Package ‘rbvs’

August 29, 2016

Type Package

Title Ranking-Based Variable Selection

Version 1.0.2

Date 2015-12-08

Author Rafal Baranowski, Patrick Breheny, Isaac Turner

Maintainer Rafal Baranowski <r.baranowski@lse.ac.uk>

Depends stats

Description Implements the Ranking-Based Variable Selection
algorithm for variable selection in high-dimensional data.

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-12-11 19:53:09

R topics documented:

`rbvs-package` *Ranking-Based Variable Selection*

Description

The package implements the Ranking-Based Variable Selection algorithm proposed in Baranowski and Fryzlewicz (2015) for variable selection in high-dimensional data.

Details

The main routine of the package is `rbvs`.

References

R. Baranowski, P. Fryzlewicz (2015), Ranking-Based Variable Selection, in submission (http://personal.lse.ac.uk/baranows/rbvs/rbvs.pdf).

`distance.cor` *Measure an impact of the covariates on the response using the distance correlation This function evaluates the distance correlation between the response* `y` *and each column in the design matrix* `x` *over subsamples in* `subsamples`.

Description

Measure an impact of the covariates on the response using the distance correlation This function evaluates the distance correlation between the response `y` and each column in the design matrix `x` over subsamples in `subsamples`.

Usage

```
distance.cor(x, y, subsamples, index = 1, ...)
```

Arguments

`x`	Matrix with `n` observations of `p` covariates in each row.
`y`	Response vector with `n` observations.
`subsamples`	Matrix with `m` indices of `N` subsamples in each column.
`index`	Positive scalar.
`...`	Not in use.

Value

Numeric `p` by `N` matrix with distance correlations evaluated for each subsample.

References

Maria L. Rizzo and Gabor J. Szekely (2014). energy: E-statistics (energy statistics). R package version 1.6.1 (`http://CRAN.R-project.org/package=energy`).

`factor.model.design` *Generate factor model design matrix.*

Description

This function enables a quick generation of random design matrices (see details).

Usage

```
factor.model.design(n, p, n.factors, sigma = 1)
```

Arguments

`n`	Number of independent realisations of the factor model.
`p`	Number of covariates.
`n.factors`	Number of factors.
`sigma`	Standard deviation for the normal distribution (see details).

Details

The elements of the matrix returned by this routine satisfy $X_{ij} = \sum_{l=1}^{n.factors} f_{ijl}\varphi_{il} + \theta_{ij}$ with $f_{ijl}, \varphi_{il}, \theta_{ij}, \varepsilon_i$ i.i.d. $\mathcal{N}(0, (sigma)^2)$.

Value

`n` by `p` matrix with independent rows following factor model (see details).

`lasso.coef` *Measure an impact of the covariates on the response using Lasso This function evaluates the Lasso coefficients regressing* `y` *onto the design matrix* `x` *over subsamples in* `subsamples`.

Description

Measure an impact of the covariates on the response using Lasso This function evaluates the Lasso coefficients regressing `y` onto the design matrix `x` over subsamples in `subsamples`.

Usage

```
lasso.coef(x, y, subsamples, nonzero = NULL, family = c("gaussian",
  "binomial"), alpha = 1, maxit = 500, tol = 0.01, lambda.ratio = 1e-06,
  nlam = 25, ...)
```

Arguments

`x`	Matrix with `n` observations of `p` covariates in each row.
`y`	Response vector with `n` observations.
`subsamples`	Matrix with `m` indices of `N` subsamples in each column.
`nonzero`	Number of non-zero coefficients estimated for each subsample.
`family`	Determines the likelihood optimised in the estimation procedure.
`alpha`	Scalar between 0 and 1 determining l2 penalty (see details).
`maxit`	Maximum number of itarations when computing the lasso coefficients.
`tol`	Scalar determining convergence of the lasso algorithm used.
`lambda.ratio`	Scalar being a fraction of 1. Used in the lasso algorithm
`nlam`	Number of penalty parameters used in the lasso algorithm.
`...`	Not in use.

Details

To solve the Lasso problem, we implement the coordinate descent algorithm as in Breheny Jian (2011).

Author(s)

Rafal Baranowski, Patrick Breheny

References

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

Breheny, Patrick, and Jian Huang. "Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection." The Annals of Applied Statistics 5.1 (2011): 232.

`mcplus.coef` *Measure an impact of the covariates on the response using MC+. This function evaluates the MC+ coefficients regressing* `y` *onto the design matrix* `x` *over subsamples in* `subsamples`.

Description

Measure an impact of the covariates on the response using MC+. This function evaluates the MC+ coefficients regressing `y` onto the design matrix `x` over subsamples in `subsamples`.

Usage

```
mcplus.coef(x, y, subsamples, nonzero = NULL, family = c("gaussian",
  "binomial"), alpha = 1, gamma = 3, maxit = 500, tol = 0.01,
  lambda.ratio = 1e-06, nlam = 25, ...)
```

Arguments

`x`	Matrix with `n` observations of `p` covariates in each row.
`y`	Response vector with `n` observations.
`subsamples`	Matrix with `m` indices of `N` subsamples in each column.
`nonzero`	Number of non-zero coefficients estimated for each subsample.
`family`	Determines the likelihood optimised in the estimation procedure.
`alpha`	Scalar between 0 and 1 determining l2 penalty (see details).
`gamma`	Scalar greater than 1. The concacivity parameter (see details).
`maxit`	Maximum number of itarations when computing the MC+ coefficients.
`tol`	Scalar determining convergence of the MC+ algorithm used.
`lambda.ratio`	Scalar being a fraction of 1. Used in the MC+ algorithm
`nlam`	Number of penalty parameters used in the MC+ algorithm.
`...`	Not in use.

Details

To solve the MC+ problem, we implement the coordinate descent algorithm as in Breheny Jian (2011).

Author(s)

Rafal Baranowski, Patrick Breheny

References

Zhang, Cun-Hui. "Nearly unbiased variable selection under minimax concave penalty." The Annals of Statistics (2010): 894-942.

Breheny, Patrick, and Jian Huang. "Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection." The Annals of Applied Statistics 5.1 (2011): 232.

`pearson.cor` *Measure an impact of the covariates on the response using Pearson correlatio. This function evaluates the Pearson correlation coefficient between the response `y` and each column in the design matrix `x` over subsamples in `subsamples`.*

Description

Measure an impact of the covariates on the response using Pearson correlatio. This function evaluates the Pearson correlation coefficient between the response `y` and each column in the design matrix `x` over subsamples in `subsamples`.

Usage

```
pearson.cor(x, y, subsamples, ...)
```

Arguments

`x`	Matrix with `n` observations of `p` covariates in each row.
`y`	Response vector with `n` observations.
`subsamples`	Matrix with `m` indices of `N` subsamples in each column.
`...`	Not in use.

Value

Numeric `p` by `N` matrix with Pearson correlations evaluated for each subsample.

`rankings` *Evaluate rankings*

Description

Returns the non-increasing order of the values in the columns of `x`. Ties are solved at random.

Usage

```
rankings(x, k.max)
```

Arguments

`x`	Numeric matrix.
`k.max`	Integer. Indices of k.max largest elements are returned.

Value

Matrix with the indices corresponding to the `k.max` largest values in `x`.

Examples

```
omega <- abs(matrix(rnorm(100*5), nrow = 10, ncol = 5))
rankings(omega, k.max = 10)
```

rbvs *Ranking-Based Variable Selection*

Description

Performs Rankings-Based Variable Selection using various measures of the dependence between the predictors and the response.

Usage

```
rbvs(x, y, ...)

## Default S3 method:
rbvs(x, y, m, B = 500, measure = c("pc", "dc", "lasso",
  "mcplus", "user"), fun = NULL, s.est = s.est.quotient, iterative = TRUE,
  use.residuals = TRUE, k.max, min.max.freq = 0, max.iter = 10,
  verbose = TRUE, ...)
```

Arguments

`x`	Matrix with `n` observations of `p` covariates in each row.
`y`	Response vector with `n` observations.
`...`	Other parameters that may be passed to `fun` and`s.est`.
`m`	Subsample size used in the RBVS algorithm.
`B`	Number of sample splits.
`measure`	Character with the name of the method used to measure the association between the response and the covariates. See Details below.
`fun`	Function used to evaluate the measure given in `measure`. It is required when method=="user". Must have at least three arguments: `x` (covariates matrix), `.y` (response vector), `subsamples` (a matrix, each row contains indices of the observations to be used); return a vector of the same length as the number of covariates in `.x`. See for example `pearson.cor` or `lasso.coef`.
`s.est`	Function used to estimate the number of important covariates based on the RBVS path. Must accept `probs` (a vector with probabilities) as an argument. See `s.est.quotient` and Details below.
`iterative`	Logical variable indicating the type of the procedure. If `TRUE`, an iterative extension of the RBVS algorithm is launched.
`use.residuals`	Logical. If true, the impact of the previously detected variables is removed from the response in the IRBVS procedure.
`k.max`	Maximum size of the subset of important variables..
`min.max.freq`	Positive integer. Optional parameter - the algorithm stops searching for the most frequent set when the frequencies reach this value.
`max.iter`	Maximum number of iterations fot the IRBVS algorithm.
`verbose`	Logical indicating wheter the progress of the algorithm should be reported.

Details

Currently supported measures are: Pearson correlation coefficient (`measure="pc"`), Distance Correlation (`measure="dc"`), the regression coefficients estimated via Lasso (`measure="lasso"`), the regression coefficients estimated via MC+ (`measure="mcplus"`).

Value

Object of class rbvs with the following fields

`measure`	Character indicating type of measure used.
`score`	List with scores at each iteration.
`subsets`	A list with subset candidates at each iteration.
`frequencies`	A list with observed frequencies at each iteration.
`ranks`	Rankings evaluated (for the last iteration `iterative=TRUE`)
`s.hat`	Vector with the number of the covariates selected at each iteration.
`active`	Vector with the selected covariates.
`timings`	Vector reporting the amount of time the (I)RBVS algorithm took at each iteration.

References

R. Baranowski, P. Fryzlewicz (2015), Ranking-Based Variable Selection, in submission (http://personal.lse.ac.uk/baranows/rbvs/rbvs.pdf).

Examples

```
set.seed(1)

x <- matrix(rnorm(200*1000),200,1000)
active <- 1:4
beta <- c(3,2.5,-1.7,-1)
y <- 1*rnorm(200) +x[,active]%*%beta
#RBVS algorithm
rbvs.object <- rbvs(x,y, iterative=FALSE)
rbvs.object$active
rbvs.object$subsets[[1]][[4]]
#IRBVS algorithm
rbvs.object <- rbvs(x,y)
rbvs.object$active
```

`s.est.quotient` *Estimate the size of the top-ranked set*

Description

Estimates the number of elements in the top-ranked set.

Usage

```
s.est.quotient(prob)
```

Arguments

`prob`	Vector with probabilities.

Details

See Baranowski and Fryzlewicz (2015).

Value

A list with the following fields:

`scores`	Vector with the values of the criterion.
`s.hat`	The estimate of the number of important covariates.

References

R. Baranowski, P. Fryzlewicz (2015), Ranking Based Variable Selection, in submission (http://personal.lse.ac.uk/baranows/rbvs/rbvs.pdf).

`standardise` *Standardise data*

Description

Standardises the columns of a numeric matrix `x` (similar to R-function `scale`). If `x` is a vector, it is treated as a 1-column matrix.

Usage

```
standardise(x, scale = TRUE)
```

Arguments

- `x` A numeric matrix (or vector).
- `scale` A logical; if `TRUE` each column of `x` is divided by the square root of the sum of its centred squares.

Details

This function is much faster than `scale`.

Value

Matrix with centred (and optionally scaled) columns.

Examples

```
x <- matrix(rnorm(100*10), nrow = 100, ncol = 10)
x <- standardise(x)
standard.deviations <- apply(x,2,sd)
print(standard.deviations)
```

`subsample` *Generates subsamples.*

Description

Generates subsamples.

Usage

```
subsample(n, m, B)
```

Arguments

- `n` The sample size.
- `m` Subsample size (an integer lower or equal than `n`).
- `B` Number of sample splits.

Details

Generates `m`-element subsamples drawn $\lfloor \frac{n}{m} \rfloor$ times from `1,...,n` independently without replacement; such subsampling is repeated `B` times.

Value

Matrix with the indices of the subsamples drawn in each column.

References

R. Baranowski, P. Fryzlewicz (2015), Ranking Based Variable Selection, in submission (http://personal.lse.ac.uk/baranows/rbvs/rbvs.pdf).

Examples

```
subsample(10,5,2)
subsample(10,3,10)
```

---

top.ranked.sets *Find k-top-ranked sets*

Description

Finds k-top-ranked sets defined in Baranowski and Fryzlewicz (2015). This routine is used inside `rbvs`; it typically will be not called directly by the user.

Usage

```
top.ranked.sets(rankings, k.max, min.max.freq = 1, active = NULL)
```

Arguments

`rankings`	Matrix with rankings in each column.
`k.max`	Positive integer.
`min.max.freq`	Maximum frequency.
`active`	A vector with previously found active variables.

Details

Uses Portable qsort_r / qsort_s library (Turner (2013)).

Value

List containing the following fields.

`frequencies`	Frequencies corresponding to the most frequent subsests at the top of the rankings.
`subsets`	The moost frequent subsets.

References

R. Baranowski, P. Fryzlewicz (2015), Ranking-Based Variable Selection, in submission (http://personal.lse.ac.uk/baranows/rbvs/rbvs.pdf).

I. Turner (2013), Portable qsort_r / qsort_s, GitHub repository (https://github.com/noporpoise/sort_r).

Index