Extensions to the ebp function in the R package emdi

Felix Skarke
Freie Universität Berlin

Ann-Kristin Kreutzmann
Freie Universität Berlin

Nora Würz
Otto Friedrich University Bamberg

Abstract

The empirical best prediction (EBP) approach proposed by Molina and Rao (2010), and generalized in Guadarrama *et al.* (2016) as the census EBP, is implemented in the `ebp` function of the R package **emdi**. A first version of the function allowed for the estimation of point and MSE estimates under non-informative sampling. The log and Box-Cox transformations were provided to make the distribution of the error terms closer to normal. For the latter, the transformation parameter can be estimated from the data as suggested in Rojas-Perilla *et al.* (2020). Their evaluation study further shows that the transformations log-shift and Dual transformation perform well. Furthermore, Guadarrama *et al.* (2018) reveal the benefits of considering the sampling design in the EBP under informative sampling. Therefore, the second version of the `ebp` function includes two new functionalities: (a) additional data-driven transformations for the EBP under non-informative sampling, the log-shift and Dual transformation, (b) the inclusion of sampling weights to consider informative sampling. The functionality of these extensions is demonstrated by examples based on synthetic data included in the package.

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

1. Introduction

The empirical best prediction (EBP) by Molina and Rao (2010) is one of the most popular unit-level models in the field of small area estimation (SAE) along with e.g., the World Bank method also known as ELL by Elbers *et al.* (2003). Therefore, it is one of the small area estimation methods implemented in the R package **emdi** (Kreutzmann *et al.* 2019). However, the approach provided by function `ebp` relies, among others, on the following assumptions: 1. normality of the error terms, and 2. a non-informative sampling design.
Several studies address the first aspect. While Diallo and Rao (2014) relax the normality assumption by allowing for skew-normal error terms, Graf *et al.* (2019) propose the usage of the more flexible distribution generalized beta of the second kind (GB2). Marino *et al.* (2019) investigate a semi-parametric empirical best predictor that estimates the distribution of the random effects from the data. Another, more straightforward approach is transforming the dependent variable. Extensive evaluation studies in Rojas-Perilla *et al.* (2020) find that data-driven transformations help to achieve at least symmetric distributions. The transformations compared are the Box-Cox (Box and Cox 1964), Dual (Yang 2006) and log-shift (Feng *et al.* 2016) transformation. Therefore, the Dual and log-shift transformations have been added

to the `ebp` function complementing the log and Box-Cox transformation that were already implemented in the first version of the package.
The second point affects the estimation when the sampling design of the survey used in the EBP is informative. Guadarrama *et al.* (2018) propose the incorporation of sampling weights by using the Pseudo empirical best linear unbiased predictor (PEBLUP) (You and Rao 2002) for the estimation of the model parameters. They show that the weighted EBP estimator has a lower bias than the unweighted EBP estimator under informative sampling and comparable results under non-informative sampling. The new version of the `ebp` function includes both options.
The first version of the `ebp` function is explained in detail in Kreutzmann *et al.* (2019). Therefore, this vignette will focus only on short introductions to the newly implemented features and will further show how to use the functionality. Throughout the vignette, it is assumed that a finite population of size N is partitioned into D domains of sizes $N_1, \ldots, N_D$. The index $i = 1, \ldots, D$ refers to an ith domain and $j = 1, \ldots, N_i$ to the jth household/individual. From the population, a random sample is drawn of size n with $n_1, \ldots, n_D$ observations belonging to that domain.

2. Additional data-driven transformations

The underlying model in the EBP is the nested error linear regression model that is a unit-level mixed model with a random intercept (Battese *et al.* 1988). One way to make the data better conform with the assumptions imposed by linear and linear mixed models is transforming the dependent variable. The transformation may help to achieve linearity, homoscedasticity and normality. The EBP crucially depends on the latter since the random domain-specific effect and the unit-level error term are drawn from a normal distribution in the Monte Carlo simulation. Therefore, we will focus on normality in this vignette even though the transformations may also improve the model in other aspects (Rojas-Perilla 2018, pp. 9-45).

2.1. Methodology

The most common transformation to achieve normality in the error terms is the log transformation, especially when the distribution of the dependent variable is right-skewed which is often observed for e.g., income. Since the logarithm cannot be applied for negative values, a deterministic shift can be added as follows:

$$y_{ij}^{*} = log(y_{ij} + s),$$

where y_{ij} is the variable of interest of domain i and unit j and s is a deterministic shift chosen such that $y_{ij} + s > 0$.
While the log transformation is especially beneficial for practitioners that are interested in the interpretation of parameters, it does not need to be the best option in a predictive model due to the missing ability to adapt to the data. In contrast, transformations with a transformation parameter λ can be fitted to the data. Rojas-Perilla *et al.* (2020) compare the log-shift (Feng *et al.* 2016), Box-Cox (Box and Cox 1964), and Dual transformation (Yang 2006) in the EBP context.
The log-shift transformation is the simplest way to make the log transformation more flexible and adpatable to the data. Instead of a deterministic shift, the data is shifted by an optimal

shift before the logarithm is applied:

$$y_{ij}^* = log(y_{ij} + \lambda),$$

where y_{ij} is the variable of interest of domain i and unit j and $\lambda >= s$ is an estimated shift. When $\lambda = s$, this transformation equals the implemented log transformation with deterministic shift.
The Box-Cox transformation is a famous transformation in the family of power transformations (Box and Cox 1964). Since the Box-Cox transformation is not suitable for negative values, a deterministic shift can be added as for the log transformation. Its shifted version is defined by:

$$y_{ij}^*(\lambda) = \begin{cases} \frac{(y_{ij}+s)^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0; \\ \log(y_{ij} + s) & \text{if } \lambda = 0, \end{cases}$$

where y_{ij} is the variable of interest of domain i and unit j and s is a deterministic shift chosen such that $y_{ij} + s > 0$. One characteristic of the Box-Cox transformation is that the cases of no transformation when $\lambda = 1$ (the data is only shifted) and applying the log transformation with a deterministic shift when $\lambda = 0$ are covered. A known drawback is the truncation of y_{ij}^*. The transformed variable y_{ij}^* is bounded from below by $\frac{1}{\lambda}$ when $\lambda > 0$, and bounded from above by $\frac{-1}{\lambda}$ when $\lambda < 0$.
The Dual transformation overcomes this issue. It is originally only defined for strictly positive values, but Rojas-Perilla *et al.* (2020) include a deterministic shift as follows:

$$y_{ij}^*(\lambda) = \begin{cases} ((y_{ij} + s)^\lambda - (y_{ij} + s)^{-\lambda})/2\lambda & \lambda > 0; \\ \log(y_{ij} + s) & \lambda = 0, \end{cases}$$

where y_{ij} is the variable of interest of domain i and unit j and s is a deterministic shift chosen such that $y_{ij} + s > 0$. The transformation parameter λ cannot be negative.
For the estimation of the transformation parameter in linear mixed regression models, Gurka *et al.* (2006) propose maximum likelihood and residual maximum likelihood (REML) methods. Rojas-Perilla *et al.* (2020) investigate the maximum likelihood based methods as well as alternative approaches. So far, the package **emdi** only provides the REML approach for the model fitting and the estimation of the transformation parameter. For the explanation of how the transformations are included in the EBP, we refer to Kreutzmann *et al.* (2019, Section 2.2) and Rojas-Perilla *et al.* (2020).

2.2. Functionality

In the following, we will show how the new transformations can be used in the `ebp` function of package **emdi**. The argument `transformation` is determining the chosen transformation. In the new version of function `ebp` following options will be available:

- `no`: No transformation
- `log`: Log transformation with a deterministic shift
- `box.cox`: Box-Cox transformation with a deterministic shift
- `dual`: Dual transformation with a deterministic shift

Transformation	Default interval
`box.cox`	c(-1, 2)
`dual`	c(0, 2)
`log.shift`	c(a, b) with $a = max(0, \lvert min(y)\rvert + 1)$, $b = \frac{(max(y)-min(y)}{2}$

Table 1: Default values for the estimation of the transformation parameter λ.

- `log.shift`: Log transformation with an optimized shift

The Box-Cox transformation is chosen to be the default transformation since it covers the options of no transformation and the logarithm. For the REML estimation of the transformation parameter λ, an interval needs to be specified. To simplify the usage, a default interval is defined for all data-driven transformations, Box-Cox, Dual and log-shift, if no specific values are chosen. Table 1 shows the default intervals implemented for function `ebp`. In the following, the different data-driven transformations are applied with the data of Austrian districts provided in the package (Kreutzmann *et al.* 2019).

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

Box-Cox transformation

The Box-Cox transformation remains the default transformation. The following code produces the same results that are shown in Kreutzmann *et al.* (2019). The estimated transformation parameter equals to 0.6046901. The summary also shows residual diagnostics that suggest a normally distributed random effect, while the Shapiro-Wilk test for the unit-level error rejects normality. A look at the kurtosis and the QQ-plots reveals that the problem lies in the tails. Outlying observations could be one driving factor for these observations.

```
R> ebp_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",
+                threshold = 10885.33)
R> summary(ebp_bc)

Empirical Best Prediction

Call:
 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",
```

```
    threshold = 10885.33)

Out-of-sample domains:  24
In-sample domains:  70

Sample sizes:
Units in sample:  1945
Units in population:  25000
                   Min. 1st Qu. Median      Mean 3rd Qu. Max.
Sample_domains       14    17.0   22.5  27.78571   29.00  200
Population_domains    5   126.5  181.5 265.95745  265.75 5857

Explanatory measures:
 Marginal_R2 Conditional_R2
   0.6325942       0.709266

Residual diagnostics:
               Skewness Kurtosis Shapiro_W    Shapiro_p
Error         0.7523871 9.646993 0.9619824 3.492626e-22
Random_effect 0.4655324 2.837176 0.9760574 1.995328e-01

ICC:  0.2086841

Transformation:
 Transformation Method Optimal_lambda Shift_parameter
        box.cox   reml      0.6046901               0
```

R> qqnorm(ebp_bc)

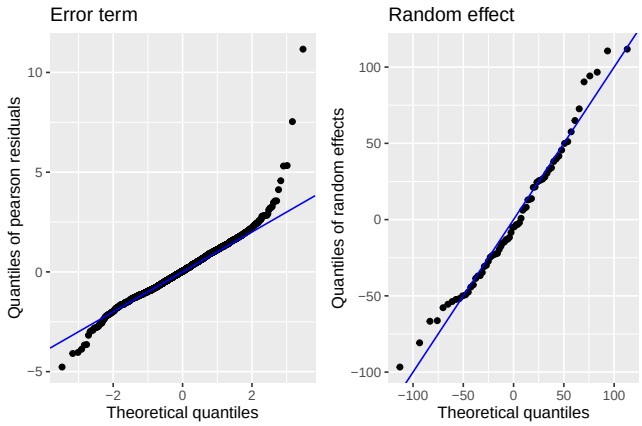


Figure 1: QQ-plots of the error term and random effect using the Box-Cox transformation.

Dual transformation

The Dual transformation in this case is similar to the Box-Cox transformation. Therefore, the estimated transformation parameter differs only slightly with a value of 0.6047161 and also the diagnostics are comparable, with the conclusion of a normally distributed random effect and a distribution of the unit-level error that shows fat tails.

```
R> ebp_dual <- 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",
+                  threshold = 10885.33, transformation = 'dual')
R> summary(ebp_dual)

Empirical Best Prediction

Call:
 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",
    threshold = 10885.33, transformation = "dual")

Out-of-sample domains:  24
In-sample domains:  70

Sample sizes:
Units in sample:  1945
Units in population:  25000
                   Min. 1st Qu. Median      Mean 3rd Qu. Max.
Sample_domains       14    17.0   22.5  27.78571   29.00  200
Population_domains    5   126.5  181.5 265.95745  265.75 5857

Explanatory measures:
 Marginal_R2 Conditional_R2
   0.6325965      0.7092674

Residual diagnostics:
              Skewness Kurtosis Shapiro_W    Shapiro_p
Error         0.752435 9.647438 0.9619800 3.487023e-22
Random_effect 0.465552 2.837214 0.9760562 1.995026e-01

ICC:  0.2086831

Transformation:
 Transformation Method Optimal_lambda Shift_parameter
           dual   reml      0.6047161               0
```

```
R> qqnorm(ebp_dual)
```

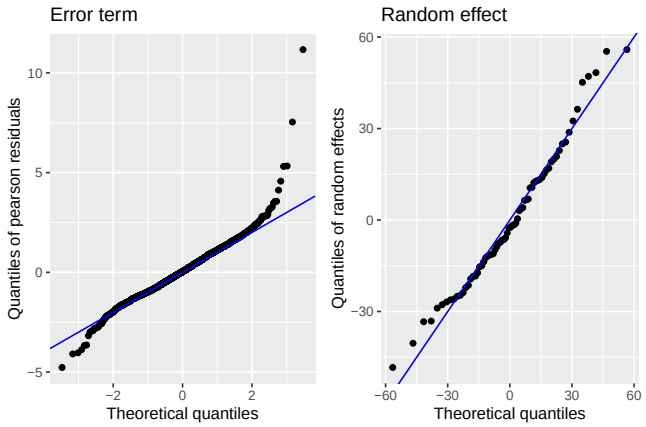


Figure 2: QQ-plots of the error term and random effect using the Dual transformation.

Log-shift transformation

In contrast to the Box-Cox and Dual transformation, the transformation parameter of the log-shift transformation is on the scale of the dependent variable. In this example, λ equals 27907.57. The diagnostics in the summary show that the log-shift transformation slightly improves the kurtosis. From the QQ-plots, it can be concluded that the normality of the unit-level error is most likely rejected because of two outliers.

```
R> ebp_ls <- 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",
+                threshold = 10885.33, transformation = 'log.shift')
R> summary(ebp_ls)

Empirical Best Prediction

Call:
 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",
    threshold = 10885.33, transformation = "log.shift")

Out-of-sample domains:  24
In-sample domains:  70

Sample sizes:
```

```
Units in sample:  1945
Units in population:  25000
                   Min. 1st Qu. Median      Mean 3rd Qu. Max.
Sample_domains       14    17.0   22.5  27.78571   29.00  200
Population_domains    5   126.5  181.5 265.95745  265.75 5857

Explanatory measures:
 Marginal_R2 Conditional_R2
   0.6233538      0.7054886

Residual diagnostics:
               Skewness Kurtosis Shapiro_W    Shapiro_p
Error         0.6222910 7.607189 0.9706711 1.705890e-19
Random_effect 0.4788713 2.726898 0.9737695 1.487627e-01

ICC:  0.2180689

Transformation:
 Transformation Method Optimal_lambda
      log.shift   reml       27907.57
```

R> qqnorm(ebp_ls)

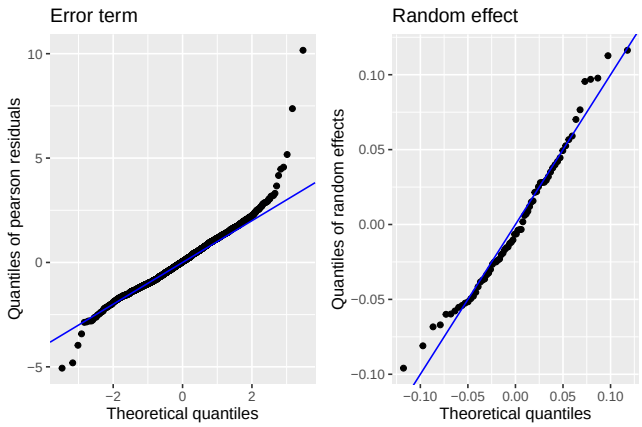


Figure 3: QQ-plots of the error term and random effect using the log-shift transformation.

Summary

While the transformations help to achieve normality for the random effect but not for the unit-level error term in this specific example, they all lead to almost symmetric distributions and show better results compared to an application without transformation.

3. Empirical Best Prediction under informative sampling

3.1. Methodology

Point Estimation

The EBP proposed by Molina and Rao (2010) assumes non-informative sampling, which means that the inclusion probability of the sample is correlated with the outcome variable of interest. The sampling design is said to be non-informative when

$$P(smp \mid y) = P(smp), \forall y \in \mathbb{R}^N, \forall smp,$$

where $P(smp \mid y)$ is the probability of sample smp.
In many applications, the first stage of the sample is selected with probability proportional to its population size. In these cases, the sampling design is very likely to be informative. In these cases, sampling weights should be included in the estimation of the model parameters in order to avoid biased results. Please note, that the inclusion of sample weights only avoid biased results in target domains that are included in the sample (Pfeffermann and Sverchkov 2007). Guadarrama *et al.* (2018) transfer the conditioning idea of the unweighted EBP to the EBP under informative sampling. Instead of conditioning on the unweighted sample mean $\bar{y}_{is}$ during prediction, they condition on the weighted sample mean $\bar{y}_{ij} = w_{i\cdot}^{-1} \sum_{j \in smp_i} w_{ij} y_{ij}$. w_{ij} is the sampling weight for the jth unit in domain i and $w_{i\cdot} = \sum_{j \in smp_i} w_{ij}$ is the sum of sampling weights within domain i.
The pseudo best (PB) estimator for $I_{ij} = i(y_{ij})$ is therefore $\tilde{I}_{ij}^{PB}(\theta) = E[i(y_{ij}) \mid \bar{y}_{iw}; \theta]$ with model parameters θ and the estimator of the additive domain parameter I_i is defined as

$$\tilde{I}_i^{PB}(\theta) = \frac{1}{N_i} \left[\sum_{j \in smp_i} i(y_{ij}) + \sum_{j \in nsmp_i} \tilde{I}_{ij}^{PB}(\theta) \right].$$

The abbreviation nsmp stands for the non-sampled observations in the census. In **emdi**, the PB is implemented as the census PB (CPB) and given by

$$\tilde{I}_i^{CPB}(\theta) = \frac{1}{N_i} \sum_{j \in i} \tilde{I}_{ij}^{PB}(\theta).$$

This means that the indicator is predicted for all observations in the census and not just for the out-of-sample elements. The reason behind the implementation is that sample observations can very rarely be identified in the census. This procedure is also mentioned in of Guadarrama *et al.* (2018).

As with the EBP, the PB estimator depends on the true values of the model parameters $\theta = (\beta, \sigma_u^2, \sigma_e^2)$, which are not known and therefore need to be estimated. θ in the pseudo EB predictor is replaced by a consistent estimator. The resulting predictor is called the pseudo empirical best predictor (PEBP). The authors mention two ways of estimating the model parameters. One feasible approach of Pfeffermann and Sverchkov (2007) is based on the sample likelihood. The likelihood is used to find the maximum likelihood (ML) estimates of the regression coefficients β and of the variances σ_u^2 and σ_e^2. In the second approach, β is estimated using the weighted method of moments of You and Rao (2002). The needed

variance parameters σ_u^2 and σ_e^2 are estimated using ML (or REML). In **emdi**, the second approach is implemented and the variance parameters are estimated by REML.

For out-of-sample observations, the following relationships hold under the nested error population model:

$$y_{ij} \mid \bar{y}_{iw} \overset{ind.}{\sim} \mathcal{N}(\mu^w_{ij|smp}, \sigma^{2w}_{ij|smp}),$$
$$\mu^w_{ij|smp} = x_{ij}^\top \beta + \gamma_{iw}(\bar{y}_{iw} - \bar{x}_{iw}^\top \beta), \quad \sigma^{2w}_{ij|smp} = \sigma_u^2(1-\gamma_{iw}) + \sigma_e^2.$$

The mean $\mu^w_{ij|smp}$ is obtained by replacing the unweighted best predictor of the domain effect u_i by its weighted version, given by $\tilde{u}_{iw} = \gamma_{iw}(\bar{y}_{iw} - \bar{x}_{iw}^\top \beta)$. This approach of conditioning on the weighted sample mean $\bar{y}_{iw}$ protects against bias due to informative sampling in sampled areas.

In **emdi**, a Monte Carlo procedure is applied to approximate the predictor for all indicators using the following algorithm:

1. The dependent variable is transformed according to chosen transformation ('no', 'log') to obtain $T(y_{ij}) = y^*_{ij}$.
2. The sample data is used to estimate the nested error linear regression model
$$y^*_{ij} = x_{ij}^\top \beta + u_i + e_{ij},\ u_i \overset{iid}{\sim} \mathcal{N}(0, \sigma_u^2),\ e_{ij} \overset{iid}{\sim} \mathcal{N}(0, \sigma_e^2)$$
with the `lme` function from the package Pinheiro *et al.* (2021). Please note, that no weights are included within the estimation of the above model using the `lme` function. The shrinkage parameter $\hat{\gamma}_{iw} = \hat{\sigma}_u^2/(\hat{\sigma}_u^2 + \hat{\sigma}_e^2\hat{\delta}_i^2)$, for $\hat{\delta}_i^2 = w_{i\cdot}^{-2}\sum_{j\in smp_i} w_{ij}^2$ is also computed and the coefficients for the fixed effects (following You and Rao 2002) are then obtained as:
$$\hat{\beta} = \left(\sum_{i=1}^{D}\sum_{j=1}^{n_i} w_{ij}x_{ij}(x_{ij} - \hat{\gamma}_{iw}\bar{x}_{iw})^\top\right)^{-1}\left(\sum_{i=1}^{D}\sum_{j=1}^{n_i} w_{ij}(x_{ij} - \hat{\gamma}_{iw}\bar{x}_{iw})y_{ij}\right),$$
where $\bar{x}_{iw} = w_{i\cdot}^{-1}\sum_{j\in smp_i} w_{ij}x_{ij}$.
3. For $l = 1, ..., L$:
   (a) For in-sample domains (domains that are part of the sample data set), a synthetic population of the target variable is generated by $y^{*(l)}_{ij} = x_{ij}^\top\hat{\beta} + \hat{u}_i + \nu_{ij}^{(l)} + e_{ij}^{(l)}$, where $\nu_i^{(l)} \overset{iid}{\sim} \mathcal{N}(0, \hat{\sigma}_u^2(1-\hat{\gamma}_{iw}))$, $e_{ij} \overset{iid}{\sim} \mathcal{N}(0, \hat{\sigma}_e^2)$ and $\hat{u}_i = \hat{\gamma}_{iw}(\bar{y}_{iw} - \bar{x}_{iw}^\top\hat{\beta})$
   For out-of-sample domains (domains with no data in the sample), the conditional expectation of u_i cannot be computed, hence for these domains a synthetic population is generated by using $y^{*(l)}_{ij} = x_{ij}\top\hat{\beta} + \nu_{ij}^{(l)} + e_{ij}^{(l)}$, where $\nu_i^{(l)} \overset{iid}{\sim} \mathcal{N}(0, \hat{\sigma}_u^2))$ and $e_{ij} \overset{iid}{\sim} \mathcal{N}(0, \hat{\sigma}_e^2)$.
   (b) The predicted dependent variable is back-transformed to the original scale $y_{ij}^{(l)} = T^{-1}(y_{ij}^{*(l)})$ and the target indicator $I_i^{(l),PEBP} = I_i(y_{ij}^{(l)})$ is calculated in each domain.
4. Final estimates are computed by taking the mean over the L Monte Carlo simulations in each domain, $\hat{I}_i^{PEBP} = \frac{1}{L}\sum_{l=1}^{L} I_i^{(l),PEBP}$

Parametric bootstrap MSE estimator

Guadarrama *et al.* (2018) moreover propose a parametric bootstrap MSE estimator that ist very similar to the procedure in Molina and Rao (2010), which is based on the method developed by González-Manteiga *et al.* (2008). The bootstrap implemented in **emdi** takes into account, that in applications the sample can can rarely be identified within the census:

1. The same nested error model is fit to the sample data (possibly under transformation) as for the point estimates and the model parameters are obtained $(\hat{\beta}, \hat{\sigma}_u^2, \hat{\sigma}_e^2)$.

2. For $b = 1, ..., B$, with large B, $u_i^{(b)} \sim \mathcal{N}(0, \hat{\sigma}_u^2)$ and $e_{ij}^{(b)} \sim \mathcal{N}(0, \hat{\sigma}_e^2)$, $j = 1, ..., N_i$, $i = 1, ..., D$ are generated independently.

3. B bootstrap populations are generated as

$$y_{ij}^{*(b)} = x_{ij}^{\top}\hat{\beta} + u_i^{(b)} + e_{ij}^{(b)}, \; j = 1, ..., N_i, \; i = 1, ..., D.$$

4. From each bootstrap population the true value of the domain indicator $I_i^{(b)} = N_i^{-1} \sum_{j=1}^{N_i} I(y_{ij}^{(b)}), \; b = 1, ..., B$ is calculated.

5. Additionally a bootstrap sample is generated as

$$y_{ij}^{*(b)} = x_{ij}^{\top}\hat{\beta} + u_i^{(b)} + e_{ij}^{(b)}, \; j = 1, ..., n_i, \; i = 1, ..., D.$$

   This sample is used in conjunction with the known population vectors $x_{ij}, j \in U_i$ to calculate the bootstrap PEBP of I_i, denoted $\hat{I}_i^{PEBP(b)}, b = 1, ..., B$.

6. A bootstrap estimator of MSE$(\hat{I}_i^{PEBP})$ is then given by

$$mse(\hat{I}_i^{PEBP}) = \frac{1}{B} \sum_{b=1}^{B} \left(\hat{I}_i^{PEBP(b)} - I_i^{(b)} \right)^2.$$

3.2. Functionality

Overall, there are only slight changes for the user from the first version of the function `ebp`. Since the Pseudo EBP method uses survey weights in the estimation part, the `ebp` function has a new argument `weights`. The argument defaults to `NULL` and is to be used like the argument `smp_domains`, when the sampling design is informative. The function expects a character string as input for the argument that indicates the name of the weights variable in the sample dataset. The variable itself has to be numeric.

Since Rojas-Perilla *et al.* (2020) find that the usage of data-driven transformations can be favourable for the estimation of the EBP under non-informative sampling, the `transformation` argument of the `ebp` function in **emdi** is set to "box.cox" implying that the dependent variable is transformed with the Box-Cox transformation. Users of the weighted version of the EBP will have to choose "no" for no transformation or "log" for a logarithmic transformation, because data-driven transformations for the PEBP are still a topic for research. If the argument

is not changed an informative message will be displayed and the estimation process will be halted.

While two options to estimate the MSE are provided for the unweighted EBP, the MSE estimation for PEBP allows only for a parametric bootstrap which is the default for both estimation approaches (`boot_type="parametric"`).

Model estimation

The original version of the EBP can still be used without any changes to the arguments of the function.The following function call almost equals the shown example in Kreutzmann *et al.* (2019):

```
R> ebp_noweights <- 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",
+                       threshold = 10885.33, MSE = TRUE)
```

```
 Bootstrap started
 10  of  50  Bootstrap iterations completed          Approximately  00:00:02:06  remaining

 20  of  50  Bootstrap iterations completed          Approximately  00:00:01:33  remaining

 30  of  50  Bootstrap iterations completed          Approximately  00:00:01:02  remaining

 40  of  50  Bootstrap iterations completed          Approximately  00:00:00:31  remaining

 Bootstrap completed
```

When using the PEBP, the aforementioned changes to two arguments are necessary in order to run the model:

- `weights`: Adding the name of the variable that indicates the sampling weights
- `transformation`: Since the default transformation, Box-Cox, is not yet available for the weighted EBP, it needs to be changed to "no" or "log"

```
R> ebp_weights <- 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",
+                     threshold = 10885.33, MSE = TRUE,
+                     weights = "weight", transformation = "log")
```

```
Bootstrap started
10  of  50  Bootstrap iterations completed          Approximately  00:00:01:42  remaining

20  of  50  Bootstrap iterations completed          Approximately  00:00:01:15  remaining

30  of  50  Bootstrap iterations completed          Approximately  00:00:00:49  remaining

40  of  50  Bootstrap iterations completed          Approximately  00:00:00:24  remaining

Bootstrap completed
```

The `model` component of an 'ebp' object that considers weights has a new list element with the weighted coefficients (You and Rao 2002). The other return components for the class are the same as for the unweighted EBP. After running the model, all the S3 methods that are available for the unweighted version of the EBP, can also be used to inspect the estimation results of the weighted EBP. An overview of all available methods can be found with `help(emdiObject)`.

Model diagnostics

The function call returned in the summary indicates if sampling weights are used for the estimation. Furthermore, the PEBP output clarifies that the returned explanatory measures and residual diagnostics belong to the mixed model used for the estimation of the variance components. While the data information as the in- and out-of-sample domains or sample sizes do not differ between the models, differences can be seen in the model diagnostics which is due to the different transformation used.

```
R> # without weights
R> summary(ebp_noweights)

Empirical Best Prediction

Call:
 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",
    threshold = 10885.33, MSE = TRUE)

Out-of-sample domains:  24
In-sample domains:  70

Sample sizes:
Units in sample:  1945
Units in population:  25000
```

```
                   Min. 1st Qu. Median      Mean 3rd Qu. Max.
Sample_domains       14    17.0   22.5  27.78571   29.00  200
Population_domains    5   126.5  181.5 265.95745  265.75 5857

Explanatory measures:
 Marginal_R2 Conditional_R2
   0.6325942       0.709266

Residual diagnostics:
               Skewness Kurtosis Shapiro_W    Shapiro_p
Error         0.7523871 9.646993 0.9619824 3.492626e-22
Random_effect 0.4655324 2.837176 0.9760574 1.995328e-01

ICC:  0.2086841

Transformation:
 Transformation Method Optimal_lambda Shift_parameter
        box.cox   reml      0.6046901               0

R> # with weights
R> summary(ebp_weights)

Empirical Best Prediction

Call:
 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",
    threshold = 10885.33, transformation = "log", MSE = TRUE,
    weights = "weight")

Out-of-sample domains:  24
In-sample domains:  70

Sample sizes:
Units in sample:  1945
Units in population:  25000
                   Min. 1st Qu. Median      Mean 3rd Qu. Max.
Sample_domains       14    17.0   22.5  27.78571   29.00  200
Population_domains    5   126.5  181.5 265.95745  265.75 5857

Explanatory measures for the mixed model:
 Marginal_R2 Conditional_R2
   0.5022296      0.5909727

Residual diagnostics for the mixed model:
```

```
                Skewness  Kurtosis Shapiro_W    Shapiro_p
Error         -2.1828119 17.863231 0.8670156 8.641339e-38
Random_effect -0.6609709  3.361441 0.9682563 7.261244e-02

ICC:  0.1782811

Transformation:
 Transformation Shift_parameter
            log               0
```

The method `plot` works independent of the usage of weights and plots residual diagnostics for the mixed model used in the estimation of the variance components.

```
R> plot(ebp_weights)
```

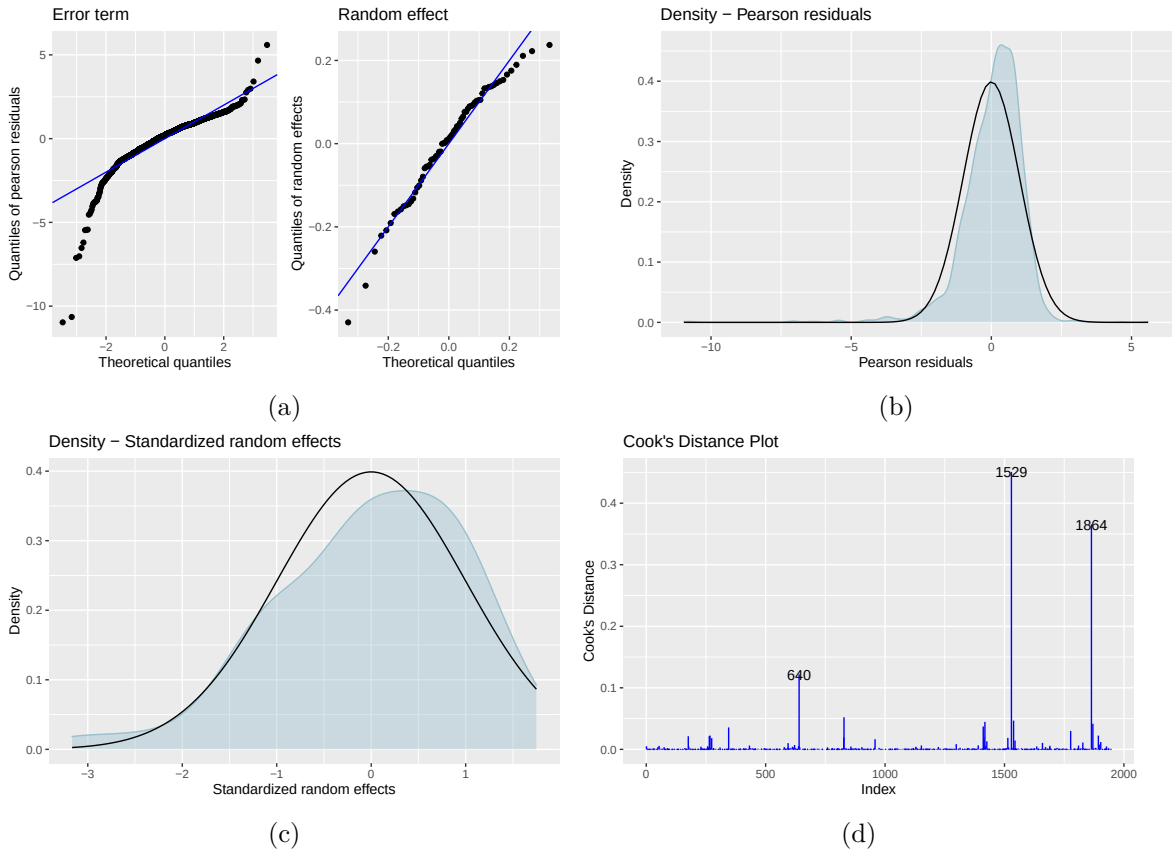


Figure 4: Output of `plot(ebp_weights)`: (a) normal quantile-quantile (Q-Q) plots of the error term and random effects, (b) and (c): kernel densities of the distribution of the error term and random effects (blue) in comparison to a standard normal distribution (black), (d): Cooks distance plot. All results refer to the error terms from the mixed model with log transformation.

To analyse the model coefficients, we added an additional argument `weights` to the `coef.ebp` method, which defaults to `FALSE`. When using the default, the coefficients for the mixed model

are displayed. Setting the argument to `TRUE` returns the weighted regression coefficients as in You and Rao (2002).

```
R> # default
R> head(coef(ebp_weights), 2)
```

```
                (Intercept) genderfemale      eqsize         cash
Neusiedl am See    9.259754  -0.01087928 -0.06553293 2.984645e-05
Oberwart           9.079019  -0.01087928 -0.06553293 2.984645e-05
                   self_empl   unempl_ben      age_ben     surv_ben
Neusiedl am See 2.297232e-05 1.988227e-05 3.017273e-05 2.969203e-05
Oberwart        2.297232e-05 1.988227e-05 3.017273e-05 2.969203e-05
                    sick_ben     dis_ben         rent    fam_allow
Neusiedl am See 2.640426e-05 3.46888e-05 1.459455e-05 3.068898e-06
Oberwart        2.640426e-05 3.46888e-05 1.459455e-05 3.068898e-06
                 house_allow      cap_inv       tax_adj
Neusiedl am See 5.035249e-05 1.752919e-05 -1.194406e-05
Oberwart        5.035249e-05 1.752919e-05 -1.194406e-05
```

```
R> # weighted coefficients
R> coef(ebp_weights, weights = TRUE)
```

```
  (Intercept)  genderfemale        eqsize          cash     self_empl
 9.150540e+00  3.396113e-03 -6.104507e-02  3.272744e-05  2.486388e-05
   unempl_ben       age_ben      surv_ben      sick_ben       dis_ben
 2.182578e-05  3.355522e-05  3.122066e-05  2.847899e-05  3.778051e-05
         rent     fam_allow   house_allow       cap_inv       tax_adj
 1.514303e-05  3.655281e-07  4.582541e-05  1.807290e-05 -1.191243e-05
```

Estimation results

The analysis of estimation results does also not differ between the estimation of the EBP or the PEBP. Exemplarily, we show the first five rows of the head count ratio and the poverty gap for the Austrian districts for both estimation approaches.

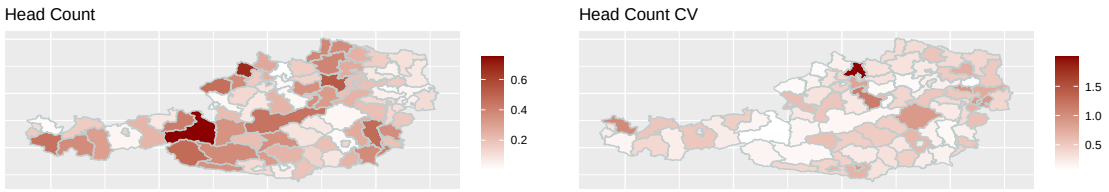
```
R> # without weights
R> head(estimators(ebp_noweights, indicator = c("Head_Count", "Poverty_Gap")))
```

```
               Domain Head_Count Poverty_Gap
1 Eisenstadt-Umgebung 0.03252174 0.006666679
2  Eisenstadt (Stadt) 0.02918919 0.007225508
3             Güssing 0.15459459 0.034668361
4         Jennersdorf 0.35877551 0.099130465
5         Mattersburg 0.07908257 0.015634492
6     Neusiedl am See 0.09316129 0.017250369
```

```
R> # with weights
R> head(estimators(ebp_weights, indicator = c("Head_Count", "Poverty_Gap")))

               Domain Head_Count Poverty_Gap
1 Eisenstadt-Umgebung 0.04939130  0.00844820
2  Eisenstadt (Stadt) 0.03567568  0.00756965
3             Güssing 0.19216216  0.03653299
4         Jennersdorf 0.39306122  0.09112366
5         Mattersburg 0.10733945  0.01791921
6     Neusiedl am See 0.11122581  0.01800804
```

The point and uncertainty estimates can also be plotted on maps to analyse the spatial distribution of e.g., poverty incidence.

```
R> # load shape file
R> load_shapeaustria()
R> # plot for PEBP (weights) for head count ratio
R> map_plot(object = ebp_weights, CV = TRUE,
+           map_obj = shape_austria_dis, indicator = c("Head_Count"),
+           map_dom_id = "PB")
```



(a) (b)

Figure 5: Map of predictions (a) and CV (b) of the head count ratio from the EBP using sampling weights.

Summary

The usage of function `ebp` changes only slightly for the user when weights are added compared to the unweighted option. All diagnostic and analysis tools are available for both options.

4. Specifying a level of aggregation

In many practical SAE applications, several spatial variables are available in the dataset. In these cases, it is possible to specify the random effect at a different level than the target areas

for which results are output. In other words, this features relaxes the requirement that the random effects be specified at the target domain level. Marhuenda *et al.* (2017) discusses such cases in model and design-based simulations. A new argument has been added to the `ebp` function (`aggregate_to`) so specify the target domain level.

4.1. Methodology

Point Estimation

For the independent determination of the target domain level to the domain level for the calculation of the random effects, there is are changes in Step 3 and 4 of the Mote Carlo procedure. These changes only apply to the index for the domains. Therefore, we have two different domain levels and index with i the domains on which the random effect is specified and with i^* the target level on which the results are output. Please note, if no entry is made in `aggregate_to` then $i^* = i$ and both levels coincide.

3. For $l = 1, ..., L$:

   (a) For in-sample domains (domains that are part of the sample data set), a synthetic population of the target variable is generated by $y_{ij}^{*(l)} = x_{ij}^{\top}\hat{\beta} + \hat{u}_i + \nu_{ij}^{(l)} + e_{ij}^{(l)}$, where $\nu_i^{(l)} \overset{iid}{\sim} \mathcal{N}(0, \hat{\sigma}_u^2(1 - \hat{\gamma}_{iw}))$, $e_{ij} \overset{iid}{\sim} \mathcal{N}(0, \hat{\sigma}_e^2)$ and $\hat{u}_i = \hat{\gamma}_{iw}(\bar{y}_{iw} - \bar{x}_{iw}^{\top}\hat{\beta})$
   For out-of-sample domains (domains with no data in the sample), the conditional expectation of u_i cannot be computed, hence for these domains a synthetic population is generated by using $y_{ij}^{*(l)} = x_{ij}\top\hat{\beta} + \nu_{ij}^{(l)} + e_{ij}^{(l)}$, where $\nu_i^{(l)} \overset{iid}{\sim} \mathcal{N}(0, \hat{\sigma}_u^2))$ and $e_{ij} \overset{iid}{\sim} \mathcal{N}(0, \hat{\sigma}_e^2)$.

   (b) The predicted dependent variable is back-transformed to the original scale $y_{ij}^{(l)} = T^{-1}(y_{ij}^{*(l)})$ and the target indicator $I_{i^*}^{(l),PEBP} = I_{i^*}(y_{i^*j}^{(l)})$ is calculated for each target domain (i^*) which can differ from the domain indicated with i for the random effect.

4. Final estimates are computed by taking the mean over the L Monte Carlo simulations in each target domain, $\hat{I}_{i^*}^{PEBP} = \frac{1}{L}\sum_{l=1}^{L} I_{i^*}^{(l),PEBP}$.

This extension for the independent specification of the target level is particularly relevant for non-linear indicators (e.g. the gini-indicator). For linear indicators such as the mean, it is convenient, but the respective point estimate for higher level could also be obtained by a weighted mean.

Parametric bootstrap MSE estimator

Accordingly, the index for the target domain is also adapted for the MSE parametric bootstrap procedure starting with step 4:

4. From each bootstrap population the true value of the target domain indicator $I_{i^*}^{(b)} = N_{i^*}^{-1}\sum_{j=1}^{N_{i^*}} I(y_{i^*j}^{(b)})$, $b = 1, ..., B$ is calculated.

5. Additionally a bootstrap sample is generated as

$$y_{ij}^{*(b)} = x_{ij}^{\top}\hat{\beta} + u_i^{(b)} + e_{ij}^{(b)},\ j = 1, ..., n_i,\ i = 1, ..., D.$$

This sample is used in conjunction with the known population vectors $x_{ij}, j \in U_i$ to calculate the bootstrap PEBP of I_{i^*}, denoted $\hat{I}_{i^*}^{PEBP(b)}, b = 1, ..., B$.

6. A bootstrap estimator of MSE($\hat{I}_{i^*}^{PEBP}$) is then given by

$$mse(\hat{I}_{i^*}^{PEBP}) = \frac{1}{B}\sum_{b=1}^{B}\left(\hat{I}_{i^*}^{PEBP(b)} - I_{i^*}^{(b)}\right)^2.$$

4.2. Functionality

Overall, there are only slight changes for the user from the first version of the function `ebp`. Since the independent target domain level must be defined, if wanted, a new argument (`aggregate_to`) is added to the `ebp` function. This argument defaults to `NULL` and is to be used like the argument `pop_domains`. The function expects a character stringas input for the argument that indicates the name of the domain target level variable in the population data. The variable itself has to be a factor or numeric.

Model estimation

The original version of the EBP can still be used without any changes to the arguments of the function. The following function call almost equals the shown example in Kreutzmann *et al.* (2019):

```
R> ebp_district <- 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",
+                       transformation = "log", na.rm = TRUE)
```

When using independent target domain level the only change is to give this variable to the argument `aggregate_to`. This variable has to be represented in the population data. Here we want to get our results on the state level.

```
R> ebp_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",
+                     transformation = "log", na.rm = TRUE,
+                     aggregate_to = "state")
```

The S3 object for the output does not differ in structure from the one where the level of the random effect and the target level are the same. When using independent target domain level, it must be noted that in the component `ind` and `MSE` the indicators are given at the target level. In the component `model`, the information about the random effect is specified at the level as defined in `pop_domains`.

Estimation results

The estimation results for the EBP with target level state and district are different in the representation as the target domain level differs.

```
R> # output on district level
R> estimators(ebp_state, indicator = c("Mean", "Gini"))

Indicator/s: Mean, Gini
         Domain     Mean      Gini
1    Burgenland 22026.35 0.3564043
2 Lower Austria 20777.26 0.3161001
3        Vienna 21316.04 0.3330059
4     Carinthia 20625.68 0.2958318
5        Styria 20645.42 0.3321182
6 Upper Austria 21588.74 0.3306284
7      Salzburg 19835.62 0.3258470
8         Tyrol 21786.78 0.3732712
9    Vorarlberg 23279.03 0.3356935
```

Instead of showing the output for all 94 districts, we present the results for the first 9 districts, which together comprise the state Burgenland. In addition, we add a variable `proportion` to this table, which describes the proportion of the respective district in the total population of Burgenland.

```
R> # output on state level while the random effect is on district level
R> tab_districts_Burgenland <-
+         estimators(ebp_district, indicator = c("Mean", "Gini"))$ind[1:9,]
R> tab_districts_Burgenland$proportion <-
+   table(eusilcA_pop$district)[1:9] / sum(table(eusilcA_pop$district)[1:9])
R> tab_districts_Burgenland

               Domain     Mean      Gini  proportion
1 Eisenstadt-Umgebung 29270.01 0.2976750 0.143929912
2  Eisenstadt (Stadt) 81636.64 0.5270380 0.046307885
3             Güssing 16966.65 0.2101298 0.092615770
4         Jennersdorf 13557.30 0.2031489 0.061326658
5         Mattersburg 21318.81 0.2508716 0.136420526
6     Neusiedl am See 19389.69 0.2173187 0.193992491
7      Oberpullendorf 17536.10 0.2189079 0.131414268
```

```
8            Oberwart 13635.81 0.2065571 0.187734668
9        Rust (Stadt) 15358.87 0.1729022 0.006257822
```

For linear indicators, such as the mean, the indicator at state level can be determined directly from the corresponding values of the districts using the proportions of the district to the total state population. For non-linear indicators, this relationship does not exist.

```
R> sum(tab_districts_Burgenland$Mean * tab_districts_Burgenland$proportion)

[1] 22026.35

R> sum(tab_districts_Burgenland$Gini * tab_districts_Burgenland$proportion)

[1] 0.2441799
```

The results can be shown on maps on variable levels to analyse spatial distributions of poverty indicators like the Gini index. To reproduce the map on district level, please download the Austrian shape file for the states (https://data-synergis.opendata.arcgis.com/maps/a16c7b8ef72f4ec2b36f7c7ebbcdf2e5) and name it `shape_austria_state`.

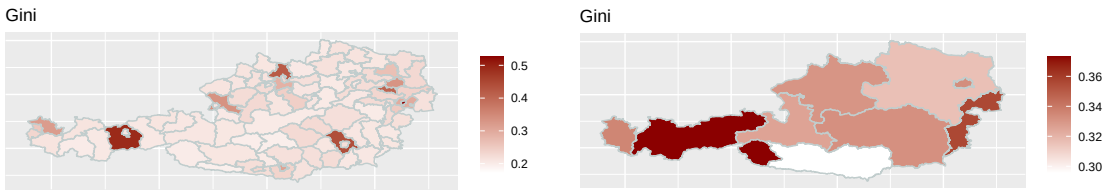
```
R> # Load shape file
R> load_shapeaustria()

R> # Create map for Gini indicator on district level
R> map_plot(object = ebp_district, map_obj = shape_austria_dis,
            indicator = c("Gini"), map_dom_id = "PB")

R> # Create map for Gini indicator on state level using map_tab for assignment
R> map_plot(object = ebp_state, map_obj = shape_austria_state,
            indicator = c("Gini"), map_dom_id = "BL", map_tab = map_tab)
```

Summary

The usage of function `ebp` changes only slightly for the user when a different aggregation level for the results is definable. All diagnostic and analysis tools are available. Please note that the model diagnostics refer to the level at which the random effect is specified. In contrast, the results are presented at the target level for aggregation.

5. Inclusion of population weights

The EBP is a unit-level SAE model. It is therefore calculated at the individual level. However, it is often convenient for researchers to use population data at the household or cluster level. In such cases, the household size or the number of inhabitants per grid can be used for weighting (Masaki *et al.* 2020; Elbers *et al.* 2003). These population weights are utilized when aggregating results to target domains and are distinct from the survey weights used to estimate the model that were discussed in Section 3 above. The argument `survey_weight`



(a) (b)

Figure 6: Map of predictions on district (a) and state (b) level of the Gini indicator from the EBP with random effects on district level and various output level.

indicates a variable in the survey and `population_weight` a variable in the population. If using the argument `population_weight` the indicators within each Monte-Carlo simulation are calculated as a weighted indicator using population weights. The changing interpretation of the results will be explained later in detail for the head count ratio.

5.1. Methodology

In terms of methodology, only the calculation of the indicator changes within each Monte Carlo repetition. The indicator is formed in step 3 (b) for the point estimates. All other extensions that also affect the indicator, such as (a) using survey weights or (b) defining a independent level for aggregation can be combined with the population weights. The basic indicator $I_i^{(l)}$ for each Monte-Carlo $l = 1, ..., L$ and every domain $i = 1, ..., D$ in the population is defined for the variable of interest y estimated for each unit $j = 1, ..., N_i$ as

$$I_i^{(l)} = I_i\left(y_{ij}^{(l)}, w_{ij}\right).$$

In addition, the weights from the population (w_{ij}) must now be taken into account in the calculation of the indicator:

$$I_i^{(l)} = I_i\left(y_{ij}^{(l)}, w_{ij}\right).$$

Using the head count ratio as an example, this would lead to weighted version for calculating the indicator

$$HCR_i^{(l)} = HCR_i\left(y_{ij}^{(l)}\right) = \frac{\sum_{j=1}^{N_i} p_{ij}}{N_i} \qquad HCR_i^{(l)} = HCR_i\left(y_{ij}^{(l)}, w_{ij}\right) = \frac{\sum_{j=1}^{N_i} p_{ij} w_{ij}}{\sum_{j=1}^{N_i} w_{ij}},$$

where $p_{ij} = 1$ indicates that the statistics y_{ij} form unit j in domain i is underneath the poverty threshold and $p_{ij} = 0$ that it is above this threshold.

To calculated the bootstrap MSE estimator, the indicator in Step 5 has to be adjusted in the same way to allow for population weights.

5.2. Functionality

Model estimation

The original version of the EBP can still be used without any changes to the arguments of the function. The following function call almost equals the shown example in Kreutzmann *et al.* (2019):

```
R> # EBP without population weights
R> ebp_unit <- 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", L = 1000,
+                   transformation = "log", na.rm = TRUE)
```

When using population weights the only change is to give this variable to the argument `population_weight`. This variable has to be represented in the population data. Here we want to take the householdsize into account for calculating the indicators.

```
R> # EBP with population weights
R> ebp_eqsize <- 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", L = 1000,
+                     transformation = "log", na.rm = TRUE,
+                     pop_weight = "eqsize")
```

The S3 object for the output does not differ in structure from the model without population weights.

Estimation results

In the following, the estimation results for without and with population weights are shown for the head count ratio.

```
R> head(estimators(ebp_unit, indicator = c("Head_Count")))
```

```
               Domain Head_Count
1 Eisenstadt-Umgebung 0.05020870
2  Eisenstadt (Stadt) 0.03132432
3             Güssing 0.19366216
4         Jennersdorf 0.35732653
5         Mattersburg 0.10378899
6     Neusiedl am See 0.09581935
```

```
R> head(estimators(ebp_eqsize, indicator = c("Head_Count")))
```

```
               Domain Head_Count
1 Eisenstadt-Umgebung 0.05203274
2  Eisenstadt (Stadt) 0.02843443
3             Güssing 0.18600426
4         Jennersdorf 0.34885788
5         Mattersburg 0.10472072
6     Neusiedl am See 0.08825237
```

The results differ slightly, because household size is correlated with poverty status. When interpreting the results, it should be specified exactly what is being addressed: (1) without population weights (`ebp_unit`) the result refers to the share of households below the poverty threshold and (2) with population weights (`ebp_equsize`) the result refers to the share of persons belonging to households below the poverty line.

Custom Indicator

The indicator can now depend on the population weights, so weights must always be indicated when the indicator is to be determined.

```
R> ebp_eqsize_custom_indicator <- 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", L = 1000,
+        transformation = "log",
+        custom_indicator = list(HCR_singleHH =
+          function(y, pop_weights, threshold) {
+            mean(y[weights == 1] < threshold)
+        }),
+        na.rm = TRUE, pop_weight = "eqsize")

R> head(estimators(ebp_eqsize_custom_indicator,
+                   indicator = c("Head_Count", "HCR_singleHH")))
```

```
               Domain Head_Count HCR_singleHH
1 Eisenstadt-Umgebung 0.04568452   0.01428571
2  Eisenstadt (Stadt) 0.03688525   0.00000000
3             Güssing 0.15349233   0.18800000
4         Jennersdorf 0.38307494   0.33750000
5         Mattersburg 0.09549892   0.07916667
6     Neusiedl am See 0.09739748   0.14047619
```

The argument `custom_indicator` allows for a customized indicator. For this purpose, a function must be defined containing the arguments `y`, `weights`, and `threshold`. Such a

customized indicator allows a variety of statistics to calculated with `ebp`. In the present case, `HCR_singleHH` is defined to estimate the share of single households that are below the poverty line respectively the share of persons living in single households that are below the poverty line. Here, we get the same results regardless of whether we still include the population weights for a weighted mean, since the `eqsize` of a single households is 1.

Summary

The usage of function `ebp` changes only slightly for the user when population weights are definable. All diagnostic and analysis tools are available. Please note, that the diagnostic on weights refer to the survey weights as explained in Chapter 3.

5.3. Conclusion

This vignette shows the most recent changes of function `ebp` in the R package **emdi**: (a) additional data-driven transformations, (b) the inclusion of sampling weights into the estimation procedure of the EBP (c) the option for an independent determination for the level of aggregation (d) the possibility to include population weights. A topic for further research is the inclusion of the above described data-driven transformations in the estimation of the PEBP.

Acknowledgments

The authors gratefully acknowledge financial support from the UK Foreign, Commonwealth & Development Office and from the World Bank Group. We further thank Guadarrama *et al.* (2018) for the provision of code along with the paper.

References

Battese GE, Harter RM, Fuller WA (1988). "An Error-Components Model for Prediction of County Crop Areas using Survey and Satellite Data." *Journal of the American Statistical Association*, **83**(401), 28–36. doi:10.1080/01621459.1988.10478561.

Box G, Cox D (1964). "An Analysis of Transformations." *Journal of the Royal Statistical Society B*, **26**(2), 211–252.

Diallo M, Rao J (2014). "Small Area Estimation of Complex Parameters Under Unit-level Models with Skew-Normal Errors." *JSM 2014*, Survey Research Methods Section.

Elbers C, Lanjouw J, Lanjouw P (2003). "Micro-level estimation of poverty and inequality." *Econometrica*, **71**(1), 355–364.

Feng Q, Hannig J, Marron JS (2016). "A Note on Automatic Data Transformation." *Stat*, **5**(1), 82–87. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/sta4.104.

González-Manteiga W, Lombardía M, Molina I, Morales D, Santamaría L (2008). "Analytic and Bootstrap Approximations of Prediction Errors Under a Multivariate Fay-Herriot Model." *Computational Statistics and Data Analysis*, **52**(12), 5242–5252.

Graf M, Marín J, Molina I (2019). "A generalized mixed model for skewed distributions applied to small area estimation." *TEST*, **28**, 565–597.

Guadarrama M, Molina I, Rao J (2016). "A comparison of small area estimation methods for poverty mapping." *Joint Issue: Statistics in Transition New Series Survey Methodology*, **17**(1), 41–66.

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

Gurka M, Edwards L, Muller K, Kupper L (2006). "Extending the Box-Cox transformation to the linear mixed model." *Journal of the Royal Statistical Society A*, **169**(Part 2), 273–288.

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, Rao JNK (2017). "Poverty Mapping in Small Areas under a Twofold Nested Error Regression Model." **180**(4), 1111–1136.

Marino M, Ranalli MG SN, Alfo M (2019). "Semiparametric Empirical Best Prediction for Small Area Estimation of Unemployment Indicators." *Annals of Applied Statistics*, **13**(2), 1166–1197.

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*, pp. 1–17.

Molina I, Rao J (2010). "Small Area Estimation of Poverty Indicators." *The Canadian Journal of Statistics*, **38**(3), 369–385. doi:10.1002/cjs.10051.

Pfeffermann D, Sverchkov M (2007). "Small-Area Estimation under Informative Probability Sampling of Areas and within the Selected Areas." *Journal of the American Statistical Association*, **102**(480), 1427–1439.

Pinheiro J, Bates D, DebRoy S, Sarkar D, R Core Team (2021). *nlme: Linear and Nonlinear Mixed Effects Models*. R package version 3.1-152, URL https://CRAN.R-project.org/package=nlme.

Rojas-Perilla N (2018). *The Use of Data-Driven Transformations and their Application in Small Area Estimation*. Ph.D. thesis, Freie Universität Berlin.

Rojas-Perilla N, Pannier S, Schmid T, Tzavidis N (2020). "Data-driven transformations in small area estimation." *Journal of the Royal Statistical Society A*, **183**(1), 121–148.

Yang Z (2006). "A modified family of power transformations." *Economics Letters*, **92**(1), 14–19. URL doi:10.1016/j.econlet.2006.01.011.

You Y, Rao J (2002). "A Pseudo-Empirical Best Linear Unbiased Prediction Approach to Small Area Estimation Using Survey Weights." *The Canadian Journal of Statistics*, **30**(3), 431–439.

Affiliation:

Felix Skarke, Ann-Kristin Kreutzmann
Institute for Statistics and Econometrics
School of Business & Economics
Freie Universität Berlin
Garystr. 21, 14195 Berlin, Germany
E-mail: felix.skarke@fu-berlin.de, ann-kristin.kreutzmann@fu-berlin.de

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