, unify the process behind simulations and third assist the researcher to design her code in a transparent way. By providing the package we not only want to make our own tools available but also open it for contributions. The package source is available on CRAN (<http://cran.r-project.org/web/packages/saeSim/>) and the repository for development on GitHub (<https://github.com/wahani/saeSim>). As GitHub allows to share and contribute source code using version-control, it is open for submissions. Apart from the availability of specific utility functions, we hope to promote and support the design of source code for simulation studies. One aspect is the design of simulations as processes of data. Furthermore, we encourage the definition of small and self contained components, i.e. functions, as this reduces the lines of code necessary to be read in order to understand its purpose.

The package already provides more than introduced in this article. To mention is the support of outlier contaminated data. At this time only representative outliers (outlying observations in the population) are supported (cf. ?). However, we plan to extend this approach for non-representative outliers (outliers are part of the sample but not the population). Furthermore, the inclusion of available random number generators available in the R-language is already possible and easy access to generate group effects as needed in mixed models is supported. Also, as the response is created by an R expression, any form of non-linearity in the relationship between response and auxiliary variables as well as error components can be modelled.

A more technical feature is a back-end for parallel computations which is a link to the `parallel` package in R (R Core Team 2014). Tools to process result data after the simulation as well as plotting methods for that data are an avenue for further research. Already available are some simple plots for the simulation setups as well as a summary method to get information on the expected runtime and structure of the resulting data.

References

- Alfons A, Templ M, Filzmoser P (2010). “An Object-Oriented Framework for Statistical Simulation: The R Package `simFrame`.” *Journal of Statistical Software*, **37**(3), 1–36. URL <http://www.jstatsoft.org/v37/i03/>.
- Allaire J, McPherson J, Xie Y, Wickham H, Cheng J, Allen J (2014). *rmarkdown: Dynamic Documents for R*. R package version 0.3.3, URL <http://CRAN.R-project.org/package=rmarkdown>.
- Bache SM, Wickham H (2014). *magrittr: magrittr - a forward-pipe operator for R*. R package version 1.0.1, URL <http://CRAN.R-project.org/package=magrittr>.
- Battese GE, Harter RM, Fuller WA (1988). “An error-components model for prediction of county crop areas using survey and satellite data.” *Journal of the American Statistical Association*, **83**(401), 28–36.
- Burgard JP, Kolb JP, Münnich R (2014). “Generation of Synthetic Universes for Micro-Simulations in Survey statistics.” *Working Paper*.
- Fay R, Herriot R (1979). “Estimation of Income for Small Places: An Application of James-Stein Procedures to Census Data.” *Journal of the American Statistical Association*, **74**(366), 269–277.
- Ghosh M, Rao JNK (1994). “Small Area Estimation: An Appraisal.” *Statistical Science*, **9**(1), 55–93.
- Knuth DE (1992). *Literate Programming*. CSLI, Stanford.
- Kolb JP (2013). *Generation of synthetic universes*. Ph.D. thesis, University of Trier, <http://ubt.opus.hbz-nrw.de/volltexte/2013/816/>.
- Leisch F (2002). “Sweave, Part I: Mixing R and LaTeX.” *R News*, **2**(3), 28–31. URL <http://CRAN.R-project.org/doc/Rnews/>.
- Molina I, Marhuenda Y (2013). *sae: Small Area Estimation*. R package version 1.0-2, URL <http://CRAN.R-project.org/package=sae>.
- Münnich R, Schürle J, Bihler W, Boonstra H, Knotterus P, Nieuwenbroek N, Haslinger A, Laaksonen S, Eckmair D, Quatember A, Wagner H, Renfer J, Oetliker U, Wiegert R (2003). “Monte Carlo Simulation Study of European Surveys.” DACSEIS Deliverables D3.1 and D3.2. URL https://www.uni-trier.de/fileadmin/fb4/projekte/SurveyStatisticsNet/Dacseis_Deliverables/DACSEIS-D3-1-D3-2.pdf.

- Pfeffermann D (2013). “New Important Developments in Small Area Estimation.” *Statistical Science*, **28**(1), 40–68. doi:10.1214/12-STS395. URL <http://dx.doi.org/10.1214/12-STS395>.
- R Core Team (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL <http://www.R-project.org/>.
- Rao JNK (2003). *Small Area Estimation*. Wiley, New York.
- Salvati N, Chandra H, Giovanna-Ranalli M, Chambers R (2010). “Small area estimation using nonparametric model-based direct estimator.” *Computational Statistics and Data Analysis*, **54** (9), 2159–2171.
- Schmid T, Münnich R (2014). “Spatial robust small area estimation.” *Statistical Papers*, **55**, 653–670. ISSN 0932-5026. URL <http://dx.doi.org/10.1007/s00362-013-0517-y>.
- Wickham H (2007). “Reshaping Data with the reshape Package.” *Journal of Statistical Software*, **21**(12), 1–20. URL <http://www.jstatsoft.org/v21/i12/>.
- Wickham H (2009). *ggplot2: elegant graphics for data analysis*. Springer New York. ISBN 978-0-387-98140-6. URL <http://had.co.nz/ggplot2/book>.
- Wickham H, Francois R (2014). *dplyr: A Grammar of Data Manipulation*. R package version 0.3.0.2, URL <http://CRAN.R-project.org/package=dplyr>.
- Yihui X (2013). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC. ISBN 978-1482203530, URL <http://yihui.name/knitr/>.

Affiliation:

Sebastian Warnholz
 Department of Economics
 Freie Universität Berlin
 D-14195 Berlin, Germany
 E-mail: Sebastian.Warnholz@fu-berlin.de
 URL: <http://www.wiwiss.fu-berlin.de/fachbereich/vwl/Schmid/Team/Warnholz.html>