Package ‘emdi’

October 10, 2018

Title Estimating and Mapping Disaggregated Indicators

Version 1.1.4

Date 2018-10-10

Description Functions that support estimating, assessing and mapping regional disaggregated indicators. So far, estimation methods comprise direct estimation and the model-based approach Empirical Best Prediction (see ``Small area estimation of poverty indicators'' by Molina and Rao (2010) <doi:10.1002/cjs.10051>), as well as their precision estimates. 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.

Depends R (>= 3.3.0)

License GPL-2

URL https://github.com/SoerenPannier/emdi

LazyData true

Copyright inst/COPYRIGHTS

RoxygenNote 6.1.0

Imports nlme, moments, ggplot2, MuMIn, gridExtra, openxlsx, reshape2, graphics, stats, parallelMap, HLMdiag, parallel, boot, rgeos, maptools, MASS, readODS

Suggests testthat, R.rsp, simFrame, laeken

VignetteBuilder R.rsp

NeedsCompilation no

Author Ann-Kristin Kreutzmann [aut], Soeren Pannier [aut, cre], Natalia Rojas-Perilla [aut], Timo Schmid [aut], Matthias Templ [aut], Nikos Tzavidis [aut]

Maintainer Soeren Pannier <soeren.pannier@fu-berlin.de>

Repository CRAN

Date/Publication 2018-10-10 17:10:07 UTC

R topics documented:

`as.data.frame.estimators.emdi`
Transforms estimators.emdi objects into a dataframe object

Description

Transforms estimators.emdi objects into a dataframe object

Usage

```
## S3 method for class 'estimators.emdi'
as.data.frame(x, ...)
```

Arguments

`x`	an object of type "estimators.emdi".
`...`	further arguments passed to or from other methods.

`as.matrix.estimators.emdi`
Transforms estimators.emdi objects into a matrix object

Description

Transforms estimators.emdi objects into a matrix object

Usage

```
## S3 method for class 'estimators.emdi'
as.matrix(x, ...)
```

Arguments

`x`	an object of type "estimators.emdi".
`...`	further arguments passed to or from other methods.

`compare_plot` *Shows plots for the comparison of direct and model-based estimates*

Description

For all indicators or a selection of indicators two plots are returned. The first plot is a scatter plot of the direct and model-based point estimates and the second is a line plot with both point estimates.

Usage

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

Arguments

`direct`	an object of type "emdi","direct", representing point and MSE estimates.
`model`	an object of type "emdi","model", representing point and 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. Defaults to "all". 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.

`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 lables but no title ("no_title"), (iii) neither axis labels nor title ("blank").

`color` a vector with two elements. The first color determines the color of the line in the scatter plot and the color for the direct estimates in the line plot. The second color specifies the color of the line for the model-based estimates.

`shape` a numeric vector with two elements. The first shape determines the shape of the points in the line plot for the direct estimates and the second shape for the model-based estimates. The options are numbered from 0 to 25.

`line_type` a character vector with two elements. The first line type determines the type of the line for the direct estimates and the second type for the model-based estimates. The options are: "twodash", "solid", "longdash", "dotted", "dotdash", "dashed" and "blank".

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

Value

A scatter plot and a line plot comparing direct and model-based estimators for each selected indicator obtained by `ggplot`.

See Also

`emdiObject`, `direct`, `ebp`

Examples

```
## Not run:
# 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(direct = emdi_direct, model = emdi_model)

# Example 2: Change plot theme
library(ggplot2)
compare_plot(emdi_direct, emdi_model, indicator = "Median",
```

```
gg_theme = theme(axis.line = element_line(size = 3, colour = "grey80"),
plot.background = element_rect(fill = "lightblue3"),
legend.position = "none"))

## End(Not run)
```

---

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 three types can be chosen, particularly no, natural log and Box-Cox 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. Three 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").
lambda	a scalar parameter that determines the Box-Cox transformation. In case of no and natural log transformation lambda can be set to NULL.

Details

For the natural log and Box-Cox 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)
```

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` 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.

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`.

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 "emdi" that provides direct estimators for regional disaggregated indicators and optionally corresponding variance estimates. Generic functions such as `estimators`, `print` and `summary` have methods that can be used to obtain further information. See `emdiObject` for descriptions of components of objects of class "emdi".

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, print.emdi, summary.emdi

Examples

```
## Not run:
# Loading sample data
data("eusilcA_smp")

# Example 1: Without 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 * laeken::weightedMedian(y, weights)}, 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)

## End(Not run)
```

---

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 three different transformation types for the dependent variable can be chosen.

Usage

```
ebp(fixed, pop_data, pop_domains, smp_data, smp_domains, L = 50,
  threshold = NULL, transformation = "box.cox", interval = c(-1, 2),
  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)
```

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. Three different transformation types for the dependent variable can be chosen (i) no transformation ("no"); (ii) log transformation ("log"); (iii) Box-Cox transformation ("box.cox"). Defaults to `"box.cox"`.

`interval` 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 c(-1,2). 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 and must only depend on the target variable `y` 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`.

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.

Value

An object of class "emdi" that provides estimators for regional disaggregated indicators and optionally corresponding MSE estimates. Generic functions such as `estimators`, `print`, `plot` and `summary` have methods that can be used to obtain further information. See `emdiObject` for descriptions of components of objects of class "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.

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.

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.

See Also

emdiObject, lme, estimators.emdi, print.emdi, plot.emdi, summary.emdi

Examples

```
## Not run:
# 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, threshold){max(y)},
my_min = function(y, threshold){min(y)}), na.rm = TRUE, cpus = 1)

## End(Not run)
```

---

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 and the Empirical Best Prediction approach by *Molina and Rao (2010)* are provided. Estimates of the mean squared error for these methods can be conducted by using a parametric bootstrap approach (*Gonzalez-Manteiga et al. 2008*). Furthermore, a mapping tool for plotting the estimates on their geographic regions is provided. Point and uncertainty measures as well as diagnostic tests can be easily extracted to Excel.

Details

The two estimation functions are called `direct` and `ebp`. For both functions several methods are available as `estimators.emdi`, `plot.emdi` (only for emdi objects obtained by function ebp), `print.emdi` and `summary.emdi`. 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.

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.

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.

`emdiObject` *Fitted emdiObject*

Description

An object of class emdi that represents point predictions of regional disaggregated indicators. Optionally it also contains corresponding MSE estimates. Depending on the estimation the object is also of class direct or model. Objects of these classes have methods for the generic functions `estimators`, `print`, `plot` (only for class model) and `summary`.

Value

The following components are always included in an emdi object:

`call` a list containing an image of the function call that produced the object.

`fixed` a formula of fixed effects used in the nested error linear regression (see also `fixed` in `ebp`). Not filled for class direct.

`framework` a list with following components:

`N_dom_smp`	number of domains in the sample.
`N_dom_unobs`	number of out-of-sample domains. Not filled for class direct.
`N_pop`	total number of units in population. Not filled for class direct.
`N_smp`	total number of units in sample.
`pop_domains_vec`	an arranged vector of the domain indicator variable. Not filled for class direct.

smp_data	an arranged data set of sample data. Not filled for class direct.
smp_domains	a character naming the domain indicator variable.
smp_domains_vec	an arranged vector of the domain indicator variable.

ind	data frame containing estimates for indicators per domain.
method	character returning the method for estimation of the optimal lambda, here "reml". Not filled for class direct.
model	an object returned by the lme function of type "lme" and representing a fitted linear mixed-effects model (for further explanations see lme and lmeObject). 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 to be TRUE in function call. If FALSE, MSE is NULL.
transformation	character returning the selected transformation type (see also transformation in ebp). 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 Box-Cox transformation 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 direct.
successful_bootstraps	a matrix with domains as rows and indicators as columns. The cells contain the number of successful bootstraps for each combination. Not filled for class model.

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.

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, lme, lmeObject

---

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).

Usage

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

Arguments

- `object` an object for which point and/or MSE estimates and/or calculated CV's are desired.
- `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. Defaults to "all". 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.
- `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`.
- `...` arguments to be passed to or from other methods, e.g. indicator.

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.

`estimators.emdi` *Presents point, MSE and/or CV estimates of an emdiObject*

Description

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

```
## 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. Defaults to "all". 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.
- `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`.

Value

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

Examples

```
## Not run:
# 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, threshold){max(y)},
my_min = function(y, threshold){min(y)}), na.rm = TRUE, cpus = 1)

# Example 1: Choose Gini coefficient, MSE and CV
estimators(emdi_model, indicator = "Gini", MSE = TRUE, CV = TRUE)

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

```
## End(Not run)
```

`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).

`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.

`head.estimators.emdi` *Returns the first part of predicted indicators and, if chosen, of MSE and CV estimators.*

Description

Returns the first part of predicted indicators and, if chosen, of MSE and CV estimators.

Usage

```
## S3 method for class 'estimators.emdi'
head(x, n = 6L, addrownums = NULL, ...)
```

Arguments

`x`	an object of type "estimators.emdi", representing point estimators and, if chosen, MSE and/or CV estimates for selected indicators.
`n`	a single integer. If positive, it determines the number of rows for the data frame. If negative, all but the n last rows of elements of the object.
`addrownums`	if there are no row names, create them from the row numbers.
`...`	arguments to be passed to or from other methods.

Value

Selected rows of the object of type "estimators.emdi".

See Also

`estimators.emdi`

Examples

```
## Not run:
# Loading data - population and sample data
data("eusilcA_pop")
data("eusilcA_smp")

# Generate emdi object with deleting missing values; 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",
na.rm = TRUE)

# Example: Choose first lines of the Gini coefficient, MSE and CV
head(estimators(emdi_model, indicator = c("Gini", "Head_Count")))

## End(Not run)
```

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 disaggregrated 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". Defaults to "all". 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.

MSE	optional logical. If TRUE, the MSE is also visualized.
CV	optional logical. If TRUE, the CV is also visualized.
map_obj	an SpatialPolygonsDataFrame object as defined by the **sp** package on which the data should be visualized.
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 differ.
color	a vector of length 2 defining the lowest and highest color in the plots.
scale_points	a structure defining the lowest, the mid and the highest value of the colorscale. If a numeric vector of length two is given, this scale will be used for every plot. Alternatively a list defining colors for each plot separately may be given. Please see the details section and examples for this.
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

ebp, emdiObject, readShapePoly

Examples

```
## Not run:
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@data$PB, emdi_model$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@data$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)


## End(Not run)
```

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

Description

Diagnostic plots of the underlying model in the EBP approach (see also ebp) are obtained. These include Q-Q plots and density plots of residuals and random effects from the nested error regression 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 return depends on the transformation such that a plot for the optimal parameter is only returned in case a Box-Cox transformation 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, ...)
```

Arguments

`x`	an object of type "emdi", "model", representing point and, if chosen, MSE estimates obtained by the EBP approach (see also ebp).
`label`	argument that enables to customize title and axis labels. There are four options to label the diagnostic plot: (i) original labels ("orig"), (ii) axis lables 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 box_cox 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.

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"),

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

`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. 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 optimal parameter +2/-1 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. This means for the default interval seq(-1, 2, by = 0.05).

`...` optional arguments passed to generic function.

Value

Two Q-Q plots in one grid, two density plots, a Cook' distance plot and a likelihood plot for the optimal parameter of the Box-Cox transformation obtained by ggplot.

See Also

emdiObject, ebp

Examples

```
## Not run:
# 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")))


## End(Not run)
```

`print.emdi` *Prints an emdiObject*

Description

Basic information of an emdi object is printed.

Usage

```
## S3 method for class 'emdi'
print(x, ...)
```

Arguments

`x`	an x of type "emdi", representing point and MSE estimates obtained by direct estimation (see also `direct`) or Empirical Best Prediction (see also `ebp`).
`...`	optional arguments passed to `print.default`.

See Also

emdiObject, ebp

`print.estimators.emdi` *Prints estimators.emdi objects*

Description

Prints estimators.emdi objects

Usage

```
## S3 method for class 'estimators.emdi'
print(x, ...)
```

Arguments

`x`	an object of type "estimators.emdi".
`...`	further arguments passed to or from other methods.

`print.summary.emdi` *Prints a summary.emdi object*

Description

The elements described in summary.emdi are printed.

Usage

```
## S3 method for class 'summary.emdi'
print(x, ...)
```

Arguments

`x`	an object of type "summary.emdi", generally resulting from applying summary to an object of type "emdi"
`...`	optional arguments passed to print.default; see the documentation on that method functions.

See Also

summary.emdi

```
subset.estimators.emdi
```

Subsets an estimators.emdi object

Description

Subsets an estimators.emdi object

Usage

```
## S3 method for class 'estimators.emdi'
subset(x, ...)
```

Arguments

`x`	an object of type "estimators.emdi".
`...`	further arguments passed to or from other methods.

Value

Selected subsets of the object of type "estimators.emdi".

See Also

`estimators.emdi`

Examples

```
## Not run:
# Loading data - population and sample data
data("eusilcA_pop")
data("eusilcA_smp")

# Generate emdi object with deleting missing values; 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",
na.rm = TRUE)

# Example: Choose last lines of the Gini coefficient, MSE and CV
subset(estimators(emdi_model, indicator = "Gini"),
       Domain %in% c("Wien", "Wien Umgebung"))

## End(Not run)
```

`summary.emdi` *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 returned object is suitable for printing with the `print.summary.emdi` method.

Usage

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

Arguments

`object`	an object of type "emdi", representing point and MSE estimates. Objects differ depending on the estimation method: direct vs. model-based.
`...`	additional arguments that are not used in this method.

Value

an object of type "summary.emdi" with following components:

`out_of_smp`	if model-based estimation, number of out-of-sample domains equivalent to `N_dom_unobs` (see `emdiObject`).
`in_smp`	number of in-sample domains equivalent to `N_dom_smp` (see `emdiObject`).
`size_smp`	number of units in sample equivalent to `N_smp` (see `emdiObject`).
`size_pop`	if model-based estimation, number of units in population equivalent to `N_pop` (see `emdiObject`).
`size_dom`	a data frame with rows Sample_domains and Population_domains (if model-based estimation) representing summary statisitics of the sample sizes across domains of sample and population data, respectively.
`transform`	if model-based estimation, a data frame with columns Transformation, Method, Optimal_lambda and Shift_parameter representing the chosen transformation type and estimation method for lambda as well as their results.
`normality`	if model-based estimation, a data frame with columns Skewness, Kurtosis, Shapiro_W and Shapiro_p where the latter two represent the results of a Shapiro-Wilks-Test for normality. Rows correspond to Pearson residuals and random effects of the nested error regression model. The functions `skewness` and `kurtosis` are from the package **moments**. Details for the Shapiro-Wilks-Test are provided by `shapiro.test`.
`icc`	if model-based estimation, the value of the intraclass coefficient.

coeff_determ — if model-based estimation, a data frame with colums Marginal_R2 and Conditional_R2 representing two R2 measures for linear mixed models from the **MuMIn** package obtained by function `r.squaredGLMM`.

call — a list containing an image of the function call that produced the object.

See Also

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

Examples

```
## Not run:
# 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, threshold){max(y)},
my_min = function(y, threshold){min(y)}), na.rm = TRUE, cpus = 1)

# Receive first overview
summary(emdi_model)

## End(Not run)
```

`tail.estimators.emdi` *Returns the last part of predicted indicators and, if chosen, of MSE and CV estimators.*

Description

Returns the last part of predicted indicators and, if chosen, of MSE and CV estimators.

Usage

```
## S3 method for class 'estimators.emdi'
tail(x, n = 6L, addrownums = NULL, ...)
```

Arguments

x — an object of type "estimators.emdi", representing point estimators and, if chosen, MSE and/or CV estimates for selected indicators.

n — a single integer. If positive, it determines the number of rows for the data frame. If negative, all but the n first rows of elements of the object.

`addrownums` if there are no row names, create them from the row numbers.

`...` arguments to be passed to or from other methods.

Value

Selected rows of the object of type "estimators.emdi".

See Also

estimators.emdi

Examples

```
## Not run:
# Loading data - population and sample data
data("eusilcA_pop")
data("eusilcA_smp")

# Generate emdi object with deleting missing values; 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",
na.rm = TRUE)

# Example: Choose last lines of the Gini coefficient, MSE and CV
tail(estimators(emdi_model, indicator = c("Gini", "Head_Count")))

## End(Not run)
```

---

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 `summary.emdi` 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 = "excel_output.xlsx", indicator = "all",
  MSE = FALSE, CV = FALSE, split = FALSE)

write.ods(object, file = "ods_output.ods", 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". Defaults to "all". 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.
- `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 diffrent files for point estimates and their precisions. Defaults to `FALSE`.

Details

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

Examples

```
## Not run:
# Loading data - population and sample data
data("eusilcA_pop")
data("eusilcA_smp")

# Generate emdi object 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, threshold){max(y)},
my_min = function(y, threshold){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, file = "excel_output_all.xlsx", indicator = "all",
MSE = TRUE, CV = TRUE)

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

# Example 3: Same as example 1 but for an ODS output
write.ods(emdi_model, file = "ods_output_all.ods", indicator = "all",
MSE = TRUE, CV = TRUE)

## End(Not run)
```

Index