Package ‘SIS’

January 27, 2020

Version 0.8-8

Date 2020-01-27

Title Sure Independence Screening

Depends R (>= 3.2.4)

Imports glmnet, ncvreg, survival

Description Variable selection techniques are essential tools for model selection and estimation in high-dimensional statistical models. Through this publicly available package, we provide a unified environment to carry out variable selection using iterative sure independence screening (SIS) (Fan and Lv (2008)<doi:10.1111/j.1467-9868.2008.00674.x>) and all of its variants in generalized linear models (Fan and Song (2009)<doi:10.1214/10-AOS798>) and the Cox proportional hazards model (Fan, Feng and Wu (2010)<doi:10.1214/10-IMSCOLL606>).

License GPL-2

RoxygenNote 7.0.2

NeedsCompilation no

Author Yang Feng [aut, cre],
Jianqing Fan [aut],
Diego Franco Saldana [aut],
Yichao Wu [aut],
Richard Samworth [aut]

Maintainer Yang Feng <yangfengstat@gmail.com>

Repository CRAN

Date/Publication 2020-01-27 17:10:07 UTC

R topics documented:

`leukemia.test` *Gene expression Leukemia testing data set from Golub et al. (1999)*

Description

Gene expression testing data of 7129 genes from 34 patients with acute leukemias (20 in class Acute Lymphoblastic Leukemia and 14 in class Acute Myeloid Leukemia) from the microarray study of Golub et al. (1999).

Usage

```
data(leukemia.test)
```

Format

A data frame with 34 observations on 7129 variables.

Source

http://wwwprod.broadinstitute.org/cgi-bin/cancer/datasets.cgi

References

Golub et al. (1999) Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. *Science*, **286**, 531-537.

`leukemia.train` *Gene expression Leukemia training data set from Golub et al. (1999)*

Description

Gene expression training data of 7129 genes from 38 patients with acute leukemias (27 in class Acute Lymphoblastic Leukemia and 11 in class Acute Myeloid Leukemia) from the microarray study of Golub et al. (1999).

Usage

```
data(leukemia.train)
```

Format

A data frame with 38 observations on 7129 variables.

Source

http://wwwprod.broadinstitute.org/cgi-bin/cancer/datasets.cgi

References

Golub et al. (1999) Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. *Science*, **286**, 531-537.

`predict.SIS` *Model prediction based on a fitted SIS object.*

Description

Similar to the usual predict methods, this function returns predictions from a fitted `'SIS'` object.

Usage

```
## S3 method for class 'SIS'
predict(
  object,
  newx,
  lambda = object$lambda,
  which = NULL,
  type = c("response", "link", "class"),
  ...
)
```

Arguments

`object`	Fitted `'SIS'` model object.
`newx`	Matrix of new values for `x` at which predictions are to be made, without the intercept term.
`lambda`	Penalty parameter `lambda` of the final fitted model by (I)SIS at which predictions are required. By default, only the lambda minimizing the criterion `tune` is returned.
`which`	Indices of the penalty parameter `lambda` of the final fitted model by (I)SIS at which predictions are required. If supplied, will overwrite the default `lambda` value.
`type`	Type of prediction required. Type `'response'` gives the fitted values for `'gaussian'`, fitted probabilities for `'binomial'`, fitted mean for `'poisson'`, and the fitted relative risk for `'cox'`. Type `'link'` returns the linear predictors for `'binomial'`, `'poisson'` and `'cox'` models; for `'gaussian'` models it is equivalent to type `'response'`. Type `'class'` applies only to `'binomial'` models, and produces the class label corresponding to the maximum probability (0-1 labels).
`...`	Not used. Other arguments to predict.

Value

The object returned depends on type.

Author(s)

Jianqing Fan, Yang Feng, Diego Franco Saldana, Richard Samworth, and Yichao Wu

References

Diego Franco Saldana and Yang Feng (2018) SIS: An R package for Sure Independence Screening in Ultrahigh Dimensional Statistical Models, *Journal of Statistical Software*, **83**, 2, 1-25.

Jianqing Fan and Jinchi Lv (2008) Sure Independence Screening for Ultrahigh Dimensional Feature Space (with discussion). *Journal of Royal Statistical Society B*, **70**, 849-911.

Jianqing Fan and Rui Song (2010) Sure Independence Screening in Generalized Linear Models with NP-Dimensionality. *The Annals of Statistics*, **38**, 3567-3604.

Jianqing Fan, Richard Samworth, and Yichao Wu (2009) Ultrahigh Dimensional Feature Selection: Beyond the Linear Model. *Journal of Machine Learning Research*, **10**, 2013-2038.

Jianqing Fan, Yang Feng, and Yichao Wu (2010) High-dimensional Variable Selection for Cox Proportional Hazards Model. *IMS Collections*, **6**, 70-86.

Jianqing Fan, Yang Feng, and Rui Song (2011) Nonparametric Independence Screening in Sparse Ultrahigh Dimensional Additive Models. *Journal of the American Statistical Association*, **106**, 544-557.

Diego Franco Saldana and Yang Feng (2014) SIS: An R package for Sure Independence Screening in Ultrahigh Dimensional Statistical Models, *Journal of Statistical Software*.

See Also

SIS

Examples

```
set.seed(0)
n = 400; p = 50; rho = 0.5
corrmat = diag(rep(1-rho, p)) + matrix(rho, p, p)
corrmat[,4] = sqrt(rho)
corrmat[4, ] = sqrt(rho)
corrmat[4,4] = 1
corrmat[,5] = 0
corrmat[5, ] = 0
corrmat[5,5] = 1
cholmat = chol(corrmat)
x = matrix(rnorm(n*p, mean=0, sd=1), n, p)
x = x%*%cholmat
testX = matrix(rnorm(10*p, mean=0, sd=1), nrow=10, ncol=p)

# gaussian response
```

```
set.seed(1)
b = c(4,4,4,-6*sqrt(2),4/3)
y=x[, 1:5]%*%b + rnorm(n)
model1=SIS(x, y, family='gaussian', tune='bic', varISIS='aggr', seed=11)

predict(model1, testX, type='response')
predict(model1, testX, which=1:10, type='response')

## Not run:
# binary response
set.seed(2)
feta = x[, 1:5]%*%b; fprob = exp(feta)/(1+exp(feta))
y = rbinom(n, 1, fprob)
model2=SIS(x, y, family='binomial', tune='bic', varISIS='aggr', seed=21)

predict(model2, testX, type='response')
predict(model2, testX, type='link')
predict(model2, testX, type='class')

predict(model2, testX, which=1:10, type='response')
predict(model2, testX, which=1:10, type='link')
predict(model2, testX, which=1:10, type='class')

# poisson response
set.seed(3)
b = c(0.6,0.6,0.6,-0.9*sqrt(2))
myrates = exp(x[, 1:4]%*%b)
y = rpois(n, myrates)
model3=SIS(x, y, family='poisson', penalty = 'lasso',tune='bic', varISIS='aggr', seed=31)

predict(model3, testX, type='response')
predict(model3, testX, type='link')

## End(Not run)
```

`prostate.test` *Gene expression Prostate Cancer testing data set from Singh et al. (2002)*

Description

Gene expression testing data of 12600 genes from 25 patients with prostate tumors and 9 normal specimens from the microarray study of Singh et al. (2002).

Usage

```
data(prostate.test)
```

Format

A data frame with 34 observations on 12600 variables.

Source

http://wwwprod.broadinstitute.org/cgi-bin/cancer/datasets.cgi

References

Singh et al. (2002) Gene Expression Correlates of Clinical Prostate Cancer Behavior. *Cancer Cell*, **1**, 203-209.

`prostate.train` *Gene expression Prostate Cancer training data set from Singh et al. (2002)*

Description

Gene expression training data of 12600 genes from 52 patients with prostate tumors and 50 normal specimens from the microarray study of Singh et al. (2002).

Usage

```
data(prostate.train)
```

Format

A data frame with 102 observations on 12600 variables.

Source

http://wwwprod.broadinstitute.org/cgi-bin/cancer/datasets.cgi

References

Singh et al. (2002) Gene Expression Correlates of Clinical Prostate Cancer Behavior. *Cancer Cell*, **1**, 203-209.

SIS *(Iterative) Sure Independence Screening ((I)SIS) and Fitting in Generalized Linear Models and Cox's Proportional Hazards Models*

Description

This function first implements the Iterative Sure Independence Screening for different variants of (I)SIS, and then fits the final regression model using the R packages **ncvreg** and **glmnet** for the SCAD/MCP/LASSO regularized loglikelihood for the variables picked by (I)SIS.

Usage

```
SIS(
  x,
  y,
  family = c("gaussian", "binomial", "poisson", "cox"),
  penalty = c("SCAD", "MCP", "lasso"),
  concavity.parameter = switch(penalty, SCAD = 3.7, 3),
  tune = c("bic", "ebic", "aic", "cv"),
  nfolds = 10,
  type.measure = c("deviance", "class", "auc", "mse", "mae"),
  gamma.ebic = 1,
  nsis = NULL,
  iter = TRUE,
  iter.max = ifelse(greedy == FALSE, 10, floor(nrow(x)/log(nrow(x)))),
  varISIS = c("vanilla", "aggr", "cons"),
  perm = FALSE,
  q = 1,
  greedy = FALSE,
  greedy.size = 1,
  seed = NULL,
  standardize = TRUE
)
```

Arguments

`x`	The design matrix, of dimensions n * p, without an intercept. Each row is an observation vector. `SIS` standardizes the data and includes an intercept by default.
`y`	The response vector of dimension n * 1. Quantitative for `family='gaussian'`, non-negative counts for `family='poisson'`, binary (0-1) for `family='binomial'`. For `family='cox'`, `y` should be an object of class `Surv`, as provided by the function `Surv()` in the package **survival**.
`family`	Response type (see above).
`penalty`	The penalty to be applied in the regularized likelihood subproblems. 'SCAD' (the default), 'MCP', or 'lasso' are provided.

concavity.parameter
: The tuning parameter used to adjust the concavity of the SCAD/MCP penalty. Default is 3.7 for SCAD and 3 for MCP.

tune
: Method for tuning the regularization parameter of the penalized likelihood subproblems and of the final model selected by (I)SIS. Options include `tune='bic'`, `tune='ebic'`, `tune='aic'`, and `tune='cv'`.

nfolds
: Number of folds used in cross-validation. The default is 10.

type.measure
: Loss to use for cross-validation. Currently five options, not all available for all models. The default is `type.measure='deviance'`, which uses squared-error for gaussian models (also equivalent to `type.measure='mse'` in this case), deviance for logistic and poisson regression, and partial-likelihood for the Cox model. Both `type.measure='class'` and `type.measure='auc'` apply only to logistic regression and give misclassification error and area under the ROC curve, respectively. `type.measure='mse'` or `type.measure='mae'` (mean absolute error) can be used by all models except the `'cox'`; they measure the deviation from the fitted mean to the response. For `penalty='SCAD'` and `penalty='MCP'`, only `type.measure='deviance'` is available.

gamma.ebic
: Specifies the parameter in the Extended BIC criterion penalizing the size of the corresponding model space. The default is `gamma.ebic=1`. See references at the end for details.

nsis
: Number of pedictors recuited by (I)SIS.

iter
: Specifies whether to perform iterative SIS. The default is `iter=TRUE`.

iter.max
: Maximum number of iterations for (I)SIS and its variants.

varISIS
: Specifies whether to perform any of the two ISIS variants based on randomly splitting the sample into two groups. The variant `varISIS='aggr'` is an aggressive variable screening procedure, while `varISIS='cons'` is a more conservative approach. The default is `varISIS='vanilla'`, which performs the traditional vanilla version of ISIS. See references at the end for details.

perm
: Specifies whether to impose a data-driven threshold in the size of the active sets calculated during the ISIS procedures. The threshold is calculated by first decoupling the predictors x_i and response y_i through a random permutation π of $(1, ..., n)$ to form a null model. For this newly permuted data, marginal regression coefficients for each predictor are recalculated. As the marginal regression coeffcients of the original data should be larger than most recalculated coefficients in the null model, the data-driven threshold is given by the qth quantile of the null coefficients. This data-driven threshold only allows a $1 - q$ proportion of inactive variables to enter the model when x_i and y_i are not related (in the null model). The default is here is `perm=FALSE`. See references at the end for details.

q
: Quantile for calculating the data-driven threshold in the permutation-based ISIS. The default is `q=1` (i.e., the maximum absolute value of the permuted estimates).

greedy
: Specifies whether to run the greedy modification of the permutation-based ISIS. The default is `greedy=FALSE`.

greedy.size
: Maximum size of the active sets in the greedy modification of the permutation-based ISIS. The default is `greedy.size=1`.

seed	Random seed used for sample splitting, random permutation, and cross-validation sampling of training and test sets.
standardize	Logical flag for x variable standardization, prior to performing (iterative) variable screening. The resulting coefficients are always returned on the original scale. Default is standardize=TRUE. If variables are in the same units already, you might not wish to standardize.

Value

Returns an object with

sis.ix0	The vector of indices selected by only SIS.
ix	The vector of indices selected by (I)SIS with regularization step.
coef.est	The vector of coefficients of the final model selected by (I)SIS.
fit	A fitted object of type ncvreg, cv.ncvreg, glmnet, or cv.glmnet for the final model selected by the (I)SIS procedure. If tune='cv', the returned fitted object is of type cv.ncvreg if penalty='SCAD' or penalty='MCP'; otherwise, the returned fitted object is of type cv.glmnet. For the remaining options of tune, the returned object is of type glmnet if penalty='lasso', and ncvreg otherwise.
path.index	The index along the solution path of fit for which the criterion specified in tune is minimized.

Author(s)

Jianqing Fan, Yang Feng, Diego Franco Saldana, Richard Samworth, and Yichao Wu

References

Diego Franco Saldana and Yang Feng (2018) SIS: An R package for Sure Independence Screening in Ultrahigh Dimensional Statistical Models, *Journal of Statistical Software*, **83**, 2, 1-25.

Jianqing Fan and Jinchi Lv (2008) Sure Independence Screening for Ultrahigh Dimensional Feature Space (with discussion). *Journal of Royal Statistical Society B*, **70**, 849-911.

Jianqing Fan and Rui Song (2010) Sure Independence Screening in Generalized Linear Models with NP-Dimensionality. *The Annals of Statistics*, **38**, 3567-3604.

Jianqing Fan, Richard Samworth, and Yichao Wu (2009) Ultrahigh Dimensional Feature Selection: Beyond the Linear Model. *Journal of Machine Learning Research*, **10**, 2013-2038.

Jianqing Fan, Yang Feng, and Yichao Wu (2010) High-dimensional Variable Selection for Cox Proportional Hazards Model. *IMS Collections*, **6**, 70-86.

Jianqing Fan, Yang Feng, and Rui Song (2011) Nonparametric Independence Screening in Sparse Ultrahigh Dimensional Additive Models. *Journal of the American Statistical Association*, **106**, 544-557.

Jiahua Chen and Zehua Chen (2008) Extended Bayesian Information Criteria for Model Selection with Large Model Spaces. *Biometrika*, **95**, 759-771.

See Also

predict.SIS

Examples

```
set.seed(0)
n = 400; p = 50; rho = 0.5
corrmat = diag(rep(1-rho, p)) + matrix(rho, p, p)
corrmat[,4] = sqrt(rho)
corrmat[4, ] = sqrt(rho)
corrmat[4,4] = 1
corrmat[,5] = 0
corrmat[5, ] = 0
corrmat[5,5] = 1
cholmat = chol(corrmat)
x = matrix(rnorm(n*p, mean=0, sd=1), n, p)
x = x%*%cholmat

# gaussian response
set.seed(1)
b = c(4,4,4,-6*sqrt(2),4/3)
y=x[, 1:5]%*%b + rnorm(n)
# SIS without regularization
model10 = SIS(x, y, family='gaussian', iter = FALSE)
model10$sis.ix0
# ISIS with regularization
model11=SIS(x, y, family='gaussian', tune='bic')
model12=SIS(x, y, family='gaussian', tune='bic', varISIS='aggr', seed=11)
model11$ix
model12$ix

## Not run:
# binary response
set.seed(2)
feta = x[, 1:5]%*%b; fprob = exp(feta)/(1+exp(feta))
y = rbinom(n, 1, fprob)
model21=SIS(x, y, family='binomial', tune='bic')
model22=SIS(x, y, family='binomial', tune='bic', varISIS='aggr', seed=21)
model21$ix
model22$ix

# poisson response
set.seed(3)
b = c(0.6,0.6,0.6,-0.9*sqrt(2))
myrates = exp(x[, 1:4]%*%b)
y = rpois(n, myrates)
model31=SIS(x, y, family='poisson', penalty = 'lasso', tune='bic', perm=TRUE, q=0.9,
            greedy=TRUE, seed=31)
model32=SIS(x, y, family='poisson', penalty = 'lasso',  tune='bic', varISIS='aggr',
            perm=TRUE, q=0.9, seed=32)
model31$ix
model32$ix

# Cox model
```

```
set.seed(4)
b = c(4,4,4,-6*sqrt(2),4/3)
myrates = exp(x[, 1:5]%*%b)
Sur = rexp(n,myrates); CT = rexp(n,0.1)
Z = pmin(Sur,CT); ind = as.numeric(Sur<=CT)
y = survival::Surv(Z,ind)
model41=SIS(x, y, family='cox', penalty='lasso', tune='bic',
           varISIS='aggr', seed=41)
model42=SIS(x, y, family='cox', penalty='lasso', tune='bic',
             varISIS='cons', seed=41)
model41$ix
model42$ix

## End(Not run)
```

---

standardize *Standardization of High-Dimensional Design Matrices*

Description

Standardizes the columns of a high-dimensional design matrix to mean zero and unit Euclidean norm.

Usage

```
standardize(X)
```

Arguments

X	A design matrix to be standardized.

Details

Performs a location and scale transform to the columns of the original design matrix, so that the resulting design matrix with p-dimensional observations $\{x_i : i = 1, ..., n\}$ of the form $x_i = (x_{i1}, x_{i2}, ..., x_{ip})$ satisfies $\sum_{i=1}^n x_{ij} = 0$ and $\sum_{i=1}^n x_{ij}^2 = 1$ for $j = 1, ..., p$.

Value

A design matrix with standardized predictors or columns.

Author(s)

Jianqing Fan, Yang Feng, Diego Franco Saldana, Richard Samworth, and Yichao Wu

References

Diego Franco Saldana and Yang Feng (2018) SIS: An R package for Sure Independence Screening in Ultrahigh Dimensional Statistical Models, *Journal of Statistical Software*, **83**, 2, 1-25.

Examples

```
set.seed(0)
n = 400; p = 50; rho = 0.5
corrmat = diag(rep(1-rho, p)) + matrix(rho, p, p)
corrmat[,4] = sqrt(rho)
corrmat[4, ] = sqrt(rho)
corrmat[4,4] = 1
corrmat[,5] = 0
corrmat[5, ] = 0
corrmat[5,5] = 1
cholmat = chol(corrmat)
x = matrix(rnorm(n*p, mean=15, sd=9), n, p)
x = x%*%cholmat

x.standard = standardize(x)
```

`tune.fit` *Using the* **glmnet** *and* **ncvreg** *packages, fits a Generalized Linear Model or Cox Proportional Hazards Model using various methods for choosing the regularization parameter* λ

Description

This function fits a generalized linear model or a Cox proportional hazards model via penalized maximum likelihood, with available penalties as indicated in the **glmnet** and **ncvreg** packages. Instead of providing the whole regularization solution path, the function returns the solution at a unique value of λ, the one optimizing the criterion specified in `tune`.

Usage

```
tune.fit(
  x,
  y,
  family = c("gaussian", "binomial", "poisson", "cox"),
  penalty = c("SCAD", "MCP", "lasso"),
  concavity.parameter = switch(penalty, SCAD = 3.7, 3),
  tune = c("cv", "aic", "bic", "ebic"),
  nfolds = 10,
  type.measure = c("deviance", "class", "auc", "mse", "mae"),
  gamma.ebic = 1
)
```

Arguments

`x` The design matrix, of dimensions n * p, without an intercept. Each row is an observation vector.

`y`	The response vector of dimension n * 1. Quantitative for `family='gaussian'`, non-negative counts for `family='poisson'`, binary (0-1) for `family='binomial'`. For `family='cox'`, `y` should be an object of class `Surv`, as provided by the function `Surv()` in the package **survival**.
`family`	Response type (see above).
`penalty`	The penalty to be applied in the regularized likelihood subproblems. 'SCAD' (the default), 'MCP', or 'lasso' are provided.
`concavity.parameter`	The tuning parameter used to adjust the concavity of the SCAD/MCP penalty. Default is 3.7 for SCAD and 3 for MCP.
`tune`	Method for selecting the regularization parameter along the solution path of the penalized likelihood problem. Options to provide a final model include `tune='cv'`, `tune='aic'`, `tune='bic'`, and `tune='ebic'`. See references at the end for details.
`nfolds`	Number of folds used in cross-validation. The default is 10.
`type.measure`	Loss to use for cross-validation. Currently five options, not all available for all models. The default is `type.measure='deviance'`, which uses squared-error for gaussian models (also equivalent to `type.measure='mse'` in this case), deviance for logistic and poisson regression, and partial-likelihood for the Cox model. Both `type.measure='class'` and `type.measure='auc'` apply only to logistic regression and give misclassification error and area under the ROC curve, respectively. `type.measure='mse'` or `type.measure='mae'` (mean absolute error) can be used by all models except the `'cox'`; they measure the deviation from the fitted mean to the response. For `penalty='SCAD'` and `penalty='MCP'`, only `type.measure='deviance'` is available.
`gamma.ebic`	Specifies the parameter in the Extended BIC criterion penalizing the size of the corresponding model space. The default is `gamma.ebic=1`. See references at the end for details.

Value

Returns an object with

`ix`	The vector of indices of the nonzero coefficients selected by the maximum penalized likelihood procedure with `tune` as the method for choosing the regularization parameter.
`a0`	The intercept of the final model selected by `tune`.
`beta`	The vector of coefficients of the final model selected by `tune`.
`fit`	The fitted penalized regression object.
`lambda`	The corresponding lambda in the final model.
`lambda.ind`	The index on the solution path for the final model.

Author(s)

Jianqing Fan, Yang Feng, Diego Franco Saldana, Richard Samworth, and Yichao Wu

References

Jerome Friedman and Trevor Hastie and Rob Tibshirani (2010) Regularization Paths for Generalized Linear Models Via Coordinate Descent. *Journal of Statistical Software*, **33**(1), 1-22.

Noah Simon and Jerome Friedman and Trevor Hastie and Rob Tibshirani (2011) Regularization Paths for Cox's Proportional Hazards Model Via Coordinate Descent. *Journal of Statistical Software*, **39**(5), 1-13.

Patrick Breheny and Jian Huang (2011) Coordiante Descent Algorithms for Nonconvex Penalized Regression, with Applications to Biological Feature Selection. *The Annals of Applied Statistics*, **5**, 232-253.

Hirotogu Akaike (1973) Information Theory and an Extension of the Maximum Likelihood Principle. In *Proceedings of the 2nd International Symposium on Information Theory*, BN Petrov and F Csaki (eds.), 267-281.

Gideon Schwarz (1978) Estimating the Dimension of a Model. *The Annals of Statistics*, **6**, 461-464.

Jiahua Chen and Zehua Chen (2008) Extended Bayesian Information Criteria for Model Selection with Large Model Spaces. *Biometrika*, **95**, 759-771.

Examples

```
set.seed(0)
data('leukemia.train', package = 'SIS')
y.train = leukemia.train[,dim(leukemia.train)[2]]
x.train = as.matrix(leukemia.train[,-dim(leukemia.train)[2]])
x.train = standardize(x.train)
model = tune.fit(x.train[,1:3500], y.train, family='binomial', tune='bic')
model$ix
model$a0
model$beta
```

Index