Package 'SAEforest'

September 7, 2022

Title Mixed Effect Random Forests for Small Area Estimation

Version 1.0.0

Description Mixed Effects Random Forests (MERFs) are a data-driven, nonparametric alternative to current methods of Small Area Estimation (SAE). 'SAEforest' provides functions for the estimation of regionally disaggregated linear and nonlinear indicators using survey sample data. Included procedures facilitate the estimation of domain-level economic and inequality metrics and assess associated uncertainty. Emphasis lies on straightforward interpretation and visualization of results. From a methodological perspective, the package builds on approaches discussed in Krennmair and Schmid (2022) <arXiv:2201.10933v2> and Krennmair et al. (2022) <arXiv:2204.10736>.

License GPL (>= 2)

URL `https://github.com/krennpa/SAEforest`, `https://krennpa.github.io/SAEforest/`

Depends R (>= 4.1.0)

Imports caret, dplyr, ggplot2, haven, ineq, lme4, maptools, pbapply, pdp, ranger, reshape2, stats, vip

Suggests R.rsp, sp, rgeos, testthat (>= 3.0.0)

Encoding UTF-8

RoxygenNote 7.2.1

LazyData true

VignetteBuilder R.rsp

NeedsCompilation no

Config/testthat/edition 3

Author Patrick Krennmair [aut, cre]

Maintainer Patrick Krennmair <patrick.krennmair@fu-berlin.de>

Repository CRAN

Date/Publication 2022-09-07 17:50:06 UTC

R topics documented:

`eusilcA_pop` *Simulated EU-SILC data - population data*

Description

The data set includes synthetic EU-SILC data and is taken from the package **emdi**. Originally, the data builds on `eusilcP` from package **simFrame** and was reduced to 17 variables containing regional variables for 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_popAgg *Simulated EU-SILC data - aggregated population data*

Description

The data set includes synthetic EU-SILC data and is taken from the package **emdi**. Originally, the data builds on `eusilcP` from package **simFrame** and is reduced to 15 variables including district identifiers as well as aggregated household level covariates. Therefore, except for the variables `ratio_n` and `district`, the variables are 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.

district factor; Austrian districts (94 levels).

`eusilcA_smp` *Simulated EU-SILC data - survey sample data*

Description

The data set includes synthetic EU-SILC data and is taken from the package **emdi**. Originally, the data builds on `eusilcP` from package **simFrame** and was reduced to 18 variables containing regional variables for states and districts.

Usage

```
eusilcA_smp
```

Format

A data frame with 1945 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.

map_indicators *Visualizes disaggregated estimates on a map*

Description

Function map_indicators visualizes estimates from a SAEforestObject on a specified map. The function can be seen as a modified wrapper of map_plot from the package **emdi**.

Usage

```
map_indicators(
  object,
  indicator = "all",
  MSE = FALSE,
  CV = FALSE,
  map_obj = NULL,
  map_dom_id = NULL,
  map_tab = NULL,
  color = c("white", "darkgreen"),
  scale_points = NULL,
  guide = "colourbar",
  return_data = FALSE,
  return_plot = FALSE,
  gg_theme = theme_minimal()
)
```

Arguments

object	An object of class SAEforest, containing estimates to be visualized.
indicator	Optional character vector specifying indicators to be mapped: (i) all calculated indicators ("all"); (ii) default indicators name: "Mean", "Quant10", "Quant25", "Median", "Quant75", "Quant90", "Gini", "Hcr", "Pgap", "Qsr" or the function name/s of "custom_indicator/s"; (iii) a vector of names of indicators. If the object is estimated with option meanOnly = TRUE, indicator arguments are ignored and only "Mean" is visualized.
MSE	Logical. If TRUE, the MSE is also visualized. Defaults to FALSE.
CV	Logical. If TRUE, the CV is also visualized. Defaults to FALSE.
map_obj	An SpatialPolygonsDataFrame object as defined by the **sp** package on which the data should be visualized.
map_dom_id	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 matches the domain variable from the population data set (first column) with the domain variable in the map_obj (second column). This should only be used if domain-level identifiers are different in both objects.

color	A vector of length 2 defining the lowest and highest color in the map.
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.
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.
return_plot	If set to TRUE, a list of individual plots produced by **ggplot2** is returned for further individual customization and processing.
gg_theme	Specify a predefined theme from **ggplot2**. Defaults to theme_minimal.

Value

Creates required plots and if selected, a fortified data.frame and a list of plots.

See Also

SAEforest, readShapePoly, SpatialPolygonsDataFrame, ggplot.

Examples

```
# Loading data
data("eusilcA_pop")
data("eusilcA_smp")
data("shape_Aut")

income <- eusilcA_smp$eqIncome
X_covar <- eusilcA_smp[, -c(1, 16, 17, 18)]

# Example 1:
# Calculating point estimates and discussing basic generic functions

model1 <- SAEforest_model(Y = income, X = X_covar, dName = "district",
                          smp_data = eusilcA_smp, pop_data = eusilcA_pop,
                          num.trees = 50)

# Create map plot for mean indicator - point and MSE estimates but no CV

map_indicators(object = model1, MSE = FALSE, CV = FALSE, map_obj = shape_Aut,
               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_Aut@data$PB, model1$Indicators$district)
```

```
# Create the mapping table based on the order obtained above
map_tab <- data.frame(pop_data_id = model1$Indicators$district[dom_ord],
                      shape_id = shape_Aut@data$BKZ)

# Create map plot for mean indicator - using the numerical domain
# identifiers of the shape file. Additionally save the figure in as a list element.

map_obj <- map_indicators(object = model1, MSE = FALSE, CV = FALSE,
                          map_obj = shape_Aut, indicator = c("Mean"),
                          map_dom_id = "BKZ", map_tab = map_tab, return_plot = TRUE)
```

`MERFranger` *Main function for unit-level MERF*

Description

This function enables the use of Mixed Effects Random Forests (MERFs) by effectively combining a random forest from **ranger** with a model capturing random effects from **lme4**. The MERF algorithm is an algorithmic procedure reminiscent of an EM-algorithm (see Details). The function is the base-function for the wrapping function (`SAEforest_model` and should not be directly used by the ordinary user. Recommended exceptions are applications exceeding the scope of existing wrapper functions or further research. The function `MERFranger` allows to model complex patterns of structural relations (see Examples). The function returns an object of class `MERFranger`, which can be used to produce unit-level predictions. In contrast to the wrapping functions, this function does not directly provide SAE estimates on domain-specific indicators.

Usage

```
MERFranger(
  Y,
  X,
  random,
  data,
  importance = "none",
  initialRandomEffects = 0,
  ErrorTolerance = 1e-04,
  MaxIterations = 25,
  na.rm = TRUE,
  ...
)
```

Arguments

`Y`	Continuous input value of target variable.
`X`	Matrix of predictive covariates.

random
: Specification of random effects terms following the syntax of lmer. Random effect terms are specified by vertical bars (|) separating expressions for design matrices from grouping factors. For further details see lmer and the example below.

data
: data.frame of sample data including the specified elements of Y and X.

importance
: Variable importance mode processed by the random forest from the **ranger**. Must be 'none', 'impurity', 'impurity_corrected', 'permutation'. For further details see ranger.

initialRandomEffects
: Numeric value or vector of initial estimate of random effects. Defaults to 0.

ErrorTolerance
: Numeric value to monitor the MERF algorithm's convergence. Defaults to 1e-04.

MaxIterations
: Numeric value specifying the maximal amount of iterations for the MERF algorithm. Defaults to 25.

na.rm
: Logical. Whether missing values should be removed. Defaults to TRUE.

...
: Additional parameters are directly passed to the random forest ranger. Most important parameters are for instance mtry (number of variables to possibly split at in each node), or num.trees (number of trees). For further details on possible parameters see ranger and the example below.

Details

There exists a generic function for predict for objects obtained by MERFranger.

The MERF algorithm iteratively optimizes two separate steps: a) the random forest function, assuming the random effects term to be correct and b) estimates the random effects part, assuming the OOB-predictions from the forest to be correct. Overall convergence of the algorithm is monitored by the log-likelihood of a joint model of both components. For further details see Krennmair & Schmid (2022) or Hajjem et al. (2014).

Note that the MERFranger object is a composition of elements from a random forest of class ranger and a random effects model of class merMod. Thus, all generic functions are applicable to corresponding objects. For further details on generic functions see ranger and lmer as well as the examples below.

Value

An object of class MERFranger includes the following elements:

Forest
: A random forest of class ranger modelling fixed effects of the model.

EffectModel
: A model of random effects of class merMod capturing structural components of MERFs and modeling random components.

RandomEffects
: List element containing the values of random intercepts from EffectModel.

RanEffSD
: Numeric value of the standard deviation of random intercepts.

ErrorSD
: Numeric value of standard deviation of unit-level errors.

VarianceCovariance
: VarCorr matrix from EffectModel.

LogLik	Vector with numerical entries showing the loglikelihood of the MERF algorithm.
IterationsUsed	Numeric number of iterations used until convergence of the MERF algorithm.
OOBresiduals	Vector of OOB-residuals.
Random	Character specifying the random intercept in the random effects model.
ErrorTolerance	Numerical value to monitor the MERF algorithm's convergence.
initialRandomEffects	Numeric value or vector of initial specification of random effects.
MaxIterations	Numeric value specifying the maximal amount of iterations for the MERF algorithm.

References

Hajjem, A., Bellavance, F., & Larocque, D. (2014). Mixed-Effects Random Forest for Clustered Data. Journal of Statistical Computation and Simulation, 84 (6), 1313–1328.

Krennmair, P., & Schmid, T. (2022). Flexible Domain Prediction Using Mixed Effects Random Forests. Journal of Royal Statistical Society: Series C (Applied Statistics) (forthcoming).

See Also

SAEforest, ranger, lmer, SAEforest_model

Examples

```
# Load Data
data("eusilcA_pop")
data("eusilcA_smp")

income <- eusilcA_smp$eqIncome
X_covar <- eusilcA_smp[, -c(1, 16, 17, 18)]

# Example 1:
# Calculating general model used in wrapper functions

model1 <- MERFranger(Y = income, X = X_covar, random = "(1|district)",
                     data = eusilcA_smp, num.trees=50)

# get individual predictions:

ind_pred <- predict(model1, eusilcA_pop)
```

plot.SAEforest *Plot function for a 'SAEforest' object*

Description

Plots model-specific characteristics of the fixed effects random forest component of the MERF from a SAEforestObject. A variable importance plot is produced to visualize the importance of individual covariates for the predictive performance of the model. For the variable importance plot, arguments are passed internally to the function vip. If requested, the plot function additionally provides a partial dependence plot (pdp) to visualize the impact of a given number of influential covariates on the target variable. The pdp plot is produced using partial from the package **pdp**. The plot-engine for both plots is **ggplot2**.

Usage

```
## S3 method for class 'SAEforest'
plot(
  x,
  num_features = 6,
  col = "darkgreen",
  fill = "darkgreen",
  alpha = 0.8,
  include_type = TRUE,
  horizontal = TRUE,
  gg_theme = theme_minimal(),
  lsize = 1.5,
  lty = "solid",
  grid_row = 2,
  out_list = FALSE,
  pdp_plot = TRUE,
  ...
)
```

Arguments

x	An object of class SAEforest including a random forest model of class ranger.
num_features	Number of features for which a partial dependence plot is required.
col	Parameter specifying the color of selected plots. The argument must be specified such that it can be processed by aes. Defaults to a character name of the color "darkgreen".
fill	Parameter specifying the fill of selected plots. The argument must be specified such that it can be processed by aes. Defaults to a character name of the color "darkgreen".
alpha	Parameter specifying the transparency of fill for vip plots. The argument must be a number in [0,1].

include_type	Logical. If set to TRUE, the type of importance specified in the fitting process of the model is included in the vip plot. Defaults to TRUE.
horizontal	Logical. If set to TRUE, the importance scores appear on the x-axis. If parameter is set to FALSE, the importance scores are plot on the y-axis. Defaults to TRUE.
gg_theme	Specify a predefined theme from **ggplot2**. Defaults to theme_minimal.
lsize	Parameter specifying the line size of pdp plots. The argument must be specified such that it can be processed by aes. Defaults to 1.5.
lty	Parameter specifying the line size of pdp plots. The argument must be specified such that it can be processed by aes. Defaults to "solid".
grid_row	Parameter specifying the amount of rows for the joint pdp plot. Defaults to 2.
out_list	Logical. If set to TRUE, a list of individual plots produced by **ggplot2** is returned for further individual customization and processing. Defaults to FALSE.
pdp_plot	Logical. If set to TRUE, partial dependence plots produced by partial from the package **pdp** are included. Defaults to TRUE.
...	Optional additional inputs that are ignored for this method.

Details

For the production of importance plots, be sure to specify the parameter of importance != 'none' before producing estimates with function SAEforest_model.

For pdp plots, note that covariates of type factor or character cannot be used for partial dependence plots. Dummy-variables can be used, however, their pdp plots are always lines connecting two effect points for 0 and 1. Most informative pdp plots can be produced for continuous predictors.

Value

Plots of variable importance and/or partial dependence of covariates ranked by corresponding importance. Additionally, a list of individual plots can be returned facilitating individual customization and exporting. See the following examples for details.

See Also

SAEforestObject

Examples

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

income <- eusilcA_smp$eqIncome
X_covar <- eusilcA_smp[, -c(1, 16, 17, 18)]

# Example 1:
# Calculating point estimates and discussing basic generic functions

model1 <- SAEforest_model(Y = income, X = X_covar, dName = "district",
```

```
                          smp_data = eusilcA_smp, pop_data = eusilcA_pop,
                          num.trees = 50)
plot(model1)
```

popNsize *Demographic population-size data*

Description

This data contains simulated population data based on aggregates from eusilcA_pop, which is based on eusilcP from package **simFrame**.

Usage

```
popNsize
```

Format

A data frame with 94 Austrian districts and corresponding synthetic population numbers:

district character; districts (94 levels).

N_i numeric; simulated population of district.

print.SAEforest *Prints a 'SAEforest' object*

Description

Basic information of an SAEforestObject is printed.

Usage

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

Arguments

x	Object of class SAEforest, representing point and MSE estimates obtained by function SAEforest_model.
...	Optional additional inputs that are ignored for this method.

Value

Prints basic information on survey data characteristics.

See Also

`SAEforestObject`

`SAEforest` *'SAEforest' - Estimating disaggregated indicators using Mixed Effects Random Forests*

Description

The package **SAEforest** promotes the use of Mixed Effects Random Forests (MERFs) for applications of Small Area Estimation (SAE). The package effectively combines functions for the estimation of regionally disaggregated linear and nonlinear economic and inequality indicators using survey sample data. Estimated models increase the precision of direct estimates from survey data, combining unit-level and aggregated population level covariate information from census or register data. Apart from point estimates, MSE estimates for requested indicators can be easily obtained. The package provides procedures to facilitate the analysis of model performance of MERFs and visualizes predictive relations from covariates and variable importance. Additionally, users can summarize and map indicators and corresponding measures of uncertainty. Methodological details for the functions in this package are found in Krennmair & Schmid (2022), Krennmair et al. (2022a) and Krennmair et al. (2022b).

Details

This package includes a main function `MERFranger` that is wrapped in `SAEforest_model` for an improved SAE workflow. Each function produces an object inheriting requested results of regionally disaggregated point and uncertainty estimates. Additionally, statistical information on model fit and variable importance is accessible through generic functions such as a summary (`summary.SAEforest`) or a class-specific plot function (`plot.SAEforest`). For a full documentation of objects of class `SAEforest` see `SAEforestObject`. An overview of all currently provided functions within this package can be be seen with `help(package="SAEforest")`.

References

Krennmair, P., & Schmid, T. (2022). Flexible Domain Prediction Using Mixed Effects Random Forests. Journal of Royal Statistical Society: Series C (Applied Statistics) (forthcoming).

Krennmair, P., & Würz, N. & Schmid, T. (2022a). Analysing Opportunity Cost of Care Work using Mixed Effects Random Forests under Aggregated Census Data.

Krennmair, P., & Schmid, T & Tzavidis, Nikos. (2022b). The Estimation of Poverty Indicators Using Mixed Effects Random Forests. Working Paper.

SAEforestObject *Fitted 'SAEforest' object*

Description

An object of class SAEforest always includes point estimates of regionally disaggregated economic and inequality indicators and a MERFmodel element including information on the model fit for fixed effects as well as random effects. Optionally an SAEforestObject includes corresponding MSE estimates. In the case of mean estimates and aggregated covariate information, the SAEforestObject additionally includes an element, capturing the number of variables used in the weighting process from aggregated covariate information. For an object of class SAEforestObject, the following generic functions are applicable: print, plot, summary and summarize_indicators. Additionally selected generic functions of **lme4** (fixef, getData, ranef, residuals, sigma, VarCorr) are directly applicable to an object of class SAEforest.

Details

Note that the MERFmodel object is a composition of elements from a random forest of class ranger and a random effects model of class merMod. Thus, all generic functions are applicable to corresponding objects. For further details on generic functions see ranger and lmer as well as the examples below.

Value

Four components are always included in an SAEforest object. MSE_estimates and AdjustedSD are NULL except MSE results are requested. An element of NrCovar only exists for SAEforest objects produced by SAEforest_model with option aggData = TRUE.

MERFmodel The included MERFmodel object comprises information on the model fit, details on the performed MERF algorithm as well as details on variance components. See below for an exact description of components.

Indicators A data frame where the first column is the area-level identifier and additional columns are the indicators of interest. Note that objects from SAEforest_model only report the "Mean".

MSE_estimates Only if MSE results requested. A data frame where the first column is the area-level identifier and additional columns are the MSE estimates for indicators of interest. Note that objects from SAEforest_model only report MSE values for the "Mean".

NrCovar Only if means under aggregated covariate information are estimated, i.e. SAEforest_model with option aggData = TRUE. A list containing variable names of covariates used for the calculation of needed calibration weights for point estimates. See Krennmair et al. (2022a) for methodological details an explanations.

Details on object of MERFmodel:

Forest A random forest of class ranger modelling fixed effects of the model.

`EffectModel`	A model of random effects of class `merMod` capturing structural components of MERFs and modeling random components.
`RandomEffects`	List element containing the values of random intercepts from `EffectModel`.
`RanEffSD`	Numeric value of standard deviation of random intercepts.
`ErrorSD`	Numeric value of standard deviation of unit-level errors.
`VarianceCovariance`	VarCorr matrix from `EffectModel`.
`LogLik`	Vector with numerical entries showing the loglikelihood of the MERF algorithm.
`IterationsUsed`	Numeric number of iterations used until convergence of the MERF algorithm.
`OOBresiduals`	Vector of OOB-residuals.
`Random`	Character specifying the random intercept in the random effects model.
`ErrorTolerance`	Numerical value to monitor the MERF algorithm's convergence.
`initialRandomEffects`	Numeric value or vector of initial specification of random effects.
`MaxIterations`	Numeric value specifying the maximal amount of iterations for the MERF algorithm.
`call`	The summarized function call producing the object.
`data_specs`	Data characteristics such as domain-specific sample sizes or number of out-of-sample areas.
`data`	Processed survey sample data.

References

Krennmair, P., & Schmid, T. (2022). Flexible Domain Prediction Using Mixed Effects Random Forests. Journal of Royal Statistical Society: Series C (Applied Statistics) (forthcoming).

Krennmair, P., & Würz, N. & Schmid, T. (2022a). Analysing Opportunity Cost of Care Work using Mixed Effects Random Forests under Aggregated Census Data.

Krennmair, P., & Schmid, T & Tzavidis, Nikos. (2022b). The Estimation of Poverty Indicators Using Mixed Effects Random Forests. Working Paper.

See Also

`SAEforest_model`, `ranger`, `lmer`

Examples

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

income <- eusilcA_smp$eqIncome
X_covar <- eusilcA_smp[,-c(1,16,17,18)]

# Example 1:
```

```
# Calculating point estimates and discussing basic generic functions

model1 <- SAEforest_model(Y = income, X = X_covar, dName = "district",
                          smp_data = eusilcA_smp, pop_data = eusilcA_pop,
                          num.trees=50, mtry = 3)

#SAEforest generics:

summary(model1)
summarize_indicators(model1)
residuals(model1)
sigma(model1)
```

SAEforest_model *Main function for the estimation of domain-level (nonlinear) indicators with MERFs*

Description

This function enables the use of Mixed Effects Random Forests (MERFs) for applications of Small Area Estimation (SAE). Unit-level survey data on a target and auxiliary covariates is required to produce reliable estimates of various disaggregated economic and inequality indicators. Option `meanOnly` saves computational time for users that are only interested in the estimation of domain-specific means using unit-level and aggregated auxiliary data. Predefined indicators include the mean, median, quantiles (`10%`, `25%`, `75%` and `90%`), the head count ratio, the poverty gap, the Gini-coefficient and the quintile share ratio. The MERF algorithm is an algorithmic procedure reminiscent of an EM-algorithm (see Details). Overall, the function serves as a coherent framework for the estimation of point estimates and if requested uncertainty estimates for the indicators. Methodological details are found in Krennmair & Schmid (2022) and Krennmair et al. (2022b). The following examples showcase further potential applications.

Usage

```
SAEforest_model(
  Y,
  X,
  dName,
  smp_data,
  pop_data,
  MSE = "none",
  meanOnly = TRUE,
  aggData = FALSE,
  smearing = TRUE,
  popnsize = NULL,
  importance = "impurity",
  OOsample_obs = 25,
  ADDsamp_obs = 0,
```

```
    w_min = 3,
    B = 100,
    B_adj = 100,
    B_MC = 100,
    threshold = NULL,
    custom_indicator = NULL,
    initialRandomEffects = 0,
    ErrorTolerance = 1e-04,
    MaxIterations = 25,
    na.rm = TRUE,
    ...
)
```

Arguments

Y	Continuous input value of target variable.
X	Matrix or data.frame of predictive covariates.
dName	Character specifying the name of the domain identifier, for which random intercepts are modeled.
smp_data	data.frame of survey sample data including the specified elements of Y and X.
pop_data	data.frame of unit-level population covariate data X. Please note that the column names of predictive covariates must match column names of smp_data. This holds especially for the name of the domain identifier.
MSE	Character input specifying the type of uncertainty estimates. Available options are: (i) "none" if only point estimates are requested, (ii) "nonparametric" following the MSE bootstrap procedure proposed by Krennmair & Schmid (2022) or by Krennmair et al. (2022a) if aggData = TRUE. (iii) "wild" only for nonlinear indicators proposed by Krennmair et al. (2022b). Defaults to "none".
meanOnly	Logical. Calculating domain-level means only. Defaults to TRUE.
aggData	Logical input indicating whether aggregated covariate information or unit-level covariate information is used for domain-level means. Defaults to FALSE, assuming unit-level covariate data.
smearing	Logical input indicating whether a smearing based approach or a Monte Carlo (MC) version for point estimates should be obtained to estimate (nonlinear) indicators. MC should be used if computational constraints prohibit a smearing approach. For theoretical details see Krennmair et al (2022b). Defaults to TRUE.
popnsize	data.frame, comprising information of population size of domains. Only needed if aggData = TRUE and a MSE is requested. Please note that the name of the domain identifier must match the column name of smp_data.
importance	Variable importance mode processed by the random forest from **ranger**. Must be 'none', 'impurity', 'impurity_corrected' or 'permutation'. A concept of variable importance is needed for the production of generic plots plot. For the estimation of domain-level means under aggregated covariate data, variable importance is needed to rank information in the process of finding suitable calibration weights (Krennmair et al., 2022b). For further information regarding measures of importance see ranger.

OOsample_obs
: Number of out-of-sample observations taken from the closest area for potentially unsampled areas. Only needed if aggData = TRUE with defaults to 25.

ADDsamp_obs
: Number of out-of-sample observations taken from the closest area if first iteration for the calculation of calibration weights fails. Only needed if aggData = TRUE with defaults to 0.

w_min
: Minimal number of covariates from which informative weights are calculated. Only needed if aggData = TRUE. Defaults to 3.

B
: Number of bootstrap replications for MSE estimation procedures. Defaults to 100.

B_adj
: Number of bootstrap replications for the adjustment of residual variance proposed by Mendez and Lohr (2001). Defaults to 100.

B_MC
: Number of bootstrap populations in the MC version for point estimates of (non-linear) indicators. Defaults to 100.

threshold
: Set a custom threshold for indicators, such as the head count ratio. The threshold can be a known numeric value or function of Y. If the threshold is NULL, 60 % of the median of Y is taken as threshold. Defaults to NULL.

custom_indicator
: A list of additional functions containing the indicators to be calculated. These functions must only depend on the target variable Y and optionally the threshold. Defaults to NULL.

initialRandomEffects
: Numeric value or vector of initial estimates of random effects. Defaults to 0.

ErrorTolerance
: Numeric value to monitor the MERF algorithm's convergence. Defaults to 1e-04.

MaxIterations
: Numeric value specifying the maximal amount of iterations for the MERF algorithm. Defaults to 25.

na.rm
: Logical. Whether missing values should be removed. Defaults to TRUE.

...
: Additional parameters are directly passed to the random forest ranger. Most important parameters are for instance mtry (number of variables to possibly split at in each node), or num.trees (number of trees). For further details on possible parameters see ranger and the example below.

Details

MERFs combine advantages of regression forests (such as implicit model-selection and robustness properties) with the ability to model hierarchical dependencies.

The MERF algorithm iteratively optimizes two separate steps: a) the random forest function, assuming the random effects term to be correct and b) estimates the random effects part, assuming the OOB-predictions from the forest to be correct. Overall convergence of the algorithm is monitored by log-likelihood of a joint model of both components. For further details see Krennmair and Schmid (2022).

Users that are only interested in the estimation of domain-level means should set meanOnly = TRUE. The MERF requires covariate micro-data. This function, however also allows for the use of aggregated covariate information, by setting aggData = TRUE. Aggregated covariate information is adaptively incorporated through calibration-weights based on empirical likelihood for the estimation of area-level means. See methodological details in Krennmair et al. (2022a)

For the estimation of (nonlinear) poverty indicators and/or quantiles, we need information on the area-specific cumulative distribution function (CDF) of Y. Krennmair et al. (2022b) propose a smearing approach originated by Duan (1983). Alternatively, Monte-Carlo methods are used to simulate the domain-specific CDF of Y.

For the estimation of the MSE, the bootstrap population is built based on a bias-corrected residual variance as discussed in Krennmair and Schmid (2022). The bootstrap bias correction follows Mendez and Lohr (2011).

Note that the MERFmodel object is a composition of elements from a random forest of class ranger and a random effects model of class merMod. Thus, all generic functions are applicable to corresponding objects. For further details on generic functions see ranger and lmer as well as the examples below.

Value

An object of class SAEforest includes point estimates for disaggregated indicators as well as information on the MERF-model. Optionally corresponding MSE estimates are returned. Several generic functions have methods for the returned object of class SAEforest. For a full list and explanation of components and possibilities for objects of class SAEforest, see SAEforestObject.

References

Duan, N. (1983). Smearing Estimate: A Nonparametric Retransformation Method. Journal of the American Statistical Association, 78(383), 605–610.

Krennmair, P., & Schmid, T. (2022). Flexible Domain Prediction Using Mixed Effects Random Forests. Journal of Royal Statistical Society: Series C (Applied Statistics) (forthcoming).

Krennmair, P., & Würz, N. & Schmid, T. (2022a). Analysing Opportunity Cost of Care Work using Mixed Effects Random Forests under Aggregated Census Data.

Krennmair, P., & Schmid, T & Tzavidis, Nikos. (2022b). The Estimation of Poverty Indicators Using Mixed Effects Random Forests. Working Paper.

Mendez, G., & Lohr, S. (2011). Estimating residual variance in random forest regression. Computational Statistics & Data Analysis, 55 (11), 2937–2950.

See Also

SAEforestObject, ranger, lmer

Examples

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

income <- eusilcA_smp$eqIncome
X_covar <- eusilcA_smp[,-c(1, 16, 17, 18)]

# Example 1:
# Calculating point estimates and discussing basic generic functions
```

```
model1 <- SAEforest_model(Y = income, X = X_covar, dName = "district",
                          smp_data = eusilcA_smp, pop_data = eusilcA_pop)

# SAEforest generics:
summary(model1)

# Example 2:
# Calculating point + MSE estimates for aggregated covariate data and passing
# arguments to the random forest.
# Note that B is unrealistically low to improve example speed

# remove factor for gender
X_covar <- X_covar[,-1]
model2 <- SAEforest_model(Y = income, X = X_covar, dName = "district",
                          smp_data = eusilcA_smp, pop_data = eusilcA_popAgg,
                          MSE = "nonparametric", popnsize = popNsize,B = 5, mtry = 5,
                          num.trees = 100, aggData = TRUE)

# SAEforest generics:
summary(model2)
summarize_indicators(model2, MSE = TRUE, CV = TRUE)

# Example 3:
# Calculating point + MSE estimates and passing arguments to the forest.
# Two additional custom indicators and the threshold is defined as a custom function of Y.
# Note that B is unrealistically low to improve example speed.

model3 <- SAEforest_model(Y = income, X = X_covar, dName = "district", smp_data = eusilcA_smp,
                         pop_data = eusilcA_pop, meanOnly = FALSE, MSE = "nonparametric",
                          B = 5, mtry = 5, num.trees = 100, threshold = function(Y){0.5 *
                          median(Y)}, custom_indicator = list(my_max = function(Y,
                          threshold){max(Y)}, mean40 = function(Y, threshold){
                          mean(Y[Y<=quantile(Y,0.4)])}), smearing = FALSE)

# SAEforest generics:
summary(model3)
summarize_indicators(model3, MSE = FALSE, CV = TRUE, indicator = c("Gini", "my_max", "mean40"))
```

---

shape_Aut — *Data on shape for Austrian districts*

Description

The data contains the borders of 94 Austrian districts and simplifies the loading of the shape file for Austrian districts. It is originally used for examples in package **emdi**.

Usage

```
shape_Aut
```

Format

A shape file of class SpatialPolygonsDataFrame for 94 Austrian districts.

The main purpose of this function is the visualization of estimation results with the plotting function map_indicators. Further information on Copyrights is found in the attached copyright statement.

See Also

Information on the class of SpatialPolygonsDataFrame from the package **sp**.

---

summarize_indicators *Presents point, MSE and CV estimates*

Description

Function summarize_indicators reports point and mean squared error (MSE) estimates as well as calculated coefficients of variation (CV) from a fitted SAEforest object.

Usage

```
summarize_indicators(object, indicator = "all", MSE = FALSE, CV = FALSE)
```

Arguments

object	Object for which point and/or MSE estimates and/or calculated CV's are requested. The object must be of class SAEforest.
indicator	Optional character vector specifying indicators to be mapped: (i) all calculated indicators ("all"); (ii) each default indicators name: "Mean", "Quant10", "Quant25", "Median", "Quant75", "Quant90", "Gini", "Hcr", "Pgap", "Qsr" or the function name/s of "custom_indicator/s"; (iii) a vector of names of indicators. If the object is estimated by SAEforest_model indicator arguments are ignored and only the "Mean" is returned.
MSE	Logical. If TRUE, MSE estimates for selected indicators per domain are added to the data frame of point estimates. Defaults to FALSE.
CV	Logical. If TRUE, coefficients of variation for selected indicators per domain are added to the data frame of point estimates. Defaults to FALSE.

Details

Objects of class summarize_indicators.SAEforest 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 summarize_indicators is an object of class summarize_indicators.SAEforest including domain-specific point and/or MSE estimates and/or calculated CV's from a SAEforest object The returned object contains the data.frame ind and a character including the names of requested indicator(s).

See Also

SAEforestObject, SAEforest_model

Examples

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

income <- eusilcA_smp$eqIncome
X_covar <- eusilcA_smp[, -c(1, 16, 17, 18)]

# Calculating point + MSE estimates and passing arguments to the forest.
# Additionally, two additional indicators and functions as threshold are added.
# Note that B and num.trees are low to speed up estimation time and must be changed for
# practical applications.

model1 <- SAEforest_model(Y = income, X = X_covar, dName = "district",
                          smp_data = eusilcA_smp, pop_data = eusilcA_pop,
                          meanOnly = FALSE, MSE = "nonparametric", B = 5, mtry = 5,
                          num.trees = 50, smearing = FALSE)

# Extract indicator and show generics:
Gini1 <- summarize_indicators(model1, MSE = TRUE, CV = TRUE, indicator = "Gini")

head(Gini1)
tail(Gini1)
as.data.frame(Gini1)
as.matrix(Gini1)
subset(Gini1, district == "Wien")
```

---

summary.SAEforest *Summarizes an 'SAEforest' object*

Description

Shows additional information about the data, the SAE model and its components. Information is extracted from a SAEforest object. The returned object is suitable for printing with print.

Usage

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

Arguments

`object`	An object of class `SAEforest` representing point and MSE estimates. Objects differ depending on the estimation method.
`...`	Optional additional inputs that are ignored for this method.

Value

An object of class `summary.SAEforest` including information about the sample and population data, the model fit and random forest specific metrics.

See Also

`SAEforestObject`

Examples

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

income <- eusilcA_smp$eqIncome
X_covar <- eusilcA_smp[, -c(1, 16, 17, 18)]

# Example 1:
# Calculating point estimates and discussing basic generic functions

model1 <- SAEforest_model(Y = income, X = X_covar, dName = "district",
                          smp_data = eusilcA_smp, pop_data = eusilcA_pop,
                          num.trees=50, mtry=3)

# SAEforest generics:
summary(model1)
```

`tune_parameters` *Tuning and cross-validation of MERF parameters*

Description

Function `tune_parameters` allows to tune parameters for the implemented MERF method. Essentially, this function can be understood as a modified wrapper for train from the package **caret**, treating MERFs as a custom method.

Usage

```
tune_parameters(
  Y,
  X,
  data,
  dName,
  trControl,
  tuneGrid,
  seed = 11235,
  gg_theme = theme_minimal(),
  plot_res = TRUE,
  return_plot = FALSE,
  na.rm = TRUE,
  ...
)
```

Arguments

`Y`	Continuous input value of target variable.
`X`	Matrix or data.frame of predictive covariates.
`data`	data.frame of survey sample data including the specified elements of `Y` and `X`.
`dName`	Character specifying the name of domain identifier, for which random intercepts are modeled.
`trControl`	Control parameters passed to train. Most important parameters are `method` ("repeatedcv" for x-fold cross-validation), `number` (the number of folds) and `repeats` (the number of repetitions). For further details see trainControl and the example below.
`tuneGrid`	A data.frame with possible tuning values. The columns must have the same names as the tuning parameters. For this tuning function the grid must comprise entries for the following parameters: `num.trees`, `mtry`, `min.node.size`, `splitrule`.
`seed`	Enabling reproducibility of for cross-validation and tuning. Defaults to `11235`.
`gg_theme`	Specify a predefined theme from **ggplot2**. Defaults to `theme_minimal`.
`plot_res`	Optional logical. If `TRUE`, the plot with results of cross-validation and tuning is shown. Defaults to `TRUE`.
`return_plot`	If set to `TRUE`, a list of the comparative plot produced by **ggplot2** is returned for further individual customization and processing.
`na.rm`	Logical. Whether missing values should be removed. Defaults to `TRUE`.
`...`	Additional parameters are directly passed to the random forest ranger and/or the training function train. For further details on possible parameters and examples see ranger or train.

Details

Tuning can be performed on the following four parameters: `num.trees` (the number of trees for a forest), `mtry` (number of variables as split candidates at in each node), `min.node.size` (minimal individual node size) and `splitrule` (general splitting rule). For details see ranger.

Value

Prints requested optimal tuning parameters and (if requested) an additional comparative plot produced by **ggplot2**.

See Also

SAEforest, MERFranger, train, ggplot

Examples

```
# Loading data
data("eusilcA_pop")
data("eusilcA_smp")
library(caret)

income <- eusilcA_smp$eqIncome
X_covar <- eusilcA_smp[, -c(1, 16, 17, 18)]

# Specific characteristics of Cross-validation
fitControl <- trainControl(method = "repeatedcv", number = 5,
                           repeats = 1)

# Define a tuning-grid
merfGrid <- expand.grid(num.trees = 50, mtry = c(3, 7, 9),
                        min.node.size = 10, splitrule = "variance")

tune_parameters(Y = income, X = X_covar, data = eusilcA_smp,
                dName = "district", trControl = fitControl,
                tuneGrid = merfGrid)
```

Index