Package ‘tipsae’

October 14, 2022

Title  Tools for Handling Indices and Proportions in Small Area Estimation

Version  0.0.7

Description  It allows for mapping proportions and indicators defined on the unit interval. It implements Beta-based small area methods comprising the classical Beta regression models, the Flexible Beta model and Zero and/or One Inflated extensions (Janki 2020 <doi:10.1080/03610926.2019.1570266>). Such methods, developed within a Bayesian framework through Stan <https://mc-stan.org/>, come equipped with a set of diagnostics and complementary tools, visualizing and exporting functions. A Shiny application with a user-friendly interface can be launched to further simplify the process.

License  GPL-3

Encoding  UTF-8

LazyData  true

RoxygenNote  7.2.1

VignetteBuilder  R.rsp

RdMacros  Rdpack

Biarch  true

NeedsCompilation  yes

Depends  R (>= 3.5.0), shiny (>= 1.0.3)

Imports  methods, Rcpp (>= 0.12.0), broom, rstan (>= 2.18.1), ggplot2 (>= 3.3.2), nlme (>= 3.1.152), stats, ggpubr, Rdpack

Suggests  rstantools (>= 2.1.1), RcppParallel (>= 5.0.1), callr, spam, spdep, maptools (>= 1.0.2), gridExtra (>= 2.3), R.rsp, rgeos, shinythemes, shinyFeedback, shinybusy, shinyWidgets, shinyjs, bayesplot, sp, loo (>= 2.3.1)

LinkingTo  BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.18.1), StanHeaders (>= 2.18.0)

SystemRequirements  GNU make

Author  Silvia De Nicolò [aut, cre] (<https://orcid.org/0000-0001-5052-6527>), Aldo Gardini [aut] (<https://orcid.org/0000-0002-2164-5815>)
The `tipsae` Package.

Description

It provides tools for mapping proportions and indicators defined on the unit interval, widely used to measure, for instance, unemployment, educational attainment and also disease prevalence. It implements Beta-based small area methods, particularly indicated for unit interval responses, comprising the classical Beta regression models, the Flexible Beta model and Zero and/or One Inflated extensions. Such methods, developed within a Bayesian framework, come equipped with a set of diagnostics and complementary tools, visualizing and exporting functions. A customized parallel computing is built-in to reduce the computational time. 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 simplify the process.

Author(s)

Silvia De Nicolò, <silvia.denicolo@unibo.it>
Aldo Gardini, <aldo.gardini@unibo.it>
References


---

benchmark

Benchmarking Procedure for Model-Based Estimates

Description

The `benchmark()` function gives the chance to perform a benchmarking procedure on model-based estimates. Benchmarking could target solely the point estimates (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.

Usage

```r
benchmark(x = , bench = , share = )
```
method = c("raking", "ratio", "double"),
H = NULL,
time = NULL,
areas = NULL
)

Arguments

x
Object of class summary_fitsae.

bench
A numeric value denoting the benchmark for the whole set of areas or a subset of areas.

share
A numeric vector of areas weights, in case of proportions it denotes the population shares.

method
The method to be specified among "raking", "ratio" and "double", see details.

H
A numeric value denoting an additional benchmark, to be specified when the "double" method is selected, corresponding to the ensemble variability.

time
A character string indicating the time period to be considered, in case of temporal models, where a benchmark can be specified only for one time period at a time.

areas
If NULL (default option), benchmarking is done on the whole set of areas, alternatively it can be done on a subset of them by indicating a vector containing the names of subset areas.

Details

The function allows performing three different benchmarking methods, according to the argument method.

- The "ratio" and "raking" methods provide benchmarked estimates that minimize the posterior expectation of the weighted squared error loss, see Datta et al. (2011) and tipsae vignette.
- The "double" method accounts for a further benchmark on the weighted ensemble variability, where H is a prespecified value of the estimators variability.

Value

A benchmark_fitsae object being a list of the following elements:

bench_est A vector including the benchmarked estimates for each considered domain.

post_risk A numeric value indicating an estimate of the overall posterior risk, aggregated for all areas. This value is only yielded when in-sample areas are treated and a single benchmarking is performed.

method The benchmarking method performed as selected in the input argument.

time The time considered as selected in the input argument.

areas The areas considered as selected in the input argument.
data_obj  A list containing input objects including in-sample and out-of-sample relevant quantities.

model_settings  A list summarizing all the assumptions of the model: sampling likelihood, presence of intercept, dispersion parametrization, random effects priors and possible structures.

model_estimates  Posterior summaries of target parameters for in-sample areas.

model_estimates_oos  Posterior summaries of target parameters for out-of-sample areas.

is_oos  Logical vector defining whether each domain is out-of-sample or not.

direct_est  Vector of direct estimates for in-sample areas.

References


See Also

summary.fitsae to produce the input object.

Examples

library(tipsae)

# loading toy dataset
data("emilia_cs")

# fitting a model
fit_beta <- fit_sae(formula_fixed = hcr ~ x, data = emilia_cs, domains = "id",
        type_disp = "var", disp_direct = "vars", domain_size = "n",
        # MCMC setting to obtain a fast example. Remove next line for reliable results.
        chains = 1, iter = 300, seed = 0)

# check model diagnostics
summ_beta <- summary(fit_beta)

# creating a subset of the areas whose estimates have to be benchmarked
subset <- c("RIMINI", "RICCIONE", "RUBICONE", "CESENA - VALLE DEL SAVIO")

# creating population shares of the subset areas
pop <- emilia_cs$pop[emilia_cs$id %in% subset]
shares_subset <- pop / sum(pop)

# perform benchmarking procedure
bmk_subset <- benchmark(x = summ_beta,
        bench = 0.13,
        share = shares_subset,
        method = "raking",
        areas = subset)

# check benchmarked estimates and posterior risk
bmk_subset$bench_est
Density Plot Function for a summary_fitsae Object

Description
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.

Usage
## S3 method for class 'summary_fitsae'
density(x, grid = TRUE, ...)

Arguments
- x: Object of class summary_fitsae.
- grid: Logical indicating whether plots are displayed in a grid (TRUE) or in sequence (FALSE).
- ...: Currently unused.

Value
Two ggplot2 objects in a grid or in sequence.

See Also
summary.fitsae to produce the input object.

Examples
library(tipsae)

# loading toy dataset
data("emilia_cs")

# fitting a model
fit_beta <- fit_sae(formula_fixed = hcr ~ x, data = emilia_cs, domains = "id",
                   type_disp = "var", disp_direct = "vars", domain_size = "n",
                   # MCMC setting to obtain a fast example. Remove next line for reliable results.
                   # MCMC setting to obtain a fast example. Remove next line for reliable results.
emilia

chairs = 1, iter = 300, seed = 0)

# check model diagnostics
summ_beta <- summary(fit_beta)

# visualize estimates and random effect densities via density() function
density(summ_beta)

---

emilia Poverty in Emilia-Romagna (Italy) Health Districts

Description

The emilia dataset 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.

Usage

emilia

Format

Dataframe with 190 observations and 8 variables.

id Character, name of the health district.
prov Character, name of NUTS-3 region related to the district.
year Numeric, year of the observation.
hcr Numeric, head-count ratio estimate (used as response variable).
vars Numeric, sampling variance of head-count ratio estimator.
n Numeric, area sample size.
x Numeric, fake covariate.
pop Numeric, population size of the area.

Details

It has been built starting from model-based estimates and related CV freely available on Emilia-Romagna region [website](#). Since it is used for illustrative purposes only, such estimates are assumed to be unreliable direct estimates, requiring a SAE procedure.

Examples

library(tipsae)
data("emilia")
emilia_cs

Poverty in Emilia-Romagna (Italy) Health Districts in 2016

Description

The emilia dataset consists of a dataset on poverty mapping concerning 38 health districts within the Emilia-Romagna region, located in North-East of Italy, with observations recorded in 2016.

Usage

emilia_cs

Format

Dataframe with 38 area observations and 8 variables.

- **id**: Character, name of the health district.
- **prov**: Character, name of NUTS-3 region related to the district.
- **year**: Numeric, year of the observation.
- **hcr**: Numeric, head-count ratio estimate (used as response variable).
- **vars**: Numeric, sampling variance of head-count ratio estimator.
- **n**: Numeric, area sample size.
- **x**: Numeric, fake covariate.
- **pop**: Numeric, population size of the area.

Details

It has been built starting from model-based estimates and related CV freely available on Emilia-Romagna region website. Since it is used for illustrative purposes only, such estimates are assumed to be unreliable direct estimates, requiring a SAE procedure.

See Also

emilia for the panel dataset including observation from 2014 to 2018.

Examples

```r
library(tipsae)
data("emilia_cs")
```
emilia_shp

Shapefile of Emilia-Romagna (Italy) Health Districts

Description

The emilia_shp shapefile consists of a SpatialPolygonsDataFrame object of 38 health districts within the Emilia-Romagna region, located in the North-East of Italy.

Usage

emilia_shp

Format

A shapefile of class SpatialPolygonsDataFrame.

COD_DIS_SA  Code of the health district.
NAME_DISTRICT  Name of the health district. It can be linked to the variable id in emilia and emilia_cs

See Also

emilia and emilia_cs for the provided datasets.

Examples

library(tipsae)
data("emilia_shp")

export

Exporting Results of a Small Area Model Fitting

Description

The function export() allows for exporting model estimates in CSV format.

Usage

export(x, file, type = "all", ...)

Arguments

x An object of class estimates_fitsae.

file A character string indicating the path (if different from the working directory) and filename of the CSV to be created. It should end with .csv.

type An option between "in", "out" and "all", indicating whether to export only in or out-of-sample areas or both.

... Additional arguments of write.csv function from utils package can be indicated.

Value

A CSV file is created in the working directory, or at the given path, exporting the estimates_fitsae object given as input.

See Also

extract to produce the input object and write.csv.

Examples

```r
## Not run:
library(tipsae)

data("emilia_cs")

# fitting a model
fit_beta <- fit_sae(formula_fixed = hcr ~ x, data = emilia_cs, domains = "id",
                     type_disp = "var", disp_direct = "vars", domain_size = "n",
                     seed = 0)

# check model diagnostics
summ_beta <- summary(fit_beta)

# extract model estimates
HB_estimates <- extract(summ_beta)

# export model estimates
export(HB_estimates, file = "results.csv", type = "all")

## End(Not run)
```
extract

Extract Posterior Summaries of Target Parameters

Description

The extract() function provides the posterior summaries of target parameters, including model-based estimates, and possibly benchmarked estimates, related to a fitted small area model.

Usage

extract(x)

Arguments

x

An object of class summary_fitsae or benchmark_fitsae.

Value

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 target parameters. When the input is a benchmark_fitsae object, benchmarked estimates are also included.

See Also

summary.fitsae and benchmark to produce the input object.

Examples

library(tipsae)

# loading toy dataset
data("emilia_cs")

# fitting a model
fit_beta <- fit_sae(formula_fixed = hcr ~ x, data = emilia_cs, domains = "id", type_disp = "var", disp_direct = "vars", domain_size = "n",
    # MCMC setting to obtain a fast example. Remove next line for reliable results.
    chains = 1, iter = 300, seed = 0)

# check model diagnostics
summ_beta <- summary(fit_beta)

# extract model estimates
HB_estimates <- extract(summ_beta)
head(HB_estimates)
Description

fit_sae() is used to fit Beta-based small area models, such as the classical Beta, zero and/or one inflated Beta and Flexible Beta models. The random effect part can incorporate either a temporal and/or a spatial dependency structure 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 and different parametrizations can be set up.

Usage

```r
fit_sae(
  formula_fixed,
  data,
  domains = NULL,
  disp_direct,
  type_disp = c("neff", "var"),
  domain_size = NULL,
  likelihood = c("beta", "flexbeta", "Infbeta0", "Infbeta1", "Infbeta01"),
  prior_reff = c("normal", "t", "VG"),
  spatial_error = FALSE,
  spatial_df = NULL,
  temporal_error = FALSE,
  temporal_variable = NULL,
  adapt_delta = 0.95,
  max_treedepth = 10,
  init = "0",
  ...
)
```

Arguments

- `formula_fixed`: An object of class "formula" specifying the linear regression fixed part at the linking level.
- `data`: An object of class "data.frame" containing all relevant quantities.
- `domains`: Data column name displaying the domain names. If NULL (default), the domains are denoted with a progressive number.
- `disp_direct`: Data column name displaying given values of sampling dispersion for each domain. In out-of-sample areas, dispersion must be NA.
- `type_disp`: Parametrization of the dispersion parameter. The choices are variance ("var") or \( \phi_d + 1 \) ("neff") parameter.
- `domain_size`: Data column name indicating domain sizes (optional). In out-of-sample areas, sizes must be NA.
fit_sae

likelihood Sampling likelihood to be used. The choices are "beta" (default), "flexbeta", "Infbeta0", "Infbeta1" and "Infbeta01".

prior_reff Prior distribution of the unstructured random effect. The choices are: "normal", "t", "VG".

spatial_error Logical indicating whether to include a spatially structured random effect.

spatial_df Object of class SpatialPolygonsDataframe with the shapefile of the studied region. Required if spatial_error = TRUE.

temporal_error Logical indicating whether to include a temporally structured random effect.

temporal_variable Data column name indicating temporal variable. Required if temporal_error = TRUE.

adapt_delta HMC option: target average proposal acceptance probability. See stan documentation.

max_treedepth HMC option: target average proposal acceptance probability. See stan documentation.

init Initial values specification. See the detailed documentation for the init argument in stan.

... Arguments passed to sampling (e.g. iter, chains).

Value

A list of class fitsae containing the following objects:

model_settings A list summarizing all the assumptions of the model: sampling likelihood, presence of intercept, dispersion parametrization, random effects priors and possible structures.

data_obj A list containing input objects including in-sample and out-of-sample relevant quantities.

stanfit A stanfit object, outcome of sampling function containing full posterior draws. For details, see stan documentation.

pars_interest A vector containing the names of parameters whose posterior samples are stored.

call Image of the function call that produced the fitsae object.

References


See Also

`sampling` for sampler options and `summary.fitsae` for handling the output.

Examples

```r
library(tipsae)

# loading toy cross sectional dataset
data("emilia_cs")

# fitting a cross sectional model
fit_beta <- fit_sae(formula_fixed = hcr ~ x, data = emilia_cs, domains = "id",
                    type_disp = "var", disp_direct = "vars", domain_size = "n",
                    # MCMC setting to obtain a fast example. Remove next line for reliable results.
                    chains = 1, iter = 300, seed = 0)

# Spatio-temporal model: it might require time to be fitted
## Not run:
# loading toy panel dataset
data("emilia")
# loading the shapefile of the concerned areas
data("emilia_shp")

# ordering the shapefile consistently with the dataset order
emilia_shp@data <- emilia_shp@data[match(unique(emilia$id), emilia_shp@data$NAME_DISTRICT),]

# fitting a spatio-temporal model
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,
                  temporal_error = TRUE,
                  temporal_variable = "year",
                  max_treedepth = 15,
                  seed = 0)

## End(Not run)
```
Description
The `map()` function enables to plot maps containing relevant model outputs by accounting for their geographical dimension. The shapefile of the area must be provided via a `SpatialPolygonsDataFrame` object.

Usage

```r
map(
  x, 
  spatial_df, 
  spatial_id_domains, 
  match_names = NULL, 
  color_palette = c("snow2", "deepskyblue4"), 
  quantity = c("HB_est", "Direct_est", "SD"), 
  time = NULL
)
```

Arguments

- **x**: An object of class `summary_fitsae` or `benchmark_fitsae`.
- **spatial_df**: A object of class `SpatialPolygonsDataFrame` (spatial polygons object) from `sp` package, accounting for the geographical dimension of the domains.
- **spatial_id_domains**: A character string indicating the name of `spatial_df` variable containing area denominations, in order to correctly match the areas.
- **match_names**: An encoding two-columns data.frame: the first with the original data coding (domains) and the second one with corresponding `spatial_df` object labels. This argument has to be specified only if `spatial_df` object labels do not match the ones provided through the original dataset.
- **color_palette**: A vector with two color strings denoting the extreme bounds of colors range to be used.
- **quantity**: A string indicating the quantity to be mapped. When a `summary_fitsae` is given as input, it can be selected among "HB_est" (model-based estimates), "SD" (posterior standard deviations) and "Direct_est" (direct estimates). While when a `benchmark_fitsae` class object is given as input, this argument turns automatically to "Bench_est", displaying the benchmarked estimates.
- **time**: A string indicating the year of interest for the quantities to be treated, in case of temporal or spatio-temporal objects.

Value

A map `ggplot2` object with colors scaled legend.

See Also

`summary.fitsae` to produce the input object and `SpatialPolygonsDataFrame` to manage the shapefile.
Examples

library(tipsae)

# loading toy dataset
data("emilia_cs")

# fitting a model
fit_beta <- fit_sae(formula_fixed = hcr ~ x, data = emilia_cs, domains = "id",
                     type_disp = "var", disp_direct = "vars", domain_size = "n",
                     # MCMC setting to obtain a fast example. Remove next line for reliable results.
                     chains = 1, iter = 300, seed = 0)

# check model diagnostics
summ_beta <- summary(fit_beta)

# load shapefile of concerned areas
data("emilia_shp")

# plot the map using model diagnostics and areas shapefile
map(x = summ_beta,
    spatial_df = emilia_shp,
    spatial_id_domains = "NAME_DISTRICT")

---

plot.benchmark_fitsae  Plot Method for benchmark_fitsae Object

Description

The method plot() provides the boxplots of original and benchmarked estimates in comparison with the benchmark value. Note that share weights are not considered.

Usage

## S3 method for class 'benchmark_fitsae'
plot(x, ...)

Arguments

x  A benchmark_fitsae object.

...  Currently unused.

Value

A ggplot2 object.

See Also

benchmark to produce the input object.
Examples

```r
library(tipsae)

# loading toy dataset
data("emilia_cs")

# fitting a model
fit_beta <- fit_sae(formula_fixed = hcr ~ x, data = emilia_cs, domains = "id",
                    type_disp = "var", disp_direct = "vars", domain_size = "n",
                    # MCMC setting to obtain a fast example. Remove next line for reliable results.
                    chains = 1, iter = 300, seed = 0)

# check model diagnostics
summ_beta <- summary(fit_beta)

# creating a subset of the areas whose estimates have to be benchmarked
subset <- c("RIMINI", "RICCIONE", "RUBICONE", "CESENA - VALLE DEL SAVIO")

# creating population shares of the subset areas
pop <- emilia_cs$pop[emilia_cs$id %in% subset]
shares_subset <- pop / sum(pop)

# perform benchmarking procedure
bmk_subset <- benchmark(x = summ_beta,
                        bench = 0.13,
                        share = shares_subset,
                        method = "raking",
                        areas = subset)

plot(bmk_subset)
```

---

`plot.smoothing_fitsae`  *Plot Method for smoothing_fitsae Object*

**Description**

The `plot()` method provides (a) the boxplot of variance estimates, when effective sample sizes are estimated through kish method; (b) a scatterplot of both original and smoothed estimates versus the area sample sizes, when variance smoothing is performed through methods `ols` and `gls`.

**Usage**

```r
## S3 method for class 'smoothing_fitsae'
plot(x, size = 2.5, alpha = 0.8, ...)
```
plot.summary_fitsae

Arguments

x A smoothing_fitsae object.

size Aesthetic option denoting the size of scatterplots points, see geom_point documentation.

alpha Aesthetic option denoting the opacity of scatterplots points, see geom_point documentation.

... Currently unused.

Value

A ggplot2 object.

See Also

smoothing to produce the input object.

Examples

library(tipsae)

# loading toy dataset
data("emilia_cs")

# perform smoothing procedure
smoo <- smoothing(emilia_cs, direct_estimates = "hcr", area_id = "id",
                 raw_variance = "vars", areas_sample_sizes = "n",
                 var_function = NULL, method = "ols")

plot(smoo)

---

plot.summary_fitsae   Plot Method for a summary_fitsae Object

Description

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.
Usage

```r
## S3 method for class 'summary_fitsae'
plot(
  x,
  size = 2.5,
  alpha = 0.8,
  n_bins = 15,
  grid = TRUE,
  label_names = NULL,
  ...
)
```

Arguments

- `x`: Object of class `summary_fitsae`.
- `size`: Aesthetic option denoting the size of scatterplots points, see `geom_point` documentation.
- `alpha`: Aesthetic option denoting the opacity of scatterplots points, see `geom_point` documentation.
- `n_bins`: Denoting the number of bins used for histogram.
- `grid`: Logical indicating whether plots are displayed in a grid (TRUE) or in sequence (FALSE).
- `label_names`: Character string indicating the model name to display in boxplot x-axis label.
- `...`: Currently unused.

Value

Four `ggplot2` objects in a grid.

See Also

- `summary.fitsae` to produce the input object.

Examples

```r
library(tipsae)

# loading toy dataset
data("emilia_cs")

# fitting a model
fit_beta <- fit_sae(formula_fixed = hcr ~ x, data = emilia_cs, domains = "id",
  type_disp = "var", disp_direct = "vars", domain_size = "n",
  # MCMC setting to obtain a fast example. Remove next line for reliable results.
  chains = 1, iter = 300, seed = 0)

# check model diagnostics
summ_beta <- summary(fit_beta)
```
# visualize diagnostics via plot() method
plot(summ_beta)

---

runShiny_tipsae  
**Launch Shiny App to Performs Small Area Estimation**

**Description**

The command launches a Shiny application that assists the user from the data loading step to the export of the outputs. See the vignette for further details.

**Usage**

```r
runShiny_tipsae()
```

**Value**

No value returned.

**Examples**

```r
library(tipsae)

# Starting the Shiny application
if(interactive()){
  runShiny_tipsae()
}
```

---

smoothing  
**Variance Smoothing and Effective Sample Sizes Estimation**

**Description**

The `smoothing()` function implements three methods, all yielding refined estimates of either variance or effective sample size, 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. All the implemented methods enable the estimation of the effective sample sizes, whereas "ols" and "gls" also perform a variance smoothing procedure.
smoothing

Usage

smoothing(
  data,
  direct_estimates,
  area_id = NULL,
  raw_variance = NULL,
  areas_sample_sizes = NULL,
  additional_covariates = NULL,
  method = c("ols", "gls", "kish"),
  var_function = NULL,
  survey_data = NULL,
  survey_area_id = NULL,
  weights = NULL,
  sizes = NULL
)

Arguments

data    A data.frame object including the direct estimates.
direct_estimates    Character string specifying the variable in data denoting the direct estimates.
area_id    Character string indicating the variable with domain names included in data, to be specified if method "kish" is selected.
raw_variance    Character string indicating the variable name for raw variance estimates included in data object, to be specified if methods "ols" or "gls" are selected.
areas_sample_sizes    Character string indicating the variable name for domain sample sizes included in data object, to be specified if methods "ols" or "gls" are selected.
additional_covariates    A vector of character strings indicating the variable names of possible additional covariates, included in data, to be added to the smoothing procedure if methods "ols" or "gls" are selected.
method    The method to be used. The choices are "kish", "ols" and "gls".
var_function    An object of class function denoting the variance function of the response variable. The default option (NULL) matches the proportion case being equal to function(x) * (1 - x).
survey_data    An additional dataset to be specified when method "kish" is selected, defined at sampling unit level (e.g., households) and comprising sampling weights, unit sizes and domain names.
survey_area_id    Character string indicating the variable denoting the domain names included in the survey_data object.
weights    Character string indicating the variable including sampling weights in survey_data object.
sizes    Character string indicating the variable including unit sizes in survey_data object.
Value
An object of class `smoothing_fitsae`, being a list of vectors including dispersion parameters estimates: both the variances and the effective sample sizes. When "ols" or "gls" method has been selected, the list incorporates also an object of class `gls` from `nlme` package.

References

See Also
`gls` for details on estimation procedure for "ols" and "gls" methods.

Examples

```r
library(tipsae)

# loading toy dataset
data("emilia_cs")

# perform smoothing procedure
smoo <- smoothing(emilia_cs, direct_estimates = "hcr", area_id = "id",
                   raw_variance = "vars", areas_sample_sizes = "n",
                   var_function = NULL, method = "ols")

summary(smoo)
```

**summary.fitsae**  
*Summary Method for fitsae Objects*

**Description**
Summarizing the small area model fitting through the distributions of estimated parameters and derived diagnostics using posterior draws.

**Usage**
```
## S3 method for class 'fitsae'
summary(
  object,
  probs = c(0.025, 0.25, 0.5, 0.75, 0.975),
  compute_loo = TRUE,
  ...)
```
Arguments

- **object**: An instance of class `fitsae`.
- **probs**: A numeric vector of quantiles of interest. The default is `c(0.025, 0.25, 0.5, 0.75, 0.975)`.
- **compute_loo**: Logical, indicating whether to compute loo diagnostics or not.
- **...**: Currently unused.

Details

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; (c) Bayesian P-values obtained with the MCMC samples.
- Shrinking Bound Rate.
- loo information criteria and related diagnostics from the loo package.

Value

A list of class `summary_fitsae` containing diagnostics objects:

- **raneff**: A list of data.frame objects storing the random effects posterior summaries divided for each type: $unstructured, $temporal, and $spatial.
- **fixed_coeff**: Posterior summaries of fixed coefficients.
- **var_comp**: Posterior summaries of model variance parameters.
- **model_estimates**: Posterior summaries of the parameter of interest $\theta_d$ for each in-sample domain $d$.
- **model_estimates_oos**: Posterior summaries of the parameter of interest $\theta_d$ for each out-of-sample domain $d$.
- **is_oos**: Logical vector defining whether each domain is out-of-sample or not.
- **direct_est**: Vector of input direct estimates.
- **post_means**: Model-based estimates, i.e. posterior means of the parameter of interest $\theta_d$ for each domain $d$.
- **sd_reduction**: Standard deviation reduction, see details section.
- **sd_dir**: Standard deviation of direct estimates, given as input if type_disp="var".
- **loo**: The object of class loo, for details see loo package documentation.
- **shrink_rate**: Shrinking Bound Rate, see details section.
- **residuals**: Residuals related to model-based estimates.
- **bayes_pvalues**: Bayesian p-values obtained via MCMC samples, see details section.
- **y_rep**: An array with values generated from the posterior predictive distribution, enabling the implementation of posterior predictive checks.
- **diag_summ**: Summaries of residuals, standard deviation reduction and Bayesian p-values across the whole domain set.
data_obj A list containing input objects including in-sample and out-of-sample relevant quantities.

model_settings A list summarizing all the assumptions of the input model: sampling likelihood, presence of intercept, dispersion parametrization, random effects priors and possible structures.

call Image of the function call that produced the input fitsae object.

References


See Also

fit_sae to estimate the model and the generic methods plot.summary_fitsae and density.summary_fitsae, and functions map, benchmark and extract.

Examples

library(tipsae)

# loading toy dataset
data("emilia_cs")

# fitting a model
fit_beta <- fit_sae(formula_fixed = hcr ~ x, data = emilia_cs, domains = "id",
                   type_disp = "var", disp_direct = "vars", domain_size = "n",
                   # MCMC setting to obtain a fast example. Remove next line for reliable results.
                   chains = 1, iter = 300, seed = 0)

# check model diagnostics via summary() method
summ_beta <- summary(fit_beta)
summ_beta
Index

* datasets
  emilia, 7
  emilia_cs, 8
  emilia_shp, 9

benchmark, 3, 11, 16, 24

density.summary_fitsae, 6, 24

emilia, 7, 8, 9
emilia_cs, 8, 9
emilia_shp, 9
export, 9
extract, 10, 11, 24

fit_sae, 12, 24

geom_point, 18, 19
gls, 22

loo, 23

map, 14, 24

plot.benchmark_fitsae, 16
plot.smoothing_fitsae, 17
plot.summary_fitsae, 18, 24

quantile, 23

runShiny_tipsae, 20

sampling, 13, 14
smoothing, 18, 20
SpatialPolygonsDataFrame, 15
stan, 13
summary.fitsae, 5, 6, 11, 14, 15, 19, 22

tipsae (tipsae-package), 2
tipsae-package, 2

write.csv, 10