The R Package smicd: Statistical Methods for Interval-Censored Data

Replies to the comments raised by the review process

Paul Walter

I am very grateful to the referees for their constructive comments. These have been very helpful in preparing the revised version of this paper. I have done my best to respond to all comments. The following notes explain how I addressed each comment. In the new version of the paper the revisions are in red font.

Replies to Reviewer 1

1. In the example Direct estimation of statistical indicators, since the data used (household income in the synthetic EU-SILC survey data) is a continuous variable which is then interval-censored for the purposes of demonstrating the algorithm, could the article also briefly show the indicators (mean, Gini coefficient, HCR, quantiles, PGAP and QSR) for the original continuous data? This could demonstrate the accuracy of the algorithm by comparing the results of the estimation process to the actual pre-binned data. Granted, it is quite simple for one to calculate these "actual" indicators themselves using the data from the laeken package, for example:

```
mean(hhincome_net)
## [1] 1659.241

quantile(hhincome_net, probs = c(.05,.1,.25,.5,.75,.9,.95))
## 5% 10% 25% 50% 75% 90% 95%
## 637.0923 810.1984 1121.9686 1506.8983 2016.2527 2641.3420 3148.5639
```

   However it would not take much space to include in the article and would demonstrate to the reader (perhaps without access to an R interpreter) the impressive proximity of the estimated indicators to those of the unbinned continuous data.

   Reply:

   I agree that it is a very good idea to include the estimated statistical indicators, using the continuous household income variable, for demonstration purposes. In the paper I have estimated the weighted indicators. Therefore, I have run the following code for the estimation of the weighted statistical indicators:

```
# Estimate threshold
threshold <- 0.6 * wtd.quantile(hhincome_net, weights = hhweight, probs = .5)

# Estimate indicators
# Mean
weighted.mean(hhincome_net, w = hhweight)

# Gini
gini(hhincome_net, weights = hhweight)$value/100

# HCR
```

```
arpr(hhincome_net, weights = hhweight, p = 0.6)$value / 100

# Quantiles
wtd.quantile(
  hhincome_net, weights = hhweight,
  probs = c(0.05, 0.1, 0.25, 0.5, 0.75, 0.9, 0.95)
)

# PGAP
pgap = sum(hhweight * (hhincome_net < threshold) * (threshold - hhincome_net)
           / threshold) / sum(hhweight)
pgap

# QSR
qsr(hhincome_net, weights = hhweight)$value
```

On page 6 of the paper I included the following text and Table 1:

For demonstration purposes, the statistical indicators are also estimated using the continuous household income variable from the synthetic EU-SILC data set (Table 1). The estimation results of the KDE algorithm using the interval-censored data are very close to those based on the continuous data. Slightly larger deviations are observable for the more extreme quantiles. This is due to the fact that these quantile estimates fall into intervals with a lower number of observations (compared to the other quantile estimates). Estimation results for these quantiles could potentially be further improved by increasing the number of `evalpoints` of the `kdeAlgo()`.

Table 1: Estimated weighted statistical indicators using the continuous household income variable from the synthetic EU-SILC data set.

mean	gini	hcr	quant10	quant25	quant50
1657.910	0.265	0.144	805.468	1114.028	1508.657
quant75	**quant90**	**pgap**	**qsr**	**quant05**	**quant95**
2017.585	2653.617	0.040	3.960	619.666	3153.425

My only other suggested revisions are spelling issues. While this review is not intended to be a proofreading exercise, these errors appear to be in the package source code and the example code, so I thought they should be mentioned.

2. In the call of the kdeAlgo function in the first example, the word “threshold” is misspelt as “treshold” in the custom quant05 and quant95 functions.

```
Indicators <- kdeAlgo(
xclass = c.hhincome, classes = intervals,
bootstrap.se = TRUE, custom_indicator =
list(
quant05 = function(y, treshold, weights) {
wtd.quantile(y, probs = 0.05, weights)
},
quant95 = function(y, treshold, weights) {
wtd.quantile(y, probs = 0.95, weights)
}
),
weights = hhweight
)
```

While a misspelled variable name would usually cause greater problems, I understand this causes no issues with the code as the threshold default is not needed for these custom indicators although it seems to be required to be included in their definition.

Reply: I have corrected the spelling mistakes in the example code on page 5-6 in the paper. As you pointed out, the misspelling did not cause and error in this example (even though specifying the argument is necessary) because the chosen custom indicators do not depend on a threshold. However, if a custom indicator does depend on a threshold, the correct spelling is crucial.

3. The word "iteration" in the label of the y-axis in the convergence plots is misspelt as "iterstion". This appears to be in the source code of the smicd package's plot.kdeAlgo function:

```
plot(point,
xlim = c(0, (x$burnin + x$samples)),
col = ifelse(point == means[1] - .Machine$double.xmin, "white", "black"),
xlab = "Iteration step", ylab = "Average up to iterstion step M"
)
```

Reply: I have corrected the spelling mistakes in the source code of the smicd package in function `plot.kde()` and `plot.sem()` (where it was also misspelled). Furthermore, I have corrected the spelling mistakes in the plots produced for the paper on page 7 and page 12.

Replies to Reviewer 2

1. The library is unusually very slow in it's implementation. The example included in the submission, it took 3.035799 hours to run the kdeAlgo() on a windows laptop with 8G memory. This is very slow, if the author can take advantage of parallel processing or the Rcpp libraries to reduce the computational time in this package.

   Reply: The implementation is slow primarily because of two reasons:

   (a) The KDE algorithm needs a certain number of iterations until the estimation results have stabilized. These iterations need to run sequentially, because each consecutive run is based on the estimation results of the previous run. Running this in parallel is therefore, up to my knowledge, not possible.

   (b) In the example code also the standard errors are estimated. This is done by a non-parametric bootstrap, which could be parallelized in future versions of the package. And I totally agree that this should be considered for future versions of the package, as it is pointed out in the section "Discussion and outlook".

   However, estimation time can also easily be reduced by lowering the number of iteration steps of the KDE algorithm. In the convergence plots of the paper it can be seen that the estimation results converge rather quickly and changes in the final estimate are very marginal after a certain number of iterations. For the paper I propose the following short paragraph that I have included in the paper on page 7:

   Lastly, it should be mentioned that the computation time can be very long if the estimation of the standard errors is enabled. Hence, if the estimation of the standard errors is not required, the argument `bootstrap.se` should be set to `FALSE`. Furthermore, it should always be checked how many iterations are needed for the desired statistical indicator to converge. Reducing the number of required iterations (arguments `burnin` and `samples`) lowers the computation time significantly (with and without standard errors).

2. The author should consider making the code for the package available on a repository such as github. This would allow the code to be critically analyzed and contributions from the users on how to improve the package can also be made.

Reply: The code is available under on GitHub. I have added the following text to the paper on page 1 to make it clear to the reader where to find the code:

The package code and the open source contribution guidelines for the package are available on GitHub. Potential code contributions, feature requests and bugs can be reported there by creating issues.

3. A development site and bug tracker such as github would significantly enhance the user experience when an error is detected.

   Reply: Bugs and feature requests can now be reported on the GitHub page. I have also included a comment in the paper to point the reader to the repository (see reply to comment 2).

4. This repository should also include open source contribution guidelines.

   Reply: I have included the open source contribution guidelines on GitHub. For the guidelines I used the following open source contribution template and adjusted it to my needs: Contribution template.

   I have also included a comment in the paper to point the reader to the repository (see reply to comment 2).