The Survey Statistician, 2024, Vol. 89, 76–85.

Book and Software Review

SAE models for indicators in the unit interval: The tipsae R package and its Shiny interface

Silvia De Nicolò^1 and Aldo Gardini^2

^{1}University of Bologna, Italy, silvia.denicolo@unibo.it
^{2}University of Bologna, Italy, aldo.gardini@unibo.it

Abstract

The **tipsae** R package is a dedicated tool for mapping proportions and indicators defined on the unit interval, e.g., common poverty and inequality measures, in the framework of small area estimation. It implements Beta-based Bayesian Hierarchical models at the area-level through Stan language. A set of diagnostics, exploratory analysis and complementary tools, such as benchmarking and variance smoothing methods, complete the package. To enhance accessibility for practitioners, we have developed a user-friendly Shiny application alongside the R package. The purpose of this paper is to illustrate its potential by describing the workflow of a typical small area analysis and by providing the underlying R code. The current version 0.0.18 is avalaible on https://cran.r-project.org/web/packages/tipsae/index.html

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

1 Introduction and software review

Timely and reliable statistical estimates defined for granular geographical levels or specific socio-demographic groups are increasingly in demand. Many of those require an extensive exploitation of survey data; nonetheless, domains or areas of study are often different from the ones originally planned in survey designs. This leads to small sample sizes and consequent unreliable estimates. In this context, the Small Area Estimation (SAE) framework provides a set of indirect estimation techniques that relies on external information, borrowing strength across areas and producing estimates of interest with an acceptable level of uncertainty.

The SAE estimators that are based on regression models are named model-based estimators. The small area models are divided into two strands: the models defined at the area level and the ones defined at the unit level. The first strategy relates area-specific survey estimators to areal covariates, while the second one ties individual observations of the underlying variable to individual covariates. We focus on area-level models as they prove to be highly convenient in practice. Indeed, they only require covariates defined at the area level and account for design-based properties; whereas unit-level models generally need auxiliary information available for the entire population and do not consider survey weights.

We focus on indicators defined on the unit interval, being common in SAE modelling because of the high presence of rates and proportions releases in official statistics. Two different bodies of literature relate with this field in the area-level context, revolving around linear mixed models with suitable transformations (Rao and Molina, 2015) and Beta regression models (Janicki, 2020). An additional strand, common in disease mapping, explicitly models counts via Binomial or Poisson models (Hobza et al., 2018; Wakefield, 2007; MacNab, 2003).

Several R packages implement SAE tools. By focusing on model-based methods, the most complete packages are **sae** (Molina and Marhuenda, 2015), **emdi** (Kreutzmann et al., 2019) and **mcmcsae** (Boonstra, 2021). The **sae** and **emdi** packages implements both area-level and unit-level models from a frequentist perspective. The latter one provides suitable model diagnostics, plots, and exporting tools. Conversely, the **mcmcsae** package implements SAE models in a Bayesian framework via Markov chain Monte Carlo (MCMC) simulation. Such package considers both area and unit level models, allowing also for spatial and temporal dependencies. It includes different prior settings, model diagnostics, and posterior predictive checks functions. Lastly, count models are implemented in the **zipsae** package (Utomo and Wulansari, 2021) with zero-inflated poisson models and **saeeb** package (Fauziah and Wulansari, 2020) with poisson-gamma models.

Among those listed, only the **emdi** package directly accounts for unit interval responses by implementing the Fay-Herriot model with arc-sin transformation (Schmid et al., 2017). In this context, small area models based on the Beta regression lack of a proper implementation and the **tipsae** package fills this gap (De Nicolò and Gardini, 2022). The package, available on CRAN, implements area-level models comprising the standard Beta regression model, Zero and/or One Inflated Beta (Wieczorek et al., 2012) and Flexible Beta (De Nicolò et al., 2023) models. Moreover, spatial, temporal and spatio-temporal dependence structures can be included. Note that Beta mixed regression models are already implemented in R through specific packages that do not accommodate peculiarities of small area models, such as the assumption of known dispersion parameter and popular structured random effects.

The Bayesian inferential framework, implemented through the Stan routine (Carpenter et al., 2017), allows to manage non-conventional likelihood assumptions and capture the uncertainty on target parameters through posterior inference. It assists the user in carrying out a complete SAE analysis, from data loading step, the implemented functions contemplate the entire process of data exploration, model estimation and validation, presentation and exportation of results. This facilitate the use Bayesian models and complex SAE methods for practitioners, building bridges between methodological and applied fields. The package comes equipped with a Shiny application to further facilitate the workflow for non-expert users.

The paper is organized as follows. Section 2 briefly illustrates the **tipsae** package and its main features, describing the various inferential settings that can be implemented. Section 3 outlines the workflow of a SAE procedure using the R Shiny application and then presents the corresponding R code. Concluding remarks are drawn in Section 4.

2 tipsae in a nutshell

The main feature of the **tipsae** package is that it includes a variety of area-level models based on the Beta likelihood with different prior settings to be defined by the users. The inclusion of several methodological tools facilitates the customization and suitability of the model given different practical situations. The statistical methods implemented in the **tipsae** package can be estimated using the function `fit_sae()` and are briefly summarized in the following.

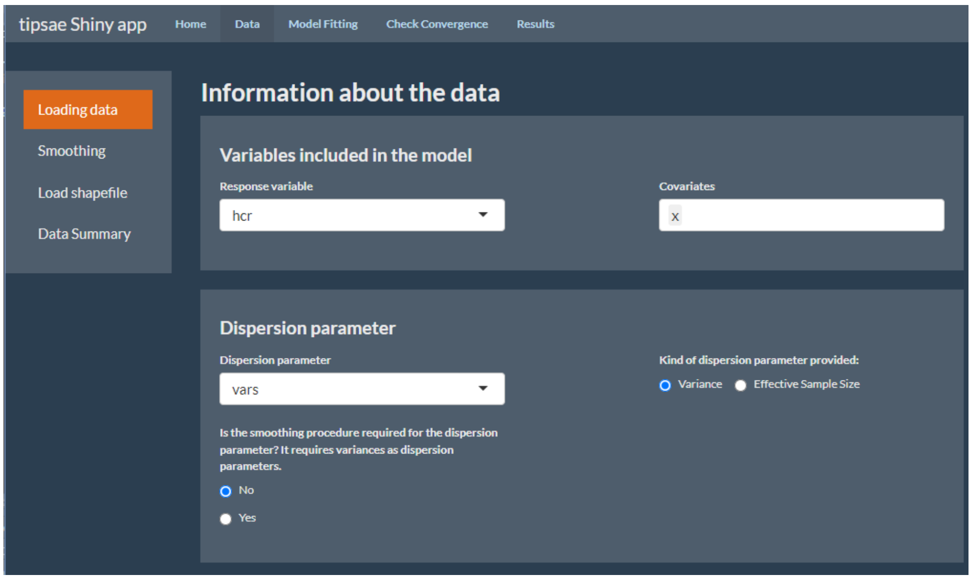
The standard Beta-regression model is useful to handle responses with double-bounded support. It easily manages different distributional shapes while having a simple location-dispersion parametrization. However, its support does not include 0 and 1 values which may be observed in some applications. A model able to encompass them is required, thus, the package includes also the Zero and/or One Inflated Beta (ZOIB) model (Wieczorek et al., 2012). Lastly, when the distribution of the response is characterized by heavy tails and/or high skewness, the standard Beta regression could fail (Bayes et al., 2012; Migliorati et al., 2018). In this case, the package includes the more suitable Flexible Beta model. It is defined as a mixture of two Beta random variables and is proposed in small area estimation by De Nicolò et al. (2023).

To facilitate practitioners, standard wide-range prior distributions are assumed for model parameters. The priors for the random effects include the case of unstructured random effects, spatially structured random effects, and temporal random effects. The unstructured random effect accounts for area-specific deviations from the synthetic predictor. The package includes three different strategies to specify its prior distribution, that can be chosen through the `prior_reff` argument of `fit_sae()` function. As a first option, a zero-mean normal prior is considered (the default option, namely `prior_reff ="normal"`). When covariates have poor explanatory power in some domains, a more flexible handling of random effect is required. As a second option, the package implements a robust Student's t prior with exponential hyperprior for degrees of freedom (`prior_reff ="t"` option, Fabrizi and Trivisano, 2016). The third option contemplates the presence of very informative covariates, in that case the variability of the small area parameters may not require the inclusion of a random effect term (Datta et al., 2011). Therefore, the variance gamma shrinkage prior is included as prior choice (`prior_reff ="VG"` option, Fabrizi et al., 2018). In all the cases, the scale (or global scale) parameter has an half-normal hyperprior whose scale can be set using the `scale_prior` argument.

By setting the argument `spatial_error = TRUE`, the user can specify a spatially structured effect to the linear predictor, in addition to the unstructured one. The vector of spatial random effect has an intrinsic conditional autoregressive (ICAR) prior (Besag et al., 1991). The spatial structure of the areas, in the form of an object of class `'SpatialPolygonsDataFrame'` (from the **sp** package, Bivand et al., 2013) or class `'sf'` (from the **sf** package, Pebesma, 2018), is required as input of the `spatial_df` argument. When multiple observations of the target indicator are observed for different time periods, a temporal model can be specified in order to borrow strength from time repetitions. The user may choose to add a temporal random effect in the linear predictor in addition to the unstructured one by setting `temporal_error = TRUE` and declare the name of the temporal variable in `temporal_variable`. Its prior is a random walk prior of order 1, assuming independence among the areas. If both temporal and spatial error argument are set to `TRUE`, a spatio-temporal model is fitted.

Posterior inference is carried out through an efficient Hamiltonian Monte Carlo (HMC) fitting algorithm and parallel computing imported from **rstan** (Stan Development Team, 2020). The package provides out-of-sample predictions available through the `export()` function. In practice, when an area is out-of-sample but its auxiliary information is observed, its direct estimate can be included in the dataset labeled as `NA`. In such a way, the function automatically draws a sample from the posterior distribution of the predictor by combining the samples drawn from the posterior of all the involved parameters.

One of the output of the `fit_sae` function is the `'stanfit' S4` object produced by the **rstan** package. This can be exploited to check convergence, monitor sampler diagnostics, and, lastly, perform an exhaustive posterior analysis by relying on existing tools, e.g. **loo** and **bayesplot** packages (Gabry et al., 2019). Moreover, specific diagnostics for small area models are produced by ad-hoc functions, facing the most relevant aspects to deepen within a SAE framework. The package supplies both visualization tool for graphical assessments and functions that easily export the final results. Lastly, variance smoothing routines for pre-processing and benchmarking procedures for post-processing are provided as complementary tools. A Shiny application (Chang et al., 2021) with an intuitive



Information about the data

Variables included in the model

Response variable: hcr

Covariates: x

Dispersion parameter

Dispersion parameter: vars

Kind of dispersion parameter provided: [✓ Variance] Effective Sample Size

Is the smoothing procedure required for the dispersion parameter? It requires variances as dispersion parameters. [✓ No] Yes

Figure 1: Focus on the Data tab: overview on the section where it is possible to load data and specify the interesting variables.

graphical user interface can be launched through `runShiny_tipsae()`.

3 Workflow of a SAE procedure: R code and R Shiny

The objective of this section is to illustrate the value of the Shiny application as a powerful tool for encouraging practitioners to embrace advanced statistical techniques. Specifically, we compare the input and output features of the Shiny app with those obtained using the R language. The application can be easily accessed by executing the following commands[1].

```
R> library(tipsae)
R> runShiny_tipsae()
```

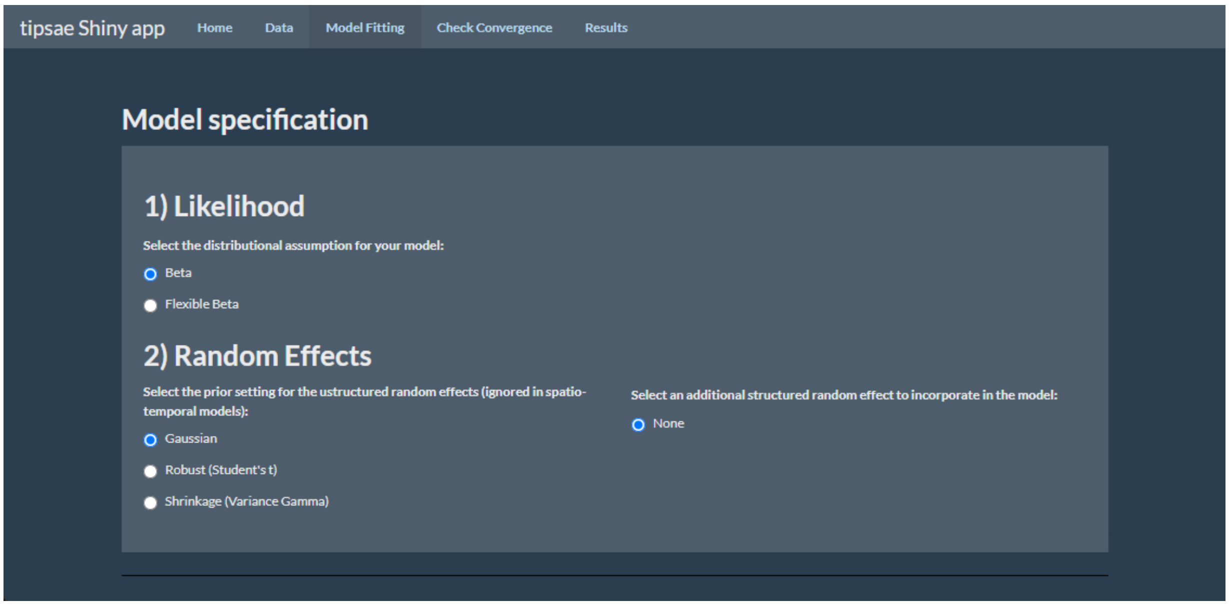
When the application is launched, a browser window appears, providing users with easy navigation through the interface. The interface is thoughtfully organized into five main sections. The **Home** page features a schematic description of the application along with pertinent references. Within this page, the 'Load dataset' button facilitates testing the application by executing an analysis using the toy dataset `emilia_cs` provided within the package. Clicking this button is equivalent to running the R command:

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

3.1 Data input and exploratory anlysis

The **Data** page is organized into four subsections, with the initial three tabs dedicated to data entry steps and pre-treatment, while the last tab offers graphical exploratory tools. In the *Loading Data* tab, users can upload a CSV file and visualize the loaded data by clicking the "View loaded dataset" button. Essential details about the data must be provided, such as the nature of inserted variables, labeling the response, covariates, and specifying dispersion values, indicating whether it is variance or effective sample size (refer to Figure 1). This tab also allows users to configure smoothing procedures if necessary. Additional information, including a possible time variable, domain ids, and sample sizes, can be specified for subsequent visual diagnostics. When a smoothing procedure is

[1]The following R packages should be installed before running the commands: tipsae, shinythemes, shinyFeedback, shinybusy, shinyWidgets, leaflet, shinyjs, sp, bayesplot.



Model specification

1) Likelihood

Select the distributional assumption for your model:

[✓ Beta]

Flexible Beta

2) Random Effects

Select the prior setting for the ustructured random effects (ignored in spatio-temporal models):

[✓ Gaussian]

Robust (Student's t)

Shrinkage (Variance Gamma)

Select an additional structured random effect to incorporate in the model:

[✓ None]

Figure 2: Model Fitting Section of **tipsae** Shiny App

configured, the *Smoothing* subsection permits users to adjust settings and visually inspect the procedure output. The implemented procedure is a Generalized Variance Function smoothing technique (Wolter, 2007, Ch.7); for a detailed methodological explanation, please consult the package vignette. In subsection *Load Shapefile*, a spatial structure can be incorporated, loading either a SHP file either an RDS file containing a `’SpatialPolygonsDataFrame’` or `’sf’` object. Such object would enable to account for spatial dependencies in the model and/or plot maps with relevant quantities. The last subsection, called *Data Summary*, provides an accurate data exploration before moving to the modelling step, depicting the distribution of the response variable and its relationship with the covariates and the dispersion measure.

3.2 Model fitting

The **Model Fitting** section employs the Stan routine for the estimation of the Bayesian models, as detailed in Carpenter et al. (2017). The *Model Specification* tab allows users to define the model likelihood, while the application automatically constraints the model choice depending on the input data. Specifically, when all response observations $y_i \in (0, 1)$, $\forall i$, the allowed options are Beta and Flexible Beta models. If some observations are recorded as zeros and/or ones, the application automatically limits the choice to Zero and/or One Inflated Beta models. Furthermore, users have the flexibility to choose their preferred prior distribution for random effects from the options listed in Section 2. If a shape file is loaded, users can also decide whether to include a spatial random effect in the predictor. It is noteworthy that when analyzing a panel dataset with an available temporal variable, the application automatically incorporates a temporal random effect.

Users can customize certain algorithm settings in the *Settings about the MCMC Algorithm* tab. This includes parameters such as the number of iterations per chain, including warm-up (set as half of the total iterations), enabling parallel computation with the appropriate tick, determining the number of chains, and specifying the number of cores. Additional HMC options include the maximum allowed tree depth (Maximum treedepth) and the target average proposal acceptance probability (adapt_delta). For a more in-depth understanding, users can refer to the Stan documentation. The ”Fit Model” button initiates the estimation process, whose progress is displayed through an iterative printed output.

Note that the model, as configured in Figure 2 and guided by the data input decisions in Figure 1,

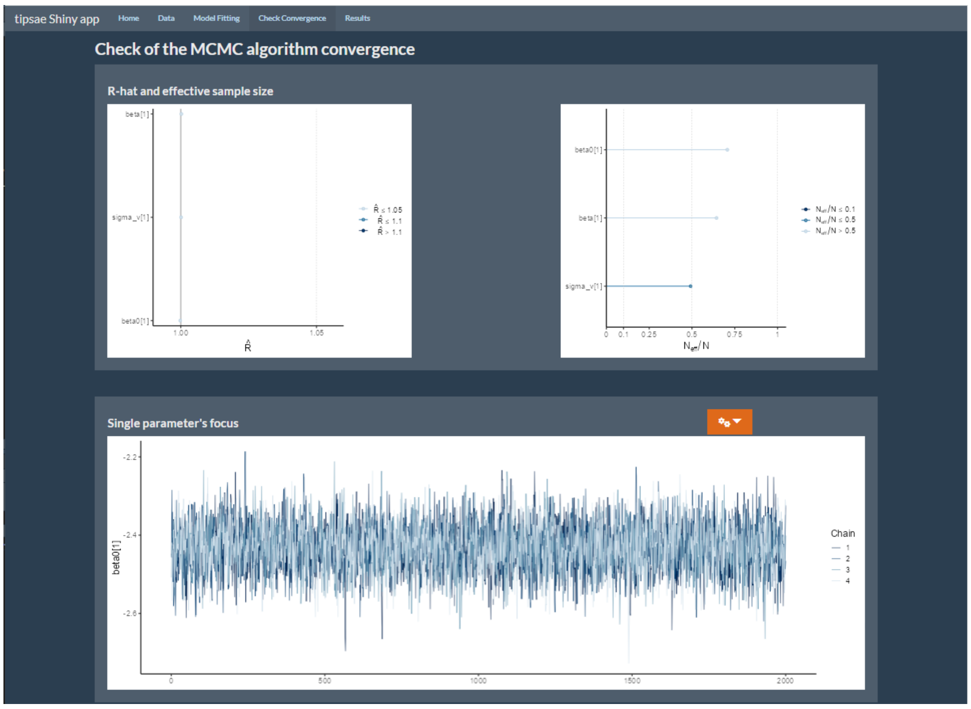


Figure 3: Check convergence section of the Shiny app.

can also be implemented using the following lines of R code. This code produces an object with the class 'fitsae'.

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

3.3 Analysing posterior results

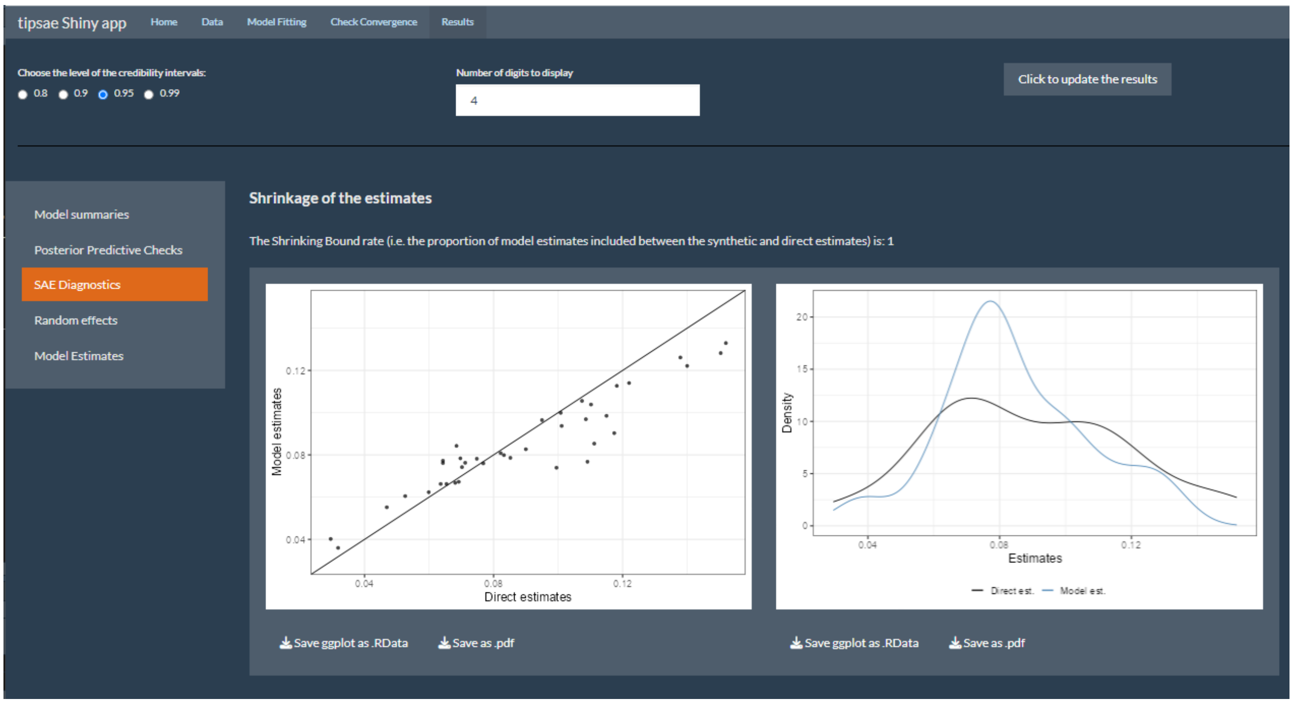
After fitting a Bayesian model through a MCMC methods, the proper convergence of the algorithm needs to be assessed before moving to the analysis of the results. This can be done by relying on the useful tool provided by the `bayesplot` package (Gabry and Mahr, 2021). Such functionalities are exploited in the **Check Convergence** section of the Shiny application, where posterior densities, chains trace-plots, autocorrelation functions and rank plots are displayed (Figure 3).

The same plots can be obtained through R code, recalling that the 'stanfit' objects is contained by the result of the `fit_sae` function.

```
R> library(bayesplot)
R> mcmc_rhat(rhat(fit_beta$stanfit, pars = c("beta0", "beta", "sigma_v")))
R> mcmc_neff(neff_ratio(fit_beta$stanfit, pars = c("beta0", "beta", "sigma_v")))
R> mcmc_trace(as.array(fit_beta$stanfit, pars = c("beta0")))
```

When the convergence of the MCMC algorithm is achieved, the user can move to the evaluation of the model results, accessing the **Results** section of the application. The *Model Summaries* sub-section provides posterior syntheses of regression coefficients and random effects. Furthermore, a summary of residuals is also reported, including an histogram, and the LOO Information Criterion can be computed through the button “Click to compute LOOIC”, enabling for model selection (Vehtari et al., 2017). Such summary measures can be explored also in R by printing the result of the `summary` method applied to a 'fitsae' object:

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



Choose the level of the credibility intervals:
0.8 0.9 [✓ 0.95] 0.99

Number of digits to display: 4

Shrinkage of the estimates

The Shrinking Bound rate (i.e. the proportion of model estimates included between the synthetic and direct estimates) is: 1

Figure 4: Results section of the Shiny application. A focus on the available SAE diagnostics.

```
R> print(summ_model)
```

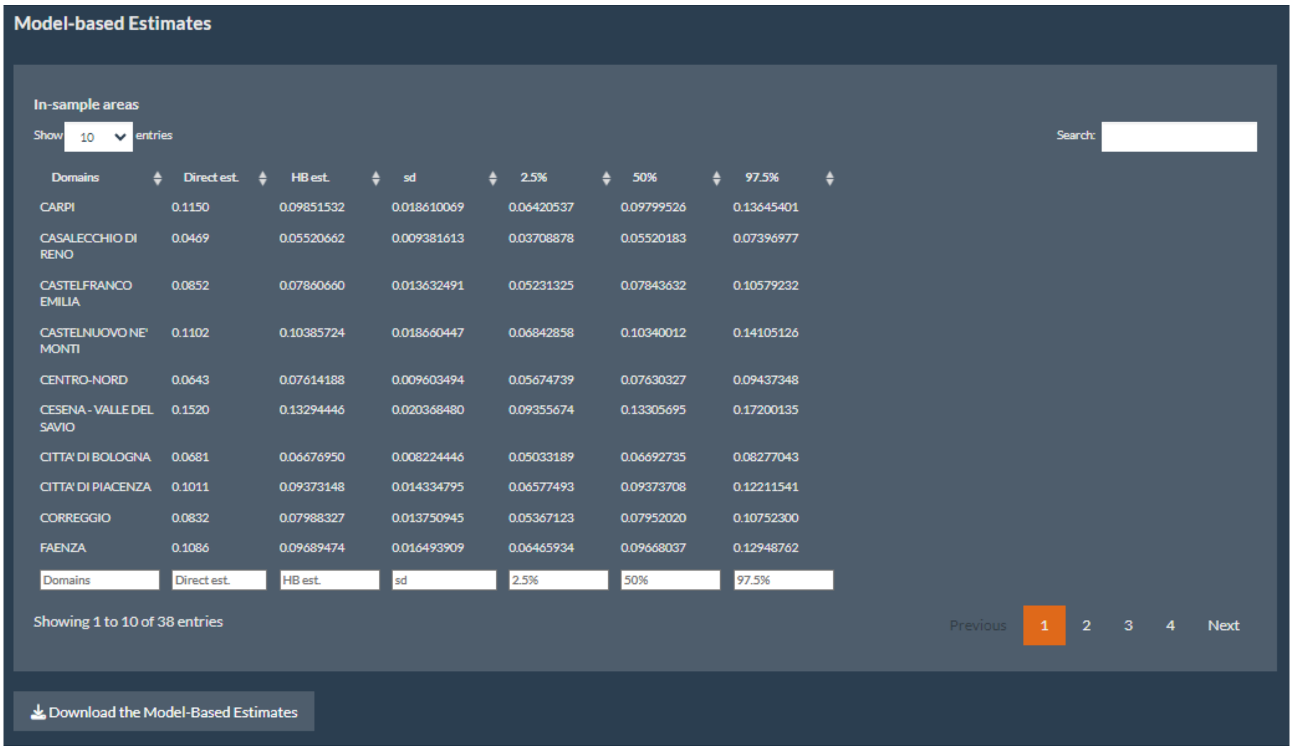
The *Posterior Predictive Check* subsection displays the sample data kernel density versus those of the datasets generated from the posterior predictive distribution, denoted with $Y_d^{\bullet}|\boldsymbol{y},\ d = 1, \ldots, D$, in order to assess the goodness of fit. Here, a specific tab focuses on area-specific Bayesian p-values, defined as $BP_d = \mathbb{P}\left[Y_d^{\bullet} > y_d|\boldsymbol{y}\right]$ (Fabrizi et al., 2011). 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. Summary statistics of Bayesian p-values are printed also in the output of the `summary` method applied to `'fitsae'` objects. In this section, also visual inspections of posterior predictive checks are shown, being implemented through the functions available in the `bayesplot` package.

Small area specific diagnostics have a proper subsection (*SAE diagnostics*), visually illustrating the shrinking process induced on model-based estimates and comparing direct and model-based estimates (Table 4). The standard deviations of both type of estimates are compared and summaries of a measure of standard deviation reduction are provided. The *Random Effect* subsection compares the standardized effects density versus the one of a standard normal and a caterpillar plot, comparing their posterior distributions for each area. Many basic plots reported in this part of the Shiny application can be obtained through the following R code:

```
R> plot(summ_model)
R> density(summ_model)
```

Lastly, the *Model Estimates* subsection displays a table with direct and model-based estimates, including relevant posterior summaries of target parameters. Such object can be downloaded in CSV format via a proper button (Figure 5). A caterpillar plot of the target parameter posteriors is also provided. To extract the whole set of estimated from the model fitted in R, the following command can be used:

```
R> extract(summ_model)
```



Domains	Direct est.	HB est.	sd	2.5%	50%	97.5%
CARPI	0.1150	0.09851532	0.018610069	0.06420537	0.09799526	0.13645401
CASALECCHIO DI RENO	0.0469	0.05520662	0.009381613	0.03708878	0.05520183	0.07396977
CASTELFRANCO EMILIA	0.0852	0.07860660	0.013632491	0.05231325	0.07843632	0.10579232
CASTELNUOVO NE' MONTI	0.1102	0.10385724	0.018660447	0.06842858	0.10340012	0.14105126
CENTRO-NORD	0.0643	0.07614188	0.009603494	0.05674739	0.07630327	0.09437348
CESENA - VALLE DEL SAVIO	0.1520	0.13294446	0.020368480	0.09355674	0.13305695	0.17200135
CITTA' DI BOLOGNA	0.0681	0.06676950	0.008224446	0.05033189	0.06692735	0.08277043
CITTA' DI PIACENZA	0.1011	0.09373148	0.014334795	0.06577493	0.09373708	0.12211541
CORREGGIO	0.0832	0.07988327	0.013750945	0.05367123	0.07952020	0.10752300
FAENZA	0.1086	0.09689474	0.016493909	0.06465934	0.09668037	0.12948762
Domains	Direct est.	HB est.	sd	2.5%	50%	97.5%

Showing 1 to 10 of 38 entries

Figure 5: Results section of the Shiny application. A focus on the table with model-based estimates.

4 Concluding remarks

The **tipsae** Shiny application serves as a simple and convenient tool to map indicators on the unit interval in a SAE framework, offering a user-friendly interface for data visualization and analysis. However, it is essential to acknowledge the inherent limitations in terms of flexibility that come with its ease of use. We refer to the use of the **tipsae** R package for a higher level of customization and for additional tools, such as the benchmarking procedure and the estimation of area-specific design effects, not implemented in the Shiny app. For a comprehensive understanding of the package itself and its features, it is recommended to refer to the CRAN vignette [2], which provides detailed information on what sets it apart. While recognizing its constraints, we encourage the survey statistics community to actively engage with Shiny applications. Its potential for fostering collaboration and enhancing data communication is noteworthy.

References

C. L. Bayes, J. L. Bazán, and C. García. A new robust regression model for proportions. *Bayesian Analysis*, 7(4):841–866, 2012.

J. Besag, J. York, and A. Mollié. Bayesian image restoration with two applications in spatial statistics. *Annals of the Institute of Statistical Mathematics*, 43(1):1–20, 1991.

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

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

B. Carpenter, A. Gelman, M. D. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. Brubaker, J. Guo,

[2]see https://cran.r-project.org/web/packages/tipsae/index.html

P. Li, and A. Riddell. Stan: A probabilistic programming language. *Journal of Statistical Software*, 76(1):1–32, 2017.

W. Chang, J. Cheng, J. Allaire, C. Sievert, B. Schloerke, Y. Xie, J. Allen, J. McPherson, A. Dipert, and B. Borges. shiny: Web application framework for r, 2021. URL https://CRAN.R-project.org/package=shiny. R package version 1.6.0.

G. S. Datta, P. Hall, and A. Mandal. 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, 2011.

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

S. De Nicolò, M. R. Ferrante, and S. Pacei. Small area estimation of inequality measures using mixtures of Beta. *Journal of the Royal Statistical Society Series A: Statistics in Society*, page NA, 2023. ISSN 0964-1998. URL https://doi.org/10.1093/jrsssa/qnad083.

E. Fabrizi and C. Trivisano. Small area estimation of the gini concentration coefficient. *Computational Statistics & Data Analysis*, 99:223–234, 2016.

E. Fabrizi, M. R. Ferrante, S. Pacei, and C. Trivisano. Hierarchical bayes multivariate estimation of poverty rates based on increasing thresholds for small domains. *Computational Statistics & Data Analysis*, 55(4):1736–1747, 2011.

E. Fabrizi, M. R. Ferrante, and C. Trivisano. Bayesian small area estimation for skewed business survey variables. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 67(4): 861–879, 2018.

R. A. Fauziah and I. Y. Wulansari. Package ‘saeeb’. 2020.

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

J. Gabry, D. Simpson, A. Vehtari, M. Betancourt, and A. Gelman. Visualization in bayesian workflow. *Journal of the Royal Statistical Society A (Statistics in Society)*, 182(2):389–402, 2019.

T. Hobza, D. Morales, and L. Santamaría. Small area estimation of poverty proportions under unit-level temporal binomial-logit mixed models. *Test*, 27:270–294, 2018.

R. Janicki. 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, 2020.

A.-K. Kreutzmann, S. Pannier, N. Rojas-Perilla, T. Schmid, M. Templ, and N. Tzavidis. The R package emdi for estimating and mapping regionally disaggregated indicators. *Journal of Statistical Software*, 91(7):1–33, 2019. doi: 10.18637/jss.v091.i07.

Y. C. MacNab. Hierarchical bayesian spatial modelling of small-area rates of non-rare disease. *Statistics in Medicine*, 22(10):1761–1773, 2003.

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

I. Molina and Y. Marhuenda. sae: An R package for small area estimation. *The R Journal*, 7(1):81, 2015.

E. Pebesma. Simple Features for R: Standardized Support for Spatial Vector Data. *The R Journal*, 10 (1):439–446, 2018. doi: 10.32614/RJ-2018-009. URL https://doi.org/10.32614/RJ-2018-009.

J. N. Rao and I. Molina. *Small-Area Estimation*. Wiley Series in Survey Methodology, 2015.

T. Schmid, F. Bruckschen, N. Salvati, and T. Zbiranski. 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, 2017.

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

F. W. Utomo and I. Y. Wulansari. Package ‘zipsae’, 2021.

A. Vehtari, A. Gelman, and J. Gabry. Practical bayesian model evaluation using leave-one-out cross-validation and waic. *Statistics and Computing*, 27(5):1413–1432, 2017.

J. Wakefield. Disease mapping and spatial regression with count data. *Biostatistics*, 8(2):158–183, 2007.

J. Wieczorek, C. Nugent, and S. Hawala. 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*, 2012.

K. M. Wolter. *Introduction to variance estimation*, volume 53. Springer, 2007.