

Applications of Statistical Simulation in the Case of EU-SILC: Using the R Package `simFrame`

Andreas Alfons

Vienna University of
Technology

Matthias Templ

Vienna University of
Technology,
Statistics Austria

Peter Filzmoser

Vienna University of
Technology

Abstract

This paper demonstrates the use of `simFrame` for various simulation designs in a practical application with EU-SILC data. It presents the full functionality of the framework regarding sampling designs, contamination models, missing data mechanisms and performing simulations separately on different domains. Due to the use of control objects, switching from one simulation design to another requires only minimal changes in the code. Using bespoke R code, on the other hand, changing the code to switch between simulation designs would require much greater effort. Furthermore, parallel computing with `simFrame` is demonstrated.

Keywords: R, statistical simulation, EU-SILC.

1. Introduction

This is a supplementary paper to “An Object-Oriented Framework for Statistical Simulation: The R Package `simFrame`” (Alfons, Templ, and Filzmoser 2010d) and demonstrates the use of `simFrame` (Alfons 2011) in R (R Development Core Team 2010) for various simulation designs in a practical application. It extends the example for design-based simulation in Alfons *et al.* (2010d) (Example 6.1). Different simulation designs in terms of sampling, contamination and missing data are thereby investigated to present the strengths of the framework.

Note that the paper is supplementary material and is supposed to be read after studying Alfons *et al.* (2010d). It does not give a detailed discussion about the motivation for the framework, nor does it describe the design or implementation of the package. Instead it is focused on showing its full functionality for design-based simulation in additional code examples with brief explanations. However, model-based simulation is not considered here.

The European Union Statistics on Income and Living Conditions (EU-SILC) is panel survey conducted in EU member states and other European countries and serves as basis for measuring risk-of-poverty and social cohesion in Europe. An important indicator calculated from this survey is the *Gini coefficient*, which is a well-known measure of inequality. In the following examples, the standard estimation method (EU-SILC 2004) is compared to two semiparametric methods under different simulation designs. The two semiparametric approaches are based on fitting a Pareto distribution (e.g., Kleiber and Kotz 2003) to the upper tail of the data. In the first approach, the classical Hill estimator (Hill 1975) is used to estimate the

shape parameter of the Pareto distribution, while the second uses the robust partial density component (PDC) estimator (Vandewalle, Beirlant, Christmann, and Hubert 2007). All these methods are implemented in the R package **laeken** (Alfons, Holzer, and Templ 2010a). For a more detailed discussion on Pareto tail modeling in the case of the Gini coefficient and a related measure of inequality, the reader is referred to Alfons, Templ, Filzmoser, and Holzer (2010e).

The example data set of **simFrame** is used as population data throughout the paper. It consists of 58 654 observations from 25 000 households and was synthetically generated from Austrian EU-SILC survey data from 2006 using the data simulation methodology by Alfons, Kraft, Templ, and Filzmoser (2010b), which is implemented R package **simPopulation** (Alfons and Kraft 2010).

2. Application of different simulation designs to EU-SILC

First, the required packages and the data set need to be loaded.

```
R> library("simFrame")
R> library("laeken")
R> data("eusilcP")
```

Then, the function to be run in every iteration is defined. Its argument **k** determines the number of households whose income is modeled by a Pareto distribution. Since the Gini coefficient is calculated based on an equivalized household income, all individuals of a household in the upper tail receive the same value.

```
R> sim <- function(x, k) {
+   x <- x[!is.na(x$eqIncome), ]
+   g <- gini(x$eqIncome, x$.weight)$value
+   eqIncHill <- fitPareto(x$eqIncome, k = k, method = "thetaHill",
+     groups = x$hid)
+   gHill <- gini(eqIncHill, x$.weight)$value
+   eqIncPDC <- fitPareto(x$eqIncome, k = k, method = "thetaPDC",
+     groups = x$hid)
+   gPDC <- gini(eqIncPDC, x$.weight)$value
+   c(standard = g, Hill = gHill, PDC = gPDC)
+ }
```

This function is used in the following examples, which are designed to exhibit the strengths of the framework. In order to change from one simulation design to another, all there is to do is to define or modify control objects and supply them to the function **runSimulation()**.

2.1. Basic simulation design

In this basic simulation design, 100 samples of 1500 households are drawn using simple random sampling. Note that the **setup()** function is not used to permanently store the samples in an object. This is simply not necessary, since the population is rather small and the sampling method is straightforward. Furthermore, the Pareto distribution is fitted to the 175 households with the largest equivalized income.

```
R> set.seed(12345)
R> sc <- SampleControl(grouping = "hid", size = 1500, k = 100)
R> results <- runSimulation(eusilcP, sc, fun = sim, k = 175)
```

In order to inspect the simulation results, methods for several frequently used generic functions are implemented. Besides `head()`, `tail()` and `summary()` methods, a method for computing summary statistics with `aggregate()` is available. By default, the mean is used as summary statistic. Moreover, the `plot()` method selects a suitable graphical representation of the simulation results automatically. A reference line for the true value can thereby be added as well.

```
R> head(results)
```

	Run	Sample	standard	Hill	PDC
1	1	1	26.56793	26.48025	25.66614
2	2	2	26.98203	27.73124	26.39318
3	3	3	27.07081	27.11886	25.52524
4	4	4	26.86841	27.70216	25.71355
5	5	5	26.43215	26.49267	25.64191
6	6	6	26.96175	27.13876	27.17536

```
R> aggregate(results)
```

standard	Hill	PDC
26.65621	26.79016	26.89564

```
R> tv <- gini(eusilcP$eqIncome)$value
R> plot(results, true = tv)
```

Figure 1 shows the resulting box plots of the simulation results for the basic simulation design. While the PDC estimator comes with larger variability, all three methods are on average quite close to the true population value. This is also an indication that the choice of the number of households for fitting the Pareto distribution is suitable.

2.2. Using stratified sampling

The most frequently used sampling designs in official statistics are implemented in **simFrame**. In order to switch to another sampling design, only the corresponding control object needs to be changed. In this example, stratified sampling by region is performed. The sample sizes for the different strata are specified by using a vector for the slot `size` of the control object.

```
R> set.seed(12345)
R> sc <- SampleControl(design = "region", grouping = "hid",
+   size = c(75, 250, 250, 125, 200, 225, 125, 150, 100),
+   k = 100)
R> results <- runSimulation(eusilcP, sc, fun = sim, k = 175)
```

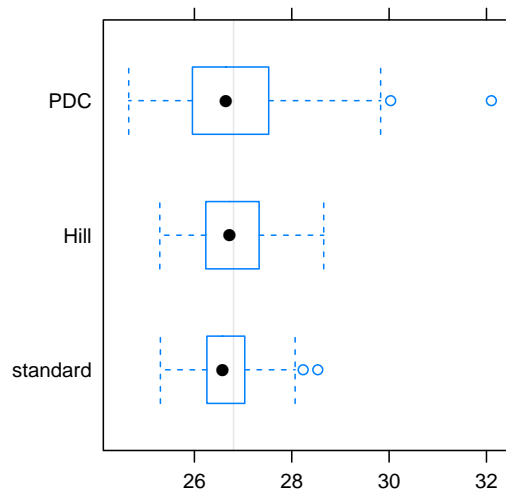


Figure 1: Simulation results for the basic simulation design.

As before, the simulation results are inspected with `head()` and `aggregate()`. A plot of the simulation results is produced as well.

```
R> head(results)
```

	Run	Sample	standard	Hill	PDC
1	1	1	27.08652	27.22293	27.66753
2	2	2	26.80670	27.35874	25.93378
3	3	3	26.68113	27.03964	26.60062
4	4	4	25.84734	26.52346	25.18298
5	5	5	26.05449	26.26848	26.60331
6	6	6	26.98439	27.01396	26.48090

```
R> aggregate(results)
```

standard	Hill	PDC
26.71792	26.85375	26.86248

```
R> tv <- gini(eusilcP$eqIncome)$value
```

```
R> plot(results, true = tv)
```

Figure 2 contains the plot of the simulation results for the simulation design with stratified sampling. The results are very similar to those from the basic simulation design with simple random sampling. On average, all three investigated methods are quite close to the true population value.

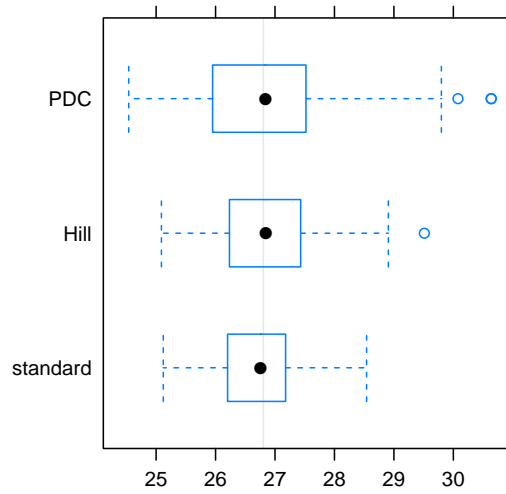


Figure 2: Simulation results for the simulation design with stratified sampling.

2.3. Adding contamination

When evaluating robust methods in simulation studies, contamination needs to be added to the data to study the influence of these outliers on the robust estimators and their classical counterparts. In **simFrame**, contamination is specified by defining a control object. Various contamination models are thereby implemented in the framework. Keep in mind that the term *contamination* is used in a technical sense here (see Alfons *et al.* 2010d; Alfons, Templ, and Filzmoser 2010c, for an exact definition) and that contamination is modeled as a two step process (see also Béguin and Hulliger 2008; Hulliger and Schoch 2009). In this example, 0.5% of the households are selected to be contaminated using simple random sampling. The equivalized income of the selected households is then drawn from a normal distribution with mean $\mu = 500\,000$ and standard deviation $\sigma = 10\,000$.

```
R> set.seed(12345)
R> sc <- SampleControl(design = "region", grouping = "hid",
+   size = c(75, 250, 250, 125, 200, 225, 125, 150, 100),
+   k = 100)
R> cc <- DCARContControl(target = "eqIncome", epsilon = 0.005,
+   grouping = "hid", dots = list(mean = 5e+05, sd = 10000))
R> results <- runSimulation(eusilcP, sc, contControl = cc,
+   fun = sim, k = 175)
```

The `head()`, `aggregate()` and `plot()` methods are again used to take a look at the simulation results. Note that a column is added that indicates the contamination level used.

```
R> head(results)
```

Run	Sample	Epsilon	standard	Hill	PDC
1	75	0.005	26.5	27.0	27.0
2	250	0.005	26.5	27.0	27.0
3	250	0.005	26.5	27.0	27.0
4	125	0.005	26.5	27.0	27.0
5	200	0.005	26.5	27.0	27.0
6	225	0.005	26.5	27.0	27.0
7	125	0.005	26.5	27.0	27.0
8	150	0.005	26.5	27.0	27.0
9	100	0.005	26.5	27.0	27.0

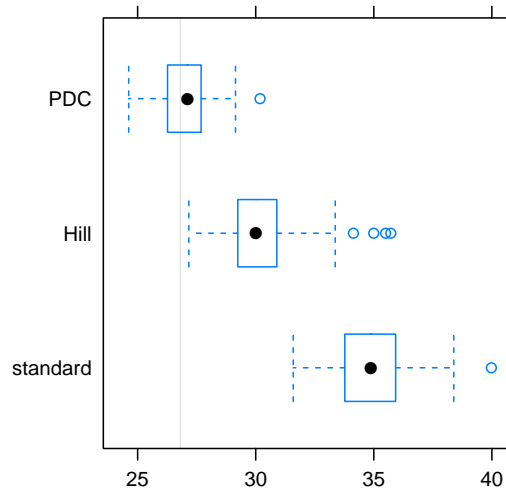


Figure 3: Simulation results for the simulation design with stratified sampling and contamination.

```

1  1      1  0.005 32.71453 29.12110 27.03731
2  2      2  0.005 34.22065 31.62709 26.24857
3  3      3  0.005 33.56878 28.49760 28.00937
4  4      4  0.005 35.26346 29.57160 26.25621
5  5      5  0.005 33.79720 29.15945 25.61514
6  6      6  0.005 34.72069 28.58610 27.22342

```

```
R> aggregate(results)
```

```

      Epsilon standard      Hill      PDC
1  0.005 34.88922 30.26179 27.02093

```

```

R> tv <- gini(eusilcP$eqIncome)$value
R> plot(results, true = tv)

```

In Figure 3, the resulting box plots are presented. The figure shows that such a small amount of contamination is enough to completely corrupt the standard estimation of the Gini coefficient. Using the classical Hill estimator to fit the Pareto distribution is still highly influenced by the outliers, whereas the PDC estimator leads to very accurate results.

2.4. Performing simulations separately on different domains

Data sets from official statistics typically contain strong heterogeneities, therefore indicators are usually computed for subsets of the data as well. Hence it is often of interest to investigate the behavior of indicators on different subsets in simulation studies. In **simFrame**, this can

be done by simply specifying the `design` argument of the function `runSimulation()`. In the case of extending the example from the previous section, the framework then splits the samples, inserts contamination into each subset and calls the supplied function for these subsets automatically. With bespoke R code, the user would need to take care of this with a loop-like structure such as a `for` loop or a function from the `apply` family.

In the following example, the simulations are performed separately for each gender. It should be noted that the value of `k` for the Pareto distribution is thus changed to 125. This is the same as Example 6.1 from Alfons *et al.* (2010d), except that a control object for sampling is supplied to `runSimulation()` instead of setting up the samples beforehand and storing them in an object.

```
R> set.seed(12345)
R> sc <- SampleControl(design = "region", grouping = "hid",
+   size = c(75, 250, 250, 125, 200, 225, 125, 150, 100),
+   k = 100)
R> cc <- DCARContControl(target = "eqIncome", epsilon = 0.005,
+   grouping = "hid", dots = list(mean = 5e+05, sd = 10000))
R> results <- runSimulation(eusilcP, sc, contControl = cc,
+   design = "gender", fun = sim, k = 125)
```

Below, the results are inspected using `head()` and `aggregate()`. The `aggregate()` method thereby computes the summary statistic for each subset automatically. Also the `plot()` method displays the results for the different subsets in different panels by taking advantage of the `lattice` system (Sarkar 2008, 2010). In order to compute the true values for each subset, the function `simSapply()` is used.

```
R> head(results)
```

	Run	Sample	Epsilon	gender	standard	Hill	PDC
1	1	1	0.005	male	34.58446	29.96658	26.61415
2	1	1	0.005	female	38.82356	33.93700	28.82045
3	2	2	0.005	male	34.34853	29.09325	27.66380
4	2	2	0.005	female	36.38429	30.06097	27.42663
5	3	3	0.005	male	33.39992	30.54211	23.96698
6	3	3	0.005	female	35.12883	30.51336	26.06518

```
R> aggregate(results)
```

	Epsilon	gender	standard	Hill	PDC
1	0.005	male	33.18580	29.00265	26.21119
2	0.005	female	35.61341	31.28984	27.69054

```
R> tv <- simSapply(eusilcP, "gender", function(x) gini(x$eqIncome)$value)
R> plot(results, true = tv)
```

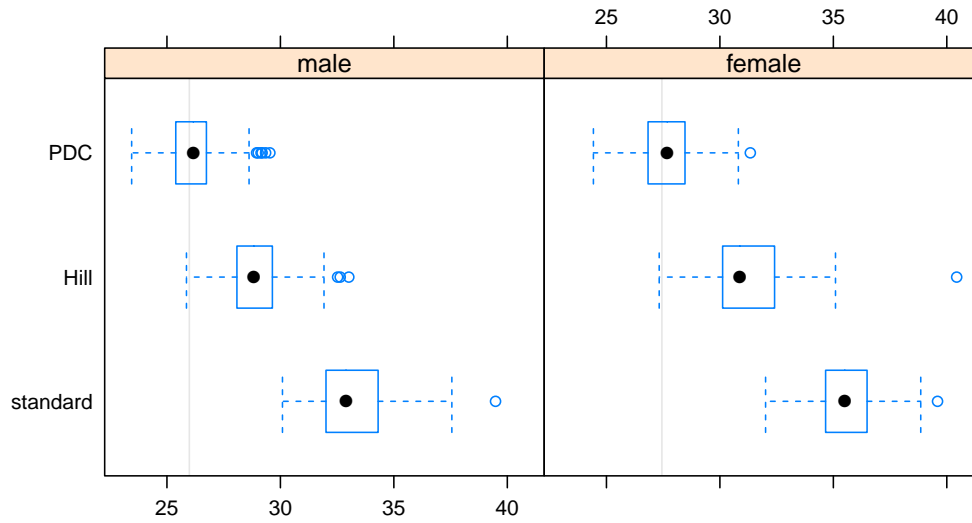


Figure 4: Simulation results for the simulation design with stratified sampling, contamination and performing the simulations separately for each gender.

The resulting plots are shown in Figure 4, which is the same as Figure 2 in [Alfons *et al.* \(2010d\)](#). Clearly, the PDC estimator leads to excellent results for both subsets, while the two classical approaches are in both cases highly influenced by the outliers.

2.5. Using multiple contamination levels

To get a more complete picture of the behavior of robust methods, more than one level of contamination is typically investigated in simulation studies. The only necessary modification of the code is to use a vector of contamination levels as the slot `epsilon` of the contamination control object. In this example, the contamination level is varied from 0% to 1% in steps of 0.25%. With bespoke R code, the user would have to add another loop-like structure to the code and collect the results in a suitable data structure. In **simFrame**, this is handled internally by the framework.

```
R> set.seed(12345)
R> sc <- SampleControl(design = "region", grouping = "hid",
+   size = c(75, 250, 250, 125, 200, 225, 125, 150, 100),
+   k = 100)
R> cc <- DCARContControl(target = "eqIncome", epsilon = c(0,
+   0.0025, 0.005, 0.0075, 0.01), dots = list(mean = 5e+05,
+   sd = 10000))
R> results <- runSimulation(eusilcP, sc, contControl = cc,
+   design = "gender", fun = sim, k = 125)
```

The simulation results are inspected as usual. Note that the `aggregate()` method in this case returns values for each combination of contamination level and gender.


```
R> head(results)
```

	Run	Sample	Epsilon	gender	standard	Hill	PDC
1	1	1	0.0000	male	26.58067	26.50425	26.35969
2	1	1	0.0000	female	27.43355	27.03526	28.16992
3	2	1	0.0025	male	31.63593	29.23365	27.12430
4	2	1	0.0025	female	31.43540	27.77698	26.85896
5	3	1	0.0050	male	33.35950	31.07040	25.97415
6	3	1	0.0050	female	35.68710	34.03560	29.11359

```
R> aggregate(results)
```

	Epsilon	gender	standard	Hill	PDC
1	0.0000	male	25.94937	26.00769	25.85311
2	0.0025	male	30.44448	27.70155	26.01033
3	0.0050	male	33.54929	29.13202	26.16786
4	0.0075	male	36.76641	31.32342	26.49026
5	0.0100	male	39.42281	33.67944	26.53749
6	0.0000	female	27.30171	27.49442	27.41323
7	0.0025	female	31.68505	29.13643	27.61790
8	0.0050	female	35.49976	30.92128	27.91607
9	0.0075	female	38.51819	33.08778	28.09784
10	0.0100	female	41.47137	35.32935	27.97407

```
R> tv <- simSapply(eusilcP, "gender", function(x) gini(x$eqIncome)$value)
R> plot(results, true = tv)
```

If multiple contamination levels are used in a simulation study, the `plot()` method for the simulation results no longer produces box plots. Instead, the average results are plotted against the corresponding contamination levels, as shown in Figure 5. The plots show how the classical estimators move away from the references line as the contamination level increases, while the values obtained with the PDC estimator remain quite accurate.

2.6. Inserting missing values

Survey data almost always contain a considerable amount of missing values. In close-to-reality simulation studies, the variability due to missing data therefore needs to be considered. Three types of missing data mechanisms are commonly distinguished in the literature (e.g., [Little and Rubin 2002](#)): missing completely at random (MCAR), missing at random (MAR) and missing not at random (MNAR). All three missing data mechanisms are implemented in the framework.

In the following example, missing values are inserted into the equivalized household income of non-contaminated households with MCAR, i.e., the households whose values are going to be set to NA are selected using simple random sampling. In order to compare the scenario without missing values to a scenario with missing values, the missing value rates 0% and 5% are used. In the latter case, the missing values are simply disregarded for fitting the Pareto distribution

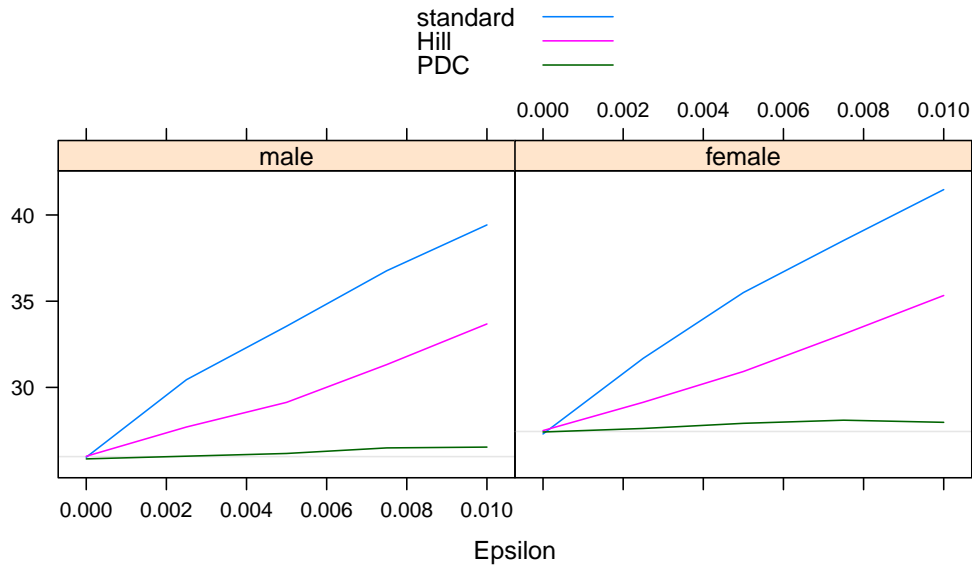


Figure 5: Simulation results for the simulation design with stratified sampling, multiple contamination levels and performing the simulations separately for each gender.

and estimating the Gini coefficient. Furthermore, the number of samples is reduced to 50 and only the contamination levels 0%, 0.5% and 1% are investigated to keep the computation time of this motivational example low.

With **simFrame**, only a control object for missing data needs to be defined and supplied to `runSimulation()`, the rest is done automatically by the framework. To apply these changes to a simulation study implemented with bespoke R code, yet another loop-like structure for the different missing value rates as well as changes in the data structure for the simulation results would be necessary.

```
R> set.seed(12345)
R> sc <- SampleControl(design = "region", grouping = "hid",
+   size = c(75, 250, 250, 125, 200, 225, 125, 150, 100),
+   k = 50)
R> cc <- DCARContControl(target = "eqIncome", epsilon = c(0,
+   0.005, 0.01), dots = list(mean = 5e+05, sd = 10000))
R> nc <- NAControl(target = "eqIncome", NARate = c(0, 0.05))
R> results <- runSimulation(eusilcP, sc, contControl = cc,
+   NAControl = nc, design = "gender", fun = sim, k = 125)
```

As always, the `head()`, `aggregate()` and `plot()` methods are used to take a look at the simulation results. It should be noted that a column is added to the results that indicates the missing value rate used and that `aggregate()` in this example returns a value for each combination of contamination level, missing value rate and gender.

```
R> head(results)
```

	Run	Sample	Epsilon	NARate	gender	standard	Hill	PDC
1	1	1	0.000	0.00	male	26.58067	27.00998	26.26273
2	1	1	0.000	0.00	female	27.43355	27.92305	26.69034
3	2	1	0.000	0.05	male	26.62313	26.54198	26.01043
4	2	1	0.000	0.05	female	27.51209	26.83574	27.25464
5	3	1	0.005	0.00	male	33.71363	28.44824	26.46635
6	3	1	0.005	0.00	female	35.47508	28.48208	27.70783

```
R> aggregate(results)
```

	Epsilon	NARate	gender	standard	Hill	PDC
1	0.000	0.00	male	25.89948	25.99777	25.74944
2	0.005	0.00	male	33.52791	29.30477	26.14659
3	0.010	0.00	male	39.45422	32.74672	26.64929
4	0.000	0.05	male	25.88434	25.87824	25.80541
5	0.005	0.05	male	33.87975	29.60079	26.18759
6	0.010	0.05	male	39.99526	33.44462	26.31274
7	0.000	0.00	female	27.17769	27.30586	27.19275
8	0.005	0.00	female	35.46414	31.37099	27.98622
9	0.010	0.00	female	41.28625	35.22113	28.19677
10	0.000	0.05	female	27.16026	27.37710	27.20892
11	0.005	0.05	female	35.85305	31.56317	27.80455
12	0.010	0.05	female	41.86453	35.44025	27.98948

```
R> tv <- simSapply(eusilcP, "gender", function(x) gini(x$eqIncome)$value)
R> plot(results, true = tv)
```

If multiple contamination levels and multiple missing value rates are used in the simulation study, conditional plots are produced by the `plot()` method for the simulation results. Figure 6 shows the resulting plots for this example. The bottom panels illustrate the scenario without missing values, while the scenario with 5% missing values is displayed in the top panels. In this case, there is not much of a difference in the results for the two scenarios.

2.7. Parallel computing

Statistical simulation is an *embarrassingly parallel* procedure, hence parallel computing can drastically reduce the computational costs. In **simFrame**, parallel computing is implemented using **snow** (Rossini, Tierney, and Li 2007; Tierney, Rossini, Li, and Sevcikova 2008). Only minimal additional programming effort due to the use of **snow** is required to adapt the code from the previous example: to initialize the computer cluster, to ensure that all packages and objects are available on each worker process, to use the function `clusterRunSimulation()` instead of `runSimulation()` and to stop the computer cluster after the simulations. In addition, random number streams (e.g., L'Ecuyer, Simard, Chen, and Kelton 2002; Sevcikova and Rossini 2009) should be used instead of the built-in random number generator.

```
R> cl <- makeCluster(4, type = "SOCK")
R> clusterEvalQ(cl, {
```

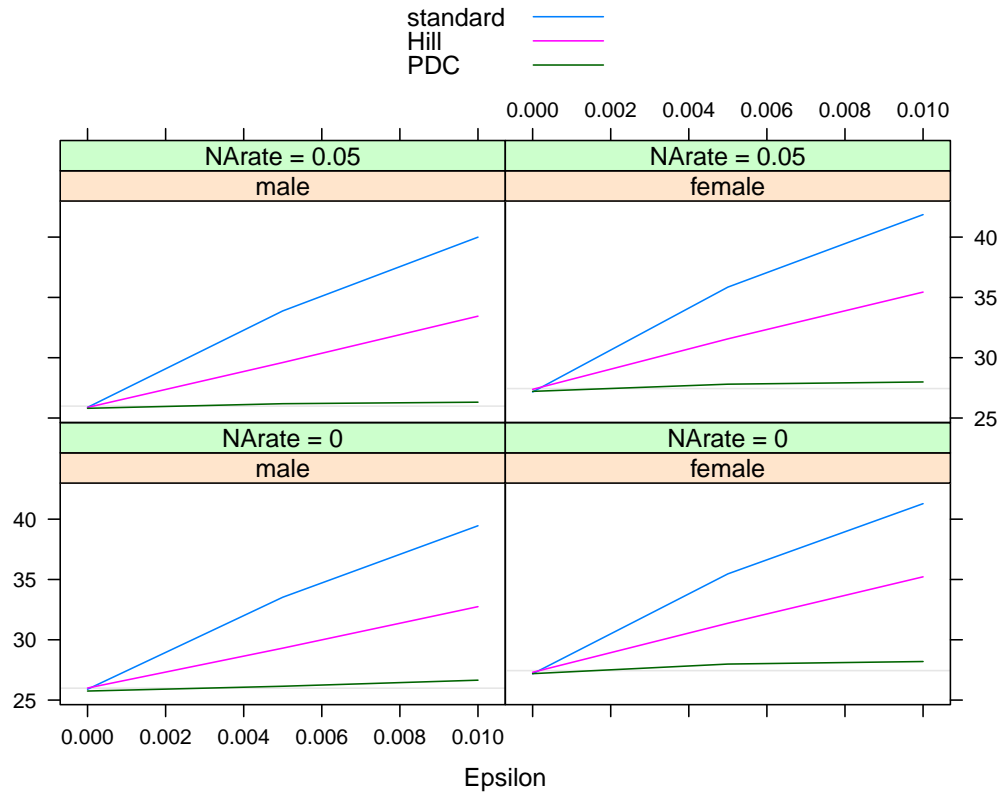


Figure 6: Simulation results for the simulation design with stratified sampling, multiple contamination levels, multiple missing value rates and performing the simulations separately for each gender.

```
+ library("simFrame")
+ library("laeken")
+ data("eusilcP")
+ })
R> clusterSetupRNG(cl, seed = 12345)
R> sc <- SampleControl(design = "region", grouping = "hid",
+ size = c(75, 250, 250, 125, 200, 225, 125, 150, 100),
+ k = 50)
R> cc <- DCARContControl(target = "eqIncome", epsilon = c(0,
+ 0.005, 0.01), dots = list(mean = 5e+05, sd = 10000))
R> nc <- NAControl(target = "eqIncome", NArate = c(0, 0.05))
R> clusterExport(cl, c("sc", "cc", "nc", "sim"))
R> results <- clusterRunSimulation(cl, eusilcP, sc, contControl = cc,
+ NAControl = nc, design = "gender", fun = sim, k = 125)
R> stopCluster(cl)
```

When the parallel computations are finished and the simulation results are obtained, they can be inspected as usual.

```
R> head(results)
```

	Run	Sample	Epsilon	NArate	gender	standard	Hill	PDC
1	1	1	0.000	0.00	male	26.20067	27.02017	23.66565
2	1	1	0.000	0.00	female	28.79194	29.23548	27.12933
3	2	1	0.000	0.05	male	26.19328	24.91570	24.07906
4	2	1	0.000	0.05	female	28.86860	27.38585	27.80012
5	3	1	0.005	0.00	male	34.46084	31.74470	24.87023
6	3	1	0.005	0.00	female	36.27429	32.14269	28.06137

```
R> aggregate(results)
```

	Epsilon	NArate	gender	standard	Hill	PDC
1	0.000	0.00	male	25.89996	25.98977	25.86451
2	0.005	0.00	male	33.56743	29.36361	26.39515
3	0.010	0.00	male	39.40362	33.05926	26.68715
4	0.000	0.05	male	25.87909	25.86055	26.00109
5	0.005	0.05	male	33.94829	29.65456	26.32813
6	0.010	0.05	male	39.95535	33.24853	26.78947
7	0.000	0.00	female	27.38636	27.52210	27.48816
8	0.005	0.00	female	35.52688	31.30099	28.03385
9	0.010	0.00	female	41.35311	35.81549	28.67901
10	0.000	0.05	female	27.38459	27.51825	27.54063
11	0.005	0.05	female	35.87991	31.74678	28.18308
12	0.010	0.05	female	41.89804	36.21921	28.41367

```
R> tv <- simSapply(eusilcP, "gender", function(x) gini(x$eqIncome)$value)
R> plot(results, true = tv)
```

Figure 7 shows the simulation results obtained with parallel computing. The plots are, of course, very similar to the plots for the previous example in Figure 6, since the design of the simulation studies is the same.

3. Conclusions

In this paper, the use of the R package **simFrame** for different simulation designs has been demonstrated in a practical application. The full functionality of the framework for design-based simulation has been presented in various code examples. These examples showed that the framework allows researchers to make use of a wide range of simulation designs with only a few lines of code. In order to switch from one simulation design to another, only control objects need to be defined or modified. Even moving from basic to highly complex designs therefore requires only minimal changes to the code. With bespoke R code, such modifications would often need a considerable amount of programming. Furthermore, parallel computing with **simFrame** can easily be done based on package **snow**.

Besides the functionality for carrying out simulation studies, methods for several frequently used generic functions are available for inspecting or summarizing the simulation results. Most

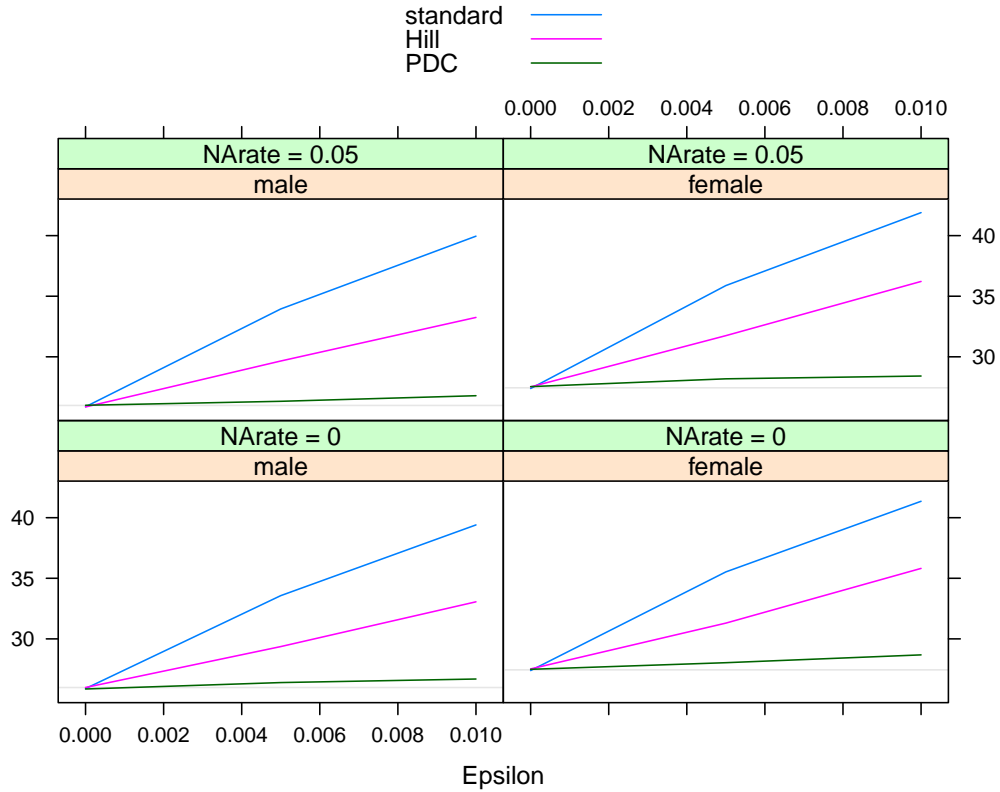


Figure 7: Simulation results obtained by parallel computing for the simulation design with stratified sampling, multiple contamination levels, multiple missing value rates and performing the simulations separately for each gender.

notably, a suitable plot method of the simulation results is selected automatically depending on their structure.

Due to this flexibility, **simFrame** is widely applicable for gaining insight into the quality of statistical methods and is a valuable addition to a researcher's toolbox.

Acknowledgments

This work was partly funded by the European Union (represented by the European Commission) within the 7th framework programme for research (Theme 8, Socio-Economic Sciences and Humanities, Project AMELI (Advanced Methodology for European Laeken Indicators), Grant Agreement No. 217322). Visit <http://ameli.surveystatistics.net> for more information on the project.

References

Alfons A (2011). *simFrame: Simulation Framework*. R package version 0.4.2, URL [http:](http://)

[//CRAN.R-project.org/package=simFrame](http://CRAN.R-project.org/package=simFrame).

- Alfons A, Holzer J, Templ M (2010a). *laeken: Laeken Indicators for Measuring Social Cohesion*. R package version 0.1.3, URL <http://CRAN.R-project.org/package=laeken>.
- Alfons A, Kraft S (2010). *simPopulation: Simulation of Synthetic Populations for Surveys based on Sample Data*. R package version 0.2, URL <http://CRAN.R-project.org/package=simPopulation>.
- Alfons A, Kraft S, Templ M, Filzmoser P (2010b). “Simulation of Synthetic Population Data for Household Surveys with Application to EU-SILC.” *Research Report CS-2010-1*, Department of Statistics and Probability Theory, Vienna University of Technology. URL <http://www.statistik.tuwien.ac.at/forschung/CS/CS-2010-1complete.pdf>.
- Alfons A, Templ M, Filzmoser P (2010c). “Contamination Models in the R Package **simFrame** for Statistical Simulation.” In S Aivazian, P Filzmoser, Y Kharin (eds.), “Computer Data Analysis and Modeling: Complex Stochastic Data and Systems,” volume 2, pp. 178–181. Minsk. ISBN 978-985-476-848-9.
- Alfons A, Templ M, Filzmoser P (2010d). “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/>.
- Alfons A, Templ M, Filzmoser P, Holzer J (2010e). “A Comparison of Robust Methods for Pareto Tail Modeling in the Case of Laeken Indicators.” In C Borgelt, G González-Rodríguez, W Trutschnig, M Lubiano, M Gil, P Grzegorzewski, O Hryniewicz (eds.), “Combining Soft Computing and Statistical Methods in Data Analysis,” volume 77 of *Advances in Intelligent and Soft Computing*, pp. 17–24. Springer-Verlag, Heidelberg. ISBN 978-3-642-14745-6.
- Béguin C, Hulliger B (2008). “The BACON-EEM Algorithm for Multivariate Outlier Detection in Incomplete Survey Data.” *Survey Methodology*, **34**(1), 91–103.
- EU-SILC (2004). “Common Cross-Sectional EU Indicators based on EU-SILC; the Gender Pay Gap.” *EU-SILC 131-rev/04*, Working group on Statistics on Income and Living Conditions (EU-SILC), Eurostat, Luxembourg.
- Hill B (1975). “A Simple General Approach to Inference about the Tail of a Distribution.” *The Annals of Statistics*, **3**(5), 1163–1174.
- Hulliger B, Schoch T (2009). “Robust Multivariate Imputation with Survey Data.” 57th Session of the International Statistical Institute, Durban.
- Kleiber C, Kotz S (2003). *Statistical Size Distributions in Economics and Actuarial Sciences*. John Wiley & Sons, Hoboken. ISBN 0-471-15064-9.
- L’Ecuyer P, Simard R, Chen E, Kelton W (2002). “An Object-Oriented Random-Number Package with Many Long Streams and Substreams.” *Operations Research*, **50**(6), 1073–1075.
- Little R, Rubin D (2002). *Statistical Analysis with Missing Data*. John Wiley & Sons, New York, 2nd edition. ISBN 0-471-18386-5.

- R Development Core Team (2010). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL <http://www.R-project.org>.
- Rossini A, Tierney L, Li N (2007). “Simple Parallel Statistical Computing in R.” *Journal of Computational and Graphical Statistics*, **16**(2), 399–420.
- Sarkar D (2008). *Lattice: Multivariate Data Visualization with R*. Springer-Verlag, New York. ISBN 978-0-387-75968-5.
- Sarkar D (2010). *lattice: Lattice Graphics*. R package version 0.19-13, URL <http://CRAN.R-project.org/package=lattice>.
- Sevcikova H, Rossini T (2009). *rlecuyer: R Interface to RNG with Multiple Streams*. R package version 0.3-1, URL <http://CRAN.R-project.org/package=rlecuyer>.
- Tierney L, Rossini A, Li N, Sevcikova H (2008). *snow: Simple Network of Workstations*. R package version 0.3-3, URL <http://CRAN.R-project.org/package=snow>.
- Vandewalle B, Beirlant J, Christmann A, Hubert M (2007). “A Robust Estimator for the Tail Index of Pareto-Type Distributions.” *Computational Statistics & Data Analysis*, **51**(12), 6252–6268.

Affiliation:

Andreas Alfons
Department of Statistics and Probability Theory
Vienna University of Technology
Wiedner Hauptstraße 7
1040 Vienna, Austria
E-mail: alfons@statistik.tuwien.ac.at
URL: <http://www.statistik.tuwien.ac.at/public/alfons/>