Package 'geneSignatureFinder'

February 19, 2015

Type Package

Title A Gene-signatures finder tools

Version 2014.02.17

Date 2012-08-20

Author Stefano M. Pagnotta, Michele Ceccarelli

Maintainer Stefano M. Pagnotta <pagnotta@unisannio.it>

Description A tool for finding an ensemble gene-signature by a steepest ascending algorithm partially supervised by survival time data.

License GPL-2

Depends R (>= 3.0.0), survival, cluster, class, parallel

Suggests

URL http://www.bioinformatics.unisannio.it/gsf

BugReports http://www.bioinformatics.unisannio.it/gsf

NeedsCompilation no

Repository CRAN

Date/Publication 2014-03-07 11:58:47

R topics documented:

`BHcorrection` *Benjamini & Hochberg (1995) method for p-values correction*

Description

This function evaluate the p-values correction according to the Benjamini & Hochberg (1995) method.

Usage

```
BHcorrection(pvs, alpha = 0.05)
```

Arguments

`pvs`	list of p-values
`alpha`	level of significance of the test

Value

list of corrected p-values

Author(s)

Stefano M. Pagnotta and Michele Ceccarelli

References

Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. J. Roy. Statist. Soc. Ser. B 57 289-300.

Examples

```
###################
data <- matrix(rnorm(50000), 10000, 5)
pvs <- apply(data, 1, function(xx) t.test(xx, alternative = "less")$p.value)
qvs <- BHcorrection(pvs)
```

BICs *Compute the bayesian information criteria*

Description

This function computes the bayesian information criteria (bic) under two hypotheses: 1) the data are drawn from a single gaussian; 2) the data are drawn from a mixture of two gaussians.

Usage

```
BICs(ddata, clusters, cutoff = 2.5)
```

Arguments

ddata	dataset on which the bayesian information criteria have to be computed
clusters	a list 0-1 of the same length of dataset indicating to which cluster a value belongs
cutoff	real value that controls the RLS steps for the robust estimates of the parameters of the gaussians

Details

The estimates of location and scale of the gaussians are obtained by carrying out reweighted least squares (RLS) steps on starting robust estimates. If center0 and deviation0 are the initial robust estimates of the gaussian, to each value in the dataset is assigned weigth = 1 if ((value - center0)/deviation0) < cutoff^2, weigth = 0 otherwise. The center and deviation estimates are updated with those value with weigth = 1.

The paremeter that controls the mixing of the two gaussians under the second hypothesis is given by mean(clusters).

Value

bics	list of two values: the first element is the bic under the first hypothesis, the second element is the bic under the secod hypothesis
bic1Parameters	list of two values that are the estimates of center and deviation under the first hypothesis
bic2Parameters	list of five values that are the estimates of center and deviation for the first cluster (clusters == 0), center and deviation for the second cluster (clusters == 1), and the estimate of the mixing parameter of the two gaussians.

Author(s)

Stefano M. Pagnotta and Michele Ceccarelli

Examples

```
data(geNSCLC)
ans <- classify(geNSCLC[, "STX1A"])
BICs(geNSCLC[, "STX1A"], ans$clusters)

# an example with missing values
data(geNSCLC)
ans <- classify(geNSCLC[, "CALCA"])
BICs(geNSCLC[!ans$missing, "CALCA"], ans$clusters[!ans$missing])
```

`classify` *Wrapper function to the classification method*

Description

This function links the classification method to the procedures to find the seed genes and the signatures. If necessary this function can be rewritten in order to use a different classification method. Actually the classification method linked is the partitioning around medoids (see pam() function for details)

Usage

```
classify(ddata)
```

Arguments

`ddata`	can be either a list of real values (gene expression levels) or a data-matrix where the rows are the samples and the columns are the genes.

Details

The function provides two clusters.

Value

If ddata is a list of real values (1 gene expression levels) the function calls an unbiased version of pam (see pamUnbiased() function for details). In this case two elements are returned

`clusters`	list of 1-2 indicators of the two clusters.
`missing`	list of T-F logical values where T labels the values set to missing by pamUnbiased().

If ddata is a real matrix (more than 1 expression levels) the function calls pam() and returns an object of pam class with an additional slot

`clusters`	list of 1-2 indicators of the two clusters.

Author(s)

Stefano M. Pagnotta and Michele Ceccarelli

See Also

pam, pam.object.

Examples

```
# univariate classification
data(geNSCLC)
sum(is.na(geNSCLC[, "SELP"]))
ans <- classify(geNSCLC[, "SELP"])
table(ans$clusters)
sum(ans$missing)
mean(geNSCLC[which(ans$clusters == 1), "SELP"], na.rm = TRUE)
mean(geNSCLC[which(ans$clusters == 2), "SELP"], na.rm = TRUE)

# multivariate classification
data(geNSCLC)
ddata <- geNSCLC[, c("STX1A", "FADD", "STC1", "RNF5")]
ans <- classify(ddata)$clusters
table(ans)
rbind(apply(ddata[ans == 1, ], 2, mean, na.rm = TRUE),
      apply(ddata[ans == 2, ], 2, mean, na.rm = TRUE))
```

---

ensembleTable — *Build a data frame populated with statistical indexes for each gene of each signature.*

Description

Its a summary table from which the ensemble signature is obtained.

Usage

```
ensembleTable(aSearchResults)
```

Arguments

aSearchResults	a vector of list

Details

aSearchResults is a vector of list having as many items as the number of signature to be developed. Each entry of the vector is a structure as results from the signatureFinder() function. An example of such a vector can be found in the source of the puttingAllTogether() function.

Value

`column no.1`	probe/gene found at least one time in some signature.
`column no.2`	counts of signature in which the probe/gene is present.
`column no.3`	mean importance of the probe/gene
`column no.4`	weighted mean importance
`column no.5 and more`	importance of the probe/gene with respect to the signature where it is found

Author(s)

Stefano M. Pagnotta and Michele Ceccarelli

See Also

`signatureFinder`, `puttingAllTogether`,

Examples

```
## see the source code of the puttingAllTogether() function.
get("puttingAllTogether")
```

---

geNSCLC *Gene expression levels on 147 samples affected by Non Small Cell Lung Cancer disease.*

Description

This is a matrix is built from the table "Normalized RT qPCR data" available at the source below. Each row is a sample and each column is a gene expression levels.

Usage

```
data(geNSCLC)
```

Format

The columns are the 158 genes and the 147 rows are the samples. This dataset is connected to the stNSCLC data.

Source

http://www.cs.toronto.edu/~juris/data/JCO07/

References

Lau, S.K., P. C. Boutros, M. Pintilie, F. H. Blackhall, C.-Q. Zhu, D. Strumpf, M. R. Johnston, G. Darling, S. Keshavjee, T. K. Waddell, N. Liu, D. Lau, L. Z. Penn, F. A. Shepherd, I. Jurisica, S. D. Der, M.-S. Tsao. A three-gene prognostic classifier for early-stage non-small cell lung cancer. J Clinical Oncology, 25(35): 5562-5569, 2007.

See Also

stNSCLC

Examples

```
data(geNSCLC)
dim(geNSCLC)
head(colnames(geNSCLC))
```

---

goodAndPoorClassification

Function to classify the unsupervised clusters found through a classification procedure in good and poor groups of samples.

Description

This function label the groups found by the unsupervised classification in good and poor.

Usage

```
goodAndPoorClassification(clustering)
```

Arguments

clustering	factor of so many elements as the number of samples containing the classification in two groups. Missing values are allowed.

Details

The labelling of the two groups in good and poor is obtained through the comparion of the expected and the observed failures computed in the test statistics of the log-rank test. The group for which the expected are more than the observed are classified are good prognosis. The variable stData has to be defined in the environment.

Value

A factor with levels "good" and "poor".

Author(s)

Stefano M. Pagnotta, Michele Ceccarelli and Pietro Zoppoli

Examples

```
data(geNSCLC)
geData <- geNSCLC

data(stNSCLC)
stData <- stNSCLC
######################
randomClustering  <- runif(nrow(stData)) > .5
survdiff(stData ~ randomClustering)
###############
clusteringByPrognosis <- goodAndPoorClassification(randomClustering)
survdiff(stData ~ clusteringByPrognosis)
```

`importance` *Compute an index of importance of the genes in the signature.*

Description

An index measuring the importance of each gene in the signature is computed. The more the index is close to 1, the most the corresponding gene is important to separate the survival curves. When the value is about 0, the gene has a low contribution in separating the survival curves. If it happens that the index is negative, the deletion of the gene from the signature improves the distance of the survival curves when all but it are in the signature.

Usage

```
importance(aSignatureFinder, deep = FALSE, cpuCluster = NULL, stopCpuCluster = TRUE)
```

Arguments

`aSignatureFinder`	Structure as results from the function signatureFinder()
`deep`	flag for future use
`cpuCluster`	structure as result from the NCPUS() function
`stopCpuCluster`	flag to control if the channel to the cpu-cluster has to be closed.

Details

The importance index associated to each gene of a signature is computed according to the leave one out strategy. Given a gene, it is removed from the signature and with the other genes in the sequence first a classification is performed and then thet test-value of the log-rank test is evaluated. These test-values are collected in the list L1GeneOutTV. All these values are compared to test-value of the signature: importance = 1 - L1GeneOutTV/aSignatureFinder$tValue.

Value

The function returns the same variable in the input signature structure and

L1GeneOutTV	a list of real values with the test-values of the log-rank test applied to all but one gene in the signature (leave-one-out strategy)
importance	A list of real values measuring the importance of each gene in the signature

Author(s)

Stefano M. Pagnotta and Michele Ceccarelli

See Also

signatureFinder.

Examples

```
data(geNSCLC)
geData <- geNSCLC
data(stNSCLC)
stData <- stNSCLC
aMakeCluster <- makeCluster(2)
geneSeed <- which(colnames(geData) == "SELP")
signature <- signatureFinder(geneSeed,
  cpuCluster = aMakeCluster, stopCpuCluster = FALSE)
signature <- importance(signature, cpuCluster = aMakeCluster)
signature
signature$importance
################################
barplot(signature$importance, main = "Importance based on L1GeneOut",
        sub = paste("Signature starting from:", signature$startingSignature))
```

NCPUS *Wrapper function to both makeCluster() and detecCore() functions of the 'parallel' library.*

Description

The function set up a cluster of 'nchips' cpu's on the local host.

Usage

```
NCPUS(nchips = FALSE)
```

Arguments

nchips	is the number of cpu's on the local pc

Details

The use of this function is depreceted. Use makeCluster() instead. The function is here for compatibility with the version 2013.08.20 of the library.

Value

The function returns a structure useful to establish a connection to the cluster of cpu's.

Note

The definition and connection to the cpu cluster can be established also by using the makeCluster() native function of the parallel package.

Author(s)

Stefano M. Pagnotta and Michele Ceccarelli

See Also

makeCluster, detectCores

Examples

```
# to count how many cpu's are on the localhost, run
detectCores()

# we suggest to set a cluster of maximal dimension -1 to leave a cpu to control the system
aMakeCluster <- makeCluster(2)
stopCluster(aMakeCluster)
```

---

puttingAllTogether — *Example of strategy to find signature from different seeds at the same time and to generate plots and summary table.*

Description

This function has been designed to show how to put the functions of the package in sequence in order to analyse microarray data for signatures. Given this function is an example, and can be run for small microarray data, the user is invited to a reverse engineering task (start by typing "puttingAllTogether") in order to define a procedure for more complex analysis sessions.

Usage

```
puttingAllTogether(workingFile = "", nchips = 2, alpha = 0.05,
                   saveSurvivalCurvesPlot = FALSE,
                   saveIndividualSignature = FALSE,
                   saveImportancePlot = FALSE)
```

Arguments

workingFile	prefix string for the log file and other output
nchips	see the NCPUS() function for details; it has to be set to the number of cpu's to use if useCpuCluster = TRUE
alpha	see the BHcorrection() function
saveSurvivalCurvesPlot	if TRUE a plot of the two survival curves of the signature are stored in a .pdf file in the working directory
saveIndividualSignature	if TRUE a .RData file is stored in the working directory with the result of the signatureFinder() function
saveImportancePlot	if TRUE a plot of the importances of the genes in the signature are stored in a .pdf file in the working directory

Details

The plotting of the figures may not work if a graphical device is not instantiated.

Value

Different files can be found in the getwd() directory.

Author(s)

Stefano M. Pagnotta and Michele Ceccarelli

Examples

```
get("puttingAllTogether")

# uncomment the following lines to performe an analysis
# of the non small cells lung cancer (NSCLC) data

##################
#data(geNSCLC)
#geData <- geNSCLC
#data(stNSCLC)
#stData <- stNSCLC

#puttingAllTogether(
# workingFile = "NSCLC",
######### modify the number of cpu according to your machine
# nchips = 2,
# saveIndividualSignature = TRUE)
```

`removeGeneFrom` *Remove the gene with the lowest importance from a signature*

Description

This function implements a pruning algorithm devoted to remove the gene with the lowest importance from the signature. The importances have to be computed before calling this function. The gene with the lowest importance is removed and the set of importances are computed again. The function is designed to be used iteratively so that all the genes with importance below a cutoff value are removed. If the signature has the results of the testGEL() function just computed, they are removed.

Usage

```
removeGeneFrom(aSignatureFinder, cutoff = 0)
```

Arguments

`aSignatureFinder`	Structure as results from the function signatureFinder()
`cutoff`	is a real value used when the function works iteratively

Value

The function return NULL when no gene is removed, otherwise return a structure as results from the importance() function with the additional slots

`removedGene`	list of genes (as string) removed from the signature
`originalSignature`	same as aSignatureFinder$signature
`originalClassification`	same as aSignatureFinder$classification
`originalTValue`	same as aSignatureFinder$tValue
`originalPValue`	same as aSignatureFinder$pValue

Author(s)

Stefano M. Pagnotta and Michele Ceccarelli

See Also

`signatureFinder`, `importance`.

Examples

```
data(geNSCLC)
geData <- geNSCLC
data(stNSCLC)
stData <- stNSCLC
# develop the signature
aMakeCluster <- makeCluster(2)
geneSeed <- which(colnames(geData) == "RNF5")
signature <- signatureFinder(geneSeed, cpuCluster = aMakeCluster,
                             stopCpuCluster = FALSE)
signature <- importance(signature, cpuCluster = aMakeCluster)
print(signature$importance)

################################
# 1 step pruning
removeGeneFrom(signature, cutoff = 0.3)$signature

################################
# iterative pruning of the signature
signature
repeat {
  tmp <- removeGeneFrom(signature, cutoff = 0.3)
  if(is.null(tmp)) break
  signature <- tmp
}
signature
signature$importance
```

`searchResultsSummaryTable` *Build a data frame populated with the results of a sequential searching of signatures starting from a list of seed-genes.*

Description

The data frame contains as many rows as the list of seed-genes and six columns as described in the value section.

Usage

```
searchResultsSummaryTable(aSearchResults)
```

Arguments

`aSearchResults` is a list of the signatures computed from a list of seed-genes. See the example.

Value

A data frame with the following columns

column no.1	starting/seed-gene.
column no.2	number of genes in the signature.
column no.3	test value of the log-rank test for the survival curves associated to the good and poor prognosis groups.
column no.4	log(p-value)/log(10), the p-value is the one corresponding to the test value of the log-rank test.
column no.5	ratio of the test values of the log-rank test computed with respect to the clustering given by the starting gene and the clustering generated by the signature
column no.6	genes in the signature

Note

The ratio of the test values in column no.5 measures the improvement of the separation of the survival curves when the classification is based on the signature instead of the starting gene.

Author(s)

Stefano M. Pagnotta and Michele Ceccarelli

See Also

help.

Examples

```
data(geNSCLC)
geData <- geNSCLC
data(stNSCLC)
stData <- stNSCLC

# CALCA = 152, SELP = 115
seedGenes <- c(152, 115)
K <- length(seedGenes)
searchResults <- vector("list", K)
names(searchResults) <- colnames(geData)[seedGenes]


aMakeCluster <- makeCluster(2)
for(k in 1:K)
  searchResults[[k]] <- signatureFinder(seedGenes[k],
    cpuCluster = aMakeCluster, stopCpuCluster = FALSE)
stopCluster(aMakeCluster)
searchResults
signaturesTable <- searchResultsSummaryTable(searchResults)
signaturesTable
```

`seedsFinder` *Evaluate some statistics on all genes in order to select those that can be used as seeds for searching the signatures.*

Description

This function works on each column (gene expression level) of the geData and returns the test-value and p-value of the log-rank test, the bayesian information criterion value under the hypothesis that tha data are drawn from a single gaussian (bic1) and a mixture of two gaussians (bic2); at the end the clustering of the samples is added.

Usage

```
seedsFinder(cutoff = 1.95, evaluateBICs = TRUE, cpuCluster = NULL)
```

Arguments

`cutoff`	argument passed to the BICs() function.
`evaluateBICs`	flag to force the computation of the bayesian information criteria.
`cpuCluster`	If a parallel search is necessary, this variable has to be set to the output of NCPUS() function.

Details

For each gene expression levels data an unbiased classification is performed resulting into two clusters coded by the values 0 and 1. The samples classified by 0 are those for which the mean is lower than that of the samples classified with 1. The classification method is the partitioning around medoids algorithm linked to the a leave-one-out re-classification strategy (see the pamUmbiased() function for further details). From the clusters two survival curves are estimated with the stData data and then tested for the null hypothesis of no difference among them (see the survdiff() function for further details) providing the tValue. The correponding pValue is given by 1-pchisq(tValue, df = 1). Two more indexes are computed, the bayesian information criteria under the hypotheses 1) the gene levels are from a univarite gaussians (bic1) and 2) the gene levels are from a mixture of two gaussians (bic2) (see the BICs() function for further details). The mixing coefficient is estimated from the classification as the fraction of samples classified as 1. The parameters of the gassians are robustly estimated.

Value

The result of the function is a matrix having so many rows as ncol(geData) and 4+nrow(geData) rows.

`column no.1: tValue`
: test-value of the log-rank test statistic under the null hypothesis that the two survival curves are equal (see details)

`column no.2: pValue`
: p-value corresponding to the test-value in column no.1; actuallly is 1-pchisq(tValue, df = 1)

`column no.3: bic1`
: value of the bayesian information criterion computed under the hypothesis that the data are drawn from a single gaussian

`column no.4: bic1`
: value of the bayesian information criterion computed under the hypothesis that the data are drawn from mixture of two gaussians

`columns from no.5 to no.4+nrow(geData)`
: result of the unbiased classification (see details)

Author(s)

Stefano M. Pagnotta and Michele Ceccarelli

See Also

pam, survdiff, BICs.

Examples

```
data(geNSCLC)
geData <- geNSCLC

data(stNSCLC)
stData <- stNSCLC

# here few genes and samples are considered to speed up the timing of the example.
# please, try
# genesToUse <- which(apply(!is.na(geData), 2, sum)/nrow(geData) > 0.75)
# geData <- geData[, genesToUse]
# and comment stData <- stData[1:50, ]
genesToUse <- which(apply(!is.na(geData), 2, sum) == nrow(geData))
geData <- geData[, genesToUse]
geData <- geData[1:50, ]
stData <- stData[1:50, ]
dim(geData)

aMakeCluster <- makeCluster(2)
aSeedsFinder <- seedsFinder(cpuCluster = aMakeCluster)
head(aSeedsFinder)
```

---

signatureFinder *Main function to find the signature.*

Description

This function implements the algorithm to find the signature using a searching strategy supervised by survival time data.

Usage

```
signatureFinder(seedGene, logFilePrefix = "", coeffMissingAllowed = 0.75,
subsetToUse = 1:ncol(geData), cpuCluster = NULL, stopCpuCluster = TRUE)
```

Arguments

seedGene	is the integer index pointing to the column (gene) of geData from which the searching strategy has to start. Optionally a list of genes (indexes pointing to the columns of geData) can be provided.
logFilePrefix	Is a string containing a prefix of the log file generated by the algorithm. No longer necessary in this upgrade of the package.
coeffMissingAllowed	This parameter controls the number of missing values tolerated by the pam classification procedure (see details).
subsetToUse	If necessary the costruction of the signature can be restricted to a subset of genes. In this case a list of the columns of geData has to be provided.
cpuCluster	If a parallel search is necessary, this variable has to be set to the output of NCPUS() function.
stopCpuCluster	flag to control if the channel to the cpu-cluster has to be closed

Details

In the global enviroment two variables have to be set up: geData and stData. geData is a matrix whose columns are the gene expressions and the rows are the samples (see geNSCLC for example). It is recommended that the columns names are instantiated. stData is a variable of the "Surv" class from the package "survival" (see stNSCLG for example).

Starting from the seed gene (a list of seeds is allowed), the next gene added is the one that maximizes the distance of the two survival curves. The list of genes grows until no more gene is able to improve the distance between the survival curves.

A gene (candidateGene) can be added to the running signature if it satisfies two controls: given the classification computed on the gene expressions of geneCandidate + runningSignature, 1) no cluster can have a dimension lower than floor(0.1 * nrow(geData)), and 2) the survival curves cannot cross. When more than 1 candidate gene is proposed, if the number of candidates is greater than 0.01*ncol(geData) the searching stops; otherwise a subset of the candidates is selected using backward strategy.

The parameter coeffMissingAllowed controls an empirical rule having in charge to prevent the crash of the pam() function. The number of joint missing values allowed in a sample described by p gene expression levels is given by floor(p^coeffMissingAllowed).

Value

The function returns a list with the following slots

signatureName	is a string for identifying the signature. By default is set to (colnames(geData)[seedGene])[1].
startingSignature	is a list of string set to colnames(geData)[seedGene]

`coeffMissingAllowed`
: same as input

`startingClassification`
: (factor) classification of the samples computed by using the gene expression levels of the startingSignature

`startingTValue`
: test-value of the log-rank test computed on the startingSignature

`startingPValue`
: p-value corresponding to the startingTValue

`signatureIDs`
: indexes pointing to the column of geData providing the sequence of gene expression levels that maximizes the distance between the two survival curves

`signature`
: labels corresponding to signatureIDs: colnames(geData)[signatureIDs]

`tValue`
: test-value of the log-rank test computed on the signature

`pValue`
: p-value corresponding to the tValue

`classification`
: (factor) classification of the samples computed by using the gene expression levels of the signature

Author(s)

Stefano M. Pagnotta and Michele Ceccarelli

See Also

geNSCLC, stNSCLC.

Examples

```
# find the signature starting from the gene SELP for the Non Small Cell Lung Cancer
#############
# set the working data
data(geNSCLC)
geData <- geNSCLC
data(stNSCLC)
stData <- stNSCLC
##############
# set the dimension of the cpu's cluster
aMakeCluster <- makeCluster(2)
################
# set the starting gene to SELP
geneSeed <- which(colnames(geData) == "SELP")
#################
# run ...
ans <- signatureFinder(geneSeed, cpuCluster = aMakeCluster)
ans
```

`signatureSummaryTable` *Build a data frame populated with the basic information of a signature.*

Description

A table with as many rows as the number of genes belonging to the signature.

Usage

```
signatureSummaryTable(aSignatureFinder)
```

Arguments

`aSignatureFinder`	Structure as results from the signatureFinder() function

Value

A data frame with the following columns

`column no.1`	genes in the signature
`column no.2`	percentage of missing data per gene
`column no.3`	importance (if computed)
`column no.4`	median level in good prognosis group (if computed)
`column no.5`	median level in poor prognosis group (if computed)
`column no.6`	mean level in good prognosis group (if computed)
`column no.7`	mean level in poor prognosis group (if computed)
`column no.8`	absolute difference of the mean levels (if computed)
`column no.9`	Welch's test-Value (if computed)
`column no.10`	p-value associated to t-value (if computed; see testGE() function)

Author(s)

Stefano M. Pagnotta and Michele Ceccarelli

See Also

`signatureFinder`, `importance`, `testGE`

Examples

```
data(geNSCLC)
geData <- geNSCLC
data(stNSCLC)
stData <- stNSCLC
# set the dimension of the cpu's cluster (use a value different from 2
# depending on the number of cpu avalilable)
aMakeCluster <- makeCluster(2)
signature <- signatureFinder(which(colnames(geData) == "HIF1a"),
  cpuCluster = aMakeCluster)
signature <- importance(signature)
signature <- testGE(signature)
signatureTable <- signatureSummaryTable(signature)
print(signatureTable)
```

---

stNSCLC *Survival times of 147 samples affected by Non Small Cell Lung Cancer disease.*

Description

This data are of the class "Surv" (defined in the library "survival") and are built by considering the columns "OS" and "STATUS" of the "Supplementary Table S1" at the reference below. The data are defined as stNSCLC <- Surv(OS, STATUS == 1). This dataset is connected to the geNSCLC data.

Usage

```
data(stNSCLC)
```

References

Lau, S.K., P. C. Boutros, M. Pintilie, F. H. Blackhall, C.-Q. Zhu, D. Strumpf, M. R. Johnston, G. Darling, S. Keshavjee, T. K. Waddell, N. Liu, D. Lau, L. Z. Penn, F. A. Shepherd, I. Jurisica, S. D. Der, M.-S. Tsao. A three-gene prognostic classifier for early-stage non-small cell lung cancer. J Clinical Oncology, 25(35): 5562-5569, 2007.

See Also

Surv,geNSCLC

Examples

```
data(stNSCLC)
library(survival)
plot(survfit(stNSCLC ~ 1))
```

testGE *Test the differential expression of the the genes in a signature with respect to the good and poor prognosis groups.*

Description

Given the clustering of the samples in good and poor prognosis associated to the signature, for each gene in the signature the test for the null hypothesis of equality of the expression levels is performed. Additional statistics are provided.

Usage

```
testGE(aSignatureFinder, permutationReplications = 1000,
  cpuCluster = NULL, stopCpuCluster = TRUE)
```

Arguments

aSignatureFinder	(structure) as results from the function signatureFinder().
permutationReplications	(integer) number of replications of the permutation test (default: 1000).
cpuCluster	structure as result from the NCPUS() function
stopCpuCluster	flag to control if the channel to the cpu-cluster has to be closed.

Details

The t-test for testing the differential expression of the genes in the signature is performed according to the procedure of Dudoit et al. (2002). The test statistics is the Welch's one and the null distribution is obtained through a permutation scheme.

Value

The function returns the same variable in the input aSignatureFinder structure and

groupMedian	real matrix with as many rows as length(aSignatureFinder$signature) and two columns containing the medians of each gene with respect to the good and poor prognosis group
medianAbsDifference	a list of real with as many elements as length(aSignatureFinder$signature) where each entry is the absolute difference of the medians computed in each group
groupMean	real matrix with as many rows as length(signature$signature) and two columns containing the means of each gene with respect to the good and poor prognosis group
meanAbsDifference	a list of real with as many elements as length(aSignatureFinder$signature) where each entry is the absolute difference of the means computed in each group

`meanDifferenceTValue`
: a list of real with as many elements as length(aSignatureFinder$signature) where each entry is the value of the test statistic

`meanDifferencePValue`
: a list of real with as many elements as length(aSignatureFinder$signature) where each entry is the p-value of the test statistic

Author(s)

Stefano M. Pagnotta and Michele Ceccarelli

References

Dudoit S. et al.: Statistical methods for identifying differentially expressed genes in replicated cDNA microarray experiments Statistica Sinica, 12, pp. 111–139, 2002.

See Also

signatureFinder,

Examples

```
# find the signature starting from the gene SELP for  the Non Small Cell Lung Cancer
# set the working data
data(geNSCLC)
geData <- geNSCLC
data(stNSCLC)
stData <- stNSCLC
# set the dimension of the cpu's cluster
# (use a value different from 2 depending on the number of cpu avalilable)
aMakeCluster <- makeCluster(2)
# set the starting gene to SELP
geneSeed <- which(colnames(geData) == "SELP")
# run ...
ans <- signatureFinder(geneSeed, logFilePrefix = "test",
cpuCluster = aMakeCluster, stopCpuCluster = FALSE)
ans
ans <- testGE(ans, cpuCluster = aMakeCluster)
ans$groupMean
ans$meanDifferencePValue
#####################
#library(gplots)
#barplot2(t(ans$groupMean), beside = TRUE,
#                   main = paste("Signature starting from:", ans$startingSignature),
#                   legend = paste(colnames(ans$groupMedian), "prognosis group"))
```

Index