Package 'denoiseR'

October 13, 2022

Version 1.0.2

Date 2020-02-23

Type Package

Title Regularized Low Rank Matrix Estimation

Author Julie Josse, Sylvain Sardy, Stefan Wager

Maintainer Julie Josse <julie.josserennes@gmail.com>

Imports irlba, Matrix, FactoMineR, stats

Suggests missMDA

Description Estimate a low rank matrix from noisy data using singular values thresholding and shrinking functions. Impute missing values with matrix completion. The method is described in <arXiv:1602.01206>.

License GPL (>= 2)

RoxygenNote 5.0.1

Depends R(>= 2.10)

NeedsCompilation no

Repository CRAN

Date/Publication 2020-02-26 07:10:09 UTC

R topics documented:

`denoiseR-package` *Regularized Low Rank Matrix Estimation*

Description

The methods implemented allow to recover a low-rank structure from noisy data. In addition, they may be used to estimate the underlying rank and to impute missing values.

Details

Package:	denoiseR
Type:	Package
Version:	1.0
Date:	2016-07-09
License:	GPL (>=2)

Author(s)

Julie Josse, Sylvain Sardy, Stefan Wager
Maintainer: Julie Josse <julie.josserennes@gmail.com>

References

Julie Josse, Sylvain Sardy, Stefan Wager. denoiseR a package for low rank matrix estimation.

See Also

URL: http://juliejosse.com/
http://web.stanford.edu/~swager/research.html
http://www.unige.ch/math/folks/sardy

`adashrink` *Adaptive Shrinkage*

Description

This function estimates a low-rank signal from Gaussian noisy data using the Adaptive Shrinker of the singular values. More precisely, the singular values are transformed using a function indexed by two parameters lambda and gamma as dl = dl * max(1-(lambda/dl)^gamma,0). This estimator is very flexible and adapts to the data whatever the noise regime. The parameters lambda and gamma are estimated by minimizing a Stein unbiased risk estimate (SURE) when the variance sigma^2 of the noise is known or a generalized SURE (GSURE) otherwise. A method using an universal threshold for lambda is also available. The estimator can be seen as a compromise between hard and soft thresholding. Singular value soft thresholding is a particular case of the method when gamma is equal to 1. It is possible to enforce the method to use soft-thresholding by setting gamma to 1.

Usage

```
adashrink(X, sigma = NA, method = c("GSURE", "QUT", "SURE"),
  gamma.seq = seq(1, 5, by = 0.1), nbsim = 500, method.optim = "BFGS",
  center = "TRUE", lambda0 = NA)
```

Arguments

`X`	a data frame or a matrix with numeric entries
`sigma`	integer, standard deviation of the Gaussian noise. By default sigma is estimated using the estim_sigma function with the MAD option
`method`	to select the two tunning parameters lambda and gamma. By default by minimizing GSURE
`gamma.seq`	a vector for the sequence of gamma. (not used when method is QUT). The values must be greater than 1. If gamma.seq is set to 1 then soft singular values soft thresholding is used.
`nbsim`	integer, number of replications used to calculate the universal threshold lambda when method is QUT
`method.optim`	the method used in the optim function. By default BFGS
`center`	boolean, to center the data. By default "TRUE"
`lambda0`	integer, the initial value for lambda used to optimize SURE and GSURE. By default the median of the singular values (must be in log scale)

Details

When sigma is known, lambda and gamma can be estimated by minimizing SURE. To do this, a grid for gamma is defined in gamma.seq (gammas must be greater than 1) and the SURE function is optimized on lambda using the optim function of the package stats (?optim) with the optimization method by default sets to "BFGS". The initial lambda can be modified in the argument lambda0. If gamma.seq is set to 1, then the SURE function is optimized in lambda only. A value for sigma has to be provided. When sigma is not known, it can be estimated using the function estim_sigma. An alternative which does not require to know or estimate sigma is estimate the two tuning parameters by minimizing GSURE. QUT consists in generating nbsim matrices of size n * p of Gaussian random variables with mean 0 and variance sigma^2 and computing the first singular value on each matrix. Then, the universal threshold lambda is calculated as the 1-alpha quantile of the null

distribution (alpha is here sqrt(log(max(n,p)))). Then, gamma is estimated by minimizing a 1-dim SURE. This method is recommended when one is particularly interested in estimating the rank of the signal. The estimated low rank matrix is given in the output mu.hat. adashrink automatically estimates the rank of the signal. Its value is given in the output nb.eigen corresponding to the number of non-zero eigenvalues.

Value

mu.hat the estimator of the signal

nb.eigen the number of non-zero singular values

gamma the optimal gamma selected by minimizing SURE or GSURE

lambda the optimal lambda selected by minimizing SURE or GSURE

singval the singular values of the estimator

low.rank the results of the SVD of the estimator

References

Josse, J. & Sardy, S. (2015). Adaptive shrinkage of singular values. Statistics and Computing.

Candes, E. J., Sing-Long C. A. and Trzasko, J. D (2012). Unbiased risk estimates for singular value thresholding and spectral estimators. IEEE Transactions on Signal Processing 61(19), 4643-4657.

See Also

estim_sigma

LRsim

Examples

```
Xsim <- LRsim(200, 500, 100, 1)
## Not run: ada.gsure <- adashrink(Xsim$X, method = "GSURE")
ada.gsure$nb.eigen
ada.gsure$singval
ada.gsure$lambda
ada.gsure$gamma

Xsim <- LRsim(200, 500, 10, 4)
sig <- estim_sigma(Xsim$X)
ada.sure <- adashrink(Xsim$X, method = "SURE", sigma = sig)
soft.sure <- adashrink(Xsim$X, gamma.seq  = 1, method = "SURE", sigma = sig)
## End(Not run)
```

estim_delta *Estimates delta for Iterated Stable Autoencoder*

Description

This function uses cross-validation to estimate delta for the Iterated Stable Autoencoder when considering Binomial noise. delta is the probability of deletion of each cell of the data matrix

Usage

```
estim_delta(X, delta = seq(0.1, 0.9, length.out = 9), nbsim = 10,
  noise = "Binomial", transformation = c("None", "CA"), pNA = 0.1,
  maxiter = 1000, threshold = 1e-08)
```

Arguments

X	a data frame or a matrix with count
delta	vector, a sequence of values for the probability of deletion of each cell of the data matrix
nbsim	number of times that pNA values are inserted and predicted in the data
noise	noise model assumed for the data. By default and only available "Binomial"
transformation	estimates a transformation of the original matrix; currently, only correspondence analysis CA is available
pNA	percentage of missing values added in the data set
maxiter	integer, maximum number of iterations of the iterative imputation algorithm
threshold	for assessing convergence of the iterative imputation algorithm (difference between two successive iterations)

Details

For each value delta, repeated learning cross-validation consists in inserting pNA percentage of missing values in the data set and predicting them with the Iterative Stable Autoencoder. More precisely, the prediction is obtained using the iterative imputation algorithm (imputecount) which alternates steps of imputation of the missing entries and estimation of the low-rank signal. This process is repeated nbsim times for all the deltas. The mean squared error of prediction is kept for each simulation and value of delta. The value of delta leading to the smallest MSEP on average over the simulations is given.

Value

msep, matrix with the MSEP obtained for each simulation and each value of delta

delta, value giving in average the smallest MSEP over the nbsim simulations

See Also

imputecount

ISA

Examples

```
# A regularized Correspondence Analysis
## Not run: library(FactoMineR)
perfume <-  read.table("http://factominer.free.fr/docs/perfume.txt",header=TRUE,
sep="\t",row.names=1)
rownames(perfume)[4] <- "Cinema"

isa.delt <- estim_delta(perfume, nbsim = 10, transformation = "CA")

isa.ca <- ISA(perfume, delta = isa.delt$delta, noise = "Binomial", transformation = "CA")
rownames(isa.ca$mu.hat) <- rownames(perfume)
colnames(isa.ca$mu.hat) <- colnames(perfume)
res.isa.ca <- CA(isa.ca$mu.hat, graph = FALSE)
plot(res.isa.ca, title = "Regularized CA", cex = 0.6, selectCol = "contrib 20")
## End(Not run)
```

`estim_sigma` *Estimate sigma*

Description

This function estimates the standard deviation sigma of the noise of the model where the data are generated from a signal of rank k corrupted by homoscedastic Gaussian noise. Two estimators are implemented. The first one, named LN, is asymptotically unbiased for sigma in the asymptotic framework where both the number of rows and the number of columns are fixed while the noise variance tends to zero (Low Noise). It is calculated by computing the residuals sum of squares (using the truncated SVD at order k as an estimator) divided by the number of data minus the number of estimated parameters. Thus, it requires as an input the rank k. The second one, MAD (mean absolute deviation) is a robust estimator defined as the ratio of the median of the singular values of X over the square root of the median of the Marcenko-Pastur distribution. It can be useful when the signal can be considered of low-rank (the rank is very small in comparison to the matrix size).

Usage

```
estim_sigma(X, k = NA, method = c("LN", "MAD"), center = "TRUE")
```

Arguments

X	a data frame or a matrix with numeric entries
k	integer specifying the rank of the signal only if method = "LN". By default k is estimated using the estim_ncp function of the FactoMineR package

method	LN for the low noise asymptotic estimate (it requires to specify the rank k) or MAD for mean absolute deviation
center	boolean, to center the data. By default "TRUE".

Details

In the low noise (LN) asymptotic framework, the estimator requires providing the rank k. Different methods are available in the litterature and if by default the user does not provide any value, we use of the function estim_ncp of the FactoMineR package with the option GCV (see ?estim_ncp).

Value

sigma the estimated value

References

Josse, J & Husson, F. (2012). Selecting the number of components in principal component analysis using cross-validation approximations. Computational Statistics & Data Analysis, 6 (56).

Gavish, M & Donoho, D. L. Optimal Shrinkage of Singular Values.

Gavish, M & Donoho, D. L. (2014). The Optimal Hard Threshold for Singular Values is 4/sqrt(3). IEEE Transactions on Information Theory, 60 (8), 5040-5053.

Josse, J. & Husson, F. (2011). Selecting the number of components in PCA using cross-validation approximations.Computational Statistics and Data Analysis. 56 (6), pp. 1869-1879.

See Also

estim_ncp

LRsim

Examples

```
Xsim <-  LRsim(100, 30, 2, 4)
res.sig <- estim_sigma(Xsim$X, k = 2)
```

---

impactfactor *Data set on metrics for scientific journals:*

Description

A subset of 443 journals of the sections Computer Science Software, Decision Sciences Statistics, Probability and Uncertainty and Mathematics Statistics and Probability and their scores for 3 metrics recorded each year from 1999 to 2013: IPP impact per publication, SNIP source normalized impact per paper (tries to weight by the number of citations per subject fieeld to adjust for different citation cultures) and the SJR SCImago journal rank (tries to capture average prestige per publication). This data contains 31 percent of missing values.

Usage

```
data(impactfactor)
```

Format

A data frame with 443 observations and 45 continuous variables

Source

journalmetrics.com

Examples

```
data(impactfactor)
## Not run: ada.NA <- imputeada(impactfactor, lambda = 4.46, gamma = 1.9)
impactfactorcomp <- ada.NA$completeObs
## End(Not run)
```

---

imputeada *Adaptive Shrinkage with missing values - Imputation*

Description

This function estimates a low-rank signal from a noisy Gaussian incomplete data using the iterative Adaptive Trace Norm (ATN) algorithm. It can be used to impute a data set. dl = dl * max(1-(lambda/dl)^gamma,0). If, the parameters lambda and gamma are not specified, they are estimated by minimizing a Missing Stein unbiased risk estimate (SURE) when the variance sigma^2 of the noise is known or a generalized SURE (GSURE) otherwise. These SURE and GSURE for missing values are implemented using finite differences.

Usage

```
imputeada(X, lambda = NA, gamma = NA, sigma = NA, method = c("GSURE",
  "SURE"), gamma.seq = seq(1, 5, by = 0.1), method.optim = "BFGS",
  center = "TRUE", scale = "FALSE", threshold = 1e-08, nb.init = 1,
  maxiter = 1000, lambda0 = NA)
```

Arguments

X	a data frame or a matrix with numeric entries
lambda	integer, value to be used in the iterative ATN algorithm
gamma	integer, value to be used in the iterative ATN algorithm
sigma	integer, standard deviation of the Gaussian noise.
method	to select the two tunning parameters lambda and gamma. By default by minimizing GSURE
gamma.seq	a vector for the sequence of gamma. The values must be greater than 1

method.optim	the method used in the optim function. By default BFGS
center	boolean, to center the data. By default "TRUE"
scale	boolean, to scale the data. By default "FALSE"
threshold,	for assessing convergence (difference between two successive iterations)
nb.init	integer, to run the iterative ATN algorithm with nbinit different initialization. By default 1.
maxiter	integer, maximum number of iterations of the iterative imputation algorithm
lambda0	integer, the initial value for lambda used to optimize SURE and GSURE. By default the median of the singular values (must be in log scale)

Details

The iterative ATN algorithm first consists in imputing missing values with initial values. Then, adashrink is performed on the completed dataset with its regularization parameter lambda and gamma. The missing entries are imputed with the estimated signal. These steps of estimation of the signal via adashrink and imputation of the missing values are iterated until convergence. At the end, both an estimation of the signal and a completed data set are provided. If lambda and gamma are not known, they can be estimated by minimizing SURE when sigma^2 is known. To do this, a grid for gamma is defined in gamma.seq (gammas must be greater than 1) and the Missing SURE function is optimized on lambda using the optim function of the package stats (?optim) with the optimization method by default sets to "BFGS". The initial lambda can be modified in the argument lambda0. When sigma is not known, it is possible to estimate the two tuning parameters by minimizing Missing GSURE. Note that Missing SURE is defined using finite differences so it is computationally costly. The estimated low rank matrix is given in the output mu.hat. imputeada automatically estimates the rank of the signal. Its value is given in the output nb.eigen corresponding to the number of non-zero eigenvalues.

Value

mu.hat the estimator of the signal

completeObs the completed data set. Observed values are the same but missing values are replaced by the estimated one in mu.hat

nb.eigen the number of non-zero singular values

gamma the given gamma or the optimal gamma selected by minimizing SURE or GSURE

lambda the given lambda or the optimal lambda selected by minimizing SURE or GSURE

singval the singular values of the estimator

low.rank the results of the SVD of the estimator

See Also

adashrink

LRsim

Examples

```
don.NA <- LRsim(200, 500, 100, 4)$X
don.NA[sample(1:(200*500),20, replace = FALSE)] <- NA
## Not run: adaNA <- imputeada(don.NA, lambda = 0.022, gamma = 2.3)
esti <- adaNA$mu.hat
comp <- adaNA$completeObs
## End(Not run)
```

`imputecount` *Imputation of count data with the Iterated Stable Autoencoder*

Description

This function estimates a low-rank signal from a noisy count incomplete data using the Iterated Stable Autoencoder. It can be used to impute a data set.

Usage

```
imputecount(X, threshold = 1e-08, maxiter = 1000, delta = 0.5,
  transformation = c("None", "CA"))
```

Arguments

`X`	a data frame or a matrix with count data containing missing values
`threshold`	for assessing convergence (difference between two successive iterations)
`maxiter`	integer, maximum number of iterations of the iterative imputation algorithm
`delta`	numeric, probability of deletion of each cell of the data matrix. By default delta = 0.5
`transformation`	estimate a transformation of the original matrix; currently, only correspondence analysis CA is available

Details

Impute the missing entries of a count data set using the iterative ISA algorithm. The iterative ISA algorithm first consists in imputing missing values with initial values. Then, ISA is performed on the completed dataset with its regularization parameter delta. The missing entries are imputed with the estimated signal. These steps of estimation of the signal via ISA and imputation of the missing values are iterated until convergence.

Value

mu.hat the estimator of the signal

completeObs the completed data set. The observed values are kept for the non-missing entries and the missing values are replaced by the predicted ones

See Also

ISA

Examples

```
#
```

ISA *Iterated Stable Autoencoder*

Description

This function estimates a low-rank signal from noisy data using the Iterated Stable Autoencoder. More precisely, it transforms a noise model into a regularization scheme using a parametric bootstrap. In the Gaussian noise model, the procedure is equivalent to shrinking the singular values of the data matrix (a non linear transformation of the singular values is applied) whereas it gives other estimators with rotated singular vectors outside the Gaussian framework. Within the framework of a Binomial or Poisson noise model, it is also possible to find the low-rank approximation of a transformed version of the data matrix for instance such as the one used in Correspondence Analysis.

Usage

```
ISA(X, sigma = NA, delta = NA, noise = c("Gaussian", "Binomial"),
  transformation = c("None", "CA"), svd.cutoff = 0.001, maxiter = 1000,
  threshold = 1e-06, nu = min(nrow(X), ncol(X)), svdmethod = c("svd",
  "irlba"), center = TRUE)
```

Arguments

X	a data frame or a matrix with numeric entries
sigma	numeric, standard deviation of the Gaussian noise. By default sigma is estimated using the estim_sigma function with the MAD option
delta	numeric, probability of deletion of each cell of the data matrix when considering Binomial noise. By default delta = 0.5
noise	noise model assumed for the data. By default "Gaussian"
transformation	estimate a transformation of the original matrix; currently, only correspondence analysis is available
svd.cutoff	singular values smaller than this are treated as numerical error
maxiter	integer, maximum number of iterations of ISA
threshold	for assessing convergence (difference between two successive iterations)
nu	integer, number of singular values to be computed - may be useful for very large matrices
svdmethod	svd by default. irlba can be specified to use a fast svd method. It can be useful to deal with large matrix. In this case, nu may be specified
center	boolean, to center the data for the Gaussian noise model. By default "TRUE"

Details

When the data are continuous and assumed to be drawn from a Gaussian distribution with expectation of low-rank and variance sigma^2, then ISA performs a regularized SVD by corrupting the data with an homoscedastic Gaussian noise (default choice) with variance sigma^2. A value for sigma has to be provided. When sigma is not known, it can be estimated using the function estim_sigma.

For count data, the subsampling scheme used to draw X can be considered as Binomial or Poisson (equivalent to Binomial, delta = 0.5). ISA regularizes the data by corrupting the data with Poisson noise or by drawing from a Binomial distribution of parameters X_ij and 1-delta divided by 1-delta. Thus it is necessary to give a value for delta. When, the data are transformed with Correspondence Analysis (transfo = "CA"), this latter noising scheme is also applied but on the data transformed with the CA weights. The estimated low rank matrix is given in the output mu.hat. ISA automatically estimates the rank of the signal. Its value is given in the output nb.eigen corresponding to the number of non-zero eigenvalues.

Value

mu.hat the estimator of the signal

nb.eigen the number of non-zero singular values

low.rank the results of the SVD of the estimator; for correspondence analysis, returns the SVD of the CA transform

nb.iter number of iterations taken by the ISA algorithm

References

Josse, J. & Wager, S. (2016). Bootstrap-Based Regularization for Low-Rank Matrix Estimation. Journal of Machine Learning Research.

See Also

estim_sigma

LRsim

Examples

```
Xsim <- LRsim(200, 500, 10, 4)
isa.gauss <- ISA(Xsim$X, sigma = 1/(4*sqrt(200*500)))
isa.gauss$nb.eigen

# isa.bin <- ISA(X, delta = 0.7, noise = "Binomial")

# A regularized Correspondence Analysis
## Not run: library(FactoMineR)
 perfume <-  read.table("http://factominer.free.fr/docs/perfume.txt",
 header=TRUE,sep="\t",row.names=1)
 rownames(perfume)[4] <- "Cinema"
 isa.ca <- ISA(perfume, delta = 0.5, noise = "Binomial", transformation = "CA")
 rownames(isa.ca$mu.hat) <- rownames(perfume)
 colnames(isa.ca$mu.hat) <- colnames(perfume)
```

```
 res.isa.ca <- CA(isa.ca$mu.hat, graph = FALSE)
 plot(res.isa.ca, title = "Regularized CA", cex = 0.6, selectCol = "contrib 20")
 res.ca <- CA(perfume, graph = FALSE)
 plot(res.ca, title = "CA", cex = 0.6, selectCol = "contrib 20")
## End(Not run)
```

`LRsim` *Low Rank Simulation*

Description

This function simulates a data set as a low-rank signal corrupted by Gaussian noise.

Usage

```
LRsim(n, p, k, SNR)
```

Arguments

`n`	integer, number of rows
`p`	integer, number of columns
`k`	integer, rank of the signal
`SNR`	numeric, signal to noise ratio

Details

A data set of size n*p and of rank k is simulated. More precisely, it is simulated as follows: A SVD is performed on a n*p matrix generated from a standard multivariate normal distribution. Then, the signal is computed using the first k singular vectors and singular values U_q D_q V_q'. The signal is scaled in such a way that the variance of each column is 1 and then a Gaussian noise with variance sigma^2 is added. The SNR is calculated as 1/ sigma sqrt(np).

Value

X the simulated data

mu the true signal

sigma the standard deviation of the noise added to the signal

Examples

```
Xsim <- LRsim(100, 30, 2, 2)
```

`optishrink` *Optimal Shrinkage*

Description

This function estimates a low-rank signal from Gaussian noisy data using the Optimal Shrinker of the singular values. More precisely, in an asymptotic framework, the estimator which applies a non-linear transformation of the singular values is the closest to the underlying signal in term of mean squared error. Two asymptotic frameworks are considered: one where both the number of rows and the number of columns are fixed while the noise variance tends to zero (Low Noise) and one where both the number of rows and of columns tend to infinity (ASYMPT) while the rank of the matrix stays fixed. In this latter, an optimal shrinker is given according to different norm losses (Frobenius, Operator, Nuclear).

Usage

```
optishrink(X, sigma = NA, center = "TRUE", method = c("ASYMPT", "LN"),
  loss = c("Frobenius", "Operator", "Nuclear"), k = NA)
```

Arguments

`X`	a data frame or a matrix with numeric entries
`sigma`	integer, standard deviation of the Gaussian noise. By default sigma is estimated using the estim_sigma function
`center`	boolean, to center the data. By default "TRUE"
`method`	asymptotic framework used either low noise LN or ASYMPT. By default ASYMPT
`loss`	by default Frobenius only if method = "ASYMPT"
`k`	integer, specifying the rank of the signal only if method = "LN". By default k is estimated using the estim_ncp function of the FactoMineR package

Details

In the low noise (LN) asymptotic framework, the estimator applies the following transformation on the first k singular values dl = dl *(dl^2-sigma^2)/dl^2. Thus, it requires providing both the rank k and a value for sigma. Concerning the rank k, different methods are available in the litterature and if by default the user does not provide any value, we use of the function estim_ncp of the FactoMineR package with the option GCV (see ?estim_ncp). The other asymptotic framework (ASYMPT) only requires providing sigma. optishrink automatically estimates the rank of the signal. Its value is given in the output nb.eigen corresponding to the number of non-zero eigenvalues. The estimated low rank matrix is given in the output mu.hat.

Value

mu.hat the estimator of the signal

nb.eigen the number of non-zero singular values

singval the singular values of the estimator

low.rank the results of the SVD of the estimator

References

Gavish, M & Donoho, D. L. (2014). Optimal Shrinkage of Singular Values.

Verbanck, M., Husson, F. & Josse, J. (2015). Regularised PCA to denoise and visualise data. Statistics & Computing. 25 (2), 471-486

See Also

estim_sigma

LRsim

Examples

```
Xsim <- LRsim(200, 500, 10, 2)
opti.ln <- optishrink(Xsim$X, method = "LN", k = 10)
opti.asympt <- optishrink(Xsim$X, method = "ASYMPT")

Xsim <- LRsim(200, 500, 100, 1)
truesigma <- 1/(1*sqrt(200*500))
opti.asympt <- optishrink(Xsim$X, method = "ASYMPT", sigma = truesigma)
opti.asympt$nb.eigen
```

Presidents *Contingency table with US Presidents speeches.*

Description

A data set on US presidents inaugural speeches.

Usage

```
data(Presidents)
```

Format

A data frame with 13 rows and 836 columns. Rows represents the US presidents (from 1940 to 2009) and columns words used during their inaugural addresses. This is a contingency table.

Source

http://www.presidency.ucsb.edu and http://www.usa-presidents.info/union/ DtmVic software (Lebart 2015) http://www.dtmvic.com/

Examples

```
## Not run:
data(Presidents)
isa.ca <- ISA(Presidents, delta = 0.1, transformation = "CA")
rownames(isa.ca$mu.hat) <- rownames(Presidents)
colnames(isa.ca$mu.hat) <- colnames(Presidents)
res.isa.ca <- CA(as.data.frame(isa.ca$mu.hat), graph = FALSE)
plot(res.isa.ca, title = "Regularized CA", cex = 0.8, selectRow = "contrib 40")
plot(res.isa.ca, title = "Regularized CA", cex = 0.6, invisible = "row" )

## End(Not run)
```

---

tumors *Brain tumors data.*

Description

43 brain tumors and 356 continuous variables corresponding to the expression data and 1 categorical variable corresponding to the type of tumors (4 types).

Usage

```
data(tumors)
```

Format

A data frame with 43 rows and 357 columns. Rows represent the tumors, columns represent the expression and the type of tumor.

Details

A genetic data frame.

Source

M. de Tayrac, S. Le, M. Aubry, J. Mosser, and F. Husson. Simultaneous analysis of distinct omics data sets with integration of biological knowledge: Multiple factor analysis approach. BMC Genomics, 10(1):32, 2009.

Examples

```
data(tumors)
## Not run:
res.ada <- adashrink(tumors[, -ncol(tumors)], method = "SURE")
res.hcpc <- HCPC(as.data.frame(res.ada$mu.hat), graph=F, consol = FALSE)
plot.HCPC(res.hcpc, choice = "map", draw.tree = "FALSE")

## End(Not run)
```

Index