Package 'macat'

April 14, 2017

Title MicroArray Chromosome Analysis Tool

Version 1.48.0

Date 2014-01-15

Author Benjamin Georgi, Matthias Heinig, Stefan Roepcke, Sebastian Schmeier, Joern Toedling

Depends Biobase, annotate

Suggests hgu95av2.db, stjudem

Description This library contains functions to investigate links between differential gene expression and the chromosomal localization of the genes. MACAT is motivated by the common observation of phenomena involving large chromosomal regions in tumor cells. MACAT is the implementation of a statistical approach for identifying significantly differentially expressed chromosome regions. The functions have been tested on a publicly available data set about acute lymphoblastic leukemia (Yeoh et al.Cancer Cell 2002), which is provided in the library 'stjudem'.

Maintainer Joern Toedling <jtoedling@yahoo.de>

License Artistic-2.0

biocViews Microarray, DifferentialExpression, Visualization

Reference Toedling J, Schmeier S, Heinig M, Georgi B, Roepcke S (2005). MACAT - MicroArray Chromosome Analysis Tool. Bioinformatics. 21(9):2112--2113.

NeedsCompilation no

R topics documented:

buildMACAT 2
compute.sliding 3
discreteKernelize 4
discretize 5
discretize.tscores 6
evalScoring 7
evaluateParameters 9
getResults 11
kernelize 12

plot.MACATevalScoring 13
preprocessedLoader 14
scoring 15
stjd 17

Index **18**

`buildMACAT` *Create MACAT list from objects in workspace*

Description

This is a wrapper around the preprocessedLoader function. Use it, when you want to build a MACAT-list structure from objects already in your workspace.

Usage

```
buildMACAT(matrix, chip, labels = NULL, chromLocObj = NULL)
```

Arguments

`matrix`	expression matrix with rows=genes and columns=samples; Rownames have to match chip; Columnnames are not mandatory.
`chip`	Identifier for used microarray
`labels`	Classlabels for samples, has to have length=number of columns in matrix
`chromLocObj`	Object of class `chromLocation` specifying the genomic position, each probe on the array is mapped to. If not provided, it is build in the function using `annotate`'s function `buildChromLocation`.

Details

This is only a convenience wrapper around the function `preprocessedLoader` for the case, that you want to build a MACAT-list from objects in your workspace.

Value

A MACAT-list structure. For an example and a description of the format see data `stjude` in package 'stjudem'.

Author(s)

MACAT development team

See Also

`preprocessedLoader`,`stjude` in package 'stjudem'

Examples

```
 X <- matrix(rnorm(200),nrow=20,ncol=10)
 rownames(X) <- c('34916_s_at','34917_at','34462_at','163_at','35219_at',
'31641_s_at','33300_at','33301_g_at','38950_r_at','41249_at',
'294_s_at','32004_s_at','33299_at','41243_at','33341_at','362_at',
'1918_at','41499_at','41500_at','41282_s_at')
 colnames(X) <- paste("Sample",1:10,sep="")
 y <- rep(c("A","B"),c(5,5))
 toy <- buildMACAT(X,"hgu95av2.db",y)
 summary(toy)
```

`compute.sliding` *Compute and plot smoothing of expression values or scores along the chromosome*

Description

'compute.sliding' computes a smoothing of the expression data or scores along the chromosome using the specified kernel function. This function is also used within the 'evalScoring' function. 'plotSliding' creates a plot of the smoothed expression values / scores.

Usage

```
compute.sliding(data, chromosome, sample, kernel, kernelparams=NULL, step.width = 1e+06)
plotSliding(data, chromosome, sample, kernel, kernelparams=NULL,
step.width=1000000, ...)
```

Arguments

`data`	A MACATData list holding the Expression values and gene locations
`chromosome`	the chromosome to be smoothed
`sample`	the sample (patient) whose expression values are smoothed
`kernel`	a kernel function (one of rbf, kNN, basePairDistance or your own)
`kernelparams`	a list of named parameters for the kernel (by default estimated from the data)
`step.width`	the smoothing is computed stepwise every step.width basepairs (default is 100000)
`...`	further graphical parameters passed on to `plot.default`

Value

for `compute.sliding`: a matrix of dimension (steps x 2) with in the first column the locations in basepairs where an interpolation is computed, and in the second column the smoothed values. `plotSliding` does not return anything and is merely called for its side-effect producing the plot.

Author(s)

MACAT development team

See Also

`kernelize`, `evalScoring`

Examples

```
data(stjd)
# just compute smoothed values:
smooth = compute.sliding(stjd, chromosome=3, sample=6, rbf,
                         kernelparams=list(gamma=1/10^13))
# compute and plot smoothed values:
plotSliding(stjd, chromosome=3, sample=6,rbf,
            kernelparams=list(gamma=1/10^13),pch=20,
            main="Chromosome 3")
```

`discreteKernelize` *Discretize and smooth expression values*

Description

returns discretized kernelized expression values and saves them to a file if argument 'saveToFile' is TRUE. For details on discretization see `discretize`.

Usage

```
discreteKernelize(data, chrom, margin = 10, step.width = 1e+05, kernel = rbf,
                  kernelparams = list(gamma = 1/10^13), saveToFile = FALSE)
```

Arguments

`data`	MACATData Object
`chrom`	chromosome to kernelize
`margin`	symmetric qunatile in percent
`step.width`	size of the interpolation steps
`kernel`	kernel function one of rbf, kNN, basePairDistance or your own
`kernelparams`	list of named kernel parameters
`saveToFile`	logicval indicating whether to write a flatfile or not; default is FALSE

Details

Filename of the flatfile is: `discrete_kernelized_seq_margin_<margin>_chrom_<chrom>.py` where <margin> is the discretization parameter and <chrom> the name of the chromosome.

Value

discretized and kernelized expression matrix

Author(s)

The MACAT Development team

See Also

`pydata`, `kernelizeAll`

Examples

```
#loaddatapkg("stjudem")
#data(stjude)
data(stjd)
discretizedKernelized = discreteKernelize(stjd, 13)
```

`discretize` *Discretize expression values*

Description

'discretize' returns the discretized expression data for all chromosomes in chrom and all samples that have a label listed in label. Discretization is performed by comparing the value gene-wise (location-wise) with the symmetric upper and lower quantile given by margin (in percent margin/2 lower and upper quantile).

Usage

```
discretize(data, chrom, label, margin = 10)
discretizeChromosome(data, chrom, margin=10)
discretizeOne(data, chrom, sample, margin=10)
```

Arguments

`data`	MACATData object
`chrom`	list of chromosomes
`label`	list of labels
`margin`	symmetric quantile in percent
`sample`	the sample for which you want discretized expression data

Value

returns a discretized expression matrix for all genes on the chromosomes in 'chrom' and all samples that have a label in 'label'.

Author(s)

MACAT development team

See Also

`discretizeAll`

`discretize.tscores` *Discretize regularized t-scores*

Description

discretize.tscores returns a discretized version of the scores in the MACATevalScoring object. Discretization is performed by comparing the value gene-wise (location-wise) with the symmetric upper and lower quantile given by margin (in percent margin/2 lower and upper quantile). discretizeAllClasses produces a flatfile readable by PYTHON.

Usage

```
discretize.tscores(scores)
discretizeAllClasses.tscores(data, chrom, nperms=10, kernel=rbf, kernelparams=NULL, step.width=10
```

Arguments

`scores`	a MACATevalScoring object obtained from evalScoring
`data`	a MACATData Object containing all expression values, geneLocations and labels (obtained from preprocessedLoader)
`chrom`	chromosome that is discretized
`nperms`	number of permutations for the computation of empirical p values (evalScoring)
`kernel`	kernel function used for smoothing one of rbf, kNN, basePairDistance or your own
`kernelparams`	list of parameters for the kernels
`step.width`	size of a interpolation step in basepairs

Details

The filename for the python flat files are `discrete_chrom_<chrom>_class_<label>.py` where <chrom> and <label> are the names of the chromosome and class label.

Value

`discretize.tscores`
: a vector of discretized tscores

`discretizeAllClasses.tscores`
: creates python flatfiles (see details)

Author(s)

The MACAT development team

See Also

`evalScoring`, `kernels`, `pythondata`

Examples

```
#loaddatapkg("stjudem")
#data(stjude)
data(stjd)
# simple scoring with short running time
scores = evalScoring(stjd, "T", 1, nperms=100, cross.validate=FALSE)
discrete = discretize.tscores(scores)
```

`evalScoring` *Score differential expression, assess significance, and smooth scores along the chromosome*

Description

This function computes for all genes on one chromosome the regularized t-statistic to score differential gene expression for two given groups of samples. Additionally these scores are computed for a number of permutations to assess significance. Afterwards these scores are smoothed with a given kernel along the chromosome to give scores for chromosomal regions.

Usage

```
evalScoring(data, class, chromosome, nperms=1000, permute="labels",
     pcompute="empirical", subset=NULL,
     newlabels=NULL,kernel=rbf,kernelparams=NULL,cross.validate=TRUE,
     paramMultipliers=2^(-4:4),ncross=10,step.width=100000,
     memory.limit=TRUE, verbose=TRUE)
```

Arguments

`data`	Gene expression data in the MACAT list format. See data(stjude) for an example.
`class`	Which of the given class labels is to be analyzed
`chromosome`	Chromosome to be analyzed
`nperms`	Number of permutations
`permute`	Method to do permutations. Default 'labels' does permutations of the class labels, which is the common and faster way to assess significance of differential expression. The altenative 'locations' does permutations of gene locations, is much slower and right now should be considered preliminary at best.
`pcompute`	Method to determine the p-value for differential expression of each gene. Is only evaluated if the argument `permute='labels'` and in that case passed on to the function `scoring`
`subset`	If a subset of samples is to be used, give vector of column- indices of these samples in the original matrix here.
`newlabels`	If other labels than the ones in the MACAT-list-structure are to be used, give them as character vector/factor here. Make sure argument 'class' is one of them.
`kernel`	Choose kernel to smooth scores along the chromose. Available are 'kNN' for k-Nearest-Neighbors, 'rbf' for radial-basis-function (Gaussian), 'basePairDistance' for a kernel, which averages over all genes within a given range of base pairs around a position.

kernelparams	Additional parameters for the kernel as list, e.g., kernelparams=list(k=5) for taking the 5 nearest neighbours in the kNN-kernel. If NULL some defaults are set within the function.
cross.validate	Logical. Should the paramter settings for the kernel function be optimized by a cross-validation?
paramMultipliers	Numeric vector. If you do cross-validation of the kernel parameters, specify the multipliers of the given (standard) parameters to search over for the optimal one.
ncross	Integer. If you do cross-validation, specify how many folds.
step.width	Defines the resolution of smoothed scores on the chromosome, is in fact the distance in base pairs between 2 positions, for which smoothed scores are to be calculated.
memory.limit	If you have a computer with lots of RAM, setting this to FALSE will increase speed of computations.
verbose	logical; should function's progress be reported to STDOUT ?; default: TRUE.

Details

Please see the package vignette for more details on this function.

Value

List of class 'MACATevalScoring' with 11 components:

original.geneid	Gene IDs of the genes on the chosen chromosome, sorted according to their position on the chromosome
original.loc	Location of genes on chromosome in base pairs from 5'end
original.score	Regularized t-score of genes on chromosome
original.pvalue	Empirical p-value of genes on chromosome. How often was a higher score observed than this one with random permutations? In other words, how significant seems this score to be?
steps	Positions on the chromosome in bp from 5', for which smoothed scores have been computed.
sliding.value	Smoothed regularized t-scores at step-positions.
lower.permuted.border	Smoothed scores from permutations, lower significance border, currently 2.5%-quantile of permutation scores.
upper.permuted.border	Smoothed scores from permutations, upper significance border, currently 97.5%-quantile of permutation scores.
chromosome	Chromosome, which has been analyzed
class	Class, which has been analyzed
chip	Identifier for used microarray

Author(s)

MACAT development team

See Also

scoring,plot.MACATevalScoring, getResults

Examples

```
data(stjd) # load example data

# if you have the data package 'stjudem' installed,
#  you should work on the full data therein, of which
#  the provided example data, is just a piece
#loaddatapkg("stjudem")
#data(stjude)

# T-lymphocyte versus B-lymphocyte on chromosome 1,
#  smoothed with k-Nearest-Neighbours kernel(k=15),
#  few permutations for higher speed
chrom1Tknn <- evalScoring(stjd,"T",chromosome="1",permute="labels",
nperms=100,kernel=kNN,kernelparams=list(k=15),step.width=100000)

# plotting on x11:
if (interactive())
   plot(chrom1Tknn)

# plotting on HTML:
if (interactive())
   plot(chrom1Tknn,"html")
```

`evaluateParameters` *Evaluate Performance of Kernel Parameters by Cross-validation*

Description

For a given data set, chromosome, class, and kernel function, this function helps in determining optimal settings for the kernel parameter(s). The performance of individual parameter setting is assessed by cross- validation.

Usage

```
evaluateParameters(data, class, chromosome, kernel, kernelparams = NULL,
                   paramMultipliers = 2^(-4:4), subset = NULL,
                   newlabels = NULL, ncross = 10, verbose = TRUE)
```

Arguments

`data`	Gene expression data in the MACAT list format. See data(stjude) for an example.
`class`	Sample class to be analyzed
`chromosome`	Chromosome to be analyzed
`kernel`	Choose kernel to smooth scores along the chromosome. Available are 'kNN' for k-Nearest-Neighbors, 'rbf' for radial-basis-function (Gaussian), 'basePairDistance' for a kernel, which averages over all genes within a given range of base pairs around a position.

`kernelparams` Additional parameters for the kernel as list, e.g., kernelparams=list(k=5) for taking the 5 nearest neighbours in the kNN-kernel. If NULL some defaults are set within the function.

`paramMultipliers` Numeric vector. If you do cross-validation of the kernel parameters, specify these as multipliers of the given (standard) kernel parameter, depending on your kernel choice (see page 5 of the vignette). The multiplication results are the kernel argument settings, among which you want to search for the optimal one using cross-validation.

`subset` If a subset of samples is to be used, give vector of column- indices of these samples in the original matrix here.

`newlabels` If other labels than the ones in the MACAT-list-structure are to be used, give them as character vector/factor here. Make sure argument 'class' is one of them.

`ncross` Integer. Specify how many folds in cross-validation.

`verbose` Logical. Should progress be reported to STDOUT?

Value

A list of class 'MACATevP' with 4 components:

`[parameterName]` List of assessed settings for the parameter [parameterName].

`avgResid` Average Residual Sum of Squares for the parameter settings in the same order as the first component.

`multiplier` Multiplier of the original parameters in the same order as the first components.

`best` List of parameter settings considered optimal by cross- validation. Can be directly inserted under the argument 'kernelparams' of the 'evalScoring' function.

Author(s)

MACAT development team

See Also

`evalScoring`

Examples

```
data(stjd)
evalkNN6 <- evaluateParameters(stjd, class="T", chromosome=6,kernel=kNN,
                               paramMultipliers=c(0.01,seq(0.2,2.0,0.2),2.5))
if (interactive()&&capabilities("X11"))
  plot(evalkNN6)
```

getResults *Access results of 'evalScoring'*

Description

This function processes the result of the evalScoring function and returns a list of probe sets within chromosome regions deemed significant by MACAT. Additional annotation for these probe sets is provided along with their identifiers.

Usage

```
getResults(MACATevalScoringOBJ)
```

Arguments

MACATevalScoringOBJ	Object of class MACATevalScoring, usually the result from evalScoring

Details

The p-values have been computed individually for probe sets (genes), not for whole chromosome regions. Thus, regions deemed significant by sliding window approach do not have to consist only of probe sets with low p-values. These probe-set p-values are not used to determine whether a region is considered significant or not. Instead the comparison between actual and interpolated scores to actual and interpolated boundaries determines whether a region is considered significant.

This function is called within the plot function for the results of evalScoring, when HTML output is desired.

Value

A list with the following components, describing probe sets within chromosome regions deemed significant:

probeID	IDs of probe sets within these chromosome regions
cytoband	chromosomal bands these probe sets have been annotated to
geneSYM	gene symbols these probe sets have been annotated to
pvalue	p-values for probe sets; see details
locusid	EntrezGene-(formerly LocusLink) IDs of these probe sets
genedescription	Description of genes the probe sets have been annotated to
probeScore	the differential expression scores for the probe sets
chromosome	chromosome, the analysis has been done for
class	sample class, the analysis has been done for

Author(s)

MACAT development team

See Also

evalScoring

Examples

```
data(stjd)
myevalres <- evalScoring(stjd, class="T", chromosome=6, nperms=10,
                         cross.validate=FALSE)
results <- getResults(myevalres)
summary(results)
results$probeID[1:20]
```

---

kernelize *Smooth expression values or scores*

Description

'kernelize' uses a kernel to smooth the data given in geneLocations by computing a weighted sum of the values vector. The weights for each position are given in the kernelweights matrix. A kernelweights matrix can be obtained by using the kernelmatrix function.

Usage

```
getsteps(geneLocations, step.width)
kernelmatrix(steps, geneLocations, kernel, kernelparams)
kernelize(values, kernelweights)
```

Arguments

geneLocations	a list of gene locations (length n)
step.width	the width of steps in basepairs
steps	a list of locations where the kernelization shall be computed
kernel	kernel function one of rbf, kNN or basePairDistance (or your own)
kernelparams	a list of named parameters for the kernel (default is fitted to the data)
values	vector of length n or matrix (m x n) of values that are to be smoothed
kernelweights	a matrix of (n x steps) where n is the length of the values vector and steps is the number of points where you wish to interpolate

Value

getsteps	a list of locations starting at min(genLocations) going to max(geneLocations) with steps of size step.width
kernelmatrix	a matrix of (n x steps) containing the kernel weights for each location in steps
kernelize	a vector of length steps or a matrix (m x steps) containing the smoothed values

Author(s)

MACAT Development team

See Also

compute.sliding, evalScoring

Examples

```
data(stjd)
genes = seq(100)
geneLocations = abs(stjd$geneLocation[genes])
geneExpression = stjd$expr[genes,]
step.width = 100000
steps = getsteps(geneLocations, step.width)
weights = kernelmatrix(steps, geneLocations, rbf, list(gamma=1/10^13))
kernelized = kernelize(geneExpression, weights)
plot(steps, kernelized[1,])
```

---

plot.MACATevalScoring *Plot function for MACATevalScoring objects.*

Description

Function plots scores, 0.025 and 0.975 quantiles of the permuted scores (grey lines), and sliding average score (red line) along the chromosome. Yellow dots highlight regions, in which the smoothed absolute scores exceed the permutation-derived quantile boundaries.

Usage

```
## S3 method for class 'MACATevalScoring'
plot(x, output = "x11",
              HTMLfilename = NULL, mytitle = NULL, new.device = TRUE, ...)
```

Arguments

x	MACATevalScoring object.
output	plot "x11" or create a "html" -file with further information. HTML-page will open automatically.
HTMLfilename	HTML-filename, default:Results<CHOMOSOME>_<CLASS>.html
mytitle	Title of HTML-page, default: "Results of class <CLASS> on chromosome <CHROMOSOME>"
new.device	if FALSE: Possibility to plot several plots in one device
...	further arguments passed on to generic function plot

Details

One can create a HTML-page on-the-fly if argument output='html'. The HTML-page provides informations about highlighted regions in the plot. Furthermore there are click-able Entrezgene-IDs for further analysis.

Author(s)

MACAT development team

See Also

evalScoring, getResults

Examples

```
# see function 'evalScoring' for an example
```

`preprocessedLoader` *Read in data and produce MACAT list*

Description

This function reads expression data either from a saved R-file (.RData,.rda), or from a tab-separated text-file (.xls). For building a MACAT-list structure from objects in your workspace, you can either use this function or the convenience wrapper 'buildMACAT'.

Usage

```
preprocessedLoader(rdatafile, chip, labels = NULL, chromLocObj = NULL,
rdafile = TRUE, tabfile = FALSE, labelfile = FALSE)
```

Arguments

`rdatafile`	Complete name of the expression data file, or the expression matrix
`chip`	Identifier of the used microarray. To date only commercial Affymetrix microarrays are supported by MACAT
`labels`	Classlabels of the samples, vector of same length as number of columns in expression matrix; alternatively complete name of textfile with one label per line
`chromLocObj`	Object of class `chromLocation` specifying the genomic position, each probe on the array is mapped to. If not provided, it is build in the function using `annotate`'s function `buildChromLocation`.
`rdafile`	Logical; is first argument a saved R-file?
`tabfile`	Logical; is first argument a tab-separated text file?
`labelfile`	Logical; is third argument a file with one label per line?

Value

List of class 'MACATData' with 6 components:

`geneName`	Identifiers of genes/probe sets in expression data
`geneLocation`	Location of genes on their chromosome as distance from 5'end in base pairs Negative numbers denote genes on the antisense strand.
`chromosome`	Chromosome of the respective gene. Components 'geneName', 'geneLocation', and 'chromosome' are in the same order.
`expr`	expression matrix with rows = genes and columns = samples/patients
`labels`	(disease) subtype of each sample, has length = number of columns of expression matrix
`chip`	Identifier for Microarray used for the experiments

Note

At present, macat can only work with Affymetrix microarrays, for which an annotation package is installed on your system. Such annotation packages can either be obtained from the Bioconductor annotation packages repository or be constructed using the Bioconductor package AnnBuilder. For an example, see the common annotation package hgu95av2.

Author(s)

MACAT development team

See Also

buildMACAT,read.table, stjd,stjude in package 'stjudem'

Examples

```
 ## Not run:
  # assume you have your HG-U95Av2 expression values in a
  # tab-separated text file, called 'foo.txt'
  mydata <- preprocessedLoader("foo.txt","hgu95av2",rdafile=FALSE,tabfile=TRUE)

## End(Not run)
```

---

scoring *Compute (regularized) t-scores for gene expression data*

Description

This function computes for all genes in an expression matrix the (regularized) t-scores (statistics) with the given class labels and a number of permutations of these labels. Each gene is also assigned a p-value either empirically from the permutation scores or from a t-distribution.

Usage

```
scoring(data, labels, method = "SAM", pcompute = "tdist",
        nperms = 1000, memory.limit = TRUE, verbose = TRUE)
```

Arguments

data	Expression matrix with rows = genes and columns = samples
labels	Vector or factor of class labels; Scoring works only with two classes!
method	Either "SAM" to compute regularized t-scores, or "t.test" to compute Student's t-statistic
pcompute	Method to compute p-values for each genes, either "empirical" to do permutations and compute p-values from them, or "tdist" to compute p-values based on respective t-distribution
nperms	Number of permutations of the labels to be investigated, if argument 'pcompute="empirical"'
memory.limit	Logical, if you have a really good computer (>2GB RAM), setting this FALSE will increase speed of computations
verbose	Logical, if progress should be reported to STDOUT

Details

If 'pcompute="empirical"', the statistic is computed based on the given class labels, afterwards for 'nperms' permutations of the labels. The p-value for each gene is then the proportion of permutation statistics that are higher or equal than the statistic from the real labels. For each gene the 2.5%- and the 97.5%-quantile of the permutation statistics are also returned as lower and upper 'significance threshold'.

If 'pcompute="tdist", the statistic is computed only based on the given class labels, and the p-value is computed from the t-distribution with (Number of samples - 2) degrees of freedom.

Value

A list, with four components:

`observed`	(Regularized) t-scores for all genes based on the given labels
`pvalues`	P-values for all genes, either from permutations or t-distribution
`expected.lower`	2.5%-quantile of permutation test-statistics, supposed to be a lower 'significance border' for the gene; or NULL if p-values were computed from t-distribution
`expected.upper`	97.5%-quantile of permutation test-statistics, supposed to be an upper 'significance border' for the gene; or NULL if p-values were computed from t-distribution

Note

In package `macat`, this function is only called internally by the function `evalScoring`

Author(s)

MACAT development team

References

Regarding the regularized t-score please see the `macat` vignette.

See Also

`evalScoring`

Examples

```
data(stjd)
# compute gene-wise regularized t-statistics for
#  T- vs. B-lymphocyte ALL:
isT <- as.numeric(stjd$labels=="T")
TvsB <- scoring(stjd$expr,isT,method="SAM",pcompute="none")
summary(TvsB$observed)
```

stjd *Subset Microarray Data from St.Jude Children Research Hospital (USA)*

Description

Example for list-structure used by many functions in MACAT. It's based on the gene expression data published by Yeoh et al. (2002) The data has been preprocessed using 'vsn' on probe level and the probe values have been summed up to probe set values using the 'median polish' procedure. This is a subset of the data, containing only the data for the 5000 probe sets with the highest variance across the samples and for 10 exemplary samples, 5 from T-lymphocytic Acute Lymphocytic Leukemia (ALL) and 5 from B-lymphocytic ALL.

Usage

```
data(stjd)
```

Format

List of class 'MACATData' with 6 components:

geneName: Identifiers of genes/probe sets in expression data

geneLocation: Location of genes on their chromosome as distance from 5'end in base pairs Negative numbers denote genes on the antisense strand.

chromosome: Chromosome of the respective gene. Components 'geneName', 'geneLocation', and 'chromosome' are in the same order.

expr: expression matrix with rows = genes and columns = samples/patients

labels: (disease) subtype of each sample, has length = number of columns of expression matrix

chip: Identifier for Microarray used for the experiments (here for the Affymetrix HG-U95av2 Oligonucleotide GeneChip)

Note

For the full data package see the Bioconductor data package `stjudem`. If it is not already installed on your system, try `source("http:\www.bioconductor.org\biocLite.R"); biocLite("stjudem")`

References

Yeoh et al. Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling. Cancer Cell. March 2002. 1: 133-143.

See Also

`buildMACAT`,`stjude` in package 'stjudem' for the complete expression data

Examples

```
data(stjd)
summary(stjd)
```

Index

∗Topic **datasets**
stjd, 17
∗Topic **file**
preprocessedLoader, 14
∗Topic **hplot**
plot.MACATevalScoring, 13
∗Topic **manip**
buildMACAT, 2
compute.sliding, 3
discreteKernelize, 4
discretize, 5
discretize.tscores, 6
evalScoring, 7
evaluateParameters, 9
getResults, 11
kernelize, 12
scoring, 15

buildMACAT, 2, *15*, *17*

compute.sliding, 3, *13*

discreteKernelize, 4
discretize, *4*, 5
discretize.tscores, 6
discretizeAll, *5*
discretizeAllClasses.tscores *(*discretize.tscores*)*, 6
discretizeChromosome *(*discretize*)*, 5
discretizeOne *(*discretize*)*, 5

evalScoring, *3*, *6*, 7, *10*, *12–14*, *16*
evaluateParameters, 9

getResults, *9*, 11, *14*
getsteps *(*kernelize*)*, 12

kernelize, *3*, 12
kernelizeAll, *4*
kernelmatrix *(*kernelize*)*, 12
kernels, *6*

Loader *(*preprocessedLoader*)*, 14

MACATData *(*preprocessedLoader*)*, 14
MACATinput *(*preprocessedLoader*)*, 14

plot.default, *3*
plot.MACATevalScoring, *9*, 13
plot.MACATevP *(*evaluateParameters*)*, 9
plotSliding *(*compute.sliding*)*, 3
preprocessedLoader, *2*, 14
pydata, *4*
pythondata, *6*

read.table, *15*

scoring, *9*, 15
stjd, *15*, 17