Package 'HTRX'

January 20, 2025

Type Package

Title Haplotype Trend Regression with eXtra Flexibility (HTRX)

Version 1.2.4

Maintainer Yaoling Yang <yaoling.yang@bristol.ac.uk>

Description
Detection of haplotype patterns that include single nucleotide polymorphisms (SNPs) and non-contiguous haplotypes that are associated with a phenotype. Methods for implementing HTRX are described in Yang Y, Lawson DJ (2023) <doi:10.1093/bioadv/vbad038> and Barrie W, Yang Y, Irving-Pease E.K, et al (2024) <doi:10.1038/s41586-023-06618-z>.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Depends R (>= 4.0.0)

Imports fastglm, caret, parallel, methods, stats, glmnet, tune, recipes

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Yaoling Yang [aut, cre] (<https://orcid.org/0000-0003-4905-8097>), Daniel Lawson [aut] (<https://orcid.org/0000-0002-5311-6213>)

Repository CRAN

Date/Publication 2024-02-09 08:00:14 UTC

Contents

HTRX-package *HTRX: Haplotype Trend Regression with eXtra flexibility*

Description

This is the software for "HTRX - Haplotype Trend Regression with eXtra flexibility (HTRX)" based on the papar Genetic risk for Multiple Sclerosis originated in Pastoralist Steppe populations, Barrie W, Yang Y, Attfield K E, et al (2022).

HTRX searches for haplotype patterns that include single nucleotide polymorphisms (SNPs) and non-contiguous haplotypes.

HTRX is a template gives a value for each SNP taking values of '0' or '1', reflecting whether the reference allele of each SNP is present or absent, or an 'X' meaning either value is allowed.

We used a two-step procedure to select the best HTRX model: `do_cv`.

Step 1: select candidate models using AIC, BIC or lasso;

Step 2: select the best model using 10-fold cross-validation.

There is also an option to directly perform 10-fold cross-validation: `do_cv_direct`. This method loses some accuracy and doesn't return the fixed features selected, but saves computational time.

Longer haplotypes are important for discovering interactions. However, too many haplotypes make original HTRX unrealistic for regions with large numbers of SNPs. We proposed "cumulative HTRX" that enables HTRX to run on longer haplotypes: `do_cumulative_htrx`.

The code for HTRX is hosted at `https://github.com/YaolingYang/HTRX`.

Author(s)

Maintainer: Yaoling Yang <yaoling.yang@bristol.ac.uk> (ORCID)

Authors:

- Daniel Lawson <Dan.Lawson@bristol.ac.uk> (ORCID)

References

Yang Y, Lawson DJ. HTRX: an R package for learning non-contiguous haplotypes associated with a phenotype. Bioinformatics Advances 3(1) (2023): vbad038.

Barrie, W., Yang, Y., Irving-Pease, E.K. et al. Elevated genetic risk for multiple sclerosis emerged in steppe pastoralist populations. Nature 625, 321–328 (2024).

Eforn, B. "Bootstrap methods: another look at the jackknife." The Annals of Statistics 7 (1979): 1-26.

Schwarz, Gideon. "Estimating the dimension of a model." The annals of statistics (1978): 461-464.

McFadden, Daniel. "Conditional logit analysis of qualitative choice behavior." (1973).

Akaike, Hirotugu. "A new look at the statistical model identification." IEEE transactions on automatic control 19.6 (1974): 716-723.

Tibshirani, Robert. "Regression shrinkage and selection via the lasso." Journal of the Royal Statistical Society: Series B (Methodological) 58.1 (1996): 267-288.

`computeR2` *Compute variance explained by models*

Description

Compute the variance explained by a linear or generalized linear model.

Usage

```
mypredict(model, newdata)

computeR2(pred, outcome, usebinary = 1)
```

Arguments

`model`	a fitted model, which is the output of `themodel`.
`newdata`	a data frame which contains all the variables included in the model. This data frame is used to make prediction on.
`pred`	a vector of the predicted outcome.
`outcome`	a vector of the actual outcome.
`usebinary`	a non-negative number representing different models. Use linear model if `usebinary=0`, use logistic regression model via `fastglm` if `usebinary=1` (by default), and use logistic regression model via `glm` if `usebinary>1`.

Details

The variance explained by a linear model is based on the conventional R^2. As for logistic regression, we use McFadden's R^2.

Value

`mypredict` returns a vector of the predicted outcome.

`computeR2` returns a positive number of the variance explained by the linear model (conventional R^2) or the generalized linear model (McFadden's R^2).

References

McFadden, Daniel. "Conditional logit analysis of qualitative choice behavior." (1973).

Examples

```
## create datasets
x=matrix(runif(100,-2,2),ncol=5)
outcome=(0.5*x[,2] - 0.8*x[,4] + 0.3*x[,5])>runif(100,-2,2)

## create binary outcome
outcome[outcome]=1
data=data.frame(outcome,x)

## compute the variance explained by features
model=themodel(outcome~.,data[1:80,],usebinary=1)
outcome_predict=mypredict(model,data[81:100,])
computeR2(outcome_predict,data[81:100,'outcome'],usebinary=1)
```

---

data_split *Data split*

Description

`kfold_split` splits data into k folds with equal sizes, which is used for cross-validation. `twofold_split` splits data into two folds, which samples the training set. Both stratified sampling and simple sampling are allowed. The details can be found in function `do_cv` and `do_cumulative_htrx`.

Usage

```
kfold_split(outcome, fold, method = "simple")

twofold_split(outcome, train_proportion = 0.5, method = "simple")
```

Arguments

`outcome`	a vector of the variable (usually the outcome) based on which the data is going to be stratified. This only works when `method="stratified"`.
`fold`	a positive integer specifying how many folds the data should be split into.
`method`	the method to be used for data split, either `"simple"` (default) or `"stratified"`.
`train_proportion`	a positive number between 0 and 1 giving the proportion of the training dataset when splitting data into 2 folds. By default, `train_proportion=0.5`.

Details

Stratified sampling works only when the `outcome` variable is binary (either 0 or 1), and it ensures each fold has almost the same number of `outcome=0` and `outcome=1`.

Simple sampling randomly splits the data into k folds.

Two-fold data split is used to select candidate models in Step 1 of HTRX or cumulative HTRX, while k-fold data split is used for 10-fold cross-validation in Step 2 which aims at selecting the best model.

Value

Both functions return a list containing the indexes of different folds.

Examples

```
## create the binary outcome (20% prevalence)
outcome=rbinom(200,1,0.2)

## simple sampling (10 folds)
kfold_split(outcome,10)

## stratified sampling (10 folds)
kfold_split(outcome,10,"stratified")

## stratified sampling (2 folds, with 50% training data)
twofold_split(outcome,0.5,"stratified")
```

---

do_cumulative_htrx *Cumulative HTRX on long haplotypes*

Description

Two-step cross-validation used to select the best HTRX model for longer haplotypes, i.e. include at least 7 single nucleotide polymorphisms (SNPs).

Usage

```
do_cumulative_htrx(
  data_nosnp,
  hap1,
  hap2 = hap1,
  train_proportion = 0.5,
  sim_times = 5,
  featurecap = 40,
  usebinary = 1,
  randomorder = TRUE,
  fixorder = NULL,
  method = "simple",
```

```
  criteria = "BIC",
  gain = TRUE,
  nmodel = 3,
  runparallel = FALSE,
  mc.cores = 6,
  rareremove = FALSE,
  rare_threshold = 0.001,
  L = 6,
  dataseed = 1:sim_times,
  fold = 10,
  kfoldseed = 123,
  htronly = FALSE,
  max_int = NULL,
  returnwork = FALSE,
  verbose = FALSE
)

do_cumulative_htrx_step1(
  data_nosnp,
  hap1,
  hap2 = hap1,
  train_proportion = 0.5,
  featurecap = 40,
  usebinary = 1,
  randomorder = TRUE,
  fixorder = NULL,
  method = "simple",
  criteria = "BIC",
  nmodel = 3,
  splitseed = 123,
  gain = TRUE,
  runparallel = FALSE,
  mc.cores = 6,
  rareremove = FALSE,
  rare_threshold = 0.001,
  L = 6,
  htronly = FALSE,
  max_int = NULL,
  verbose = FALSE
)

extend_haps(
  data_nosnp,
  featuredata,
  train,
  featurecap = dim(featuredata)[2],
  usebinary = 1,
  gain = TRUE,
```

```
  runparallel = FALSE,
  mc.cores = 6,
  verbose = FALSE
)

make_cumulative_htrx(
  hap1,
  hap2 = hap1,
  featurename,
  rareremove = FALSE,
  rare_threshold = 0.001,
  htronly = FALSE,
  max_int = NULL
)
```

Arguments

data_nosnp	a data frame with outcome (the outcome must be the first column with colnames(data_nosnp)[1]="outcome"), fixed covariates (for example, sex, age and the first 18 PCs) if there are, and without SNPs or haplotypes.
hap1	a data frame of the SNPs' genotype of the first genome. The genotype of a SNP for each individual is either 0 (reference allele) or 1 (alternative allele).
hap2	a data frame of the SNPs' genotype of the second genome. The genotype of a SNP for each individual is either 0 (reference allele) or 1 (alternative allele). By default, hap2=hap1 representing haploid.
train_proportion	a positive number between 0 and 1 giving the proportion of the training dataset when splitting data into 2 folds. By default, train_proportion=0.5.
sim_times	an integer giving the number of simulations in Step 1 (see details). By default, sim_times=5.
featurecap	a positive integer which manually sets the maximum number of independent features. By default, featurecap=40.
usebinary	a non-negative number representing different models. Use linear model if usebinary=0, use logistic regression model via fastglm if usebinary=1 (by default), and use logistic regression model via glm if usebinary>1.
randomorder	logical. If randomorder=TRUE (default), use random order of all the SNPs to add SNPs in cumulative HTRX.
fixorder	a vector of the fixed order of SNPs to be added in cumulative HTRX. This only works by setting randomorder=FALSE. Otherwise, fixorder=NULL (default). The length of fixorder can be smaller than the total number of SNPs, i.e. users can specify the order of some instead of all of the SNPs.
method	the method used for data splitting, either "simple" (default) or "stratified".
criteria	the criteria for model selection, either "BIC" (default), "AIC" or "lasso".
gain	logical. If gain=TRUE (default), report the variance explained in addition to fixed covariates; otherwise, report the total variance explained by all the variables.

nmodel	a positive integer specifying the number of candidate models that the criterion selects. By default, nmodel=3.
runparallel	logical. Use parallel programming based on mclapply function from R package "parallel" or not. Note that for Windows users, mclapply doesn't work, so please set runparallel=FALSE (default).
mc.cores	an integer giving the number of cores used for parallel programming. By default, mc.cores=6. This only works when runparallel=TRUE.
rareremove	logical. Remove rare SNPs and haplotypes or not. By default, rareremove=FALSE.
rare_threshold	a numeric number below which the haplotype or SNP is removed. This only works when rareremove=TRUE. By default, rare_threshold=0.001.
L	a positive integer. The cumulative HTRX starts with haplotypes templates containing L SNPs. By default, L=6. Let nsnp be the number of SNPs in total, L must be smaller than nsnp-1.
dataseed	a vector of the seed that each simulation in Step 1 (see details) uses. The length of dataseed must be the same as sim_times. By default, dataseed=1:sim_times.
fold	a positive integer specifying how many folds the data should be split into for cross-validation.
kfoldseed	a positive integer specifying the seed used to split data for k-fold cross validation. By default, kfoldseed=123.
htronly	logical. If htronly=TRUE, only haplotypes with interaction between all the SNPs will be selected. Please set max_int=NULL when htronly=TRUE. By default, htronly=FALSE.
max_int	a positive integer which specifies the maximum number of SNPs that can interact. If no value is given, interactions between all the SNPs will be considered.
returnwork	logical. If returnwork=TRUE, return a vector of the maximum number of features that are assessed in each simulation, excluding the fixed covariates. This is used to assess how much computational 'work' is done in Step 1(2) of HTRX (see details). By default, returnwork=FALSE.
verbose	logical. If verbose=TRUE, print out the inference steps. By default, verbose=FALSE.
splitseed	a positive integer giving the seed that a single simulation in Step 1 (see details) uses.
featuredata	a data frame of the feature data, e.g. haplotype data created by HTRX or SNPs. These features exclude all the data in data_nosnp, and will be selected using 2-step cross-validation.
train	a vector of the indexes of the training data.
featurename	a character giving the names of features (haplotypes).

Details

Longer haplotypes are important for discovering interactions. However, there are 3^k-1 haplotypes in HTRX if the region contains k SNPs, making HTRX (do_cv) unrealistic to apply on for regions with large numbers of SNPs. To address this issue, we proposed "cumulative HTRX" (do_cumulative_htrx) that enables HTRX to run on longer haplotypes, i.e. haplotypes which include at least 7 SNPs (we recommend). There are 2 steps to implement cumulative HTRX.

Step 1: extend haplotypes and select candidate models.

(1) Randomly sample a subset (50 use stratified sampling when the outcome is binary. This subset is used for all the analysis in (2) and (3);

(2) Start with L randomly chosen SNPs from the entire k SNPs, and keep the top M haplotypes that are chosen from the forward regression. Then add another SNP to the M haplotypes to create 3M+2 haplotypes. There are 3M haplotypes obtained by adding "0", "1" or "X" to the previous M haplotypes, as well as 2 bases of the added SNP, i.e. "XX...X0" and "XX...X1" (as "X" was implicitly used in the previous step). The top M haplotypes from them are then selected using forward regression. Repeat this process until obtaining M haplotypes which include k-1 SNPs;

(3) Add the last SNP to create 3M+2 haplotypes. Afterwards, if `criteria="AIC"` or `criteria="BIC"`, start from a model with fixed covariates (e.g. 18 PCs, sex and age), and perform forward regression on the subset, i.e. iteratively choose a feature (in addition to the fixed covariates) to add whose inclusion enables the model to explain the largest variance, and select s models with the lowest Akaike information criterion (AIC) or Bayesian Information Criteria (BIC) to enter the candidate model pool; If `criteria="lasso"`, using least absolute shrinkage and selection operator (lasso) to directly select the best s models to enter the candidate model pool;

(4) repeat (1)-(3) B times, and select all the different models in the candidate model pool as the candidate models.

Step 2: select the best model using k-fold cross-validation.

(1) Randomly split the whole data into k groups with approximately equal sizes, using stratified sampling when the outcome is binary;

(2) In each of the k folds, use a fold as the validation dataset, a fold as the test dataset, and the remaining folds as the training dataset. Then, fit all the candidate models on the training dataset, and use these fitted models to compute the additional variance explained by features (out-of-sample variance explained) in the validation and test dataset. Finally, select the candidate model with the biggest average out-of-sample variance explained in the validation set as the best model, and report the out-of-sample variance explained in the test set.

Function `do_cumulative_htrx_step1` is the Step 1 (1)-(3) described above. Function `extend_haps` is used to select haplotypes in the Step 1 (2) described above. Function `make_cumulative_htrx` is used to generate the haplotype data (by adding a new SNP into the haplotypes) from M haplotypes to 3M+2 haplotypes, which is also described in the Step 1 (2)-(3).

When investigating haplotypes with interactions between at most 2 SNPs, L is suggested to be no bigger than 10. When investigating haplotypes with interactions between at most 3 SNPs, L should not be bigger than 9. If haplotypes with interactions between more than 4 SNPs are investigated, L is suggested to be 6 (which is the default value).

Value

`do_cumulative_htrx` returns a list containing the best model selected, and the out-of-sample variance explained in each test set.

`do_cv_step1` returns a list of three candidate models selected by a single simulation.

`extend_haps` returns a character of the names of the selected features.

`make_cumulative_htrx` returns a data frame of the haplotype matrix.

References

Yang Y, Lawson DJ. HTRX: an R package for learning non-contiguous haplotypes associated with a phenotype. Bioinformatics Advances 3.1 (2023): vbad038.

Barrie, W., Yang, Y., Irving-Pease, E.K. et al. Elevated genetic risk for multiple sclerosis emerged in steppe pastoralist populations. Nature 625, 321–328 (2024).

Eforn, B. "Bootstrap methods: another look at the jackknife." The Annals of Statistics 7 (1979): 1-26.

Schwarz, Gideon. "Estimating the dimension of a model." The annals of statistics (1978): 461-464.

McFadden, Daniel. "Conditional logit analysis of qualitative choice behavior." (1973).

Akaike, Hirotugu. "A new look at the statistical model identification." IEEE transactions on automatic control 19.6 (1974): 716-723.

Tibshirani, Robert. "Regression shrinkage and selection via the lasso." Journal of the Royal Statistical Society: Series B (Methodological) 58.1 (1996): 267-288.

Examples

```
## use dataset "example_hap1", "example_hap2" and "example_data_nosnp"
## "example_hap1" and "example_hap2" are
## both genomes of 8 SNPs for 5,000 individuals (diploid data)
## "example_data_nosnp" is a simulated dataset
## which contains the outcome (binary), sex, age and 18 PCs

## visualise the covariates data
## we will use only the first two covariates: sex and age in the example
head(HTRX::example_data_nosnp)

## visualise the genotype data for the first genome
head(HTRX::example_hap1)

## we perform cumulative HTRX on all the 8 SNPs using 2-step cross-validation
## to compute additional variance explained by haplotypes
## If the data is haploid, please set hap2=HTRX::example_hap1
## If you want to compute total variance explained, please set gain=FALSE
## For Linux/MAC users, we recommend setting runparallel=TRUE

cumu_CV_results <- do_cumulative_htrx(HTRX::example_data_nosnp[1:500,1:3],
                                      HTRX::example_hap1[1:500,],
                                      HTRX::example_hap2[1:500,],
                                      train_proportion=0.5,sim_times=1,
                                      featurecap=10,usebinary=1,
                                      randomorder=TRUE,method="stratified",
                                      criteria="BIC",gain=TRUE,
                                      runparallel=FALSE,verbose=TRUE)

#This result would be more precise when setting larger sim_times and featurecap
```

do_cv *Two-stage HTRX: Model selection on short haplotypes*

Description

Two-step cross-validation used to select the best HTRX model. It can be applied to select haplotypes based on HTR, or select single nucleotide polymorphisms (SNPs).

Usage

```
do_cv(
  data_nosnp,
  featuredata,
  train_proportion = 0.5,
  sim_times = 5,
  featurecap = dim(featuredata)[2],
  usebinary = 1,
  method = "simple",
  criteria = "BIC",
  gain = TRUE,
  nmodel = 3,
  dataseed = 1:sim_times,
  runparallel = FALSE,
  mc.cores = 6,
  fold = 10,
  kfoldseed = 123,
  returnwork = FALSE,
  verbose = FALSE
)

do_cv_step1(
  data_nosnp,
  featuredata,
  train_proportion = 0.5,
  featurecap = dim(featuredata)[2],
  usebinary = 1,
  method = "simple",
  criteria = "BIC",
  nmodel = 3,
  splitseed = 123,
  runparallel = FALSE,
  mc.cores = 6,
  verbose = FALSE
)

infer_step1(
  data_nosnp,
```

```
  featuredata,
  train,
  criteria = "BIC",
  featurecap = dim(featuredata)[2],
  usebinary = 1,
  nmodel = nmodel,
  runparallel = FALSE,
  mc.cores = 6,
  verbose = FALSE
)

infer_fixedfeatures(
  data_nosnp,
  featuredata,
  train = (1:nrow(data_nosnp))[-test],
  test,
  features,
  coefficients = NULL,
  gain = TRUE,
  usebinary = 1,
  R2only = FALSE,
  verbose = FALSE
)
```

Arguments

data_nosnp	a data frame with outcome (the outcome must be the first column with colnames(data_nosnp)[1]="outcome"), fixed covariates (for example, sex, age and the first 18 PCs) if there are, and without SNPs or haplotypes.
featuredata	a data frame of the feature data, e.g. haplotype data created by HTRX or SNPs. These features exclude all the data in data_nosnp, and will be selected using 2-step cross-validation.
train_proportion	a positive number between 0 and 1 giving the proportion of the training dataset when splitting data into 2 folds. By default, train_proportion=0.5.
sim_times	an integer giving the number of simulations in Step 1 (see details). By default, sim_times=5.
featurecap	a positive integer which manually sets the maximum number of independent features. By default, featurecap=40.
usebinary	a non-negative number representing different models. Use linear model if usebinary=0, use logistic regression model via fastglm if usebinary=1 (by default), and use logistic regression model via glm if usebinary>1.
method	the method used for data splitting, either "simple" (default) or "stratified".
criteria	the criteria for model selection, either "BIC" (default), "AIC" or "lasso".
gain	logical. If gain=TRUE (default), report the variance explained in addition to fixed covariates; otherwise, report the total variance explained by all the variables.

nmodel	a positive integer specifying the number of candidate models that the criterion selects. By default, nmodel=3.
dataseed	a vector of the seed that each simulation in Step 1 (see details) uses. The length of dataseed must be the same as sim_times. By default, dataseed=1:sim_times.
runparallel	logical. Use parallel programming based on mclapply function from R package "parallel" or not. Note that for Windows users, mclapply doesn't work, so please set runparallel=FALSE (default).
mc.cores	an integer giving the number of cores used for parallel programming. By default, mc.cores=6. This only works when runparallel=TRUE.
fold	a positive integer specifying how many folds the data should be split into for cross-validation.
kfoldseed	a positive integer specifying the seed used to split data for k-fold cross validation. By default, kfoldseed=123.
returnwork	logical. If returnwork=TRUE, return a vector of the maximum number of features that are assessed in each simulation, excluding the fixed covariates. This is used to assess how much computational 'work' is done in Step 1(2) of HTRX (see details). By default, returnwork=FALSE.
verbose	logical. If verbose=TRUE, print out the inference steps. By default, verbose=FALSE.
splitseed	a positive integer giving the seed of data split.
train	a vector of the indexes of the training data.
test	a vector of the indexes of the test data.
features	a character of the names of the fixed features, excluding the intercept.
coefficients	a vector giving the coefficients of the fixed features, including the intercept. If the fixed features don't have fixed coefficients, set coefficients=NULL (default).
R2only	logical. If R2only=TRUE, function infer_fixedfeatures only returns the variance explained in the test data. By default, R2only=FALSE.

Details

Function do_cv is the main function used for selecting haplotypes from HTRX or SNPs. It is a two-step algorithm and is used for alleviating overfitting.

Step 1: select candidate models. This is to address the model search problem, and is chosen to obtain a set of models more diverse than traditional bootstrap resampling.

(1) Randomly sample a subset (50 Specifically, when the outcome is binary, stratified sampling is used to ensure the subset has approximately the same proportion of cases and controls as the whole data;

(2) If criteria="AIC" or criteria="BIC", start from a model with fixed covariates (e.g. 18 PCs, sex and age), and perform forward regression on the subset, i.e. iteratively choose a feature (in addition to the fixed covariates) to add whose inclusion enables the model to explain the largest variance, and select s models with the lowest Akaike information criterion (AIC) or Bayesian Information Criteria (BIC) to enter the candidate model pool; If criteria="lasso", using least absolute shrinkage and selection operator (lasso) to directly select the best s models to enter the candidate model pool;

(3) repeat (1)-(2) B times, and select all the different models in the candidate model pool as the candidate models.

Step 2: select the best model using k-fold cross-validation.

(1) Randomly split the whole data into k groups with approximately equal sizes, using stratified sampling when the outcome is binary;

(2) In each of the k folds, use a fold as the validation dataset, a fold as the test dataset, and the remaining folds as the training dataset. Then, fit all the candidate models on the training dataset, and use these fitted models to compute the additional variance explained by features (out-of-sample variance explained) in the validation and test dataset. Finally, select the candidate model with the biggest average out-of-sample variance explained in the validation set as the best model, and report the out-of-sample variance explained in the test set.

Function `do_cv_step1` is the Step 1 (1)-(2) described above. Function `infer_step1` is the Step 1 (2) described above. Function `infer_fixedfeatures` is used to fit all the candidate models on the training dataset, and compute the additional variance explained by features (out-of-sample R2) in the test dataset, as described in the Step 2 (2) above.

Value

`do_cv` returns a list containing the best model selected, and the out-of-sample variance explained in each test set.

`do_cv_step1` and `infer_step1` return a list of three candidate models selected by a single simulation.

`infer_fixedfeatures` returns a list of the variance explained in the test set if `R2only=TRUE`, otherwise, it returns a list of the variance explained in the test set, the model including all the variables, and the null model, i.e. the model with outcome and fixed covariates only.

References

Yang Y, Lawson DJ. HTRX: an R package for learning non-contiguous haplotypes associated with a phenotype. Bioinformatics Advances 3.1 (2023): vbad038.

Barrie, W., Yang, Y., Irving-Pease, E.K. et al. Elevated genetic risk for multiple sclerosis emerged in steppe pastoralist populations. Nature 625, 321–328 (2024).

Eforn, B. "Bootstrap methods: another look at the jackknife." The Annals of Statistics 7 (1979): 1-26.

Schwarz, Gideon. "Estimating the dimension of a model." The annals of statistics (1978): 461-464.

McFadden, Daniel. "Conditional logit analysis of qualitative choice behavior." (1973).

Akaike, Hirotugu. "A new look at the statistical model identification." IEEE transactions on automatic control 19.6 (1974): 716-723.

Tibshirani, Robert. "Regression shrinkage and selection via the lasso." Journal of the Royal Statistical Society: Series B (Methodological) 58.1 (1996): 267-288.

Examples

```
## use dataset "example_hap1", "example_hap2" and "example_data_nosnp"
## "example_hap1" and "example_hap2" are
## both genomes of 8 SNPs for 5,000 individuals (diploid data)
```

```
## "example_data_nosnp" is an example dataset
## which contains the outcome (binary), sex, age and 18 PCs

## visualise the covariates data
## we will use only the first two covariates: sex and age in the example
head(HTRX::example_data_nosnp)

## visualise the genotype data for the first genome
head(HTRX::example_hap1)

## we perform HTRX on the first 4 SNPs
## we first generate all the haplotype data, as defined by HTRX
HTRX_matrix=make_htrx(HTRX::example_hap1[1:300,1:4],
                      HTRX::example_hap2[1:300,1:4])

## If the data is haploid, please set
## HTRX_matrix=make_htrx(HTRX::example_hap1[1:300,1:4],
##                       HTRX::example_hap1[1:300,1:4])

## then perform HTRX using 2-step cross-validation in a single small example
## to compute additional variance explained by haplotypes
## If you want to compute total variance explained, please set gain=FALSE
CV_results <- do_cv(HTRX::example_data_nosnp[1:300,1:2],
                    HTRX_matrix,train_proportion=0.5,
                    sim_times=1,featurecap=4,usebinary=1,
                    method="simple",criteria="BIC",
                    gain=TRUE,runparallel=FALSE,verbose=TRUE)

#This result would be more precise when setting larger sim_times and featurecap
```

`do_cv_direct` *Direct HTRX: k-fold cross-validation on short haplotypes*

Description

Direct k-fold cross-validation used to compute the out-of-sample variance explained by selected features from HTRX. It can be applied to select haplotypes based on HTR, or select single nucleotide polymorphisms (SNPs).

Usage

```
do_cv_direct(
  data_nosnp,
  featuredata,
  featurecap = dim(featuredata)[2],
  usebinary = 1,
  method = "simple",
  criteria = "BIC",
  gain = TRUE,
```

```
  runparallel = FALSE,
  mc.cores = 6,
  fold = 10,
  kfoldseed = 123,
  verbose = FALSE
)
```

Arguments

data_nosnp	a data frame with outcome (the outcome must be the first column), fixed covariates (for example, sex, age and the first 18 PCs) if there are, and without SNPs or haplotypes.
featuredata	a data frame of the feature data, e.g. haplotype data created by HTRX or SNPs. These features exclude all the data in data_nosnp, and will be selected using 2-step cross-validation.
featurecap	a positive integer which manually sets the maximum number of independent features. By default, featurecap=40.
usebinary	a non-negative number representing different models. Use linear model if usebinary=0, use logistic regression model via fastglm if usebinary=1 (by default), and use logistic regression model via glm if usebinary>1.
method	the method used for data splitting, either "simple" (default) or "stratified".
criteria	the criteria for model selection, either "BIC" (default), "AIC" or "lasso".
gain	logical. If gain=TRUE (default), report the variance explained in addition to fixed covariates; otherwise, report the total variance explained by all the variables.
runparallel	logical. Use parallel programming based on mclapply function from R package "parallel" or not. Note that for Windows users, mclapply doesn't work, so please set runparallel=FALSE (default).
mc.cores	an integer giving the number of cores used for parallel programming. By default, mc.cores=6. This only works when runparallel=TRUE.
fold	a positive integer specifying how many folds the data should be split into for cross-validation.
kfoldseed	a positive integer specifying the seed used to split data for k-fold cross validation. By default, kfoldseed=123.
verbose	logical. If verbose=TRUE, print out the inference steps. By default, verbose=FALSE.

Details

Function do_cv_direct directly performs k-fold cross-validation: features are selected from the training set using a specified criteria, and the out-of-sample variance explained by the selected features are computed on the test set. This function runs faster than do_cv with large sim_times, but may lose some accuracy, and it doesn't return a fixed set of features.

Value

do_cv_direct returns a list of the out-of-sample variance explained in each of the test set, and the features selected in each of the k training sets.

References

Yang Y, Lawson DJ. HTRX: an R package for learning non-contiguous haplotypes associated with a phenotype. Bioinformatics Advances 3.1 (2023): vbad038.

Barrie, W., Yang, Y., Irving-Pease, E.K. et al. Elevated genetic risk for multiple sclerosis emerged in steppe pastoralist populations. Nature 625, 321–328 (2024).

Eforn, B. "Bootstrap methods: another look at the jackknife." The Annals of Statistics 7 (1979): 1-26.

Schwarz, Gideon. "Estimating the dimension of a model." The annals of statistics (1978): 461-464.

McFadden, Daniel. "Conditional logit analysis of qualitative choice behavior." (1973).

Akaike, Hirotugu. "A new look at the statistical model identification." IEEE transactions on automatic control 19.6 (1974): 716-723.

Tibshirani, Robert. "Regression shrinkage and selection via the lasso." Journal of the Royal Statistical Society: Series B (Methodological) 58.1 (1996): 267-288.

Examples

```
## use dataset "example_hap1", "example_hap2" and "example_data_nosnp"
## "example_hap1" and "example_hap2" are
## both genomes of 8 SNPs for 5,000 individuals (diploid data)
## "example_data_nosnp" is an example dataset
## which contains the outcome (binary), sex, age and 18 PCs

## visualise the covariates data
## we will use only the first two covariates: sex and age in the example
head(HTRX::example_data_nosnp)

## visualise the genotype data for the first genome
head(HTRX::example_hap1)

## we perform HTRX on the first 4 SNPs
## we first generate all the haplotype data, as defined by HTRX
HTRX_matrix=make_htrx(HTRX::example_hap1[,1:4],
                      HTRX::example_hap2[,1:4])

## If the data is haploid, please set
## HTRX_matrix=make_htrx(HTRX::example_hap1[,1:4],
##                       HTRX::example_hap1[,1:4])

## next compute the maximum number of independent features
featurecap=htrx_max(nsnp=4,cap=10)
## then perform HTRX using direct cross-validation
## If we want to compute the total variance explained
## we can set gain=FALSE in the above example

htrx_results <- do_cv_direct(HTRX::example_data_nosnp[,1:3],
                             HTRX_matrix,featurecap=featurecap,
                             usebinary=1,method="stratified",
                             criteria="lasso",gain=TRUE,
                             runparallel=FALSE,verbose=TRUE)
```

example_data_nosnp *Example covariate data*

Description

Example covariate data including outcome (binary), sex, age and 18 PCs for 5,000 individuals.

Usage

```
data("example_data_nosnp")
```

Format

A data frame with 5,000 observations on a binary outcome named `outcome` and 20 numeric covariates named `sex`, `age` and `PC1-PC18`.

Examples

```
data(example_data_nosnp)
```

example_hap1 *Example genotype data for the first genome*

Description

Example genotype data for the first genome of 8 SNPs for 5,000 individuals.

Usage

```
data("example_hap1")
```

Format

A data frame with 5,000 observations on 8 binary variables named `SNP1-SNP8` ("0" denotes the reference allele while "1" denotes the alternative allele).

Examples

```
data(example_hap1)
```

example_hap2 *Example genotype data for the second genome*

Description

Example genotype data for the second genome of 8 SNPs for 5,000 individuals.

Usage

```
data("example_hap2")
```

Format

A data frame with 5,000 observations on 8 binary variables named SNP1-SNP8 ("0" denotes the reference allele while "1" denotes the alternative allele).

Examples

```
data(example_hap2)
```

htrx_max *Maximum independent features for HTRX*

Description

The maximum number of independent features in principle from haplotypes (i.e. interactions between SNPs) generated by Haplotype Trend Regression with eXtra flexibility (HTRX).

Usage

```
htrx_max(nsnp, n_haps = NULL, cap = 40, max_int = NULL, htr = FALSE)
```

Arguments

nsnp	a positive integer giving the number of single nucleotide polymorphisms (SNPs) included in the haplotypes.
n_haps	a positive integer giving the number of haplotypes, which is also the number of columns of the HTRX or HTR matrix.
cap	a positive integer which manually sets the maximum number of independent features. By default, cap=40.
max_int	a positive integer which specifies the maximum number of SNPs that can interact. If no value is given (by default), interactions between all the SNPs will be considered.
htr	logical. If htr=TRUE, the functions returns the maximum number of independent features for HTR. By default, htr=FALSE.

Details

The maximum number of independent features in principle is $2^n snp$-1 for haplotypes containing interactions between all different numbers of SNPs. However, if `max_int` < `nsnp`, i.e. only the interactions between at most `max_int` SNPs are investigated, there will be fewer maximum number of independent features. You can also manually set the upper limit of independent features (by setting `cap`) that can be included in the final HTRX or HTR model.

Value

`htrx_max` returns a positive integer giving the maximum number of independent features to be included in the analysis.

Examples

```
## the maximum number of independent haplotypes consisted of 4 SNPs from HTRX
htrx_max(nsnp=4,n_haps=(3^4-1))
```

---

htrx_nfeatures *Total number of features for HTRX*

Description

The total number of features in principle from haplotypes (i.e. interactions between SNPs) generated by Haplotype Trend Regression with eXtra flexibility (HTRX) .

Usage

```
htrx_nfeatures(nsnp, max_int = NULL, htr = FALSE)
```

Arguments

`nsnp`	a positive integer giving the number of single nucleotide polymorphisms (SNPs) included in the haplotypes.
`max_int`	a positive integer which specifies the maximum number of SNPs that can interact. If no value is given (by default), interactions between all the SNPs will be considered.
`htr`	logical. If `htr=TRUE`, the function returns the total number of features for HTR. By default, `htr=FALSE`.

Details

The total number of features in principle is $2^n snp$-1 for haplotypes containing interactions between all different numbers of SNPs. However, if `max_int` < `nsnp`, i.e. only the interactions between at most `max_int` SNPs are investigated, there will be fewer total number of features.

Value

htrx_nfeatures returns a positive integer giving the total number of features that each analysis includes.

Examples

```
## the total number of haplotypes consisted of 6 SNPs
## for at most 3-SNP interactions
htrx_nfeatures(nsnp=6,max_int=3)
```

---

make_htrx *Generate haplotype data*

Description

Generate the feature data, either the genotype data for single nucleotide polymorphisms (SNPs) (make_snp), the feature data for Haplotype Trend Regression (HTR) (make_htr), or the feature data for Haplotype Trend Regression with eXtra flexibility (HTRX) (make_htrx).

Usage

```
make_htrx(
  hap1,
  hap2 = hap1,
  rareremove = FALSE,
  rare_threshold = 0.001,
  fixedfeature = NULL,
  max_int = NULL
)

make_htr(hap1, hap2 = hap1, rareremove = FALSE, rare_threshold = 0.001)

make_snp(hap1, hap2 = hap1, rareremove = FALSE, rare_threshold = 0.001)
```

Arguments

hap1	a data frame of the SNPs' genotype of the first genome. The genotype of a SNP for each individual is either 0 (reference allele) or 1 (alternative allele).
hap2	a data frame of the SNPs' genotype of the second genome. The genotype of a SNP for each individual is either 0 (reference allele) or 1 (alternative allele). By default, hap2=hap1 representing haploid.
rareremove	logical. Remove rare SNPs and haplotypes or not. By default, rareremove=FALSE.
rare_threshold	a numeric number below which the haplotype or SNP is removed. This only works when rareremove=TRUE. By default, rare_threshold=0.001.
fixedfeature	a character consisted of the names of haplotypes. This parameter can be NULL (by default) if all the haplotypes are used as variables.
max_int	a positive integer which specifies the maximum number of SNPs that can interact. If no value is given, interactions between all the SNPs will be considered.

Details

If there are n SNPs, there are 2^n different haplotypes created by HTR, and 3^n-1 different haplotypes created by HTRX.

When the data is haploid, please use the default setting hap2=hap1.

Value

a data frame of the feature data (either for SNPs, HTR or HTRX).

Examples

```
## create SNP data for both genomes (diploid data)
hap1=as.data.frame(matrix(0,nrow=100,ncol=4))
hap2=as.data.frame(matrix(0,nrow=100,ncol=4))
colnames(hap1)=colnames(hap2)=c('a','b','c','d')
p=runif(4,0.01,0.99)
for(j in 1:4){
  hap1[,j]=rbinom(100,1,p[j])
  hap2[,j]=rbinom(100,1,p[j])
}

## create the SNP data without removing rare SNPs
make_snp(hap1,hap2)

## create feature data for "HTR" removing haplotypes rarer than 0.5%
make_htr(hap1,hap2,rareremove=TRUE,0.005)

## create feature data for "HTRX"
## retaining haplotypes with interaction across at most 3 SNPs
make_htrx(hap1,hap2,max_int=3)

## create feature data for feature "01XX" and "X101"
## without removing haplotypes
make_htrx(hap1,hap2,fixedfeature=c("01XX","X101"))

## If the data is haploid instead of diploid
## create feature data for "HTRX" without removing haplotypes
make_htrx(hap1,hap1)
```

---

themodel *Model fitting*

Description

Model-agnostic functions for model fitting (both linear and generalized linear models).

Usage

```
themodel(formula, data, usebinary = 1, clean = TRUE)
```

Arguments

formula	a formula for model-fitting, starting with outcome~.
data	a data frame contains all the variables included in the formula. The outcome must be the first column with colnames(data)[1]="outcome".
usebinary	a non-negative number representing different models. Use linear model if usebinary=0, use logistic regression model via fastglm if usebinary=1 (by default), and use logistic regression model via glm if usebinary>1.
clean	logical. If clean=TRUE (by default), remove additional storages that the predict function, "AIC" and "BIC" criteria do not need.

Details

This function returns a fitted model (either linear model or logistic regression model). For logistic regression, we use function fastglm from fastglm package, which is much faster than glm.

Value

a fitted model.

Examples

```
## create the dataset for variables and outcome
x=matrix(runif(100,-2,2),ncol=5)
outcome=0.5*x[,2] - 0.8*x[,4] + 0.3*x[,5]
data1=data.frame(outcome,x)

## fit a linear model
themodel(outcome~.,data1,usebinary=0)

## create binary outcome
outcome=outcome>runif(100,-2,2)
outcome[outcome]=1
data2=data.frame(outcome,x)

## fit a logistic regression model
themodel(outcome~.,data2,usebinary=1)
```

Index