Package ‘sae.projection’

July 23, 2025

Type Package

Title Small Area Estimation Using Model-Assisted Projection Method

Version 0.1.4

Description Combines information from two independent surveys using a model-assisted projection method. Designed for survey sampling scenarios where a large sample collects only auxiliary information (Survey 1) and a smaller sample provides data on both variables of interest and auxiliary variables (Survey 2). Implements a working model to generate synthetic values of the variable of interest by fitting the model to Survey 2 data and predicting values for Survey 1 based on its auxiliary variables (Kim & Rao, 2012) <doi:10.1093/biomet/asr063>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

URL https://github.com/Alfrzlp/sae.projection

BugReports https://github.com/Alfrzlp/sae.projection/issues

Imports FSelector, glmnet, xgboost, survey, cli, doParallel, dplyr, methods, parsnip, recipes, rlang, rsample, stats, tune, workflows, yardstick, bonsai, ranger, randomForest, themis, lightgbm, caret

RoxygenNote 7.3.2

Depends R (>= 4.3.0), tidymodels

NeedsCompilation no

Repository CRAN

Date/Publication 2025-07-06 14:40:02 UTC

VignetteBuilder knitr

Suggests knitr, rmarkdown, quarto, testthat (>= 3.0.0)

Config/testthat/edition 3

Author Ridson Al Farizal P [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-0617-0214>),
Azka Ubaidillah [aut] (ORCID: <https://orcid.org/0000-0002-3597-0459>),
Silvi Ajeng Larasati [aut],
Amelia Rahayu [aut]

Maintainer Ridson Al Farizal P <ridsonalfarizal15@gmail.com>

Contents

`df_svy22` *df_svy22*

Description

A dataset from a survey conducted at the province level in Indonesia in 2022.

Usage

```
df_svy22
```

Format

A data frame with 74.070 rows and 11 variables.

PSU Primary Sampling Unit

WEIGHT Weight from survey

PROV province code

REGENCY regency/municipality code

STRATA Strata

income Income

neet Not in education employment or training status

sex sex (1: male, 2: female)

age age

disability disability status (0: False, 1: True)

edu last completed education

`df_svy23` *df_svy23*

Description

A dataset from a survey conducted at the province level in Indonesia in 2023.

Usage

```
df_svy23
```

Format

A data frame with 66.245 rows and 11 variables.

PSU Primary Sampling Unit

WEIGHT Weight from survey

PROV province code

REGENCY regency/municipality code

STRATA Strata

income Income

neet Not in education employment or training status

sex sex (1: male, 2: female)

age age

disability disability status (0: False, 1: True)

edu last completed education

`df_svy_A` *df_svy_A*

Description

A simulation dataset from a small sample survey, presented only at provincial level (Domain 1).

Usage

```
df_svy_A
```

Format

A data frame with 2000 rows and 20 variables with 40 domains.

province Province code

id_ind Unique identifier for each respondent

num Sample number

weight Weight from survey

x1 Predictor variables X1

x2 Predictor variables X2

x3 Predictor variables X3

x4 Predictor variables X4

x5 Predictor variables X5

x6 Predictor variables X6

x7 Predictor variables X7

x8 Predictor variables X8

x9 Predictor variables X9

x10 Predictor variables X10

x11 Predictor variables X11

x12 Predictor variables X12

x13 Predictor variables X13

x14 Predictor variables X14

x15 Predictor variables X15

Y Target variable (1: Yes, 0: No)

`df_svy_B` *df_svy_B*

Description

A simulation dataset from a large sample survey, presented at the regency level (Domain 2).

Usage

```
df_svy_B
```

Format

A data frame with 8000 rows and 20 variables with 40 domains.

province Province code

regency Regency code

id_ind Unique identifier for each respondent

num Sample number

weight Weight from survey

x1 Predictor variables X1

x2 Predictor variables X2

x3 Predictor variables X3

x4 Predictor variables X4

x5 Predictor variables X5

x6 Predictor variables X6

x7 Predictor variables X7

x8 Predictor variables X8

x9 Predictor variables X9

x10 Predictor variables X10

x11 Predictor variables X11

x12 Predictor variables X12

x13 Predictor variables X13

x14 Predictor variables X14

x15 Predictor variables X15

`ma_projection` *Model-Assisted Projection Estimator*

Description

The function addresses the problem of combining information from two or more independent surveys, a common challenge in survey sampling. It focuses on cases where:

- **Survey 1:** A large sample collects only auxiliary information.
- **Survey 2:** A much smaller sample collects both the variables of interest and the auxiliary variables.

The function implements a model-assisted projection estimation method based on a working model. The working models that can be used include several machine learning models that can be seen in the details section

Usage

```
ma_projection(
  formula,
  cluster_ids,
  weight,
  strata = NULL,
  domain,
  summary_function = "mean",
  working_model,
  data_model,
  data_proj,
  model_metric,
  cv_folds = 3,
  tuning_grid = 10,
  parallel_over = "resamples",
  seed = 1,
  return_yhat = FALSE,
  ...
)
```

Arguments

`formula` A model formula. All variables used must exist in both `data_model` and `data_proj`.

`cluster_ids` Column name (character) or formula specifying cluster identifiers from highest to lowest level. Use `~0` or `~1` if there are no clusters.

`weight` Column name in `data_proj` representing the survey weights.

`strata` Column name for stratification; use `NULL` if no strata are used.

`domain` Character vector specifying domain variable names in both datasets.

`summary_function` A function to compute domain-level estimates (default: `"mean"`, `"total"`, `"variance"`).

`working_model` A parsnip model object specifying the working model (see `@details`).

`data_model` Data frame (small sample) containing both target and auxiliary variables.

`data_proj` Data frame (large sample) containing only auxiliary variables.

`model_metric` A `yardstick::metric_set()` function, or `NULL` to use default metrics.

`cv_folds` Number of folds for k-fold cross-validation.

`tuning_grid` Either a data frame with tuning parameters or a positive integer specifying the number of grid search candidates.

`parallel_over` Specifies parallelization mode: `"resamples"`, `"everything"`, or `NULL`. If "resamples", then tuning will be performed in parallel over resamples alone. Within each resample, the preprocessor (i.e. recipe or formula) is processed once, and is then reused across all models that need to be fit. If "everything", then tuning will be performed in parallel at two levels. An outer parallel loop will iterate over resamples. Additionally, an inner parallel loop will iterate over all unique combinations of preprocessor and model tuning parameters for that specific resample. This will result in the preprocessor being re-processed multiple times, but can be faster if that processing is extremely fast.

seed	Integer seed for reproducibility.
return_yhat	Logical; if TRUE, returns predicted y values for data_model.
...	Additional arguments passed to svydesign.

Details

The following working models are supported via the **parsnip** interface:

- linear_reg() – Linear regression
- logistic_reg() – Logistic regression
- linear_reg(engine = "stan") – Bayesian linear regression
- logistic_reg(engine = "stan") – Bayesian logistic regression
- poisson_reg() – Poisson regression
- decision_tree() – Decision tree
- nearest_neighbor() – k-Nearest Neighbors (k-NN)
- naive_bayes() – Naive Bayes classifier
- mlp() – Multi-layer perceptron (neural network)
- svm_linear() – Support vector machine with linear kernel
- svm_poly() – Support vector machine with polynomial kernel
- svm_rbf() – Support vector machine with radial basis function (RBF) kernel
- bag_tree() – Bagged decision tree
- bart() – Bayesian Additive Regression Trees (BART)
- rand_forest(engine = "ranger") – Random forest (via ranger)
- rand_forest(engine = "aorsf") – Accelerated oblique random forest (AORF; Jaeger et al. 2022, 2024)
- boost_tree(engine = "lightgbm") – Gradient boosting (LightGBM)
- boost_tree(engine = "xgboost") – Gradient boosting (XGBoost)

For a complete list of supported models and engines, see Tidy Modeling With R.

Value

A list containing:

- model – The fitted working model object.
- prediction – A vector of predictions from the working model.
- df_result – A data frame with:
  - domain – Domain identifier.
  - ypr – Projection estimator results for each domain.
  - var_ypr – Estimated variance of the projection estimator.
  - rse_ypr – Relative standard error (in \

References

1. Kim, J. K., & Rao, J. N. (2012). Combining data from two independent surveys: a model-assisted approach. Biometrika, 99(1), 85-100.

Examples

```
## Not run:
library(sae.projection)
library(dplyr)
library(bonsai)

df_svy22_income <- df_svy22 %>% filter(!is.na(income))
df_svy23_income <- df_svy23 %>% filter(!is.na(income))

# Linear regression
lm_proj <- ma_projection(
  income ~ age + sex + edu + disability,
  cluster_ids = "PSU", weight = "WEIGHT", strata = "STRATA",
  domain = c("PROV", "REGENCY"),
  working_model = linear_reg(),
  data_model = df_svy22_income,
  data_proj = df_svy23_income,
  nest = TRUE
)


df_svy22_neet <- df_svy22 %>% filter(between(age, 15, 24))
df_svy23_neet <- df_svy23 %>% filter(between(age, 15, 24))


# LightGBM regression with hyperparameter tunning
show_engines("boost_tree")
lgbm_model <- boost_tree(
  mtry = tune(), trees = tune(), min_n = tune(),
  tree_depth = tune(), learn_rate = tune(),
  engine = "lightgbm"
)

lgbm_proj <- ma_projection(
  formula = neet ~ sex + edu + disability,
  cluster_ids = "PSU",
  weight = "WEIGHT",
  strata = "STRATA",
  domain = c("PROV", "REGENCY"),
  working_model = lgbm_model,
  data_model = df_svy22_neet,
  data_proj = df_svy23_neet,
  cv_folds = 3,
  tuning_grid = 3,
  nest = TRUE
)
```

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

---

projection_randomforest *Projection Estimator with Random Forest Algorithm*

Description

Kim and Rao (2012), the synthetic data obtained through the model-assisted projection method can provide a useful tool for efficient domain estimation when the size of the sample in survey B is much larger than the size of sample in survey A.

The function projects estimated values from a small survey (survey A) onto an independent large survey (survey B) using the random forest classification algorithm. The two surveys are statistically independent, but the projection relies on shared auxiliary variables. The process includes data preprocessing, feature selection, model training, and domain-specific estimation based on survey design principles "two stages one phase". The function automatically selects standard estimation or bias-corrected estimation based on the parameter `bias_correction`.

`bias_correction` = `TRUE` can only be used if there is `psu, ssu, strata` on the `data_model`. If it doesn't, then it will automatically be `bias_correction` = `FALSE`

Usage

```
projection_randomforest(
  data_model,
  target_column,
  predictor_cols,
  data_proj,
  domain1,
  domain2,
  psu,
  ssu = NULL,
  strata = NULL,
  weights,
  split_ratio = 0.8,
  feature_selection = TRUE,
  bias_correction = FALSE
)
```

Arguments

`data_model`	The training dataset, consisting of auxiliary variables and the target variable.
`target_column`	The name of the target column in the `data_model`.
`predictor_cols`	A vector of predictor column names.
`data_proj`	The data for projection (prediction), which needs to be projected using the trained model. It must contain the same auxiliary variables as the `data_model`

`domain1`	Domain variables for survey estimation (e.g., "province")
`domain2`	Domain variables for survey estimation (e.g., "regency")
`psu`	Primary sampling units, representing the structure of the sampling frame.
`ssu`	Secondary sampling units, representing the structure of the sampling frame (default is NULL).
`strata`	Stratification variable, ensuring that specific subgroups are represented (default is NULL).
`weights`	Weights used for the direct estimation from `data_model` and indirect estimation from `data_proj`.
`split_ratio`	Proportion of data used for training (default is 0.8, meaning 80 percent for training and 20 percent for validation).
`feature_selection`	Selection of predictor variables (default is `TRUE`)
`bias_correction`	Logical; if `TRUE`, then bias correction is applied, if `FALSE`, then bias correction is not applied. Default is `FALSE`.

Value

A list containing the following elements:

- `model` The trained Random Forest model.
- `importance` Feature importance showing which features contributed most to the model's predictions.
- `train_accuracy` Accuracy of the model on the training set.
- `validation_accuracy` Accuracy of the model on the validation set.
- `validation_performance` Confusion matrix for the validation set, showing performance metrics like accuracy, precision, recall, etc.
- `data_proj` The projection data with predicted values.

if `bias_correction = FALSE`:

- `Domain1` Estimations for Domain 1, including estimated values, variance, and relative standard error (RSE).
- `Domain2` Estimations for Domain 2, including estimated values, variance, and relative standard error (RSE).

if `bias_correction = TRUE`:

- `Direct` Direct estimations for Domain 1, including estimated values, variance, and relative standard error (RSE).
- `Domain1_corrected_bias` Bias-corrected estimations for Domain 1, including estimated values, variance, and relative standard error (RSE).
- `Domain2_corrected_bias` Bias-corrected estimations for Domain 2, including estimated values, variance, and relative standard error (RSE).

References

1. Kim, J. K., & Rao, J. N. (2012). Combining data from two independent surveys: a model-assisted approach. Biometrika, 99(1), 85-100.

Examples

```
library(survey)
library(caret)
library(dplyr)

data_A <- df_svy_A
data_B <- df_svy_B

# Get predictor variables from data_model
x_predictors <- data_A %>% select(5:19) %>% names()

# Run projection_randomforest with bias correction
rf_proj_corrected <- projection_randomforest(
                data_model = data_A,
                target_column = "Y",
                predictor_cols = x_predictors,
                data_proj = data_B,
                domain1 = "province",
                domain2 = "regency",
                psu = "num",
                ssu = NULL,
                strata = NULL,
                weights = "weight",
                feature_selection = TRUE,
                bias_correction = TRUE)

rf_proj_corrected$Direct
rf_proj_corrected$Domain1_corrected_bias
rf_proj_corrected$Domain2_corrected_bias
```

`projection_xgboost` *Projection Estimator with XGBoost Algorithm*

Description

Kim and Rao (2012), proposed a model-assisted projection estimation method for two independent surveys, where the first survey (**A1**) has a large sample that only collects auxiliary variables, while the second survey (**A1**) has a smaller sample but contains information on both the focal variable and auxiliary variables. This method uses a **Working Model (WM)** to relate the focal variable to the auxiliary variable based on data from **A2**, and then predicts the value of the focal variable for **A1**. A projection estimator is then obtained from the (**A2**) sample using the resulting synthetic values. This approach produces estimators that are asymptotically unbiased and can improve the efficiency of domain estimation, especially when the sample size in survey 1 is much larger compared to survey 2.

This function applies the XGBoost algorithm to project estimated values from a small survey onto an independent larger survey. While the two surveys are statistically independent, the projection is based on common auxiliary variables. The process in this function involves data preprocessing, feature selection, getting the best model with hyperparameter tuning, and performing domain-specific estimation following survey design principles.

Usage

```
projection_xgboost(
  target_col,
  data_model,
  data_proj,
  id,
  STRATA = NULL,
  domain1,
  domain2,
  weight,
  task_type,
  test_size = 0.2,
  nfold = 5,
  corrected_bias = FALSE,
  feature_selection = TRUE
)
```

Arguments

`target_col`	The name of the column that contains the target variable in the `data_model`.
`data_model`	A data frame or a data frame extension (e.g., a tibble) representing the training dataset, which consists of auxiliary variables and the target variable. This dataset is characterized by a smaller sample size and provides information on both the variable of interest and the auxiliary variables.
`data_proj`	A data frame or a data frame extension (e.g., a tibble) representing the projection dataset, which is characterized by a larger sample size that collects only auxiliary information or general-purpose variables. This dataset must contain the same auxiliary variables as the `data_model` and is used for making predictions based on the trained model.
`id`	Column name specifying cluster ids from the largest level to the smallest level, where ~0 or ~1 represents a formula indicating the absence of clusters.
`STRATA`	The name of the column that specifies the strata; set to NULL if no stratification is required.#' @param test_size Proportion of data used for training (default is 0.8, meaning 80% for training and 20% for validation).
`domain1`	Domain variables for higher-level survey estimation. (e.g., "province")
`domain2`	Domain variables for more granular survey estimation at a lower administrative level. (e.g., "regency")

`weight` The name of the column in `data_proj` that represents the survey weight, usually used for the purpose of indirect estimation .

`task_type` A string that specifies the modeling objective, indicating whether the task is for classification or regression. Use "classification" for tasks where the goal is to categorize data into discrete classes, such as predicting whether an email is spam or not. Use "regression" for tasks where the goal is to predict a continuous outcome, such as forecasting sales revenue or predicting house prices.

`test_size` The proportion of data used for testing, with the remaining data used for training.

`nfold` The number of data partitions used for cross-validation (n-fold validation).

`corrected_bias` A logical value indicating whether to apply bias correction to the estimation results from the modeling process. When set to TRUE, this parameter ensures that the estimates are adjusted to account for any systematic biases, leading to more accurate and reliable predictions.

`feature_selection`
Selection of predictor variables (default is `TRUE`)

Value

A list containing the following components:

`metadata` A list of metadata about the modeling process, including:

- `method`: Description of the method used (e.g., "Projection Estimator With XGBoost Algorithm"),
- `model_type`: The type of model, either "classification" or "regression",
- `feature_selection_used`: Logical, whether feature selection was used,
- `corrected_bias_applied`: Logical, whether bias correction was applied,
- `n_features_used`: Number of predictor variables used,
- `model_params`: The hyperparameters and settings of the final XGBoost model,
- `features_selected` (optional): Names of features selected, if feature selection was applied.

`estimation` A list of projection estimation results, including:

- `projected_data`: The dataset used for projection (e.g., kabupaten/kota) with predicted values,
- `domain1_estimation`: Estimated values for domain 1 (e.g., province level), including:
  - `Estimation`, `RSE`, `var`

  ,
- `domain2_estimation`: Estimated values for domain 2 (e.g., regency level), including:
  - `Estimation`, `RSE`, `var`

`performance` (Only if applicable) A list of model performance metrics:

- `mean_train_accuracy`, `final_accuracy`, `confusion_matrix` (for classification),
- `mean_train_rmse`, `final_rmse` (for regression).

`bias_correction` (Optional) A list of bias correction results, returned only if `corrected_bias = TRUE`, including:

- `direct_estimation`: Direct estimation before correction,
- `corrected_domain1`: Bias-corrected estimates for domain 1,
- `corrected_domain2`: Bias-corrected estimates for domain 2.

References

1. Kim, J. K., & Rao, J. N. (2012). Combining data from two independent surveys: a model-assisted approach. Biometrika, 99(1), 85-100.
2. Kim and Rao (2012), the synthetic data obtained through the model-assisted projection method can provide a useful tool for efficient domain estimation when the size of the sample in survey 1 is much larger than the size of sample in survey 2.

Examples

```
library(xgboost)
library(caret)
library(FSelector)
library(glmnet)
library(recipes)

Data_A <- df_svy_A
Data_B <- df_svy_B

hasil <- projection_xgboost(
                            target_col = "Y",
                            data_model = Data_A,
                            data_proj = Data_B,
                            id = "num",
                            STRATA = NULL,
                            domain1 = "province",
                            domain2 = "regency",
                            weight = "weight",
                            nfold = 3,
                            test_size = 0.2 ,
                            task_type = "classification",
                            corrected_bias = TRUE,
                            feature_selection = TRUE)
```

Index