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. This vignette provides an overview of the additional features provided by **povmap**. The package adds two notable new features. 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 EBP. The second main new feature is the incorporation of benchmarking. If 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. Two additional options are provided to transform data and 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, 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. Informative sampling with the povmap package

The **povmap** package offers, in addition to the methodology of Guadarrama *et al.* (2018), the possibility to consider the 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 Empirical Best Predictor (EBP) models, which are special cases of a linear mixed model. The `weights` argument in the `lme` command provides an alternative method 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 the informative sampling for the estimation of the model 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 informative sampling 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, informative sampling is also included within the estimation of the data-driven parameter for the transformation. The 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.

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", 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
+ )
```

Estimation results

The results can be called with the estimator function as in **emdi**.

```
R> head(estimators(emdi_model_Guadarrama, indicator = "Mean"))

               Domain     Mean
1 Eisenstadt-Umgebung 30926.73
2  Eisenstadt (Stadt) 94261.32
3             Güssing 17008.43
4         Jennersdorf 13281.90
5         Mattersburg 21830.83
6     Neusiedl am See 19492.90
```

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

                 Domain     Mean
1 Eisenstadt-Umgebung 31071.60
2  Eisenstadt (Stadt) 98082.34
3             Güssing 16990.86
4         Jennersdorf 13263.20
5         Mattersburg 21922.16
6     Neusiedl am See 19443.56
```

A comparison between the two different approaches to informative sampling 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")))

                 Domain     Mean
1 Eisenstadt-Umgebung 28253.88
2  Eisenstadt (Stadt) 55696.54
3             Güssing 17192.22
4         Jennersdorf 13106.49
5         Mattersburg 21593.67
6     Neusiedl am See 19083.43
```

```
R> head(estimators(emdi_model_nlme_lambda, indicator = "Mean"))

                 Domain     Mean
1 Eisenstadt-Umgebung 28252.67
2  Eisenstadt (Stadt) 55547.81
3             Güssing 17200.02
4         Jennersdorf 13098.56
5         Mattersburg 21603.73
6     Neusiedl am See 19082.95
```

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 (τ) it follows

$$\sum_{i=1}^{D} \xi_i \hat{I}_i^{bench} = \tau,$$

where ξ_i stands for the share of the population size of each area in the total population size (N_i/N). This formula changes to

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

if there are $k = 1, ..., K$ higher level domains for benchmarking to. Therefore, K values for benchmarking (τ_k) are needed and ξ_{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 `ebp()` function, the previously described formula will be modified. In this case, the value of ξ_{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 two different methods (`raking` and `ratio`) are available within the `ebp()` function. The estimates are adjusted according to

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

For `raking`, all small area estimates $(\hat{I}_i)$ are adjusted by the same value. Therefore, ϕ_{ki} equals ξ_{ki}. A ratio adjustment (`ratio`) is being conducted if $\phi_{ki} = \xi_{ki}/\hat{I}_i$. Hence, large estimates are corrected more than smaller ones. Because the literature only discusses benchmarking for linear indicators, we only offer benchmarking for the two indicators ’`Mean`’ and ’`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 `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 `ebp()` function so that benchmarking can be performed with different options.

Arguments	Short description	Default
`benchmark`	For external benchmarking: benchmark value(s) (a named numeric vector, or a data.frame) For internal benchmarking: a vector containing the name of the indicators to be benchmarked	`NULL`
`benchmark_type`	Type of benchmarking	ratio
`benchmark_level`	The name of the domain variable for the benchmark level, if benchmarking to multiple levels instead of globally	`NULL`
`benchmark_weights`	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).	`NULL`

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 `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.

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

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

```
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")
```

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"))

              Domain Mean_bench
1 Eisenstadt-Umgebung   27804.23
2  Eisenstadt (Stadt)   54230.22
3             Güssing   17373.81
4         Jennersdorf   13546.48
5         Mattersburg   21488.25
6     Neusiedl am See   19208.26

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, indicator = "Mean_bench"))

              Domain Mean_bench
1 Eisenstadt-Umgebung   21692.41
2  Eisenstadt (Stadt)   42309.54
3             Güssing   13554.76
4         Jennersdorf   10568.75
5         Mattersburg   16764.78
6     Neusiedl am See   14985.97
```

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

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

              Domain     Mean Mean_bench
1 Eisenstadt-Umgebung 28253.88   22881.16
2  Eisenstadt (Stadt) 55696.54   45105.37
3             Güssing 17192.22   13922.97
4         Jennersdorf 13106.49   10614.18
5         Mattersburg 21593.67   17487.45
6     Neusiedl am See 19083.43   15454.56
```

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> 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> 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, 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.
```

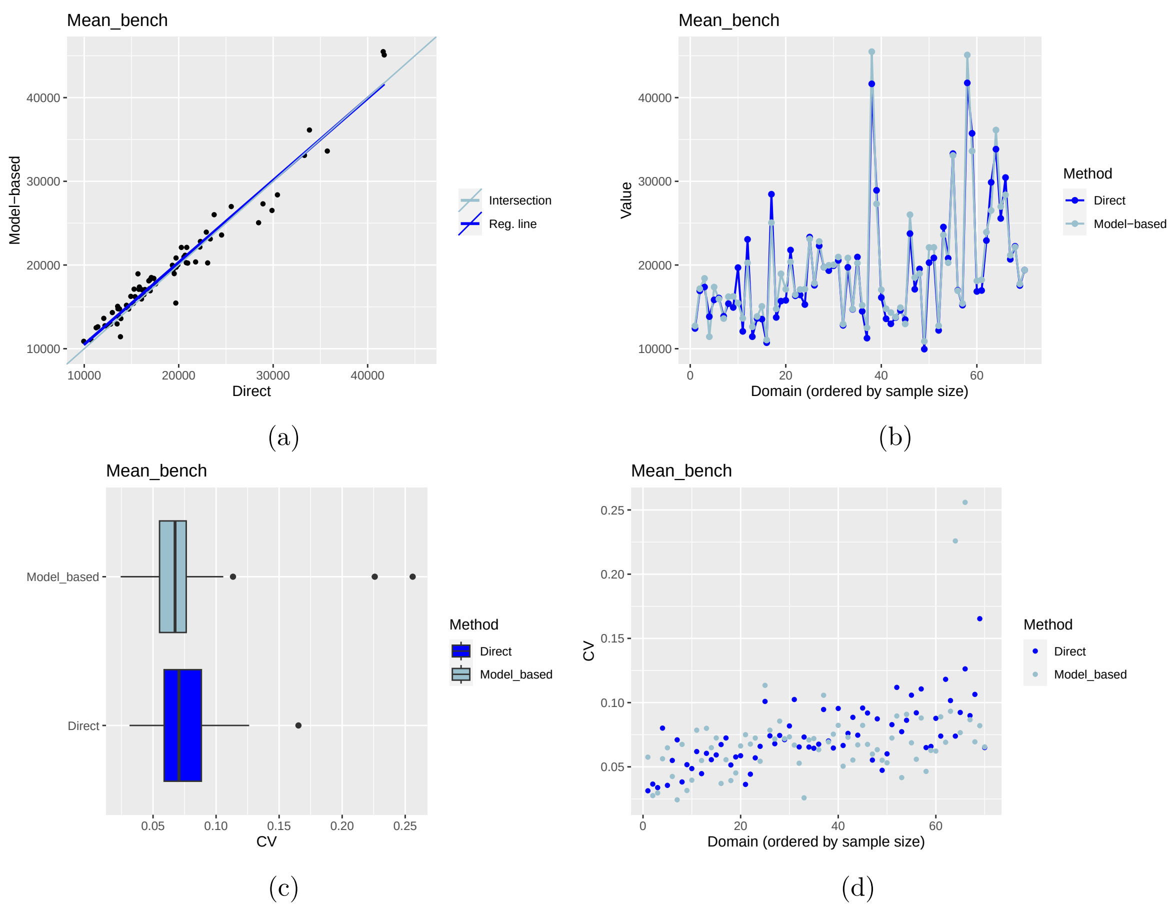


Figure 1: Output `compare_plot()`

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 transformation (the rank-order transformation and the arcsine transformation) are incorparted 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 $\left(y^*_{ij} = sin^{-1}\left(\sqrt(y_{ij})\right)\right)$, 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
- `ordernorm`: rank-order transformation using the **bestNormalize** package (Peterson and Cavanaugh 2019)
- `arcsin`: arcsine transformation for proportions

rank-order transformation

The `ordernorm` 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 `ordernorm` 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 `ordernorm` transformation to enable a comparison.

```
R> ebp_no <- 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 = "no"
+ )

> 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.

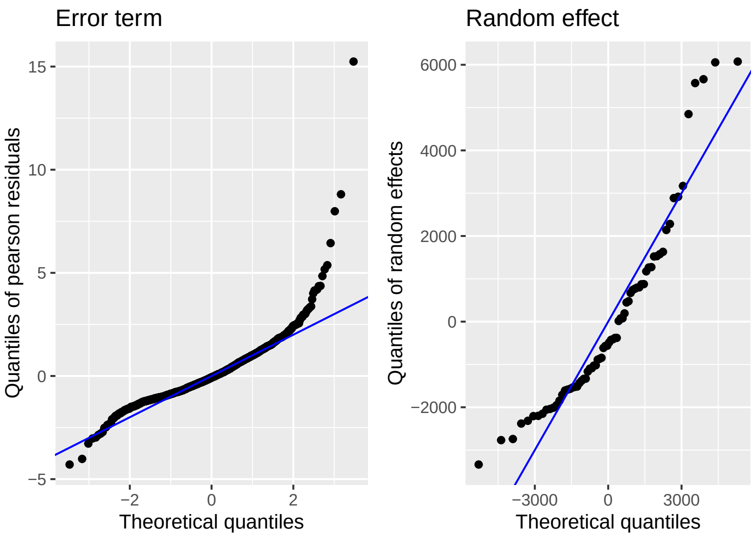
Overall, the rank-order transformation helps to preserve the normality assumptions for the error terms. By using functions like `summary()` or `qqnorm()`, information about the distributions of both errors can be obtained.

```
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         -0.309379683 4.324697 0.9851796 2.599253e-13
Random_effect -0.004873077 2.262509 0.9859713 6.252016e-01
```

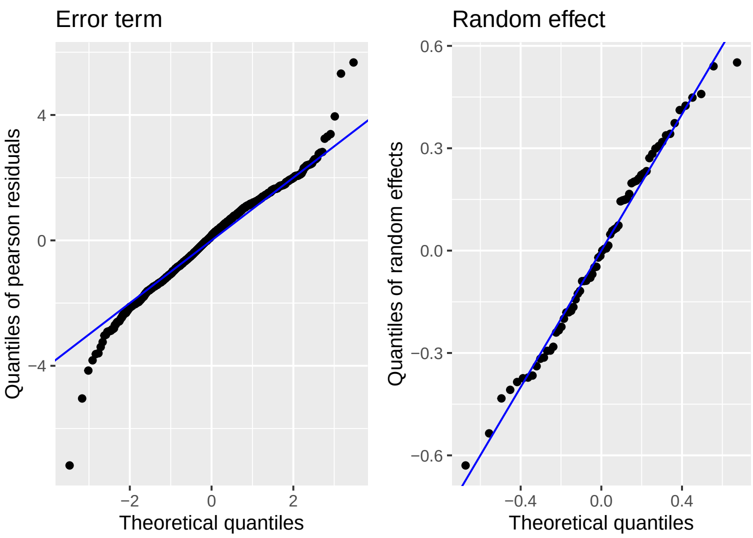
Comparing the two outputs, it is immediately apparent that the normal distribution assumptions are better fulfilled by using of the `ordernorm` transformation. In this case the normally assumption on the random effects is not rejected and for the error term the skewness and kurtosis is reduced.

The QQ-plots show the same findings.

```
R> qqplot(ebp_no)
R> qqplot(ebp_ordernorm)
```



(a)



(b)

Figure 2: Output `qqnorm()` for using no transformation (a) and the ordernorm transformation (b)

arcsine transformation

To demonstrate the arcsine transformation, an example percentage variable is created by calculating the household income share relative to the maximum income.

```
R> eusilcA_smp$eqIncome_prop <- eusilcA_smp$eqIncome / max(eusilcA_smp$eqIncome)
```

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

```
R> ebp_no <- ebp(
```

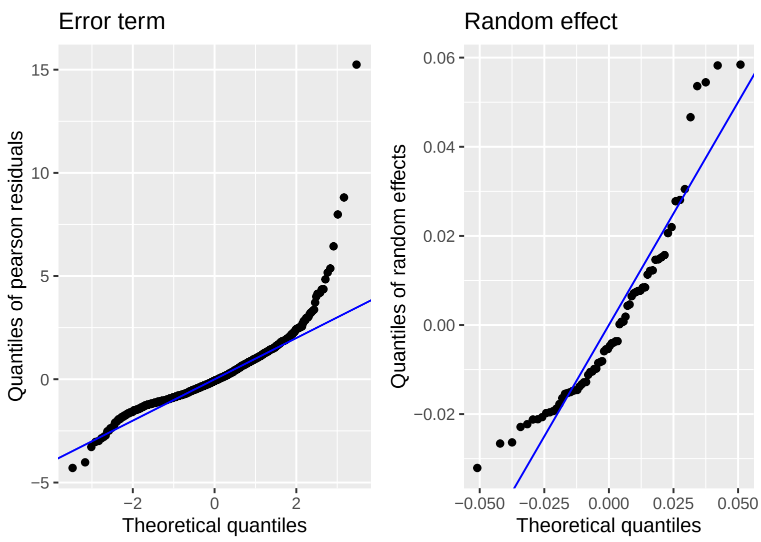
```
+   fixed = eqIncome_prop ~ gender + eqsize + cash + self_empl +
+     unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent +
+     fam_allow + house_allow + cap_inv + tax_adj,
+   pop_data = eusilcA_pop, pop_domains = "district",
+   smp_data = eusilcA_smp, smp_domains = "district",
+   na.rm = TRUE, transformation = "no"
+ )

R> ebp_arcsin <- ebp(
+   fixed = eqIncome_prop ~ 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",
+   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.

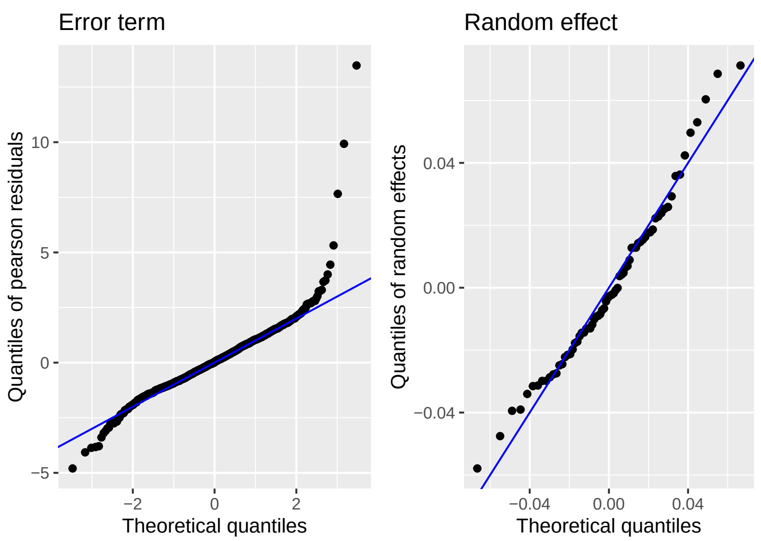
```
R> summary(ebp_no)$normality

              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_arcsin)$normality

               Skewness  Kurtosis Shapiro_W    Shapiro_p
Error         1.5013512 19.258656 0.9224222 1.544219e-30
Random_effect 0.5365206  3.105318 0.9751956 1.787370e-01
```

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> qqplot(ebp_no)
R> qqplot(ebp_arcsin)
```



(a)



(b)

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 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 iterations (`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.

The two 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 two new arguments.

Argument	Description	Default
`nlme_maxiter`	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()`.	1000
`nlme_tolerance`	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()`.	1e-6

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).

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. It is worth noting that 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 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` writes and executes an R script called `povmap.R` from within Stata. This script loads previously saved population and sample data 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 on the local machine
2. The **haven** and **povmap** packages to be installed into R. These can be installed using `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("NoraWuerz/povmap")`.
3. The **Rscript** package to be installed in Stata, by typing `ssc install rscript` in Stata.
4. The `Rpovmap.ado` and `Rpovmap.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 ado directories)

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

```
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.

```
Rpovmap 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> 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
```

```
Domain              Mean
Eisenstadt-Umgebung 30926.729
Eisenstadt (Stadt)  94261.315
Güssing             17008.432
Jennersdorf         13281.905
Mattersburg         21830.831
```

`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, 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

Bell WR, Datta GS, Ghosh M (2013). "Benchmarking small area estimators." *Biometrika*, **100**(1), 189–202.

Datta G, Ghosh M, Steorts R, Maples J (2011). "Bayesian benchmarking with applications to small area estimation." *Test*, **20**, 574–588.

Guadarrama M, Molina I, Rao J (2018). "Small area estimation of general parameters under complex sampling designs." *Computational Statistics & Data Analysis*, **121**, 20–40.

Horvitz D, Thompson D (1952). "A Generalization of Sampling Without Replacement from a Finite Universe." *Journal of the American Statistical Association*, **47**(260), 663–685. doi:10.1080/01621459.1952.10483446.

Kreutzmann AK, Pannier S, Rojas-Perilla N, Schmid T, Templ M, Tzavidis N (2019). "The R Package **emdi** for Estimating and Mapping Regionally Disaggregated Indicators." *Journal of Statistical Software*, **91**(7), 1–33. doi:10.18637/jss.v091.i07.

Marhuenda Y, Molina I, Morales D (2013). "Small Area Estimation with Spatio-Temporal Fay-Herriot Models." *Computational Statistics and Data Analysis*, **58**, 308–325. doi:10.1016/j.csda.2012.09.002.

Masaki T, Newhouse D, Silwal AR, Bedada A, Engstrom R (2020). "Small area estimation of non-monetary poverty with geospatial data." *Statistical Journal of the IAOS*, **38**(3), 1035–1051.

Peterson RA, Cavanaugh JE (2019). "Ordered quantile normalization: a semiparametric transformation built for the cross-validation era." *Journal of applied statistics*.

Pfeffermann D, Sikov A, Tiller R (2014). "Single-and two-stage cross-sectional and time series benchmarking procedures for small area estimation." *Test*, **23**, 631–666.

Pfeffermann D, Skinner CJ, Holmes DJ, Goldstein H, Rasbash J (1998). "Weighting for unequal selection probabilities in multilevel models." *Journal of the Royal Statistical Society: series B (statistical methodology)*, **60**(1), 23–40.

Pinheiro J, Bates D, DebRoy S, Sarkar D, R Core Team (2015). "nlme: Linear and nonlinear mixed effects models." *R package version 3.1-122*. URL https://cran.r-project.org/web/packages/nlme/index.html.

Rabe-Hesketh S, Skrondal A (2006). "Multilevel modelling of complex survey data." *Journal of the Royal Statistical Society Series A: Statistics in Society*, **169**(4), 805–827.

Skarke F, Kreutzmann AK, Würz N (2023). *Extensions to the ebp function in the R package **emdi***. R package Vignette, URL https://CRAN.R-project.org/package=emdi.

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