Package ‘ascrda’

February 19, 2015

Type Package
Title Augmented SCRDA
Version 1.15
Date 2011-12-13
Author M. S. Islam and A. I. McLeod
Maintainer <aimcleod@uwo.ca>
Depends R (>= 2.10),class,nnet,rda,MASS,sfsmisc,e1071,pamr
Description Uses nearest neighbor autocovariates with SCRDA, provides cross-validation functions for the delete-d method, a simulation method is provided for generating synthetic microarray data, a consistent and simple interface is provided for functions from other R packages for SCRDA, PAM, SVM, DLDA, kNN.
LazyLoad yes
LazyData yes
Classification/ACM G.4, I.5
Classification/MSC 62H30
License GPL (>= 2)
Repository CRAN
Date/Publication 2011-12-15 10:10:02

R topics documented:

  ascrda-package .............................................. 2
  ascrda .................................................. 3
  FitDLDA .................................................. 5
  FitkNN .................................................. 7
  FitPAM .................................................. 9
  FitRda .................................................. 10
  FitSvm .................................................. 12
  getTrainTest ............................................ 14
Augmented shrunken centroid discriminant analysis

Description

This paper uses nearest neighbour autocovariates (Holmes and Adams, 2003) together with regularized shrunken centroid discriminant analysis (Guo, Hastie and Tibshirani, 2007) for discriminant analysis or class prediction. This technique was described in the thesis by Islam (2008). This method is especially useful in small-n-big-p problems as in microarrays where n, the number of subjects is usually less than 100 and p, the number of genes is over 10,000. Other functions implemented in this package include nearest neighbor autocovariates, the delete-d method for cross-validation and a method for simulating non-linear microarray data. An interesting microarray dataset with smokers (Spira et al., 2004) is provided. For convenience, a consistent interface is provided for other classification methods including nearest neighbours, diagonal linear discriminant analysis and support vector machines.

Details

Package: ascrda
Type: Package
Version: 1.15
Date: 2011-12-13
License: GPL (>= 2.10)
LazyLoad: yes
LazyData: yes

Author(s)

M. S. Islam and A. I. McLeod <aimcleod@uwo.ca>

References

**ascrda**

**Test error rate for ASCRDA**

**Description**

The misclassification rate on the test data is computed for the following methods: ASCRDAk - SCRDA with nearest neighbor autocovariates, k=kHat; ASCRDA1 - SCRDA with nearest neighbor autocovariate, k=1; SCRDA - regular SCRDA, shrunken centroid regularized discriminant analysis, k=0;

**Usage**

```r
glimpse(ascrda(X, Y, Xt, Yt, alpha = seq(0, 0.99, 0.11), delta = seq(0, 3, 0.2), SCRDAmethod=c("ASCRDAk","ASCRDA1","SCRDA"), ...))
```

**Arguments**

- `X` training, data matrix, n-by-G
- `Y` training output
- `Xt` test data matrix
- `Yt` test output
- `alpha` alpha in rda function
- `delta` delta in rda function
- `SCRDAmethod` one of ASCRDAk, ASCRDA1, SCRDA. Default is ASCRDAk
- `...` optional arguments passed to rda

**Value**

Vector of length 2, misclassification error rate and K, number of knn neighbours selected

**Author(s)**

M. S. Islam and A. I. McLeod, email: aimecleod@uwo.ca
References


Examples

```r
#Example 1. Tiny test example - 1.4 seconds
set.seed(132561)#for exact reproducibility
M<-symm(n=c(10,10), nt=c(10,10), B=c(5,5), m=c(5,5))
startTime <- proc.time()[3]
ans<-ascrda(M$X, M$y, M$Xt, M$yt)
endTime <- proc.time()[3]
TotalTime <- endTime-startTime
out <- c(ans[[1]], TotalTime)
names(out)<-c("MisclassificationRate", "TotalTime")
out
#
#MisclassificationRate TotalTime
# 0.45 1.51

## Not run:
#Example 2. large example. About 78 sec
#training sample size = 200
#test sample size = 1000
#number of genes = 1000
set.seed(132561)#for exact reproducibility
BScenario <- list(c(50, 10), c(10, 50), c(10, 10), c(50, 50))
mscenario <- list(c(20, 100), c(100, 20), c(100, 100), c(20, 20))
M <- symm(n=c(100,100), nt=c(500,500), rho = c(0.9, 0.9),
B=BScenario[[1]], m=mscenario[[1]], fe=0.05)
StartTime <- proc.time()[3]
ANS1 <- ascrda(M$X, M$y, M$Xt, M$yt)
EndTime1 <- proc.time()[3]
ANS2 <- ascrda(M$X, M$y, M$Xt, M$yt, SCRDAmethod = "ASCRDA1")
EndTime2 <- proc.time()[3]
ANS3 <- ascrda(M$X, M$y, M$Xt, M$yt, SCRDAmethod = "SCRDA")
EndTime3 <- proc.time()[3]
TotalTimes <- c(EndTime1-StartTime,EndTime2-EndTime1,EndTime3-EndTime2)
Errs <- c(ANS1[[1]],ANS2[[1]],ANS3[[1]])
out <- matrix(c(Errs, TotalTimes),ncol=2)
rownames(out) <- c("ASCRDAk", "ASCRDA1", "SCRDA")
colnames(out) <- c("ErrorRate", "Time")
out
#
# ErrorRate Time
#ASCRDAk 0.076 43.39
#ASCRDA1 0.185 22.02
#SCRDA 0.300 12.22
```
FitDLDA

Diagonal linear/quadratic discriminant analysis

Description

Diagonal linear/quadratic DA

Usage

FitDLDA(X, Y, Xt, Yt, pool)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>design matrix, training sample</td>
</tr>
<tr>
<td>Y</td>
<td>output, training, consecutive integers</td>
</tr>
<tr>
<td>Xt</td>
<td>design matrix, test sample</td>
</tr>
<tr>
<td>Yt</td>
<td>output, test, consecutive integers</td>
</tr>
<tr>
<td>pool</td>
<td>set to FALSE for quadratic</td>
</tr>
</tbody>
</table>

Details

FitDLDA is very fast. Often there is little difference between the linear and quadratic methods.

Value

list with two components: misclassification rate and confusion matrix for test.

Author(s)

AIM
References

R package: sfsmisc. Utilities from Seminar fuer Statistik ETH Zurich. Martin Maechler. 2010

Examples

```r
# Example 1. small example - takes about 1.23 sec
# Not run:
set.seed(132561)#for exact reproducibility
M<-symma(n=c(50,50), nt=c(100,100), B=c(20,4), m=c(20,100), fE=0.05)
StartTime <- proc.time()[3]
ANS1 <- fitDLDA(M$y, M$yt, M$Xt, M$yt)
EndTime1 <- proc.time()[3]
ANS2 <- fitDLDA(M$y, M$yt, M$Xt, M$yt, pool=FALSE)
EndTime2 <- proc.time()[3]
TotalTime1 <-EndTime1-StartTime
TotalTime2 <-EndTime2-EndTime1
names(TotalTime) <- c("DLDA","DQDA")
TotalTime
ANS1
ANS2

# Example 2. large example
#training sample size = 200
#test sample size = 1000
#number of genes = 1000
# Not run: set.seed(132561)#for exact reproducibility
BSceano<- list(c(50,10),c(10,50), c(10, 10), c(50, 50))
msceano<- list(c(20,100),c(100, 20),c(100, 100),c(20, 20))
M <- symma(n=c(100,100), nt=c(500,500), rho = c(0.9, 0.9),
    B=BSceano[[1]], m=msceano[[1]], fE=0.05)
StartTime <- proc.time()[3]
ANS <- fitDLDA(M$X, M$yt, M$Xt, M$yt)
EndTime <- proc.time()[3]
TotalTime <-EndTime-StartTime
ans <- c(ANS$Err, TotalTime)
names(ans) <- c("MisclassificationRate", "TotalTime")
ans
##> ans
##MisclassificationRate TotalTime
##0.312 0.110

# Example 3. Spira Dataset
#Spira Dataset
y <- as.numeric(Spira$Patients$STATUS)
X <- Spira$x
genes <- Spira$genes
X <- X[,genes]
set.seed(132561)#for exact reproducibility
```
FitkNN

**kNN classification**

**Description**

kNN classification

**Usage**

FitkNN(X, Y, Xt, Yt, k = 1)

**Arguments**

- **X**
  - design matrix, training sample
- **Y**
  - response, training sample
- **Xt**
  - design matrix, test sample
- **Yt**
  - response, test
- **k**
  - number of nearest neighbours

**Value**

list with two components: misclassification rates and confusion matrices for both the training and test samples.

**Author(s)**

AIM
Examples

# Example 1. Small example with 2 classes.
set.seed(132561)#for exact reproducibility
M <- symm(n=c(50,50), nt=c(100,100), B=c(20,4), m=c(20,100), fE=0.05)
StartTime <- proc.time()[3]
ANS <- FittkNN(M$X, M$y, M$Xt, M$yt)
EndTime <- proc.time()[3]
TotalTime <-EndTime-StartTime
out <- c(ANS$r[2], TotalTime)
names(out) <- c("MisclassificationRate", "TotalTime")
out

# Example 2. large example
# training sample size = 200
# test sample size = 1000
# number of genes = 1000
## Not run: #FittkNN is fast but not symma
set.seed(132561)#for exact reproducibility
BSenario <- list(c(50, 10),c(10, 50), c(10, 10), c(50, 50))
msenario <- list(c(20,100),c(100, 20),c(100, 100),c(20, 20))
M <- symm(n=c(100,100), nt=c(500,500), rho = c(0.9, 0.9),
  B=BSenario[[1]], m=msenario[[1]], FE=0.05)
StartTime <- proc.time()[3]
ANS <- FittkNN(M$X, M$y, M$Xt, M$yt)
EndTime <- proc.time()[3]
TotalTime <-EndTime-StartTime
ans <- c(ANS$r[2], TotalTime)
names(ans) <- c("MisclassificationRate", "TotalTime")
ans

## End(Not run)

## Example 3. With 3 classes. Still fast - about 0.3 sec
# Spira dataset
y <- as.numeric(Spira$patients$STATUS)
X <- Spira$x
genesis <- Spira$genes
X <- X[, genes]
set.seed(132561)#for exact reproducibility
M <- getTrainTest(X, y, d=0.25)
StartTime <- proc.time()[3]
ANS <- FittkNN(M$X, M$y, M$Xt, M$yt)
EndTime <- proc.time()[3]
TotalTime <-EndTime-StartTime
out <- c(ANS$r[2], TotalTime)
names(out) <- c("MisclassificationRate", "TotalTime")
out
ANS$CMats[[2]]
Description

Class prediction using nearest centroid using regularization.

Usage

FitPAM(X, Y, Xt, Yt, PamMethod=c("default", "kNN1", "kNN"))

Arguments

X design matrix, training sample
Y output, training, consecutive integers
Xt design matrix, test sample
Yt output, test, consecutive integers
PamMethod one of "default", "kNN1", "kNN". Default is no autocovariate adjustment

Details

See references.

Value

list with two components: Err = misclassification rate confusionMatrix = confusion matrix for test.

Author(s)

A. I. McLeod, aimcleod@uwo.ca

References

R. Tibshirani et al. (2003), "Class Prediction by Nearest Shrunken Centroids, with Applications to DNA Microarrays" Statistical Science, 18, 104-117.

See Also

pamr.train
Examples

```r
# takes about 0.5 sec
# Spira Dataset
y <- as.numeric(Spira$Patients$STATUS)
X <- Spira$x
genes <- Spira$genes
X <- X[, genes]
set.seed(132561)  # for exact reproducibility
M <- getTrainTest(X, y, d=0.25)
StartTime <- proc.time()[3]
ANS <- fitpam(M$x, M$y, M$x, M$yt)
EndTime <- proc.time()[3]
TotalTime <- EndTime - StartTime
TotalTime
ANS

# Further examples (continuing from above)
fitpam(M$x, M$y, M$x, M$yt, PamMethod="kNN1")
fitpam(M$x, M$y, M$x, M$yt, PamMethod="kNN")
```

**FitRda**  

Regularized Discriminant Analysis

Description

Fits RDA, computes mis-classification rates on training and test data.

Usage

`FitRda(X, y, Xt, yt, alpha = seq(0, 0.99, 0.11), delta = seq(0, 3, 0.2), ...)`

Arguments

- **X**  
  training, expression matrix
- **y**  
  response vector taking values 1, 2, ...
- **Xt**  
  test, expression matrix
- **yt**  
  response vector for test data taking values 1, 2, ...
- **alpha**  
  tuning parameter
- **delta**  
  tuning parameter
- **...**  
  optional arguments, passed to rda

Value

Error rates for training and test data if rda.cv() works. Otherwise no result returned. You need to use: `out<-FitRda(...)` and `exists(out)` to determine if an error occurred.
Author(s)
M. S. Islam and A. I. McLeod, email: aimcleod@uwo.ca

References

Examples

#Example 1
#Tiny test example - 1.4 seconds
M<-symma(n=c(10,10), nt=c(10,10), B=c(5,5), m=c(5,5))
startTime <- proc.time()[3]
ans<-FitRda(M$X, M$y, M$Xt, M$yt)
endTime <- proc.time()[3]
TotalTime <- endTime-startTime
out <- c(ans[2], TotalTime)
names(out)<-c("MisclassificationRate", "TotalTime")
out
#
## Not run:
#Example 2. large example. About 12.6 sec
#training sample size = 200
#test sample size = 1000
#number of genes = 1000
set.seed(132561)#for exact reproducibility
B Scenario <- list(c(50, 10), c(10, 50), c(10, 10), c(50, 50))
M Scenario <- list(c(20, 100), c(100, 20), c(100, 100), c(20, 20))
M <- symna(n=c(100,100), nt=c(500,500), rho = c(0.9, 0.9),
B=B Scenario[[1]], M=M Scenario[[1]], fe=0.05)
startTime <- proc.time()[3]
ANS <- FitRda(M$X, M$y, M$Xt, M$yt)
endTime <- proc.time()[3]
TotalTime <- endTime-startTime
ans <- c(ANS[2], TotalTime)
names(ans) <- c("MisclassificationRate", "TotalTime")
an
##> ans
## MisclassificationRate TotalTime
## 0.30 12.61

## End(Not run)

## Not run:
#Example 3. With 3 classes. About 76.5 sec
#Spira Dataset
y <- as.numeric(Spira$Patients$STATUS)
X <- Spira$X
set.seed(132561)#for exact reproducibility
M <- getTrainTest(X, y, d=0.25)
startTime <- proc.time()[3]
FitSvm

Classification using Support Vector Machine

Description

Fits Support Vector Machine, computes mis-classification rates on training and test data.

Usage

FitSvm(X,Y,Xt,Yt, tuningQ=FALSE, cross=10, ...)

Arguments

X training, data matrix, input
Y training output, factor
Xt test, data matrix, input
Yt test output, factor
tuningQ use cross-validation to choose tuning parameter
cross k for k-fold cv
... optional arguments passed to svm()

Value

Error rates for training and test data

Author(s)

M. S. Islam and A. I. McLeod, email: aimcleod@uwo.ca
References


Examples

#Example 1.
#Simple fast example - about 0.25 sec.
#Small example. p=400 genes; training sample size = 100; test sample size = 200.
set.seed(132561) #for exact reproducibility
M <- symma(n=c(50,50), nt=c(100,100), B=c(20,4), m=c(20,100), fE=0.05)
FitSvm(M$X, M$y, M$Xt, M$yt)

#Example 2.
#Small example. Compare tuning and no tuning.
#p=400 genes; training sample size = 100; test sample size = 200.
## Not run:
set.seed(132561) #for exact reproducibility
M <- symma(n=c(50,50), nt=c(100,100), B=c(20,4), m=c(20,100), fE=0.05)
startTime <- proc.time()[3]
ANS1 <- FitSvm(M$X, M$y, M$Xt, M$yt)
EndTime1 <- proc.time()[3]
ANS2 <- FitSvm(M$X, M$y, M$Xt, M$yt, tuning=TRUE)
EndTime2 <- proc.time()[3]
TotalTime1 <-EndTime1-StartTime
TotalTime2 <-EndTime2-EndTime1
TotalTime <- c(TotalTime1, TotalTime2)
names(TotalTime) <- c("default","tuning")
Err <- c(ANS1$r[2],ANS2$r[2])
names(Err) <- c("default","tuning")
m <- matrix(c(Err,TotalTime), ncol=2)
dimnames(m) <- list(names(Err), c("MisclassificationRate", "TotalTime"))
m
#
#OUTPUT
##> m
##
##       MisclassificationRate TotalTime
## default 0.26   0.25
## tuning  0.39  35.48
#
#Note: tuning did not improve accuracy and timing increased dramatically!

## End(Not run)

#Example 3. Large Example
## Not run:
Form training and test samples using delete-d method stratified sampling.

```r
#training sample size = 200
#test sample size = 1000
#number of genes = 1000
set.seed(132561)#for exact reproducibility
BScenario <- list(c(50, 10), c(10, 50), c(10, 10), c(50, 50))
mScenario <- list(c(20, 100), c(100, 20), c(100, 100), c(20, 20))
M <- syma(n=c(100,100), nt=c(500,500), rho=c(0.9, 0.9),
         B=BScenario[[1]], m=mScenario[[1]], fE=0.05)
startTime <- proc.time()[3]
ANS <- FitSvm(M$X, M$y, M$Xt, M$yt)
endTime <- proc.time()[3]
TotalTime <- endTime-startTime
ans <- c(ANS$r[2], TotalTime)
names(ans) <- c("MisclassificationRate", "TotalTime")
ans
#> ans

#MisclassificationRate          TotalTime
#  0.219            1.610

## End(Not run)

#Example 4. With 3 classes
#
#Spira Dataset
#takes about 0.25 seconds.
#So full cross-validation with 1000 replications ~8.6 hours
#Here p=9968; n=75; 3 classes
## Not run:
#Spira Dataset
y <- as.numeric(Spira$Patients$STATUS)
X <- Spira$X
genes <- Spira$genes
X <- X[,genes]
set.seed(132561)#for exact reproducibility
M <- getTrainTest(X, y, d=0.25)
startTime <- proc.time()[3]
ANS <- FitSvm(M$X, M$y, M$Xt, M$yt)
endTime <- proc.time()[3]
TotalTime <- endTime-startTime
out <- c(ANS$r[2], TotalTime)
names(out) <- c("MisclassificationRate", "Time")
out
TotalTime
## End(Not run)
```
**getTrainTest**

**Description**

Supports delete-d cross validation. For specified d, the training and test samples are constructed using a stratified sample to preserve the proportions in each class.

**Usage**

```r
getTrainTest(X, y, d = 0.25)
```

**Arguments**

- `X`: inputs in matrix form, n-by-p matrix
- `y`: output, p-vector
- `d`: proportion for hold-out sample, 0<d<1.

**Details**

Kim (2009) compares various CV methods and recommends the delete-d also known as the hold-out method for better accuracy than bootstrap or k-fold methods.

**Value**

- `X`: training inputs
- `y`: training output
- `Xt`: test inputs
- `yt`: test output

**Author(s)**

M. S. Islam and A. I. McLeod, email: aimeleod@uwo.ca

**References**


**Examples**

```r
X <- matrix(rnorm(90), nrow=30)
y <- rep(1:3, 10)
M <- getTrainTest(X, y, d=0.25)
```
**khat**  
*Optimal k in kNN using pseudolikelihood*

**Description**

Uses the profile pseudolikelihood to obtain the estimate for k, the number of nearest neighbors parameter in kNN.

**Usage**

```r
khat(X, Y, kmax = ceiling(length(Y) * 0.5), plot = TRUE)
```

**Arguments**

- `X`  
  An n-by-p matrix of covariates

- `Y`  
  Outputs with Q classes

- `kmax`  
  The maximum size of k

- `plot`  
  if TRUE, plot the profile deviance otherwise no plot

**Details**

When Q=2, the glm algorithm is used to compute the profile pseudologlikelihood and for Q>2, the function `multinom` in `nnet` is used.

**Value**

The estimate of k obtained by maximizing the pseudolikelihood is returned.

The result is returned invisibly if plot is TRUE.

**Author(s)**

M. S. Islam and A. I. McLeod  
Maintainer: <aimcleod@uwo.ca>

**References**

[http://biomet.oxfordjournals.org/cgi/content/abstract/90/1/99](http://biomet.oxfordjournals.org/cgi/content/abstract/90/1/99)

**See Also**

`multinom`
nnc

Examples

# Two classes example
library("MASS") # need synth.tr
X <- synth.tr[,1:2]
Y <- synth.tr[,3]
khat(X=X, Y=Y)

# Three classes example
library("MASS") # need lda
Y <- iris[,5]
X <- iris[,1:4]
kopt <- khat(X, Y)
kopt

# Mis-classification rates on training data. Of course FLDA does better in this case.
y <- factor(Y)
ans <- knn(train=X, test=X, k=kopt, cl=y)
etakNN <- sum(ans!=y)/length(y)
iris.lda <- lda(X, y)
yfitFLDA <- predict(iris.lda, newdata=X, dimen=1)$class
etaFLDA <- sum(yfitFLDA!=y)/length(y)
etac <- c(etaFLDA, etakNN)
names(eta) <- c("FLDA", "KNN")
etac

nnc

Computes nearest neighbor autocovariates

Description

Given n training examples in inputs X and output Y with two classes, the corresponding nearest neighbor autocovariates are computed. When there are more than 2 classes, X and Y contain the inputs and outputs for two of the classes and Xb contains inputs for all other classes.

Usage

nnc(X, Y, k)

Arguments

X
inputs, matrix with columns corresponding to inputs and rows to examples

Y
output, vector of length n, where n is the number of examples. Vector may be numeric, factor or character.

k
neighborhood size
Details

The k-th nearest neighbor autovariate is defined between two classes which we may denote by -1 and 1. For a given observation, we find all the k-nearest neighbors, and sum the number corresponding to each class. Then take the difference and divide by k. Symbolically,

\[ \sum_{j \in k_i} (I(y_j = 1) - I(y_j = -1))/k \]

sum j in k_i (I(y_j = 1) - I(y_j = -1))/k where eqn_k_i_k_j denotes the indicies of all observations in the k-th neighbor of the i-th observation.

Value

vector of length n of nearest neighbor autocovariates corresponding to training data for given k

Author(s)

M. S. Islam and A. I. McLeod Maintainer: <aimcleod@uwo.ca>

References


See Also

knn, knn.cv

Examples

#Example 1. With 2 classes.
library("MASS")
X <- synth.tr[,1:2]
Y <- synth.tr[,3]
# Nearest neighbor auto-covariates for synthetic data
sapply(1:10, FUN = function(k) nnc(X=X, Y=Y, k))

#Example 2. Iris dataset with 3 classes.
#Find both autocovariates
X <- iris[,1:4]
Y<- iris[,5]
k<-15
z<-nnc(X=X, Y=Y, k=k)
**nncTest**

*Nearest neighbor autocovariates for test data*

**Description**

The relative proportion of class \( j \) to class 1 based on the training data is computed for each point in the test data. This is useful in cross-validation with regularized discriminant analysis and as well as with statistical models such as logistic regression.

**Usage**

\[
\text{nncTest}(X, Y, Xtest, k)
\]

**Arguments**

- \( X \) training inputs, \( n \times p \) matrix, where \( n=\)sample size, \( p=\)number of input variables
- \( Y \) training outputs with \( K \) levels, length \( n \)
- \( Xtest \) test inputs, \( m \times p \) matrix, \( m=\)sample size
- \( k \) neighborhood size

**Value**

vector of length \( m \) of nearest neighbor autocovariates corresponding to each row of \( XTest \)

**Author(s)**

M. S. Islam and A. I. McLeod Maintainer: <aimcleod@uwo.ca>

**References**


**See Also**

- `nnc`, `knn`, `knn.cv`

**Examples**

```r
# Example 1. With 2 classes.
library("MASS") # need synth.tr
X <- synth.tr[,1:2]
Y <- synth.tr[,3]
XTest <- synth.te[,1:2]
k <- 21
nncTest(X=X, Y=Y, XTest=XTest, k)
```
Description

Microarray experiment lung tissue for 3 groups: smokers (C: 34), former smokers (F: 18) and never smokers (N: 23); n=75, p=9968.

Usage

data(Spira)

Format

A list with three named components Patients, X, and genes. The component Patients is a dataframe containing covariates about the 75 patients who participated. The second component X contains the microarray data corresponding to the 75 patients. The matrix X has 75 rows and 9968 columns. The vector genes indicates the 487 genes (columns) in X that were selected using SCRDA.

Details

The first component of Spira is a dataframe which is named Patients. The components are:

- PATIENT_IDa numeric vector
- STATUSa factor with levels C F N corresponding to current smoker, former smoker and never smoker
- START_YEARa numeric vector
- END_YEARa numeric vector
- PACK_YEARSa numeric vector
- AGEa numeric vector
- SEXa factor with levels F M
- RACIALa factor with levels AFA AMI ASI CAU HIS OTH
- ya factor with levels smoker nonsmoker

Source

http://pulm.bumc.bu.edu/aged/index.html

References


http://www.pnas.org/cgi/citemap?id=pnas;101/27/10143
Examples

```r
# compute mean expression for each gene
data(Spira)
attach(Spira)
apply(X, 2, mean)
# list names of covariates
names(Patients)
```

Description

Training and test expression matrices of size $G$-by-$n$ where $G$ is the number of genes and $n$ is the number of samples are generated as well as vectors of length $n$ for the two output classes.

Usage

```r
synma(n = c(100, 100), nt = c(500, 500), rho = c(0.9, 0.9), B = c(20, 20), m = c(100, 100), fE = 0.02, E = 0.02, Si = 5)
```

Arguments

- `n`: vector length 2, c(number healthy, number diseases), training sample
- `nt`: vector length 2, c(number healthy, number diseases), test sample
- `rho`: vector length 2, c(correlation healthy, correlation diseases)
- `B`: vector length 2, c(block size healthy, block size disease)
- `m`: vector length 2, c(# blocks healthy, # blocks disease)
- `fE`: fraction of genes expressed in 'diseased' patients
- `E`: Expression level of diseased genes
- `Si`: Variance inflation factor applied to the 'diseased' group

Details

The number of genes is determined as $G=B[1]*m[1] = B[2]*m[2]$. If these are not equal an informative error message is given.

Value

A list with four elements:

- `x`: expression matrix, training, $(B*m)$-by-$(n[1]+n[2])$, where $B=B[1]$ and $m=m[1]$
Author(s)

M. S. Islam and A. I. McLeod, email: aimcleod@uwo.ca

References


Examples

# A micro-sized microarray!
ans<-synma(n=c(3,3),nt=c(2,2),B=c(3,3),m=c(3,3),fE=0.5)
ans
Index

*Topic classification
  FitPAM, 9
*Topic classif
  ascrda, 3
  FitDLDA, 5
  FitkNN, 7
  FitRda, 10
  FitSvm, 12
  getTrainTest, 14
  khat, 16
  nnc, 17
  nnctest, 19
  synma, 21
*Topic datasets
  Spira, 20
*Topic package
  ascrda-package, 2

ascrda, 3
ascrda-package, 2

FitDLDA, 5
FitkNN, 7
FitPAM, 9
FitRda, 10
FitSvm, 12
getTrainTest, 14
khat, 16
knn, 18, 19
knn.cv, 18, 19
multinom, 16
nnc, 17, 19
nnctest, 19
pamr.train, 9
Spira, 20
synma, 21