saeSim: Simulation Tools for Small Area Estimation

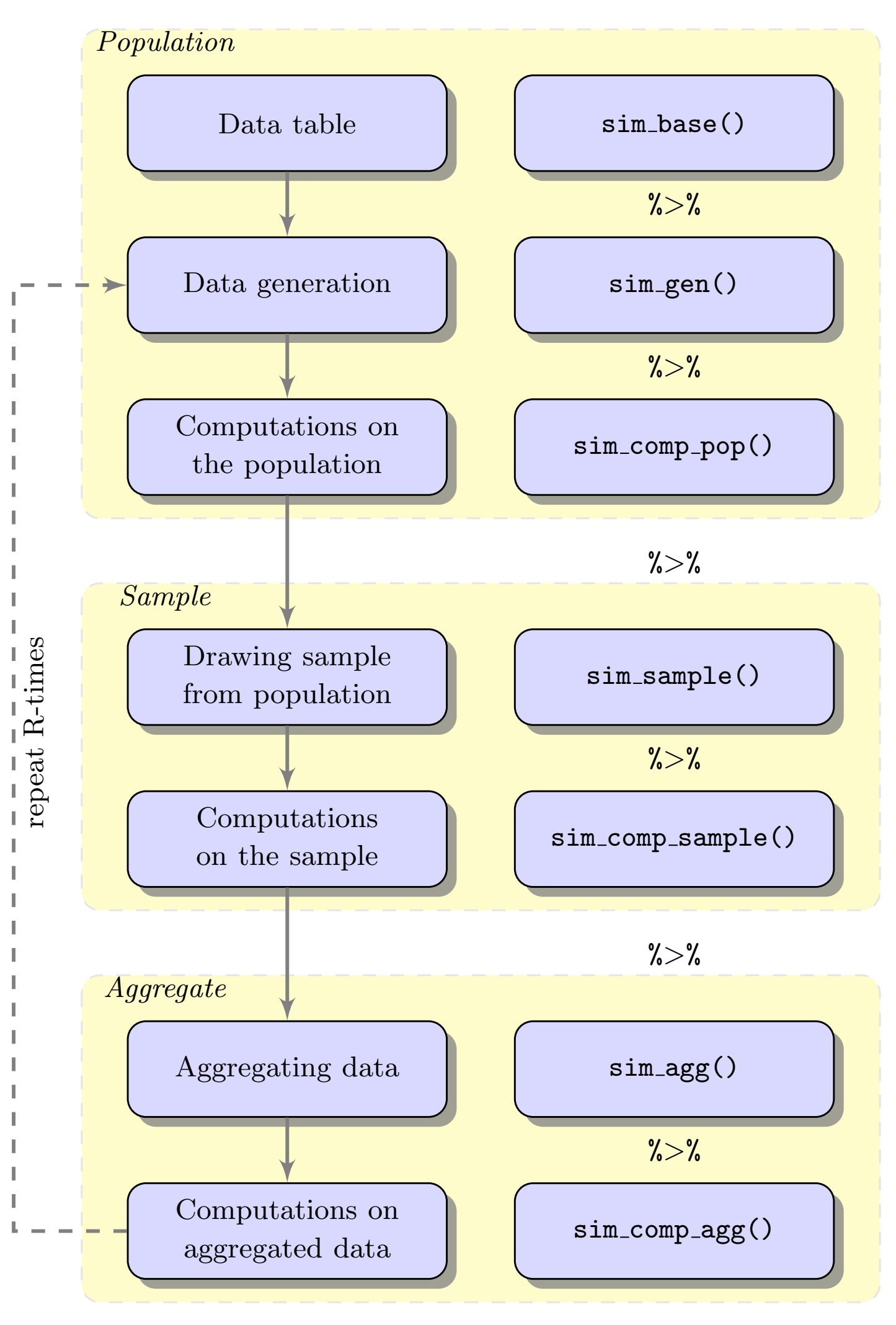
Sebastian Warnholz & Timo Schmid

Sebastian.Warnholz@fu-berlin.de

Introduction

With this package we provide tools for the composition of simulation studies in the context of small area estimation. We want to support and encourage the publication of source code by providing a standardised set of tools for the community.

- The field of Small Area Estimation is concerned with the development and application of statistical methods to report statistics for small domains. *Small* refers to the small number of sampled units within groups.
- Model and design based simulation studies have been used to introduce new methods to the field.
- Although we have small area estimation in mind, the framework may prove useful for the composition of micro simulations in general.
- The package highlights a specific way to map a simulation study into R, namely in terms of a pipeline where a data frame is modified in each step.
- With this package the composition of a simulation study is reduced to *chaining the steps together*.



Data generation

`saeSim` provides several functions to make the composition and generation of synthetic population data more efficient.

- You can use predefined generators, e.g. for spatially correlated variates, or include univariate random number generators from R.
- `sim_gen` defines the position in the chain. `gen_generic` controls the generation process and accepts any random number generator as argument. Also we defined shortcuts like `sim_gen_generic` to be less verbose.
- In the following you see a definition for drawing numbers from a linear mixed model. The response is constructed with an R expression and set by `sim_resp_eq`.

```r
setup <- sim_base() %>% # default with 100 areas (domains) and 100 units each
  sim_gen(gen_generic(rnorm, sd = 4, name = "x")) %>%
  sim_gen(gen_generic(rnorm, sd = 4, name = "e")) %>%
  sim_gen_generic(rnorm, sd = 2, groupVars = "idD", name = "v") %>%
  sim_resp_eq(y = 100 + 2 * x + v + e)
setup
```

```
##
## # data.frame [10,000 x 6]
##     idD   idU          x         e        v         y
## * <int> <int>      <dbl>     <dbl>    <dbl>     <dbl>
## 1     1     1  3.9462670  1.841257 2.443888 112.17768
## 2     1     2 -5.7587514  7.802529 2.443888  98.72891
## 3     1     3  0.7469758 -1.089258 2.443888 102.84858
## 4     1     4 -5.2633582  3.373478 2.443888  95.29065
## 5     1     5 -1.8117415  8.351238 2.443888 107.17164
## 6     1     6 -1.3475448  1.510108 2.443888 101.25891
## # ... with 9,994 more rows
```

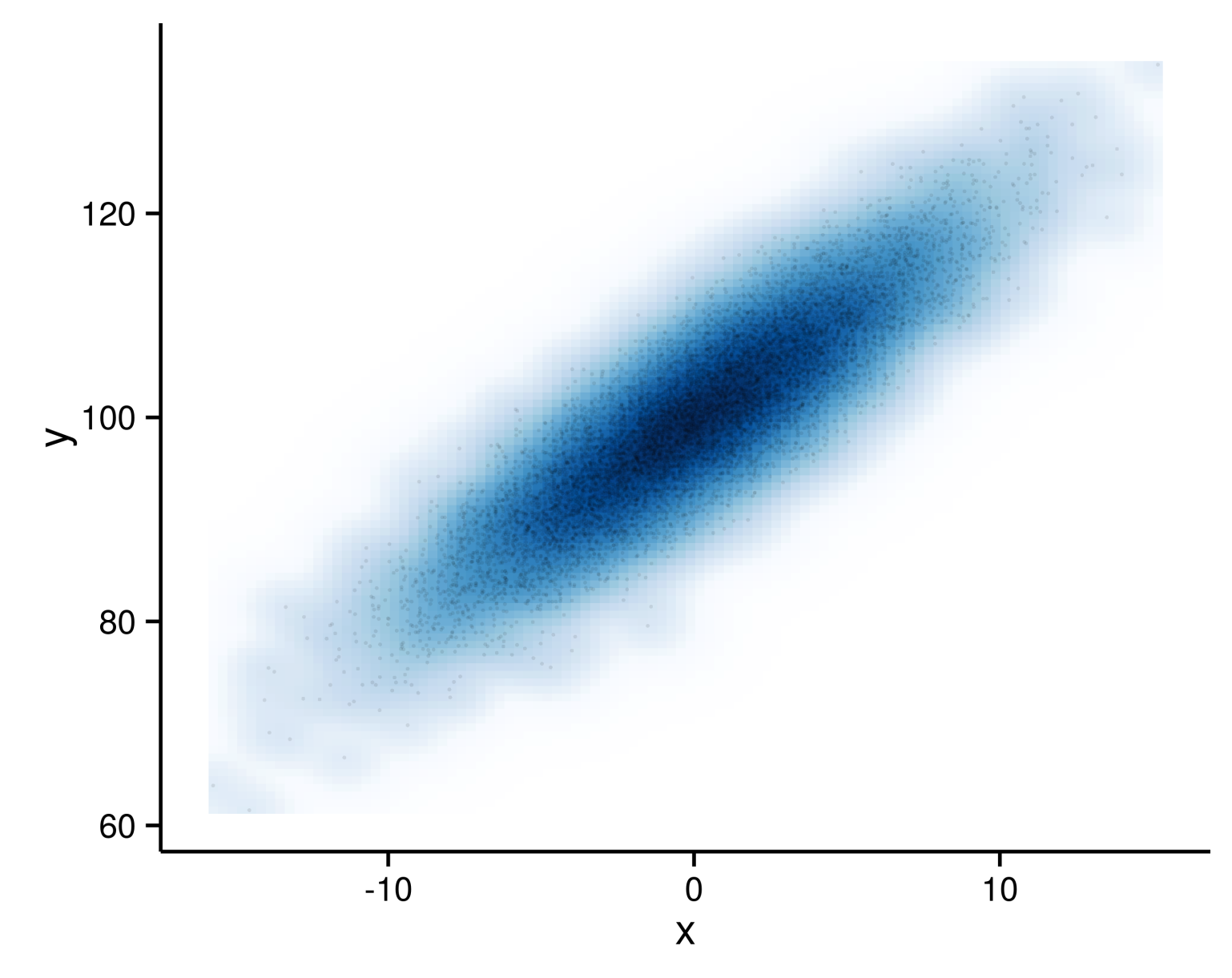
Outliers

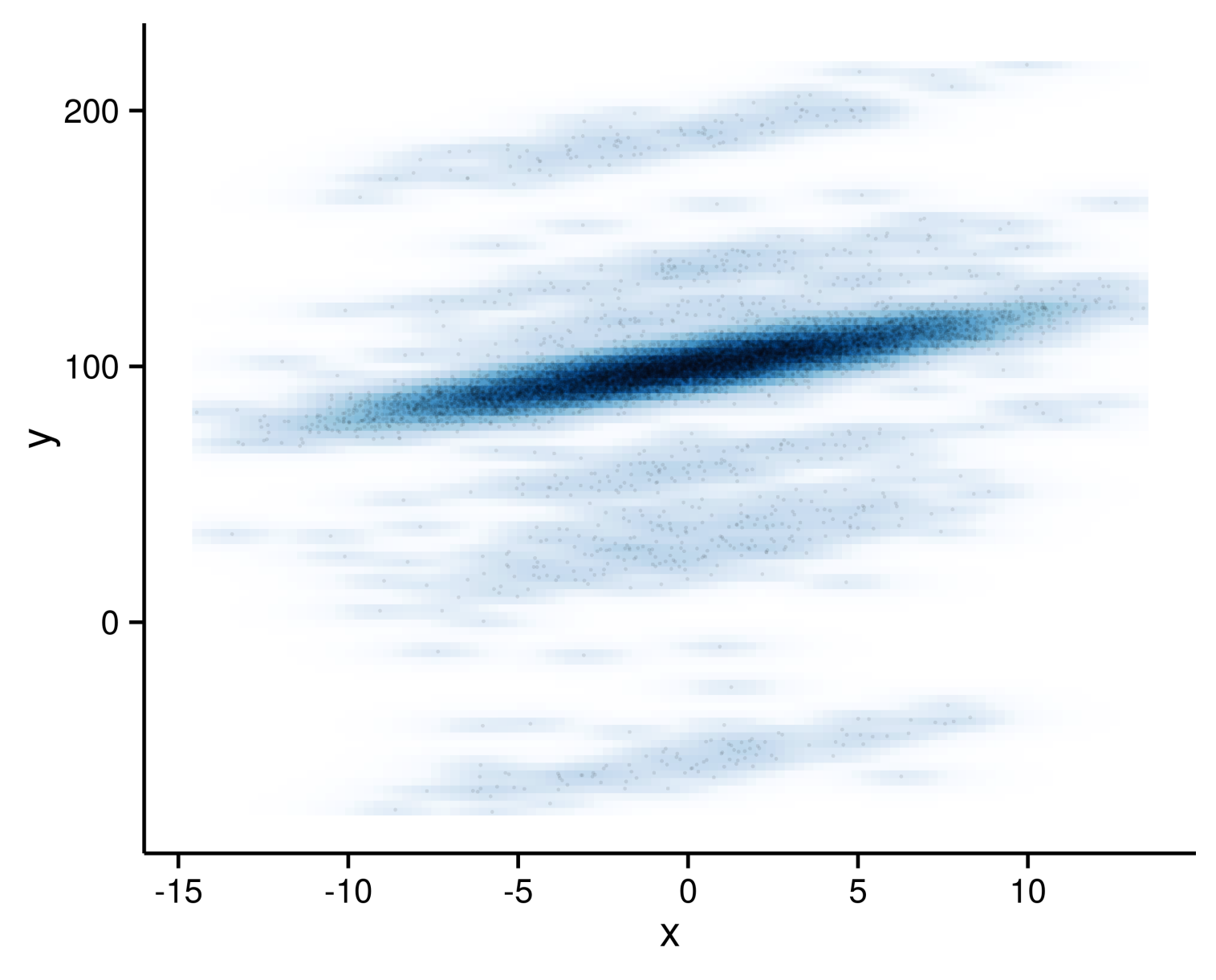
Outliers can be added to the data generation. `sim_gen_cont` provides the interface to control the degree and level of contamination.

- Set the frequency or probability for adding contaminated observations.
- You can choose between unit and area level outliers, i.e. *outlying* units or groups in the data.

```r
setupCont <- setup %>%
  sim_gen_cont(gen_generic(rnorm, sd = 50, name = "e"), 0.01, "unit") %>%
  sim_gen_vc(sd = 50, areaVar = "idD", nCont = 0.1, type = "area")

autoplot(setup)
autoplot(setupCont)
```





Parallel computations

Simulation studies are embarrassingly parallel. For parallel computations we utilize `parallelMap` which makes it easy to switch between different parallel back-ends in R, e.g. multicore, socket, MPI and BatchJobs.

- To start the simulation use the function `sim` and specify the number of runs.
- To start the simulation in parallel define the `mode` and number of `cpus`.
- Packages and also objects can be loaded on all nodes with extra arguments to `sim`.
- Here we perform 16 runs on 4 cores. `sim` always returns a `list` and we combine the results directly using `rbind_all`.

```r
sim(setup, R = 16, mode = "socket", cpus = 4) %>% dplyr::bind_rows() %>% dim()
```

```
## Starting parallelization in mode=socket with cpus=4.
## Mapping in parallel: mode = socket; cpus = 4; elements = 16.
## Stopped parallelization. All cleaned up.
## [1] 160000      8
```

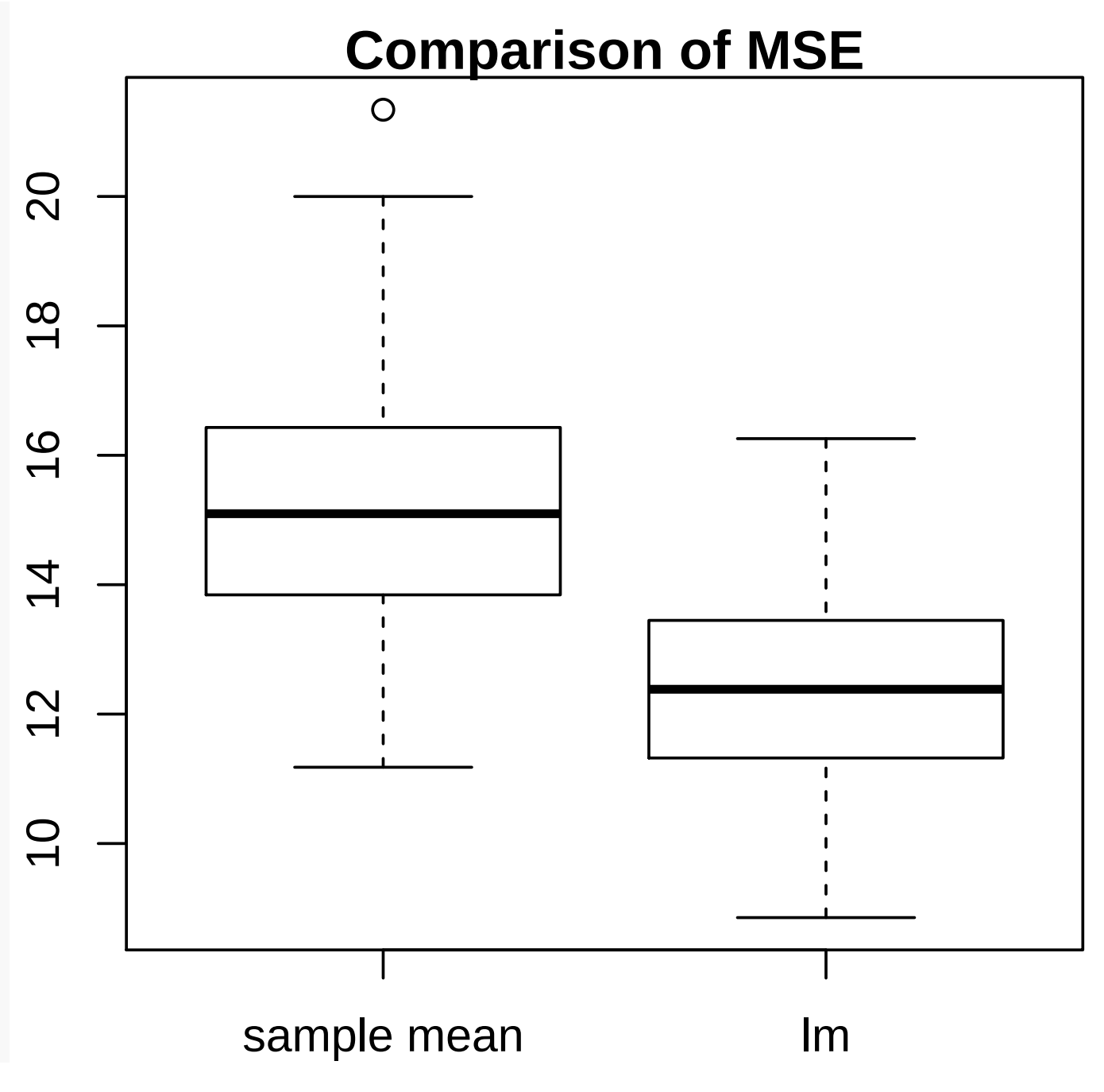
Extensions

To make `saeSim` more than just a tool for data generation define your own components and arrange them in the process.

- This is a complete example beginning with data generation, followed by simple random sampling in each domain and completing with a comparison of the mean squared error of the sample mean and the average of a linear predictor in each domain.
- Here we define `comp_lm` to use a linear model to predict the area means. The data is aggregated for each domain by the aggregation function `sim_agg`.
- At this time we do not see the need to provide tools for further reshaping and aggregation of results. Instead we rely on existing packages like `dplyr` which is used after the call to `sim`.

```r
comp_lm <- function(dat) {
  dat$lm <- predict(lm(y ~ x, data = dat))
  dat
}

simDat <- setup %>% sim_sample() %>%
  sim_comp_popMean() %>%
  sim_comp_sample(comp_lm) %>%
  sim_agg() %>% sim(R = 100) %>%
  bind_rows() %>% group_by(idD) %>%
  summarise(
    MSE_direct = mean((y - popMean)^2),
    MSE_lm = mean((lm - popMean)^2)
    )
boxplot(simDat$MSE_direct, simDat$MSE_lm,
        names = c("sample mean", "lm"),
        main = "Comparison of MSE")
```



Design and naming

You do not have to redefine, reuse instead!

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

- `setup1` and `setup2` only differ in the specific way samples are drawn.
- `sim_sample` inserts the sampling function at the appropriate position in the process.
- For every step in the process tools are named using the corresponding prefix, e.g. `gen_generic` for *data generation* or `sample_fraction` for *drawing samples*.

Remarks

- We hope to encourage the small area estimation community to contribute and publish the code of simulation studies alongside articles.
- `saeSim` is a suggestion how such studies can be conducted. The design aims to use common tools and to remove control structures from the code. At the same time we encourage the definition of short and self contained functions as *components*.
- The package is not complete. If you have suggestions you are welcome to contribute to `https://github.com/wahani/saeSim`.

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/*

Bache, S. M. & Wickham, H. (2014), *magrittr: A Forward-Pipe Operator for R*. R package version 1.5.
URL: *http://CRAN.R-project.org/package=magrittr*

Warnholz, S. & Schmid, T. (2016), 'Simulation Tools for Small Area Estimation: Introducing the R-package saeSim', *Austrian Journal of Statistics* **45**(1), 55–69.

Wickham, H. & Francois, R. (2015), *dplyr: A Grammar of Data Manipulation*. R package version 0.4.1.
URL: *http://CRAN.R-project.org/package=dplyr*

Timo Schmid

Department of Economics
Freie Universität Berlin
Timo.Schmid@fu-berlin.de

Sebastian Warnholz

Department of Economics
Freie Universität Berlin
Sebastian.Warnholz@fu-berlin.de