Package 'bigstep'

April 5, 2017

Type Package

Title Stepwise Selection for Large Data Sets

Version 0.7.4

Date 2017-4-5

Author Piotr Szulc

Maintainer Piotr Szulc <piotr.michal.szulc@gmail.com>

Description Selecting linear and generalized linear models for large data sets using modified stepwise procedure and modern selection criteria (like modifications of Bayesian Information Criterion). Selection can be performed on data which exceed RAM capacity. Special selection strategy is available, faster than classical stepwise procedure.

License GPL-3

Depends R (>= 3.2.2)

Imports stats, methods, utils, RcppEigen, speedglm, bigmemory, R.utils, matrixStats

Suggests testthat, devtools

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2017-04-05 21:57:37 UTC

R topics documented:

aic 2
bic 2
bigstep 3
combineBigMatrices 7
fitLinear 8
fitLogistic 8
fitPoisson 9
maic 10

maic2 10
mbic 11
mbic2 12
selectModel 12
singleTests 15

Index **17**

`aic` *AIC*

Description

Calculate AIC (Akaike Information Criterion).

Usage

```
aic(loglik, n, k)
```

Arguments

`loglik`	a numeric, the log-likelihood.
`n`	an integer > 0, a number of observations.
`k`	an integer >= 0, a number of selected variables.

Value

A number, a value of AIC.

Examples

```
aic(loglik=10, n=100, k=5)
```

`bic` *BIC*

Description

Calculate BIC (Bayesian Information Criterion).

Usage

```
bic(loglik, n, k)
```

Arguments

loglik	a numeric, the log-likelihood.
n	an integer > 0, a number of observations.
k	an integer >= 0, a number of selected variables.

Value

A number, a value of BIC.

Examples

```
bic(loglik=10, n=100, k=5)
```

---

bigstep *Model selection*

Description

Model selection using the stepwise procedure and the chosen criterion.

Details

To find the best model (linear or generalized), the following algorithm (a modification of the stepwise selection) is used [3]. In the first step the likelihood ratio tests between two regression models are performed: 1) with only the intercept, 2) with the intercept and every single variable from the matrix `X`. P-values are calculated and variables with p > `minpv` are excluded from the model selection procedure. In the second step (multi-forward) we start with the null model and add variables which decrease `crit.multif` (in order from the smallest p-value). The step is finished after we add `maxf` variables or none of remaining variables improve `crit.multif`. Then the classical backward selection is performed (with `crit`). When there is no variables to remove, the last step, the classical stepwise procedure, is performed (with `crit`).

Results from this four-step procedure should be very similar to the classical stepwise procedure (when we start with the null model and do not omit variables with high p-values) but the first one is much quicker. The most time-consuming part is the forward step in the stepwise selection (in the multi-forward step we do not add the best variable but any which decrease `crit.multif`) and it is performed less often when we start with a reasonable model (sometimes you can find the best model without using the stepwise selection). But you can omit the first three steps if you set `multif=FALSE` and `minpv=1`. Resignation from the multi-forward step can be reasonable when you expect that the final model should be very small (a few variables).

If your data are too big to store in RAM, you should read them with the `read.big.matrix` function from the `bigmemory` packages. The `selectModel` function will recognize that `X` is not an ordinary matrix and split your data to smaller parts. It will not change results but is necessary to work with big data.

The default criterion in the model selection procedure is a modification of the Bayesian Information Criterion, mBIC [1]. It was constructed to control the so-called Family-wise Error Rate (FWER) at

the level near 0.05 when you have a lot of explanatory variables and only a few of them should stay in the final model. If you are interested in controlling the so-called False Discovery Rate (FDR) is such type of data, you can change `crit` to `mBIC2` [2], which controls FDR at the level near 0.05. There are more criteria to choose from or you can easily define your own (see 'Examples')

If you do not have the desing matrix in one file, you have to combine them. It can be problematic if your data are big, so you can use the `combineBigMatrices` function from this package. You can use it also when you have to transpose `X` (because you have variables in rows) or you want to change names of columns.

Author(s)

Piotr Szulc

References

[1] M. Bogdan, J.K. Ghosh, R.W. Doerge (2004), "Modifying the Schwarz Bayesian Information Criterion to locate multiple interacting quantitative trait loci", Genetics 167: 989-999.

[2] F. Frommlet, A. Chakrabarti, M. Murawska, M. Bogdan (2011), "Asymptotic Bayes optimality under sparsity for generally distributed effect sizes under the alternative". Technical report at arXiv:1005.4753.

[3] F. Frommlet, F. Ruhaltinger, P. Twarog, M. Bogdan (2012), "A model selection approach to genome wide association studies", Computational Statistics and Data Analysis 56: 1038-1051.

Examples

```
## Not run:
library(bigstep)
library(RcppEigen)

# data1
set.seed(1)
n <- 100
M <- 50
X <- matrix(rnorm(n*M), ncol=M)
colnames(X) <- 1:M
snp <- 10*(1:5)
y <- rowSums(X[, snp]) + rnorm(n)

# you can use classical criteria to such type of data:
selectModel(X, y, crit=aic)
model <- selectModel(X, y, crit=bic)
summary(fastLm(X[, model], y))


# data2
set.seed(1)
n <- 1e3
M <- 1e4
X <- matrix(rnorm(M*n), ncol=M)
colnames(X) <- 1:M
```

```
snp <- 1e3*(1:10)
y <- rowSums(X[, snp]) + rnorm(n, sd=2)
Q <- matrix(rnorm(n*5), n, 5)  # additional variables
colnames(Q) <- -(1:5)

# single tests + multi-forward + multi-backward + stepwise using mBIC
selectModel(X, y, p=M)
selectModel(X, y, p=M, multif=FALSE)  # only single tests + stepwise
selectModel(X, y, p=M, multif=FALSE, minpv=1)  # only stepwise

# you can start with a model with variables from Q and force that
# 1st and 5th would not be removed
selectModel(X, y, Xm=Q, stay=c(1, 5), p=M, crit=mbic2)

# after reducing the size of matrix X (removing variables with
# p-value > minpv), save it to a file (you can use it in another
# model selection, it will be faster)
selectModel(X, y, p=M, file.out="Xshort")
Xs <- read.table("Xshort.txt", header=TRUE)
selectModel(Xs, y, p=M, minpv=1)

# selectModel(X, y, crit=bic)  # bad idea...

# you can define your own criterion
# (it can depend on log-likelihood (loglik), the number of observations (n),
# the number of variables currently in a model (k), the matrix with variables
# currenly in a model (Xm) and constants)
myCrit <- function(loglik, k, n, c1=2, c2=4) {
  -c1*loglik + 10*sqrt(k*c2)
}
selectModel(X, y, multif=FALSE, crit=myCrit, c1=0.18)
selectModel(X, y, multif=FALSE,
           crit=function(loglik, k, n) -0.17*loglik + 10*sqrt(k*4))

# your criterion can depend on eg. the average correlation between variables
# currently in a model
myCrit2 <- function(loglik, Xm) {
  n <- nrow(Xm)
  k <- ncol(Xm)
  m.cor <- 0
  np <- 1
  if (k > 1) {
    pairs <- combn(1:k, 2)
    np <- ncol(pairs)
    for (i in 1:np) {
      res <- suppressWarnings(abs(cor(Xm[, pairs[1, i]], Xm[, pairs[2, i]])))
      if (is.na(res))
        res <- 0
      m.cor <- m.cor + res
    }
  }
 -2*loglik + k*log(n) + n*k*m.cor/np
}
```

```
selectModel(X, y, multif=FALSE, crit=myCrit2)

# you can define your own fitting function
# (it can depend on X and y and should return the log-likelihood)
fitLinear2 <- function(X, y) {
  model <- RcppEigen::fastLmPure(X, y, method=2)
  rss <- sum(model$residuals^2)
  n <- length(y)
  loglik <- -n/2*log(rss/n)
  return(loglik)
}
system.time(selectModel(X, y, p=M, fitFun=fitLinear))
system.time(selectModel(X, y, p=M, fitFun=fitLinear2))
# fastLmPure with method=2 is faster but sometimes gives errors


# data3 (big data)
library(bigmemory)
# if it is possible, set type="char", reading will be quicker
X <- read.big.matrix("X.txt", sep=" ", type="char", head=TRUE)
y <- read.table("Trait.txt")
Q <- read.table("Q.txt")
M <- ncol(X)
selectModel(X, y, p=M)
selectModel(X, y, p=M, minpv=0.001) # if you do not have time...


# data4
set.seed(1)
n <- 50
M <- 1e3
X <- matrix(runif(M*n), ncol=M)
colnames(X) <- 1:M
snp <- 1e2*(1:5)
mu <- rowSums(X[, snp])
y <- rpois(n, exp(mu))
selectModel(X, y, p=M, fitFun=fitLinear, multif=FALSE)
selectModel(X, y, p=M, fitFun=fitPoisson, multif=FALSE)

set.seed(1)
n <- 100
X <- matrix(runif(M*n, -5, 5), ncol=M)
colnames(X) <- 1:M
mu <- rowSums(X[, snp])
p <- 1/(1 + exp(-mu))
y <- rbinom(n, 1, p)
selectModel(X, y, p=M, fitFun=fitLogistic, multif=FALSE)

## End(Not run)
```

combineBigMatrices *Combine, transpose big matrices and change colnames*

Description

Combine and/or transpose big matrices and write to a file with new names of columns

Usage

```
combineBigMatrices(X, Z = NULL, file.out = "XZ.txt", transX = FALSE,
  transZ = FALSE, new.names = NULL, maxp = 1e+07, verbose = TRUE)
```

Arguments

X	an object of class big.matrix.
Z	an object of class big.matrix. Specify if you want to combine matrices.
file.out	a character string. The name of the output file.
transX	a logical. If TRUE, the matrix X will be transposed.
transZ	a logical. If TRUE, the matrix Z will be transposed.
new.names	a vector of strings. Specify if you want to change names of variables.
maxp	a numeric. If X is big, it will be splitted into parts with maxp elements. It will not change results, but it is necessary if your computer does not have enough RAM. Set to a lower value if you still have problems.
verbose	a logical. Set FALSE if you do not want to see any information during the selection procedure.

Value

A numeric, a number of variables successfully written.

Examples

```
## Not run:
library(bigmemory)
X <- matrix(sample(0:2, 20, replace=TRUE), 2, 10)
Z <- matrix(sample(0:2, 20, replace=TRUE), 2, 10)
write.table(X, "X.txt", col.names=FALSE, row.names=FALSE)
write.table(Z, "Z.txt", col.names=FALSE, row.names=FALSE)
X <- read.big.matrix("X.txt", sep=" ", type="char")
Z <- read.big.matrix("Z.txt", sep=" ", type="char")
# if it is possible, set type="char", reading will be quicker
names <- c("X1", "X2", "Z1", "Z2")
combineBigMatrices(X, Z, transX=TRUE, transZ=TRUE, new.names=names)

## End(Not run)
```

fitLinear *Linear regression*

Description

Fit the linear regression model and calculate the log-likelihood.

Usage

```
fitLinear(X, y)
```

Arguments

X	a numeric matrix.
y	a numeric vector.

Value

A number, the log-likelihood.

Examples

```
set.seed(1)
n <- 100
M <- 10
X <- matrix(rnorm(M*n), ncol=M)
y <- X[, 2] - X[, 3] + X[, 6] - X[, 10] + rnorm(n)
fitLinear(X, y)
```

fitLogistic *Logistic regression*

Description

Fit the logistic regression model and calculate the log-likelihood.

Usage

```
fitLogistic(X, y)
```

Arguments

X	a numeric matrix.
y	a numeric vector.

Value

A number, the log-likelihood.

Examples

```
set.seed(1)
n <- 100
M <- 10
X <- matrix(rnorm(M*n), ncol=M)
mu <- X[, 2] - X[, 3] + X[, 6] - X[, 10]
p <- 1/(1 + exp(-mu))
y <- rbinom(n, 1, p)
fitLogistic(X, y)
```

---

fitPoisson *Poisson regression*

Description

Fit the Poisson regression model and calculate the log-likelihood.

Usage

```
fitPoisson(X, y)
```

Arguments

X	a numeric matrix.
y	a numeric vector.

Value

A number, the log-likelihood.

Examples

```
set.seed(1)
n <- 100
M <- 10
X <- matrix(rnorm(M*n), ncol=M)
mu <- X[, 2] - X[, 3] + X[, 6] - X[, 10]
y <- rpois(n, exp(mu))
fitPoisson(X, y)
```

maic *mAIC*

Description

Calculate mAIC (modified Akaike Information Criterion).

Usage

```
maic(loglik, n, k, p, const = 4)
```

Arguments

loglik	a numeric, the log-likelihood.
n	an integer > 0, a number of observations.
k	an integer >= 0, a number of selected variables.
p	an integer > 0, a number of all variables or a weight.
const	numeric > 0, the expected number of significant variables.

Value

A number, a value of mAIC.

Examples

```
maic(loglik=10, n=100, k=5, p=50)
```

maic2 *mAIC2*

Description

Calculate mAIC2 (the second version of modified Akaike Information Criterion).

Usage

```
maic2(loglik, n, k, p, const = 4)
```

Arguments

loglik	a numeric, the log-likelihood.
n	an integer > 0, a number of observations.
k	an integer >= 0, a number of selected variables.
p	an integer > 0, a number of all variables or a weight.
const	numeric > 0, the expected number of significant variables.

Value

A number, a value of mAIC2.

Examples

```
mbic2(loglik=10, n=100, k=5, p=50)
```

`mbic` *mBIC*

Description

Calculate mBIC (modified Bayesian Information Criterion).

Usage

```
mbic(loglik, n, k, p, const = 4)
```

Arguments

`loglik`	a numeric, the log-likelihood.
`n`	an integer > 0, a number of observations.
`k`	an integer >= 0, a number of selected variables.
`p`	an integer > 0, a number of all variables or a weight.
`const`	numeric > 0, the expected number of significant variables.

Value

A number, a value of mBIC.

Examples

```
mbic(loglik=10, n=100, k=5, p=50)
```

mbic2 *mBIC2*

Description

Calculate mBIC2 (the second version of modified Bayesian Information Criterion).

Usage

```
mbic2(loglik, n, k, p, const = 4)
```

Arguments

loglik	a numeric, the log-likelihood.
n	an integer > 0, a number of observations.
k	an integer >= 0, a number of selected variables.
p	an integer > 0, a number of all variables or a weight.
const	numeric > 0, the expected number of significant variables.

Value

A number, a value of mBIC2.

Examples

```
mbic2(loglik=10, n=100, k=5, p=50)
```

selectModel *Model selection*

Description

Model selection using the stepwise procedure and the chosen criterion.

Usage

```
selectModel(X, y, fitFun = fitLinear, crit = mbic, Xm = NULL,
  stay = NULL, minpv = 0.15, multif = TRUE, crit.multif = bic,
  maxf = min(ncol(X), 70), minb = 0, fastST = FALSE, maxp = 1e+06,
  verbose = TRUE, file.out = NULL, ...)
```

Arguments

X	a numeric matrix or an object of class big.matrix (see 'Details'). The rows of X contain the samples, the columns of X contain the observed variables. If your have variables in rows, see 'Details'.
y	a numeric vector of responses. The length of y must equal the number of rows of X.
fitFun	a function which fits the regression model and calculate the logarithm of the likelihood function (loglike). You can use your own function or one of these: fitLinear, fitLogistic, fitPoisson.
crit	a function defining the model selection criterion. You can use your own function or one of these: bic, mbic, mbic2, aic, maic, maic2.
Xm	a numeric matrix. Additional variables which will be included in the first step of the model selection procedure.
stay	a numeric vector. Columns from Xm which should be included in the model in all selection steps.
minpv	a numeric. Variables from X with p-values for the likelihood ratio tests (see 'Details') higher than minpv will be excluded from the model selection procedure. If you do not want to do this, set minpv=1 (not recommended if you have big data).
multif	a logical. If TRUE, the multi-forward step will be performed (see 'Details').
crit.multif	a function defining the model selection criterion in the multi-forward step. See crit (but only criteria with small penalties are recommended).
maxf	a numeric, a maximal number of variables in the final model in the multi-forward step.
minb	a numeric, a minimal number of variables in the final model in the backward selection (see 'Details').
fastST	a logical. If TRUE, the Pearson correlation coefficients between y and all columns of X are calculated instead of the likelihood ratio tests (see 'Details'). It is faster but works only if you do not have any missing values.
maxp	a numeric. If X is big, it will be splitted into parts with maxp elements. It will not change results, but it is necessary if your computer does not have enough RAM. Set to a lower value if you still have problems.
verbose	a logical. Set FALSE if you do not want to see any information during the selection procedure.
file.out	a character string, If not NULL (and minpv<1), the variables with p-value < minpv will be saved to file.out (txt file).
...	optional arguments to crit or crit.multif.

Details

To find the best model (linear or generalized), the following algorithm (a modification of the stepwise selection) is used [3]. In the first step the likelihood ratio tests between two regression models are performed: 1) with only the intercept, 2) with the intercept and every single variable from the

matrix X. P-values are calculated and variables with p > minpv are excluded from the model selection procedure. In the second step (multi-forward) we start with the null model and add variables which decrease crit.multif (in order from the smallest p-value). The step is finished after we add maxf variables or none of remaining variables improve crit.multif. Then the classical backward selection is performed (with crit). When there is no variables to remove, the last step, the classical stepwise procedure, is performed (with crit).

Results from this four-step procedure should be very similar to the classical stepwise procedure (when we start with the null model and do not omit variables with high p-values) but the first one is much quicker. The most time-consuming part is the forward step in the stepwise selection (in the multi-forward step we do not add the best variable but any which decrease crit.multif) and it is performed less often when we start with a reasonable model (sometimes you can find the best model without using the stepwise selection). But you can omit the first three steps if you set multif=FALSE and minpv=1. Resignation from the multi-forward step can be reasonable when you expect that the final model should be very small (a few variables).

If your data are too big to store in RAM, you should read them with the read.big.matrix function from the bigmemory packages. The selectModel function will recognize that X is not an ordinary matrix and split your data to smaller parts. It will not change results but is necessary to work with big data.

The default criterion in the model selection procedure is a modification of the Bayesian Information Criterion, mBIC [1]. It was constructed to control the so-called Family-wise Error Rate (FWER) at the level near 0.05 when you have a lot of explanatory variables and only a few of them should stay in the final model. If you are interested in controlling the so-called False Discovery Rate (FDR) is such type of data, you can change crit to mBIC2 [2], which controls FDR at the level near 0.05. There are more criteria to choose from or you can easily define your own (see 'Examples')

If you do not have the desing matrix in one file, you have to combine them. It can be problematic if your data are big, so you can use the combineBigMatrices function from this package. You can use it also when you have to transpose X (because you have variables in rows) or you want to change names of columns.

Value

The names of variables in the final model.

Author(s)

Piotr Szulc

References

[1] M. Bogdan, J.K. Ghosh, R.W. Doerge (2004), "Modifying the Schwarz Bayesian Information Criterion to locate multiple interacting quantitative trait loci", Genetics 167: 989-999.

[2] F. Frommlet, A. Chakrabarti, M. Murawska, M. Bogdan (2011), "Asymptotic Bayes optimality under sparsity for generally distributed effect sizes under the alternative". Technical report at arXiv:1005.4753.

[3] F. Frommlet, F. Ruhaltinger, P. Twarog, M. Bogdan (2012), "A model selection approach to genome wide association studies", Computational Statistics and Data Analysis 56: 1038-1051.

Examples

```
set.seed(1)
n <- 100
M <- 10
X <- matrix(rnorm(M*n), ncol=M)
colnames(X) <- 1:M
y <- X[, 2] - X[, 3] + X[, 6] - X[, 10] + rnorm(n)
selectModel(X, y, p=M)

# more examples: type ?bigstep
```

`singleTests` *Single tests*

Description

Perform the likelihood ratio tests between two regression models: 1) with only the intercept, 2) with the intercept and every single variable from the matrix `X`.

Usage

```
singleTests(X, y, fitFun = fitLinear, fastST = FALSE, maxp = 1e+06,
  verbose = TRUE)
```

Arguments

`X`	a numeric matrix or an object of class big.matrix (see 'Details'). The rows of `X` contain the samples, the columns of `X` contain the observed variables. If your have variables in rows, see 'Details'.
`y`	a numeric vector of responses. The length of `y` must equal the number of rows of `X`.
`fitFun`	a function which fits the regression model and calculate the logarithm of the likelihood function (loglike). You can use your own function or one of these: `fitLinear`, `fitLogistic`, `fitPoisson`.
`fastST`	a logical. If `TRUE`, the Pearson correlation coefficients between `y` and all columns of `X` are calculated instead of the likelihood ratio tests (see ?bigstep). It is faster but works only if you do not have any missing values.
`maxp`	a numeric. If `X` is big, it will be splitted into parts with `maxp` elements. It will not change results, but it is necessary if your computer does not have enough RAM. Set to a lower value if you still have problems.
`verbose`	a logical. Set `FALSE` if you do not want to see any information during the selection procedure.

Value

A numeric vector with p-values of the likelihood ratio test (or the Pearson correlation test if `fastST=TRUE`).

Examples

```
set.seed(1)
n <- 100
M <- 10
X <- matrix(rnorm(M*n), ncol=M)
y <- X[, 2] - X[, 3] + X[, 6] - X[, 10] + rnorm(n)
singleTests(X, y)
```

Index

aic, 2

bic, 2
bigstep, 3

combineBigMatrices, 7

fitLinear, 8
fitLogistic, 8
fitPoisson, 9

maic, 10
maic2, 10
mbic, 11
mbic2, 12

selectModel, 12
singleTests, 15