Package: povmap (via r-universe)

September 13, 2024

Title Extension to the 'emdi' Package

Version 1.0.1

Date 2024-01-18

Description The R package 'povmap' supports small area estimation of means and poverty headcount rates. It adds several new features to the 'emdi' package (see ` ` The R Package emdi for Estimating and Mapping Regionally Disaggregated Indicators'' by Kreutzmann et al. (2019) <doi:10.18637/jss.v091.i07>). These include new options for incorporating survey weights, ex-post benchmarking of estimates, two additional transformations, several new convenient functions to assist with reporting results, and a wrapper function to facilitate access from 'Stata'.

Depends R (>= 4.2.0)

License GPL-2

URL https://github.com/SSA-Statistical-Team-Projects/povmap

LazyData true

Encoding UTF-8

Copyright inst/COPYRIGHTS

RoxygenNote 7.2.3

Imports nlme, moments, ggplot2, MuMIn, gridExtra, openxlsx, reshape2, stats, stringr, parallelMap, HLMdiag, parallel, boot, MASS, readODS, formula.tools, saeRobust, spdep, bestNormalize, survey, sf

Suggests testthat, R.rsp, simFrame, laeken, graphics

VignetteBuilder R.rsp

Repository https://ssa-statistical-team-projects.r-universe.dev

RemoteUrl https://github.com/ssa-statistical-team-projects/povmap

RemoteRef HEAD

RemoteSha cdc5ffbcb27c91a06f620891ed512acb91666435

Contents

benchmark *Benchmark Function*

Description

This function benchmarks the EBLUP estimates of an area-level model.

Usage

```
benchmark(object, benchmark, share, type = "raking", overwrite = FALSE)
```

Arguments

object	an object of type "fh".
benchmark	a number determining the benchmark value.
share	a vector containing the shares of the population size per area and the total population size (N_d/N).Values must be sorted like the domains in the fh object.
type	Character indicating the type of benchmarking. Types that can be chosen (i) Raking ("raking"), (ii) Ratio adjustment ("ratio"), (iii) MSE adjustment ("MSE_adj"). Defaults to "raking".
overwrite	if TRUE, the benchmarked FH estimates are added to the ind object of the emdi object and the MSE estimates are set to NULL since these are not benchmarked. Defaults to FALSE.

Details

The benchmarking algorithm only works, if FH estimates are available. The type "MSE_adj" only works, if MSE estimates are available. If overwrite is set to TRUE, the emdi object is returned, but the benchmarked FH estimates are added to the ind object of the emdi object and the MSE estimates are set to NULL since these are not benchmarked.

Value

A data frame containing a domain indicator (Domain), direct estimates (Direct), point predictions (FH), benchmarked point predictions (FH_Bench) and a variable indicating out-of-sample domains Out (1 for out-of-sample, 0 for in-sample) . If overwrite is set to TRUE, the fh object is returned, but the point predictions of the ind data frame are complemented by the benchmarked results.

References

Datta,G. S., Ghosh, M., Steorts, R. and Maples, J. (2010) Bayesian benchmarking with applications to small area estimation. Test, 20, 574–588.

Examples

```
# Loading data - population and sample data
data("eusilcA_popAgg")
data("eusilcA_smpAgg")

# Combine sample and population data
combined_data <- combine_data(
  pop_data = eusilcA_popAgg,
  pop_domains = "Domain",
  smp_data = eusilcA_smpAgg,
  smp_domains = "Domain"
)

# Estimate Fay-Herriot model
fh_std <- fh(
  fixed = Mean ~ cash + self_empl, vardir = "Var_Mean",
  combined_data = combined_data, domains = "Domain", method = "ml",
  MSE = TRUE
)

# Benchmark the point estimates

# Example 1: Receive data frame with point estimates and their benchmarked
# results
fh_bench <- benchmark(fh_std,
  benchmark = 20140.09,
  share = eusilcA_popAgg$ratio_n, type = "ratio"
)

# Example 2: Add benchmarked results to fh object
fh_bench <- benchmark(fh_std,
  benchmark = 20140.09,
  share = eusilcA_popAgg$ratio_n, type = "ratio", overwrite = TRUE
)
```

`combine_data` *Combines Sample and Population Data*

Description

This function combines the aggregated population information with the aggregated sample data. The merge is based on the domains. Out-of-sample domains will have NA values for the variables from the sample data.

Usage

```
combine_data(pop_data, pop_domains, smp_data, smp_domains)
```

Arguments

pop_data	a data frame with population data.
pop_domains	a character string indicating the domain variable that is included in pop_data.
smp_data	a data frame with sample data.
smp_domains	a character string indicating the domain variable that is included in smp_data.

Value

a combined data set.

---

compare *Compare Function*

Description

Function compare is a generic function used to assess the quality of the model-based estimates by comparing them with the direct estimates.

Method compare.fh assesses the quality of the model-based estimates of the Fay-Herriot model by comparing them with the direct estimates based on a goodness-of-fit test proposed by *Brown et al. (2001)* and by computing the correlation between the regression-synthetic part of the Fay-Herriot model and the direct estimates.

Usage

```
compare(object, ...)

## S3 method for class 'fh'
compare(object, ...)
```

Arguments

object	an object of type "fh".
...	further arguments passed to or from other methods.

Value

The return of compare depends on the class of its argument. The documentation of particular methods gives detailed information about the return of that method.

For the method for class "fh", the null hypothesis, the value W of the test statistic, the degrees of freedom and the p value of the Brown test; and the correlation coefficient of the synthetic part and the direct estimator *(Chandra et al. 2015)* are returned.

References

Brown, G., R. Chambers, P. Heady, and D. Heasman (2001). Evaluation of small area estimation methods: An application to unemployment estimates from the UK LFS. Symposium 2001 - Achieving Data Quality in a Statistical Agency: A Methodological Perspective, Statistics Canada.

Chandra, H., Salvati, N. and Chambers, R. (2015), A Spatially Nonstationary Fay-Herriot Model for Small Area Estimation, Journal of the Survey Statistics and Methodology, 3, 109-135.

`compare_plot` *Shows Plots for the Comparison of Estimates*

Description

Function `compare_plot` is a generic function used to produce plots comparing point and existing MSE/CV estimates of direct and model-based estimation for all indicators or a selection of indicators.

Methods `compare_plot.direct`, `compare_plot.ebp` and `compare_plot.fh` produce plots comparing point and existing MSE/CV estimates of direct and model-based estimation for all indicators or a selection of indicators for objects of type "emdi". The direct and model-based point estimates are compared by a scatter plot and a line plot for each selected indicator. If the input arguments MSE and CV are set to TRUE, two extra plots are created, respectively: the MSE/CV estimates of the direct and model-based estimates are compared by boxplots and scatter plots.

Usage

```
compare_plot(
  model,
  direct,
  indicator = "all",
  MSE = FALSE,
  CV = FALSE,
  label = "orig",
  color = c("blue", "lightblue3"),
  shape = c(16, 16),
  line_type = c("solid", "solid"),
  gg_theme = NULL,
  ...
)

## S3 method for class 'direct'
compare_plot(
  model = NULL,
  direct = NULL,
  indicator = "all",
  MSE = FALSE,
  CV = FALSE,
```

```
  label = "orig",
  color = c("blue", "lightblue3"),
  shape = c(16, 16),
  line_type = c("solid", "solid"),
  gg_theme = NULL,
  ...
)

## S3 method for class 'ebp'
compare_plot(
  model = NULL,
  direct = NULL,
  indicator = "all",
  MSE = FALSE,
  CV = FALSE,
  label = "orig",
  color = c("blue", "lightblue3"),
  shape = c(16, 16),
  line_type = c("solid", "solid"),
  gg_theme = NULL,
  ...
)

## S3 method for class 'fh'
compare_plot(
  model = NULL,
  direct = NULL,
  indicator = "all",
  MSE = FALSE,
  CV = FALSE,
  label = "orig",
  color = c("blue", "lightblue3"),
  shape = c(16, 16),
  line_type = c("solid", "solid"),
  gg_theme = NULL,
  ...
)
```

Arguments

`model`	a model object of type "emdi", either "ebp" or "fh", representing point and MSE estimates.
`direct`	an object of type "direct","emdi", representing point and MSE estimates. If the input argument `model` is of type "ebp", `direct` is required. If the input argument `model` is of type "fh", the `direct` component is already included in the input argument `model`.
`indicator`	optional character vector that selects which indicators shall be returned: (i) all calculated indicators ("all"); (ii) each indicator name: "Mean", "Quantile_10",

"Quantile_25", "Median", "Quantile_75", "Quantile_90", "Head_Count", "Poverty_Gap", "Gini", "Quintile_Share" or the function name/s of "custom_indicator/s"; (iii) groups of indicators: "Quantiles", "Poverty", "Inequality" or "Custom". If two of these groups are selected, only the first one is returned. Note, additional custom indicators can be defined as argument for the EBP approaches (see also ebp) and do not appear in groups of indicators even though these might belong to one of the groups. If the model argument is of type "fh", indicator can be set to "all", "Direct", FH", or "FH_Bench" (if emdi object is overwritten by function benchmark). Defaults to "all".

MSE — optional logical. If TRUE, the MSE estimates of the direct and model-based estimates are compared via boxplots and scatter plots.

CV — optional logical. If TRUE, the coefficient of variation estimates of the direct and model-based estimates are compared via boxplots and scatter plots.

label — argument that enables to customize title and axis labels. There are three options to label the evaluation plots: (i) original labels ("orig"), (ii) axis labels but no title ("no_title"), (iii) neither axis labels nor title ("blank").

color — a vector with two elements. The first color determines the color for the regression line in the scatter plot and the color for the direct estimates in the remaining plots. The second color specifies the color of the intersection line in the scatter plot and the color for the model-based estimates in the remaining plots. Defaults to c("blue", "lightblue3").

shape — a numeric vector with two elements. The first shape determines the shape of the points in the scatterplot and the shape of the points for the direct estimates in the remaining plots. The second shape determines the shape for the points for the model-based estimates. The options are numbered from 0 to 25. Defaults to c(16, 16).

line_type — a character vector with two elements. The first line type determines the line type for the regression line in the scatter plot and the line type for the direct estimates in the remaining plots. The second line type specifies the line type of the intersection line in the scatter plot and the line type for the model-based estimates in the remaining plots. The options are: "twodash", "solid", "longdash", "dotted", "dotdash", "dashed" and "blank". Defaults to c("solid", "solid").

gg_theme — theme list from package **ggplot2**. For using this argument, package **ggplot2** must be loaded via library(ggplot2). See also Example 2.

... — further arguments passed to or from other methods.

Details

Since all of the comparisons need a direct estimator, the plots are only created for in-sample domains.

Value

Plots comparing direct and model-based estimators for each selected indicator obtained by ggplot.

A scatter plot and a line plot comparing direct and model-based estimators for each selected indicator obtained by ggplot. If the input arguments MSE and CV are set to TRUE two extra plots are

created, respectively: the MSE/CV estimates of the direct and model-based estimates are compared by boxplots and scatter plots.

See Also

emdiObject, direct, ebp, fh

Examples

```
# Examples for comparisons of direct estimates and models of type ebp

# Loading data - population and sample data
data("eusilcA_pop")
data("eusilcA_smp")

# Generation of two emdi objects
emdi_model <- ebp(
  fixed = eqIncome ~ gender + eqsize + cash +
    self_empl + unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent +
    fam_allow + house_allow + cap_inv + tax_adj, pop_data = eusilcA_pop,
  pop_domains = "district", smp_data = eusilcA_smp, smp_domains = "district",
  threshold = function(y) {
    0.6 * median(y)
  }, L = 50, MSE = TRUE,
  na.rm = TRUE, cpus = 1
)

emdi_direct <- direct(
  y = "eqIncome", smp_data = eusilcA_smp,
  smp_domains = "district", weights = "weight", threshold = 11161.44,
  var = TRUE, boot_type = "naive", B = 50, seed = 123, na.rm = TRUE
)

# Example 1: Receive first overview
compare_plot(model = emdi_model, direct = emdi_direct)

# Example 2: Change plot theme
library(ggplot2)
compare_plot(emdi_model, emdi_direct,
  indicator = "Median",
  gg_theme = theme(
    axis.line = element_line(size = 3, colour = "grey80"),
    plot.background = element_rect(fill = "lightblue3"),
    legend.position = "none"
  )
)

# Example for comparison of direct estimates and models of type fh

# Loading data - population and sample data
data("eusilcA_popAgg")
data("eusilcA_smpAgg")
```

```
# Combine sample and population data
combined_data <- combine_data(
  pop_data = eusilcA_popAgg,
  pop_domains = "Domain",
  smp_data = eusilcA_smpAgg,
  smp_domains = "Domain"
)

# Generation of the emdi object
fh_std <- fh(
  fixed = Mean ~ cash + self_empl, vardir = "Var_Mean",
  combined_data = combined_data, domains = "Domain",
  method = "ml", MSE = TRUE
)
# Example 3: Receive first overview
compare_plot(fh_std)

# Example 4: Compare also MSE and CV estimates
compare_plot(fh_std, MSE = TRUE, CV = TRUE)
```

`compare_pred` *Compare Predictions of Model Objects*

Description

Function `compare_pred` is a generic function used to compare predictions of two model objects.

Method `compare_pred.emdi` compares predictions of two emdi objects.

Usage

```
compare_pred(object1, object2, MSE = FALSE, ...)

## S3 method for class 'emdi'
compare_pred(object1, object2, MSE = FALSE, ...)
```

Arguments

`object1`	an object of type "emdi".
`object2`	an object of type "emdi".
`MSE`	if `TRUE`, MSE estimates are also returned. Defaults to `FALSE`.
`...`	further arguments passed to or from other methods.

Value

Data frame containing the point estimates of both emdi objects. If column names are duplicated, the suffixes "_1" and "_2" are added to their names. "_1" and "_2" standing for object1 and object2, respectively. If `MSE` is set to `TRUE`, the data frame also contains the MSE estimates of the emdi objects.

See Also

direct, ebp, fh

Examples

```
# Example for class ebp
emdi_model_1 <- ebp(
  fixed = eqIncome ~ gender + eqsize + cash + self_empl +
    unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent + fam_allow +
    house_allow + cap_inv + tax_adj, pop_data = eusilcA_pop,
  pop_domains = "district", smp_data = eusilcA_smp, smp_domains = "district",
  na.rm = TRUE
)

emdi_model_2 <- ebp(
  fixed = eqIncome ~ gender + eqsize + cash + self_empl +
    unempl_ben + age_ben + surv_ben, pop_data = eusilcA_pop,
  pop_domains = "district", smp_data = eusilcA_smp, smp_domains = "district",
  na.rm = TRUE
)

compare_pred(emdi_model_1, emdi_model_2)
```

---

data_transformation *Tranforms Dependent Variables*

Description

Function data_transformation transforms the dependent variable from the formula object fixed in the given sample data set. Thus, it returns the original sample data set with transformed dependent variable. For the transformation five types can be chosen, particularly no, natural log, Box-Cox, Dual and Log-Shift transformation.

Usage

```
data_transformation(fixed, smp_data, transformation, lambda)
```

Arguments

fixed	a two-sided linear formula object describing the fixed-effects part of the nested error linear regression model with the dependent variable on the left of a ~ operator and the explanatory variables on the right, separated by + operators. The argument corresponds to the argument fixed in function lme.
smp_data	a data frame that needs to comprise all variables named in fixed. If transformed data is further used to fit a nested error linear regression model, smp_data also needs to comprise the variable named in smp_domains (see ebp).

`transformation` a character string. Five different transformation methods for the dependent variable can be chosen (i) no transformation ("no"); (ii) natural log transformation ("log"); (iii) Box-Cox transformation ("box.cox"); (iv) Dual transformation ("dual"); (v) Log-Shift transformation ("log.shift")..

`lambda` a scalar parameter that determines the transformations with transformation parameter. In case of no and natural log transformation `lambda` can be set to NULL.

Details

For the natural log, Box-Cox and Dual transformation, the dependent variable is shifted such that all values are greater than zero since the transformations are not applicable for values equal to or smaller than zero. The shift is calculated as follows:

$$shift = |min(y)| + 1 \qquad if \qquad min(y) <= 0$$

Function `data_transformation` works as a wrapper function. This means that the function manages the selection of the three different transformation functions `no_transform`, `log_transform` and `box_cox`.

Value

a named list with two elements, a data frame containing the data set with transformed dependent variable (`transformed_data`) and a shift parameter `shift` if present. In case of no transformation, the original data frame is returned and the shift parameter is NULL.

See Also

`lme`

Examples

```
# Loading data - sample data
data("eusilcA_smp")

# Transform dependent variable in sample data with Box-Cox transformation
transform_data <- data_transformation(eqIncome ~ gender + eqsize + cash +
  self_empl + unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent +
  fam_allow + house_allow + cap_inv + tax_adj, eusilcA_smp, "box.cox", 0.7)
```

---

direct *Direct estimation of disaggregated indicators*

Description

Function `direct` estimates indicators only based on sample information. The variance is estimated via a naive or calibrated bootstrap. The estimation is adapted from the estimation of direct indicators in package **laeken**.

Usage

```
direct(
  y,
  smp_data,
  smp_domains,
  weights = NULL,
  design = NULL,
  threshold = NULL,
  var = FALSE,
  boot_type = "naive",
  B = 50,
  seed = 123,
  X_calib = NULL,
  totals = NULL,
  custom_indicator = NULL,
  na.rm = FALSE,
  HT = FALSE
)
```

Arguments

y	a character string indicating the variable that is used for estimating the indicators. The variable must be contained in the sample data.
smp_data	survey data containing variable y as well as sampling domains, and weights if selected.
smp_domains	a character containing the name of a variable that indicates domains in the sample data. The variable must be numeric or a factor.
weights	a character string containing the name of a variable for the sampling weights in the sample data. This argument is optional and defaults to NULL.
design	a character string containing the name of a variable for different strata for stratified sampling designs. This argument is optional and defaults to NULL.
threshold	a number defining a threshold. Alternatively, a threshold may be defined as a function of y and weights returning a numeric value. Such a function will be evaluated once for the point estimation and in each iteration of the parametric bootstrap. See Example 2 for using a function as threshold. A threshold is needed for calculation e.g. of head count ratios and poverty gaps. The argument defaults to NULL. In this case, the threshold is set to 60% of the median of the variable that is selected as y similarly to the at-risk-of-poverty rate used in the EU (see also *Social Protection Committee 2001*). However, any desired threshold can be chosen.
var	if TRUE, estimates for the variance are calculated using a naive or calibrated bootstrap. Defaults to FALSE.
boot_type	a character string containing the name of the bootstrap specification. Either a "naive" or a "calibrate" bootstrap can be used. See also bootVar. Defaults to naive.

B	a number determining the number of bootstrap populations for the bootstrap variance. Defaults to 50.
seed	an integer to set the seed for the random number generator. Random number generation is used in the bootstrap approach. If seed is set to NULL, seed is chosen randomly. Defaults to 123.
X_calib	a numeric matrix including calibration variables if the calibrated bootstrap is chosen. Defaults to NULL.
totals	a numeric vector providing the population totals if the calibrated bootstrap is chosen. If a vector is chosen, the length of the vector needs to equal the number of columns in X_calib. Defaults to NULL. In this case, the sampling weights are used to calculate the totals.
custom_indicator	a list of functions containing the indicators to be calculated additionally. Such functions must and must only depend on the target variable y, the weights and the threshold (numeric value) (see Example 3) even though some arguments might not be used in the additional function. Defaults to NULL.
na.rm	if TRUE, observations with NA values are deleted from the sample data. Defaults to FALSE.
HT	if TRUE use Horvitz Thompson estimator.

Details

The set of predefined indicators includes the mean, median, four further quantiles (10%, 25%, 75% and 90%), head count ratio, poverty gap, Gini coefficient and the quintile share ratio.

Value

An object of class "direct", "emdi" that provides direct estimators for regional disaggregated indicators and optionally corresponding variance estimates. Several generic functions have methods for the returned object. For a full list and descriptions of the components of objects of class "emdi", see emdiObject.

References

Alfons, A. and Templ, M. (2013). Estimation of Social Exclusion Indicators from Complex Surveys: The R Package **laeken**. Journal of Statistical Software, 54(15), 1-25.

Social Protection Committee (2001). Report on Indicators in the Field of Poverty and Social Exclusions, Technical Report, European Union.

See Also

emdiObject, lme, estimators.emdi, emdi_summaries

Examples

```
# Loading sample data
data("eusilcA_smp")

# Example 1: With weights and naive bootstrap
emdi_direct <- direct(y = "eqIncome", smp_data = eusilcA_smp,
smp_domains = "district", weights = "weight", threshold = 11064.82, var = TRUE,
boot_type = "naive", B = 50, seed = 123, X_calib = NULL, totals = NULL,
na.rm = TRUE)

# Example 2: With function as threshold
emdi_direct <- direct(y = "eqIncome", smp_data = eusilcA_smp,
smp_domains = "district", weights = "weight", threshold =
function(y, weights){0.6 * wtd.quantile(y, weights, 0.5)}, na.rm = TRUE)

# Example 3: With custom indicators
emdi_direct <- direct(y = "eqIncome", smp_data = eusilcA_smp,
smp_domains = "district", weights = "weight", threshold = 10859.24,
var = TRUE, boot_type = "naive", B = 50, seed = 123, X_calib = NULL,
totals = NULL, custom_indicator = list(my_max = function(y, weights,
threshold){max(y)}, my_min = function(y, weights, threshold){min(y)}),
na.rm = TRUE)
```

---

ebp *Empirical Best Prediction for Disaggregated Indicators*

Description

Function ebp estimates indicators using the Empirical Best Prediction approach by *Molina and Rao (2010)*. Point predictions of indicators are obtained by Monte-Carlo approximations. Additionally, mean squared error (MSE) estimation can be conducted by using a parametric bootstrap approach (see also *Gonzalez-Manteiga et al. (2008)*). The unit-level model of *Battese, Harter and Fuller (1988)* is fitted by the restricted maximum likelihood (REML) method and one of five different transformation types for the dependent variable can be chosen. This approach can be extended to data under informative sampling using weights and is based on *Guadarrama et al. (2018)*. Model estimation combines the uni-level model of *Battese, Harter and Fuller (1988)* and the approach of *You and Rao (2002)* using survey weights. At the moment, only the log-transformation is supported for this method.

Usage

```
ebp(
  fixed,
  pop_data,
  pop_domains,
  smp_data,
  smp_domains,
```

```
  L = 50,
  threshold = NULL,
  transformation = "box.cox",
  interval = "default",
  MSE = FALSE,
  B = 50,
  seed = 123,
  boot_type = "parametric",
 parallel_mode = ifelse(grepl("windows", .Platform$OS.type), "socket", "multicore"),
  cpus = 1,
  custom_indicator = NULL,
  na.rm = FALSE,
  weights = NULL,
  pop_weights = NULL,
  aggregate_to = NULL,
  weights_type = "Guadarrama",
  benchmark = NULL,
  benchmark_type = "ratio",
  benchmark_level = NULL,
  benchmark_weights = NULL,
  nlme_maxiter = 1000,
  nlme_tolerance = 1e-06,
  nlme_opt = "nlminb",
  nlme_optimmethod = "BFGS",
  nlme_method = "REML",
  nlme_mstol = 1e-07,
  nlme_msmaxiter = 1000,
  nlme_returnobject = FALSE,
  rescale_weights = FALSE,
  Ydump = NULL
)
```

Arguments

`fixed`	a two-sided linear formula object describing the fixed-effects part of the nested error linear regression model with the dependent variable on the left of a ~ operator and the explanatory variables on the right, separated by + operators. The argument corresponds to the argument `fixed` in function `lme`.
`pop_data`	a data frame that needs to comprise the variables named on the right of the ~ operator in `fixed`, i.e. the explanatory variables, and `pop_domains`.
`pop_domains`	a character string containing the name of a variable that indicates domains in the population data. The variable can be numeric or a factor but needs to be of the same class as the variable named in `smp_domains`.
`smp_data`	a data frame that needs to comprise all variables named in `fixed` and `smp_domains`.
`smp_domains`	a character string containing the name of a variable that indicates domains in the sample data. The variable can be numeric or a factor but needs to be of the same class as the variable named in `pop_domains`.

`L` a number determining the number of Monte-Carlo simulations that must be at least 1. Defaults to 50. For practical applications, values larger than 200 are recommended (see also *Molina, I. and Rao, J.N.K. (2010)*).

`threshold` a number defining a threshold. Alternatively, a threshold may be defined as a `function` of `y` returning a numeric value. Such a function will be evaluated once for the point estimation and in each iteration of the parametric bootstrap. A threshold is needed for calculation e.g. of head count ratios and poverty gaps. The argument defaults to `NULL`. In this case, the threshold is set to 60% of the median of the variable that is selected as dependent variable similary to the at-risk-of-poverty rate used in the EU (see also *Social Protection Committee 2001*). However, any desired threshold can be chosen.

`transformation` a character string. Five different transformation types for the dependent variable can be chosen (i) no transformation ("no"); (ii) log transformation ("log"); (iii) Box-Cox transformation ("box.cox"); (iv) Dual transformation ("dual"); (v) Log-Shift transformation ("log.shift"); (Vi) rank-order transformation ("ordernorm"). Defaults to `"box.cox"`.

`interval` a string equal to 'default' or a numeric vector containing a lower and upper limit determining an interval for the estimation of the optimal parameter. The interval is passed to function `optimize` for the optimization. Defaults to 'default' which equals c(-1,2) for Box-Cox, c(0,2) for Dual and an interval based on the range of y for Log-Shift transformation. If the convergence fails, it is often advisable to choose a smaller more suitable interval. For right skewed distributions, the negative values may be excluded, also values larger than 1 are seldom observed.

`MSE` if `TRUE`, MSE estimates using a parametric bootstrap approach are calculated (see also *Gonzalez-Manteiga et al. (2008)*). Defaults to `FALSE`.

`B` a number determining the number of bootstrap populations in the parametric bootstrap approach (see also *Gonzalez-Manteiga et al. (2008)*) used in the MSE estimation. The number must be greater than 1. Defaults to 50. For practical applications, values larger than 200 are recommended (see also *Molina, I. and Rao, J.N.K. (2010)*).

`seed` an integer to set the seed for the random number generator. For the usage of random number generation, see Details. If seed is set to `NULL`, seed is chosen randomly. Defaults to `123`.

`boot_type` character string to choose between different MSE estimation procedures,currently a `"parametric"` and a semi-parametric `"wild"` bootstrap are possible. Defaults to `"parametric"`.

`parallel_mode` modus of parallelization, defaults to an automatic selection of a suitable mode, depending on the operating system, if the number of `cpus` is chosen higher than 1. For details, see `parallelStart`.

`cpus` number determining the kernels that are used for the parallelization. Defaults to 1. For details, see `parallelStart`.

`custom_indicator`
a list of functions containing the indicators to be calculated additionally. Such functions must depend on the target variable `y`, and optional can depend on `pop_weights` and the `threshold`. Defaults to `NULL`.

`na.rm` if TRUE, observations with NA values are deleted from the population and sample data. For the EBP procedure complete observations are required. Defaults to FALSE.

`weights` a character string containing the name of a variable that indicates weights in the sample data. If a character string is provided a weighted version of the ebp will be used. The variable has to be numeric. Defaults to NULL.

`pop_weights` a character string containing the name of a variable that indicates population weights in the populatation data. If a character string is provided weighted indicators are estimated using population weights. The variable has to be numeric. Defaults to NULL.

`aggregate_to` a character string containing the name of a variable from population data that indicates the target domain level for which the results are to be displayed. The variable can be numeric or a factor. Defaults to NULL.

`weights_type` a character string. Two different methods for survey weights are available (i) EBP under informative sampling from *Guadarrama et al. (2018)* ("Guadarrama"); (ii) considering survey weights by using the weighting options of `nlme` from *Pinheiro and Bates (2023)* ("nlme"); (iii) considering survey weights by using the weighting options of `nlme` and use these weights also to determine the optimal transformation parameter lambda ("nlme_lambda"). Defaults to "Guadarrama".

`benchmark` The input depends on the type of benchmarking to be performed. (i) Benchmarking with a fixed value: (a) with one value for each indicator: a named vector containing the numeric benchmark value(s). The names of the vector matchs to the chosen indicators. Benchmarking is available for "Mean" and "Head_Count". (b) with values for the sub-level specified in the argument benchmark_level: a data.frame composed of a variable of class character containing the domain names at which the benchmarkaing is performed and variable(s) with benchmark value(s) of class numeric. Benchmarking is supplied for the Mean and the Head_Count ratio. Therefore, the names of the data.frame must match for the first variable the benchmark_level and for the other(s) to Mean and Head_Count. (ii) Benchmarking with the survey data: a vector containing the names of the chosen indicators. In this case, survey weights (weights) are needed. Benchmarking is available for "Mean" and "Head_Count".

`benchmark_type` a character indicating the type of benchmarking. Types that can be chosen (i) Raking ("raking"), (ii) Ratio adjustment ("ratio"), and for head count, ratio adjustment of the complement ("ratio_complement". Defaults to "ratio"

`benchmark_level`
a character indicating the level at which the benchmarking is performed. This name must be represented in the sample and population data as variable name.

`benchmark_weights`
the name of variable containing benchmark weights. This is only possible for internal benchmarking and enable users to benchmark with weights differing from the survey weights (Default for weighting for internal benchmarking).

`nlme_maxiter` an integer indicating the maximum number of iterations the lme function from package `nlme` will run for parameter convergence. Defaults to 1000.

`nlme_tolerance` a real number indicating the tolerance criterion for the the `lme` function from package `nlme`. Defaults to 1e^-6.

`nlme_opt` a string indicating the optimizer to be used by the `lme` function from package `nlme`, either "nlminb" (the default) or "optim".

`nlme_optimmethod` a string indicating the optimization method to be used with the optim optimizer the `lme` function from packages `nlme` and `optim` Defaults to "BFGS".

`nlme_method` a string indicating the method to be used by the `lme` function from package `nlme`, either "REML" (the default) or "ML".

`nlme_mstol` a real number indicating the tolerance criterion for the the optimization step of the `lme` function from package `nlme`. Defaults to 1e^-7.

`nlme_msmaxiter` an integer indicating the maximum number of iterations for the optimization step of the `lme` function from package `nlme` will run for parameter convergence. Defaults to 1000.

`nlme_returnobject` a logical indicating whether the fitted object should be returned with a warning (instead of an error via stop()) when the maximum number of iterations is reached without convergence of the algorithm. Defaults to FALSE

`rescale_weights` a logical indicating if the sample weights are scaled. If `FALSE` (default), the sample weights do not change. When `TRUE` , the sample weights are rescaled such that the average weight is 1 within each domain.

`Ydump` a string specifying the name of a .csv file to save all simulated values of the dependent value, model predictions, and error terms used for point estimation.

Details

For Monte-Carlo approximations and in the parametric bootstrap approach random number generation is used. Thus, a seed is set by the argument `seed`.

The set of predefined indicators includes the mean, median, four further quantiles (10%, 25%, 75% and 90%), head count ratio, poverty gap, Gini coefficient and the quintile share ratio.

Since the sample observations often cannot be identified in practical applications, a modified approach by Guadarrama et al. (2016) called census EBP is implemented for the point estimation. For the MSE estimation, the bootstrap sample is not extracted from the superpopulation, but generated by the estimated model parameters. The lower the ratio between the sample and the population size, the closer are the results to the proposed approach by Molina and Rao (2010).

Value

An object of class "ebp", "emdi" that provides estimators for regional disaggregated indicators and optionally corresponding MSE estimates. Several generic functions have methods for the returned object. For a full list and descriptions of the components of objects of class "emdi", see `emdiObject`.

References

Battese, G.E., Harter, R.M. and Fuller, W.A. (1988). An Error-Components Model for Predictions of County Crop Areas Using Survey and Satellite Data. Journal of the American Statistical Association, Vol.83, No. 401, 28-36.

Gonzalez-Manteiga, W. et al. (2008). Bootstrap mean squared error of a small-area EBLUP. Journal of Statistical Computation and Simulation, 78:5, 443-462.

Guadarrama, M., Molina, I. and Rao, J.N.K. (2016). A comparison of small area estimation methods for poverty mapping. Joint Issue: Statistics in Transition New Series Survey Methodology, Vol.17, No. 1, 41–66.

Guadarrama, M., Molina, I. and Rao, J.N.K. (2018). Small area estimation of general parameters under complex sampling designs. Computational Statistics & Data Analysis, Vol. 121, 20-40.

Kreutzmann, A., Pannier, S., Rojas-Perilla, N., Schmid, T., Templ, M. and Tzavidis, N. (2019). The R Package emdi for Estimating and Mapping Regionally Disaggregated Indicators, Journal of Statistical Software, Vol. 91, No. 7, 1–33, <doi:10.18637/jss.v091.i07>

Molina, I. and Rao, J.N.K. (2010). Small area estimation of poverty indicators. The Canadian Journal of Statistics, Vol. 38, No.3, 369-385.

Social Protection Committee (2001). Report on indicators in the field of poverty and social exclusions, Technical Report, European Union. You, Y., Rao, J.N.K. (2002). A pseudo-empirical best linear unbiased prediction approach to small area estimation using survey weights. The Canadian Journal of Statistics. Vol. 30, No. 3, 431–439.

See Also

emdiObject, lme, estimators.emdi, plot.emdi, emdi_summaries

Examples

```
# Loading data - population and sample data
data("eusilcA_pop")
data("eusilcA_smp")

# Example 1: With default setting but na.rm=TRUE
emdi_model <- ebp(
  fixed = eqIncome ~ gender + eqsize + cash + self_empl +
    unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent + fam_allow +
    house_allow + cap_inv + tax_adj, pop_data = eusilcA_pop,
  pop_domains = "district", smp_data = eusilcA_smp, smp_domains = "district",
  na.rm = TRUE
)


# Example 2: With MSE, two additional indicators and function as threshold -
# Please note that the example runs for several minutes. For a short check
# change L and B to lower values.
```

```
emdi_model <- ebp(
  fixed = eqIncome ~ gender + eqsize + cash +
    self_empl + unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent +
    fam_allow + house_allow + cap_inv + tax_adj, pop_data = eusilcA_pop,
  pop_domains = "district", smp_data = eusilcA_smp, smp_domains = "district",
  threshold = function(y) {
    0.6 * median(y)
  }, transformation = "log",
  L = 50, MSE = TRUE, boot_type = "wild", B = 50, custom_indicator =
    list(
      my_max = function(y) {
        max(y)
      },
      my_min = function(y) {
        min(y)
      }
    ), na.rm = TRUE, cpus = 1, nlme_opt="optim"
)

# Example 3: With default setting but na.rm=TRUE under informative sampling.
emdi_model <- ebp(
  fixed = eqIncome ~ gender + eqsize + cash + self_empl +
    unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent + fam_allow +
    house_allow + cap_inv + tax_adj, pop_data = eusilcA_pop,
  pop_domains = "district", smp_data = eusilcA_smp, smp_domains = "district",
  weights = "weight", transformation = "log", na.rm = TRUE
)

# Example 4: With default setting and random effect on the district level
# while the output is at state level
emdi_model <- ebp(
  fixed = eqIncome ~ gender + eqsize + cash + self_empl +
    unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent + fam_allow +
    house_allow + cap_inv + tax_adj, pop_data = eusilcA_pop,
  pop_domains = "district", smp_data = eusilcA_smp, smp_domains = "district",
  na.rm = TRUE, aggregate_to = "state"
)

# Example 5: With default setting using pop_weights to get weighted
# indicators according to equivalized household size and an using an
# custom_indicator using pop_weights
emdi_model <- ebp(
  fixed = eqIncome ~ gender + eqsize + cash + self_empl +
    unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent + fam_allow +
    house_allow + cap_inv + tax_adj, pop_data = eusilcA_pop,
  pop_domains = "district", smp_data = eusilcA_smp, smp_domains = "district",
  custom_indicator =
    list(HCR_singleHH = function(y, pop_weights, threshold) {
                              mean(y[pop_weights == 1] < threshold)
                          }
    ), na.rm = TRUE, pop_weights = "hhsize"
)
```

ebp_compute_cv *Coefficient of Variation (CV) estimations for Unit EBP Model Headcount Estimates*

Description

Function ebp_compute_cv estimates CVs for the headcount of the unit model EBP functions using three different methods. CV, by definition, is the ratio of mean square error of the head count to the head count estimates. Therefore, the CV types are distinguished by the method of estimating the mean square.

Usage

```
ebp_compute_cv(
  model,
  calibvar = NULL,
  boot_type = "calibrate",
  designvar = NULL,
  threshold = NULL,
  B = model$call$B
)
```

Arguments

model	an object returned by the ebp function of type "emdi ebp", representing point and MSE estimates
calibvar	the calibration variable to be used in method 1
boot_type	the bootstrap type "calibrated" or "naive" to be used in method 1
designvar	the survey design variable to be used in estimating the design effect for method 3.
threshold	a number defining a threshold. The argument defaults to NULL. In this case, the threshold is set to 60% of the median of the variable that is selected as dependent variable similary to the at-risk-of-poverty rate used in the EU (see also *Social Protection Committee 2001*). However, any desired threshold can be chosen.
B	number of bootstrap iterations for variance estimation. Defaults to number of bootstrap iteration in ebp obeject (specified in model).

Details

Method 1 uses the calibrated/naive bootstrapping of the MSE which allows to calibrate each bootstrap sample on auxiliary information using the direct function.' Calibrated bootstrap improves on the bias of the naive bootstrap when used in the complex survey context (see *Rao and Wu (1988)*) for more details.

Method 2 employs the Horowitz Thompson variance estimation technique to compute MSE i.e. each household is assigned the probability selection within the sample under a given sampling scheme. The computation employs sae::direct function.

Method 3 finally uses the design effect adjusted naive calibrated MSE. The design effect is estimated using the `survey::svydesign` function.

Value

dataframe containing different types of CV values for the headcount

Examples

```
data("eusilcA_pop")
data("eusilcA_smp")

# estimate a unit model
ebp_model <- ebp(fixed = eqIncome ~ gender + eqsize + cash +
                    self_empl + unempl_ben + age_ben + surv_ben + sick_ben +
                    dis_ben + rent + fam_allow + house_allow + cap_inv +
                    tax_adj,
                  pop_data = eusilcA_pop, pop_domains = "district",
                  smp_data = eusilcA_smp, smp_domains = "district",
                  na.rm = TRUE, weights = "weight",
                  pop_weights = "hhsize", MSE = TRUE, weights_type = "nlme",
                  B = 2, L = 2)

# compute CV table
ebp_compute_cv(model = ebp_model, calibvar = "gender")
```

---

ebp_normalityfit *Output Model fit and normality assumptions*

Description

The function uses the results of the ebp function to produce output a table showing marginal R-square, conditional R-squared as well as the skewness and kurtosis of the random and idiosyncratic error terms

Usage

```
ebp_normalityfit(model)
```

Arguments

`model`	an object returned by the ebp function of type "emdi ebp"

Value

dataframe with marginal R-square, conditional R-squared as well as the skewness and kurtosis of the random and idiosyncratic error term

Examples

```
data("eusilcA_pop")
data("eusilcA_smp")

ebp_model <- ebp(
 fixed = eqIncome ~ gender + eqsize + cash + self_empl +
   unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent + fam_allow +
   house_allow + cap_inv + tax_adj, pop_data = eusilcA_pop,
 pop_domains = "district", smp_data = eusilcA_smp, smp_domains = "district",
 L = 2, na.rm = TRUE
 )

 ebp_normalityfit(model = ebp_model)
```

---

ebp_reportcoef_table *Produce coefficient table for reporting*

Description

This function takes the object of class 'ebp' to present the regression model results having specified the number of decimal places.

Usage

```
ebp_reportcoef_table(model, decimals = 3)
```

Arguments

model	an object returned by the ebp function of type "emdi ebp", representing point and MSE estimates
decimals	the number of decimals to report on coefficient estimates

Value

dataframe with regression model results

Examples

```
data("eusilcA_pop")
data("eusilcA_smp")

ebp_model <- ebp(
 fixed = eqIncome ~ gender + eqsize + cash + self_empl +
   unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent + fam_allow +
   house_allow + cap_inv + tax_adj, pop_data = eusilcA_pop,
 pop_domains = "district", smp_data = eusilcA_smp, smp_domains = "district",
 L = 2, na.rm = TRUE)
```

```
ebp_reportcoef_table(ebp_model, 4)
```

---

ebp_reportdescriptives *Create Descriptive Statistics for Small Area Estimation Report*

Description

This function estimates the coefficient of variation at level specified, basic statistics such number of units, regions and target areas as well as the threshold on which SAE is applied and the outcome indicator of interest (i.e. poverty line and poverty rate). These indicators are all expressed for the census and survey

Usage

```
ebp_reportdescriptives(
  model,
  direct,
  pop_data,
  pop_domains,
  smp_data,
  threshold = NULL,
  weights = NULL,
  pop_weights = NULL,
  CV_level,
  indicator = "Head_Count"
)
```

Arguments

model	an object returned by the ebp function of type "emdi ebp", representing point and MSE estimates
direct	an object of type "direct","emdi", representing point and MSE estimates.
pop_data	the population/census/training data
pop_domains	the target area variable within 'pop_data'
smp_data	sample data
threshold	a number defining a threshold. The argument defaults to NULL. In this case, the threshold is set to 60% of the median of the variable that is selected as dependent variable similary to the at-risk-of-poverty rate used in the EU (see also *Social Protection Committee 2001*). However, any desired threshold can be chosen.

weights	a character string containing the name of a variable that indicates weights in the sample data. If a character string is provided a weighted version of the ebp will be used. The variable has to be numeric. Defaults to NULL.
pop_weights	a character string containing the name of a variable that indicates population weights in the populatation data. If a character string is provided weighted indicators are estimated using population weights. The variable has to be numeric. Defaults to NULL.
CV_level	the variable level at which Coefficient of Variation should be computed
indicator	a character string containing the name of the indicator to compute the Coefficient of Variation for. Defaults to "Head_Count"

Value

an list containing three dataframes (first dataframe with direct an ebp CV values, second dataframe with basic statistics and third dataframe with national poverline and rate for census and survey

Examples

```
data("eusilcA_pop")
data("eusilcA_smp")

# estimate a unit model
ebp_model <- ebp(fixed = eqIncome ~ gender + eqsize + cash +
                   self_empl + unempl_ben + age_ben + surv_ben + sick_ben +
                   dis_ben + rent + fam_allow + house_allow + cap_inv +
                   tax_adj,
                 pop_data = eusilcA_pop, pop_domains = "district",
                 smp_data = eusilcA_smp, smp_domains = "district",
                 na.rm = TRUE, weights = "weight",
                 pop_weights = "hhsize", MSE = TRUE, weights_type = "nlme",
                 B = 2, L = 2)

# estimate direct
direct_est <- direct(y = "eqIncome", smp_data = eusilcA_smp,
                     smp_domains = "district", weights = "weight",
                     var = TRUE, B = 2)

# descritives
ebp_reportdescriptives(model = ebp_model, direct = direct_est,
                       smp_data = eusilcA_smp, weights = "weight",
                       pop_weights = "hhsize", CV_level = "state",
                       pop_data = eusilcA_pop, pop_domains = "district")
```

---

ebp_report_byrank *Produce EBP Head Count Population/Rate by Rank*

Description

This function combines the ebp object with the census data to produce report tables that rank head count estimates either by population of poor or the head count rates themselves in descending order. The function allows the user to select the first/last "x" number of areas by name as well.

Usage

```
ebp_report_byrank(
  model,
  pop_data,
  pop_domains,
  pop_weights = NULL,
  byrank_indicator = "count",
  number_to_list = NULL,
  head = TRUE,
  indicator = "Head_Count"
)
```

Arguments

model	an object returned by the ebp function of type "emdi ebp".
pop_data	the population/census/training data
pop_domains	a character string containing the name of a variable that indicates domains in the population data. The variable can be numeric or a factor but needs to be of the same class as the variable named in smp_domains.
pop_weights	a character string containing the name of a variable that indicates population weights in the populatation data. If a character string is provided weighted indicators are estimated using population weights. The variable has to be numeric. Defaults to NULL. Please note that pop_weights should only be used if in the pop_data not individual data is provided and thus the number of persons per unit (e.g. household, grid) must be indicated.
byrank_indicator	if argument is "count", the function ranks the product of Head_Count (from object of class ‘ebp‘) and ‘pop_weights‘, otherwise it the function simply ranks Head_Count output within ‘ebp‘ object
number_to_list	an integer, the first ‘number_to_list‘ number of target areas to produce from ‘byrank_indicator‘ ordering.
head	a logical, if ‘TRUE‘ the top ‘number_to_list‘ results will be returned and if ‘FALSE‘ the bottom ‘number_to_list‘ will be returned
indicator	a character string containing the name of the indicator to rank. Defaults to "Head_Count"

Value

dataframe containing population size, head count values and counts of poor population

Examples

```
data("eusilcA_pop")
data("eusilcA_smp")

ebp_model <- ebp(
 fixed = eqIncome ~ gender + eqsize + cash + self_empl +
    unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent + fam_allow +
    house_allow + cap_inv + tax_adj,
 pop_data = eusilcA_pop, pop_domains = "district",
 smp_data = eusilcA_smp, smp_domains = "district", L = 2,
 weights = "weight", weights_type = "nlme", na.rm = TRUE,
 pop_weights = "hhsize")

# full data of highest population below threshold by rank (descending order)
ebp_report_byrank(model = ebp_model,
                  pop_data = eusilcA_pop,
                  pop_domains = "district",
                  pop_weights = "hhsize")

# full data of highest rate below threshold by rank (descending order)
ebp_report_byrank(model = ebp_model,
                  pop_data = eusilcA_pop,
                  pop_domains = "district",
                  pop_weights = "hhsize",
                  byrank_indicator = "rate")

# bottom 10 poverty count below threshold by rank (in ascending order)
ebp_report_byrank(model = ebp_model,
                  pop_data = eusilcA_pop,
                  pop_domains = "district",
                  pop_weights = "hhsize",
                  number_to_list = 10,
                  head = FALSE)
```

`ebp_test_means` *Perform test for difference between survey and census means*

Description

This function computes weighted means of the same set of variables within the census and the survey. A test for difference of the means are performed for each variable with two-tailed p-values returned.

Usage

```
ebp_test_means(varlist, smp_data, pop_data, weights = NULL, pop_weights = NULL)
```

Arguments

varlist	character vector, the set of variables of interest
smp_data	the survey data
pop_data	the population data
weights	a character string containing the name of a variable that indicates weights in the sample data. If a character string is provided a weighted version of the ebp will be used. The variable has to be numeric. Defaults to NULL.
pop_weights	a character string containing the name of a variable that indicates population weights in the populatation data. If a character string is provided weighted indicators are estimated using population weights. The variable has to be numeric. Defaults to NULL. Please note that pop_weights should only be used if the samples and population data are at different levels (e.g.: smp_data at individual level and pop_data at household level, then pop_weights is needed for the comparison with a variable indicating household size).

Value

dataframe with census and survey means and test results for their difference.

Examples

```
data("eusilcA_pop")
data("eusilcA_smp")

variables <- c("gender", "eqsize", "cash", "self_empl",
               "unempl_ben", "age_ben", "surv_ben",
               "sick_ben", "dis_ben", "rent", "fam_allow",
               "house_allow", "cap_inv", "tax_adj")

ebp_test_means(varlist = variables,
               pop_data = eusilcA_pop,
               smp_data = eusilcA_smp,
               weights = "weight")
```

---

emdi *A package for estimating and mapping disaggregated indicators*

Description

The package **emdi** supports estimating and mapping regional disaggregated indicators. For estimating these indicators, direct estimation, the unit-level Empirical Best Prediction approach by *Molina and Rao (2010)*, the extension for data under informative selection by *Guadarrama et al. (2018)*, the area-level model by *Fay and Herriot (1979)* and various extensions of it (adjusted variance estimation methods, log and arcsin transformation, spatial, robust and measurement error models) are

provided. Depending on the particular method, analytical, bootstrap and jackknife MSE estimation approaches are implemented. The assessment of the used model is supported by a summary and diagnostic plots. For a suitable presentation of estimates, map plots can be easily created. Furthermore, results can easily be exported to Excel. Additionally, for the area-level models a stepwise variable selection function, benchmarking options and spatial correlation tests are provided.

Details

The three estimation functions are called `direct`, `ebp` and `fh`. For all functions, several methods are available as `estimators.emdi` and `emdi_summaries`. For a full list, please see `emdiObject`. Furthermore, functions `map_plot` and `write.excel` help to visualize and export results. An overview of all currently provided functions can be requested by `library(help=emdi)`.

References

Battese, G.E., Harter, R.M. and Fuller, W.A. (1988). An Error-Components Model for Predictions of County Crop Areas Using Survey and Satellite Data. Journal of the American Statistical Association, Vol.83, No. 401, 28-36.

Fay, R. E. and Herriot, R. A. (1979), Estimates of income for small places: An application of James-Stein procedures to census data, Journal of the American Statistical Association 74(366), 269-277.

Kreutzmann, A., Pannier, S., Rojas-Perilla, N., Schmid, T., Templ, M. and Tzavidis, N. (2019). The R Package emdi for Estimating and Mapping Regionally Disaggregated Indicators, Journal of Statistical Software, Vol. 91, No. 7, 1–33, <doi:10.18637/jss.v091.i07>

Molina, I. and Rao, J.N.K. (2010). Small area estimation of poverty indicators. The Canadian Journal of Statistics, Vol. 38, No.3, 369-385. Guadarrama, M., Molina, I. and Rao, J.N.K. (2018). Small area estimation of general parameters under complex sampling designs. Computational Statistics & Data Analysis, Vol. 121, 20-40.

`emdiObject` *Fitted emdiObject*

Description

An object of class emdi that represents point predictions of regional disaggregated indicators. Optionally, it also contains corresponding MSE estimates. Three different estimation approaches are implemented: direct estimation (class 'direct), the Fay-Herriot model (class "fh"), and the empirical best prediction (class "ebp"). Objects of these classes have methods for various generic functions. See Details for more information.

Details

Objects of class "emdi" have following methods: `compare_pred`, `estimators`, `plot.emdi`, `predict.emdi`, `qqnorm.emdi`

Objects of class "direct", "ebp" and "fh" have methods for following generic functions: compare_plot, getData, getGroups, getGroupsFormula, getResponse, plot (for documentation, see plot.emdi), print, qqnorm (for documentation, see qqnorm.emdi) and summary (for documentation, see emdi_summaries).

Objects of class "ebp" and "fh" additionally have methods for following generic functions: coef (for default documentation, see coef), confint (for default documentation, see confint), family (for default documentation, see family), fitted (for default documentation, see fitted.values), fixef, formula (for default documentation, see formula), getVarCov, intervals, logLik (for default documentation, see logLik), nobs (for default documentation, see nobs), ranef, residuals (for default documentation, see residuals), terms (for default documentation, see terms), vcov (for default documentation, see vcov)

Objects of class "ebp" have additionally methods for following generic functions: sigma (for default documentation, see sigma)

Objects of class "fh" have additionally methods for following generic functions: compare, extractAIC (for default documentation, see extractAIC) and step.

Value

The following components are always included in an emdi object but not always filled and with different components depending on the estimation approach:

call	the function call that produced the object.
fixed	for details, see fixed in fh and ebp. Not filled for class "direct".
framework	a list with components that describe the data setup, e.g., number of domains in the sample.
ind	data frame containing estimates for indicators per domain.
method	character returning the method for the estimation approach used to fit the linear mixed model and for the the optimal lambda (for class "ebp"), here "reml", or a list returning method for the estimation of the variance of the random effect and the applied MSE estimation (for class "fh"). Not filled for class "direct".
model	list containing a selection of model components. Not filled for class "direct".
MSE	data frame containing MSE estimates corresponding to the point predictions in ind per indicator per domain if MSE is selected in function call. If FALSE, MSE is NULL.
transformation	character or list containing information about applied transformation and, if appropriate, backtransformation. Not filled for class "direct".
transform_param	a list with two elements, optimal_lambda and shift_par, where the first contains the optimal parameter for a transformation with transformation parameter or NULL for no and log transformation and the second the potential shift parameter in the log or Box-Cox transformation and NULL for no transformation. Not filled for class "fh" and "direct".

successful_bootstraps
: for class "direct", a matrix with domains as rows and indicators as columns. The cells contain the number of successful bootstraps for each combination. For non-robust spatial Fay-Herriot, string with number of successful bootstraps. Not filled for other models.

References

Alfons, A. and Templ, M. (2013). Estimation of Social Exclusion Indicators from Complex Surveys: The R Package **laeken**. Journal of Statistical Software, 54(15), 1-25.

Fay R.E., Herriot R.A. (1979) Estimates of income for small places: An application of James–Stein procedures to census data. Journal of the American Statistical Association, Vol. 74, 269–277.

Molina, I. and Rao, J.N.K. (2010). Small area estimation of poverty indicators. The Canadian Journal of Statistics, Vol. 38, No.3, 369-385.

See Also

direct, ebp, fh, lme, lmeObject

---

emdi_summaries *Summarizes an emdiObject*

Description

Additional information about the data and model in small area estimation methods and components of an emdi object are extracted. The generic function summary has methods for classes "direct", "ebp" and "fh" and the returned object is suitable for printing with the print.

Usage

```
## S3 method for class 'direct'
summary(object, ...)

## S3 method for class 'ebp'
summary(object, ...)

## S3 method for class 'fh'
summary(object, ...)
```

Arguments

object	an object of type "direct", "ebp" or "fh", representing point and MSE estimates. Objects differ depending on the estimation method.
...	additional arguments that are not used in this method.

Value

an object of type "summary.direct", "summary.ebp" or "summary.fh" with information about the sample and population data, the usage of transformation, normality tests and information of the model fit.

References

Lahiri, P. and Suntornchost, J. (2015), Variable selection for linear mixed models with applications in small area estimation, The Indian Journal of Statistics 77-B(2), 312-320.

Marhuenda, Y., Morales, D. and Pardo, M.C. (2014). Information criteria for Fay-Herriot model selection. Computational Statistics and Data Analysis 70, 268-280.

Nakagawa S, Schielzeth H (2013). A general and simple method for obtaining R2 from generalized linear mixed-effects models. Methods in Ecology and Evolution, 4(2), 133-142.

See Also

emdiObject, direct, ebp, fh, r.squaredGLMM, skewness, kurtosis, shapiro.test

Examples

```
# Example for models of type ebp

# Loading data - population and sample data
data("eusilcA_pop")
data("eusilcA_smp")

# Example with two additional indicators
emdi_model <- ebp(
  fixed = eqIncome ~ gender + eqsize + cash +
    self_empl + unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent +
    fam_allow + house_allow + cap_inv + tax_adj, pop_data = eusilcA_pop,
  pop_domains = "district", smp_data = eusilcA_smp, smp_domains = "district",
  threshold = function(y) {
    0.6 * median(y)
  }, L = 50, MSE = TRUE, B = 50,
  custom_indicator = list(
    my_max = function(y) {
      max(y)
    },
    my_min = function(y) {
      min(y)
    }
  ), na.rm = TRUE, cpus = 1
)

# Example 1: Receive first overview
summary(emdi_model)
```

```
# Example for models of type fh

# Loading data - population and sample data
data("eusilcA_popAgg")
data("eusilcA_smpAgg")

# Combine sample and population data
combined_data <- combine_data(
  pop_data = eusilcA_popAgg,
  pop_domains = "Domain",
  smp_data = eusilcA_smpAgg,
  smp_domains = "Domain"
)

# Generation of the emdi object
fh_std <- fh(
  fixed = Mean ~ cash + self_empl, vardir = "Var_Mean",
  combined_data = combined_data, domains = "Domain",
  method = "ml", MSE = TRUE
)

# Example 2: Receive first overview
summary(fh_std)
```

`estimators` *Presents Point, MSE and CV Estimates*

Description

Function `estimators` is a generic function used to present point and mean squared error (MSE) estimates and calculated coefficients of variation (CV).

Method `estimators.emdi` presents point and MSE estimates for regional disaggregated indicators. Coefficients of variation are calculated using these estimators. This method enables to select for which indicators the estimates shall be returned. The returned object is suitable for printing with the `print.estimators.emdi` method.

Usage

```
estimators(object, indicator, MSE, CV, ...)

## S3 method for class 'emdi'
estimators(object, indicator = "all", MSE = FALSE, CV = FALSE, ...)
```

Arguments

`object`	an object of type "emdi", representing point and, if chosen, MSE estimates.

indicator	optional character vector that selects which indicators shall be returned: (i) all calculated indicators ("all"); (ii) each indicator name: "Mean", "Quantile_10", "Quantile_25", "Median", "Quantile_75", "Quantile_90", "Head_Count", "Poverty_Gap", "Gini", "Quintile_Share" or the function name/s of "custom_indicator/s"; (iii) groups of indicators: "Quantiles", "Poverty", "Inequality" or "Custom". If two of these groups are selected, only the first one is returned. Note, additional custom indicators can be defined as argument for model-based approaches (see also ebp) and do not appear in groups of indicators even though these might belong to one of the groups. If the model argument is of type "fh", indicator can be set to "all", "Direct", FH", or "FH_Bench" (if emdi object is overwritten by function benchmark). Defaults to "all".
MSE	optional logical. If TRUE, MSE estimates for selected indicators per domain are added to the data frame of point estimates. Defaults to FALSE.
CV	optional logical. If TRUE, coefficients of variation for selected indicators per domain are added to the data frame of point estimates. Defaults to FALSE.
...	other parameters that can be passed to function estimators.

Details

Objects of class "estimators.emdi" have methods for following generic functions: head and tail (for default documentation, see head), as.matrix (for default documentation, see matrix), as.data.frame (for default documentation, see as.data.frame), subset (for default documentation, see subset).

Value

The return of estimators depends on the class of its argument. The documentation of particular methods gives detailed information about the return of that method.

The return of estimators.emdi is an object of type "estimators.emdi" with point and/or MSE estimates and/or calculated CV's per domain obtained from emdiObject$ind and, if chosen, emdiObject$MSE. These objects contain two elements, one data frame ind and a character naming the indicator or indicator group ind_name.

See Also

emdiObject, direct, ebp, fh

Examples

```
# Loading data - population and sample data
data("eusilcA_pop")
data("eusilcA_smp")

# Generate emdi object with additional indicators; here via function ebp()
emdi_model <- ebp(
  fixed = eqIncome ~ gender + eqsize + cash +
    self_empl + unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent +
    fam_allow + house_allow + cap_inv + tax_adj, pop_data = eusilcA_pop,
  pop_domains = "district", smp_data = eusilcA_smp, smp_domains = "district",
  threshold = 11064.82, transformation = "box.cox",
```

```
  L = 50, MSE = TRUE, B = 50, custom_indicator =
    list(
      my_max = function(y) {
        max(y)
      },
      my_min = function(y) {
        min(y)
      }
    ), na.rm = TRUE, cpus = 1
)

# Example 1: Choose Gini coefficient, MSE and CV
gini <- estimators(emdi_model, indicator = "Gini", MSE = TRUE, CV = TRUE)
head(gini)
tail(gini)
as.data.frame(gini)
as.matrix(gini)
subset(gini, Domain = "Wien")

# Example 2: Choose custom indicators without MSE and CV
estimators(emdi_model, indicator = "Custom")
```

`eusilcA_pop` *Simulated eusilc data - population data*

Description

The data set is synthetic EU-SILC data based on the data set `eusilcP` from package **simFrame**. The data set is reduced to 17 variables containing three regional variables for the states and districts.

Usage

```
eusilcA_pop
```

Format

A data frame with 25000 observations and 17 variables:

eqIncome numeric; a simplified version of the equivalized household income.

eqsize numeric; the equivalized household size according to the modified OECD scale.

gender factor; the person's gender (levels: male and female).

cash numeric; employee cash or near cash income (net).

self_empl numeric; cash benefits or losses from self-employment (net).

unempl_ben numeric; unemployment benefits (net).

age_ben numeric; old-age benefits (net).

surv_ben numeric; survivor's benefits (net).

sick_ben numeric; sickness benefits (net).

dis_ben numeric; disability benefits (net).

rent numeric; income from rental of a property or land (net).

fam_allow numeric; family/children related allowances (net).

house_allow numeric; housing allowances (net).

cap_inv numeric; interest, dividends, profit from capital investments in unincorporated business (net).

tax_adj numeric; repayments/receipts for tax adjustment (net).

state factor; state (nine levels).

district factor; districts (94 levels).

hhsize numeric; persons in household.

`eusilcA_popAgg` *Simulated eusilc data - aggregated population data*

Description

The data set is synthetic EU-SILC data based on the data set `eusilcP` from package **simFrame**. The data set is reduced to 15 variables including a regional variable for the districts and contains the household level data that is aggregated on the district level. Therefore, except for the variables `ratio_n` and `Domain`, the variables are the mean values per district.

Usage

```
eusilcA_popAgg
```

Format

A data frame with 94 observations and 15 variables:

eqsize numeric; the equivalized household size according to the modified OECD scale.

cash numeric; employee cash or near cash income (net).

self_empl numeric; cash benefits or losses from self-employment (net).

unempl_ben numeric; unemployment benefits (net).

age_ben numeric; old-age benefits (net).

surv_ben numeric; survivor's benefits (net).

sick_ben numeric; sickness benefits (net).

dis_ben numeric; disability benefits (net).

rent numeric; income from rental of a property or land (net).

fam_allow numeric; family/children related allowances (net).

house_allow numeric; housing allowances (net).

cap_inv numeric; interest, dividends, profit from capital investments in unincorporated business (net).

tax_adj numeric; repayments/receipts for tax adjustment (net).

ratio_n numeric; ratios of the population size per area and the total population size.

Domain factor; Austrian districts (94 levels).

---

eusilcA_prox *Proximity matrix for spatial area-level models*

Description

A data set comprising the row-standardized proximities between the domains of the eusilcA_smpAgg data set.

Usage

```
eusilcA_prox
```

Format

A data set with dimensions number of areas (94) times number of areas (94). Values lie between 0 and 1. The respective row sums amount to 1.

Details

For a description of how to create the proximity matrix, see the package vignette "A Framework for Producing Small Area Estimates Based on Area-Level Models in R".

---

eusilcA_smp *Simulated eusilc data - sample data*

Description

The data set is a simple random sample of data set eusilcA_pop which is based on eusilcP from package **simFrame**.

Usage

```
eusilcA_smp
```

Format

A data frame with 1000 observations and 18 variables:

eqIncome numeric; a simplified version of the equivalized household income.

eqsize numeric; the equivalized household size according to the modified OECD scale.

gender factor; the person's gender (levels: male and female).

cash numeric; employee cash or near cash income (net).

self_empl numeric; cash benefits or losses from self-employment (net).

unempl_ben numeric; unemployment benefits (net).

age_ben numeric; old-age benefits (net).

surv_ben numeric; survivor's benefits (net).

sick_ben numeric; sickness benefits (net).

dis_ben numeric; disability benefits (net).

rent numeric; income from rental of a property or land (net).

fam_allow numeric; family/children related allowances (net).

house_allow numeric; housing allowances (net).

cap_inv numeric; interest, dividends, profit from capital investments in unincorporated business (net).

tax_adj numeric; repayments/receipts for tax adjustment (net).

state factor; state (nine levels).

district factor; districts (94 levels).

weight numeric; constant weight.

hhsize numeric; persons in household.

`eusilcA_smpAgg` *Simulated eusilc data - aggregated sample data*

Description

The data set is a simple random sample of data set `eusilcA_pop` which is based on `eusilcP` from package **simFrame**. The data set is aggregated on the district level and contains different variables that are related to income and a regional variable for the districts.

Usage

```
eusilcA_smpAgg
```

Format

A data frame with 94 observations and 8 variables:

Mean numeric; mean of a simplified version of the equivalized household income.

MTMED numeric; share of households who earn more than the national median income.

Cash numeric; mean of employee cash or near cash income.

Var_Mean numeric; variance of a simplified version of the equivalized household income.

Var_MTMED numeric; variance of the share of households who earn more than the national median income.

Var_Cash numeric; variance of the employee cash or near cash income.

n numeric; effective sample sizes.

Domain factor; Austrian districts (94 levels).

`fh` *Standard and Extended Fay-Herriot Models for Disaggregated Indicators*

Description

Function `fh` estimates indicators using the Fay-Herriot approach by *Fay and Herriot (1979)*. Empirical best linear unbiased predictors (EBLUPs) and mean squared error (MSE) estimates are provided. Additionally, different extensions of the standard Fay-Herriot model are available:
Adjusted estimation methods for the variance of the random effects (see *Li and Lahiri (2010)* and *Yoshimori and Lahiri (2014)*) are offered. Log and arcsin transformation for the dependent variable and two types of backtransformation can be chosen - a crude version and the one introduced by *Slud and Maiti (2006)* for log transformed variables and a naive and bias-corrected version following *Hadam et al. (2020)* for arcsin transformed variables. A spatial extension to the Fay-Herriot model following *Petrucci and Salvati (2006)* is also included. In addition, it is possible to estimate a robust version of the standard and of the spatial model (see *Warnholz (2016)*). Finally, a Fay-Herriot model can be estimated when the auxiliary information is measured with error following *Ybarra and Lohr (2008)*.

Usage

```
fh(
  fixed,
  vardir,
  combined_data,
  domains = NULL,
  method = "reml",
  interval = NULL,
  k = 1.345,
  mult_constant = 1,
  transformation = "no",
  backtransformation = NULL,
```

```
  eff_smpsize = NULL,
  correlation = "no",
  corMatrix = NULL,
  Ci = NULL,
  tol = 1e-04,
  maxit = 100,
  MSE = FALSE,
  mse_type = "analytical",
  B = c(50, 0),
  seed = 123
)
```

Arguments

`fixed`	a two-sided linear formula object describing the fixed-effects part of the linear mixed regression model with the dependent variable on the left of a ~ operator and the explanatory variables on the right, separated by + operators.
`vardir`	a character string indicating the name of the variable containing the domain-specific sampling variances of the direct estimates that are included in `combined_data`.
`combined_data`	a data set containing all the input variables that are needed for the estimation of the Fay-Herriot model: the direct estimates, the sampling variances, the explanatory variables and the domains. In addition, the effective sample size needs to be included, if the arcsin transformation is chosen.
`domains`	a character string indicating the domain variable that is included in `combined_data`. If `NULL`, the domains are numbered consecutively.
`method`	a character string describing the method for the estimation of the variance of the random effects. Methods that can be chosen (i) restricted maximum likelihood (REML) method (`"reml"`), (ii) maximum likelihood method (`"ml"`), (iii) adjusted REML following *Li and Lahiri (2010)* (`"amrl"`), (iv) adjusted ML following *Li and Lahiri (2010)* (`"ampl"`), (v) adjusted REML following *Yoshimori and Lahiri (2014)* (`"amrl_yl"`), (vi) adjusted ML following *Yoshimori and Lahiri (2014)* (`"ampl_yl"`), (vii) robustified maximum likelihood with robust EBLUP prediction following *Warnholz (2017)* (`"reblup"`), (viii) robustified maximum likelihood with robust and bias-corrected EBLUP prediction following *Warnholz (2017)* (`"reblupbc"`), (ix) estimation of the measurement error model of *Ybarra and Lohr (2008)* (`"me"`). Defaults to `"reml"`.
`interval`	optional argument, if method `"reml"` and `"ml"` in combination with `correlation` equals `"no"` is chosen or for the adjusted variance estimation methods `"amrl"`, `"amrl_yl"`, `"ampl"` and `"ampl_yl"`. Is internally set to `c(0, var(direct estimates))`. If a transformation is applied, the interval is internally set to `c(0, var(transformed(direct estimates)))`. If desired, `interval` can be specified to a numeric vector containing a lower and upper limit for the estimation of the variance of the random effects. Defaults to `NULL`.
`k`	numeric tuning constant. Required argument when the robust version of the standard or spatial Fay-Herriot model is chosen. Defaults to `1.345`. For detailed information, please refer to *Warnholz (2016)*.

`mult_constant` numeric multiplier constant used in the bias corrected version of the robust estimation methods. Required argument when the robust version of the standard or spatial Fay-Herriot model is chosen. Default is to make no correction for realizations of direct estimator within `mult_constant = 1` times the standard deviation of direct estimator. For detailed information, please refer to *Warnholz (2016)*.

`transformation` a character that determines the type of transformation of the dependent variable and of the sampling variances. Methods that can be chosen (i) no transformation ("no"), (ii) log transformation ("log") of the dependent variable and of the sampling variances, (iii) arcsin transformation ("arcsin") of the dependent variable and of the sampling variances following. Defaults to "no". For more information, how the direct estimate and its variance are transformed, please see the package vignette "A Framework for Producing Small Area Estimates Based on Area-Level Models in R".

`backtransformation`
a character that determines the type of backtransformation of the EBLUPs and MSE estimates. Required argument when a transformation is chosen. Available methods are (i) crude bias-correction following *Rao (2015)* when the log transformation is chosen ("bc_crude"), (ii) bias-correction following *Slud and Maiti (2006)* when the log transformations is chosen ("bc_sm"), (iii) naive back transformation when the arcsin transformation is chosen ("naive"), (iii) bias-corrected back transformation following *Hadam et al. (2020)* when the arcsin transformation is chosen ("bc"). Defaults to NULL.

`eff_smpsize` a character string indicating the name of the variable containing the effective sample sizes that are included in `combined_data`. Required argument when the arcsin transformation is chosen. Defaults to NULL.

`correlation` a character determining the correlation structure of the random effects. Possible correlations are (i) no correlation ("no"), (ii) incorporation of a spatial correlation in the random effects ("spatial"). Defaults to "no".

`corMatrix` matrix or data frame with dimensions number of areas times number of areas containing the row-standardized proximities between the domains. Values must lie between 0 and 1. The columns and rows must be sorted like the domains in `fixed`. For an example how to create the proximity matrix, please refer to the vignette. Required argument when the correlation is set to "spatial". Defaults to NULL.

`Ci` array with dimension number of estimated regression coefficients times number of estimated regression coefficients times number of areas containing the variance-covariance matrix of the explanatory variables for each area. For an example of how to create the array, please refer to the vignette. Required argument within the Ybarra-Lohr model (`method = me`). Defaults to NULL.

`tol` a number determining the tolerance value for the estimation of the variance of the random effects. Required argument when method "reml" and "ml" in combination with `correlation ="spatial"` are chosen or for the variance estimation methods "reblup", "reblupbc" and "me". Defaults to 0.0001.

`maxit` a number determining the maximum number of iterations for the estimation of the variance of the random effects. Required argument when method "reml"

and "ml" in combination with correlation equals "spatial" is chosen or for the variance estimation methods "reblup", "reblupbc" and "me". Defaults to 100.

MSE — if TRUE, MSE estimates are calculated. Defaults to FALSE.

mse_type — a character string determining the estimation method of the MSE. Methods that can be chosen (i) analytical MSE depending on the estimation method of the variance of the random effect ("analytical"), (ii) a jackknife MSE ("jackknife"), (iii) a weighted jackknife MSE ("weighted_jackknife"), (iv) bootstrap ("boot"), (v) approximation of the MSE based on a pseudo linearisation ("pseudo"), (vi) naive parametric bootstrap for the spatial Fay-Herriot model ("spatialparboot"), (vii) bias corrected parametric bootstrap for the spatial Fay-Herriot model ("spatialparbootbc"), (viii) naive nonparametric bootstrap for the spatial Fay-Herriot model ("spatialnonparboot"), (ix) bias corrected nonparametric bootstrap for the spatial Fay-Herriot model ("spatialnonparbootbc"). Options (ii)-(iv) are of interest when the arcsin transformation is selected. Option (ii) must be chosen when an Ybarra-Lohr model is selected (method = me). Options (iv) and (v) are the MSE options for the robust extensions of the Fay-Herriot model. For an extensive overview of the possible MSE options, please refer to the vignette. Required argument when MSE = TRUE. Defaults to "analytical".

B — either a single number or a numeric vector with two elements. The single number or the first element defines the number of bootstrap iterations when a bootstrap MSE estimator is chosen. When the standard FH model is applied and the information criteria by Marhuenda et al. (2014) should be computed, the second element of B is needed and must be greater than 1. Defaults to c(50,0). For practical applications, values larger than 200 are recommended.

seed — an integer to set the seed for the random number generator. For the usage of random number generation see details. If seed is set to NULL, seed is chosen randomly. Defaults to 123.

Details

In the bootstrap approaches, random number generation is used. Thus, a seed is set by the argument seed.

Out-of-sample EBLUPs are available for all area-level models except for the bc_sm backtransformation and for the robust models.
Out-of-sample MSEs are available for the analytical MSE estimator of the standard Fay-Herriot model with reml and ml variance estimation, the crude backtransformation in case of log transformation and the bootstrap MSE estimator for the arcsin transformation.

For a description of how to create the proximity matrix for the spatial Fay-Herriot model, see the package vignette. If the presence of out-of-sample domains, the proximity matrix needs to be subsetted to the in-sample domains.

Value

An object of class "fh", "emdi" that provides estimators for regional disaggregated indicators like means and ratios and optionally corresponding MSE estimates. Several generic functions have methods for the returned object. For a full list and descriptions of the components of objects of class "emdi", see `emdiObject`.

References

Chen S., Lahiri P. (2002), A weighted jackknife MSPE estimator in small-area estimation, "Proceeding of the Section on Survey Research Methods", American Statistical Association, 473 - 477.

Datta, G. S. and Lahiri, P. (2000), A unified measure of uncertainty of Statistica Sinica 10(2), 613-627.

Fay, R. E. and Herriot, R. A. (1979), Estimates of income for small places: An application of James-Stein procedures to census data, Journal of the American Statistical Association 74(366), 269-277.

González-Manteiga, W., Lombardía, M. J., Molina, I., Morales, D. and Santamaría, L. (2008) Analytic and bootstrap approximations of prediction errors under a multivariate Fay-Herriot model. Computational Statistics & Data Analysis, 52, 5242–5252.

Hadam, S., Wuerz, N. and Kreutzmann, A.-K. (2020), Estimating regional unemployment with mobile network data for Functional Urban Areas in Germany, Refubium - Freie Universitaet Berlin Repository, 1-28.

Jiang, J., Lahiri, P., Wan, S.-M. and Wu, C.-H. (2001), Jackknifing in the Fay–Herriot model with an example. In Proc. Sem. Funding Opportunity in Survey Research, Washington DC: Bureau of Labor Statistics, 75–97.

Jiang, J., Lahiri, P.,Wan, S.-M. (2002), A unified jackknife theory for empirical best prediction with M-estimation, Ann. Statist., 30, 1782-810.

Li, H. and Lahiri, P. (2010), An adjusted maximum likelihood method for solving small area estimation problems, Journal of Multivariate Analyis 101, 882-902.

Marhuenda, Y., Morales, D. and Pardo, M.C. (2014). Information criteria for Fay-Herriot model selection. Computational Statistics and Data Analysis 70, 268-280.

Neves, A., Silva, D. and Correa, S. (2013), Small domain estimation for the Brazilian service sector survey, ESTADISTICA 65(185), 13-37.

Prasad, N. and Rao, J. (1990), The estimation of the mean squared error of small-area estimation, Journal of the American Statistical Association 85(409), 163-171.

Petrucci, A., Salvati, N. (2006), Small Area Estimation for Spatial Correlation in Watershed Erosion Assessment, Journal of Agricultural, Biological and Environmental Statistics, 11(2), 169–182.

Rao, J. N. K. (2003), Small Area Estimation, New York: Wiley.

Rao, J. N. K. and Molina, I. (2015), Small area estimation, New York: Wiley.

Slud, E. and Maiti, T. (2006), Mean-squared error estimation in transformed Fay-Herriot models, Journal of the Royal Statistical Society:Series B 68(2), 239-257.

Warnholz, S. (2016), saeRobust: Robust small area estimation. R package.

Warnholz, S. (2016b). Small area estimation using robust extensions to area level models. Ph.D. thesis, Freie Universitaet Berlin.

Ybarra, L. and Lohr, S. (2008), Small area estimation when auxiliary information is measured with error, Biometrika, 95(4), 919-931.

Yoshimori, M. and Lahiri, P. (2014), A new adjusted maximum likelihood method for the Fay-Herriot small area model, Journal of Multivariate Analysis 124, 281-294.

Examples

```
# Loading data - population and sample data
data("eusilcA_popAgg")
data("eusilcA_smpAgg")

# Combine sample and population data
combined_data <- combine_data(
  pop_data = eusilcA_popAgg,
  pop_domains = "Domain",
  smp_data = eusilcA_smpAgg,
  smp_domains = "Domain"
)

# Example 1: Standard Fay-Herriot model and analytical MSE
fh_std <- fh(
  fixed = Mean ~ cash + self_empl, vardir = "Var_Mean",
  combined_data = combined_data, domains = "Domain", method = "ml",
  MSE = TRUE
)

# Example 2: arcsin transformation of the dependent variable
fh_arcsin <- fh(
  fixed = MTMED ~ cash + age_ben + rent + house_allow,
  vardir = "Var_MTMED", combined_data = combined_data, domains = "Domain",
  method = "ml", transformation = "arcsin", backtransformation = "bc",
  eff_smpsize = "n", MSE = TRUE, mse_type = "boot", B = c(50, 0)
)

# Example 3: Spatial Fay-Herriot model
# Load proximity matrix
data("eusilcA_prox")
fh_spatial <- fh(
```

```
  fixed = Mean ~ cash + self_empl, vardir = "Var_Mean",
  combined_data = combined_data, domains = "Domain", method = "reml",
  correlation = "spatial", corMatrix = eusilcA_prox, MSE = TRUE,
  mse_type = "analytical"
)

# Example 4: Robust Fay-Herriot model
fh_robust <- fh(
  fixed = Mean ~ cash + self_empl, vardir = "Var_Mean",
  combined_data = combined_data, domains = "Domain", method = "reblupbc",
  k = 1.345, mult_constant = 1, MSE = TRUE, mse_type = "pseudo"
)

# Example 5: Ybarra-Lohr model
# Create MSE array
P <- 1
M <- length(eusilcA_smpAgg$Mean)
Ci_array <- array(data = 0, dim = c(P + 1, P + 1, M))
for (i in 1:M) {
  Ci_array[2, 2, i] <- eusilcA_smpAgg$Var_Cash[i]
}
fh_yl <- fh(
  fixed = Mean ~ Cash, vardir = "Var_Mean",
  combined_data = eusilcA_smpAgg, domains = "Domain", method = "me",
  Ci = Ci_array, MSE = TRUE, mse_type = "jackknife"
)
```

`fixef` *Extract Fixed Effects from an emdi Object*

Description

Methods `fixef.ebp` and `fixef.fh` extract the fixed effects from an emdi object of class "ebp" or "fh".

Usage

```
## S3 method for class 'ebp'
fixef(object, ...)

## S3 method for class 'ebp'
fixed.effects(object, ...)

## S3 method for class 'fh'
fixef(object, ...)

## S3 method for class 'fh'
fixed.effects(object, ...)
```

Arguments

object	an object of type "emdi", depending on the used method either "ebp" or "fh".
...	additional arguments that are not used in this method.

Details

The alias fixed.effects can also be used instead of fixef. The generic function fixef is imported from package nlme and re-exported to make the S3-methods available, even though the nlme package itself is not loaded or attached. For default documentation, see fixed.effects.

Value

For classes "ebp" and "fh" a vector containing the fixed effects is returned.

See Also

ebp, fh, fixed.effects

Examples

```
# Example for class ebp
emdi_model <- ebp(
  fixed = eqIncome ~ gender + eqsize + cash + self_empl +
    unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent + fam_allow +
    house_allow + cap_inv + tax_adj, pop_data = eusilcA_pop,
  pop_domains = "district", smp_data = eusilcA_smp, smp_domains = "district",
  na.rm = TRUE
)

fixef(emdi_model)
```

---

getData *Extract emdi Object Data*

Description

Methods getData.direct, getData.ebp and getData.fh extract the data frame used to fit the model.

Usage

```
## S3 method for class 'direct'
getData(object, ...)

## S3 method for class 'ebp'
getData(object, ...)

## S3 method for class 'fh'
getData(object, ...)
```

Arguments

object	an object of type "emdi", depending on the method either "direct", "ebp" or "fh".
...	additional arguments that are not used in this method.

Details

The generic function getData is imported from package nlme and re-exported to make the S3-methods available, even though the nlme package itself is not loaded or attached. For default documentation, see getData.

Value

Data frame used to fit the model. For classes "direct" and "ebp" the (untransformed) sample data is returned. For class "fh" the combined data set is returned.

See Also

direct, ebp, fh, getData

Examples

```
# Example for class direct
emdi_direct <- direct(
  y = "eqIncome", smp_data = eusilcA_smp,
  smp_domains = "district", weights = "weight", threshold = 11064.82,
  var = TRUE, boot_type = "naive", B = 50, seed = 123, X_calib = NULL,
  totals = NULL, na.rm = TRUE
)

getData(emdi_direct)
```

---

getGroups *Extract Grouping Factors from an emdi Object*

Description

Methods getGroups.direct, getGroups.ebp and getGroups.fh extract grouping factors from an emdi object.

Usage

```
## S3 method for class 'direct'
getGroups(object, ...)

## S3 method for class 'ebp'
getGroups(object, ...)
```

```
## S3 method for class 'fh'
getGroups(object, ...)
```

Arguments

object	an object of type "emdi", depending on the method either "direct", "ebp" or "fh".
...	additional arguments that are not used in this method.

Details

The generic function getGroups is imported from package nlme and re-exported to make the S3-methods available, even though the nlme package itself is not loaded or attached. For default documentation, see getGroups.

Value

A vector containing the grouping factors.

See Also

direct, ebp, fh, getGroups

Examples

```
# Example for class direct
emdi_direct <- direct(
  y = "eqIncome", smp_data = eusilcA_smp,
  smp_domains = "district", weights = "weight", threshold = 11064.82,
  var = TRUE, boot_type = "naive", B = 50, seed = 123, X_calib = NULL,
  totals = NULL, na.rm = TRUE
)

getGroups(emdi_direct)
```

---

getGroupsFormula *Extract Grouping Formula from an emdi Object*

Description

Methods getGroupsFormula.direct, getGroupsFormula.ebp and getGroupsFormula.fh extract the grouping formula from an emdi object.

Usage

```
## S3 method for class 'direct'
getGroupsFormula(object, ...)

## S3 method for class 'ebp'
getGroupsFormula(object, ...)

## S3 method for class 'fh'
getGroupsFormula(object, ...)
```

Arguments

object	an object of type "emdi", depending on the method either "direct", "ebp" or "fh".
...	additional arguments that are not used in this method.

Details

The generic function getGroupsFormula is imported from package nlme and re-exported to make the S3-methods available, even though the nlme package itself is not loaded or attached. For default documentation, see getGroupsFormula.

Value

A one-sided formula.

See Also

direct, ebp, fh, getGroupsFormula

Examples

```
# Example for class ebp
emdi_model <- ebp(
  fixed = eqIncome ~ gender + eqsize + cash + self_empl +
    unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent + fam_allow +
    house_allow + cap_inv + tax_adj, pop_data = eusilcA_pop,
  pop_domains = "district", smp_data = eusilcA_smp, smp_domains = "district",
  na.rm = TRUE
)

getGroupsFormula(emdi_model)
```

getResponse *Extract Response Variable from an emdi Object*

Description

Methods `getResponse.direct`, `getResponse.ebp` and `getResponse.fh` extract the response variable from an emdi object.

Usage

```
## S3 method for class 'direct'
getResponse(object, ...)

## S3 method for class 'ebp'
getResponse(object, ...)

## S3 method for class 'fh'
getResponse(object, ...)
```

Arguments

`object`	an object of type "emdi", depending on the method either "direct", "ebp" or "fh".
`...`	additional arguments that are not used in this method.

Details

The generic function `getResponse` is imported from package nlme and re-exported to make the S3-methods available, even though the nlme package itself is not loaded or attached. For default documentation, see `getResponse`.

Value

Vector containing the response variable.

See Also

`direct`, `ebp`, `fh`, `getResponse`

Examples

```
# Example for class ebp
emdi_model <- ebp(
  fixed = eqIncome ~ gender + eqsize + cash + self_empl +
    unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent + fam_allow +
    house_allow + cap_inv + tax_adj, pop_data = eusilcA_pop,
  pop_domains = "district", smp_data = eusilcA_smp, smp_domains = "district",
  na.rm = TRUE
)
```

```
getResponse(emdi_model)
```

---

getVarCov *Extract Variance-covariance Matrix from an emdi Object*

Description

Methods getVarCov.ebp and getVarCov.fh extract the variance-covariance matrix from a fitted model of class "ebp" or "fh".

Usage

```
## S3 method for class 'ebp'
getVarCov(obj, individuals = 1, type = "random.effects", ...)

## S3 method for class 'fh'
getVarCov(obj, individuals = 1, type = "random.effects", ...)
```

Arguments

obj	an object of type "emdi", either "ebp" or "fh".
individuals	vector of levels of the in-sample domains can be specified for the types "conditional" or "marginal".
type	a character that determines the type of variance-covariance matrix. Types that can be chosen (i) random-effects variance-covariance matrix ("random.effects"), (ii) conditional variance-covariance matrix ("conditional"), (iii) marginal variance-covariance matrix ("marginal"). Defaults to "random.effects".
...	additional arguments that are not used in this method.

Details

The generic function getVarCov is imported from package nlme and re-exported to make the S3-methods available, even though the nlme package itself is not loaded or attached. For default documentation, see getVarCov.

Value

A variance-covariance matrix or a list of variance-covariance matrices, if more than one individual is selected. For method getVarCov.ebp, the dimensions of the matrices are 1 x 1 for type "random.effects" and number of in-sample domains x number of in-sample domains for types "conditional" and "marginal". For method getVarCov.fh, for all types the dimensions of the matrices are 1 x 1. For type "marginal" the diagonal elements of the variance covariances matrices are returned for the chosen individual. Please note, if the correlation argument of the "fh" object is set to spatial, the variance covariance matrix has non-zero off-diagonal elements, because the assumption of independence of the error terms does not hold. For the non-spatial models, the off-diagonal elements are zero.

See Also

ebp, fh, getVarCov

Examples

```
# Example for class fh
combined_data <- combine_data(
  pop_data = eusilcA_popAgg,
  pop_domains = "Domain",
  smp_data = eusilcA_smpAgg,
  smp_domains = "Domain"
)

fh_std <- fh(
  fixed = Mean ~ cash + self_empl, vardir = "Var_Mean",
  combined_data = combined_data, domains = "Domain",
  method = "ml", MSE = TRUE
)

getVarCov(fh_std)
```

---

intervals *Confidence Intervals on Coefficients of an emdi Object*

Description

Methods `intervals.ebp` and `intervals.fh` provide the approximate confidence intervals on the coefficients (fixed effects) of an emdi object.

Usage

```
## S3 method for class 'ebp'
intervals(object, level = 0.95, parm = NULL, ...)

## S3 method for class 'fh'
intervals(object, level = 0.95, parm = NULL, ...)
```

Arguments

`object`	an object of type "emdi", depending on the method either "ebp" or "fh".
`level`	an optional numeric value with the confidence level for the intervals. Defaults to 0.95.
`parm`	vector of names to specify which parameters are to be given confidence intervals. If `NULL`, all parameters are taken into account. Defaults to `NULL`.
`...`	additional arguments that are not used in this method.

Details

The generic function `intervals` is imported from package nlme and re-exported to make the S3-methods available, even though the nlme package itself is not loaded or attached. For default documentation, see `intervals`.

Value

A matrix with rows corresponding to the parameters and columns containing the lower confidence limits (lower), the estimated values (est.), and upper confidence limits (upper).

See Also

`direct`, `ebp`, `fh`, `intervals`

Examples

```
# Example for class ebp
emdi_model <- ebp(
  fixed = eqIncome ~ gender + eqsize + cash + self_empl +
    unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent + fam_allow +
    house_allow + cap_inv + tax_adj, pop_data = eusilcA_pop,
  pop_domains = "district", smp_data = eusilcA_smp, smp_domains = "district",
  na.rm = TRUE
)

intervals(emdi_model)
```

---

load_shapeaustria *Loading the Shape File for Austrian Districts*

Description

The function simplifies to load the shape file for austrian districts.

Usage

```
load_shapeaustria()
```

Details

The shape file contains the borders of Austrian districts. Thus, it can be used for the visualization of estimation results for Austrian districts.

Value

A shape file of class `SpatialPolygonsDataFrame`.

map_plot *Visualizes Regional Disaggregated Estimates on a Map*

Description

Function map_plot creates spatial visualizations of the estimates obtained by small area estimation methods or direct estimation.

Usage

```
map_plot(
  object,
  indicator = "all",
  MSE = FALSE,
  CV = FALSE,
  map_obj = NULL,
  map_dom_id = NULL,
  map_tab = NULL,
  color = c("white", "red4"),
  scale_points = NULL,
  guide = "colourbar",
  return_data = FALSE
)
```

Arguments

object	an object of type emdi, containing the estimates to be visualized.
indicator	optional character vector that selects which indicators shall be returned: (i) all calculated indicators ("all"); (ii) each indicator name: "Mean", "Quantile_10", "Quantile_25", "Median", "Quantile_75", "Quantile_90", "Head_Count", "Poverty_Gap", "Gini", "Quintile_Share" or the function name/s of "custom_indicator/s"; (iii) groups of indicators: "Quantiles", "Poverty" or "Inequality". Note, additional custom indicators can be defined as argument for model-based approaches (see also ebp) and do not appear in groups of indicators even though these might belong to one of the groups. If the model argument is of type "fh", indicator can be set to "all", "Direct", FH", or "FH_Bench" (if emdi object is overwritten by function benchmark). Defaults to "all".
MSE	optional logical. If TRUE, the MSE is also visualized. Defaults to FALSE.
CV	optional logical. If TRUE, the CV is also visualized. Defaults to FALSE.
map_obj	an sf, data.frame object as defined by the **sf** package on which the data should be visualized. The typical example is polygon shapefile object
map_dom_id	a character string containing the name of a variable in map_obj that indicates the domains.

map_tab	a data.frame object with two columns that match the domain variable from the census data set (first column) with the domain variable in the map_obj (second column). This should only be used if the IDs in both objects (‘map_obj‘ and ‘object‘) differ.
color	a vector of length 2 defining the lowest and highest color in the plots.
scale_points	a structure defining the lowest and the highest value of the colorscale. If a numeric vector of length two is given, this scale will be used for every plot.
guide	character passed to scale_colour_gradient from **ggplot2**. Possible values are "none", "colourbar", and "legend".
return_data	if set to TRUE, a fortified data frame including the map data as well as the chosen indicators is returned. Customized maps can easily be obtained from this data frame via the package **ggplot2**. Defaults to FALSE.

Value

Creates the plots demanded, and, if selected, a fortified data.frame containing the mapdata and chosen indicators.

See Also

direct, ebp, fh, emdiObject

Examples

```
data("eusilcA_pop")
data("eusilcA_smp")

# Generate emdi object with additional indicators; here via function ebp()
emdi_model <- ebp(
  fixed = eqIncome ~ gender + eqsize + cash +
    self_empl + unempl_ben + age_ben + surv_ben + sick_ben +
    dis_ben + rent + fam_allow + house_allow + cap_inv +
    tax_adj, pop_data = eusilcA_pop,
  pop_domains = "district", smp_data = eusilcA_smp,
  smp_domains = "district", threshold = 11064.82,
  transformation = "box.cox", L = 50, MSE = TRUE, B = 50
)

# Load shape file
load_shapeaustria()

# Create map plot for mean indicator - point and MSE estimates but no CV
map_plot(
  object = emdi_model, MSE = TRUE, CV = FALSE,
  map_obj = shape_austria_dis, indicator = c("Mean"),
  map_dom_id = "PB"
)

# Create a suitable mapping table to use numerical identifiers of the shape
# file
```

```
# First find the right order
dom_ord <- match(shape_austria_dis$PB, emdi_model$ind$Domain)

# Create the mapping table based on the order obtained above
map_tab <- data.frame(
  pop_data_id = emdi_model$ind$Domain[dom_ord],
  shape_id = shape_austria_dis$BKZ
)

# Create map plot for mean indicator - point and CV estimates but no MSE
# using the numerical domain identifiers of the shape file

map_plot(
  object = emdi_model, MSE = FALSE, CV = TRUE,
  map_obj = shape_austria_dis, indicator = c("Mean"),
  map_dom_id = "BKZ", map_tab = map_tab
)
```

`plot.emdi` *Plots for an emdi Object*

Description

Diagnostic plots of the underlying model in the EBP (see also `ebp`) or Fay-Herriot (see also `fh`) approaches are obtained. These include Q-Q plots and density plots of residuals and random effects from the nested error linear regression model/ the Fay-Herriot model, a Cook's distance plot for detecting outliers and the log-likelihood of the estimation of the optimal parameter in Box-Cox transformations (the latter two only for ebp). The return depends on the transformation such that a plot for the optimal parameter is only returned in case a transformation with transformation parameter is chosen. The range of the x-axis is optional but necessary to change if there are convergence problems. All plots are obtained by `ggplot`.

Usage

```
## S3 method for class 'emdi'
plot(
  x,
  label = "orig",
  color = c("blue", "lightblue3"),
  gg_theme = NULL,
  cooks = TRUE,
  range = NULL,
  ...
)

## S3 method for class 'ebp'
```

```
plot(
  x,
  label = "orig",
  color = c("blue", "lightblue3"),
  gg_theme = NULL,
  cooks = TRUE,
  range = NULL,
  ...
)

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

## S3 method for class 'fh'
plot(
  x,
  label = "orig",
  color = c("blue", "lightblue3"),
  gg_theme = NULL,
  cooks = TRUE,
  range = NULL,
  ...
)
```

Arguments

x	an object of type "emdi", either "ebp" or "fh", representing point and, if chosen, MSE estimates obtained by the EBP or Fay-Herriot approach (see also ebp and fh).
label	argument that enables to customize title and axis labels. There are three instant options to label the diagnostic plot: (i) original labels ("orig"), (ii) axis labels but no title ("no_title"), (iii) neither axis labels nor title ("blank"). (iv) individual labels by a list that needs to have below structure. Six elements can be defined called `qq_res`, `qq_ran`, `d_res`, `d_ran`, `cooks` and `opt_lambda` for the six different plots and these list elements need to have three elements each called `title`, `y_lab` and `x_lab`. Only the labels for the plots that should be different to the original need to be specified. Please see the details section for an example with the default labels.
color	a character vector with two elements. The first element defines the color for the line in the QQ-plots, for the Cook's Distance plot and for the Box-Cox plot. The second element defines the color for the densities.
gg_theme	theme list from package **ggplot2**. For using this argument, package **ggplot2** must be loaded via `library(ggplot2)`. See also Example 4.
cooks	if `TRUE`, a Cook's distance plot is returned when the ebp function is used. The used method `mdffits.default` from the package **HLMdiag** struggles when data sets get large. In these cases, `cooks` should be set to `FALSE`. It defaults to `TRUE`.

range	optional sequence determining the range of the x-axis for plots of the optimal transformation parameter that defaults to NULL. In that case a range of the default interval is used for the plots of the optimal parameter. This leads in some cases to convergence problems such that it should be changed to e.g. the selected interval. The default value depends on the chosen data driven transformation and equals the default interval for the estimation of the optimal parameter.
...	optional arguments passed to generic function.

Details

The default settings of the label argument are as follows (please note that the title for opt_lambda depends on the chosen transformation, for the example Box-Cox is shown):

list(

qq_res = c(title="Error term", y_lab="Quantiles of pearson residuals", x_lab="Theoretical quantiles"),

qq_ran = c(title="Random effect", y_lab="Quantiles of random effects", x_lab="Theoretical quantiles"),

d_res = c(title="Density - Pearson residuals", y_lab="Density", x_lab="Pearson residuals"),

d_ran = c(title="Density - Standardized random effects", y_lab="Density", x_lab="Standardized random effects"),

cooks = c(title="Cook's Distance Plot", y_lab="Cook's Distance", x_lab="Index"),

opt_lambda = c(title="Box-Cox - REML", y_lab="Log-Likelihood", x_lab="expression(lambda)"))

Value

Two Q-Q plots in one grid, two density plots, a Cook's distance plot and a likelihood plot for the optimal parameter of transformations with transformation parameter obtained by ggplot. The latter two plots are only provided for ebp object.

See Also

emdiObject, ebp, fh

Examples

```
# Examples for models of type ebp
# Loading data - population and sample data
data("eusilcA_pop")
data("eusilcA_smp")

# With default setting but na.rm = TRUE; with Box-Cox transformation
emdi_model <- ebp(
  fixed = eqIncome ~ gender + eqsize + cash + self_empl +
    unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent + fam_allow +
    house_allow + cap_inv + tax_adj, pop_data = eusilcA_pop,
  pop_domains = "district", smp_data = eusilcA_smp, smp_domains = "district",
  na.rm = TRUE
```

```
)

# Example 1: Creation of default diagnostic plots
plot(emdi_model)

# Example 2: Creation of diagnostic plots without labels and titles,
# different colors and without Cook's distance plot.
plot(emdi_model,
  label = "no_title", color = c("red", "yellow"),
  cooks = FALSE
)

# Example 3: Creation of diagnostic plots where labels and title differs for
# residual plot
plot(emdi_model,
  label = list(qq_res = c(
    title = "Pearson resid.",
    y_lab = "Quant.", x_lab = "Theo. Quant."
  )), color = c("red", "yellow"),
  cooks = FALSE
)

# Example 4: Usage of theme from ggplot2 within plot.emdi
library(ggplot2)
plot(emdi_model, gg_theme = theme(
  panel.background =
    element_rect(fill = "white", colour = "white"),
  plot.title = element_text(face = "bold"),
  title = element_text(color = "navy")
))

# Example for models of type fh

# Loading data - population and sample data
data("eusilcA_popAgg")
data("eusilcA_smpAgg")

# Combine sample and population data
combined_data <- combine_data(
  pop_data = eusilcA_popAgg,
  pop_domains = "Domain",
  smp_data = eusilcA_smpAgg,
  smp_domains = "Domain"
)

# Generation of the emdi object
fh_std <- fh(
  fixed = Mean ~ cash + self_empl, vardir = "Var_Mean",
  combined_data = combined_data, domains = "Domain",
  method = "ml", MSE = TRUE
)

# Example 5: Creation of default diagnostic plots for Fay-Herriot model
```

```
plot(fh_std)
```

`predict.emdi` *Predictions from emdi Objects*

Description

Method `predict.emdi` extracts the direct estimates, the empirical best linear unbiased or empirical best predictors for all domains from an emdi object.

Usage

```
## S3 method for class 'emdi'
predict(object, ...)
```

Arguments

`object`	an object of type "emdi".
`...`	additional arguments that are not used in this method.

Details

For a better selection of prediction results, it is referred to use the generic function `estimators`. The methods for object of class "emdi" allows to select among the indicators of interest.

Value

Data frame with domain predictors.

See Also

`direct`, `ebp`, `fh`

Examples

```
# Example for class ebp
emdi_model <- ebp(
  fixed = eqIncome ~ gender + eqsize + cash + self_empl +
    unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent + fam_allow +
    house_allow + cap_inv + tax_adj, pop_data = eusilcA_pop,
  pop_domains = "district", smp_data = eusilcA_smp, smp_domains = "district",
  na.rm = TRUE
)

predict(emdi_model)
```

qqnorm.emdi *Quantile-quantile Plots for an emdi Object*

Description

Normal quantile-quantile plots of the underlying model in the EBP (see also ebp) or Fay-Herriot (see also fh) approaches are obtained. The plots are obtained by ggplot.

Usage

```
## S3 method for class 'emdi'
qqnorm(y, color = c("blue", "lightblue3"), gg_theme = NULL, ...)

## S3 method for class 'ebp'
qqnorm(y, color = c("blue", "lightblue3"), gg_theme = NULL, ...)

## S3 method for class 'fh'
qqnorm(y, color = c("blue", "lightblue3"), gg_theme = NULL, ...)

## S3 method for class 'direct'
qqnorm(y, ...)
```

Arguments

y	a model object of type "emdi", either "ebp" or "fh", representing point and, if chosen, MSE estimates obtained by the EBP or Fay-Herriot approach (see also ebp and fh).
color	a character vector with two elements. The first element defines the color for the line in the QQ-plots, for the Cook's Distance plot and for the Box-Cox plot. The second element defines the color for the densities.
gg_theme	theme list from package **ggplot2**. For using this argument, package **ggplot2** must be loaded via library(ggplot2). See also Example 4.
...	optional arguments passed to generic function.

Value

Two Q-Q plots in one grid obtained by ggplot.

See Also

emdiObject, ebp, fh

Examples

```
# Examples for models of type ebp
# Loading data - population and sample data
data("eusilcA_pop")
data("eusilcA_smp")

# With default setting but na.rm = TRUE; with Box-Cox transformation
emdi_model <- ebp(
  fixed = eqIncome ~ gender + eqsize + cash + self_empl +
    unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent + fam_allow +
    house_allow + cap_inv + tax_adj, pop_data = eusilcA_pop,
  pop_domains = "district", smp_data = eusilcA_smp, smp_domains = "district",
  na.rm = TRUE
)

# Example 1: Creation of default diagnostic plots
qqnorm(emdi_model)

# Example for models of type fh

# Loading data - population and sample data
data("eusilcA_popAgg")
data("eusilcA_smpAgg")

# Combine sample and population data
combined_data <- combine_data(
  pop_data = eusilcA_popAgg,
  pop_domains = "Domain",
  smp_data = eusilcA_smpAgg,
  smp_domains = "Domain"
)

# Generation of the emdi object
fh_std <- fh(
  fixed = Mean ~ cash + self_empl, vardir = "Var_Mean",
  combined_data = combined_data, domains = "Domain",
  method = "ml", MSE = TRUE
)

# Example 5: Creation of default diagnostic plots for Fay-Herriot model
qqnorm(fh_std)
```

---

ranef *Extract Random Effects of emdi Objects*

Description

Methods `ranef.ebp` and `ranef.fh` extract the fixed effects from an emdi object of class "ebp" or "fh".

Usage

```
## S3 method for class 'ebp'
ranef(object, ...)

## S3 method for class 'ebp'
random.effects(object, ...)

## S3 method for class 'fh'
ranef(object, ...)

## S3 method for class 'fh'
random.effects(object, ...)
```

Arguments

object	an object of type "emdi", depending on the used method either "ebp" or "fh".
...	additional arguments that are not used in this method.

Details

The alias random.effects can also be used instead of ranef. The generic function ranef is imported from package nlme and re-exported to make the S3-methods available, even though the nlme package itself is not loaded or attached. For default documentation, see random.effects.

Value

A vector containing the estimated random effects at domain level is returned.

See Also

ebp, fh, random.effects

Examples

```
# Example for class ebp
emdi_model <- ebp(
  fixed = eqIncome ~ gender + eqsize + cash + self_empl +
    unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent + fam_allow +
    house_allow + cap_inv + tax_adj, pop_data = eusilcA_pop,
  pop_domains = "district", smp_data = eusilcA_smp, smp_domains = "district",
  na.rm = TRUE
)

ranef(emdi_model)
```

`spatialcor.tests` *Spatial Autocorrelation Tests*

Description

This function computes two spatial autocorrelation tests: Moran's I and Geary's C.

Usage

```
spatialcor.tests(direct, corMatrix)
```

Arguments

`direct`	a vector containing direct estimates. The elements of `direct` must be sorted like the elements in `corMatrix`.
`corMatrix`	matrix or data frame with dimensions number of areas times number of areas containing the row-standardized proximities between the domains. Values must lie between `0` and `1`. The columns and rows must be sorted like the domains in `direct`.

Details

When creating the proximity matrix `corMatrix`, please make sure that the elements of `direct` and `corMatrix` are sorted equally and that `direct` and `corMatrix` do not contain any `NA`s. For a description of how to create the proximity matrix, see the package vignette "A Framework for Producing Small Area Estimates Based on Area-Level Models in R". If direct estimates do not exist for every area contained in the proximity matrix, the proximity matrix needs to be subsetted to the areas contained in the direct vector.

Value

The values of the test statistics and their corresponding p values.

References

Bivand, R. (2019), spdep: Spatial Dependence: Weighting Schemes, Statistics. R package.

Examples

```
# Loading data - sample data and proximity matrix
data("eusilcA_smpAgg")
data("eusilcA_prox")

# Compute spatial correlation tests
spatialcor.tests(
  direct = eusilcA_smpAgg$Mean,
  corMatrix = eusilcA_prox
)
```

step *Step Function*

Description

This generic function selects a model by different criteria in a stepwise algorithm.

Usage

```
step(object, scope, criteria, direction, trace, steps, ...)

## Default S3 method:
step(object, ...)

## S3 method for class 'fh'
step(
  object,
  scope = NULL,
  criteria = "AIC",
  direction = "both",
  trace = TRUE,
  steps = 1000,
  ...
)
```

Arguments

object	an object of type "fh" that contains the chosen information criterion or of type "lm" for the default method.
scope	formula or a list including two formulas (lower and upper) specifying the models considered in the step function. Defaults to NULL.
criteria	a character string describing the model selection criterion. Criteria that can be chosen are "AIC", "AICc", "AICb1", "AICb2", "BIC", "KIC", "KICc", "KICb1", or "KICb2". Defaults to "AIC".
direction	a character string describing the direction of stepwise algorithm. Directions that can be chosen are "both", "backward" or "forward". Defaults to "both". If no scope argument is provided, the default is "backward".
trace	if TRUE, information about the single steps is provided during the stepwise procedure. Defaults to TRUE.
steps	a number determining the maximum number of steps. Defaults to 1000.
...	additional arguments that are not used in this method.

Details

The default method of the generic function step applies the step function for lm models of the stats package. Please refer to the documentation of the step function of the stats package for details.

The information criteria "AICc", "AICb1", "AICb2", "KIC", "KICc", "KICb1" and "KICb2" are especially developed for Fay-Herriot models by *Marhuenda et al. (2014)*. They are based on a bootstrap algorithm. If one of the criteria is chosen, make sure that the bootstrap iterations (B) of the "fh" object are set to a positive number. For some model extensions of the Fay-Herriot model only the "AIC" and the "BIC" information criteria are provided and for some none of the information criteria are defined. Check the model_select component of the "fh" object (objectname$model$model_select). If no criteria are provided, it is not possible to apply the stepwise variable selection algorithm.

Value

The return of step depends on the class of its argument. Please refer to the documentation of the step function of the stats package for details of the default method.

For the fh method information about the resulting "best" model due to the chosen information criterion is provided:

call	the function call that produced the object.
coefficients	data frame containing the estimated regression coefficients, the standard errors and the t- and p-values of the explanatory variables.

References

Marhuenda, Y., Morales, D. and Pardo, M.C. (2014). Information criteria for Fay-Herriot model selection. Computational Statistics and Data Analysis 70, 268-280.

See Also

step

emdiObject, fh

Examples

```
# Loading data - population and sample data
data("eusilcA_popAgg")
data("eusilcA_smpAgg")

# Combine sample and population data
combined_data <- combine_data(
  pop_data = eusilcA_popAgg,
  pop_domains = "Domain",
  smp_data = eusilcA_smpAgg,
  smp_domains = "Domain"
)

# Estimate FH model that contains all variables that should be considered
fh_std <- fh(
  fixed = Mean ~ cash + self_empl + unempl_ben,
```

```
  vardir = "Var_Mean", combined_data = combined_data,
  domains = "Domain", method = "ml", B = c(0, 50)
)

# Example 1: Use default settings
step(fh_std)

# Example 2: Choose "KICb2" information criterion
step(fh_std, criteria = "KICb2")
```

`write.excel` *Exports an emdiObject to an Excel File or OpenDocument Spreadsheet*

Description

Function `write.excel` enables the user to export point and MSE estimates as well as diagnostics from the `summary` to an Excel file. The user can choose if the results should be reported in one or several Excel sheets. Furthermore, a selection of indicators can be specified. Respectively the function `write.ods` enables the export to OpenDocument Spreadsheets. Note that while `write.exel` will create a single document `write.ods` will create a group of files.

Usage

```
write.excel(
  object,
  file = NULL,
  indicator = "all",
  MSE = FALSE,
  CV = FALSE,
  split = FALSE,
  model = FALSE
)

write.ods(
  object,
  file = NULL,
  indicator = "all",
  MSE = FALSE,
  CV = FALSE,
  split = FALSE
)
```

Arguments

`object`	an object of type "emdi", representing point and MSE estimates.

file	path and filename of the spreadsheet to create. It should end on .xlsx or .ods respectively.
indicator	optional character vector that selects which indicators shall be returned: (i) all calculated indicators ("all"); (ii) each indicator name: "Mean", "Quantile_10", "Quantile_25", "Median", "Quantile_75", "Quantile_90", "Head_Count", "Poverty_Gap", "Gini", "Quintile_Share" or the function name/s of "custom_indicator/s"; (iii) groups of indicators: "Quantiles", "Poverty" or "Inequality". Note, additional custom indicators can be defined as argument for model-based approaches (see also ebp) and do not appear in groups of indicators even though these might belong to one of the groups. If the model argument is of type "fh", indicator can be set to "all", "Direct", FH", or "FH_Bench" (if emdi object is overwritten by function benchmark). Defaults to "all".
MSE	logical. If TRUE, the MSE of the emdiObject is exported. Defaults to FALSE.
CV	logical. If TRUE, the CV of the emdiObject is exported. Defaults to FALSE.
split	logical. If TRUE, point estimates, MSE and CV are written to different sheets in the Excel file. In write.ods TRUE will result in different files for point estimates and their precisions. Defaults to FALSE.
model	logical if TRUE, the estimation model is exported. #' Defaults to FALSE.

Details

These functions create an Excel file via the package **openxlsx** and ODS files via the package **readODS**. Both packages require a zip application to be available to R. If this is not the case the authors of **openxlsx** suggest the first of the following two ways.

- Install Rtools from: http://cran.r-project.org/bin/windows/Rtools/ and modify the system PATH during installation.
- If Rtools is installed, but no system path variable is set. One can set such a variable temporarily to R by a command like: `Sys.setenv("R_ZIPCMD" = "PathToTheRToolsFolder/bin/zip.exe")`.

To check if a zip application is available they recommend the command `shell("zip")`.

Value

An Excel file is created in your working directory, or at the given path. Alternatively multiple ODS files are created at the given path.

See Also

direct, emdiObject, ebp, fh

Examples

```
# Loading data - population and sample data
data("eusilcA_pop")
data("eusilcA_smp")

# Generate emdi object with two additional indicators; here via function
```

```
# ebp()
emdi_model <- ebp(
  fixed = eqIncome ~ gender + eqsize + cash +
    self_empl + unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent +
    fam_allow + house_allow + cap_inv + tax_adj, pop_data = eusilcA_pop,
  pop_domains = "district", smp_data = eusilcA_smp, smp_domains = "district",
  threshold = function(y) {
    0.6 * median(y)
  }, L = 50, MSE = TRUE, B = 50,
  custom_indicator = list(
    my_max = function(y) {
      max(y)
    },
    my_min = function(y) {
      min(y)
    }
  ), na.rm = TRUE, cpus = 1
)

# Example 1: Export estimates for all indicators and uncertainty measures
# and diagnostics to Excel
write.excel(emdi_model, indicator = "all",  MSE = TRUE, CV = TRUE)

# Example 2: Single Excel sheets for point, MSE and CV estimates
write.excel(emdi_model, indicator = "all", MSE = TRUE, CV = TRUE,
            split = TRUE)

# Example 3: Same as example 1 but for an ODS output, skipped due to lack of zip app
# write.ods(emdi_model, indicator = "all", MSE = TRUE, CV = TRUE)
```

`wtd.quantile` *Quick function to estimate weighted quantiles*

Description

Quick function to estimate weighted quantiles

Usage

```
wtd.quantile(x, weights = NULL, probs = NULL)
```

Arguments

`x`	a numeric vector
`weights`	a numeric vector for the weights
`probs`	probabilities

Value

weighted quantile

Index