Austrian Journal of Statistics
MMMMMM YYYY, Volume VV, 1–xx.
http://www.ajs.or.at/

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

Sebastian Warnholz
Freie Universität Berlin

Timo Schmid
Freie Universität Berlin

Abstract

The abstract of the article in English

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

1. Introduction

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. New statistical methods are applied in model-based and design-based simulation studies. Considering the demands for reproducible research we propose a framework for simulation studies inside the field of small area estimation.

Reproducible Research has become a widely discussed topic in general and also in the field of statistics. Thanks to the many mostly open-source tools like the R-language (**?**) and LaTeX, and also packages like knitr (**?**) and Sweave (**?**) and more recently rmarkdown (**?**), the integration of text and source code for statistical analysis is possible. Not only are tools available to make research reproducible, also the demand of making the analysis of articles reproducible is rising which is supported by the rising number of Open Access Journals. Publishing source-code and data alongside an article draws special attention to authoring an 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.* (**?**, p.99)

In this article we want to introduce a new package for the R-language to support the process of making simulation studies reproducible as well as supporting a *human-readable* interface. With this package we have three objectives in specific: First, making tools for data generation available and reusable. Second, unify the process behind simulation studies inside the field of Small Area Estimation. And third, making source-code of simulation studies available, such that it supports the conducted research in a transparent manner.

In section ?? we will give a short introduction to small area estimation and the role of simulation studies within that field. Section ?? will then introduce a framework for simulation studies and how it is supported by saeSim. To illustrate some of the package's features we present a case study in section ??.

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 (?). 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 neighboured or *similar* domains. This can be achieved by using additional information from census data 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 ? and the area-level model introduced by ?.

The unit level model (?) can be expressed as:

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

where $i = 1,\dots,D$ and $j = 1,\dots,n_i$. y_{ij} is the the dependent variable for domain i and unit j, and x_{ij} the corresponding auxiliary information for that unit. Furthermore v and e are independent. This model can be seen as a linear mixed model from which the best linear unbiased predictor (BLUP) can be derived and is used for the domain prediction (?).

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 ? 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 with $i = 1,\dots,D$ and D being the total number of areas. 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 the 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:

$$y_i = x_i^\top\beta + v_i + e_i. \tag{1}$$

Model ?? is effectively a random-intercept model where the distribution of the error term e_i is heterogeneous and known.

3. A simulation framework

In our opinion simulation studies can best be summarised when understood as a process of data manipulation. Thus the main focus of saeSim is to define the steps in that process which can then be *cleanly* defined and repeated. Before we go into any detail of the functionality of the package we will discuss the process behind simulation studies and later 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, as illustrated in figure **??**. 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.

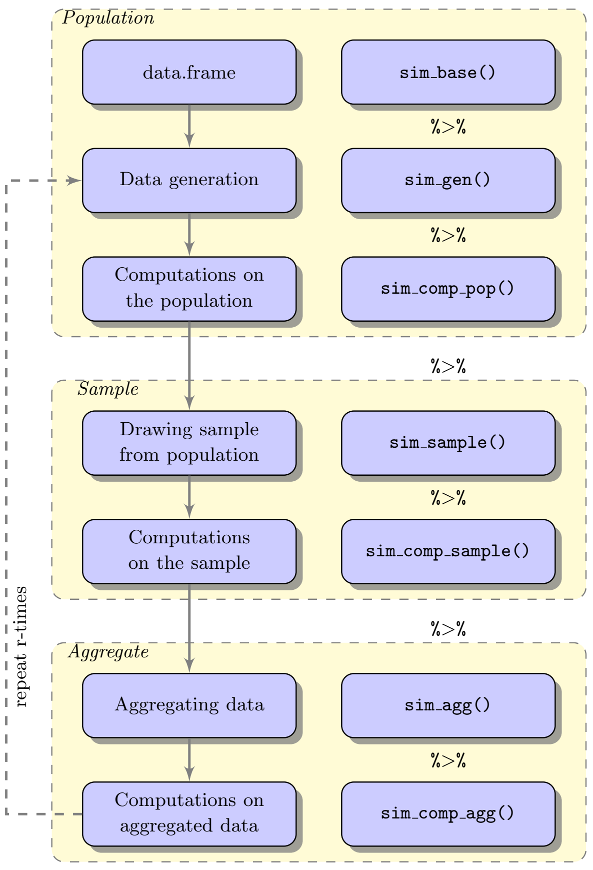


Figure 1: Process of simulation.

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 **??**) 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 **??**) 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 **??**).

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 **??**). 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 **??**). 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 **??**) 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 *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 is a widely used approach for data manipulation and promoted recently in the package dplyr (**?**). This definition of interfaces, the return value of one component is the input of the next, is also used in saeSim.

Understanding a simulation as a process of manipulating one data object, see the second column in figure **??** how the different steps in a simulation can be accessed. It is important to note that the functions in figure **??** 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. As the first-layer functions, also 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 (**?**).

Braking the responsibility into what is applied and when it is applied addresses several aspects of reproducibility. First, by defining the interface between all components, it is easy to combine them in any combination and thus easy to share and reuse. Second, by controlling when components are applied we avoid the necessity of control structures in syntax and emphasise on the definition of components. The following example shows these aspects of the package using a predefined simulation setup:

```
> 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. The pipe operator %>% is used to add new components to the setup. As said before the composition of a simulation in that manner will focus on the definition of components and hide control structures. The next example will repeat the simulation stored in setup1 two times. The results are returned in a list of data.frames which have five rows as we sample 5 observations:

```
> setup1 %>% sim(R = 2) %>% sapply(nrow)

[1] 5 5
```

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 will present a case study and focus more on the concrete functionality provided by the package.

4. Case study

4.1. Model-based simulation

```
> suppressPackageStartupMessages(library(saeSim))
> suppressPackageStartupMessages(library(sae))
> suppressPackageStartupMessages(library(dplyr))
> suppressPackageStartupMessages(library(reshape2))
> suppressPackageStartupMessages(library(ggplot2))
> set.seed(1) # for reproduction of the results
```

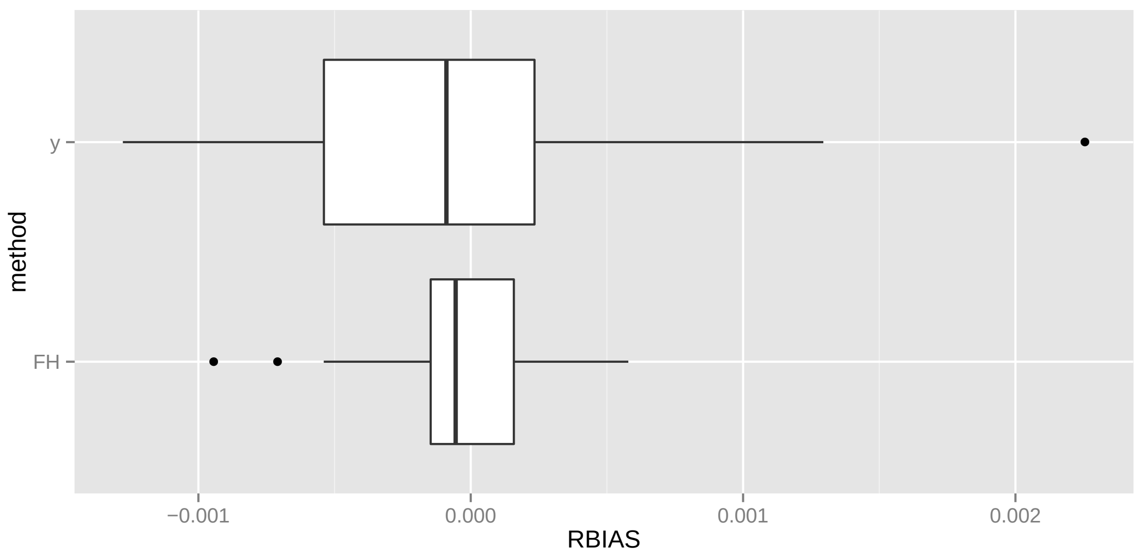
```
> comp_FH <- function(dat) {
+   # Computes the EBLUP under a Fay-Herriot model
+   modelFH <- eblupFH(y ~ x, vardir, data = dat, method = "FH")
+   dat$FH <- modelFH$eblup
+   dat
+ }
```

```
> simResults <- base_id(nDomains = 40, nUnits = 1) %>%
+   sim_gen_fe() %>%
+   sim_gen_re() %>%
+   sim_gen(comp_var(vardir = seq(0.1, 4, length.out = 40))) %>%
+   sim_gen(comp_var(e = rnorm(nrow(dat), sd = sqrt(vardir)))) %>%
+   sim_resp_eq(y = 100 + 2 * x + v + e) %>%
+   sim_comp_agg(comp_var(trueStat = y - e)) %>%
+   sim_comp_agg(comp_FH) %>%
+   sim(R = 500) %>%
+   rbind_all
```

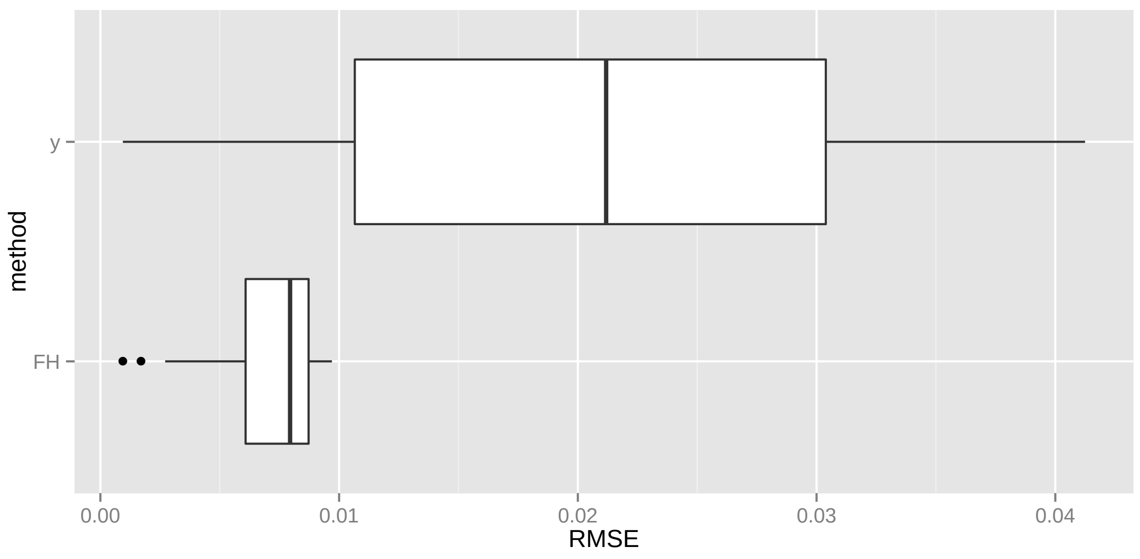
```
> ggDat <- simResults %>%
+   melt(id.vars = c("idD", "trueStat"),
+        measure.vars = c("FH", "y"),
+        variable.name = "method",
+        value.name = "prediction") %>%
+   mutate(RBIAS = (prediction - trueStat)/trueStat,
+          RRMSE = (prediction - trueStat)^2/trueStat) %>%
+   group_by(idD, method) %>%
+   summarise(RBIAS = mean(RBIAS),
+             RMSE = mean(RRMSE)) %>%
+   as.data.frame
> head(ggDat)
```

```
  idD method         RBIAS         RMSE
1   1     FH -0.0001326439 0.0009412468
2   1      y -0.0001285281 0.0009413428
3   2     FH  0.0002999555 0.0017042498
4   2      y  0.0002214372 0.0018404586
5   3     FH  0.0004056162 0.0027148056
6   3      y  0.0003529474 0.0032745696
```

```
> ggplot(ggDat, aes(y = RBIAS, x = method)) + geom_boxplot() + coord_flip()
```



```
> ggplot(ggDat, aes(y = RMSE, x = method)) + geom_boxplot() + coord_flip()
```



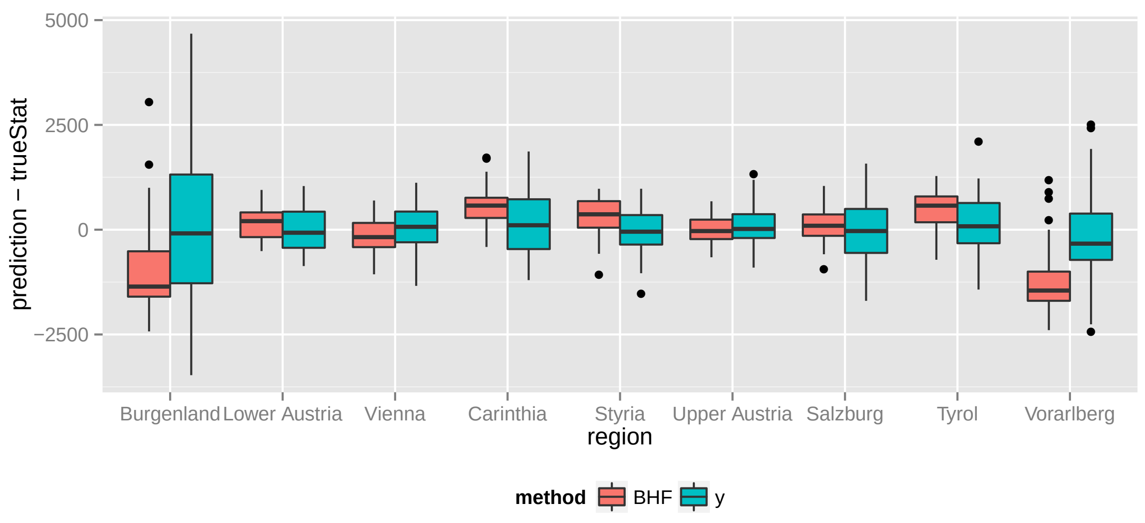
4.2. Design-based simulation

```
> data(eusilcP, package = "simFrame")
> simDat <- eusilcP %>%
+   mutate(agesq = age^2, eqIncome = as.numeric(eqIncome)) %>%
+   filter(main) %>%
+   select(region, eqIncome, age, agesq, gender) %>%
+   base_add_id(c("region"))
> attr(simDat, "popMeans") <-
+   simDat %>%
+   group_by(idD, region) %>%
+   summarise(age = mean(age),
+             agesq = mean(agesq),
+             genderFemale = mean(as.integer(gender) - 1),
+             trueStat = mean(eqIncome))
> attr(simDat, "popN") <-
```

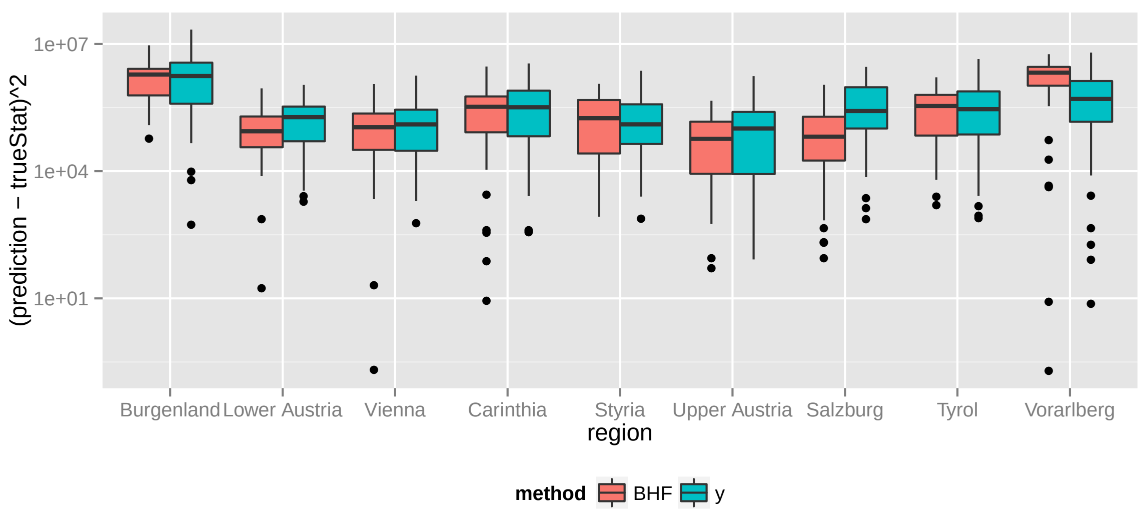
```
+   simDat %>%
+   group_by(idD) %>%
+   summarise(N = n())
> attr(simDat, "popN")

Source: local data frame [9 x 2]

  idD    N
1   1  799
2   2 4619
3   3 5857
4   4 1723
5   5 3386
6   6 4071
7   7 1671
8   8 1889
9   9  985

> attr(simDat, "popMeans")

Source: local data frame [9 x 6]
Groups: idD

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

>

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

```
> simResults <- simDat %>%
+   sim_sample(sample_fraction(0.1, groupVars = "idD")) %>%
+   sim_comp_sample(comp_BHF) %>%
+   sim_comp_sample(comp_direct) %>%
+   sim_agg(agg_results) %>%
+   sim(R = 50) %>%
+   rbind_all
> ggDat <- simResults %>%
+   melt(id.vars = c("idD", "region", "trueStat"),
+        measure.vars = c("BHF", "y"),
+        variable.name = "method",
+        value.name = "prediction")
```

```
> 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

Use this package to share and publish simulation studies alongside papers. Contribute to the package to make your ideas available. Contribute to the package and make your whole simulation study available.

- Link to simFrame
- Package Features
  - outliers
  - sampling
  - non-linear models
- How to contribute?
- parallel computations

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`

Austrian Journal of Statistics `http://www.ajs.or.at/`
published by the Austrian Society of Statistics `http://www.osg.or.at/`
Volume VV *Submitted:* yyyy-mm-dd
MMMMMM YYYY *Accepted:* yyyy-mm-dd