The R Package tipsae: Tools for Mapping Proportions and Indicators on the Unit Interval

Silvia De Nicolò
Università di Bologna

Aldo Gardini
Università di Bologna

Abstract

The **tipsae** package implements a set of small area estimation tools for mapping proportions and indicators defined on the unit interval. It provides for small area models defined at area level, including the classical Beta regression, Zero and/or One inflated Beta and Flexible Beta ones, possibly accounting for spatial and/or temporal dependency structures. The models, developed within a Bayesian framework, are estimated through Stan language, allowing fast estimation and customized parallel computing. The additional features of the **tipsae** package, such as diagnostics, visualization and exporting functions as well as variance smoothing and benchmarking functions, improve the user experience through the entire process of estimation, validation and outcome presentation. A Shiny application with a user-friendly interface further eases the implementation of Bayesian models for SAE analysis.

Keywords: Bayesian Inference, Beta Regression Models, Small Area Estimation, Shiny, Stan.

1. Introduction

The growing demand for timely and reliable statistical estimates leads to extensive exploitation of survey data at an increasingly greater level of disaggregation. However, domains or areas of study are often different from the ones for which the survey was originally planned, leading to possibly unreliable direct estimates due to observations-poor samples. Small Area Estimation (SAE) tackles this problem by providing a set of indirect estimation techniques, relying on external information, which borrow strength across areas and increase the efficiency of the estimates. Indirect estimators based on explicit regression models are labelled model-based estimators and assume a relationship between the target variable and explanatory variables, which remains constant across areas. Classical small area models embrace two basic linear mixed models: the Fay-Herriot model and the Battese-Harter-Fuller model, which are foundational for the strand of area-level models and unit-level models, respectively (Rao and Molina 2015). While the former relates area-specific target quantities to area covariates, the second one relates individual observations of the underlying variables of interest to individual covariates.

Hereafter, we focus on area-level models due to their practical convenience. They require, in fact, only data aggregated at the area-level, avoiding both computational and data disclosure issues. In area-level contexts, a well-established body of literature is concentrated on Gaussian models. However, many quantities of interest have specific features that are not considered in the Gaussian setting and need to be accounted for, such as a bounded or double bounded support and distributions markedly skewed or heavy-tailed. Specifically, we focus on unit interval responses, common in SAE modelling because of the growing need for rates and proportions releases in official statistics, such as Head-Count Ratio for poverty mapping or Health Insurance Coverage rates. Not to mention the treatment of other measures of interest defined in (0,1) or [0,1], such as some inequality measures (e.g., Gini index).

In this regard, two different bodies of literature revolve around linear mixed models with suitable transformations (Rao and Molina 2015) and Beta regression models (Janicki 2020). For the first approach, we recall the works by Marhuenda, Molina, and Morales (2013); Marhuenda, Morales, and del Carmen Pardo (2014), Morales, Pagliarella, and Salvatore (2015), and Esteban, Morales, Pérez, and Santamaría (2012); Esteban, Lombardía, López-Vizcaíno, Morales, and Pérez (2020) that provide Fay-Herriot extensions to deal with proportions. The second strand focus on classical Beta regression, both in the univariate case (Liu, Lahiri, and Kalton 2007; Bauder, Luery, and Szelepka 2015; Fabrizi and Trivisano 2016; Giovinazzi and Cocchi 2021) and in the multivariate ones (Fabrizi, Ferrante, Pacei, and Trivisano 2011; Souza and Moura 2016), considering also zero and/or one inflated extensions (Wieczorek, Nugent, and Hawala 2012; Fabrizi, Ferrante, and Trivisano 2016, 2020). Lastly, a Beta mixture approach in SAE has been proposed by De Nicolò, Ferrante, and Pacei (2021).

By considering the SAE field as a whole, there is a clear imbalance between a plethora of methodological proposals defined in academic literature and the tight circle of methods actually used in official statistics and applied researches. A bridge-building process between methodological and applied fields is needed, involving collaboration, dissemination, and development of user-friendly tools to facilitate tough steps. With the latter aim, several routines for SAE have been released by developer teams of R (R Core Team 2021), SAS (SAS Institute Inc. 2003), SPSS (IBM Corporation 2010), and STATA (Stata Corporation 2007). Our focus is on R routines due to flexibility and availability reasons as well as for the equipment of complementary tools. Several R packages have been developed to implement SAE tools, and in the following, we attempt to provide a clear overview focusing on model-based methods.

In general, the most complete released packages are:

- **sae** (Molina and Marhuenda 2015). It implements a wide range of small area methods from a frequentist perspective, including both area-level and unit-level models.
- **emdi** (Kreutzmann, Pannier, Rojas-Perilla, Schmid, Templ, and Tzavidis 2019). It allows making inference on both area-level and unit-level models in a frequentist framework, providing model diagnostics, plots, and exporting tools.
- **mcmcsae** (Boonstra 2021). It comprises hierarchical area and unit-level models estimated via Markov Chain Monte Carlo (MCMC) simulation, allowing for spatial and temporal dependencies. It includes different prior settings, model diagnostics, and posterior predictive checks functions.

Among listed packages, only the **emdi** package directly accounts for unit interval responses at area-level by providing the arc-sin transformation in a Gaussian setting (Schmid, Bruckschen, Salvati, and Zbiranski 2017). Thus, while a Fay-Herriot model for unit interval responses may be implemented via existing packages, Beta-based small area models lack proper implementations.

The **tipsae** package (De Nicolò and Gardini 2022) aims at filling this gap by implementing Beta-based small area models specified at the area-level on measures that can assume values in $(0,1)$, $[0,1)$, $(0,1]$, and $[0,1]$ intervals. We decided to operate in a Bayesian fashion in order to exploit the advantages brought by approaching this inferential framework via MCMC methods. For instance, it is possible to easily manage non-Gaussian assumptions, incorporate structured random effects, obtain straightforward estimates for out-of-sample areas, and capture the uncertainty about all target parameters through posterior inference. Nowadays, several tools are available to implement Bayesian models with probabilistic languages: our choice falls on Stan (Carpenter, Gelman, Hoffman, Lee, Goodrich, Betancourt, Brubaker, Guo, Li, and Riddell 2017), that can be easily employed to fit statistical models within R packages thanks to the tools provided by the **rstantools** package (Gabry, Goodrich, and Lysy 2020).

The main features of the **tipsae** package are listed in the following:

- It includes a variety of area-level models based on the Beta likelihood. Besides the standard Beta-regression model, Zero and/or One Inflated Beta (ZOIB) and Flexible Beta models can be chosen. Moreover, particular dependence structures can be modelled, including spatial and/or temporal random effects.

- It implements an efficient Hamiltonian Monte Carlo (HMC) fitting algorithm and customized parallel computing imported from **rstan** (Stan Development Team 2020). We also tested other languages that build MCMC samplers, and Stan turned out to be the most efficient one for Beta regression models, which are particularly tricky to handle due to the non-orthogonality between location and scale parameters.

- The '`stanfit`' `S4` object produced by the **rstan** package can be exploited to check convergence, monitor sampler diagnostics, and, lastly, perform an exhaustive posterior analysis, relying on existing tools such as **loo** (Vehtari, Gabry, Magnusson, Yao, Bürkner, Paananen, and Gelman 2020) and **bayesplot** (Gabry and Mahr 2021) packages. In this way, users familiar with posterior predictive checks can carefully assess the model performance.

- Specific diagnostics for small area models are produced by ad-hoc functions, facing the most relevant aspects to deepen within the SAE framework. We implemented both visualization tools for graphical assessments and functions that easily export the final results. Moreover, variance smoothing routines and benchmarking procedures are also provided, remarking that, to the best of our knowledge, the first tool is not available in any existing SAE package.

- To further facilitate the workflow for non-expert users of R, a Shiny application (Chang, Cheng, Allaire, Sievert, Schloerke, Xie, Allen, McPherson, Dipert, and Borges 2021) with an intuitive graphical user interface can be launched through the `runShiny_tipsae()` function. The application assists the user in carrying out a complete SAE analysis, exploiting all the main features of the **tipsae** package.

The paper is organized as follows: covered models and implemented methodology are set out in Section 2, the datasets made available in the package are presented in Section 3, while Section 4 provides a step-by-step description of inputs and outputs of the available functions. Section 5 outlines the features of the Shiny application and, eventually, Section 6 contains some concluding remarks, discussing possible extensions that could be supplied.

2. Methodology

In this section, the theory behind the statistical methods implemented in the **tipsae** package is summarized. The main aspects are those related to the area-level models for indices and proportions that can be estimated using the function `fit_sae()`.

From now on, we consider a finite population of size N that is partitioned into D small areas having sizes $N_1, \dots, N_D$. We are interested in estimating a generic measure defined on the unit interval that we denote as θ_d, $d = 1, \dots, D$. To this aim, a random sample of size n is drawn from the whole population using a possibly complex survey design, obtaining subsamples of sizes $n_1, \dots, n_D$, specified for each domain. Among them, we define the first $\tilde{D}$ domains, with $\tilde{D} \leq D$ as the ones actually observed, i.e., with $n_d > 0$. The observations recorded at the individual level are aggregated to produce the direct estimates y_d, that are stored in the vector $\boldsymbol{y}$ and are the observed determinations of the direct estimator Y_d for a quantity of interest θ_d, with $d = 1, \dots \tilde{D}$. The Bayesian area-level model is specified for Y_d, including also a set of auxiliary variables $\mathbf{x}_d$, which are assumed to be available for each domain.

The details about the statistical models that can be set through the argument `likelihood` are discussed in Section 2.1. Furthermore, a small area model usually includes also random effects in the linear predictor. The random effect part, hereafter indicated with e_d, can incorporate either a temporal and/or a spatial dependency structure, as will be discussed in Section 2.2, devoted to the prior specification settings. In addition, different prior assumptions can be specified for the unstructured random effects, allowing for robust and shrinking priors.

In small area models, the dispersion parameters are generally assumed as given and previously estimated from the data. Separate estimation could involve a smoothing procedure to refine the sampling variances estimates and reduce their errors. Section 2.3 describes the proposed algorithms to carry out this step if required. Eventually, Section 2.4 outlines the main aspects of posterior inference: we will mainly focus on the out-of-sample treatment, diagnostics, and goodness-of-fit tools employed to validate or select the models and, lastly, the benchmarking procedures complementing SAE analysis.

2.1. Area-Level Models: Likelihoods

The statistical models available in **tipsae** are set out in the following sections, whereas a comprehensive overview of the key quantities under each model is provided in Table 1. In particular, we specify the response support, the conditional expectation, constituting the predictor for θ_d, the conditional variance, allowed parametrizations, and the out-of-sample predictor (denoted with θ_d^{oos}). From now on, $\boldsymbol{\eta}$ indicates the vector of all the model parameters.

The Beta Model

Let us consider the mean-precision parametrization of the Beta random variable (Ferrari and Cribari-Neto 2004): in this case, if $Y \sim \text{Beta}(\mu\phi, (1-\mu)\phi)$, then its probability density function is

$$f_B(y; \mu, \phi) = \frac{\Gamma(\phi)}{\Gamma(\mu\phi)\,\Gamma((1-\mu)\phi)} y^{\mu\phi-1}(1-y)^{(1-\mu)\phi-1}, \quad y \in (0,1),$$

where $\mu \in (0,1)$ is the location parameter and $\phi \in (0, +\infty)$ is the dispersion one. In SAE

Model	Support	$\theta_d = \mathbb{E}\left[Y_d \mid \boldsymbol{\eta}\right]$	$\mathbb{V}\left[Y_d \mid \boldsymbol{\eta}\right]$	Admitted `type_disp`	Predictor for θ_d^{oos}
`"beta"`	$(0;1)$	μ_d	$\frac{\mu_d(1-\mu_d)}{\phi_d+1}$	Both	$\text{logit}^{-1}(\boldsymbol{x}_d^T\boldsymbol{\beta} + e_d)$
`"flexbeta"`	$(0;1)$	$p\lambda_{1d} + (1-p)\lambda_{2d}$	$\frac{\theta_d(1-\theta_d)+p(1-p)\phi_d(\lambda_{1d}-\lambda_{2d})^2}{\phi_d+1}$	`"var"`	-
`"Infbeta0"`	$[0;1)$	$(1-p_d^z)\mu_d$	$(1-p_d^0)\left[\frac{\mu_d(1-\mu_d)}{\phi_d+1} + p_d^0\mu_d^2\right]$	`"neff"`	$(1-p_d^0)\text{logit}^{-1}(\boldsymbol{x}_d^T\boldsymbol{\beta} + e_d)$
`"Infbeta1"`	$(0;1]$	$p_d^o + (1-p_d^o)\mu_d$	$(1-p_d^1)\left[\frac{\mu_d(1-\mu_d)}{\phi_d+1} + p_d^1(1-\mu_d)^2\right]$	`"neff"`	$p_d^1 + (1-p_d^1)\text{logit}^{-1}(\boldsymbol{x}_d^T\boldsymbol{\beta} + e_d)$
`"Infbeta01"`	$[0;1]$	$p_d^o + (1-p_d^z-p_d^o)\mu_d$	$p_d^o(1-\zeta_d) + (1-\alpha_d)\times$ $\times\left[\frac{\mu_d(1-\mu_d)}{\phi_d+1} + \alpha_d(\zeta_d-\mu_d)^2\right]$	`"neff"`	$p_d^1 + (1-p_d^0-p_d^1)\times$ $\times\text{logit}^{-1}(\boldsymbol{x}_d^T\boldsymbol{\beta} + e_d)$

Table 1: Relevant quantities for each model implemented in **tipsae**.

context, the Beta regression area-level model is usually specified as

$$\begin{aligned} Y_d|\mu_d, \phi_d &\overset{ind}{\sim} \text{Beta}\left(\mu_d\phi_d, (1-\mu_d)\phi_d\right), \\ \text{logit}\left(\mu_d\right) &= \mathbf{x}_d^T\boldsymbol{\beta} + e_d, \quad d = 1, \dots, D; \end{aligned}$$

where $\boldsymbol{\beta}$ is the vector of regression coefficients and ϕ_d is the area specific dispersion parameter, usually assumed to be known to guarantee identifiability. Recalling the expression of $\mathbb{V}\left[Y_d|\boldsymbol{\eta}\right]$ from Table 1, it can be shown that, when the target response is a proportion, the parameter ϕ_d is related to the effective sample size, i.e., the corresponding sample size under simple random sampling (Janicki 2020). For a more complete explanation of those aspects, we refer to the discussion in Section 2.3. On the other hand, if a generic indicator (e.g., Gini index) is considered, the meaning of ϕ_d becomes less clear. For this reason, we let the user specify the model parametrization (argument `type_disp`), choosing between:

- `"neff"` option, namely an estimate of the effective sample size $\phi_d + 1$ is provided;
- `"var"` option, in which an estimate of the sampling variance of the direct estimator i.e., $\widehat{\mathbb{V}}[Y_d]$, is used. In this case, the parameters ϕ_d are retrieved using the relations in Table 1, replacing $\mathbb{V}\left[Y_d|\boldsymbol{\eta}\right]$ with $\widehat{\mathbb{V}}[Y_d]$, and substantially changing model parameterization.

The Flexible Beta Model

When the distribution of the response is characterized by heavy tails and/or high skewness, the standard Beta regression could fail in properly modelling Y_d (Bayes, Bazán, and García 2012; Migliorati, Di Brisco, and Ongaro 2018). To improve the model performances in these conditions, the standard Beta distribution can be replaced by the Flexible Beta distribution. The Flexible Beta small area model has been proposed by De Nicolò *et al.* (2021). It is defined as a mixture of two Beta random variables having a common dispersion parameter ϕ_d:

$$\begin{aligned} Y_d|\lambda_{1d}, \lambda_{2d}, \phi_d, p &\overset{ind}{\sim} p \ \text{Beta}\left(\lambda_{1d}\phi_d, (1-\lambda_{1d})\phi_d\right) + \\ &\qquad + (1-p)\ \text{Beta}\left(\lambda_{2d}\phi_d, (1-\lambda_{2d})\phi_d\right), \\ \text{logit}\left(\lambda_{2d}\right) &= \mathbf{x}_d^T\boldsymbol{\beta} + e_d, \qquad d = 1, \dots, D. \end{aligned}$$

In this case, only the direct estimator variance (i.e., `disp_type = "var"`) can be used as input to determine the dispersion parameter of the model. Therefore, ϕ_d is expressed as a function of the sampling variances and other model parameters (see the expression of $\mathbb{V}\left[Y_d|\boldsymbol{\eta}\right]$ in Table 1). The Flexible Beta distribution is characterized by four parameters: this enhances the model flexibility, if compared to the standard Beta distribution, leading to better performances in modelling not well-behaved measures and, consequently, reducing the bias of model-based estimators.

The Zero-One Inflated Beta Model

The supports of Beta and Flexible Beta models do not include the extremes 0 and 1. However, in some applications, zero and one values are observed, and a model able to encompass them is required. Therefore, following Wieczorek *et al.* (2012), we include in the package the ZOIB model, specified as:

$$\begin{aligned}
&Y_d|\mu_d, \phi_d, p_d^z, p_d^o \overset{ind}{\sim} p_d^z \mathbb{1}\{Y_d = 0\} + p_d^o \mathbb{1}\{Y_d = 1\} + \\
&\qquad\qquad\qquad + (1 - p_d^z - p_d^o)\text{Beta}\left(\mu_d\phi_d, (1-\mu_d)\phi_d\right)\mathbb{1}\{0 < Y_d < 1\} \\
&\text{logit}\,(p_d^z) = \mathbf{x}_d^T \boldsymbol{\beta}_p^z, \quad \text{logit}\,(p_d^o) = \mathbf{x}_d^T \boldsymbol{\beta}_p^o, \\
&\text{logit}\,(\mu_d) = \mathbf{x}_d^T \boldsymbol{\beta} + e_d, \quad d = 1, \dots, D;
\end{aligned}$$

where p_d^z and p_d^o denote the probabilities of observing zero and one values, respectively. They are modelled by means of a logit regression model having coefficients $\boldsymbol{\beta}_p^z$ and $\boldsymbol{\beta}_p^o$. The notation $\mathbb{1}\{A\}$ defines the indicator function that assumes value 1 if the event A is observed, and 0 otherwise. The user can specify a model that accounts both for zeroes and ones setting `likelihood = "Infbeta01"`; however, simpler versions inflating only the ones or the zeroes are also available (`"Infbeta1"` and `"Infbeta0"`, respectively). Relevant quantities for each version of the ZOIB model are listed in Table 1, having defined $\alpha_d = p_d^z + p_d^o$ and $\zeta_d = p_d^o/\alpha_d$. For further details, see Ospina and Ferrari (2010).

2.2. Prior distributions

To facilitate practitioners, standard wide-range prior distributions are assumed for the parameters included in the model. Starting from the priors for the regression coefficients, we decided to follow the default prior specification strategy of the popular **rstanarm** package (Goodrich, Gabry, Ali, and Brilleman 2020). Firstly, auxiliary variables are standardized in order to avoid issues related to possibly different magnitudes. Thus, posterior results for the regression coefficients must be interpreted accordingly. A weakly informative prior for the intercept β_0 is specified:

$$\beta_0 \sim \mathcal{N}(0, 2.5^2),$$

and independent normal priors are also assigned to the coefficients related to standardized covariates:

$$\beta_j \overset{ind}{\sim} \mathcal{N}(0, 2.5^2), \;\; j = 1, \dots, p.$$

Note that the same prior setting is also assumed for coefficients $\boldsymbol{\beta}_p^z$ and $\boldsymbol{\beta}_p^o$ involved in ZOIB models.

As regards the Flexible Beta model, we additionally specify the following priors for the mixing probability p and the differences between the means of mixture components:

$$\begin{aligned}
&p \sim \text{Beta}(2, 2), \\
&\lambda_{1d} - \lambda_{2d}|p, \lambda_{2d} \sim \text{Unif}\left(0, \min\left\{\frac{1-\lambda_{2d}}{p}, \sqrt{\frac{\mathbb{V}(Y_d|\boldsymbol{\eta})}{p(1-p)}}\right\}\right),
\end{aligned}$$

following De Nicolò *et al.* (2021).

The priors for the random effects are discussed in the following: the case of unstructured random effects is faced in Section 2.2.1, spatially structured random effects are described in Section 2.2.2, and temporal random effects in Section 2.2.3.

Unstructured Random Effects

The basic assumption on the random effect is $e_d = v_d$, where v_d is an unstructured area-specific random effect accounting for deviations from the synthetic predictor. We propose three different strategies to specify its prior distribution, that can be chosen through the `prior_reff` argument of `fit_sae()`. Firstly, a zero-mean normal prior with scale σ_v is considered (`"normal"` option, default), putting a half-normal prior for σ_v, in line with Gelman (2006):

$$\begin{aligned} & v_d|\sigma_v \overset{ind}{\sim} \mathcal{N}\left(0, \sigma_v^2\right), \; d = 1, \ldots, D; \\ & \sigma_v \sim \text{Half-}\mathcal{N}(0, 2.5^2). \end{aligned}$$

The choice of such half-normal prior is usually weakly informative if compared to the scale of the random effects.

When covariates have poor explanatory power, in some domains, it is possible to observe large deviations of the predicted value from the observed one, requiring more flexible handling of random effect through a robust prior. Among those proposed in the literature, we implement the one introduced by Figueroa-Zúñiga, Arellano-Valle, and Ferrari (2013), and previously considered in the small area framework by Fabrizi and Trivisano (2016). It consists of a Student's t prior with exponential hyperprior for degrees of freedom ν and half-normal hyperprior for the scale σ_v (`"t"` option):

$$\begin{aligned} & v_d|\nu, \sigma_v \overset{ind}{\sim} t\left(\nu, 0, \sigma_v\right), \; d = 1, \ldots, D; \\ & \nu \sim \text{Exponential}(0.1); \\ & \sigma_v \sim \text{Half-}\mathcal{N}(0, 2.5^2). \end{aligned}$$

The notation $t\left(\nu, 0, \sigma_v\right)$ indicates a Student's t distribution with ν degrees of freedom, location parameter equal to 0, and scale σ_v.

In other cases, the variability of the small area parameters may not require the inclusion of a random effect term in presence of very informative covariates (Datta, Hall, and Mandal 2011b). Therefore, the variance gamma shrinkage prior introduced by Brown and Griffin (2010) and implemented in a small area application by Fabrizi, Ferrante, and Trivisano (2018) is included as a prior choice for v_d (`"VG"` option). This option enables for shrinking to 0 the random effects related to a subset of the areas by mimicking the behaviour of a spike-and-slab prior. Following Fabrizi *et al.* (2018), we propose a general hyperparameters choice that induces a prior variance of the random effects equal to 0.5:

$$\begin{aligned} v_d|\psi_d, \lambda & \overset{ind}{\sim} \mathcal{N}\left(0, \frac{\psi_d}{\lambda}\right), \; d = 1, \ldots, D; \\ \psi_d & \overset{ind}{\sim} \text{Gamma}(0.5, 1), \; d = 1, \ldots, D; \\ \lambda & \sim \text{Gamma}(2, 1). \end{aligned}$$

It can be noted that the independent ψ_d are local scales, whereas λ is a global precision hyperparameter.

Spatially Structured Random Effects

Setting the argument `spatial_error` equal to `TRUE`, we let the user add a spatially structured effect s_d to the linear predictor, leading to the formulation $e_d = v_d + s_d$. For the vector

$\boldsymbol{s} = (s_1, \ldots, s_D)$, we assume an intrinsic conditional autoregressive (ICAR) prior (Besag, York, and Mollié 1991), i.e., an improper prior whose density is proportional to:

$$f\left(\boldsymbol{s}|\sigma_s\right) \propto \exp\left\{-\frac{1}{2\sigma_s^2}\boldsymbol{s}^T\tilde{\mathbf{K}}_s^{-}\boldsymbol{s}\right\},$$

where $\tilde{\mathbf{K}}^{-}$ is the generalized inverse of a singular precision matrix. To describe its structure, we first define $\mathbf{K} = \mathbf{D} - \mathbf{W}$, where $\mathbf{D}$ is a diagonal matrix containing the number of connections for each area and $\mathbf{W}$ is the adjacency matrix (the generic entry $[w]_{ij}$ is 1 if area i and j are adjacent and 0 otherwise). Following Freni-Sterrantino, Ventrucci, and Rue (2018), the actual precision matrix $\tilde{\mathbf{K}}$ is obtained with a scaling procedure aimed at reducing the impact of the structure on the prior variability, keeping into consideration the possible presence of $G \geq 1$ disconnected graphs in the model (e.g., islands). Note that $G-1$ dummy variables are added to the linear predictor in order to obtain island-specific means, placing a sum-to-zero constraint on the random effects related to the same island. Islands defined by singleton areas are also allowed, even if they do not constitute a graph counted in G. Lastly, a half-normal prior is fixed for the hyperparameter σ_s. For further details on the implementation of ICAR priors in Stan, see Morris, Wheeler-Martin, Simpson, Mooney, Gelman, and DiMaggio (2019).

To include a spatially structured random effect, an object of class '`SpatialPolygonsDataFrame`' (from the **sp** package, Bivand, Pebesma, and Gomez-Rubio 2013) is required as input of the `spatial_df` argument, carefully checking that the order of its rows and the order of the `data` input are coherent.

Temporally Structured Random Effects

If multiple observations of the target indicator are available for different time periods, a suitable model can be specified, in order to borrow strength from time repetitions. In this framework, a second subscript must be added in the notation: Y_{dt} indicates the direct estimator for area d at time $t = 1, \ldots, T$, whereas e_{dt} is the random effect component in the linear predictor. The user can choose to add a temporal random effect u_{dt} to the unstructured one ($e_{dt} = v_d + u_{dt}$) setting `spatial_error = TRUE`. If both temporal and spatial random effects are declared in `fit_sae()`, then a spatio-temporal model is fitted, removing the unstructured random effect ($e_{dt} = s_d + u_{dt}$).

As prior for the sequence of random effects $\{u_{dt}\}_t$, we specify a random walk prior of order 1, assuming independence among the areas (Rao and Molina 2015). It represents a flexible prior that can be defined recursively as:

$$u_{dt}|u_{d,t-1}, \sigma_u \sim \mathcal{N}\left(u_{d,t-1}, \sigma_u^2\right), \ t = 2, \ldots, T;$$

implicitly assuming a uniform improper prior on u_{d1}. Sum-to-zero constraints are placed for each area-specific time sequences, to guarantee the identifiability of all the parameters in the linear predictor. Even then, a half-normal prior is fixed for the hyperparameter σ_u and the contribution of the correlation structure to the prior variability is mitigated by adopting a scaling procedure (Riebler, Sørbye, Simpson, and Rue 2016).

2.3. Data Pre-Processing

Before stepping into estimation, we propose an elective function for refining raw variance estimates, which are inputs of our models. It can be useful both for reducing their sampling error and estimating the effective sample size parameter $\phi_d + 1$. The `smoothing()` function implements three methods, all yielding refined estimates of either variance or $\phi_d + 1$, to account for indicators with different variance functions. The output estimates are ready to be used as known parameters in an area-level model, and they need to be added to the analysed '`data.frame`' object.

Let us consider that, under simple random sampling, a general variance function has the following structure:

$$\mathbb{V}_{\text{srs}}\left[Y_d\right] = \frac{f(\theta_d)}{n_d},$$

where n_d is the sample size. Note that if the target quantity is a proportion, then $f(\theta_d) = \theta_d(1-\theta_d)$. However, when dealing with complex survey designs, the selection process invariably introduces a correlation structure in the data. In this way, the information actually available may be lower than the one provided by a sample of the same size under simple random sampling. In order to formalize this concept, we need to introduce the effective sample size $\tilde{n}_d$. It can be estimated as $\tilde{n}_d = n_d/\text{deff}$, where deff is the design effect, defined as the ratio between the complex design-based variance $\mathbb{V}_{\text{cd}}\left[Y_d\right]$ and $\mathbb{V}_{\text{srs}}\left[Y_d\right]$. Clearly, under simple random sampling $\tilde{n}_d$ equals n_d.

All the three implemented methods enable the estimation of the effective sample sizes, whereas `"ols"` and `"gls"` also perform a variance smoothing procedure. The argument `method` allows to choose among:

- `"kish"`, implementing an area-specific design effect estimation proposed by Kish (1992). It employs solely the design weights and requires an additional data frame as input of the `survey_data` argument, whose structure is specified in Section 4.3. The specific design effect is estimated as:

  $$\text{deff}_d = n_d \cdot \sum_{h \in d} \frac{W_{dh}^2}{n_{dh}}$$

  where h refers to a generic sampling unit in area d (e.g., the household). Indicating with subscript c the generic individual in sampling unit h, we define $W_{dh} = \hat{N}_{dh}/\hat{N}_d$, $\hat{N}_{dh} = \sum_{c \in h} w_{dhc}$, $\hat{N}_d = \sum_{h \in d} w_{dh}$ and $n_d = \sum_{h \in d} n_{dh}$. We denote with w_{dh} and n_{dh} the design weight and the sample size of unit h in area d, respectively; while w_{dhc} is the individual design weight. Thus, the design-based variance can be defined as

  $$\mathbb{V}_{cd}\left[Y_d\right] = \frac{f(\theta_d)}{n_d}\text{deff}_d, \tag{1}$$

  while $\phi_d + 1 = \tilde{n}_d = n_d/\text{deff}_d$. This method has already been used in small area context by Wieczorek and Hawala (2011) and Liu *et al.* (2007). Kalton, Brick, and Le (2005) found this approximation accurate for proportions ranging between 0.2 and 0.8.

- `"ols"`, implementing a variance smoothing model using a Generalized Variance Function approach, as in Fabrizi *et al.* (2011) and Fabrizi and Trivisano (2016). Considering the design-based variance as

  $$\mathbb{V}_{cd}\left[Y_d\right] = \frac{f(\theta_d)}{n_d}\text{deff},$$

the smoothing procedure is based on the assumption that the design effect does not vary across areas. By assuming $\hat{\mathbb{V}}_{raw}[Y_d]$ as a raw estimator of complex survey variance with large error, let us specify the following smoothing equation:

$$\frac{f(Y_d)}{\hat{\mathbb{V}}_{raw}[Y_d]} = \psi n_d + \epsilon_d,$$

where $\psi = 1/\text{deff}$ and ϵ_d are zero-mean and homoscedastic residuals. The model is estimated using ordinary least squares via the `gls()` function from **nlme** package (Pinheiro, Bates, DebRoy, Sarkar, and R Core Team 2021), providing the smoothed dispersion parameters defined as $\hat{\phi}_d + 1 = \hat{\psi} n_d$ and the refined estimate as $\hat{\mathbb{V}}[Y_d] = f(y_d)/(\hat{\phi}_d + 1)$.

- `"gls"`, extending the `"ols"` method in case of heteroskedasticity of the error component ϵ_d of equation (1). The default method assumes only heteroskedastic error with a power variance function on absolute fitted values (see Pinheiro *et al.* 2021, for further details).

2.4. Posterior Inference

We are interested in making posterior inference on θ_d. Since we are not dealing with conjugate models, not even conditionally, the posterior inference is carried out through MCMC draws. As a point estimate, the optimal Bayes estimator of θ_d under quadratic loss is considered, i.e., the posterior mean. We indicate it with the notation:

$$\hat{\theta}_d^{HB} = \mathbb{E}[\theta_d|\boldsymbol{y}] \quad d = 1, \dots D. \tag{2}$$

The point estimates can be complemented with uncertain measures like the posterior standard deviation and credible intervals, determined by the quantiles of the posterior distribution. The generic method `summary()` applied on as `S3` object of class '`fitsae`' produces by default point estimates (posterior mean and median) and credible intervals (at 95% and 50% levels) for predictors, basic model parameters, and random effects.

Out-of-Sample Treatment

The package provides an automatic out-of-sample prediction. This feature is available for all considered `likelihood`, except for Flexible Beta, since in this specific case, θ_d depends on its sampling variance, which is not available in case of out-of-samples.

Recalling that θ_d^{oos}, $d = \tilde{D}, \dots, D$ denotes the out-of-sample target quantity, their predictors are reported in Table 1. Note that they depend on e_d: when spatial and temporal dependencies are defined, s_d and u_{dt} gain information from the assumed correlation structure, whereas v_d is always drawn from a zero-mean distribution, contributing only to the posterior variability of θ_d^{oos}. Exploiting the MCMC estimation framework, it is possible to obtain a sample from the posterior of θ_d^{oos} by combining the samples drawn from the posterior of the involved parameters. Eventually, the point predictor defined in (2) holds also for out-of-sample observations, together with the other posterior summaries.

Diagnostics and Goodness-of-fit Tools

The method `summary()` returns, in addition, goodness-of-fit and model validation diagnostics, as well as SAE-specific diagnostics. In the following, we provide a brief theoretical overview of such measures.

One of the main advantages of estimating models within the Bayesian framework is the plethora of tools that allow investigating model performances. Among the most relevant ones, we can find those relying on the posterior predictive distribution, that we denote with $Y_d^\bullet|\boldsymbol{y},\ d = 1, \ldots, D$. Area-specific Bayesian p values (BP_d) under the following discrepancy measure (You and Rao 2002; Fabrizi *et al.* 2011) are computed:

$$BP_d = \mathbb{P}\left[Y_d^\bullet > y_d|\boldsymbol{y}\right], \quad d = 1, \ldots, D. \tag{3}$$

In absence of systematic deviations, the expected Bayesian p value is 0.5, whereas values near 0 or 1 highlight issues of over-estimation and under-estimation, respectively.

Information criteria are widely used in Bayesian inference to compare models with different specifications, e.g., diverse distributional assumptions, random effects structures, or covariates. Following Vehtari, Gelman, and Gabry (2017), we consider the approximate leave-one-out cross-validation information criterion (LOOIC) computed using Pareto-smoothed importance sampling. It can be retrieved through the **loo** package and is provided together with the approximate standard errors for estimated predictive errors.

Stepping into SAE-specific diagnostics, the standard deviation reduction (SDR_d) indicator is commonly used to assess the decrease of uncertainty associated with the employment of a small area model. It is obtained evaluating

$$SDR_d = 1 - \sqrt{\frac{\mathbb{V}\left[\theta_d|\boldsymbol{y}\right]}{\mathbb{E}[\mathbb{V}\left[Y_d|\boldsymbol{\eta}\right]|\boldsymbol{y}]}}, \quad d = 1, \ldots, D, \tag{4}$$

where the denominator is defined in this way when `type_disp = "neff"`, taking into account the fact that $\mathbb{V}\left[Y_d|\boldsymbol{\eta}\right]$ has a posterior distribution to be summarized. Conversely, if `type_disp = "var"`, the denominator is replaced by $\hat{\mathbb{V}}\left[Y_d\right]$. This diagnostic has to be considered with caution when performing model selection since it does not account for the design bias of different model-based estimators, which could be relevant even when the model is correct.

Lastly, the Shrinking Bound Rate (SBR) is computed:

$$SBR = \frac{1}{\tilde{D}} \sum_{d=1}^{\tilde{D}} \mathbb{1}\{\hat{\theta}_d^{HB} \in (p_d^*, Y_d)\}, \tag{5}$$

where $p_d^* = \exp(\mathbf{x}_d^T\boldsymbol{\beta})/\left[1 + \exp(\mathbf{x}_d^T\boldsymbol{\beta})\right]$ is the synthetic estimate of θ_d. In fact, in the standard Fay-Herriot model, the shrinking process is clearly identified by the shape of the best linear unbiased predictor, for known values of β and σ_v^2 such as,

$$\gamma_d Y_d + (1 - \gamma_d)p_d^* \quad \text{with} \quad \gamma_d = \frac{\sigma_v^2}{\sigma_v^2 + \mathbb{V}[Y_d|\boldsymbol{\eta}]}.$$

Beta regression models do not provide a closed form predictor, since the conditional distribution of θ_d, $\forall d = 1, \ldots, D$ does not belong to a standard family. Janicki (2020) shows that, in a Beta regression model with standard diffuse priors, $\hat{\theta}_d^{HB}$ converges to the direct estimate as $\mathbb{V}(Y_d|\eta) \to 0$ and the synthetic estimates as $\sigma_v^2 \to 0$. The first property has also been proved by Fabrizi *et al.* (2020). However, $\hat{\theta}_d^{HB}$ is not bounded by its convergence limits, conjecturing $Y_d < \hat{\theta}_d^{HB} < p_d^*$ will hold only for $\mathbb{V}(Y_d|\boldsymbol{\eta})$ sufficiently small (Janicki 2020). Thus, checking whether model estimates fit inside the bound, could yield important insights into the shrinking process and estimators consistency.

Benchmarking Procedure

The `benchmark()` function gives the chance to perform a benchmarking procedure on model-based estimates. The need for benchmarking arises since model-based estimates may widely differ from direct estimates and, consequently, model estimates aggregates may widely differ from corresponding direct estimates. However, latter quantities refer to a larger geographical area or a larger socio-demographic group whose target domains are a subset of, and, therefore. are considered to be reliable. This feature may introduce drawbacks in many situations (e.g., when small area estimates are used to allocate funding), and exact benchmarking is required to avoid surpluses or shortfalls (Zhang and Bryant 2020). When adopting a benchmarking approach, model-based estimates are constrained to direct estimates of supra-domain sets.

Existing methods generally address the benchmark issue as a constraint to be imposed. The difference in between various methods is about the way such constraints are interpreted and incorporated in the estimation. Some methods estimate the small area models and then modify the resulting point estimators to satisfy the benchmarking constraints as a two-step procedure (Datta, Ghosh, Steorts, and Maples 2011a). Other methods treat benchmarks as vincula on the underlying small area parameters or on their point estimators which are directly incorporated into the probabilistic structure of the small area model, either in a revised likelihood or in the prior distributions (Pfeffermann, Sikov, and Tiller 2014; Ranalli, Montanari, and Vicarelli 2018; Zhang and Bryant 2020). By considering only Bayesian benchmarking methods, when models yield a full posterior distribution for all unknown quantities after benchmarking, they end up to be categorized as fully Bayesian. While, methods such as the one we adopted, which derive posterior distributions without benchmarking and separately benchmark point estimates, do not fall within this definition. An up-to-date review can be found in Zhang and Bryant (2020).

In our non-fully Bayesian approach, widely explained by Datta *et al.* (2011a), point estimates from a Bayesian model, estimated via the `fit_sae()` function, are adjusted to obtain a new set of estimates that satisfies the constraints. Benchmarking could solely target the point estimators (single benchmarking) or, alternatively, also the ensemble variability (double benchmarking). Furthermore, an estimate of the overall posterior risk is provided, aggregated for all areas. This value is only yielded when in-sample areas are treated and a single benchmarking is performed.

The considered benchmarking procedures require the definition of a set of area-specific weights, which in the case of proportions are defined as $w_d = N_d / \sum_{j=1}^{D} N_j$, where N_d is the population size for area d. The benchmark is indicated with B, and it could be the reliable direct estimate referring to a larger area or a prespecified value from another data source or, eventually, $B = \sum_{d=1}^{D} w_d Y_d$, if the aim is to perform internal benchmarking. The function allows performing three different benchmarking methods, according to the argument `method`.

- The `"ratio"` method provides benchmarked estimates $\hat{\theta}_d^{BM}$ that minimize the posterior expectation of the weighted squared error loss. The benchmarked estimates are

$$\hat{\theta}_d^{BM} = \hat{\theta}_d^{HB} + \frac{B - \sum_d w_d \hat{\theta}_d^{HB}}{s} r_d, \tag{6}$$

where $r_d = \hat{\theta}_d^{HB}$, and $s = \sum_d w_d \hat{\theta}_d^{HB}$. Datta *et al.* (2011a) provide also the posterior risk for the whole set of benchmarked estimates:

$$\sum_d \frac{w_d}{r_d} \left[\mathbb{V}[\theta_d|\boldsymbol{y}] + \frac{\left(B - \sum_d w_d Y_d\right)^2}{s^2} r_d^2 \right]. \tag{7}$$

- The "raking" method provides the benchmarked estimate in (6) and the posterior risk (7) with $r_d = 1$ and $s = 1$.
- The "double" method extends this procedure accounting for a further benchmark on the weighted ensemble variability. The simultaneous constraints are $\sum_d w_d \hat{\theta}_d^{BM} = B$ and $\sum_d w_d (\hat{\theta}_d^{BM} - B)^2 = H$, where H is a prespecified value of the estimators variability taken from other sources. The expression of the resulting benchmarked estimate is:

$$\hat{\theta}_d^{BM} = B + \sqrt{\frac{H}{\sum_d w_d \left(\hat{\theta}_d^{HB} - \sum_d w_d \hat{\theta}_d^{HB}\right)^2}} \left(\hat{\theta}_d^{HB} - \sum_d w_d \hat{\theta}_d^{HB}\right).$$

Note that the benchmarking procedure can be performed in case of temporal or spatio-temporal models by specifying multiple time-period benchmarks.

3. Datasets

In SAE field, data typically come from multiple sources. Direct estimators and their sampling variances typically result from survey data, aggregated at area-level, while covariates come from census and/or administrative/register sources. As a consequence, explanatory variables, aggregated at area level, are required to be defined at population level i.e., without error, and potentially correlated with the target variable. In order to outline the workflow of **tipsae** package, its functions are illustrated in Section 4 and applied to an example dataset, released within the package. The whole dataset is named `emilia` and consists of a panel on poverty mapping concerning 38 health districts within the Emilia-Romagna region, located in North-East of Italy, with annual observations recorded from 2014 to 2018. We built it starting from model-based estimates and related CV freely available on Emilia-Romagna region website [1]. Since it is used for illustrative purposes only, such estimates are assumed to be unreliable direct estimates, requiring a SAE procedure.

We considered the Head-Count Ratio estimates as direct (`$hcr`) and its associated variance as sampling variance (`$vars`). A fake standardized covariate `$x` has been generated. We also provide area sample sizes (`$n`), population sizes (`$pop`), province identification (`$prov`), years (`$year`) and health district name (`$id`). The `emilia` dataset can be loaded as follows.

```
R> library("tipsae")
R> data("emilia")
R> head(emilia)
```

```
                   id prov year    hcr         vars  n       x    pop
1 CASALECCHIO DI RENO   BO 2014 0.0404 9.090478e-05 42 -0.2624 108261
```

[1] https://statistica.regione.emilia-romagna.it/documentazione/pubblicazioni/documenti_catalogati/stima-poverta-2009-2018-distretti-sociosanitari-province-emilia-romagna

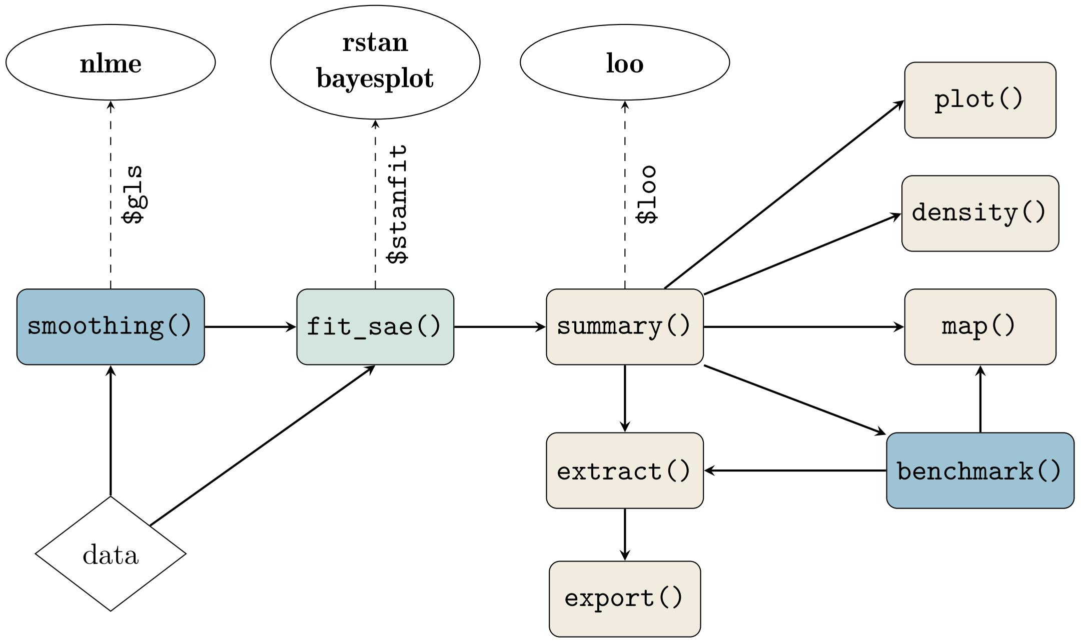

Figure 1: Flowchart that describe the structure of the tools implemented in **tipsae** package.

```
2   CITTA' DI BOLOGNA   BO 2014 0.0825 6.404001e-05 285 -0.0008 371151
3               IMOLA   BO 2014 0.1033 3.120275e-04  49 -0.0522 130007
4         PIANURA EST   BO 2014 0.0633 1.025764e-04 190 -0.4007 154213
5       PIANURA OVEST   BO 2014 0.0625 1.562500e-04  10 -0.2277  80951
6      PORRETTA TERME   BO 2014 0.1276 6.643609e-04  26 -0.4434  56428
```

A cross-sectional subset concerning a single year (2016) is taken from `emilia`, for non-temporal models illustration purpose: it is named `emilia_cs` and can be loaded as follows.

```
R> data("emilia_cs")
```

4. Workflow

In this section, a typical flow of a SAE analysis is outlined with step-by-step instructions, showing the potentials of **tipsae** tools. As illustrated with a flowchart in Figure 1, the package is structured into three parts that relate to: model building and fitting (●, Section 4.1), diagnostics and results displaying (●, Section 4.2), and complementary tools for SAE analysis (●, Section 4.3). Figure 1 displays also the possible connections with external functions, drawn with dashed arrows, useful to further exploit the produced objects.

4.1. Model Building and Fitting

The first step of the workflow represents the core of our package, concerning the estimation of models with the diverse extensions and parametrizations defined in Section 2. The sole function `fit_sae()` allows users to construct personalized models and fit them using Stan routines, called up through the `sampling()` function of **rstan** package. It also allows customized parallel computing when the model runs on multiple chains. A simple parallelization can be set out using the following command, which imposes a number of R processes equal to the number of CPU cores.

```
R> options(mc.cores = parallel::detectCores())
```

The function `setDefaultClusterOptions()` from **parallel** package can be used to change the default options for parallelization. For further details, see **rstan** guidelines.

A complete list of the input arguments of the `fit_sae()` function is specified in Table 2, and a first example of model fitting on the `emilia_cs` dataset is provided. Firstly, we consider model default options: a Beta likelihood and a Gaussian prior for unstructured random effects. Since `emilia_cs` dataset contains the sampling variance as a measure of dispersion, `disp_direct` must be fixed equal to `"var"`, setting a mean-variance parametrization. Moreover, argument `domains_size` has to be specified for having visual design consistency diagnostics in the subsequent plotting function.

The estimation can be done in practice by running the `fit_sae()` function as follows. For the sake of reproducibility, we set `seed=0`.

```
R> fit_beta <- fit_sae(formula_fixed = hcr ~ x,
+                      data = emilia_cs,
+                      domains = "id",
+                      type_disp = "var",
+                      disp_direct = "vars",
+                      domain_size = "n",
+                      seed = 0)
```

Note that further arguments, concerning `sampling()` function options, can be additionally specified. In particular, we mention those related to HMC algorithm setting such as `iter`, allowing to set the number of iterations per chain (default equal to 2000), `warmup`, determining the number of iterations per chain to be discarded as warm-up period (default `iter/2`), `chains`, fixing the number of independent Markov chains (default 4).

Different models can be estimated relying on diverse assumptions, being subsequently compared with each other. For example, we assume a Flexible Beta likelihood and a variance gamma shrinking prior for the unstructured random effect, in order to propose a more flexible model for the data. Given the increasing complexity of model assumptions, more HMC iterations are required, together with a higher proposal acceptance probability (`adapt_delta`).

```
R> fit_FB <- fit_sae(formula_fixed = hcr ~ x,
+                    data = emilia_cs,
+                    domains = "id",
+                    type_disp = "var",
+                    disp_direct = "vars",
+                    domain_size = "n",
+                    likelihood = "flexbeta",
+                    prior_reff = "VG",
+                    adapt_delta = 0.99,
```

Argument	Short description	Default
`formula_fixed`	`formula` object specifying the fixed regression part.	-
`data`	`data.frame` containing all relevant quantities.	-
`domains`	`data` column name displaying domains names. If `NULL` (default) the domains are denoted with a progressive number.	`NULL`
`type_disp`	Parametrization of the dispersion parameter. The choices are variance (`"var"`) or $\phi_d + 1$ (`"neff"`) parameter.	`"neff"`
`disp_direct`	`data` column name displaying given values of sampling dispersion for each domain. In out-of-sample areas, dispersion must be `NA`.	-
`domain_size`	`data` column name indicating domain sizes (optional). In out-of-sample areas, sizes must be `NA`.	`NULL`
`likelihood`	Sampling likelihood to be used. The choices are `"beta"`, `"flexbeta"`, `"Infbeta0"`, `"Infbeta1"` and `"Infbeta01"`.	`"beta"`
`prior_reff`	Prior distribution of the unstructured random effect. The choices are: `"normal"`, `"t"`, `"VG"`.	`"normal"`
`spatial_error`	Logical indicating whether to include a spatially structured random effect.	`FALSE`
`spatial_df`	Object of class `SpatialPolygonsDataFrame` with the shapefile of the studied region. Required if `spatial_error = TRUE`.	`NULL`
`temporal_error`	Logical indicating whether to include a temporally structured random effect.	`FALSE`
`temporal_variable`	`data` column name indicating temporal variable. Required if `temporal_error = TRUE`.	`NULL`
`adapt_delta`	HMC option: target average proposal acceptance probability. See `Stan` documentation.	`0.95`
`max_treedepth`	HMC option: maximum allowed tree depth for each transition. See `Stan` documentation.	`10`
`init`	HMC option: initial values setting. The choices are: `"0"`, `"random"`, or manual setup via list or function. See `Stan` documentation.	`"0"`
`...`	Further inputs for the `sampling` function.	

Table 2: Input arguments for function `fit_sae()`.

Position	Name	Short description
1	`model_settings`	List summarizing all the assumptions of the model: sampling likelihood, presence of intercept, dispersion parametrization, random effects priors and possible structures.
2	`data_obj`	List containing input objects including in-sample and out-of-sample relevant quantities.
3	`stanfit`	'`stanfit`' object, outcome of `sampling()` function containing full posterior draws. For details, see **rstan** documentation.
4	`pars_interest`	Vector containing the names of parameters whose posterior samples are stored.
5	`call`	Image of the function call that produced the '`fitsae`' object.

Table 3: Components of '`fitsae`' objects.

```
+                    iter = 8000,
+                    seed = 0)

Warnings:
1: There were 10 divergent transitions after warmup. See
http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
to find out why this is a problem and how to eliminate them.
2: Examine the pairs() plot to diagnose sampling problems
```

The `fit_sae()` function returns an S3 object of class '`fitsae`', being a list of relevant items that are listed in Table 3. The core element is the `$stanfit` object, incorporating posterior draws and raw MCMC information to be extracted, whereas the remaining elements only provide details about the function call and model settings.

4.2. Diagnostics and Results Displaying

After the MCMC drawing, a careful check on algorithm convergence is required, in order to validate posterior results. With this aim, our suggestion is to exploit the plethora of diagnostic methods implemented for '`stanfit`' objects within the **bayesplot** package. For example, the following code generates the trace-plots related to the `fit_beta` model, as in Figure 2, useful to visually inspect the convergence of the chains to a unique stationary distribution.

```
R> library("bayesplot")
R> post_beta <- as.array(fit_beta\$stanfit, pars = c("beta0", "beta"))
R> mcmc_trace(x = post_beta)
```

The '`stanfit`' object also provides useful visual diagnostics to deepen the warnings printed by Stan, such as those about the maximum tree depth and divergent transitions after the warm-up period.

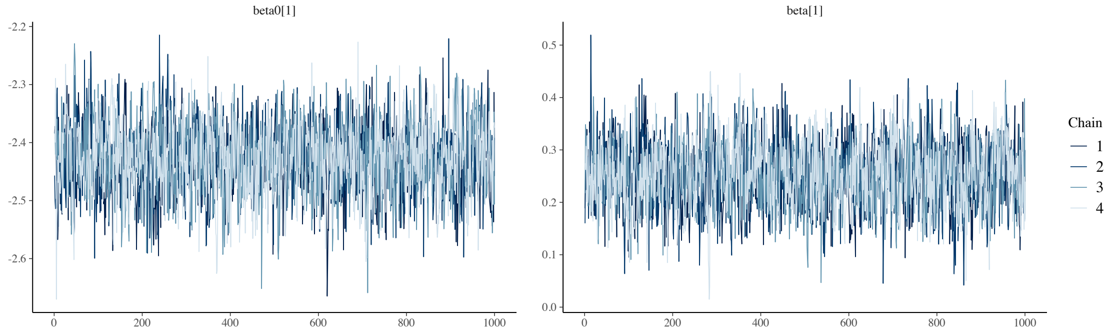

Figure 2: Traceplots of the parameters β_0 and β_1 of the Beta regression model.

However, small area diagnostics are required at this stage, in order to check whether results meet specific properties which turn out to be desirable in such context. Peculiar diagnostic measures can be obtained through `summary()` method applied on ‘`fitsae`’ objects. Besides the printed output, the method produces an object of class ‘`summary_fitsae`’ which contains relevant information for posterior inference. Argument `probs` allow specifying the quantiles of interest to be visualized as posterior summary measures. The logical argument `compute_loo` allows deciding whether LOOIC should be computed or not.

```
R> summ_beta <- summary(fit_beta)

Warnings:
Some Pareto k diagnostic values are too high.
See help('pareto-k-diagnostic') for details.

R> summ_beta

Summary for the SAE model call:
 fit_sae(formula_fixed = hcr ~ x, domains = "id", disp_direct = "vars",
    type_disp = "var", domain_size = "n", data = emilia_cs, seed = 0)

----- S.D. of the random effects: posterior summaries -----

         mean    sd  2.5%  25%   50%   75% 97.5%
sigma_v 0.267 0.055 0.168 0.23 0.263 0.299 0.388

----- Fixed effects coefficients: posterior summaries -----

              mean    sd   2.5%    25%    50%    75%  97.5%
(Intercept) -2.428 0.060 -2.550 -2.467 -2.428 -2.387 -2.309
x            0.253 0.061  0.135  0.213  0.253  0.293  0.372

--------------- Model diagnostics summaries ---------------

                   Min. 1st Qu. Median  Mean 3rd Qu.  Max.
```

```
Residuals        -0.016  -0.004  0.002 0.004   0.011 0.032
S.D. Reduction   -0.100   0.197  0.254 0.240   0.318 0.390
Bayesian p-value  0.172   0.339  0.459 0.461   0.555 0.785

Shrinkage Bound Rate: 100 %

LOO Information Criterion:
          Estimate    SE
 elpd_loo   87.719 3.629
 p_loo      17.787 2.546
 looic    -175.439 7.259
```

If printed, the produced summary displays:

- Posterior summaries about the fixed effect coefficients and the scale parameters related to unstructured and possible structured random effects.
- Model diagnostics summaries of (a) model residuals; (b) standard deviation reductions computed using (4); (c) Bayesian p values obtained approximating the (3) with the MCMC samples.
- Shrinking Bound Rate, defined in (5).
- LOO information criteria and related diagnostics from the **loo** package.

What Can Accidentally Be Done with a '`summary_fitsae`' Object

The '`summary_fitsae`' object contains additional valuable elements for further exploration. For instance, the `$loo` element consists of the whole object of class '`loo`' which may be employed in external functions, such as the ones provided by **loo** package e.g., for model comparison, as follows.

```
R> summ_FB <- summary(fit_FB)

Warnings:
Some Pareto k diagnostic values are too high.
See help('pareto-k-diagnostic') for details.

R> library("loo")
R> loo_compare(list("beta" = summ_beta$loo, "flexbeta" = summ_FB$loo))

         elpd_diff se_diff
flexbeta  0.0       0.0
beta     -6.9       2.9
```

The output shows that the Flexible Beta model has a significantly higher expected log pointwise predictive density for a new dataset, gaining in prediction power with respect to the default model.

Another element that can be employed in external functions to assess model goodness of fit is `$y_rep`, an array with values generated from the posterior predictive distribution, enabling the implementation of posterior predictive checks through the **bayesplot** package. The observed data, required for the checks, can be extracted through `$direct_est` element. The following code allows comparing the empirical densities of generated samples under the considered models, reported in Figure 3.

```
R> library("ggplot2")
R> ppc_dens_overlay(y = summ_beta$direct_est, yrep = summ_beta$y_rep[1:100,]) +
+    ggtitle("Beta likelihood")
R> ppc_dens_overlay(y = summ_FB$direct_est, yrep = summ_FB$y_rep[1:100,]) +
+    ggtitle("Flexible Beta likelihood")
```

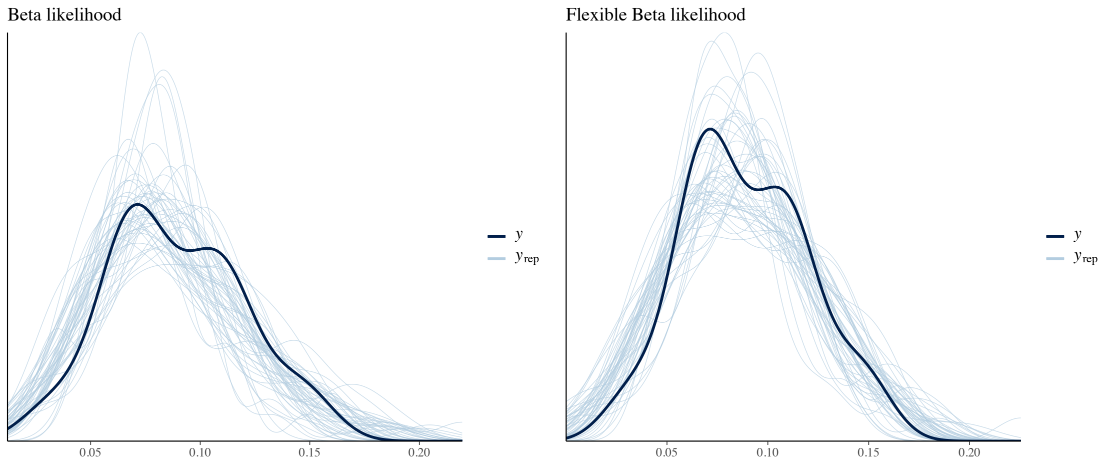

Figure 3: The empirical densities from posterior predictive samples (y_{rep}) versus the observed data one (y).

Lastly, all the posterior summaries related to random effects are stored in the `$raneff` element, being a list of '`data.frame`' objects, one for each type: `$unstructured`, `$temporal`, and `$spatial`. Such outputs may be exploited to produce meaningful plots, e.g., the caterpillar plot of Figure 4, created via the following code.

```
R> ggplot(summ_beta$raneff$unstructured, aes(x = reorder(Domains, mean))) +
+    geom_point(aes(y = mean)) +
+    geom_linerange(aes(ymin = `2.5%`, ymax = `97.5%`)) +
+    geom_hline(yintercept = 0, lty = 2) +
+    ylab("Random effect") + xlab("") +
+    theme_bw(base_size = 12) +
+    theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```

Ad-Hoc Plot Functions

Our package comes equipped with ad-hoc functions for visual diagnostic tools. The `S3` object '`summary_fitsae`' can be used as input for `plot()` and `density()` visual methods as well as for `map()` function.

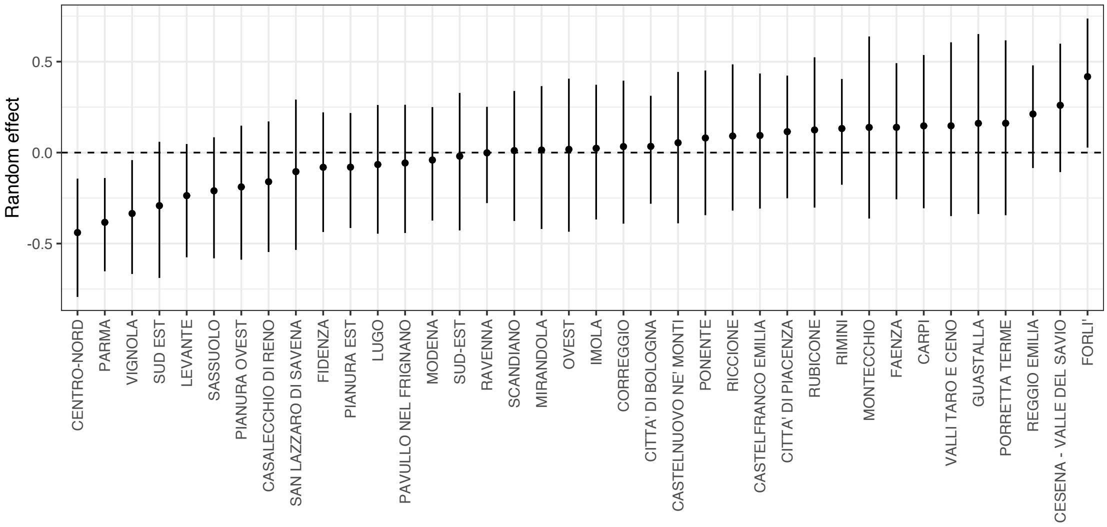

Figure 4: Caterpillar plot of unstructured random effects from Beta regression model.

The generic method `plot()` provides, in a grid (default) or sequence, (a) a scatterplot of direct estimates versus model-based estimates, visually capturing the shrinking process, (b) a Bayesian p values histogram, (c) a boxplot of standard deviation reduction values, and, if areas sample sizes are provided as input in `fit_sae()`, (d) a scatterplot of model residuals versus sample sizes, in order to check for design-consistency i.e., as long as sizes increase residuals should converge to zero. The following code line produces Figure 5.

```
R> plot(summ_beta)
```

The method `density()` provides, in a grid (default) or sequence, the density plot of direct estimates versus HB model estimates and the density plot of standardized posterior means of the random effects versus standard normal, in order to check for Gaussian assumption. Figure 6 is produced as the output of the following command.

```
R> density(summ_beta)
```

Lastly, the `map()` function enables the investigation of the analysed phenomenon by accounting for its geographical dimension, if it exists. More in detail, a '`SpatialPolygonsDataFrame`' object from the **sp** package should be provided as input in `spatial_df` argument. The `spatial_id_domains` argument must receive as input the name of `spatial_df` variable containing area denominations, in order to correctly match the areas. If such names match the ones provided through the original dataset, no extra arguments are required. Otherwise, the `match_names` argument should receive an encoding two-columns '`data.frame`': the first with the original data coding (`domains`) and the second one with corresponding `spatial_df` object labels. The feature to be displayed on the map can be defined in `quantity` argument, choosing among HB model estimates `HB_est`, direct estimates `Direct_est`, posterior standard deviations `SD`, and benchmarked estimates `Bench_est` when a '`benchmark_fitsae`' class object is given as input (see Section 4.3). The following code loads the Emilia-Romagna

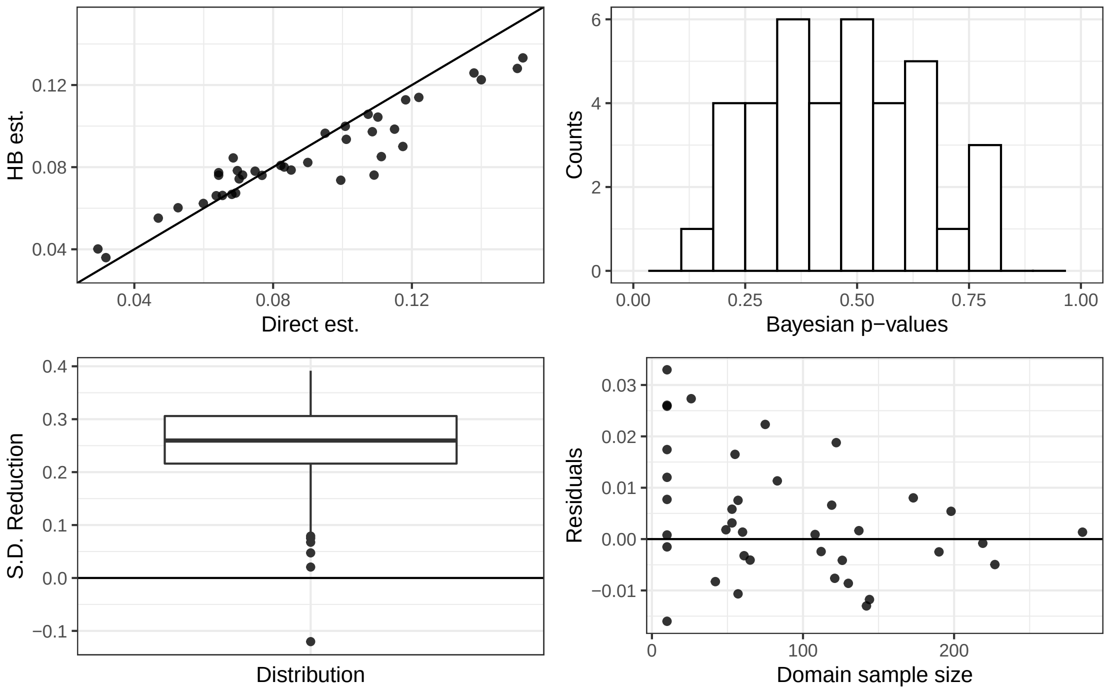

Figure 5: `plot()` method outcome on object of class '`summary_fitsae`'.

health districts shapefile and produces the maps in Figure 7, with model-based estimates and their posterior standard deviations.

```
R> data("emilia_shp")
R> map(x = summ_beta,
+      spatial_df = emilia_shp,
+      spatial_id_domains = "NAME_DISTRICT")
R> map(x = summ_beta,
+      spatial_df = emilia_shp,
+      quantity = "SD",
+      spatial_id_domains = "NAME_DISTRICT")
```

Take-Home Function

Lastly, '`summary_fitsae`' object provides target parameters posterior and model-based estimates, visually accessible through the function `extract()` as follows.

```
R> HB_estimates <- extract(summ_beta)
R> head(HB_estimates$in_sample)
```

```
               Domains Direct est.    HB est.          sd       2.5%
1 CASALECCHIO DI RENO      0.0469 0.05511808 0.009143088 0.03696825
2   CITTA' DI BOLOGNA      0.0681 0.06668932 0.008168604 0.05040972
3               IMOLA      0.0692 0.06723489 0.011631916 0.04467289
```

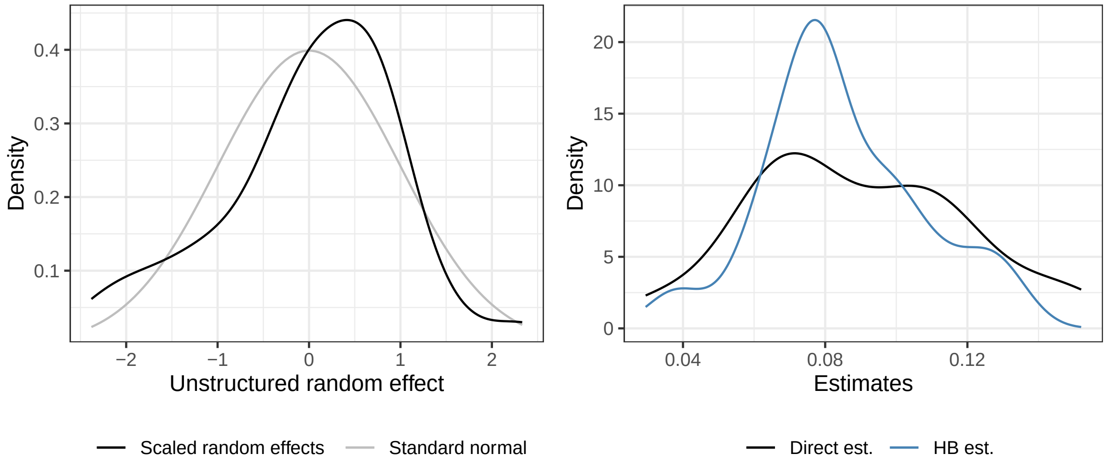

Figure 6: `density()` method visual outcome.

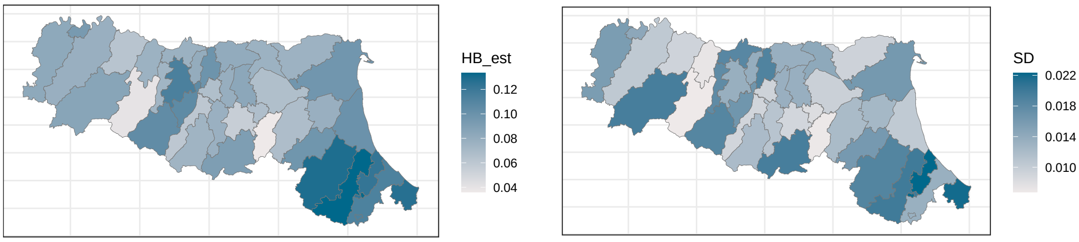

Figure 7: `map()` function visual outcome.

```
4          PIANURA EST      0.0636 0.06621838 0.009554082 0.04787149
5        PIANURA OVEST      0.0685 0.08465454 0.013695722 0.05789687
6       PORRETTA TERME      0.1174 0.09035749 0.019929132 0.05449263
         25%        50%        75%      97.5%
1 0.04897258 0.05504724 0.06121161 0.07330119
2 0.06132649 0.06695005 0.07234576 0.08205602
3 0.05913559 0.06721644 0.07501379 0.09011058
4 0.05944194 0.06623004 0.07286028 0.08465238
5 0.07552132 0.08449351 0.09385837 0.11194273
6 0.07645193 0.08868046 0.10297773 0.13293759
```

The function returns an object of class ‘`estimates_fitsae`’, being a list of two data frames, distinguishing between `$in_sample` and `$out_of_sample` areas, which gathers domains name, direct and HB estimates, as well as posterior summaries of parameters θ_d, $\forall d$.

A function for exporting such results in CSV format is directly accessible, with name `export()`. This function requires an ‘`estimate_fitsae`’ object and a character string naming the output file (argument `file`). It is also possible to indicate whether to export both in and out of sample areas results (default, `type = "all"`), or only in or out of sample areas, (`"in"` or "out", respectively), as follows.

```
R> export(HB_estimates,
+         file = "results.csv",
+         type = "all")
```

Additional arguments of `write.csv()` function from **utils** package can be further indicated.

4.3. Complementary Tools

Complementary tools for small-area analysis provided by the package are the smoothing and benchmarking functions. The `smoothing()` function allows for data pre-processing of sampling variance estimates and retrieving effective sample sizes, as described in Section 2.3. After its usage, output results have to be incorporated in the dataset used as input of the `fit_sae()` function. The `smoothing()` function requires as input the `data` including the direct estimates, whose variable name has to be specified in `direct_estimates` argument, the method to be used among `"ols"`, `"gls"` and `"kish"` (`method`), and the specification of a variance function $f(\theta)$, through `var_function` argument. The default option (`NULL`) for $f(\theta)$ matches the proportion case, being equal to $\theta(1-\theta)$, while for other measures it can widely differ, for instance, the Gini index variance can be approximated to $f(\theta) = \theta^2(1-\theta^2)$ (Fabrizi and Trivisano 2016) and therefore the following object has to be provided in `var_function` argument:

```
R> gini_variance <- function(x){ x^2 * (1 - x^2) }
```

If method `"ols"` or `"gls"` is chosen, the function requires the raw variance estimates (argument `raw_variance`), areas sample sizes (`areas_sample_sizes`), and, possibly, additional covariates (`additional_covariates`), all of them being column names of the '`data.frame`' provided to the `data` argument. On the other hand, method `"kish"` requires the domain names (`area_id`, as column name in `data`) and the specification of an additional dataset (`survey_data`), defined at sampling unit level (e.g., households). Such dataset must include sampling weights (`weights`), unit sizes (`sizes`) and domain names (`survey_area_id`), in order to allow for matching. The output is an object of '`smoothing_fitsae`' class, being a list of vectors including dispersion parameters estimates: both the variance and $\hat{\phi}_d$. If `"ols"` or `"gls"` method has been selected, the list incorporates also an object of class '`gls`' from **nlme** package, ready to be further explored through **nlme** additional tools. The `plot()` method is available for '`smoothing_fitsae`' objects, showing a boxplot of variance estimates, when effective sample sizes are estimated through `"kish"` method, or a scatterplot of both original and smoothed estimates versus sample sizes, when variance smoothing is performed through `"ols"` or `"gls"`.

```
R> smoo <- smoothing(data = emilia_cs,
+                    direct_estimates = "hcr",
+                    area_id = "id",
+                    raw_variance = "vars",
+                    areas_sample_sizes = "n",
+                    var_function = NULL,
+                   method = "ols")
R> smoo

Smoothing procedure for the dispersion parameters

* Adopted method: ols
* Variance function:
function(mu) {
mu * (1 - mu)
}
---------------------------------------------------------------------
Generalized Variance Function regression:

Generalized least squares fit by REML
  Model: as.formula(paste0("y ~ -1", str))
  Data: regdata
       AIC      BIC    logLik
  481.1331 484.3549 -238.5666

Coefficients:
     Value Std.Error  t-value  p-value
n 2.888026 0.1825709 15.81866       0

Standardized residuals:
       Min         Q1        Med         Q3        Max
-1.5003066 -0.3865189  0.4100642  0.7766002  3.1200482

Residual standard error: 127.9598
Degrees of freedom: 38 total; 37 residual
---------------------------------------------------------------------

Summaries of involved quantities

* Smoothed variance estimates:
    Min.  1st Qu.   Median     Mean.   3rd Qu.     Max.
7.71e-05 1.74e-04 3.81e-04 1.01e-03  1.20e-03  4.17e-03


* Differences between smoothed and raw variances:
     Min.   1st Qu.    Median     Mean    3rd Qu.     Max.
-3.33e-03 -5.02e-04 -5.50e-05  -6.28e-04  2.21e-05  2.28e-04

* Smoothed Phi:
   Min. 1st Qu.  Median    Mean   3rd Qu.    Max.
   27.9    85.6   181.0   256.0    372.0    822.0

R> emilia_cs$smoo_phi <- smoo$phi
```

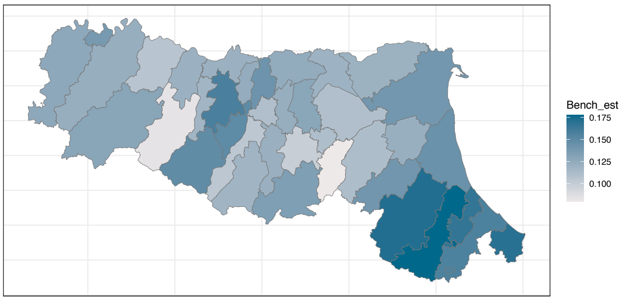

Figure 8: Benchmarked estimates plotted through `map()` function.

```
R> emilia_cs$smoo_vars <- smoo$vars
```

The `benchmark()` function implements benchmarking procedures, described in Section 2.4.3, on model-based estimates provided by indicating a '`summary_fitsae`' object, given a vector of areas weights (`share`), in our case the population shares, a benchmark value (`bench`), and a method among `"raking"`, `"ratio"` and `"double"` (`method`). When the double benchmarking method is selected, the user must also indicate a second benchmark through the `H` argument, corresponding to the ensemble variability. The output is an object of class '`benchmark_fitsae`', being a list including the vector of benchmark estimates, the posterior risk, and relevant information about the call. The method `plot()` is available for '`benchmark_fitsae`' objects, displaying boxplots of original and benchmarked estimates in comparison with benchmark value. A '`benchmark_fitsae`' object may be also used as input of `map()` function, in order to spatially display benchmarked estimates, `extract()` or `export()` functions. The first option is included in the following code, whose visual output is in Figure 8.

```
R> shares <-  emilia_cs$pop / sum(emilia_cs$pop)
R> bmk <- benchmark(x = summ_beta,
+                   bench = 0.13,
+                   share = shares,
+                   method = "raking")
R> map(x = bmk,
+      spatial_df = emilia_shp,
+      spatial_id_domains = "NAME_DISTRICT")
```

Benchmarking can be done on the whole set of areas (default option) or even on a subset of them. In the latter case, the vector containing the names of the considered areas has to be indicated through the `areas` argument. Moreover, the function automatically takes out-of-sample estimates if they are involved in the benchmarking procedure. Benchmark estimates and posterior risk are stored within an object of class '`benchmark_fitsae`'.

```
R> subset <- c("RIMINI", "RICCIONE", "RUBICONE",
+              "CESENA - VALLE DEL SAVIO")
```

```
R> pop <- emilia_cs$pop[emilia_cs$id %in% subset]
R> shares_subset <- pop / sum(pop)
R> bmk_subset <- benchmark(x = summ_beta,
+                          bench = 0.13,
+                          share = shares_subset,
+                          method = "raking",
+                          areas = subset)
R> bmk_subset

Benchmarked estimates

* Adopted method: raking
* Benchmark for indicator: 0.13
* Weighted sum of original estimates: 0.122
* Number of considered areas: RIMINI, RICCIONE, RUBICONE,
  CESENA - VALLE DEL SAVIO

---------------------------------------------------------------------
Summaries of involved quantities
* Shares:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.169   0.198   0.212   0.250   0.264   0.407

* Benchmarked estimates:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.121   0.129   0.133   0.132   0.136   0.142

* Posterior Risk: 0

* Differences between original and benchmarked estimates:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.00837 -0.00837 -0.00837 -0.00837 -0.00837 -0.00837
```

For temporal models, a benchmark can be specified only for one time period at a time, indicated in the `time` argument.

4.4. Spatio-Temporal Examples

As explained in Section 2, it is possible to fit models that incorporate a spatial dependency structure, a temporal dependency structure or even both of them. The first extension, useful when the domains of interest are geographical entities, relaxes the assumption of spatial independence. Commonly, the boundaries across areas are arbitrarily set, and thus it can be reasonable to assume that the quantities of interest belonging to neighbouring areas are correlated. This can happen when dealing with data where the spatial dimension is relevant, e.g., agricultural, environmental, economic and epidemiological analyses. A spatial extension can be implemented through the `fit_sae()` function by switching to `TRUE` the `spatial_error` argument and supplying an object of class '`SpatialPolygonsDataFrame`' in `spatial_df` argument, being careful to include it ordered as the `data` object.

When dealing with panel data, such as the `emilia` dataset, a temporal dependency structure has to be taken into account due to the presence of repeated measures across time. It is possible to implement a temporal model by switching to `TRUE` the `temporal_error` argument and by providing the name of the dataset temporal variable in `temporal_variable` argument. Note that if a spatio-temporal model is required, the domain records should keep the same ordering within each recorded time in the `data` object. Hence, it is possible to re-order the shapefile accordingly, using the following commands.

```
R> data("emilia")
R> data("emilia_shp")
R> emilia_shp_ord@data <- emilia_shp@data[match(unique(emilia$id),
+                                      emilia_shp@data$NAME_DISTRICT), ]
```

The following code allows estimating a spatio-temporal model under a Beta likelihood. In presence of structured random effects within the model, our suggestion is to increase the `max_treedepth` argument above `10`, to improve the mixing of the HMC algorithm. After estimating the model, the ‘`fitsae`’ object can be explored through `summary()` method.

```
R> fit_ST <- fit_sae(formula_fixed = hcr ~ x,
+                    domains = "id",
+                    disp_direct = "vars",
+                    type_disp = "var",
+                    domain_size = "n",
+                    data = emilia,
+                    spatial_error = TRUE,
+                    spatial_df = emilia_shp_ord,
+                    temporal_error = TRUE,
+                    temporal_variable = "year",
+                    max_treedepth = 15,
+                    seed = 0)
R> summ_ST <- summary(fit_ST)
R> summ_ST
```

```
Summary for the SAE model call:
 fit_sae(formula_fixed = hcr ~ x, domains = "id", disp_direct = "vars",
    type_disp = "var", domain_size = "n", data = emilia,
    spatial_error = TRUE, spatial_df = emilia_shp,
    temporal_error = TRUE,  temporal_variable = "year",
    max_treedepth = 15, seed = 0, iter = 2000)

----- S.D. of the random effects: posterior summaries -----

         mean    sd  2.5%   25%   50%  75% 97.5%
sigma_t 0.104 0.022 0.061 0.090 0.104 0.12 0.148
sigma_s 0.297 0.055 0.205 0.259 0.292 0.33 0.418

----- Fixed effects coefficients: posterior summaries -----
```

```
              mean    sd   2.5%    25%    50%    75%  97.5%
(Intercept) -2.273 0.016 -2.305 -2.284 -2.273 -2.261 -2.242
x            0.123 0.020  0.084  0.109  0.123  0.136  0.163

--------------- Model diagnostics summaries ---------------

                   Min. 1st Qu. Median  Mean 3rd Qu.  Max.
Residuals        -0.024  -0.006  0.001 0.002   0.009 0.036
S.D. Reduction    0.113   0.372  0.456 0.445   0.517 0.677
Bayesian p-value  0.076   0.314  0.465 0.475   0.608 0.978

Shrinkage Bound Rate: 100 %

LOO Information Criterion:
          Estimate     SE
 elpd_loo  484.699  8.332
 p_loo      47.063  4.892
 looic    -969.398 16.664
```

In case of temporal or spatio-temporal object, it is possible to select the year of interest for map plotting via `map()` or when performing benchmarking as follows:

```
R> shares <- aggregate(emilia$pop, list(emilia$year),
+                      function(x) x / sum(x))
R> shares <- as.vector(t(shares[,-1]))
R> bmk_st <- benchmark(summ_ST,
+           bench = 0.09,
+           share = shares[1:38],
+           method = "raking",
+           time = "2014")
```

5. The Shiny Interface

The basic steps, constituting the workflow described in Section 4, have been embedded within a Shiny application that assists the user from the data loading step to the export of the outputs. The application can be launched without any preliminary action running the following command.

```
R> runShiny_tipsae()
```

A browser window is opened, which allows users to navigate on the application, being organized into 5 main tabs briefly described in what follows.

1. *Home*-page, where a schematic description of the application is provided.

2. *Data*-page, concerning the step of the data entry, providing also graphical exploratory tools. In the *Loading Data* subsection, a CSV file must be loaded, specifying the contents of the imported variable (e.g., response, covariates, dispersion parameter, etc.). The `"ols"` and `"gls"` smoothing procedure (see Section 2.3) can be carried out in the *Smoothing* part. Whereas, in *Load shapefile*, it is possible to include in the procedure a spatial structure: the user can choose to directly load a SHP file or an RDS file containing a '`SpatialPolygonsDataFrame`' object. The last tab, named *Data Summary*, allows an accurate data exploration before moving to the modelling step.

3. *Model Fitting*: where a small area model can be fitted. The application automatically constrains the model choice among those allowed by the input data. The progress of the model fitting is printed.

4. Once computations are completed, the mixing of the MCMC algorithm can be checked through graphical tools within *Check Convergence* tab.

5. Lastly, if the algorithm has properly converged, the *Results* tab can be visualized. Besides all the outputs described in Section 4, further graphical tools concerning the random effects are also reported.

Notice that the Shiny app does not include all the package tools. Specifically, the benchmarking procedure has not been implemented and the smoothing procedure does not include as option the `"kish"` method. Such options, however, may be included in future releases of the package.

6. Conclusions and Future Developments

The **tipsae** package is a dedicated tool for mapping proportions and indicators defined on the unit interval, widely used to measure, for instance, unemployment, educational attainment and also disease prevalence. To the best of our knowledge, it is the first package implementing Beta-based small area methods, particularly indicated for unit interval responses. Such methods, developed within a Bayesian framework, come equipped with a set of diagnostics and complementary tools, visualizing and exporting functions. The features of the **tipsae** package assist the user in carrying out a complete SAE analysis through the entire process of estimation, validation and results presentation, making the application of Bayesian algorithms and complex SAE methods straightforward. A Shiny application with a user-friendly interface can be launched to further simplifies the process.

Additional features to be integrated in future releases could be, firstly, the implementation of shrinking priors for the regression coefficients, useful for variable selection when several covariates are employed. Secondly, the Beta zero and/or one inflated version already implemented could fail when very few zero or one values are observed. Thus, a possible extension could comprise further flexible alternatives. Lastly, other directions may focus on model extensions for variance shrinking (You and Chapman 2006; Sugasawa, Tamae, and Kubokawa 2017), able to relax the assumption of known dispersion parameter, and for covariates measured with error (Arima, Datta, and Liseo 2015).

References

Arima S, Datta GS, Liseo B (2015). "Bayesian Estimators for Small Area Models When Auxiliary Information is Measured With Error." *Scandinavian Journal of Statistics*, **42**(2), 518–529.

Bauder M, Luery D, Szelepka S (2015). "Small Area Estimation of Health Insurance Coverage in 2010-2013." *Technical report*.

Bayes CL, Bazán JL, García C (2012). "A New Robust Regression Model for Proportions." *Bayesian Analysis*, **7**(4), 841–866.

Besag J, York J, Mollié A (1991). "Bayesian Image Restoration With Two Applications in Spatial Statistics." *Annals of the Institute of Statistical Mathematics*, **43**(1), 1–20.

Bivand RS, Pebesma E, Gomez-Rubio V (2013). *Applied Spatial Data Analysis with R, Second edition*. Springer-Verlag. URL https://asdar-book.org.

Boonstra HJ (2021). ***mcmcsae**: Markov Chain Monte Carlo Small Area Estimation*. R package version 0.7.0, URL https://CRAN.R-project.org/package=mcmcsae.

Brown PJ, Griffin JE (2010). "Inference With Normal-Gamma Prior Distributions in Regression Problems." *Bayesian Analysis*, **5**(1), 171–188.

Carpenter B, Gelman A, Hoffman MD, Lee D, Goodrich B, Betancourt M, Brubaker M, Guo J, Li P, Riddell A (2017). "Stan: A Probabilistic Programming Language." *Journal of Statistical Software*, **76**(1), 1–32.

Chang W, Cheng J, Allaire J, Sievert C, Schloerke B, Xie Y, Allen J, McPherson J, Dipert A, Borges B (2021). "shiny: Web Application Framework for R." R package version 1.6.0, URL https://CRAN.R-project.org/package=shiny.

Datta GS, Ghosh M, Steorts R, Maples J (2011a). "Bayesian Benchmarking With Applications to Small Area Estimation." *Test*, **20**(3), 574–588.

Datta GS, Hall P, Mandal A (2011b). "Model Selection by Testing for the Presence of Small-Area Effects, and Application to Area-Level Data." *Journal of the American Statistical Association*, **106**(493), 362–374.

De Nicolò S, Ferrante MR, Pacei S (2021). "Put Inequality on the Map: Small Area Estimation of Inequality Measures using a Beta Mixture." Working Paper.

De Nicolò S, Gardini A (2022). ***tipsae**: Tools for Handling Indices and Proportions in Small Area Estimation*. R package version 0.0.4, URL https://CRAN.R-project.org/package=tipsae.

Esteban MD, Lombardía MJ, López-Vizcaíno E, Morales D, Pérez A (2020). "Small Area Estimation of Proportions Under Area-Level Compositional Mixed Models." *Test*, **29**(3), 793–818.

Esteban MD, Morales D, Pérez A, Santamaría L (2012). "Small Area Estimation of Poverty Proportions Under Area-Level Time Models." *Computational Statistics & Data Analysis*, **56**(10), 2840–2855.

Fabrizi E, Ferrante M, Trivisano C (2016). "Hierarchical Beta Regression Models for the Estimation of Poverty and Inequality Parameters in Small Areas." *Analysis of Poverty Data by Small Area Methods. John Wiley and Sons*, pp. 299–314.

Fabrizi E, Ferrante MR, Pacei S, Trivisano C (2011). "Hierarchical Bayes Multivariate Estimation of Poverty Rates Based on Increasing Thresholds for Small Domains." *Computational Statistics & Data Analysis*, **55**(4), 1736–1747.

Fabrizi E, Ferrante MR, Trivisano C (2018). "Bayesian Small Area Estimation for Skewed Business Survey Variables." *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, **67**(4), 861–879.

Fabrizi E, Ferrante MR, Trivisano C (2020). "A Functional Approach to Small Area Estimation of the Relative Median Poverty Gap." *Journal of the Royal Statistical Society A (Statistics in Society)*, **183**(3), 1273–1291.

Fabrizi E, Trivisano C (2016). "Small Area Estimation of the Gini Concentration Coefficient." *Computational Statistics & Data Analysis*, **99**, 223–234.

Ferrari S, Cribari-Neto F (2004). "Beta Regression for Modelling Rates and Proportions." *Journal of Applied Statistics*, **31**(7), 799–815.

Figueroa-Zúñiga JI, Arellano-Valle RB, Ferrari SL (2013). "Mixed Beta Regression: A Bayesian Perspective." *Computational Statistics & Data Analysis*, **61**, 137–147.

Freni-Sterrantino A, Ventrucci M, Rue H (2018). "A Note on Intrinsic Conditional Autoregressive Models for Disconnected Graphs." *Spatial and Spatio-Temporal Epidemiology*, **26**, 25–34.

Gabry J, Goodrich B, Lysy M (2020). "**rstantools**: Tools for Developing R Packages Interfacing with Stan." R package version 2.1.1, URL https://CRAN.R-project.org/package=rstantools.

Gabry J, Mahr T (2021). ***bayesplot**: Plotting for Bayesian Models.* R package version 1.8.1, URL https://mc-stan.org/bayesplot.

Gelman A (2006). "Prior Distributions for Variance Parameters in Hierarchical Models." *Bayesian Analysis*, **1**(3), 515–534.

Giovinazzi F, Cocchi D (2021). "Social Integration of Second Generation Students in the Italian School System." *Social Indicators Research*, pp. 1–21.

Goodrich B, Gabry J, Ali I, Brilleman S (2020). ***rstanarm**: Bayesian Applied Regression Modeling via Stan.* R package version 2.21.1, URL https://mc-stan.org/rstanarm.

IBM Corporation (2010). *IBM SPSS Statistics 19.* IBM Corporation, Armonk, NY. URL http://www-01.ibm.com/software/analytics/spss/.

Janicki R (2020). "Properties of the Beta Regression Model for Small Area Estimation of Proportions and Application to Estimation of Poverty Rates." *Communications in Statistics-Theory and Methods*, **49**(9), 2264–2284.

Kalton G, Brick J, Le T (2005). "Estimating Components of Design Effects for Use in Sample Design." *Household Sample Surveys in Developing and Transition Countries (Sales No. E. 05. XVII. 6)*.

Kish L (1992). "Weighting for Unequal Pi." *Journal of Official Statistics*, **8**(2), 183.

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.

Liu B, Lahiri P, Kalton G (2007). "Hierarchical Bayes Modeling of Survey-Weighted Small Area Proportions." In *Proceedings of the American Statistical Association, Survey Research Section*, pp. 3181–3186.

Marhuenda Y, Molina I, Morales D (2013). "Small Area Estimation With Spatio-Temporal Fay-Herriot Models." *Computational Statistics & Data Analysis*, **58**, 308–325.

Marhuenda Y, Morales D, del Carmen Pardo M (2014). "Information Criteria for Fay-Herriot Model Selection." *Computational Statistics & Data Analysis*, **70**, 268–280.

Migliorati S, Di Brisco AM, Ongaro A (2018). "A New Regression Model for Bounded Responses." *Bayesian Analysis*, **13**(3), 845–872.

Molina I, Marhuenda Y (2015). "**sae**: An R Package for Small Area Estimation." *The R Journal*, **7**(1), 81.

Morales D, Pagliarella MC, Salvatore R (2015). "Small Area Estimation of Poverty Indicators Under Partitioned Area-Level Time Models." *SORT*, **39**(1), 19–34.

Morris M, Wheeler-Martin K, Simpson D, Mooney SJ, Gelman A, DiMaggio C (2019). "Bayesian Hierarchical Spatial Models: Implementing the Besag York Mollié Model in Stan." *Spatial and Spatio-Temporal Epidemiology*, **31**, 100301.

Ospina R, Ferrari SL (2010). "Inflated Beta Distributions." *Statistical Papers*, **51**(1), 111–126.

Pfeffermann D, Sikov A, Tiller R (2014). "Single- and Two-Stage Cross-Sectional and Time Series Benchmarking Procedures for Small Area Estimation." *Test*, **23**(4), 631–666.

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.

Ranalli MG, Montanari GE, Vicarelli C (2018). "Estimation of Small Area Counts With the Benchmarking Property." *Metron*, **76**(3), 349–378.

Rao JN, Molina I (2015). *Small-Area Estimation*. Wiley Series in Survey Methodology.

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org.

Riebler A, Sørbye SH, Simpson D, Rue H (2016). "An Intuitive Bayesian Spatial Model for Disease Mapping That Accounts for Scaling." *Statistical Methods in Medical Research*, **25**(4), 1145–1165.

SAS Institute Inc (2003). *SAS/STAT Software, Version 9.1*. Cary, NC. URL https://www.sas.com/.

Schmid T, Bruckschen F, Salvati N, Zbiranski T (2017). "Constructing Sociodemographic Indicators for National Statistical Institutes by Using Mobile Phone Data: Estimating Literacy Rates in Senegal." *Journal of the Royal Statistical Society A (Statistics in Society)*, **180**(4), 1163–1190.

Souza DF, Moura FA (2016). "Multivariate Beta Regression with Application in Small Area Estimation." *Journal of Official Statistics*, **32**(3), 747–768.

Stan Development Team (2020). ***RStan**: The R Interface to Stan*. R package version 2.21.2, URL http://mc-stan.org.

Stata Corporation (2007). *Stata Data Analysis and Statistical Software*. Stata Corporation. URL http://www.stata.com/.

Sugasawa S, Tamae H, Kubokawa T (2017). "Bayesian Estimators for Small Area Models Shrinking both Means and Variances." *Scandinavian Journal of Statistics*, **44**(1), 150–167.

Vehtari A, Gabry J, Magnusson M, Yao Y, Bürkner PC, Paananen T, Gelman A (2020). ***loo**: Efficient Leave-One-Out Cross-Validation and WAIC for Bayesian Models*. R package version 2.4.1, URL https://mc-stan.org/loo.

Vehtari A, Gelman A, Gabry J (2017). "Practical Bayesian Model Evaluation Using Leave-One-Out Cross-Validation and WAIC." *Statistics and Computing*, **27**(5), 1413–1432.

Wieczorek J, Hawala S (2011). "A Bayesian Zero-One Inflated Beta Model for Estimating Poverty in US Counties." In *Proceedings of the American Statistical Association, Section on Survey Research Methods, Alexandria, VA: American Statistical Association.*

Wieczorek J, Nugent C, Hawala S (2012). "A Bayesian Zero-One Inflated Beta Model for Small Area Shrinkage Estimation." In *Proceedings of the 2012 Joint Statistical Meetings, American Statistical Association, Alexandria, VA.*

You Y, Chapman B (2006). "Small Area Estimation Using Area Level Models and Estimated Sampling Variances." *Survey Methodology*, **32**(1), 97.

You Y, Rao J (2002). "Small Area Estimation Using Unmatched Sampling and Linking Models." *Canadian Journal of Statistics*, **30**(1), 3–15.

Zhang JL, Bryant J (2020). "Fully Bayesian Benchmarking of Small Area Estimation Models." *Journal of Official Statistics*, **36**(1), 197–223.

Affiliation:

Silvia De Nicolò
Dipartimento di Scienze Statistiche "P.Fortunati"

Università di Bologna
Via Belle Arti, 41
40126, Bologna (BO), Italy
E-mail: silvia.denicolo@unibo.it
URL: https://www.unibo.it/sitoweb/silvia.denicolo/en
and
Aldo Gardini
Dipartimento di Scienze Statistiche "P.Fortunati"
Università di Bologna
Via Belle Arti, 41
40126, Bologna (BO), Italy
E-mail: aldo.gardini@unibo.it
URL: https://www.unibo.it/sitoweb/aldo.gardini/en