Package 'svyVarSel'

July 23, 2025

Title Variable Selection for Complex Survey Data

Version 1.0.1

Maintainer Amaia Iparragirre <amaia.iparragirre@ehu.eus>

Description Fit design-based linear and logistic elastic nets with complex survey data considering the sampling design when defining training and test sets using replicate weights. Methods implemented in this package are described in: A. Iparragirre, T. Lumley, I. Barrio, I. Arostegui (2024) <doi:10.1002/sta4.578>.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

Imports survey, glmnet

Depends R (>= 2.10)

LazyData true

NeedsCompilation no

Author Amaia Iparragirre [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0002-0660-6535>),
Thomas Lumley [aut],
Irantzu Barrio [aut],
Inmaculada Arostegui [aut]

Repository CRAN

Date/Publication 2024-10-15 15:10:02 UTC

Contents

`replicate.weights` *Replicate weights*

Description

This function allows calculating replicate weights.

Usage

```
replicate.weights(
  data,
 method = c("JKn", "dCV", "bootstrap", "subbootstrap", "BRR", "split", "extrapolation"),
  cluster = NULL,
  strata = NULL,
  weights = NULL,
  design = NULL,
  k = 10,
  R = 1,
  B = 200,
  train.prob = 0.7,
  method.split = c("dCV", "bootstrap", "subbootstrap"),
  rw.test = FALSE,
  dCV.sw.test = FALSE
)
```

Arguments

`data`	A data frame with information on (at least) cluster and strata indicators, and sampling weights. It could be `NULL` if the sampling design is indicated in the `design` argument (see `design`).
`method`	A character string indicating the method to be applied to define replicate weights. Choose between one of these: `JKn`, `dCV`, `bootstrap`, `subbootstrap`, `BRR`, `split`, `extrapolation`.
`cluster`	A character string indicating the name of the column with cluster identifiers in the data frame indicated in `data`. It could be `NULL` if the sampling design is indicated in the `design` argument (see `design`).
`strata`	A character string indicating the name of the column with strata identifiers in the data frame indicated in `data`. It could be `NULL` if the sampling design is indicated in the `design` argument (see `design`).
`weights`	A character string indicating the name of the column with sampling weights in the data frame indicated in `data`. It could be `NULL` if the sampling design is indicated in the `design` argument (see `design`).
`design`	An object of class `survey.design` generated by `survey::svydesign()`. It could be `NULL` if information about `cluster`, `strata`, `weights` and `data` are given.

k	A numeric value indicating the number of folds to be defined. Default is k=10. Only applies for the dCV method.
R	A numeric value indicating the number of times the sample is partitioned. Default is R=1. Only applies for dCV, split or extrapolation methods.
B	A numeric value indicating the number of bootstrap resamples. Default is B=200. Only applies for bootstrap and subbootstrap methods.
train.prob	A numeric value between 0 and 1, indicating the proportion of clusters (for the method split) or strata (for the method extrapolation) to be set in the training sets. Default is train.prob=0.7. Only applies for split and extrapolation methods.
method.split	A character string indicating the way in which replicate weights should be defined in the split method. Choose one of the following: dCV, bootstrap or subbootstrap. Only applies for split method.
rw.test	A logical value. If TRUE, the function returns in the output object the replicate weights to the corresponding test sets. If FALSE, only the replicate weights of the training sets are returned. Default is rw.test = FALSE.
dCV.sw.test	A logical value. If TRUE original sampling weights for the units in the test sets are returned instead of the replicate weights. Default is dCV.sw.test = FALSE. Only applies for dCV method.

Details

Some of these methods (specifically JKn, bootstrap, subbootstrap and BRR), were previously implemented in the survey R-package, to which we can access by means of the function as.svrepdesign() (the names of the methods are kept as in as.svrepdesign()). Thus, the function replicate.weights() depends on this function to define replicate weights based on these options. In contrast, dCV, split and extrapolation have been expressly defined to be incorporated into this function.

Selecting any of the above-mentioned methods, the object returned by this function is a new data frame, which includes new columns into the original data set, each of them indicating replicate weights for different training (always) and test (optionally, controlled by the argument rw.test) subsets. The number of new columns and the way in which they are denoted depend on the values set for the arguments, in general, and on the replicate weights method selected, in particular. The new columns indicating training and test sets follow a similar structure for any of the selected methods. Specifically, the structure of the names of the training sets is the following: rw_r_x_train_t where x=1,...,R indicates the x-th partition of the sample and t=1,...,T the t-th training set. Similarly, the structure of the new columns indicating the test sets is the following: rw_r_x_test_t or sw_r_x_test_t, where x indicates the partition and t the number of the test set. In addition, for some of the methods we also indicate the fold or set to which each unit in the data set has been included in each partition. This information is included as fold_t or set_t, depending on the method. See more detailed information below.

Value

This function returns a new data frame with new columns, each of them indicating replicate weights for different subsets.

Examples

```
data(simdata_lasso_binomial)

# JKn ---------------------------------------------------------------------
newdata <- replicate.weights(data = simdata_lasso_binomial,
                             method = "JKn",
                             cluster = "cluster",
                             strata = "strata",
                             weights = "weights",
                             rw.test = TRUE)

# dCV ---------------------------------------------------------------------
newdata <- replicate.weights(data = simdata_lasso_binomial,
                             method = "dCV",
                             cluster = "cluster",
                             strata = "strata",
                             weights = "weights",
                             k = 10, R = 20,
                             rw.test = TRUE)

# subbootstrap ------------------------------------------------------------
newdata <- replicate.weights(data = simdata_lasso_binomial,
                             method = "subbootstrap",
                             cluster = "cluster",
                             strata = "strata",
                             weights = "weights",
                             B = 100)

# BRR ---------------------------------------------------------------------
newdata <- replicate.weights(data = simdata_lasso_binomial,
                             method = "BRR",
                             cluster = "cluster",
                             strata = "strata",
                             weights = "weights",
                             rw.test = TRUE)

# split -------------------------------------------------------------------
newdata <- replicate.weights(data = simdata_lasso_binomial,
                             method = "split",
                             cluster = "cluster",
                             strata = "strata",
                             weights = "weights",
                             R=20,
                             train.prob = 0.5,
                             method.split = "subbootstrap",
                             rw.test = TRUE)

# extrapolation -----------------------------------------------------------
newdata <- replicate.weights(data = simdata_lasso_binomial,
                            method = "extrapolation",
                            cluster = "cluster",
                            strata = "strata",
```

```
                                weights = "weights",
                                R=20,
                                train.prob = 0.5,
                                rw.test = TRUE)
```

`simdata_lasso_binomial` *Simulated complex survey data*

Description

This data has been simulated in order to provide the users with an example data set.

Usage

```
simdata_lasso_binomial

simdata_lasso_binomial
```

Format

`simdata_lasso_binomial`**:**

A data frame with 1,720 rows and 54 columns:

y dichotomous response variable

x.1,...,x.50 covariates

strata strata identifiers

cluster cluster identifiers

weights sampling weights

`simdata_lasso_binomial`**:**

A data frame with 1720 rows and 54 columns:

x.1,...,x.50 Covariates

y Dichotomous response variable

strata Column indicating the strata

cluster Column indicating the cluster

weights Sampling weights ...

welnet *Weighted (linear or logistic) elastic-nets for complex survey data*

Description

This function allows as to fit elastic nets (linear or logistic) to complex survey data, considering sampling weights in the estimation process and selects the lambda that minimizes the error based on different replicate weights methods.

Usage

```
welnet(
  data = NULL,
  col.y = NULL,
  col.x = NULL,
  cluster = NULL,
  strata = NULL,
  weights = NULL,
  design = NULL,
  family = c("gaussian", "binomial"),
  alpha = 1,
  lambda.grid = NULL,
 method = c("dCV", "JKn", "bootstrap", "subbootstrap", "BRR", "split", "extrapolation"),
  k = 10,
  R = 1,
  B = 200,
  dCV.sw.test = FALSE,
  train.prob = 0.7,
  method.split = c("dCV", "bootstrap", "subbootstrap"),
  print.rw = FALSE
)
```

Arguments

data	A data frame with information about the response variable and covariates, as well as sampling weights and strata and cluster indicators. It could be NULL if the sampling design is indicated in the design argument.
col.y	A numeric value indicating the number of the column in which information on the response variable can be found or a character string indicating the name of that column.
col.x	A numeric vector indicating the numbers of the columns in which information on the covariates can be found or a vector of character strings indicating the names of these columns.
cluster	A character string indicating the name of the column with cluster identifiers. It could be NULL if the sampling design is indicated in the design argument.

strata	A character string indicating the name of the column with strata identifiers. It could be NULL if the sampling design is indicated in the design argument.
weights	A character string indicating the name of the column with sampling weights. It could be NULL if the sampling design is indicated in the design argument.
design	An object of class survey.design generated by survey::svydesign(). It could be NULL if information about cluster, strata, weights and data are given.
family	A character string indicating the family to fit LASSO models. Choose between gaussian (to fit linear models) or binomial (for logistic models).
alpha	A numeric value between 0 and 1, indicating the elastic-net parameter. alpha=1 is the LASSO penalty (equivalent to using wlasso()) and alpha=0 the ridge penalty. Default alpha=1.
lambda.grid	A numeric vector indicating a grid for penalization parameters. The default option is lambda.grid = NULL, which considers the default grid selected by the function glmnet::glmnet().
method	A character string indicating the method to be applied to define replicate weights. Choose between one of these: JKn, dCV, bootstrap, subbootstrap, BRR, split, extrapolation.
k	A numeric value indicating the number of folds to be defined. Default is k=10. Only applies for the dCV method.
R	A numeric value indicating the number of times the sample is partitioned. Default is R=1. Only applies for dCV, split or extrapolation methods.
B	A numeric value indicating the number of bootstrap resamples. Default is B=200. Only applies for bootstrap and subbootstrap methods.
dCV.sw.test	A logical value indicating the method for estimating the error for dCV method. FALSE, (the default option) estimates the error for each test set and defines the cross-validated error based on the average strategy. Option TRUE estimates the cross-validated error based on the pooling strategy
train.prob	A numeric value between 0 and 1, indicating the proportion of clusters (for the method split) or strata (for the method extrapolation) to be set in the training sets. Default is train.prob = 0.7. Only applies for split and extrapolation methods.
method.split	A character string indicating the way in which replicate weights should be defined in the split method. Choose one of the following: dCV, bootstrap or subbootstrap. Only applies for split method.
print.rw	A logical value. If TRUE, the data set with the replicate weights is saved in the output object. Default print.rw=FALSE.

Details

The parameter alpha indicates the penalty to be used to fit the elastic net as in glmnet:

$(1-\alpha)/2||\beta||_2^2 + \alpha||\beta||_1$.

Value

The output object of the function `welnet()` is an object of class `welnet`. This object is a list containing 4 or 5 elements, depending on the value set to the argument `print.rw`. Below we describe the contents of these elements:

- `lambda`: A list containing information of two elements:
  - `grid`: A numeric vector indicating all the values considered for the tuning parameter.
  - `min`: A numeric value indicating the value of the tuning parameter that minimizes the average error (i.e., selected optimal tuning parameter).
- `error`: A list containing information of two elements:
  - `average`: A numeric vector indicating the average error corresponding to each tuning parameter.
  - `all`: A numeric matrix indicating the error of each test set for each tuning parameter.
- `model`: A list containing information of two elements in relation to the fitted models. Note that all these models are fitted considering the whole data set (and not uniquely the training sets).
  - `grid`: A list with the information about the models fitted for each of the tuning parameters considered (i.e., all the values in the `lambda$grid` object):
    - `a0`: a numeric vector of model intercepts across the whole grid of tuning parameters (hence, of the same length as `lambda$grid`).
    - `beta`: a matrix of regression coefficients corresponding to all the considered covariates across the whole grid of tuning parameters (the number of rows is equal to the number of covariates considered and the number of columns to the length of `lambda$grid`).
    - `df`: a numeric vector of the degrees of freedom (i.e., the number of coefficients different from zero) across the whole grid of tuning parameters (hence, of the same length as `lambda$grid`).
  - `min`: A list with the information about the model fitted considering uniquely the tuning parameter that minimizes the error in the training models (i.e., the optimal tuning parameter selected between the elements in `lambda$grid`):
    - `a0`: a numeric value indicating the intercept value of the selected model.
    - `beta`: a matrix of regression coefficients corresponding to all the considered covariates for the selected tuning parameters (the number of rows is equal to the number of covariates considered and the number of columns is one).
    - `df`: a numeric value indicating the degrees of freedom (i.e., the number of coefficients different from zero) of the selected model.
- `data.rw`: A data frame containing the original data set and the replicate weights added to define training and test sets. Only included in the output object if `print.rw=TRUE`.
- `call`: an object containing the information about the way in which the function has been run.

Examples

```
data(simdata_lasso_binomial)
mcv <- welnet(data = simdata_lasso_binomial,
              col.y = "y", col.x = 1:50,
```

```
                family = "binomial",
                alpha = 0.5,
                cluster = "cluster", strata = "strata", weights = "weights",
                method = "dCV", k=10, R=1)

# Or equivalently:

mydesign <- survey::svydesign(ids=~cluster, strata = ~strata, weights = ~weights,
                              nest = TRUE, data = simdata_lasso_binomial)
mcv <- welnet(col.y = "y", col.x = 1:50, design = mydesign,
                family = "binomial", alpha = 0.5,
                method = "dCV", k=10, R=1)
```

`welnet.plot` *Plot welnet object*

Description

Plot welnet object

Usage

```
welnet.plot(x)
```

Arguments

x	an object of class "welnet".

Value

a graph

Examples

```
data(simdata_lasso_binomial)
mcv <- welnet(data = simdata_lasso_binomial, col.y = "y", col.x = 1:50,
                family = "binomial", alpha = 0.5, cluster = "cluster", strata = "strata",
                weights = "weights", method = "dCV", k=5, R=1)
welnet.plot(mcv)
```

wlasso *Weighted LASSO prediction models for complex survey data*

Description

This function allows as to fit LASSO prediction (linear or logistic) models to complex survey data, considering sampling weights in the estimation process and selects the lambda that minimizes the error based on different replicating weights methods.

Usage

```
wlasso(
  data = NULL,
  col.y = NULL,
  col.x = NULL,
  cluster = NULL,
  strata = NULL,
  weights = NULL,
  design = NULL,
  family = c("gaussian", "binomial"),
  lambda.grid = NULL,
 method = c("dCV", "JKn", "bootstrap", "subbootstrap", "BRR", "split", "extrapolation"),
  k = 10,
  R = 1,
  B = 200,
  dCV.sw.test = FALSE,
  train.prob = 0.7,
  method.split = c("dCV", "bootstrap", "subbootstrap"),
  print.rw = FALSE
)
```

Arguments

data	A data frame with information about the response variable and covariates, as well as sampling weights and strata and cluster indicators. It could be NULL if the sampling design is indicated in the design argument.
col.y	A numeric value indicating the number of the column in which information on the response variable can be found or a character string indicating the name of that column.
col.x	A numeric vector indicating the numbers of the columns in which information on the covariates can be found or a vector of character strings indicating the names of these columns.
cluster	A character string indicating the name of the column with cluster identifiers. It could be NULL if the sampling design is indicated in the design argument.
strata	A character string indicating the name of the column with strata identifiers. It could be NULL if the sampling design is indicated in the design argument.

weights	A character string indicating the name of the column with sampling weights. It could be NULL if the sampling design is indicated in the design argument.
design	An object of class survey.design generated by survey::svydesign(). It could be NULL if information about cluster, strata, weights and data are given.
family	A character string indicating the family to fit LASSO models. Choose between gaussian (to fit linear models) or binomial (for logistic models).
lambda.grid	A numeric vector indicating a grid for penalization parameters. The default option is lambda.grid = NULL, which considers the default grid selected by the function glmnet::glmnet().
method	A character string indicating the method to be applied to define replicate weights. Choose between one of these: JKn, dCV, bootstrap, subbootstrap, BRR, split, extrapolation.
k	A numeric value indicating the number of folds to be defined. Default is k=10. Only applies for the dCV method.
R	A numeric value indicating the number of times the sample is partitioned. Default is R=1. Only applies for dCV, split or extrapolation methods.
B	A numeric value indicating the number of bootstrap resamples. Default is B=200. Only applies for bootstrap and subbootstrap methods.
dCV.sw.test	A logical value indicating the method for estimating the error for dCV method. FALSE, (the default option) estimates the error for each test set and defines the cross-validated error based on the average strategy. Option TRUE estimates the cross-validated error based on the pooling strategy
train.prob	A numeric value between 0 and 1, indicating the proportion of clusters (for the method split) or strata (for the method extrapolation) to be set in the training sets. Default is train.prob = 0.7. Only applies for split and extrapolation methods.
method.split	A character string indicating the way in which replicate weights should be defined in the split method. Choose one of the following: dCV, bootstrap or subbootstrap. Only applies for split method.
print.rw	A logical value. If TRUE, the data set with the replicate weights is saved in the output object. Default print.rw=FALSE.

Value

The output object of the function wlasso() is an object of class wlasso. This object is a list containing 4 or 5 elements, depending on the value set to the argument print.rw. Below we describe the contents of these elements:

- lambda: A list containing information of two elements:
  - grid: A numeric vector indicating all the values considered for the tuning parameter.
  - min: A numeric value indicating the value of the tuning parameter that minimizes the average error (i.e., selected optimal tuning parameter).
- error: A list containing information of two elements:

  - average: A numeric vector indicating the average error corresponding to each tuning parameter.
  - all: A numeric matrix indicating the error of each test set for each tuning parameter.
- model: A list containing information of two elements in relation to the fitted models. Note that all these models are fitted considering the whole data set (and not uniquely the training sets).
  - grid: A list with the information about the models fitted for each of the tuning parameters considered (i.e., all the values in the lambda$grid object):
    * a0: a numeric vector of model intercepts across the whole grid of tuning parameters (hence, of the same length as lambda$grid).
    * beta: a matrix of regression coefficients corresponding to all the considered covariates across the whole grid of tuning parameters (the number of rows is equal to the number of covariates considered and the number of columns to the length of lambda$grid).
    * df: a numeric vector of the degrees of freedom (i.e., the number of coefficients different from zero) across the whole grid of tuning parameters (hence, of the same length as lambda$grid).
  - min: A list with the information about the model fitted considering uniquely the tuning parameter that minimizes the error in the training models (i.e., the optimal tuning parameter selected between the elements in lambda$grid):
    * a0: a numeric value indicating the intercept value of the selected model.
    * beta: a matrix of regression coefficients corresponding to all the considered covariates for the selected tuning parameters (the number of rows is equal to the number of covariates considered and the number of columns is one).
    * df: a numeric value indicating the degrees of freedom (i.e., the number of coefficients different from zero) of the selected model.
- data.rw: A data frame containing the original data set and the replicate weights added to define training and test sets. Only included in the output object if print.rw=TRUE.
- call: an object containing the information about the way in which the function has been run.

Examples

```
data(simdata_lasso_binomial)
mcv <- wlasso(data = simdata_lasso_binomial,
              col.y = "y", col.x = 1:50,
              family = "binomial",
              cluster = "cluster", strata = "strata", weights = "weights",
              method = "dCV", k=10, R=1)

# Or equivalently:

mydesign <- survey::svydesign(ids=~cluster, strata = ~strata, weights = ~weights,
                              nest = TRUE, data = simdata_lasso_binomial)
mcv <- wlasso(col.y = "y", col.x = 1:50, design = mydesign,
              family = "binomial",
              method = "dCV", k=10, R=1)
```

`wlasso.plot` *Plot weighted LASSO object*

Description

Plot weighted LASSO object

Usage

```
wlasso.plot(x)
```

Arguments

x	an object of class "wlasso".

Value

a graph

Examples

```
data(simdata_lasso_binomial)
mcv <- wlasso(data = simdata_lasso_binomial, col.y = "y", col.x = 1:50,
              family = "binomial", cluster = "cluster", strata = "strata",
              weights = "weights", method = "dCV", k=10, R=1)
wlasso.plot(mcv)
```

Index