povmap: Extension to the emdi package for small area estimation

Ifeanyi Edochie
World Bank Group

David Newhouse
World Bank Group

Timo Schmid
Otto Friedrich
University Bamberg

Nora Würz
Otto Friedrich
University Bamberg

Abstract

The R package povmap is designed to facilitate the production of small area estimates of means and poverty headcount rates. It adds several new features to the emdi package. These include new options for incorporating survey weights, ex-post benchmarking of estimates, two additional transformations, several new convenient functions to assist with reporting results, and a wrapper function to facilitate access from Stata.

Keywords: official statistics, survey statistics, small area estimation.

1. Introduction

The povmap package adds new features to the emdi package (Kreutzmann et al. (2019)) that are particularly appropriate for estimating headcount poverty rates and means. The name of the package is taken from the C-based Povmap software frequently used to generate poverty maps (Zhao (2006)). This vignette provides an overview of the additional features provided by povmap. The package adds two notable new features to the emdi package. The first is the ability to incorporate sample weights through the nlme package, which enables the use of weights when undertaking data-driven transformations. While initial explorations suggest that using the two types of weights give similar results, more exploration and investigation is needed to fully explore the use of nlme weights in the context of Empirical Best Predictor (EBP) models. The second main new feature is the incorporation of benchmarking. In response to the need for benchmarking in small area estimation, the ebp() function now includes additional arguments (benchmark, benchmark_level, and benchmark_type). These additions allow users to address both external and internal benchmarking problems for the mean and the head count ratio. Benchmarking, if selected, is also applied in the parametric bootstrap procedure used to estimate Mean Squared Error. Two additional options are provided to transform the dependent variable, in order to better meet the normality assumptions of the underlying model. Finally, several minor changes and functionalities are introduced that can be highly beneficial for practitioners, including several function to assist with reporting results and a convenient wrapper for Stata users.

2. Survey weights - new options

The standard version of the EBP assumes non-informative sampling, which means that the
inclusion probability of the sample is not linked to the outcome variable of interest. In most practical applications, informative sampling is present, and it is important to allow for weights when estimating the EBP model. In the `emdi` package, the argument `weights` gives users the possibility to include weights. The method of Guadarrama et al. (2018) is used and is implemented in the R package `emdi` (Skarke et al. 2023).

2.1. Survey weights with the `povmap` package

The `povmap` package offers, in addition to the methodology of Guadarrama et al. (2018), the possibility to adjust for informative sampling via the `weights` argument in the `nlme` package (Pinheiro et al. 2015). The well-known `nlme` package allows the estimation of linear mixed models via the function `lme` and is used in both the `emdi` and `povmap` package for the estimation of EBP models, which are special cases of a linear mixed model. The `weights` argument in the `lme` command provides an alternative method to incorporate survey weights to adjust for informative sampling. The `povmap` package now allows users to include weights via the `lme` function. There are two different possibilities to use the `nlme` package in this context: (1) to include weights when estimating the model parameters for the EBP (cf. step 2 in Kreutzmann et al. (2019) on page 7) or (2) when using data-driven transformations, using the weights both to select the optimal transformation parameter and for estimating the model (cf. step 1 in Kreutzmann et al. (2019) on page 7). This selection is now enabled with the argument `weights_type` in the `povmap` package. The default are the inclusion of weights following Guadarrama et al. (2018) ("Guadarrama"). If "nlme" is selected, weights are included within the `lme` function for estimating the linear-mixed model for the EBP. If "nlme_lambda" is selected, weights are included both when estimating the linear mixed model and when estimating the data-driven parameter for the transformation. In both cases, each residual is assumed to have variance equal to the inverse of the weight for that observation, so that each observation is weighted using its specified weight.

For all three weight options, the estimated shrinkage factor $\gamma_{iw}$ takes the weights into account, using the formula:

$$\gamma_{iw} = \frac{\hat{\sigma}_u^2}{\hat{\sigma}_u^2 + \hat{\sigma}_\varepsilon^2 \delta_{iw}^2}$$

where

$$\delta_{iw}^2 = \frac{\sum_{j \in i} w_j^2}{\left(\sum_{j \in i} w_j\right)^2}$$

and $\hat{\sigma}_u^2$ and $\hat{\sigma}_\varepsilon^2$ are the estimated variance components of the area effect and idiosyncratic error term. $w_j$ is the weight assigned to unit $j$ located in target area $i$. When no weights are specified, $w_j = 1$ for all units $j$, and this reduces to:

$$\gamma_{iw} = \frac{\hat{\sigma}_u^2}{\hat{\sigma}_u^2 + \frac{\hat{\sigma}_\varepsilon^2}{N_i}}$$

Where $N_i$ is the number of units in target area $i$. The main advantage of the `nmle`-type integration of informative sampling as implemented in the `povmap` package is that it is compatible with all transformations. In contrast, the version with the "Guadarrama"-weights is
only compatible with no transformation or the log transformation. A second difference be-
tween the methods is that when using nlme weights, the variance component estimates \( \hat{\sigma}_u^2 \)
and \( \hat{\sigma}_\epsilon^2 \) are estimated using a weighted linear mixed model.

2.2. Functionality

Model estimation

To demonstrate the functionalities of the packages we show all three types of weights. First, load the data.

R> library("povmap")
R> # Load sample data set
R> data("eusilcA_smp")
R> data('eusilcA_pop')

For comparison, we will first show the Guadarrama et al. (2018) informative sampling under the
log transformation and also perform the nlme-version with log transformation.

R> emdi_model_Guadarrama <- ebp(
+ fixed = eqIncome ~ gender + eqsize + cash + self_empl +
+ unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent +
+ fam_allow + house_allow + cap_inv + tax_adj,
+ pop_data = eusilcA_pop, pop_domains = "district",
+ smp_data = eusilcA_smp, smp_domains = "district", weights = "weight",
+ weights_type = "Guadarrama", transformation = "log", na.rm = TRUE
+ )

R> emdi_model_nlme_log <- ebp(
+ fixed = eqIncome ~ gender + eqsize + cash + self_empl +
+ unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent +
+ fam_allow + house_allow + cap_inv + tax_adj,
+ pop_data = eusilcA_pop, pop_domains = "district",
+ smp_data = eusilcA_smp, smp_domains = "district", weights = "weight",
+ weights_type = "nlme", transformation = "log", na.rm = TRUE
+ )

We further want to compare the two nlme-versions under the box-cox transformation. weights_type
'Guadarrama' can only be selected with log and no transformation. Thus, under the Box-Cox
transformation, only the weights_type 'nlme' or 'nlme_lambda' is possible.

R> emdi_model_nlme_bc <- ebp(
+ fixed = eqIncome ~ gender + eqsize + cash + self_empl +
+ unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent +
+ fam_allow + house_allow + cap_inv + tax_adj,
+ pop_data = eusilcA_pop, pop_domains = "district",
+ smp_data = eusilcA_smp, smp_domains = "district", weights = "weight",
+ weights_type = "nlme", transformation = "log", na.rm = TRUE
+ )
povmap: Extension to the emdi package for small area estimation

```r
+ smp_data = eusilcA_smp, smp_domains = "district",
+ weights = "weight", weights_type = "nlme", na.rm = TRUE
+
R> emdi_model_nlme_lambda <- ebp(
+ fixed = eqIncome ~ gender + eqsize + cash + self_empl +
+ unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent +
+ fam_allow + house_allow + cap_inv + tax_adj,
+ pop_data = eusilcA_pop, pop_domains = "district",
+ smp_data = eusilcA_smp, smp_domains = "district",
+ weights = "weight", weights_type = "nlme_lambda", na.rm = TRUE
+
R> head(estimators(emdi_model_Guadarrama, indicator = "Mean"))

<table>
<thead>
<tr>
<th>Domain</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eisenstadt-Umgebung</td>
<td>30926.73</td>
</tr>
<tr>
<td>Eisenstadt (Stadt)</td>
<td>94261.32</td>
</tr>
<tr>
<td>Güssing</td>
<td>17008.43</td>
</tr>
<tr>
<td>Jennersdorf</td>
<td>13281.90</td>
</tr>
<tr>
<td>Mattersburg</td>
<td>21830.83</td>
</tr>
<tr>
<td>Neusiedl am See</td>
<td>19492.90</td>
</tr>
</tbody>
</table>

R> head(estimators(emdi_model_nlme_log, indicator = "Mean"))

<table>
<thead>
<tr>
<th>Domain</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eisenstadt-Umgebung</td>
<td>31071.60</td>
</tr>
<tr>
<td>Eisenstadt (Stadt)</td>
<td>98082.34</td>
</tr>
<tr>
<td>Güssing</td>
<td>16990.86</td>
</tr>
<tr>
<td>Jennersdorf</td>
<td>13263.20</td>
</tr>
<tr>
<td>Mattersburg</td>
<td>21922.16</td>
</tr>
<tr>
<td>Neusiedl am See</td>
<td>19443.56</td>
</tr>
</tbody>
</table>

A comparison between the two different approaches to weighting is possible under the log transformation. In the example here, the two differ only very slightly with a median relative bias of 0.34%.

And now the results under box Cox transformation:

R> head(estimators(emdi_model_nlme_bc, indicator = "Mean"))

<table>
<thead>
<tr>
<th>Domain</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eisenstadt-Umgebung</td>
<td>28253.88</td>
</tr>
</tbody>
</table>
```
A more detailed investigation of which specification is to be recommended requires extensive simulation studies. Nevertheless, the results differ only slightly, which is reflected in a low median of the relative bias of 0.05%.

3. Benchmarking

In small area estimation, the model-based small area estimates need not match the direct survey estimate for a higher area. This can be concerning if the sample size for the higher area is large enough that the direct estimate is considered reliable and has official status. To address these issues, benchmarking is used, which involves calibrating individual area-level estimates so that they aggregate to match the direct estimates for a higher area.

There are two types of benchmarking problems: external and internal. External benchmarking calibrates survey estimates to match estimates from external data sources. Internal benchmarking involves calibrating small area estimates to higher-level aggregates, such as regional or national totals obtained from the same survey.

Until now, benchmarking has been offered in the `emdi` package for the `fh()` function. The function `ebp()` now includes the additional arguments `benchmark` to enter an external benchmark value, `benchmark_level` to set the level for benchmarking, and `benchmark_type` to specify the type of benchmarking and `benchmark_weights` to allow users to specify weights to calculate benchmarking that differ from the survey weights.

3.1. Methodology

The idea of benchmarking is that the aggregated small area estimates from the EBP, weighted by population, should sum up to estimates of a higher regional level that are assumed to be reliable (Datta et al. 2011; Bell et al. 2013; Pfeffermann et al. 2014). If this value is one global one ($\tau$) it follows

$$\sum_{i=1}^{D} \xi_i \hat{y}_{i}^{\text{bench}} = \tau,$$
where $\xi_i$ stands for the share of the population size of each area in the total population size ($N_i/N$). This formula changes to

$$\sum_{i \in k} \xi_{ki} \hat{I}_{i}^{\text{bench}} = \tau_k,$$

if there are $k = 1, ..., K$ higher level domains for benchmarking to. Therefore, $K$ values for benchmarking ($\tau_k$) are needed and $\xi_{ki}$ stands for the share of the population size of each area in $k$ ($N_i/N_k$). If population weights are provided as an argument in the \textbf{ebp()} function, the previously described formula will be modified. In this case, the value of $\xi_{ki}$ is calculated as the ratio of the summed population weights in $i$ regarding $k$. Therefore, the resulting formula is $\xi_{ki} = \sum_{h \in i} pw_h / \sum_{h \in k} pw_h$, where $pw_h$ is the population weight of unit $h$.

To calculated the benchmark values, three different methods (\textbf{raking}, \textbf{ratio}, and \textbf{ratio_complement}) are available within the \textbf{ebp()} function. For \textbf{raking} and the \textbf{ratio} methods, the estimates are adjusted according to

$$\hat{I}_{i}^{\text{bench}} = \hat{I}_i + \left( \sum_{i \in k} \xi_{ki} \phi_{ki} \right)^{-1} \left( \tau_k - \sum_{i \in k} \xi_{ki} \hat{I}_i \right) \phi_{ki}.$$

For \textbf{raking}, all small area estimates ($\hat{I}_i$) are adjusted by the same value. Therefore, $\phi_{ki}$ equals $\xi_{ki}$. If a ratio adjustment (\textbf{ratio}) is selected, $\phi_{ki} = \xi_{ki}/\hat{I}_i$. Hence, large estimates are corrected more than smaller ones. In the ratio case, the equation above is equivalent to the simpler formula:

$$\hat{I}_{i}^{\text{bench}} = \hat{I}_i + \frac{\tau_k}{\sum_{i \in k} \xi_{ki}}.$$

When estimating headcount poverty rates, $\hat{I}_{i}^{\text{bench}}$ is guaranteed to be positive. However, $\hat{I}_{i}^{\text{bench}}$ can exceed 1 in cases where $\tau_k > \sum_{i \in k} \xi_{ki} \hat{I}_i$. We therefore offer the \textbf{ratio_complement} method, which adjusts the estimated share of the population that is not poor by a constant fraction to ensure consistency with the survey.

$$\hat{I}_{i}^{\text{bench}} = 1 - \left( 1 - \hat{I}_i \right) \frac{(1 - \tau_k)}{\sum_{i \in k} \xi_{ki}(1 - \hat{I}_i)}.$$

This method also ensures that the estimated poverty rates are consistent with the survey estimates at the higher level. While the resulting estimates can be negative, they cannot exceed one. They therefore provide an alternative when headcount estimates using ratio benchmarking exceed one.

Because the literature only discusses benchmarking for linear indicators, we only offer benchmarking for the two indicators \textit{Mean} and \textit{Head_Count}. The adjustment shown here is also applied within the MSE bootstrap procedure, so that an MSE can also be obtained for these adjusted estimators.

For internal benchmarking, the survey data is used to automatically calculate direct estimates (Horvitz and Thompson 1952) for benchmarking. Therefore, survey weights must be available and specified in the argument \textbf{weights}. When the benchmark weights option is not specified, the specified survey weights are assumed to be the benchmark weights.
3.2. Functionality

The following three arguments have been added to the \texttt{ebp()} function so that benchmarking can be performed with different options.

<table>
<thead>
<tr>
<th>Arguments</th>
<th>Short description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>benchmark</td>
<td>For external benchmarking: benchmark value(s) (a named numeric vector, or a data.frame)</td>
<td>NULL</td>
</tr>
<tr>
<td></td>
<td>For internal benchmarking: a vector containing the name of the indicators to be benchmarked (i.e. c(&quot;Mean&quot;,&quot;Head_Count&quot;))</td>
<td></td>
</tr>
<tr>
<td>benchmark_type</td>
<td>Type of benchmarking</td>
<td>ratio</td>
</tr>
<tr>
<td>benchmark_level</td>
<td>The name of the domain variable for the benchmark level, if benchmarking to multiple levels instead of globally</td>
<td>NULL</td>
</tr>
<tr>
<td>benchmark_weights</td>
<td>The name of variable containing benchmark weights. This is only possible for internal benchmarking and enable users to benchmark with weights that differ from the survey weights (Default for internal benchmarking).</td>
<td>NULL</td>
</tr>
</tbody>
</table>

Model estimation

To demonstrate the functionalities of the package, we show examples of both external and internal benchmarking

External benchmark An external benchmark value comes from another data source and is considered reliable such as a value published by a statistical office. More than one value can be specified in the \texttt{ebp()} function. If there are several levels in the data, values can also be supplied for a higher level as the small area estimates level for benchmarking.

\begin{verbatim}
R> library("povmap")
R> # Load sample data set
R> data("eusilcA_smp")
R> data('eusilcA_pop')
\end{verbatim}

The following lines add a global benchmark value for the head count ratio to the \texttt{ebp()} function otherwise this call almost equals the shown example Kreutzmann et al. (2019):

\begin{verbatim}
R> benchmark <- mean(eusilcA_smp$eqIncome)
R> names(benchmark) <- c("Mean")

R> ebp_bench_external <- ebp(
+     fixed = eqIncome ~ gender + eqsize + cash + self_empl + unempl_ben +
+     age_ben + surv_ben + sick_ben + dis_ben + rent + fam_allow +
+     house_allow + cap_inv + tax_adj,
+     pop_data = eusilcA_pop, pop_domains = "district",
+     smp_data = eusilcA_smp, smp_domains = "district",
+     na.rm = TRUE, benchmark = benchmark, benchmark_type = "ratio")
\end{verbatim}
The method used here for the inclusion of benchmarking is "ratio".
To add external benchmark values a `data.frame` must be supplied via the `benchmark` argument which, in addition to the benchmark values, also contains the names of the benchmark domains. Therefore, the additional argument `benchmark_level` is needed to specify the variable name of the benchmark level within the sample and population data.

```
R> median_state <- tapply(eusilcA_smp$eqIncome, eusilcA_smp$state, median)
R> benchmark_table <- data.frame(state = names(median_state), Mean = median_state)
R> ebp_bench_external_state <- ebp(
+   fixed = eqIncome ~ gender + eqsize + cash + self_empl + unempl_ben +
+   age_ben + surv_ben + sick_ben + dis_ben + rent + fam_allow +
+   house_allow + cap_inv + tax_adj,
+   pop_data = eusilcA_pop, pop_domains = "district",
+   smp_data = eusilcA_smp, smp_domains = "district",
+   na.rm = TRUE, benchmark = benchmark_table, benchmark_type = "ratio",
+   benchmark_level = "state")
```

**Internal benchmark**  For internal benchmarking no benchmark value has to be supplied. The sample data itself is used to benchmark the small area estimates (i) to a global value or (ii) to a higher geographic level than the small area level. Within the argument `benchmark` the user must specify for which indicator ("Mean", "Head_Count" or both) benchmarking should be carried out. Please note, the argument `weights` is needed to do internal benchmarking, because the results are benchmarked to weighted sample means. To do benchmarking on higher domain level the argument `benchmark_level` is used. The option `benchmark_weights` allows the user to specify a set of weights used for benchmarking that differs from the use of sample weights. This can be useful if, for example, the sample weights are normalized to give each target area equal weight, in which case `benchmark_weights` can specify the non-normalized original survey weights.

```
R> ebp_bench_internal_state <- ebp(
+   fixed = eqIncome ~ gender + eqsize + cash + self_empl + unempl_ben +
+   age_ben + surv_ben + sick_ben + dis_ben + rent + fam_allow +
+   house_allow + cap_inv + tax_adj,
+   pop_data = eusilcA_pop, pop_domains = "district",
+   smp_data = eusilcA_smp, smp_domains = "district",
+   weights = "weight", weights_type = "nlme",
+   na.rm = TRUE, benchmark = c("Mean"), benchmark_type = "ratio",
+   benchmark_level = "state", MSE = TRUE)
```

**Estimation results**

**External benchmark**  For the global external benchmark, the following results are obtained and it can be easily checked that the benchmarking leads to the correct value.
R> head(estimators(ebp_bench_external, indicator = "Mean_bench"))

<table>
<thead>
<tr>
<th>Domain</th>
<th>Mean_bench</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eisenstadt-Umgebung</td>
<td>27804.23</td>
</tr>
<tr>
<td>Eisenstadt (Stadt)</td>
<td>54230.22</td>
</tr>
<tr>
<td>Güssing</td>
<td>17373.81</td>
</tr>
<tr>
<td>Jennersdorf</td>
<td>13546.48</td>
</tr>
<tr>
<td>Mattersburg</td>
<td>21488.25</td>
</tr>
<tr>
<td>Neusiedl am See</td>
<td>19208.26</td>
</tr>
</tbody>
</table>

R> sum(ebp_bench_external$ind$Mean_bench *
       + table(ebp_bench_external$framework$pop_domains_vec)/
       + ebp_bench_external$framework$N_pop)

[1] 20140.09

R> mean(eusilcA_smp$eqIncome)

[1] 20140.09

The example that benchmarks the results to a higher domain level above the small area estimates leads to the following results.

R> head(estimators(ebp_bench_external_state, indicator = "Mean_bench"))

<table>
<thead>
<tr>
<th>Domain</th>
<th>Mean_bench</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eisenstadt-Umgebung</td>
<td>21692.41</td>
</tr>
<tr>
<td>Eisenstadt (Stadt)</td>
<td>42309.54</td>
</tr>
<tr>
<td>Güssing</td>
<td>13554.76</td>
</tr>
<tr>
<td>Jennersdorf</td>
<td>10568.75</td>
</tr>
<tr>
<td>Mattersburg</td>
<td>16764.78</td>
</tr>
<tr>
<td>Neusiedl am See</td>
<td>14985.97</td>
</tr>
</tbody>
</table>

**Internal benchmark**  For the internal benchmarking at the state level the following results are obtained.

R> head(estimators(ebp_bench_internal_state, indicator = c("Mean", "Mean_bench")))

<table>
<thead>
<tr>
<th>Domain</th>
<th>Mean</th>
<th>Mean_bench</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eisenstadt-Umgebung</td>
<td>28253.88</td>
<td>22881.16</td>
</tr>
<tr>
<td>Eisenstadt (Stadt)</td>
<td>55696.54</td>
<td>45105.37</td>
</tr>
<tr>
<td>Güssing</td>
<td>17192.22</td>
<td>13922.97</td>
</tr>
<tr>
<td>Jennersdorf</td>
<td>13106.49</td>
<td>10614.18</td>
</tr>
<tr>
<td>Mattersburg</td>
<td>21593.67</td>
<td>17487.45</td>
</tr>
<tr>
<td>Neusiedl am See</td>
<td>19083.43</td>
<td>15454.56</td>
</tr>
</tbody>
</table>
If the goal is to compare the benchmarked value to the direct estimator, 'Mean_bench' or 'Head_Count_bench' must be added manually to the direct estimator. This value corresponds to the 'Mean' or the 'Head_Count'.

```r
R> emdi_direct <- direct(
+   y = "eqIncome", smp_data = eusilcA_smp, smp_domains = "district",
+   weights = "weight", var = TRUE, boot_type = "naive", B = 50, na.rm = TRUE)
```

```r
R> emdi_direct$ind$Mean_bench <- emdi_direct$ind$Mean
R> emdi_direct$MSE$Mean_bench <- emdi_direct$MSE$Mean
R> compare_plot(ebp_bench_internal_state, direct = emdi_direct,
+   CV = TRUE, indicator = "Mean_bench")
```

Not all domains contained in the model estimation have been found in the direct estimation. Following plots will only contain results for estimates available in both objects.

![Figure 1: Output compare_plot()](image)
4. Further transformations

The linear mixed models used by povmap assume that the area effect and residual are distributed normally. However, if the dependent variable in the model is skewed, this assumption will likely not hold in practice. Therefore, transformations are required to make the distributions of the error term closer to normal. Two further transformations (the rank-order transformation and the arcsine transformation) are incorporated to the ebp() function of the povmap package.

The rank-order transformation is particularly useful when dealing with non-normally distributed data or outliers. By converting the original values into their corresponding ranks, the transformed data can exhibit a more symmetrical distribution. Masaki et al. (2020) uses the rank-order transformation to make the distributions of the error term closer to normal and reduce discrepancies between official national poverty rates and the small area estimates. We use the procedure included in the bestNormalize package (Peterson and Cavanaugh 2019) to back-transform the rank-order transformation, using linear interpolation within the range of the data and a shifted approximation to extrapolate outside the range of the data. More research is needed to verify that this approach works well when estimating means.

Similarly, the arcsine transformation serves as a valuable tool when analysing proportions or percentages. As proportions are bounded by 0 and 1, their distribution can deviate from normality. The arcsine transformation, which applies the inverse sine function to the square root of the proportion \( y_{ij}^* = \sin^{-1} (\sqrt{(y_{ij}})) \), can stabilize the variance and improve the distributional properties of the data. The ebp() function transform the data, calculate the linear mixed model on the transformed data, and back-transform the data to the original scale to estimate the poverty indicators.

4.1. Functionality

In the following, we will show how the additional transformations can be used in the ebp() function of package povmap. The argument transformation is determining the chosen transformation. In the povmap package following options are available:

- no, log, box.cox, dual, log.shift as in the emdi package
- ordnorm: rank-order transformation using the bestNormalize package (Peterson and Cavanaugh 2019)
- arcsin: arcsine transformation for proportions

rank-order transformation

The ordnorm transformation can be directly applied in estimating poverty indicators from equivalent income using the ebp() function. The distribution of equivalent income exhibits clear outliers. The ordnorm transformation helps to better meet the normality assumptions of the errors in the estimation process. In the following, the ebp() function is performed without transformation and with ordnorm transformation to enable a comparison.

\begin{verbatim}
R> ebp_no <- ebp(
+   fixed = eqIncome ~ gender + eqsize + cash + self_empl +
\end{verbatim}
povmap: Extension to the emdi package for small area estimation

> ebp_ordernorm <- ebp(
+     fixed = eqIncome ~ gender + eqsize + cash + self_empl +
+     + unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent +
+     + fam_allow + house_allow + cap_inv + tax_adj,
+     + pop_data = eusilcA_pop, pop_domains = "district",
+     + smp_data = eusilcA_smp, smp_domains = "district",
+     + na.rm = TRUE, transformation = "ordernorm"
+ )

The bestNormalize package (Peterson and Cavanaugh 2019) provides also the back-transformation inv_ordernorm, which is needed to make the results interpretable at the target level. During the execution of the transformation, a warning message is generated if any values fall outside the original range (of the initial data) during the inverse transformation (Peterson and Cavanaugh 2019). The warning message will indicate the number of values that exceed the original value range.

Overall, the rank-order transformation helps to preserve the normality assumptions for the error terms. By using functions like summary() or qqnorm(), information about the distributions of both errors can be obtained.

R> summary(ebp_no)$normality

        Skewness Kurtosis Shapiro_W Shapiro_p
    Error    2.40813  26.206861  0.8806197    2.841389e-36
Random_effect  1.18355  4.098958  0.8957952   2.655502e-05

R> summary(ebp_ordernorm)$normality

        Skewness Kurtosis Shapiro_W Shapiro_p
    Error      -0.30938  4.324697   0.9851796  2.599253e-13
Random_effect -0.00487  2.262509   0.9859713   6.252016e-01

Comparing the two outputs, it is immediately apparent that the normal distribution assumptions are better fulfilled by using of the ordernorm transformation. In this case the normally assumption on the random effects is not rejected and for the error term the skewness and kurtosis is reduced.

The QQ-plots show the same findings.

R> qqnorm(ebp_no)
R> qqnorm(ebp_ordernorm)
To demonstrate the arcsine transformation, an example percentage variable is created by calculating the household income share relative to the maximum income.

\begin{verbatim}
R> eusilcA_smp$eqIncome_prop <- eusilcA_smp$eqIncome / max(eusilcA_smp$eqIncome)
\end{verbatim}

Subsequently, the `ebp()` function is applied without and with the arcsine transformation.

\begin{verbatim}
R> ebp_no <- ebp(
R> ebp_arcsine <- ebp(
\end{verbatim}
+ fixed = eqIncome_prop ~ gender + eqsize + cash + selfempl +
+ unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent +
+ fam_allow + house_allow + cap_inv + tax_adj,
+ pop_data = eusilcA_pop, pop_domains = "district",
+ smp_data = eusilcA_smp, smp_domains = "district",
+ na.rm = TRUE, transformation = "no"
+
R> ebp_arcsin <- ebp(
+ fixed = eqIncome_prop ~ gender + eqsize + cash + selfempl +
+ unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent +
+ fam_allow + house_allow + cap_inv + tax_adj,
+ pop_data = eusilcA_pop, pop_domains = "district",
+ smp_data = eusilcA_smp, smp_domains = "district",
+ transformation = "arcsin", na.rm = TRUE
+
)

Overall, the arcsine transformation helps to make the distribution of the error term more normal. By using functions like `summary()` or `qqnorm()`, information about the distributions of both errors can be obtained.

R> summary(ebp_no)$normality

<table>
<thead>
<tr>
<th></th>
<th>Skewness</th>
<th>Kurtosis</th>
<th>Shapiro_W</th>
<th>Shapiro_p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>2.40813</td>
<td>26.206861</td>
<td>0.8806197</td>
<td>2.841389e-36</td>
</tr>
<tr>
<td>Random_effect</td>
<td>1.18355</td>
<td>4.098958</td>
<td>0.8957952</td>
<td>2.655502e-05</td>
</tr>
</tbody>
</table>

R> summary(ebp_arcsin)$normality

<table>
<thead>
<tr>
<th></th>
<th>Skewness</th>
<th>Kurtosis</th>
<th>Shapiro_W</th>
<th>Shapiro_p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>1.5013512</td>
<td>19.258656</td>
<td>0.9224222</td>
<td>1.544219e-30</td>
</tr>
<tr>
<td>Random_effect</td>
<td>0.5365206</td>
<td>3.105318</td>
<td>0.9751956</td>
<td>1.787370e-01</td>
</tr>
</tbody>
</table>

All normal assumptions on the errors are rejected in all settings. However, the skewness and kurtosis improves by applying the arcsin transformation. The QQ-plots show this graphically.

R> qqnorm(ebp_no)
R> qqnorm(ebp_arcsin)
Figure 3: Output `qqnorm()` for using no transformation (a) and the arcsin transformation (b)

5. Further arguments for the ebp function

5.1. nlme control options

The `ebp()` function utilizes the `nlme` package for estimating linear mixed models. Specifically, the `ebp()` function relies on the `lme` function within `nlme` to perform its computations. The `lme` function allows you to set optimization and convergence values using the `nlmeControl` parameter. By modifying the control values manually, you can prevent issues such as non-convergence within the `lme` function, which may occur when the maximum number of itera-
tions (maxiter) is reached. This can occur when estimating Mean Squared Error using the
parametric bootstrap, even in cases where the model can be estimated using the sample data.

Three main arguments of the nlmeControl function are directly offered to users as additional
arguments within the ebp() function. The following table provides an overview of these three
new arguments.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlme_maxiter</td>
<td>Specifies the maximum number of iterations allowed for convergence. If the maximum number of iterations is reached without convergence, the algorithm stops. By adjusting this argument, you can control the maximum iterations for the lme() function within ebp().</td>
<td>1000</td>
</tr>
<tr>
<td>nlme_tolerance</td>
<td>Sets the tolerance level for convergence. Convergence is considered achieved when the change in the estimated parameters falls below this tolerance value. Modifying this argument allows you to influence the convergence criteria for the lme() function within ebp().</td>
<td>1e-6</td>
</tr>
<tr>
<td>nlme_opt</td>
<td>Specifies the optimizer to be used, either 'nlminb' or 'optim'. Using 'optim' can occasionally avoid false convergence errors in the parametric bootstrap procedure.</td>
<td>&quot;nlminb&quot;</td>
</tr>
</tbody>
</table>

These additional arguments provide flexibility and control over the convergence behavior of the ebp() function, ensuring that you can tailor the estimation process to your specific needs.

5.2. Rescaling of weights

The argument rescale_weights (default FALSE) gives the user the option to decide if the weights should be scaled to a mean weight of 1 within each target domain. If rescale_weights is TRUE, the weights for each target area sum to the sample size, corresponding to "Method 2" in Pfeffermann et al. (1998). The decision to rescale or not to rescale the weights directly influences the results of the ebp() function. You will find an overview of weighting for linear mixed models in Pfeffermann et al. (1998); Rabe-Hesketh and Skrondal (2006).

5.3. Ydump

The argument Ydump (default NULL) gives the user the option to output the simulated values for the dependent variable and its components to a .csv file. The user should specify the name of the csv file, including the .csv extension and optionally the path, in the Ydump argument. The output file contains N*L rows, where N is the number of population units and L is the number of replications, and six columns:

1. L, ranging from 1 to L, indicating the replication number
2. Domain, indicating the target domain
3. Simulated_Y, indicating the simulated value of the indicator after back-transformation to the original scale
4. XBetahat, indicating the fixed effect portion of the prediction in the transformed scale.
5. eta, the randomly drawn area effect in the transformed scale

6. epsilon, the randomly drawn idiosyncratic error term in the transformed scale

Currently this option is only available for the L replications used to generate the point estimates. Even if MSE is set to TRUE, values drawn for the parametric bootstrap replications will not be included.

6. New functionalities for a user-friendly reporting of results

To enhance user-friendliness, the povmap package offers new functionalities, particularly focusing on the head count ratio. Furthermore, reports can be generated providing information on the estimators and their corresponding coefficient of variation (CV), as well as the underlying sample and population data. Additionally, details about the linear mixed model used (coefficients, model fit) in the ebp() object are directly outputted. A table comparing different CVs can be created directly. This table includes the CV for the head count ratio estimated using the ebp() function, as well as three different types of CVs for the corresponding direct estimator, which arise from different approaches to MSE estimation. In addition, a new argument has been added to the direct() function, allowing for the determination of Horvitz-Thompson variance approximation. For a more detailed description of this method for variance estimation, see Marhuenda et al. (2013).
Function | Description
---|---
**ebp_reportdescriptives** | Report descriptive statistics
  - CV (for head count ratio)
  - basic informations on survey and population data
  - national poverty rates and lines (survey vs. population)

**ebp_test_means** | Weighted means for the input variables (typically auxiliary variables of the ebp) for the survey and population data → Comparison using test of difference

**ebp_reportcoef_table** | User-friendly output of the coefficients of the linear mixed regression model with standard errors

**ebp_report_byrank** | Produce report tables that rank head count estimates either by population of poor or the head count rates themselves in descending order

**ebp_compute_cv** | Estimates three different types of CVs for the head count ratio for direct estimates:
  - CV using calibrated/naive bootstrapping of the MSE
  - CV using Horowitz Thompson variance estimation technique to compute MSE
  - CV using design effect adjusted naive calibrated MSE
These functions also return the direct and model-based headcount rates, as well as the CV for the model-based headcount estimates.

**ebp_normalityfit** | Table showing marginal R-square, conditional R-squared as well as the skewness and kurtosis of the random and idiosyncratic error terms

### 7. Stata integration of povmap

For users that are more comfortable working in Stata than R, the povmap package includes Rpovmap.ado and Rpovmap.hlp files, which run povmap from within Stata. Rpovmap.ado creates and executes an R script called povmap.R from within Stata. This script loads previously saved population and sample .dta files into R and calls the ebp function in povmap with specified options. The results are saved in an Excel spreadsheet and optionally an R object, which can be loaded in R for further analysis as desired.

Rpovmap.ado requires the following:

1. R to be installed.
2. The haven and povmap packages to be installed into R. These can be installed using the install.packages("haven") and install.packages("povmap") commands in R. If the devtools package is installed and loaded into memory, povmap can also be installed directly from Github using install_github("SSA-Statistical-Team-Projects/povmap").
3. The Rscript package to be installed in Stata, by typing ssc install rscript in Stata.
4. The `R povmap.ado` and `R povmap.hlp` to be present in either Stata’s current directory, or in a Stata-recognized ado directory (typing `cd` in Stata will show the current directory, and typing `adopath` will show the location of the Stata ado directories).

Sample data `eusilcA_smp.dta` and `eusilcA.dta` are included in the package. These were created using the `write_dta` function in the `haven` package, using the following R code:

```r
R> library(haven)
R> data("eusilcA_pop")
R> data("eusilcA_smp")
R> write_dta(data=eusilcA_pop,path="eusilcA_pop")
R> write_dta(data=eusilcA_smp,path="eusilcA_smp")
```

These can be used to replicate the analysis in section 2.2 of this vignette, using the following command within Stata.

```
R povmap eqIncome gender eqsize cash self_emp unempl_ben age_ben surv_ben sick_ben dis_ben rent fam_allow house_allow cap_inv tax_adj,
pop_data(eusilcA_pop.dta) smp_data(eusilcA_smp.dta)
smp_domains(district) pop_domains(district) weights(weight)
weights_type(Guadarrama) transformation(log) na_rm(TRUE)
saveobject(emdi_model_Guadarrama) savexls(emdi_model_Guadarrama.xlsx)
```

This command produces two output files in the current folder: `emdi_model_Guadarrama.xlsx` and the saved R object `emdi_model_Guadarrama`, which contains the `emdi` object `ebp_results`. The files can also be saved in a directory specified by the user as part of the string in the `savexls` and `saveobject` options.

The `saveobject` option is recommended to allow for further analysis from within R. For example, after using the `setwd()` function to set the current directory in R to the folder in which `emdi_model_Guadarrama` was saved, executing:

```r
R> load("emdi_model_Guadarrama")
R> ebp_reportcoef_table(ebp_results)
```

displays model coefficients in R.

To analyze the results in Stata, use the `import excel` command to load the saved estimates.

```
.import excel using "emdi_model_Guadarrama", sheet("Point Estimators") firstrow clear
.list Domain Mean in 1/5, clean noobs
```

<table>
<thead>
<tr>
<th>Domain</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eisenstadt-Umgebung</td>
<td>30926.729</td>
</tr>
<tr>
<td>Eisenstadt (Stadt)</td>
<td>94261.315</td>
</tr>
<tr>
<td>Güssing</td>
<td>17008.432</td>
</tr>
<tr>
<td>Jennersdorf</td>
<td>13281.905</td>
</tr>
<tr>
<td>Mattersburg</td>
<td>21830.831</td>
</tr>
</tbody>
</table>
Rpovmap treats all labeled variables in the sample and population data as factor variables, using the as_factor function in the haven package. In this example, the gender variable takes on values of 1 or 2 in the data, but is appropriately treated as a factor variable instead of a continuous variable in model estimation.

Typing help Rpovmap from within Stata will load the help file listing the full set of options, which mirror those in the R povmap package.

8. Conclusion

This vignette has presented the new functionalities of the povmap package compared to the emdi package. These functionalities include options for incorporating sample weights, benchmarking, additional transformations, additional arguments for the ebp() function, user-friendly output options, and a convenient wrapper for Stata users.

Acknowledgments

The team gratefully acknowledges funding from the Knowledge for Change Program’s phase-IV programmatic research project “Understanding Trends in Sub-National Differences in Economic Well-Being in Low and Middle- Income Countries”. We thank Jed Friedman, Haishan Fu, Keith Garrett, Talip Kilic, Pierella Paci, and Utz Pape for support.
References


Affiliation:
Ifeanyi Edochie, David Newhouse
World Bank Group
1818 H Street NW, Washington DC, 20433
E-mail: iedochie@worldbank.org, dnewhouse@worldbank.org

Timo Schmid, Nora Würz
Chair of Statistics and Econometrics
Faculty of Social and Economic Sciences
Otto Friedrich University Bamberg
Feldkirchenstraße 21, 96045 Bamberg, Germany
E-mail: timo.schmid@uni-bamberg.de, nora.wuerz@uni-bamberg.de