Package ‘SMLE’

January 20, 2025

Title Joint Feature Screening via Sparse MLE

Version 2.1-1

Maintainer Qianxiang Zang <qzang023@uottawa.ca>

Description Feature screening is a powerful tool in processing ultrahigh dimensional data. It attempts to screen out most irrelevant features in preparation for a more elaborate analysis. Xu and Chen (2014)<doi:10.1080/01621459.2013.879531> proposed an effective screening method SMLE, which naturally incorporates the joint effects among features in the screening process. This package provides an efficient implementation of SMLE-screening for high-dimensional linear, logistic, and Poisson models. The package also provides a function for conducting accurate post-screening feature selection based on an iterative hard-thresholding procedure and a user-specified selection criterion.

License GPL-3

Depends R(>= 4.0.0)

Imports glmnet, matrixcalc, mvnfast

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

NeedsCompilation no

Author Qianxiang Zang [aut, cre],
Chen Xu [aut],
Kelly Burkett [aut],

Repository CRAN

Date/Publication 2024-02-12 21:50:03 UTC

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Contents

SMLE-package 2
coef.smle 3
Gen_Data 4

logLik.smle 7
plot.selection 7
plot.smle 8
predict.smle 9
print.smle 10
pvals 11
SMLE 11
smle_select 16
summary.smle 19
synSNP 20
vote_update 20

Index **22**

`SMLE-package` *Joint SMLE-screening for generalized linear models*

Description

Feature screening is a powerful tool in processing ultrahigh dimensional data. It attempts to screen out most irrelevant features in preparation for a more elaborate analysis. This package provides an efficient implementation of SMLE-screening for linear, logistic, and Poisson models, where the joint effects among features are naturally incorporated in the screening process. The package also provides a function for conducting accurate post-screening feature selection based on an iterative hard-thresholding procedure and a user-specified selection criterion.

Details

Package:	smle
Type:	Package
Version:	2.1-1
Date:	2024-02-12
License:	GPL-3

Input a $n \times 1$ response vector Y and a $n \times p$ predictor (feature) matrix X. The package outputs a set of $k < n$ features that seem to be most relevant for joint regression. Moreover, the package provides a data simulator that generates synthetic datasets from high-dimensional GLMs, which accommodate both numerical and categorical features with commonly used correlation structures.

Key functions:
`Gen_Data`
`SMLE`
`smle_select`

Author(s)

Qianxiang Zang, Chen Xu, Kelly Burkett
Maintainer: Qianxiang Zang <qzang023@uottawa.ca>

References

Xu, C. and Chen, J. (2014) The Sparse MLE for Ultrahigh-Dimensional Feature Screening *Journal of the American Statistical Association*, **109**(507), 1257–1269.

Friedman, J., Hastie, T. and Tibshirani, R. (2010) Regularization Paths for Generalized Linear Models via Coordinate Descent *Journal of Statistical Software*, **33**(1), 1-22.

Examples

```
set.seed(1)
#Generate correlated data
Data <- Gen_Data(n = 200, p = 5000, correlation = "MA",family = "gaussian")
print(Data)

# joint feature screening via SMLE
fit <- SMLE(Y = Data$Y, X = Data$X, k = 10, family = "gaussian")
print(fit)
summary(fit)
plot(fit)

#Are there any features missed after screening?
setdiff(Data$subset_true, fit$ID_retained)

# Elaborative selection after screening
fit_s <- smle_select(fit, gamma_ebic = 0.5, vote = FALSE)

#Are there any features missed after selection?
setdiff(Data$subset_true, fit_s$ID_selected)
print(fit_s)
summary(fit_s)
plot(fit_s)
```

---

coef.smle *Extract coefficients from fitted model*

Description

Extract coefficients from fitted model for either a `'smle'` or `'selection'` object.

Usage

```
## S3 method for class 'smle'
coef(object, refit = TRUE, ...)

## S3 method for class 'selection'
coef(object, refit = TRUE, ...)
```

Arguments

`object`	Returned object from either the function `SMLE()` or `smle_select()`.
`refit`	A logical flag that controls what coefficients are being return. Default is `TRUE`.
`...`	This argument is not used and listed for method consistency.

Value

Fitted coefficients based on the screened or selected model specified in the object. If `refit = TRUE`, the coefficients are estimated by re-fitting the final screened/selected model with `glm()`. If `refit = FALSE` the coefficients estimated by the IHT algorithm are returned.

Examples

```
set.seed(1)
Data<-Gen_Data(n=100, p=5000, family = "gaussian", correlation="ID")
fit<-SMLE(Y = Data$Y, X = Data$X, k=15, family = "gaussian")
coef(fit)
fit_s<-smle_select(fit)
coef(fit_s)
```

Gen_Data *Data simulator for high-dimensional GLMs*

Description

This function generates synthetic datasets from GLMs with a user-specified correlation structure. It permits both numerical and categorical features, whose quantity can be larger than the sample size.

Usage

```
Gen_Data(
  n = 200,
  p = 1000,
  sigma = 1,
  num_ctgidx = NULL,
  pos_ctgidx = NULL,
  num_truecoef = NULL,
  pos_truecoef = NULL,
  level_ctgidx = NULL,
```

```
  effect_truecoef = NULL,
  correlation = c("ID", "AR", "MA", "CS"),
  rho = 0.2,
  family = c("gaussian", "binomial", "poisson")
)
```

Arguments

`n`	Sample size, number of rows for the feature matrix to be generated.
`p`	Number of columns for the feature matrix to be generated.
`sigma`	Parameter for noise level.
`num_ctgidx`	The number of features that are categorical. Set to `FALSE` for only numerical features. Default is `FALSE`.
`pos_ctgidx`	Vector of indices denoting which columns are categorical.
`num_truecoef`	The number of features (columns) that affect response. Default is 5.
`pos_truecoef`	Vector of indices denoting which features (columns) affect the response variable. If not specified, positions are randomly sampled. See Details for more information.
`level_ctgidx`	Vector to indicate the number of levels for the categorical features in `pos_ctgidx`. Default is 2.
`effect_truecoef`	Effect size corresponding to the features in `pos_truecoef`. If not specified, effect size is sampled based on a uniform distribution and direction is randomly sampled. See Details.
`correlation`	Correlation structure among features. `correlation = "ID"` for independent, `correlation = 'MA'` for moving average, `correlation = "CS"` for compound symmetry, `correlation = "AR"` for auto regressive. Default is `"ID"`. For more information see Details.
`rho`	Parameter controlling the correlation strength, default is `0.2`. See Details.
`family`	Model type for the response variable. `"gaussian"` for normally distributed data, `poisson` for non-negative counts, `"binomial"` for binary (0-1).

Details

Simulated data (y_i, x_i) where $x_i = (x_{i1}, x_{i2}, ..., x_{ip})$ are generated as follows: First, we generate a p by 1 model coefficient vector beta with all entries being zero, except for the positions specified in `pos_truecoef`, on which `effect_truecoef` is used. When `pos_truecoef` is not specified, we randomly choose `num_truecoef` positions from the coefficient vector. When `effect_truecoef` is not specified, we randomly set the strength of the true model coefficients as follow:

$$(0.5 + U)Z,$$

where U is sampled from a uniform distribution from 0 to 1, and Z is sampled from a binomial distribution $P(Z = 1) = 1/2, P(Z = -1) = 1/2$.

Next, we generate a n by p feature matrix X according to the model selected with `correlation` and specified as follows.

Independent (ID): all features are independently generated from $N(0,1)$.

Moving average (MA): candidate features $x_1, ..., x_p$ are joint normal, marginally $N(0,1)$, with $cov(x_j, x_{j-1}) = \rho$, $cov(x_j, x_{j-2}) = \rho/2$ and $cov(x_j, x_h) = 0$ for $|j-h| > 3$.

Compound symmetry (CS): candidate features $x_1, ..., x_p$ are joint normal, marginally $N(0,1)$, with $cov(x_j, x_h) = \rho/2$ if j, h are both in the set of important features and $cov(x_j, x_h) = \rho$ when only one of j or h are in the set of important features.

Auto-regressive (AR): candidate features $x_1, ..., x_p$ are joint normal, marginally $N(0,1)$, with $cov(x_j, x_h) = \rho^{|j-h|}$ for all j and h. The correlation strength ρ is controlled by the argument `rho`.

Then, we generate the response variable Y according to its response type, which is controlled by the argument `family` For the Gaussian model, $y_i = x_i\beta + \epsilon_i$ where ϵ_i is $N(0,1)$ for i from 1 to n. For the binary model let $\pi_i = P(Y = 1|x_i)$. We sample y_i from Bernoulli(π_i) where logit(π_i) $= x_i\beta$. Finally, for the Poisson model, y_i is generated from the Poisson distribution with the link π_i = $\exp(x_i\beta)$. For more details see the reference below.

Value

`call`	The call that produced this object.
`Y`	Response variable vector of length n.
`X`	Feature matrix or data.frame (matrix if `num_ctgidx =FALSE` and data.frame otherwise).
`subset_true`	Vector of column indices of X for the features that affect the response variables (relevant features).
`coef_true`	Vector of effects for the features that affect the response variables.
`categorical`	Logical flag whether the model contains categorical features.
`CI`	Indices of categorical features when `categorical = TRUE`.

rho,family,correlation are return of arguments passed in the function call.

References

Xu, C. and Chen, J. (2014). The Sparse MLE for Ultrahigh-Dimensional Feature Screening, *Journal of the American Statistical Association*, **109**(507), 1257-1269

Examples

```
#Simulating data with binomial response and auto-regressive structure.
set.seed(1)
Data <- Gen_Data(n = 500, p = 2000, family = "binomial", correlation = "AR")
cor(Data$X[,1:5])
print(Data)
```

`logLik.smle` *Extract log-likelihood*

Description

This is a method written to extract the log-likelihood from 'smle' and 'selection' objects. It refits the model by glm() based on the response and the features selected after screening or selection, and returns an object of 'logLik' from the generic.

Usage

```
## S3 method for class 'smle'
logLik(object, ...)

## S3 method for class 'selection'
logLik(object, ...)
```

Arguments

object	An object of class 'smle' or 'sdata'.
...	Forwarded arguments.

Value

Returns an object of class 'logLik'. This is a number with at least one attribute, "df" (degrees of freedom), giving the number of (estimated) parameters in the model. For more details, see the generic logLik() in **stats**.

Examples

```
set.seed(1)
Data<-Gen_Data(n=100, p=5000, family = "gaussian", correlation="ID")
fit<-SMLE(Y=Data$Y, X=Data$X, k=9, family = "gaussian")
logLik(fit)
```

`plot.selection` *Plots to visualize the post-screening selection*

Description

This function constructs a sparsity vs. selection criterion curve for a 'selection' object. When EBIC is used with voting, it also constructs a histogram showing the voting result.

Usage

```
## S3 method for class 'selection'
plot(x, ...)
```

Arguments

`x`	A 'selection' object as the output from smle_select().
`...`	Additional arguments to the plot() function.

Value

No return value.

Examples

```
set.seed(1)
Data <- Gen_Data(correlation = "MA", family = "gaussian")
fit <- SMLE(Y = Data$Y, X = Data$X, k = 20, family = "gaussian")
fit_s <- smle_select(fit, vote = TRUE)
plot(fit_s)
```

---

plot.smle *Plots to visualize SMLE screening*

Description

This function returns two plot windows. By default, the first shows 1) the solution path (estimated coefficient by iteration step) for the retained features. By default, the second plot contains 4 plots to assess convergence: 2) log-likelihood, 3) Euclidean distance between the current and the previous coefficient estimates, 4) the number of tries in u-search (see details of SMLE()), and 5) the number of features changed in the current active set.

Usage

```
## S3 method for class 'smle'
plot(x, num_path = NULL, label = TRUE, which_path = NULL, out_plot = 1, ...)
```

Arguments

`x`	A 'smle' object as the output from SMLE().
`num_path`	The number of top coefficients to be shown. Default is equal to the number of features retained in the model.
`label`	Logical flag for whether to label each curve with the feature index. Default is TRUE.
`which_path`	A vector to control which features are shown in addition to the paths for the most significant coefficients.

`out_plot`	A number from 1 to 5 indicating which plot is to be shown in the separate window; the default for solution path plot is "1". See Description for plot labels 2-5.
`...`	Additional arguments passed to the second plot.

Value

No return value.

Examples

```
set.seed(1)
Data <- Gen_Data(correlation = "CS")
fit <- SMLE(Y = Data$Y,X = Data$X, k = 20, family = "gaussian")
plot(fit)
```

`predict.smle` *Prediction based on SMLE screening and selection*

Description

For a model object of class `'smle'` or `'selection'`, this function returns the predicted response values after re-fitting the model using `glm`.

Usage

```
## S3 method for class 'smle'
predict(object, newdata = NULL, type = c("link", "response", "terms"), ...)

## S3 method for class 'selection'
predict(object, newdata = NULL, type = c("link", "response", "terms"), ...)
```

Arguments

`object`	A `'smle'` or `'selection'` object.
`newdata`	Matrix of new values for the features at which predictions are to be made. If omitted, the fitted linear predictors are used.
`type`	The type of prediction required by `predict.glm()`.
`...`	Further arguments passed to `predict.glm()`.

Value

A prediction vector with length equal to the number of rows of `newdata`.

Examples

```
set.seed(1)
Data_sim <- Gen_Data(n = 420, p = 1000, sigma = 0.5, family = "gaussian")
train_X <- Data_sim$X[1:400,]; test_X <- Data_sim$X[401:420,]
train_Y <- Data_sim$Y[1:400]; test_Y <- Data_sim$Y[401:420]
fit1 <- SMLE(Y = train_Y, X = train_X, family = "gaussian", k = 10)

#Fitted responses vs true responses in training data
predict(fit1)[1:10]
train_Y[1:10]

#Predicted responses vs true responses in testing data
predict(fit1, newdata = test_X)
test_Y
```

---

print.smle *Print an object*

Description

This function prints information about the fitted model from a call to SMLE() or smle_select(), or about the simulated data from a call to Gen_Data(). The object passed as an argument to print is returned invisibly.

Usage

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

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

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

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

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

Arguments

x	Fitted object.
...	This argument is not used and listed for method consistency.

Value

Return argument invisibly.

Examples

```
set.seed(1)
Data<-Gen_Data(correlation = "MA", family = "gaussian")
Data
fit<-SMLE(Y = Data$Y, X = Data$X, k = 20, family = "gaussian")
print(fit)
summary(fit)
```

---

pvals — *p values of synthetic genetic association study data set*

Description

The first column is the chromosome number. The second columns is SNP name. The third column is the genomic position of the SNP on the whole data set. The marginal p-values of each SNPs is pre-calculated and saved in the fourth column.

Usage

```
data(pvals)
```

Format

An object of class `data.frame` with 10031 rows and 4 columns.

SMLE — *Joint feature screening via sparse maximum likelihood estimation for GLMs*

Description

Input a n by 1 response Y and a n by p feature matrix X; the function uses SMLE to retain only a set of $k < n$ features that seem to be most related to the response variable. It thus serves as a pre-processing step for an elaborative analysis. In SMLE, the joint effects between features are naturally accounted for; this makes the screening more reliable. The function uses the efficient iterative hard thresholding (IHT) algorithm with step parameter adaptively tuned for fast convergence. Users can choose to further conduct an elaborative selection after SMLE-screening. See `smle_select()` for more details.

Usage

```
SMLE(formula = NULL, ...)

## Default S3 method:
SMLE(
  formula = NULL,
  X = NULL,
  Y = NULL,
  data = NULL,
  k = NULL,
  family = c("gaussian", "binomial", "poisson"),
  keyset = NULL,
  intercept = TRUE,
  categorical = TRUE,
  group = TRUE,
  codingtype = NULL,
  coef_initial = NULL,
  max_iter = 500,
  tol = 10^(-3),
  selection = F,
  standardize = TRUE,
  fast = FALSE,
  U = 1,
  U_rate = 0.5,
  penalize_mod = TRUE,
  ...
)

## S3 method for class 'formula'
SMLE(formula, data, k = NULL, keyset = NULL, categorical = NULL, ...)
```

Arguments

`formula` An object of class `'formula'` (or one that can be coerced to that class): a symbolic description of the model to be fitted. It should be `NULL` when `X` and `Y` are used.

`...` Additional arguments to be passed to `smle_select()` if `selection = TRUE`. See `smle_select()` documentation for more details.

`X` The n by p feature matrix X with each column denoting a feature (covariate) and each row denoting an observation vector. The input should be a `'matrix'` object for numerical data, and `'data.frame'` for categorical data (or a mixture of numerical and categorical data). The algorithm will treat covariates having class `'factor'` as categorical data and extend the data frame dimension by the dummy columns needed for coding the categorical features.

`Y` The response vector Y of dimension n by 1. Quantitative for `family = "gaussian"`, non-negative counts for `family = "poisson"`, binary (0-1) for `family = "binomial"`. Input Y should be `'numeric'`.

`data` An optional data frame, list or environment (or object coercible by `as.data.frame()` to a `'data.frame'`) containing the features in the model. It is required if `'formula'` is used.

`k` Total number of features (including `keyset`) to be retained after screening. Default is the largest integer not exceeding $0.5\log(n)n^{1/3}$.

`family` Model assumption between Y and X; either a character string representing one of the built-in families, or else a glm() family object. The default model is Gaussian linear.

`keyset` A numeric vector with column indices for the key features that do not participate in feature screening and are forced to remain in the model. The column indices for the key features should be from `data` if `'formula'` is used or in `X` if `X` and `Y` are provided. The class of `keyset` can be `'numeric'`,`'integer'` or `'character'`. Default is `NULL`.

`intercept` A logical flag to indicate whether to an intercept be used in the model. An intercept will not participate in screening.

`categorical` A logical flag for whether the input feature matrix includes categorical features(either `'factor'` or `'character'`). `FALSE` treats all features as numerical and not check for whether there are categorical features; `TRUE` treats the data as having some categorical features and the algorithm determines which columns contain the categorical features. If all features are known to be numerical, it will be faster to run SMLE with this argument set to `FALSE`. we will need to find which columns are the categorical features. Default is `TRUE`.

`group` Logical flag for whether to treat the dummy covariates of a categorical feature as a group. (Only for categorical data, see Details). Default is `TRUE`.

`codingtype` Coding types for categorical features; default is `"DV"`. `codingtype = "all"` convert each level to a 0-1 vector. `codingtype = "DV"` conducts deviation coding for each level in comparison with the grand mean. `codingtype = "standard"` conducts standard dummy coding for each level in comparison with the reference level (first level).

`coef_initial` A p-dimensional vector for the initial coefficient value of the IHT algorithm. The default is to use Lasso with the sparsity closest to $n-1$.

`max_iter` Maximum number of iteration steps. Default is 500.

`tol` A tolerance level to stop the iterations, when the squared sum of differences between two successive coefficient updates is below it. Default is 10^{-3}.

`selection` A logical flag to indicate whether an elaborate selection is to be conducted by `smle_select()` after screening. If `TRUE`, the function will return a `'selection'` object, see `smle_select()` documentation. Default is `FALSE`.

`standardize` A logical flag for feature standardization, prior to performing feature screening. The resulting coefficients are always returned on the original scale. If features are in the same units already, you might not wish to standardize. Default is `standardize = TRUE`.

`fast` Set to `TRUE` to enable early stop for SMLE-screening. It may help to boost the screening efficiency with a little sacrifice of accuracy. Default is `FALSE`, see Details.

U
: A numerical multiplier of initial tuning step parameter in IHT algorithm. Default is 1. For binomial model, a larger initial value is recommended; a smaller one is recommended for poisson model.

U_rate
: Decreasing rate in tuning step parameter $1/u$ in IHT algorithm. See Details.

penalize_mod
: A logical flag to indicate whether adjustment is used in ranking groups of features. This argument is applicable only when categorical = TRUE with group = TRUE. When penalize_mod = TRUE, a factor of $\sqrt{J}$ is divided from the L_2 effect of a group with J members. Default is TRUE.

Details

With the input Y and X, SMLE() conducts joint feature screening by running iterative hard thresholding algorithm (IHT), where the default initial value is set to be the Lasso estimate with the sparsity closest to the sample size minus one.

In SMLE(), the initial value for step size parameter $1/u$ is determined as follows. When coef_initial = 0, we set $1/u = U/\sqrt{p}$. When coef_initial != 0, we generate a sub-matrix X_0 using the columns of X corresponding to the non-zero positions of coef_initial and set $1/u = U/\sqrt{p}||X||^2_\infty$ and recursively decrease the value of step size by U_rate to guarantee the likelihood increment. This strategy is called u-search.

SMLE() terminates IHT iterations when either tol or max_iter is satisfied. When fast = TRUE, the algorithm also stops when the non-zero members of the coefficient estimates remain the same for 10 successive iterations or the log-likelihood difference between coefficient estimates is less than 0.01 times the log-likelihood increase of the first step, or tol$\sqrt{k}$ is satisfied.

In SMLE(), categorical features are coded by dummy covariates with the method specified in codingtype. Users can use group to specify whether to treat those dummy covariates as a single group feature or as individual features. When group = TRUE with penalize_mod = TRUE, the effect for a group of J dummy covariates is computed by

$$\beta_i = \sqrt{(\beta_1)^2 + ... + (\beta_J)^2}/\sqrt{J},$$

which will be treated as a single feature in IHT iterations. When group = FALSE, a group of J dummy covariates will be treated as J individual features in the IHT iterations; in this case, a categorical feature is retained after screening when at least one of the corresponding dummy covariates is retained.

Since feature screening is usually a preprocessing step, users may wish to further conduct an elaborative feature selection after screening. This can be done by setting selection = TRUE in SMLE() or applying any existing selection method on the output of SMLE().

Value

call
: The call that produced this object.

ID_retained
: A vector indicating the features retained after SMLE-screening. The output includes both features retained by SMLE() and the features specified in keyset.

coef_retained
: The vector of coefficients estimated by IHT for the retained features. When the retained set contains a categorical feature, the value returns a group effect if group = TRUE, or returns the strongest dummy covariate effect if group = FALSE.

`path_retained` IHT iteration path with columns recording the coefficient updates.

`num_retained` Number of retained features after screening.

`intercept` The estimated intercept value by IHT, if `intercept` = `TRUE`.

`steps` Number of IHT iterations.

`likelihood_iter`
A list of log-likelihood updates over the IHT iterations.

`Usearch` A vector giving the number of attempts to find a proper $1/u$ at each iteration step.

`modified_data` A list containing data objects generated by SMLE.

`CM`: Design matrix of class `'matrix'` for numeric features (or `'data.frame'` with categorical features).

`DM`: A matrix with dummy variable features added. (only if there are categorical features).

`dum_col`: Number of levels for all categorical features.

`CI`: Indices of categorical features in `CM`.

`DFI`: Indices of categorical features in `IM`.

`iteration_data` A list containing data objects that track the coefficients over iterations.

`IM`: Iteration path matrix with columns recording IHT coefficient updates.

`beta0`: Inital value of regression coefficient for IHT.

`feature_name`: A list contains the names of selected features.

`FD`: A matrix that contains feature indices retained at each iteration step.

`X`, `Y`, `data`, `family`, `categorical` and `codingtype` are return of arguments passed in the function call.

References

UCLA Statistical Consulting Group. *coding systems for categorical variables in regression analysis*. https://stats.oarc.ucla.edu/r/library/r-library-contrast-coding-systems-for-categorical-variabl Retrieved May 28, 2020.

Xu, C. and Chen, J. (2014). The Sparse MLE for Ultrahigh-Dimensional Feature Screening, *Journal of the American Statistical Association*, **109**(507), 1257-1269.

Examples

```
# Example 1:
set.seed(1)
Data <- Gen_Data( n= 200, p = 5000, family = "gaussian", correlation = "ID")
fit <- SMLE( Y = Data$Y , X = Data$X, k = 9,family = "gaussian")
summary(fit)
Data$subset_true %in% fit$ID_retained # Sure screening check.
plot(fit)

# Example 2:
set.seed(1)
Data_sim2 <- Gen_Data(n = 420, p = 1000, family = "gaussian", num_ctgidx = 5,
                      pos_ctgidx = c(1,3,5,7,9), effect_truecoef= c(1,2,3,-4,-5),
```

```
                      pos_truecoef = c(1,3,5,7,8), level_ctgidx = c(3,3,3,4,5))
train_X <- Data_sim2$X[1:400,]; test_X <- Data_sim2$X[401:420,]
train_Y <- Data_sim2$Y[1:400]; test_Y <- Data_sim2$Y[401:420]
fit <- SMLE(Y = train_Y, X = train_X, family = "gaussian", group = TRUE, k = 15)
predict(fit, newdata = test_X)
test_Y

# Example 3:
library(datasets)
data("attitude")
set.seed(1)
noise <- matrix(rnorm(30*100, mean = mean(attitude$rating) , sd = 1), ncol = 100)
colnames(noise) <- paste("Noise", seq(100), sep = ".")
df <- data.frame(cbind(attitude, noise))
fit <- SMLE(rating ~., data = df)
fit
```

`smle_select` *Elaborative post-screening selection with SMLE*

Description

The features retained after screening are still likely to contain some that are not related to the response. The function `smle_select()` is designed to further identify the relevant features using `SMLE()`. Given a response and a set of K features, this function first runs `SMLE(fast = TRUE)` to generate a series of sub-models with sparsity k varying from `k_min` to `k_max`. It then selects the best model from the series based on a selection criterion.

When criterion EBIC is used, users can choose to repeat the selection with different values of the tuning parameter γ, and conduct importance voting for each feature. When `vote = T`, this function fits all the models with γ specified in `gamma_seq` and features with frequency higher than `vote_threshold` will be selected in `ID_voted`.

Usage

```
smle_select(object, ...)

## S3 method for class 'sdata'
smle_select(
  object,
  k_min = 1,
  k_max = NULL,
  subset = NULL,
  gamma_ebic = 0.5,
  vote = FALSE,
  keyset = NULL,
```

```
    criterion = "ebic",
    codingtype = c("DV", "standard", "all"),
    gamma_seq = c(seq(0, 1, 0.2)),
    vote_threshold = 0.6,
    parallel = FALSE,
    num_clusters = NULL,
    ...
  )

  ## Default S3 method:
  smle_select(
    object = NULL,
    Y = NULL,
    X = NULL,
    family = "gaussian",
    keyset = NULL,
    ...
  )

  ## S3 method for class 'smle'
  smle_select(object, ...)
```

Arguments

`object`	Object of class `'smle'` or `'sdata'`. Users can also input a response vector and a feature matrix.
`...`	Further arguments passed to or from other methods.
`k_min`	The lower bound of candidate model sparsity. Default is 1.
`k_max`	The upper bound of candidate model sparsity. Default is the number of columns in feature matrix.
`subset`	An index vector indicating which features (columns of the feature matrix) are to be selected. Not applicable if a `'smle'` object is the input.
`gamma_ebic`	The EBIC tuning parameter, in $[0, 1]$. Default is 0.5.
`vote`	The logical flag for whether to perform the voting procedure. Only available when `criterion = "ebic"`.
`keyset`	A numeric vector with column indices for the key features that do not participate in feature screening and are forced to remain in the model. See SMLE for details.
`criterion`	Selection criterion. One of "`ebic`","`bic`","`aic`". Default is "`ebic`".
`codingtype`	Coding types for categorical features; for more details see `SMLE()` documentation.
`gamma_seq`	The sequence of values for `gamma_ebic` when `vote = TRUE`.
`vote_threshold`	A relative voting threshold in percentage. A feature is considered to be important when it receives votes passing the threshold. Default is 0.6.
`parallel`	A logical flag to use parallel computing to do voting selection. Default is `FALSE`. See Details.

num_clusters — The number of compute clusters to use when `parallel` = `TRUE`. The default will be 2 times cores detected.

Y — Input response vector (when `object` = `NULL`).

X — Input features matrix (when `object` = `NULL`).

family — Model assumption; see `SMLE()` documentation. Default is Gaussian linear.

When input is a `'smle'` or `'sdata'` object, the same model will be used in the selection.

Details

This function accepts three types of input objects; 1) `'smle'` object, as the output from `SMLE()`; 2) `'sdata'` object, as the output from `Gen_Data()`; 3) other response and feature matrix input by users.

Note that this function is mainly designed to conduct an elaborative selection after feature screening. We do not recommend using it directly for ultra-high-dimensional data without screening.

Value

call — The call that produced this object.

ID_selected — A list of selected features.

coef_selected — Fitted model coefficients.

intercept — Fitted model intercept.

criterion_value — Values of selection criterion for the candidate models with various sparsity.

categorical — A logical flag whether the input feature matrix includes categorical features

ID_pool — A vector containing all features selected during voting.

ID_voted — A vector containing the features selected when `vote` = `T`.

CI — Indices of categorical features when `categorical` = `TRUE`.

`X`, `Y`, `family`, `gamma_ebic`, `gamma_seq`, `criterion`, `vote`, `codyingtype`, `vote_threshold` are return of arguments passed in the function call.

References

Chen. J. and Chen. Z. (2012). "Extended BIC for small-n-large-p sparse GLM." *Statistica Sinica*, **22**(2), 555-574.

Examples

```
set.seed(1)
Data<-Gen_Data(correlation = "MA", family = "gaussian")
fit<-SMLE(Y = Data$Y, X = Data$X, k = 20, family = "gaussian")

fit_bic<-smle_select(fit, criterion = "bic")
summary(fit_bic)

fit_ebic<-smle_select(fit, criterion = "ebic", vote = TRUE)
```

```
summary(fit_ebic)
plot(fit_ebic)
```

---

summary.smle *Summarize SMLE-screening and selection*

Description

This function prints a summary of a 'smle' (or a 'selection') object. In particular, it shows the features retained after SMLE-screening (or selection) with the related convergence information.

Usage

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

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

Arguments

object	A 'smle' or 'selection' object.
...	This argument is not used and listed for method consistency.

Value

No return value.

Examples

```
set.seed(1)
Data <- Gen_Data(correlation = "MA", family = "gaussian")
fit <- SMLE(Y = Data$Y, X = Data$X, k = 20, family = "gaussian")
summary(fit)
fit_s <- smle_select(fit)
summary(fit_s)
```

synSNP *Synthetic genetic association study data set*

Description

This simulated data set consists of 10,031 genetic variants (SNPs) and a continuous response variable measured on 800 individuals. The genotypes were sampled from genotypic distributions derived from the 1000 Genomes project using the R package **sim1000G**. The genotype is coded as 0, 1, or 2 by counting the number of minor alleles (the allele that is less common in the sample). The continuous response variable was simulated from a normal distribution with mean that depends additively on the causal SNPs.

Usage

```
data(synSNP)
```

Format

An object of class 'data.frame' with 800 rows and 10,032 columns.

References

The 1000 Genomes Project Consortium (2015). Global reference for human genetic variation, *Nature*, **526**(7571), 68-74.s

Examples

```
data(synSNP)
Y_SNP <- synSNP[,1]
X_SNP <- synSNP[,-1]
fit <- SMLE(Y = Y_SNP, X = X_SNP, k = 40)
summary(fit)
plot(fit)
```

vote_update *Extract and adjust voting from SMLE selection*

Description

When smle_select() is used with criterion = "ebic" and vote = TRUE, users can use vote_update() to adjust the voting threshold without a need of rerun smle_select().

Usage

```
vote_update(object, ...)

## S3 method for class 'selection'
vote_update(object, vote_threshold = 0.6, ...)
```

Arguments

object	A 'selection' object as the output from smle_select().
...	This argument is not used and listed for method consistency.
vote_threshold	A voting threshold in percentage. A feature is considered to be important when it receives votes passing the threshold. Default is 0.6.

Value

The function returns a vector indicating the features selected by EBIC voting with the specified vote_threhold.

Examples

```
set.seed(1)
Data <- Gen_Data(n = 100, p = 3000, correlation = "MA", rho = 0.7, family = "gaussian")
colnames(Data$X)<- paste("X.",seq(3000) , sep = "")
fit <- SMLE(Y = Data$Y, X = Data$X, k = 20, family = "gaussian")
fit_s <- smle_select(fit, criterion = "ebic", vote = TRUE)
plot(fit_s)
fit_s
vote_update(fit_s, vote_threshold = 0.4)
```

Index

∗ **datasets**
    pvals, 11
    synSNP, 20

as.data.frame, *13*

coef.selection (coef.smle), 3
coef.smle, 3

Gen_Data, 4, *10*, *18*
glm, *4*, *7*, *9*

logLik, *7*
logLik.selection (logLik.smle), 7
logLik.smle, 7

plot, *8*
plot.selection, 7
plot.smle, 8
predict.glm, *9*
predict.selection (predict.smle), 9
predict.smle, 9
print.sdata (print.smle), 10
print.selection (print.smle), 10
print.smle, 10
print.summary.selection (print.smle), 10
print.summary.smle (print.smle), 10
pvals, 11

SMLE, *4*, *8*, *10*, 11, *14*, *16–18*
SMLE-package, 2
smle_select, *4*, *8*, *10–13*, *16*, 16, *20, 21*
summary.selection (summary.smle), 19
summary.smle, 19
synSNP, 20

vote_update, *20*, 20