Package ‘demi’

February 19, 2015

Type Package
Title Differential Expression from Multiple Indicators
Description Implementation of the DEMI method for the analysis of high-density microarray data.
URL http://biit.cs.ut.ee/demi
Version 1.1.2
Date 2015-01-02
Depends R (>= 3.0.0), utils
Imports R.utils, affxparser, affy, oligo, pplyr, methods, devtools
Suggests ggplot2, reshape
License GPL (>= 2)
Collate 'AllClasses.R' 'AllGenerics.R' 'Constructors.R'
  'DEMICel-methods.R' 'DEMIClust-methods.R' 'DEMIDiff-methods.R'
  'DEMIExperiment-methods.R' 'DEMIGroup-methods.R'
  'DEMIResult-methods.R' 'cluster-methods.R' 'custom-methods.R'
  'cytoband-methods.R' 'diffexp-methods.R' 'messages.R'
  'normalization-methods.R' 'other-methods.R' 'zzz.R'
NeedsCompilation no
Author Sten Ilmjarv [aut, cre],
  Hendrik Luuk [aut]
Maintainer Sten Ilmjarv <sten.ilmjarv@gmail.com>
Repository CRAN
Date/Publication 2015-02-13 20:22:57

R topics documented:

  addCustomTargets ........................................... 4
  addCytoband ................................................. 6
  adjust4maxprobes ......................................... 7
  attachResult ................................................. 7
  calcHypergeoExon ........................................... 9
<table>
<thead>
<tr>
<th>R topics documented:</th>
</tr>
</thead>
<tbody>
<tr>
<td>calcHypergeoProb</td>
</tr>
<tr>
<td>celMatrixNormalize</td>
</tr>
<tr>
<td>check4probe</td>
</tr>
<tr>
<td>check4target</td>
</tr>
<tr>
<td>checkDEMIExperiment_analysis</td>
</tr>
<tr>
<td>checkDEMIExperiment_celpath</td>
</tr>
<tr>
<td>checkDEMIExperiment_experiment</td>
</tr>
<tr>
<td>checkDEMIExperiment_maxprobes</td>
</tr>
<tr>
<td>checkDEMIExperiment_maxtargets</td>
</tr>
<tr>
<td>checkDEMIExperiment_normalization</td>
</tr>
<tr>
<td>checkDEMIExperiment_pmsize</td>
</tr>
<tr>
<td>cleanorganismname</td>
</tr>
<tr>
<td>cluster</td>
</tr>
<tr>
<td>createGroup</td>
</tr>
<tr>
<td>customObject</td>
</tr>
<tr>
<td>demi</td>
</tr>
<tr>
<td>demi.comp.test</td>
</tr>
<tr>
<td>demi.t.test</td>
</tr>
<tr>
<td>demi.wilcox.test</td>
</tr>
<tr>
<td>demi.wilcox.test.fast</td>
</tr>
<tr>
<td>DEMICel</td>
</tr>
<tr>
<td>DEMICel-class</td>
</tr>
<tr>
<td>DEMIClust</td>
</tr>
<tr>
<td>DEMIClust-class</td>
</tr>
<tr>
<td>DEMIDiff</td>
</tr>
<tr>
<td>DEMIDiff-class</td>
</tr>
<tr>
<td>demiequal</td>
</tr>
<tr>
<td>DEMIExperiment</td>
</tr>
<tr>
<td>DE_MIExperiment-class</td>
</tr>
<tr>
<td>DEMIGroup</td>
</tr>
<tr>
<td>DEMIGroup-class</td>
</tr>
<tr>
<td>DEMIMessages</td>
</tr>
<tr>
<td>DEMIPathway</td>
</tr>
<tr>
<td>DEMIResult-class</td>
</tr>
<tr>
<td>demisummary</td>
</tr>
<tr>
<td>diffexp</td>
</tr>
<tr>
<td>diffSpliceScore</td>
</tr>
<tr>
<td>findCytoband</td>
</tr>
<tr>
<td>getAlignment</td>
</tr>
<tr>
<td>getAnalysis</td>
</tr>
<tr>
<td>getAnnotation</td>
</tr>
<tr>
<td>getArrayType</td>
</tr>
<tr>
<td>getCelMatrix</td>
</tr>
<tr>
<td>getCelpath</td>
</tr>
<tr>
<td>getCluster</td>
</tr>
<tr>
<td>getClustMethod</td>
</tr>
<tr>
<td>getCutoffPvalue</td>
</tr>
<tr>
<td>getCytoband</td>
</tr>
</tbody>
</table>
R topics documented:

getDEMIClust ........................................ 80
getExperiment ....................................... 82
getGroup ............................................ 84
getGroupA ........................................... 86
getGroupB ........................................... 88
getGroupNames ...................................... 90
getIndexA ............................................ 91
getIndexB ............................................ 93
getMaxprobes ........................................ 95
getMaxtargets ....................................... 97
getName ............................................... 99
getNormMatrix ....................................... 100
getOrganism ......................................... 102
getPathway .......................................... 104
getProbeLevel ....................................... 106
getResult ............................................ 108
getResultTable ...................................... 110
getTargetAnnotation ................................ 112
getTargetProbes ..................................... 114
initialize.DEMICel .................................. 116
initialize.DEMIClust ................................ 117
initialize.DEMIDiff .................................. 117
initialize.DEMIExperiment ....................... 118
initialize.DEMIGroup ................................ 118
initialize.DEMIResult .............................. 119
loadAnnotation ...................................... 119
loadBlat ............................................ 120
loadCel ............................................. 121
loadCytoband ....................................... 121
loadDEMIlibrary .................................... 122
loadPathway ......................................... 123
makeDEMIResultsTable .............................. 124
makeUCSCLink ....................................... 124
matchExonGene ...................................... 125
norm.quantile ....................................... 126
norm.rank ........................................... 127
probe.levels ........................................ 128
probe.plot .......................................... 129
totalMatches_all .................................... 131
totalMatches_cluster ............................... 132
validDEMIClust ..................................... 133
validDEMIExperiment ............................... 133
wprob .................................................. 134

Index 135
addCustomTargets

Add new alignments to the alignment table

Description

The function addCustomTargets adds additional rows to the alignment table. Before adding new alignments the user needs to verify that the format of the rows corresponds to the format of the alignment table. These new alignments will then be incorporated in the analysis.

Usage

addCustomTargets(object = "DEMIExperiment", blat = "data.frame",
anno = "data.frame", overwrite = FALSE)

Arguments

object

A DEMIExperiment object. Determines the experiment where the additional alignment information will be added to.

blat

A data.frame. Represents the added alignments of probes to the target sequences.

anno

A data.frame. Represents the added annotation of the target sequences. If the parameter anno has not been defined then custom annotation will be populated with NA.

overwrite

A logical. If FALSE the previous alignment table will be overwritten. By default it is set to FALSE.

Details

The user needs to make sure that the proper fields in the additional alignment information are not missing. To see which fields are required use the function colnames(getAlignment(x)) on the DEMIExperiment object. The two fields that are always required are 'probeID' and 'targetID', the others are optional. All the other fields will be generated automatically and set to NA. If the user knows the annotation information for the targets in the alignment table then it is recommended to add that information to the annotation table. To see what are the fields of annotation table use the function colnames(getAnnotation(x)) on the DEMIExperiment object. This information will then be seen later in the analysis results. When adding custom annotations the only field that is required is the 'targetID', all other fields are optional however for later use it is better to add some more information about the target like it’s description.

Value

Returns the DEMIExperiment object where the additional information has been added to the alignment table.

Author(s)

Sten Ilmjarv
## Examples

```r
## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/"

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN02_GSM247697.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN03_GSM247700.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN04_GSM247701.CEL.gz", sep = "" ) )

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library(R.utils)
for( i in list.files( destfolder ) ) {
  gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function demi

# Set up an experiment
demiexp <- DEMIExperiment(analysis = 'gene', celpath = destfolder,
  experiment = 'myexperiment', organism = 'homo_sapiens')

# Create a custom annotation for one target ID
anno <- data.frame("RANDOM_ID", "Some kind of description")
colnames(anno) <- c("targetID", "description")
```
addCytoband

Add karyotype information to DEMI differential expression results

Description

The function `addCytoband` adds karyotype information to the results of DEMI differential expression 'genome' analysis. It is used internally in DEMI analysis.

Usage

`addCytoband(result, cyto)`

Arguments

- `result` A `data.frame`. A 'genome' analysis genome section that contains chromosome name, region start and region end coordinates.
- `cyto` A `data.frame`. A `data.frame` describing karyotype information of the organism used in the analysis.

Value

A `data.frame` where karyotype information has been added to the input 'result' table.

Author(s)

Sten Ilmjarv

See Also

`findCytoband` which this function wraps
adjust4maxprobes

 Adjust the DEMI analysis by maxprobes analysis

Description

The function adjust4maxprobes adjust the number of probes if the maxprobes has been set in the DEMIExperiment object. It is used internally in DEMI analysis.

Usage

adjust4maxprobes(targetMatches, maxprobes)

Arguments

targetMatches A data.frame. The original number of probes per target stored in a data.frame.
maxprobes A numeric. Specifies the maximum number of probes a target a target is adjusted against.

Value

A data.frame with the number of probes per target have been adjusted by maxprobes.

Author(s)

Sten Ilmjarv

attachResult

Attach results from DEMIDiff object to DEMIExperiment object

Description

The function attachResult attaches results stored in a DEMIDiff object to the underlying DEMIExperiment object. This function is useful because DEMIDiff can store results only for one differential expression analysis run whereas DEMIExperiment object can store all the results done on the same metadata stored in the DEMIExperiment object. So the user is allowed to keep several DEMI differential expression analysis results in one DEMIExperiment object for ease of use.

Usage

attachResult(object, diffObject)

## S4 method for signature 'DEMIExperiment,DEMIDiff'
attachResult(object, diffObject)
Arguments

object A DEMIExperiment object. The user needs to make sure that the DEMIExperiment object where the results will be added is identical to the DEMIExperiment object whose metadata was used to calculate differential expression.

diffObject A DEMIDiff object. The results from the diffObject parameter will be added to the results of the DEMIExperiment object in the object parameter.

Details

When adding results to DEMIExperiment object from a DEMIDiff object the user needs to make sure that the DEMIExperiment object that is stored under DEMIDiff object is identical to the DEMIExperiment object where the results will be added to. You can access the DEMIExperiment object from the DEMIDiff object with the function getExperiment(x) where x is a DEMIDiff object. With the function identical you can check if the DEMIExperiment objects are indeed identical.

Value

Returns a DEMIExperiment updated with the results from DEMIDiff object.

Author(s)

Sten Ilmjarv

See Also

DEMIExperiment, DEMIDiff, getExperiment, identical

Examples

```r
## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/"

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ),
)
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ),
)
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
```
calcHypergeoExon

Calculates hypergeometric probability in DEMI analysis

destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN02_GSM247697.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN03_GSM247700.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN04_GSM247701.CEL.gz", sep = "" )

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library( R.utils )
for( i in list.files( destfolder ) ) {
  gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function attachResult.

# Set up an experiment
demiexp <- DEMIExperiment( analysis = 'gene', celpath = destfolder,
experiment = 'myexperiment', organism = 'homo_sapiens' )

# Create clusters with an optimized wilcoxon's rank sum test incorporated within demi that
# precalculates the probabilities
demiclust <- DEMIClust( demiexp, group = c( "BRAIN", "UHR" ), clust.method = demi.wilcox.test.fast )

# Calculate differential expression
demidiff <- DEMI DIFF( demiclust )

# Attach the differential expression analysis results to the original 'DEMIExperiment' object
demiexp <- attachResult( demiexp, demidiff )

## End(Not run)

calcHypergeoExon

Calculates hypergeometric probability in DEMI analysis

Description

Calculates hypergeometric probability in DEMI analysis. It is only used for 'exon' analysis in
DEMI. It is used internally in DEMI analysis.

Usage

calcHypergeoExon(x)
calcHypergeoProb

Arguments

x A data.frame.

Value
A numeric that represents a the hypergeometric probability p-value.

Author(s)
Sten Ilmjarv

Description
Calculates hypergeometric probability in DEMI analysis. It is universal for all DEMI analysis except for 'exon' analysis. It is used internally in DEMI analysis.

Usage
calcHypergeoProb(x)

Arguments
x A data.frame.

Value
A numeric that represents a the hypergeometric probability p-value.

Author(s)
Sten Ilmjarv
**Description**

The function `celMatrixNormalize` initializes the normalization of the raw expression matrix in the `DEMIExperiment` object. It is used internally in DEMI analysis.

**Usage**

```r
celMatrixNormalize(object, fun)
```

```r
## S4 method for signature 'DEMIExperiment,function'
celMatrixNormalize(object, fun)
```

**Arguments**

- `object`: A `DEMIExperiment` object. It stores the raw expression matrix.
- `fun`: A function. The function used to normalize the raw expression matrix.

**Value**

Returns a `DEMIExperiment` object updated with normalized expression matrix.

**Author(s)**

Sten Ilmjarv

---

**Description**

The function `check4probe` checks if the probe ID’s specified in the probes vector are present in the alignment data of the specified `DEMIExperiment` object.

**Usage**

```r
check4probe(object, probes)
```

```r
## S4 method for signature 'DEMIExperiment,vector'
check4probe(object, probes)
```
Arguments

object
probes

A DEMIExperiment object.
A vector. A vector of probe ID’s.

Details

To see which probes are available in the alignment data use the function getAlignment(x)$probeID where x is an object of class DEMIExperiment.

Value

Returns NULL if all the probes are exist in the alignment data, else returns an error message.

Author(s)

Sten Ilmjarv

See Also

DEMIExperiment

Examples

## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitester/testdata/

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/" ),
check4target = paste( destfolder, "BRAIN02_GSM247697.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN03_GSM247700.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN04_GSM247701.CEL.gz", sep = "" )

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library( R.utils )
for( i in list.files( destfolder ) ) {
  gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function check4probe

# Set up an experiment
demiexp <- DEMIExperiment( analysis = 'gene', celpath = destfolder,
                          experiment = 'myexperiment', organism = 'homo_sapiens' )

# Create clusters with an optimized wilcoxon's rank sum test incorporated within demi that
# precalculates the probabilities
demiclust <- DEMIClust( demiexp, group = c( "BRAIN", "UHR" ), clust.method = demi.wilcox.test.fast )

# Check for probe ID's
check4probe( demiexp, c( 1155955, 100210 ) )

# To see what probes gave the error
setdiff( c( 1155955, 100210 ), getAlignment( demiexp )$probeID )

## End(Not run)

---

check4target  Checks if the targets are available

Description

The function check4target checks if the targets specified in the target vector are present in the
alignment data of the specified DEMIExperiment object.

Usage

check4target(object, target)

## S4 method for signature 'DEMIExperiment,vector'
check4target(object, target)
Arguments

object  A DEMIExperiment object.
target  A vector. Depending on the analysis the target can be an ensembl gene ID or gene symbol (e.g. 'MAOB'), ensembl transcript ID, ensembl peptide ID or genomic region ID.

Value

Returns TRUE if all the targets exists, else stops with an error message.

Author(s)

Sten Ilmjarv

See Also

DEMIExperiment

Examples

## Not run:

to use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/"

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN02_GSM247697.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN03_GSM247700.CEL.gz", sep = "" ) )
checkDEMIExperiment_analysis

Determines if the analysis is correct

Description
Checks if the analysis is correct for DEMI analysis. It can be either 'gene', 'transcript', 'exon' or 'genome'. It is used internally in DEMI analysis.

Usage
checkDEMIExperiment_analysis(analysis)

Arguments
analysis A character.

Value
Returns NULL if analysis is ok, else returns an error message.
checkDEMIExperiment_celpath

*Checks if celpath is correct*

**Description**
Checks if celpath is correct for DEMI analysis. The celpath stores CEL directory or CEL files. It is used internally in DEMI analysis.

**Usage**
```r
checkDEMIExperiment_celpath(celpath)
```

**Arguments**
- **celpath** A character.

**Value**
Returns NULL if celpath is ok, else returns an error message.

**Author(s)**
Sten Ilmjarv

---

checkDEMIExperiment_experiment

*Checks if the experiment is correct*

**Description**
Checks if the experiment is correct for the DEMI analysis. It is used internally in DEMI analysis.

**Usage**
```r
checkDEMIExperiment_experiment(experiment)
```

**Arguments**
- **experiment** A character.
checkDEMIExperiment_maxprobes

Value
Returns NULL if experiment is ok, else returns an error message.

Author(s)
Sten Ilmjarv

Description
Checks if maxprobes is correct for the DEMI analysis. It is used internally in DEMI analysis.

Usage
checkDEMIExperiment_maxprobes(maxprobes)

Arguments
maxprobes A numeric.

Value
Returns NULL if maxprobes is ok, else returns an error message.

Author(s)
Sten Ilmjarv

checkDEMIExperiment_maxtargets

Description
Checks if maxtargets is correct for the DEMI analysis. It is used internally in DEMI analysis.

Usage
checkDEMIExperiment_maxtargets(maxtargets)

Arguments
maxtargets A numeric.
checkDEMIExperiment_normalization

Value

Returns NULL if maxtargets is ok, else returns an error message.

Author(s)

Sten Ilmjarv

Description

Checks if normalization is correct for the DEMI analysis. ‘normalization’ stands for normalization method or ‘norm.method’ in the DEMIExperiment object. It is used internally in DEMI analysis.

Usage

checkDEMIExperiment_normalization(normalization)

Arguments

normalization A function.

Value

Returns NULL if normalization is ok, else returns an error message.

Author(s)

Sten Ilmjarv

See Also

DEMIExperiment
**checkDEMIExperiment_pmsize**

*Checks if pmsize is correct*

---

**Description**

Checks if pmsize is correct for the DEMI analysis. The 'pmsize' denotes perfect match size meaning the size of continues perfect alignment between the probe and the target sequence. It is used internally in DEMI analysis.

**Usage**

```
checkDEMIExperiment_pmsize(pmsize)
```

**Arguments**

- `pmsize` : A numeric.

**Value**

Returns NULL if pmsize is ok, else returns an error message.

**Author(s)**

Sten Ilmjarv

---

**cleanorganismname**  
*Cleans the organism name from redundant characters*

---

**Description**

Cleans the organism name from redundant. It is used internally in DEMI analysis.

**Usage**

```
cleanorganismname(organism)
```

**Arguments**

- `organism` : A character. A character specifying the organism name.

**Value**

A character representing clean organism name.

**Author(s)**

Sten Ilmjarv
cluster

Initializes the clustering of probes into clusters

Description
The function `cluster` clusters probes with the function specified in the `clust.method` parameter of the `DEMIClust` object. This function is used internally by DEMI analysis when clusters are created automatically from normalized expression matrix.

Usage
```
cluster(object)
```

Arguments
- `object` A `DEMIClust` object.

Value
Returns a `DEMIClust` object that is updated with a list of clustered probes.

Author(s)
Sten Ilmjarv

See Also
`DEMIClust`

createGroup

Creates a DEMIGroup object

Description
The function `createGroup` creates a `DEMIGroup` object for `DEMIClust` object. This function is used internally by DEMI analysis when clusters are created automatically from normalized expression matrix.

Usage
```
createGroup(object)
```

Arguments
- `object` A `DEMIClust` object.

Value
Returns a `DEMIGroup` object that is updated with a list of clustered probes.

Author(s)
Sten Ilmjarv

See Also
`DEMIClust`
customObject

Arguments

object A DEMIClust object.

Value

Returns a DEMIClust object that includes a DEMIGroup objec as the group parameter of the object.

Author(s)

Sten Ilmjarv

See Also

demiclust, demigroup

customObject Checks if the DEMIClust object is user defined or automatically generated

Description

The function customObject determines if the DEMIClust is a custom object defined by the users clusters or built automatically by a clustering method. It is used internally in DEMI analysis.

Usage

customObject(object)

## S4 method for signature 'DEMIClust'
customObject(object)

Arguments

object A DEMIClust object.

Value

Returns FALSE if the DEMIClust object was built automatically. Returns TRUE if the DEMIClust is user defined.

Author(s)

Sten Ilmjarv

See Also

DEMIClust
Examples

## Not run:


# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/"

# Download packed CEL files and change the names according to the feature they represent (for example to include UHR or BRAIN in them to denote the features).
# It is good practice to name the files according to their features which allows easier identification of the files later.

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN02_GSM247697.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN03_GSM247700.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN04_GSM247701.CEL.gz", sep = "" ) )

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library(R.utils)
for( i in list.files( destfolder ) ) {
gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function customObject

# Set up an experiment
demiexp <- DEMIExperiment( analysis = 'gene', celpath = destfolder, experiment = 'myexperiment', organism = 'homo_sapiens' )

# Create clusters with an optimized wilcoxon's rank sum test incorporated within demi that precalculates the probabilities
demiclust <- DEMIClust( demiexp, group = c( "BRAIN", "UHR" ), clust.method = demi.wilcox.test.fast )
# Check if user defined 'DEMIClust' object
customObject( demicluster )

# Define a custom 'DEMIClust' object with user defined clusters
demiclust_custom <- DEMIClust( demiexp, cluster = list( customcluster = c(1190, 1998, 2007) ) )

# Check if user defined 'DEMIClust' object
customObject( demiclust_custom )

## End(Not run)

---

**demi** *A wrapper for DEMI analysis*

### Description

Function *demi* is a wrapper for the whole DEMI analysis. First it creates a *DEMIExperiment* object, then uses it to create a *DEMIClust* object that contains the list of clustered probes and then performs differential expression analysis by running the function *DEMIDiff* that creates *DEMIDiff* object. The latter contains the results of the differential expression analysis. It also prints out the results to the working directory. If parameter `pathway` is set to `TRUE`, it also performs gene ontology analysis on the results in *DEMIDiff* object to determine statistically significant gene ontology categories (it also prints out those in the working directory with the file containing the string 'pathway'). It then returns a list containing the *DEMIExperiment* object where the results have been attached to and a *data.frame* that contains the functional annotation analysis results. NB! The results will be printed out in the working directory.

### Usage

```r
demi(analysis = "transcript", celpath = character(),
     experiment = character(), organism = character(), maxtargets = 0,
     maxprobes = character(), pmsize = 25, sectionsize = character(),
     group = character(), norm.method = norm.rrank, filetag = character(),
     cluster = list(), clust.method = function() { }, cutoff.pvalue = 0.05,
     pathway = logical())
```

### Arguments

- **analysis**: A character. Defines the analysis type. It can be either 'transcript', 'gene', 'exon' or 'genome'. The default value is 'transcript'. For 'genome' analysis sectionsize parameter needs to be defined as well.
- **celpath**: A character. It can point to the directory containing CEL files or is a vector that points directly to the CEL files.
- **experiment**: A character. A custom name of the experiment defined by the user (e.g. 'my-experiment').
organism A character. The name of the species the microarrays are measuring (e.g. 'homo_sapiens' or 'mus_musculus') given in lowercase and words separated by underscore.

maxtargets A numeric. The maximum number of allowed targets (e.g. genes or transcripts) one probe can match against. If to set it to 1 it means that the probe can match only one gene. If the analysis is set to 'transcript' the program still calculates the number of matches on genes. Hence a probe matching two transcripts on the same gene would be included but a probe matching two transcripts on different genes would not be included. The value needs to be a positive integer or 0. By default maxtargets is set to 0.

maxprobes A character. Sets the number of unique probes a target is allowed to have a match against. All the targets that yield more alignments to different probes then set by maxprobes will be scaled down to the number defined by the maxprobes parameter. It can be either a positive integer or set as 'median' or 'max' - 'median' meaning the median number of probes matching to all targets and 'max' meaning the maximum number of probes matching to a target. By default maxprobes is not set which is the same as setting maxprobes to 'max'.

pmsize A numeric. The minimum number of consecutive nucleotides that need to match perfectly against the target sequence. It can be either 23, 24 or 25. This means that alignments with smaller perfect match size will not be included in the experiment set up. The default value is 25.

sectionsize A numeric. This is only used if the analysis parameter is set to 'genome'. It defines the length of the genomic target region used in the 'genome' analysis.

group A character. Defines the groups that are used for clustering (e.g 'group = c("test", "control")'). It uses grep function to locate the group names from the CEL file names and then builds index vectors determining which files belong to which groups.

norm.method A function. Defines a function used to normalize the raw expression values. The default normalization function is norm.rank.

filetag A character. This is a custom string that can be used to identify the experiment. It incorporates it to the names of the output files.

cluster A list. Holds the probes of different clusters in a list.

clust.method A function. Defines the function used for clustering. The user can build a custom clustering function. The input of the custom function needs to be a DEMIClust object and the output is a list of probes, where each list corresponds to a specific cluster. The default function is demi.wilcox.test that implements the wilcox.test function. However we recommend to use the function demi.wilcox.test.fast that uses a custom wilcox.test and runs a lot faster.

cutoff.pvalue A numeric. Sets the cut-off p-value used for determining statistical significance of the probes when clustering the probes into clusters.

pathway A logical. If set to TRUE the functional annotation analysis is done on top of differential expression analysis.
Details

Instead of automatically clustered probes DEMIClust object can use user defined lists of probes for later calculation of differential expression. This is done by setting the cluster parameter. It overrides the default behaviour and no actual clustering occurs. Instead the list of probes defined in the cluster parameter are considered as already clustered probes. The list needs to contain proper names for probe vectors so that they would be recognizable later. Also instead of using the default clustering method the user can write his/her own function for clustering probes based on the expression values.

Further specification of the parameters:

- **maxtargets** When analysis is set to 'gene' then all probes that match to more genes then allowed by maxtargets parameter will not be included in the analysis. For 'transcript' and 'exon' analysis the number is also calculated on a gene level. For example if maxtargets is set to one and a probe matches to two transcripts but on the same gene, then this probe will still be used in the analysis. However if the probe matches two transcripts on different genes then this probe will not be included in the analysis. For 'genome' analysis the probe in most cases matches to two genomic sections because adjacent sections overlap by 50 probe will still be used in the analysis.

- **norm.method** Every user can apply their own normalization method by writing a custom normalization function. The function should take in raw expression matrix and return the normalized expression matrix where probe ID's are kept as rownames and column names are CEL file names. The normalized expression matrix will then be stored as part of the DEMIExperiment object.

- **sectionsize** The sectionsize parameter defines the length of the genomic target region. Currently sectionsize can be set as: 100000, 500000 and 1000000. All adjacent sections, except the ones on chromosome ends, overlap with the next adjacent section by 50 genomic section. This parameter is required when analysis is set to 'genome'.

- **group** All the CEL files used in the analysis need to contain at least one of the names specified in the group parameter because they determine what groups to compare against each other. It is also a good practice to name the CEL files to include their common features. However if a situation arises where the group/feature name occurs in all filenames then the user can set group names with specific filenames by seperating names in one group with the "|" symbol. For example group = c( "FILENAME1" | "FILENAME2" | "FILENAME3", "FILENAME4" | "FILENAME5" | "FILENAME6" ). These two groups are then used for clustering the probes expression values.

- **norm.method** The norm.method defines a function to use for the normalization of raw expression matrix. The user can implement his/her own function for the normalization procedure. The function should take in raw expression matrix and return the normalized expression matrix where probe ID's are kept as rownames and column names are CEL file names.

- **clust.method** The user can write his/her own function for clustering probes according to their expression values. The custom function should take DEMIClust object as the only parameter and output a list. The output list should contain the name of the clusters and the corresponding probe ID's. For example return( list( cluster1 = c(1:10), cluster2 = c(11:20), cluster3 = c(21:30) ) ).

- **cluster** This parameter allows to calculate differential expression on user defined clusters of probe ID's. It needs to be a list of probe ID's where the list names correspond to the cluster names. For example list( cluster1 = c(1:10), cluster2(1:10) ). When using this
approach you need to make sure that all the probe ID’s given in the clusters are available in the analysis. Otherwise an error message will be produced and you need to remove those probes that have no alignment in the analysis. When setting this parameter the default behaviour will be overridden and no default clustering will be applied.

Value

A list containing the DEMIExperiment object where differential expression results have been added to and a data.frame consisting of the functional annotation analysis results.

Author(s)

Sten Ilmarv

See Also

DEMIExperiment, DEMIClust, DEMIPathway, DEMIDiff, demi.wilcox.test.fast, wilcox.test

Examples

```r
## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

destdir <- "demitest/testdata"

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/"

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN01_GSM247697.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN02_GSM247700.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN03_GSM247701.CEL.gz", sep = "" )
```

library( R.utils )
for( i in list.files( destfolder ) ) {
  gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function demi

# Do DEMI analysis with functional annotation analysis
demires <- demi(analysis = 'gene', celpath = destfolder, group = c( "BRAIN", "UHR" ),
  experiment = 'myexperiment', organism = 'homo_sapiens',
  clust.method = demi.wilcox.test.fast, pathway = TRUE)

# Do DEMI analysis without functional annotation analysis
demires <- demi(analysis = 'gene', celpath = destfolder, group = c( "BRAIN", "UHR" ),
  experiment = 'myexperiment', organism = 'homo_sapiens',
  clust.method = demi.wilcox.test.fast, pathway = FALSE)

# Retrieve results from the created object
head( getResultTable( demires$experiment ) )

## End(Not run)

demi.comp.test  Cluster probes into higher and lower clusters based on their differential signalling

Description

Performs higher or lower comparison test on normalized expression matrix defined in the DEMIClust object. Only probes whose expression values in one group are all either bigger or smaller then the expression values in the comparative group are termed with significant differential expression.

Usage

demi.comp.test(x = "DEMIClust")

Arguments

x
A DEMIClust object. The DEMIClust object containing normalized expression values used for statistical significance test on differential signalling of probes. The object contains the column indexes of groups (e.g. 'test' and 'control') used in the analysis.
Value

A list. Returns a list containing different sets of probes that behave similarly under current statistical test (e.g. up- or down-regulated probes).

Author(s)

Sten Ilmjarv

Examples

```r
# Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files fill be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/"

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/")
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/")
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/")
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/")
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/")
destfile = paste( destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/")
destfile = paste( destfolder, "BRAIN02_GSM247697.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/")
destfile = paste( destfolder, "BRAIN03_GSM247700.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/")
destfile = paste( destfolder, "BRAIN04_GSM247701.CEL.gz", sep = "" )

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library( R.utils )
for( i in list.files( destfolder ) ) {
  gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function demi.comp.test
```
# Basic experiment set up.
demiexp <- DEMIExperiment(analysis = 'gene', celpath = destfolder,
experiment = 'myexperiment', organism = 'homo_sapiens')

# Create clusters with default behaviour
demiclust <- DEMIClust(demiexp, group = c("BRAIN", "UHR"))

# Retrieve probes whose differential signalling was statistically significant
sigprobes <- demi.comp.test(demiclust)

# However it makes more sense to incorporate the method straight into \code{DEMIClust} object
demiclust <- DEMIClust(demiexp, group = c("BRAIN", "UHR"), clust.method = demi.comp.test)

# Retrieve the probes whose differential signalling was statistically significant
sigprobes <- getCluster(demiclust)

# Retrieve the cluster names since we have both up-regulated and down-regulated probe clusters
names(sigprobes)

# Retrieve the up-regulated probes whose cluster names contain the sign '^[H]'
head(sigprobes[grep('^\[H\]', names(sigprobes))])

# Retrieve the down-regulated probes whose cluster names contain the sign '^[L]'
head(sigprobes[grep('^\[L\]', names(sigprobes))])

## End(Not run)

demi.t.test(x = "DEMIClust")

.transpose

Cluster probes into higher and lower clusters based on their differential signalling

Description
Performs t.test on normalized expression value matrix defined in 'DEMIClust' object.

Usage
demi.t.test(x = "DEMIClust")

Arguments
x A DEMIClust object. The DEMIClust object containing normalized expression values used for statistical significance test on differential signalling of probes. The object contains the column indexes of groups (e.g. 'test' and 'control') used in the analysis.

Value
A list. Returns a list containing different sets of probes that behave similarly under current statistical test (e.g. up- or down-regulated probes).
Author(s)
Sten Ilmjarv

See Also
t.test which this function wraps.

Examples

## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/"

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/")
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/")
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/")
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/")
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/")
destfile = paste( destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/")
destfile = paste( destfolder, "BRAIN02_GSM247697.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/")
destfile = paste( destfolder, "BRAIN03_GSM247700.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/")
destfile = paste( destfolder, "BRAIN04_GSM247701.CEL.gz", sep = "" )

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library( R.utils )
for( i in list.files( destfolder ) ) {
gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function demi.t.test
# Basic experiment set up.
demiexp <- DEMIExperiment(analysis = 'gene', celpath = destfolder,
experiment = 'myexperiment', organism = 'homo_sapiens')

# Create clusters with default behaviour
demiclust <- DEMIClust( demiexp, group = c( "BRAIN", "UHR" ) )

# Retrieve probes whose differential signalling was statistically significant
sigprobes <- demi.t.test( demiclust )

# However it makes more sense to incorporate the method straight into \code{DEMIClust} object
demiclust <- DEMIClust( demiexp, group = c( "BRAIN", "UHR" ), clust.method = demi.t.test )

# Retrieve the probes whose differential signalling was statistically significant
sigprobes <- getCluster( demiclust )

# Retrieve the cluster names since we have both up-regulated and down-regulated probe clusters
names( sigprobes )

# Retrieve the up-regulated probes whose cluster names contain the sign '{h}'
head( sigprobes[[grep('^\[H\]$', names( sigprobes ))]] )

# Retrieve the down-regulated probes whose cluster names contain the sign '{l}'
head( sigprobes[[grep('^\[L\]$', names( sigprobes ))]] )

## End(Not run)

---

demi.wilcox.test  

Cluster probes into higher and lower clusters based on their differential signalling

Description

Performs \code{wilcox.test} on normalized expression value matrix defined in \code{DEMIClust} object.

Usage

\code{demi.wilcox.test(x = "DEMIClust")}

Arguments

\code{x}  
A \code{DEMIClust} object. The \code{DEMIClust} object containing normalized expression values used for statistical significance test on differential signalling of probes. The object contains the column indexes of groups (e.g. 'test' and 'control') used in the analysis.

Value

A list. Returns a list containing different sets of probes that behave similarly under current statistical test (e.g. up- or down-regulated probes).
Author(s)
Sten Ilmjarv

See Also
wilcox.test which this function wraps.

Examples

## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

destfolder <- "demi/testdata/"

# Set the destination folder where the downloaded files fill be located.
# It can be any folder of your choosing.
destfolder <- "demi/testdata/"

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.

download.file( paste(ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/")
destfile = paste(destfolder, "UHR01_GSM247694.CEL.gz", sep = "")
download.file( paste(ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/")
destfile = paste(destfolder, "UHR02_GSM247695.CEL.gz", sep = "")
download.file( paste(ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/")
destfile = paste(destfolder, "UHR03_GSM247698.CEL.gz", sep = "")
download.file( paste(ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/")
destfile = paste(destfolder, "UHR04_GSM247699.CEL.gz", sep = "")
download.file( paste(ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/")
destfile = paste(destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "")
download.file( paste(ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/")
destfile = paste(destfolder, "BRAIN02_GSM247697.CEL.gz", sep = "")
download.file( paste(ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/")
destfile = paste(destfolder, "BRAIN03_GSM247700.CEL.gz", sep = "")
download.file( paste(ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/")
destfile = paste(destfolder, "BRAIN04_GSM247701.CEL.gz", sep = "")

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library(R.utils)
for( i in list.files( destfolder ) ) {
gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function demi.wilcox.test
```r
# Basic experiment set up
demiexp <- DEMIExperiment(analysis = 'gene', celpath = destfolder,
experiment = 'myexperiment', organism = 'homo_sapiens')

# Create clusters with default behaviour
demiclust <- DEMIClust(demiexp, group = c("BRAIN", "UHR"))

# Retrieve probes whose differential signalling was statistically significant
sigprobes <- demi.wilcox.test(demiclust)

# However it makes more sense to incorporate the method straight into \code{DEMIClustr} object
demiclust <- DEMIClust(demiexp, group = c("BRAIN", "UHR"), clust.method = demi.wilcox.test)

# Retrieve the probes whose differential signalling was statistically significant
sigprobes <- getCluster(demiclust)

# Retrieve the cluster names since we have both up-regulated and down-regulated probe clusters
names(sigprobes)

# Retrieve the up-regulated probes whose cluster names contain the sign \'[H]\'
head(sigprobes[grep("^[H]", names(sigprobes))])

# Retrieve the down-regulated probes whose cluster names contain the sign \'[L]\'
head(sigprobes[grep("^[L]", names(sigprobes))])

## End(Not run)
```

---

**demi.wilcox.test.fast**  
*Cluster probes into higher and lower clusters based on their differential signalling*

---

**Description**

Performs a modified `wilcox.test` on normalized expression value matrix defined in `DEMIClustr` object. It precalculates the probabilities of the rank sums and makes the algorithm run a lot quicker.

**Usage**

`demi.wilcox.test.fast(x = "DEMIClustr")`

**Arguments**

- `x`  
  A `DEMIClustr` object. The `DEMIClustr` object containing normalized expression values used for statistical significance test on differential signalling of probes. The object contains the column indexes of groups (e.g. 'test' and 'control') used in the analysis.
Value

A list. Returns a list containing different sets of probes that behave similarly under current statistical test (e.g. up- or down-regulated probes).

Author(s)

Sten Ilmjarv

See Also

wilcox.test which this function mimics and wprob which this function implements.

Examples

```r
## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files fill be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata"

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.
download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN02_GSM247700.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN03_GSM247701.CEL.gz", sep = "" ) )
#
# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won’t need them.
library( R.utils )
for( i in list.files( destfolder ) ) {
}
gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function demi.wilcox.test.fast

# Basic experiment set up.
demiexp <- DEMIExperiment(analysis = 'gene', celpath = destfolder, experiment = 'myexperiment', organism = 'homo_sapiens')

# Create clusters with default behaviour
demiclust <- DEMIClust( demiexp, group = c( "BRAIN", "UHR" ) )

# Retrieve probes whose differential signalling was statistically significant
sigprobes <- demi.wilcox.test.fast( demiclust )

# However it makes more sense to incorporate the method straight into \code(DEMIClust) object
demiclust <- DEMIClust( demiexp, group = c( "BRAIN", "UHR" ), clust.method = demi.wilcox.test.fast )

# Retrieve the probes whose differential signalling was statistically significant
sigprobes <- getCluster( demiclust )

# Retrieve the cluster names since we have both up-regulated and down-regulated probe clusters
names( sigprobes )

# Retrieve the up-regulated probes whose cluster names contain the sign '^[H]'
head( sigprobes[grep("^[H]", names( sigprobes ))] )

# Retrieve the down-regulated probes whose cluster names contain the sign '^[L]'
head( sigprobes[grep("^[L]", names( sigprobes ))] )

## End(Not run)

DEMICel  

---

DEMICel *Creates a DEMICel object*

### Description

A DEMICel holds the raw and normalized expression matrices. It is used internally in DEMI analysis.

### Usage

```r
DEMICel(celMatrix = matrix(), normMatrix = matrix())
```

### Arguments

- **celMatrix**: A matrix. The raw expression matrix.
- **normMatrix**: A matrix. The normalized expression matrix.
Details

Both expression matrices store the expression values in their columns and the column name represents the original CEL file name. The row names represent probe ID’s which makes it easy to retrieve specific expression data for specific probes.

Value

A DEMICel object that holds the raw and normalized expression matrices.

Author(s)

Sten Ilmjarv

DEMICel-class  

Class DEMICel

Description

The class DEMICel holds the raw and normalized expression matrices.

Arguments

celMatrix  A matrix. The raw expression matrix.
normMatrix A matrix. The normalized expression matrix.

Author(s)

Sten Ilmjarv

DEMIClust  

Creates a DEMIClust object

Description

A DEMIClust object clusters probes by their expression profile. The clustering is done with a function defined by the clust.method parameter. One could also define custom clusters by defining the cluster parameter with a list of probes. It then stores the clusters of probes as a DEMIClust object.

Usage

DEMIClust(experiment = "DEMIExperiment", group = character(), clust.method = function() {}, cluster = list(), cutoff.pvalue = 0.05)
DEMIClust

Arguments

- **experiment**: A DEMIExperiment object. Holds the DEMIExperiment object whose metadata (such as normalized expression values) is used to cluster the probes.

- **group**: A character. Defines the groups that are used for clustering (e.g. `group = c("TEST", "CONTROL")`). It uses `grep` function to locate the group names from the CEL file names and then builds index vectors determining which files belong to which groups.

- **clust.method**: A function. Defines the function used for clustering. The user can build a custom clustering function. The input of the custom function needs to be the `demiclust` object and the output is a list of probes, where each list corresponds to a specific cluster. The default function is `demi.wilcox.test` that implements the `wilcox.test` function. However we recommend to use the function `demi.wilcox.test.fast` that uses a custom `wilcox.test` and runs a lot faster.

- **cluster**: A list. Holds the probes of different clusters in a list.

- **cutoff.pvalue**: A numeric. Sets the cut-off p-value used for determining statistical significance of the probes when clustering the probes into clusters. Default is 0.05.

Details

Instead of automatically clustered probes DEMIClust object can use user defined lists of probes for later calculation of differential expression. This is done by setting the cluster parameter. It overrides the default behaviour of the DEMIClust object and no actual clustering occurs. Instead the list of probes defined in the cluster parameter are considered as already clustered probes. The list needs to contain proper names for probe vectors so that they would be recognizable later. Also instead of using the default clustering method the user can write his/her own function for clustering probes based on the expression values.

Further specification of the parameters:

- **group**: All the CEL files used in the analysis need to contain at least one of the names specified in the group parameter because they determine what groups to compare against each other. It is also a good practice to name the CEL files to include their common features. However if a situation arises where the group/feature name occurs in all filenames then the user can set group names with specific filenames by seperating names in one group with the "|" symbol. For example `group = c("FILENAME1|FILENAME2|FILENAME3","FILENAME4|FILENAME5|FILENAME6")`. These two groups are then used for clustering the probes expression values.

- **clust.method**: The user can write his/her own function for clustering probes according to their expression values. The custom function should take DEMIClust object as the only parameter and output a list. The output list should contain the name of the clusters and the corresponding probe ID's. For example `return( list( cluster1 = c(1:10), cluster2 = c(11:20), cluster3 = c(21:30) ) ).

- **cluster**: This parameter allows to calculate differential expression on user defined clusters of probe ID's. It needs to be a list of probe ID's where the list names correspond to the cluster names. For example `list( cluster1 = c(1:10),cluster2(1:10) )`. When using this approach you need to make sure that all the probe ID's given in the clusters are available in the analysis. Otherwise an error message will be produced and you need to remove those probes.
that have no alignment in the analysis.  When setting this parameter the default behaviour will
be overridden and no default clustering will be applied.

Value

A DEMIClust object.

Author(s)

Sten Ilmjarv

See Also

DEMIEperiment, demi.wilcox.test, demi.wilcox.test.fast, demi.comp.test, wprob

Examples

## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/"

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN02_GSM247697.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN03_GSM247700.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN04_GSM247701.CEL.gz", sep = "" )

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
DEMIClust-class

# Also we will remove the original unpacked files for we won't need them.
library( R.utils )
for( i in list.files( destfolder ) ) {
  gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function DEMIClust

# Set up an experiment.
demiexp <- DEMIExperiment(analysis = 'gene', celpath = destfolder,
                           experiment = 'myexperiment', organism = 'homo_sapiens')

# Create clusters with default behaviour
demiclust <- DEMIClust( demiexp, group = c( "BRAIN", "UHR" ) )

# Create clusters with an optimized wilcoxon's rank sum test incorporated within demi that
# precalculates the probabilities.
# The user can specify his/her own function for clustering.
demiclust <- DEMIClust( demiexp, group = c( "BRAIN", "UHR" ), clust.method = demi.wilcox.test.fast )

# Create a 'DEMIClust' object with custom lists of probeID's
demiclust <- DEMIClust( demiexp, cluster = list( customcluster = c(1190, 1998, 2007) ) )

# To retrieve the clusters use
getCluster( demiclust )

# To retrieve cluster names use
names( getCluster( demiclust ) )

## End(Not run)

DEMIClust-class  Class DEMIClust

Description

The class DEMIClust stores the probe clusters in a DEMIClust object.

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>experiment</td>
<td>A DEMIExperiment object. Holds the DEMIExperiment object whose metadata (such as normalized expression values) is used to cluster the probes.</td>
</tr>
<tr>
<td>group</td>
<td>A DEMIGroup object. Defines the groups that are used for clustering. The DEMIGroup object uses CEL file names to determine which files belong to which group. It uses grep function to locate the group names from the CEL file names.</td>
</tr>
<tr>
<td>clust.method</td>
<td>A function. Defines the function used for clustering. The user can build a custom clustering function. The input of the custom function needs to be the same DEMIClust object and the output is a list of probes, where each list corresponds</td>
</tr>
</tbody>
</table>
to a specific cluster. The default function is \texttt{demi.wilcox.test} that utilizes \texttt{wilcox.test}.

\begin{itemize}
  \item \texttt{cluster} \hspace{1cm} A list. Holds the probes of different clusters in a list.
  \item \texttt{cutoff.pvalue} \hspace{1cm} A numeric. Sets the cut-off p-value used for determining statistical significance of the probes when clustering the probes into clusters. Default is 0.05.
\end{itemize}

\textbf{Author(s)}

Sten Ilmjarpv

---

\textbf{DEMIDiff} \hspace{1cm} \textit{Creates a DEMIDiff object}

\textbf{Description}

The DEMIDiff object calculates differential expression and holds the analysis results, results of clustering and the original metadata of the experiment. To retrieve the results from the DEMIDiff object use the the function \texttt{getResultsTable} that returns the results as a \texttt{data.frame}.

\textbf{Usage}

\texttt{DEMIDiff(cluster = "DEMIClust")}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{cluster} \hspace{1cm} A DEMIClust object. The DEMIClust object that holds the clusters used in the analysis.
\end{itemize}

\textbf{Details}

The DEMIDiff object calculates the differential expression for every cluster in the DEMIClust object set by the cluster parameter. The results are then stored in the DEMIDiff object under the slot \texttt{result} as a DEMIResult object. This object can be retrieved with the function \texttt{getResult} but most of the times it is recommended to use the function \texttt{getResultTable} which returns the results in a \texttt{data.frame} sorted by the FDR values.

\textbf{Value}

A DEMIDiff object.

\textbf{Author(s)}

Sten Ilmjarpv

\textbf{See Also}

DEMIExperiment, DEMIClust, DEMIResult, getResultTable, getResult, attachResult
Examples

```r
## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/"

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.
download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ),
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ),
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ),
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ),
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "" ),
download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN02_GSM247697.CEL.gz", sep = "" ),
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN03_GSM247700.CEL.gz", sep = "" ),
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN04_GSM247701.CEL.gz", sep = "" ) )

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library(R.utils)
for( i in list.files( destfolder ) ) {
  gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function DEMIDiff

# Set up an experiment.
demiexp <- DEMIExperiment(analysis = 'gene', celpath = destfolder,
experiment = 'myexperiment', organism = 'homo_sapiens')

# Create clusters with an optimized wilcoxon's rank sum test incorporated within demi that
# precalculates the probabilities.
demiclust <- DEMIClust( demiexp, group = c("BRAIN", "UHR"), clust.method = demi.wilcox.test.fast )
```
# Calculate differential expression
demidiff <- DEMIDiff( demiclust )

# Retrieve the results in a 'data.frame'
head( getResultTable( demidiff ) )

# Attach the results to the original 'DEMIExperiment' object
demiexp <- attachResult( demiexp, demidiff )

# Retrieve the results from the 'DEMIExperiment' object
head( getResultTable( demiexp ) )

## End(Not run)

---

### DEMIDiff-class

**Class** DEMIDiff

**Description**

The class DEMIDiff holds the analysis results, results of clustering and the original metadata of the experiment.

**Arguments**

- **cluster**: A DEMIClust object. Holds information about the clusters in a DEMIClust object that it itself contains the DEMIExperiment object were experiment metadata such as alignment information and annotation is held.
- **name**: A character. A specific name of the differential expression analysis (e.g. 'BRAINvsUHR') that is generated according to the group names.
- **result**: A DEMIResult object. Holds the DEMIResult object of the analysis.

**Author(s)**

Sten Ilmjarv

---

### demiequal

**Cluster probes that have no statistically significant differential signalling**

**Description**

Performs wilcox.test on normalized expression value matrix defined in DEMIClust object and selects only these probes that have no differential signalling.
Usage

demiequal(x = "DEMIClust")

Arguments

x  
A DEMIClust object. The DEMIClust object containing normalized expression values used for statistical significance test on differential signalling of probes. The object contains the column indexes of groups (e.g. ‘test’ and ‘control’) used in the analysis.

Value

A list. Returns a list containing probes that did not have statistically significant differential signalling.

Author(s)

Sten Ilmjarv

See Also

wilcox.test which this function wraps.

Examples

## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.
#
# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/" ),
DEMIExperiment

### Description

This function creates a DEMIExperiment object. It loads and stores the experiment metadata such as annotation and alignment information and raw expression matrix from CEL files. It then normalizes the raw expression matrix and stores both expression matrices in a DEMICel object stored under the created DEMIExperiment object.
Usage

DEMIEExperiment(analysis = "transcript", celpath = character(),
experiment = character(), organism = character(), maxtargets = 0,
maxprobes = character(), pmsize = 25, sectionsize = character(),
norm.method = norm.rank, filetag = character())

Arguments

analysis A character. Defines the analysis type. It can be either 'transcript', 'gene', 'exon' or 'genome'. The default value is 'transcript'. For 'genome' analysis sectionsize parameter needs to be defined as well.
celpath A character. It can point to the directory containing CEL files or is a vector that points directly to the CEL files.
experiment A character. A custom name of the experiment defined by the user (e.g. 'my-experiment').
organism A character. The name of the species the microarrays are measuring (e.g. 'homo_sapiens' or 'mus_musculus') given in lowercase letters and words are separated by underscore.
maxtargets A numeric. The maximum number of allowed targets (e.g. genes or transcripts) one probe can have a match against. If to set it to 1 it means that the probe can match only one gene. If the analysis is set to 'transcript' the program still calculates the number of matches on genes, not transcripts. Hence a probe matching two transcripts on the same gene would be included but a probe matching two transcripts on different genes would not be included. The value needs to be a positive integer or 0. By default maxtargets is set to 0.
maxprobes A character. Sets the number of unique probes a target is allowed to have a match against. All the targets that yield more alignments to different probes then set by maxprobes will be scaled down to the number defined by the maxprobes parameter. It can be either a positive integer or set as 'median' or 'max' - 'median' meaning the median number of probes matching to all targets and 'max' meaning the maximum number of probes matching to a target. By default maxprobes is not set which is the same as setting maxprobes to 'max'.
pmsize A numeric. The minimum number of consecutive nucleotides that need to match perfectly against the target sequence. It can be either 23, 24 or 25. This means that alignments with smaller perfect match size will not be included in the experiment set up. The default value is 25.
sectionsized A numeric. This is only used if the analysis parameter is set to 'genome'. It defines the length of the genomic target region used in the 'genome' analysis. Currently the only available section sizes are 100000, 500000 and 1000000.
norm.method A function. Defines a function used to normalize the raw expression values. The default normalization function is norm.rank.
filetag A character. This is a custom string that can be used to identify the experiment. At the current development stage this parameter is used only when using the function demi, where the output files will contain the specified filetag.
Details

After the analysis has been completed the user can add the results from the analysis to the original DEMIExperiment object with the function attachResult. Then the function getResultSetTable can be used to retrieve the results from the DEMIExperiment object. Other useful functions are getNormMatrix to retrieve normalized expression matrix and getCelMatrix to retrieve the raw expression matrix. In both cases the probe ID’s are present as row names.

Further specification of the parameters:

- **maxtargets** When analysis is set to ‘gene’ then all probes that match to more genes then allowed by maxtargets parameter will not be included in the analysis. For ‘transcript’ and ‘exon’ analysis the number is also calculated on a gene level. For example if maxtargets is set to one and a probe matches to two transcripts but on the same gene, then this probe will still be used in the analysis. However if the probe matches two transcripts on different genes then this probe will not be included in the analysis. For ‘genome’ analysis the probe in most cases matches to two genomic sections because adjacent sections overlap by 50 probe will still be used in the analysis.

- **norm.method** Every user can apply their own normalization method by writing a custom normalization function. The function should take in raw expression matrix and return the normalized expression matrix where probe ID’s are kept as rownames and column names are CEL file names. The normalized expression matrix will then be stored as part of the DEMIExperiment object.

- **sectionsise** The sectionsise parameter defines the length of the genomic target region. Currently sectionsise can be set as: 100000, 500000 and 1000000. All adjacent sections, except the ones on chromosome ends, overlap with the next adjacent section by 50 genomic section. This parameter is required when analysis is set to ‘genome’.

- **norm.method** The norm.method defines a function to use for the normalization of raw expression matrix. The user can implement his/her own function for the normalization procedure. The function should take in raw expression matrix and return the normalized expression matrix where probe ID’s are kept as rownames and column names are CEL file names.

Value

A DEMIExperiment object.

Author(s)

Sten Ilmjarv

See Also

DEMCIsult, DEMIResult, getResultSetTable, getResultSet, attachResult

Examples

```r
## Not run:

# To use the example we need to download a subset of CEL files from
```
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "" ),
destfile = paste( destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "" ),
destfile = paste( destfolder, "BRAIN02_GSM247697.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "" ),
destfile = paste( destfolder, "BRAIN03_GSM247700.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "" ),
destfile = paste( destfolder, "BRAIN04_GSM247701.CEL.gz", sep = "" ) )

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library( R.utils )
for( i in list.files( destfolder ) ) {
gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function DEMIExperiment

# Basic experiment set up.
demiexp <- DEMIExperiment(analysis = 'gene', celpath = destfolder,
experiment = 'myexperiment', organism = 'homo_sapiens')

# Run basic experiment set up but this time do 'transcript' analysis.
demiexp <- DEMIExperiment(analysis = 'transcript', celpath = destfolder,
experiment = 'myexperiment', organism = 'homo_sapiens')

# Run basic experiment set up but this time do 'transcript' analysis.
demiexp <- DEMIExperiment(analysis = 'exon', celpath = destfolder,
experiment = 'myexperiment', organism = 'homo_sapiens')

# For genome analysis do not forget to specify the sectionsize parameter.
demiexp <- DEMIExperiment(analysis = 'genome', celpath = destfolder,
experiment = 'myexperiment', organism = 'homo_sapiens', sectionsize = 500000)

# Specify experiment with specific pmsize; the standard length for Affymetrix microarray
# probe is 25 nucleotides.
demiexp <- DEMIExperiment(analysis = 'gene', celpath = destfolder, experiment = 'myexperiment', organism = 'homo_sapiens', pmsize = 23)

# Specify experiment by setting maxtargets to 1.
demiexp <- DEMIExperiment(analysis = 'gene', celpath = destfolder, experiment = 'myexperiment', organism = 'homo_sapiens', maxtargets = 1)

# Specify experiment by setting maxprobes to 'median'.
demiexp <- DEMIExperiment(analysis = 'gene', celpath = destfolder, experiment = 'myexperiment', organism = 'homo_sapiens', maxprobes = 'median')

# Retrieve the alignment information from the DEMIExperiment object.
head( getAlignment( demiexp ) )

# Retrieve the annotation information from the DEMIExperiment object.
head( getAnnotation( demiexp ) )

# Retrieve the raw expression matrix from the DEMIExperiment object.
head( getCelMatrix( demiexp ) )

# Retrieve the normalized expression matrix from the DEMIExperiment object.
head( getNormMatrix( demiexp ) )

# If the user has done the analysis and wishes to add the results to the original
# DEMIExperiment object.

# Create clusters with an optimized Wilcoxon's rank sum test incorporated within demi that
# precalculates the probabilities.
demiclust <- DEMIClust( demiexp, group = c("BRAIN", "UHR"), clust.method = demi.wilcox.test.fast )

# Calculate differential expression
demidiff <- DEMIDiff( demiclust )

# Attach the results to the original DEMIExperiment object
demiexp <- attachResult( demiexp, demidiff )

# Retrieve the results from the DEMIExperiment object
head( getResultTable( demiexp ) )

## End(Not run)
DEMIEExperiment-class

Description

The class DEMIEExperiment defines an experiment. It holds the raw and normalized expression data as well as annotation information for selected analysis (either 'gene', 'transcript', 'exon' or 'genome'). It can be used to hold all the analysis results (DEMIDiff objects) done on the same DEMIEExperiment object.

Arguments

- **analysis**: A character. Defines the analysis type. It can be either 'transcript', 'gene', 'exon' or 'genome'. The default value is 'transcript'. For 'genome' analysis sectionsize parameter needs to be defined as well.
- **celpath**: A character. It can point to the directory containing the CEL files or is a vector that points directly to the CEL files.
- **experiment**: A character. A custom name of the experiment defined by the user (e.g. 'my-experiment').
- **organism**: A character. The name of the species the microarrays are measuring (e.g. 'homo_sapiens' or 'mus_musculus') given in lowercase and words separated by underscore.
- **arraytype**: A character. Holds the platform name of the microarrays used in the analysis.
- **maxtargets**: A numeric. The maximum number of allowed targets (e.g. genes or transcripts) one probe can have a match against. If to set it to 1 it means that the probe can match only one gene. If the analysis is set to 'transcript' the program still calculates the number of matches on genes. Hence a probe matching two transcripts on the same gene would be included but a probe matching two transcripts on different genes would not be included. The value needs to be a positive integer or 0.
- **maxprobes**: A character. Sets the number of unique probes a target is allowed to have a match against. All the targets that yield more alignments to different probes then set by maxprobes will be scaled down to the number defined by the maxprobes parameter. It can be either a positive integer or set as 'median' or 'max' - 'median' meaning the median number of probes matching to all targets and 'max' meaning the maximum number of probes matching to a target.
- **pmsize**: A numeric. The minimum number of consecutive nucleotides that need to match perfectly against the target sequence. It can be either 23, 24 or 25. This means that alignments with smaller perfect match size will not be included in the experiment.
- **sectionsized**: A numeric. This is only used if the analysis parameter is set to 'genome'. It defines the length of the genomic target region used in the 'genome' analysis.
- **norm.method**: A function. Defines a function used to normalize the raw expression values. The function should take in raw expression matrix and return the normalized expression matrix where probe ID’s are kept as rownames and column names are CEL file names.
- **filetag**: A character. This is a custom string that can be used to identify the experiment. At the current development stage this parameter is used only when using the function demi, where the output files will contain the specified filetag.
annoTable: A data.frame. Holds the annotation information used in the experiment.
blatTable: A data.frame. Holds the alignment information of probes and their corresponding targets.
cyto band: A data.frame. Only used in the 'genome' analysis. Holds the karyotype information for every chromosome of the species specified by the organism parameter.
p athway: A data.frame. Only used in the 'gene' and 'transcript' analysis. Holds the genes for every gene ontology category.
exprsdata: A DEMICel object. Holds the raw and normalized expression matrices in a DEMICel object.
results: A list. Can be used to store all the results as DEMIDiff objects done on the same DEMIExperiment object.

Author(s)
Sten Ilmjarv

---

**DEMIGroup**

*Creates a DEMIGroup object*

**Description**

A DEMIGroup object holds the group annotations such as the column indexes of both groups and names of the groups. It is used internally in DEMI analysis.

**Usage**

```
DEMIGroup(groupA = character(), groupB = character(), indexA = numeric(),
          indexB = numeric(), groupNames = character())
```

**Arguments**

- `groupA`: A character. Holds the name of group A.
- `groupB`: A character. Holds the name of group B.
- `indexA`: A numeric. A vector of column indexes belonging to group A.
- `indexB`: A vector. A vector of column indexes belonging to group B.
- `groupNames`: A character. Holds the names of custom groups created by the user.

**Details**

The DEMIGroup can hold both automatically generated annotations that depend on the group names or custom annotations specified by the user. The automatically generated ones are created by scanning for the specified group names in the column names of the normalized expression matrix. It then retrieves the column indexes where the specified group names occur. The custom group names are just stored in the groupNames vector and all the other parameters of the DEMIGroup object will be left empty.
Value

A DEMIGroup object that holds the group annotations.

Author(s)

Sten Ilmjarv

Description

The class DEMIGroup holds the information about the groups. Such as the column indexes of both groups and names of the groups.

Arguments

groupA  A character. Holds the name of group A.
groupB  A character. Holds the name of group B.
indexA  A numeric. A vector of column indexes belonging to group A.
indexB  A numeric. A vector of column indexes belonging to group B.
groupNames  A character. Holds the names of custom groups created by the user.

Author(s)

Sten Ilmjarv

Description

This list contains all the messages shown to the user when running DEMI analysis. It is used internally in DEMI analysis.

Usage

DEMIMessages
Format

List of 33

$ demiequal : List of 3
..$ main : chr "# Clustering probes into 'equal' cluster based on Wilcoxon rank sum test if
..$ custom.approach : chr "tUsing an optimized implementation to perform Wilcoxon rank sum test\n..$ standard.approach: chr "\tUsing the native implementation of the Wilcoxon rank sum test (function

$ demi.wilcox.test.fast : List of 3
..$ main : chr "# Clustering probes into 'higher' and 'lower' clusters based on Wilcoxon rank sum test
..$ custom.approach : chr "Using an optimized implementation to perform Wilcoxon rank sum test\n..$ standard.approach: chr "\tUsing the native implementation of the Wilcoxon rank sum test (function

$ demi.wilcox.test 
..$ main : chr "# Clustering probes into 'higher' and 'lower' clusters based on Wilcoxon rank sum test
..$ exact.false: chr "tUsing wilcox with parameter 'exact = FALSE' since sample sizes are sufficiently
..$ exact.true : chr "\tUsing wilcox with parameter 'exact = TRUE'.\n
$ demiExperiment_t.test : List of 1
..$ main: chr "# Clustering probes into 'higher' and 'lower' clusters based on t-test (function 't.test

$ demi.comp.test : List of 1
..$ main: chr "# Clustering probes into 'higher' and 'lower' clusters\n
$ takestime : function (no.of.probes)
..- attr(*, "srcref")=Class 'srcref' atomic [1:8] 41 15 41 131 29 145 41 41
..- ...- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e9424f740>

$ probesdone : function (no.of.probes)
..- attr(*, "srcref")=Class 'srcref' atomic [1:8] 42 16 42 99 30 113 42 42
..- ...- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e9424f740>

$ sectionsizeMissing : function ()
..- attr(*, "srcref")=Class 'srcref' atomic [1:8] 49 24 49 118 38 132 49 49
..- ...- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e9424f740>

$ positiveIntegerOrZero : function (parameter)
..- attr(*, "srcref")=Class 'srcref' atomic [1:8] 50 27 50 117 41 131 50 50
..- ...- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e9424f740>

$ sectionsizeUse : function ()
..- attr(*, "srcref")=Class 'srcref' atomic [1:8] 52 52 67 34 121 52 52
..- ...- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e9424f740>

$ parameterOfClassMissing : function (parameter, cls)
..- attr(*, "srcref")=Class 'srcref' atomic [1:8] 53 29 53 130 43 144 53 53
..- ...- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e9424f740>

$ paramAndParamNotEqualLength: function (param1, param2)
..- attr(*, "srcref")=Class 'srcref' atomic [1:8] 54 33 54 142 47 156 54 54
..- ...- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e9424f740>

$ wrongDefinition : function ()
..- attr(*, "srcref")=Class 'srcref' atomic [1:8] 55 21 58 94 35 150 55 58
..- ...- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e9424f740>

$ savingWhatTo : function (what, to)
..- attr(*, "srcref")=Class 'srcref' atomic [1:8] 59 18 59 82 32 96 59 59
..- ...- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e9424f740>

$ customTargets : List of 11
..- attr(*, "srcref")=Class 'srcref' atomic [1:8] 64 23 64 132 51 160 64 64
$. notFolder :function (parameter)
  ...- attr(*, "srcref")=Class 'srcref' atomic [1:8] 113 17 113 91 45 119 113 113
  ...- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e9424f740>
$. folderEmpty :function (parameter)
  ...- attr(*, "srcref")=Class 'srcref' atomic [1:8] 114 19 114 130 47 158 114 114
  ...- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e9424f740>
$. notACELFile :function (file, parameter)
  ...- attr(*, "srcref")=Class 'srcref' atomic [1:8] 115 19 115 135 47 163 115 115
  ...- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e9424f740>
$. maxprobesError :chr "The parameter 'maxprobes' has to be a positive integer or set as 'mean'"
  ...- attr(*, "srcref")=Class 'srcref' atomic [1:8] 119 22 117 195 50 223 117 117
  ...- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e9424f740>
$. probesExcluded :function (excludedProbes)
  ...- attr(*, "srcref")=Class 'srcref' atomic [1:8] 120 32 120 288 60 316 120 120
  ...- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e9424f740>
$. noTargetsFound :chr "No targets were found"
  ...- attr(*, "srcref")=Class 'srcref' atomic [1:8] 124 31 124 120 59 148 124 124
  ...- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e9424f740>
$. attachErrorResultsExist: function (variable)
  ...- attr(*, "srcref")=Class 'srcref' atomic [1:8] 125 27 125 114 55 142 125 125
  ...- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e9424f740>
$. includedCELFiles :function (celfiles)
  ...- attr(*, "srcref")=Class 'srcref' atomic [1:8] 126 29 126 142 57 170 126 126
  ...- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e9424f740>
$. usingMicroarrayPlatform :function (platform)
  ...- attr(*, "srcref")=Class 'srcref' atomic [1:8] 127 31 127 143 59 171 127 127
  ...- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e9424f740>
$. libraryNotInstalled :function (packageName)
  ...- attr(*, "srcref")=Class 'srcref' atomic [1:8] 128 25 128 142 53 170 128 128
  ...- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e9424f740>
$. searchingInternetRepo :function (packageName)
  ...- attr(*, "srcref")=Class 'srcref' atomic [1:8] 130 26 130 133 54 161 130 130
  ...- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e9424f740>
$. packageNotFoundFromRepo :function (packageName, package_url)
  ...- attr(*, "srcref")=Class 'srcref' atomic [1:8] 131 23 131 179 51 207 131 131
  ...- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e9424f740>
$. installingPackage :function (packageName, repository)
  ...- attr(*, "srcref")=Class 'srcref' atomic [1:8] 132 27 132 169 55 197 132 132
  ...- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e9424f740>
DEMIMessages

...$ cantLoadWhatTable : function (whatTable)
...  ... attr(*, "srcref") = Class 'srcref' atomic [1:8] 133 25 133 116 53 144 133 133
...  ...  ... attr(*, "srcfile") = Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e942f740>
...$ loadingBlaSuccess : function (package)
...  ... attr(*, "srcref") = Class 'srcref' atomic [1:8] 134 26 134 132 54 160 134 134
...  ...  ... attr(*, "srcfile") = Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e942f740>
...$ blatDoesNotExist : function (blat, platform, packageName)
...  ... attr(*, "srcref") = Class 'srcref' atomic [1:8] 135 25 135 192 53 220 135 135
...  ...  ... attr(*, "srcfile") = Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e942f740>
...$ pmsizeInvalid : function (pmsizes)
...  ... attr(*, "srcref") = Class 'srcref' atomic [1:8] 136 22 136 182 50 210 136 136
...  ...  ... attr(*, "srcfile") = Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e942f740>
...$ ignoreStrandMatches : function (strand)
...  ... attr(*, "srcref") = Class 'srcref' atomic [1:8] 137 27 137 112 55 140 137 137
...  ...  ... attr(*, "srcfile") = Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e942f740>
...$ loadingCytoband : chr "# Loading cytoband information"
...$ loadingPathway : chr "# Loading pathway information"
...$ check4ProbeError : function (difference)
...  ... attr(*, "srcref") = Class 'srcref' atomic [1:8] 140 24 142 124 52 173 140 142
...  ...  ... attr(*, "srcfile") = Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e942f740>
...$ check4TargetError : function (notfound)
...  ... attr(*, "srcref") = Class 'srcref' atomic [1:8] 143 25 143 181 53 209 143 143
...  ...  ... attr(*, "srcfile") = Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e942f740>
...$ zeroTargetsFound : chr "0 specified targets were found in the experiment"
$ DEMIResults : List of 2
...$ resultsContain : chr "the results list can only contain objects of class 'DEMIResult'
...  ... resultsUndefined : chr "no results have been defined"
$ diffexp : List of 2
...$ matchesCluster : chr "\t\tCalculating matches for probes in cluster ...
...  ... matchesAll : chr "\t\tCalculating matches over all probes ...
$ normalization : List of 3
...$ main : chr "# Normalizing expression values"
...$ normrank : chr "- using 'relative rank' as the normalization method"
...$ normquantile : chr "- using 'quantile normalization' as the normalization method"
$ parameterMissing : function (parameter)
...  ... attr(*, "srcref") = Class 'srcref' atomic [1:8] 171 22 171 93 36 107 171 171
...  ...  ... attr(*, "srcfile") = Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e942f740>
$ parameterNotOfClass : function (parameter, cls)
...  ... attr(*, "srcref") = Class 'srcref' atomic [1:8] 172 25 172 116 39 130 172 172
...  ...  ... attr(*, "srcfile") = Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e942f740>
$ isNotError : function (parameter, what)
...  ... attr(*, "srcref") = Class 'srcref' atomic [1:8] 173 16 173 93 30 107 173 173
...  ...  ... attr(*, "srcfile") = Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e942f740>
$ hasToBeNumericBetween : function (parameter, start, end)
...  ... attr(*, "srcref") = Class 'srcref' atomic [1:8] 174 27 174 142 41 156 174 174
...  ...  ... attr(*, "srcfile") = Classes 'srcfilecopy', 'srcfile' <environment: 0x7f8e942f740>
$ tooManyParameters : function (parameter, len)
...  ... attr(*, "srcref") = Class 'srcref' atomic [1:8] 175 23 175 111 37 125 175 175
Author(s)

Sten Ilmjarv, Hendrik Luuk

---

### DEMIPathway

**Functional annotation of DEMI results**

---

**Description**

The function DEMIPathway performs functional annotation analysis on DEMI differential expression results stored in the DEMIDiff object. It takes into account the number of up- and down-regulated targets as well as the total number of targets for each functional category to calculate the statistical significance of the functional annotation. PS! This function can only be used if in the underlying DEMIExperiment object the analysis parameter was set as 'gene' or 'transcript' for it will before functional annotation only on genes.

**Usage**

DEMIPathway(object = "DEMIDiff")

**Arguments**

object A DEMIDiff object. The DEMIDiff object contains the results to differential expression analysis that will be used for functional annotation analysis.

**Value**

Returns the results of the functional annotation analysis in a data.frame.
Examples

```r
## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/"

data files (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste(destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ))
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste(destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ))
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste(destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ))
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste(destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ))
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/" ),
destfile = paste(destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "" ))
download.file( paste( ftpaddress, "GSM247679/suppl/GSM247679.CEL.gz", sep = "/" ),
destfile = paste(destfolder, "BRAIN02_GSM247679.CEL.gz", sep = "" ))
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste(destfolder, "BRAIN03_GSM247700.CEL.gz", sep = "" ))
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/" ),
destfile = paste(destfolder, "BRAIN04_GSM247701.CEL.gz", sep = "" ))

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library(R.utils)
for( i in list.files( destfolder ) ) {
gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function demi

# Set up an experiment
demiexp <- DEMIExperiment(analysis = 'gene', celpath = destfolder,
experiment = 'myexperiment', organism = 'homo_sapiens')
```
# Create clusters with an optimized wilcoxon’s rank sum test incorporated within demi that
# precalculates the probabilities.
demicluster <- DEMIClust( demiexp, group = c( "BRAIN", "UHR" ), clust.method = demi.wilcox.test.fast )

calculate differential expression
demidiff <- DEMIDiff( demicluster )

# Perform functiona annotation analysis on the DEMI analysis results
demipath <- DEMIPathway( demidiff )

# End(Not run)

---

**DEMIResult-class**  
*Class DEMIResult*

**Description**

The class DEMIResult holds the results of the specified clusters.

**Arguments**

- **group**  
  A DEMIGroup object. Holds the information about the groups that were used for clustering.

- **result**  
  A list. Holds the analysis results for each cluster.

**Author(s)**

Sten Ilmarv

**demisummary**  
*Returns the mean normalized expression levels for the specified targets*

**Description**

The function demisummary returns the mean normalized expression levels for the specified targets. It returns the mean expression values for the whole dataset as well as for individual groups. Depending on the analysis parameter of the underlying DEMIExperiment object the target can be ensembl gene ID or gene symbol (e.g. ‘MAOB’), ensembl transcript ID, ensembl peptide ID or genomic region ID.
Usage

demisummary(object, target)

## S4 method for signature 'DEMIDiff'
demisummary(object, target)

## S4 method for signature 'DEMIEperiment'
demisummary(object, target)

Arguments

object A DEMIExperiment, DEMIDiff object.
target A vector. Depending on the analysis the target can be ensembl gene ID or gene symbol (e.g. 'MAOB'), ensembl transcript ID, ensembl peptide ID or genomic region ID.

Details

To see available targets used in the analysis you can try head(getAnnotation(x)) where x is an object of class DEMIExperiment. Alternatively you could use head(getAnnotation(getExperiment(y))) where y is of class DEMIDiff.

If no results have been attached to the DEMIExperiment object then it only returns the mean normalized expression values for the whole dataset not for individual groups. To attach results to DEMIExperiment object use the function attachResult(x, y) where x is an object of class DEMIExperiment and y is an object of class DEMIDiff that stores the results.

Value

Returns the mean normalized expression levels of the specified targets.

Author(s)

Sten Ilmjarv

See Also

DEMIExperiment, DEMIDiff, attachResult

Examples

## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files fill be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/"
# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN02_GSM247697.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN03_GSM247700.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN04_GSM247701.CEL.gz", sep = "" ) )

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library( R.utils )
for( i in list.files( destfolder ) ) {
gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function demisummary

# Set up an experiment
demiexp <- DEMIExperiment( analysis = 'gene', celpath = destfolder,
experiment = 'myexperiment', organism = 'homo_sapiens' )

# Create clusters with an optimized wilcoxon's rank sum test incorporated within demi that
# precalculates the probabilities
demiclust <- DEMIClust( demiexp, group = c( "BRAIN", "UHR" ), clust.method = demi.wilcox.test.fast )

# Calculate differential expression
demidiff <- DEMIDiff( demiclust )

# Retrieve the mean normalized expression values for the specified targets
demisummary( demiexp, c("MAOB") )
demisummary( demidiff, "MAOB" )

# Attach results from 'DEMIDiff' object to 'DEMIExperiment' object
demiexp_attached <- attachResult( demiexp, demidiff )

# Retrieve mean normalized expression values again and note these are also retrieved for specific
diffexp

# groups
demisummary( demiexp_attached, "MAQB" )

## End(Not run)

diffexp

*Initializes the differential expression analysis*

**Description**

The function `diffexp` performs differential expression analysis. This function is used internally by DEMI analysis.

**Usage**

diffexp(object)

## S4 method for signature 'DEMIDiff'
diffexp(object)

**Arguments**

- `object` A DEMIDiff object.

**Value**

Returns the DEMIDiff object that is updated with the results from differential expression analysis.

**Author(s)**

Sten Ilmjarv

**See Also**

DEMIDiff
diffSpliceScore  Calculate differential splice scores

Description

The function diffSpliceScore calculates differential splice scores in DEMI analysis. It is used internally in DEMI analysis. In the current implementation of DEMI it is not used.

Usage

diffSpliceScore(x)

Arguments

x       A data.frame.

default

Value

Returns the differential splice score as numeric.

Author(s)

Sten Ilmjarv

findCytoband  Finds cytoband for the specified genome region

Description

The function findCytoband finds cytoband for a genome region specified by the chromosome, region start and region end coordinates. It is used internally in DEMI analysis.

Usage

findCytoband(x, cytoband = "data.frame")

Arguments

x       A vector. A vector of "chr", "start" and "end" information about the genome region.

cytoband  A data.frame. A data.frame containing karyotype information.

default

Value

A karyotype character corresponding to the input genomic region.
getAlignment

Author(s)
Sten Ilmjarv

getAlignment  Returns the blatTable parameter representing alignment information

Description
Returns the blatTable of the DEMIExperiment object. It is a data.frame that stores the alignment information (such as probe matches on targets) used in DEMI analysis.

Usage
getAlignment(object)

## S4 method for signature 'DEMIExperiment'
getAlignment(object)

Arguments
object  A DEMIExperiment object.

Value
Returns the blatTable parameter of the DEMIExperiment object that is a data.frame.

Author(s)
Sten Ilmjarv

See Also
DEMIExperiment

Examples
## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "deimtest/testdata/"

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
getAnalysis

Returns the analysis parameter

Description

Returns the analysis parameter of the DEMIExperiment object. It is a character that represents the type of DEMI analysis. It can be either 'gene', 'transcript', 'exon' or 'genome'.

# It is good practice to name the files according to their features which allows easier identification of the files later.

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/") ,
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/") ,
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/") ,
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/") ,
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/") ,
destfile = paste( destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/") ,
destfile = paste( destfolder, "BRAIN02_GSM247697.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/") ,
destfile = paste( destfolder, "BRAIN03_GSM247700.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/") ,
destfile = paste( destfolder, "BRAIN04_GSM247701.CEL.gz", sep = "" ) )

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library( R.utils )
for( i in list.files( destfolder ) ) {
gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function getAlignment

# Set up an experiment
demiexp <- DEMIExperiment( analysis = 'gene', celpath = destfolder ,
experiment = 'myexperiment', organism = 'homo_sapiens' )

# Retrieve the 'blatTable' parameter representing alignment information
head( getAlignment( demiexp ) )

## End(Not run)
getAnalysis

Usage

getAnalysis(object)

## S4 method for signature 'DEMIExperiment'
getAnalysis(object)

Arguments

object A DEMIExperiment object.

Value

Returns the analysis parameter of the DEMIExperiment object that is a character.

Author(s)

Sten Ilmjarv

See Also

DEMIExperiment

Examples

## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR05_GSM247696.CEL.gz", sep = "" ) )

destfile = paste( destfolder, "UHR06_GSM247696.CEL.gz", sep = "" )

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR05_GSM247696.CEL.gz", sep = "" ) )

destfile = paste( destfolder, "UHR06_GSM247696.CEL.gz", sep = "" )
getAnnotation

Returns the annoTable parameter representing annotation information.

Description

Returns the annoTable of the DEMIExperiment object. It is a data.frame that stores the information about target annotations (such as it’s ID’s and description) used in DEMI analysis.

Usage

getAnnotation(object)

## S4 method for signature 'DEMIExperiment'
getAnnotation(object)

Arguments

object A DEMIExperiment object.

Value

Returns the annoTable parameter of the DEMIExperiment object that is a data.frame.
getAnnotation

Author(s)
Sten Ilmjarv

See Also
demiExperiment

Examples

## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/"

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN02_GSM247697.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN03_GSM247700.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN04_GSM247701.CEL.gz", sep = "" ) )

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library( R.utils )
for( i in list.files( destfolder ) ) {
gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function getAnnotation
getArraytype

# Set up an experiment
demiexp <- DEMIExperiment( analysis = 'gene', celpath = destfolder, experiment = 'myexperiment', organism = 'homo_sapiens' )

# Retrieve the 'annotation' parameter representing annotation information
ead( getAnnotation( demiexp ) )

## End(Not run)

getArraytype  

**Returns the arraytype parameter**

---

**Description**

Returns the arraytype parameter of the DEMIExperiment object. It is a character that represents the microarray platform used in DEMI analysis.

**Usage**

getArraytype(object)

## S4 method for signature 'DEMIExperiment'
getArraytype(object)

**Arguments**

object A DEMIExperiment object.

**Value**

Returns the arraytype parameter of the DEMIExperiment object that is a character.

**Author(s)**

Sten Ilmjarv

**Examples**

## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/"
getCelMatrix

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.
download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN01_GSM247697.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN02_GSM247700.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN03_GSM247701.CEL.gz", sep = "" ) )

destfolder <- "GSM47701"
destfile <- paste( destfolder, "myexperiment.CEL.gz", sep = "" )
deretrievefile( destfile, force = TRUE )

destfile <- paste( destfolder, "myexperiment.ArrayType", sep = "" )
deretrievefile( destfile, force = TRUE )

cleanup( destfile )

cleanup( destfolder )

getCelMatrix

Returns the raw expression matrix

Description

Returns the raw expression matrix of the DEMIExperiment object. It is a matrix where column
names indicate different file names and row names indicate probe ID’s.
Usage

getCelMatrix(object)

## S4 method for signature 'DEMIExperiment'
getCelMatrix(object)

Arguments

object A DEMIExperiment object.

Value

Returns the raw expression matrix of the DEMIExperiment object that is a matrix.

Author(s)

Sten Ilmjarv

See Also

DEMIExperiment, DEMICel

Examples

## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "" ) )
getCelpath

Returns the celpath parameter

Description

Returns the celpath parameter of the DEMIExperiment object. It is a character that represents the directory where CEL files are located or is a vector of individual CEL files used in the DEMI analysis.

Usage

getCelpath(object)

## S4 method for signature 'DEMIExperiment'
getCelpath(object)

Arguments

object A DEMIExperiment object.

Value

Returns the celpath parameter of the DEMIExperiment object that is a character.
Author(s)
Sten Ilmjarv

See Also
demiexperiment

Examples

```r
## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/"

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.
download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN02_GSM247697.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN03_GSM247700.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN04_GSM247701.CEL.gz", sep = "" ) )

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library( R.utils )
for( i in list.files( destfolder ) ) {
  gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function getCePath
```
getCluster

# Set up an experiment
demiexp <- DEMIExperiment( analysis = 'gene', celpath = destfolder, experiment = 'myexperiment', organism = 'homo_sapiens' )

# Retrieve the 'celpath' parameter
getcelpath( demiexp )

## End(Not run)

getCluster  Returns the cluster parameter

Description

Returns the cluster parameter of the DEMIClust object. It is a list that represents clusters containing probes.

Usage

getCluster(object)

## S4 method for signature 'DEMIClust'
getCluster(object)

Arguments

object A DEMIClust object.

Value

Returns cutoff.pvalue parameter of the DEMIClust object.

Author(s)

Sten Ilmjarv

See Also

DEMIClust

Examples

## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.
# Set the destination folder where the downloaded files fill be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/"

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.
download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN02_GSM247697.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN03_GSM247700.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN04_GSM247701.CEL.gz", sep = "" ) )

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library( R.utils )
for( i in list.files( destfolder ) ) {
gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function getCluster

# Set up an experiment
demiexp <- DEMIExperiment( analysis = 'gene', celpath = destfolder,
experiment = "myexperiment", organism = "homo_sapiens" )

# Create clusters with an optimized wilcoxon's rank sum test incorporated within demi that
# precalculates the probabilities
demiclust <- DEMIClust( demiexp, group = c( "BRAIN", "UHR" ), clust.method = demi.wilcox.test.fast )

# Retrieve the 'cluster' parameter
getCluster( demiclust )

## End(Not run)
getClustMethod

Returns the clust.method parameter

Description

Returns the clust.method parameter of the DEMIClust object. It is a function that is used for clustering the probes into clusters.

Usage

getClustMethod(object)

## S4 method for signature 'DEMIClust'
getClustMethod(object)

Arguments

object A DEMIClust object.

Value

Returns the clust.method parameter of the DEMIClust object that is a function.

Author(s)

Sten Ilmjarv

See Also

DEMIClust

Examples

## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.
getCutoffPvalue

Returns the cutoff.pvalue parameter of the 'DEMIClust' object. It is used to determine the cutoff significance level of probe signalling when applying the clustering function.

Description

Returns the cutoff.pvalue parameter of the 'DEMIClust' object. It is used to determine the cutoff significance level of probe signalling when applying the clustering function.
Usage

getcutoffPvalue(object)

## S4 method for signature 'DEMIClust'
getcutoffPvalue(object)

Arguments

object A DEMIClust object.

Value

Returns the cutoffPvalue parameter of the 'DEMIClust' object.

Author(s)

Sten Ilmarj

See Also

DEMIClust

Examples

## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files fill be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/"

download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ),
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ),
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ),
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ),
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR05_GSM247700.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR06_GSM247701.CEL.gz", sep = "" ),
download.file( paste( ftpaddress, "GSM247702/suppl/GSM247702.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR07_GSM247702.CEL.gz", sep = "" ),
download.file( paste( ftpaddress, "GSM247703/suppl/GSM247703.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR08_GSM247703.CEL.gz", sep = "" ),
download.file( paste( ftpaddress, "GSM247704/suppl/GSM247704.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR09_GSM247704.CEL.gz", sep = "" ),
download.file( paste( ftpaddress, "GSM247705/suppl/GSM247705.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR10_GSM247705.CEL.gz", sep = "" ),
download.file( paste( ftpaddress, "GSM247706/suppl/GSM247706.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR11_GSM247706.CEL.gz", sep = "" ),
download.file( paste( ftpaddress, "GSM247707/suppl/GSM247707.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR12_GSM247707.CEL.gz", sep = "" ),
download.file( paste( ftpaddress, "GSM247708/suppl/GSM247708.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR13_GSM247708.CEL.gz", sep = "" ),
download.file( paste( ftpaddress, "GSM247709/suppl/GSM247709.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR14_GSM247709.CEL.gz", sep = "" ),
download.file( paste( ftpaddress, "GSM247710/suppl/GSM247710.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR15_GSM247710.CEL.gz", sep = "" ),
download.file( paste( ftpaddress, "GSM247711/suppl/GSM247711.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR16_GSM247711.CEL.gz", sep = "" ),
download.file( paste( ftpaddress, "GSM247712/suppl/GSM247712.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR17_GSM247712.CEL.gz", sep = "" ),
download.file( paste( ftpaddress, "GSM247713/suppl/GSM247713.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR18_GSM247713.CEL.gz", sep = "" ),
download.file( paste( ftpaddress, "GSM247714/suppl/GSM247714.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR19_GSM247714.CEL.gz", sep = "" ),
download.file( paste( ftpaddress, "GSM247715/suppl/GSM247715.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR20_GSM247715.CEL.gz", sep = "" ),
download.file( paste( ftpaddress, "GSM247716/suppl/GSM247716.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR21_GSM247716.CEL.gz", sep = "" ) )

# ...
getCytoband

Returns the cytoband parameter representing karyotype information

description

Returns the cytoband parameter of the DEMIExperiment object. It is a data.frame that stores the karyotype information of the chromosomes.

Usage

getCytoband(object)

Arguments

object A DEMIExperiment object.
getCytoband

Details

If the analysis parameter in DEMIExperiment object is set to 'genome' then genome sections are being annotated by their karyotypes. The annotation information is stored in the cytoband parameter of the DEMIExperiment object.

Value

Returns the cytoband parameter of the DEMIExperiment object that is a data.frame.

Author(s)

Sten Ilmjarv

See Also

DEMIExperiment

Examples

## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/"

download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN02_GSM247697.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN03_GSM247700.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN04_GSM247701.CEL.gz", sep = "" ) )
getDEMIClust

# We need the gunzip function (located in the R.utils package) to unpack the gz files.  
# Also we will remove the original unpacked files for we won't need them.  
library( R.utils )  
for( i in list.files( destfolder ) ) {  
gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )  
}  

# Now we can continue the example of the function getCytoband  

# Set up an experiment. Note that the cytoband can only retrieved when the analysis  
# has been set to genome.  
demiexp_genome <- DEMIExperiment( analysis = 'genome', celpath = destfolder,  
experiment = 'myexperiment', organism = 'homo_sapiens', sectionsize = 500000 )  

# Retrieve the 'cytoband' parameter representing karyotype information  
head( getCytoband( demiexp_genome ) )  

## End(Not run)

---

### getDEMIClust

**Returns the cluster parameter**

### Description

Returns the cluster of the DEMIdiff object. It is a DEMIClust object.

### Usage

```
getDEMIClust(object)
```

**## S4 method for signature 'DEMIdiff'**

```
getDEMIClust(object)
```

### Arguments

- `object` A DEMIdiff object.

### Value

Returns the cluster parameter of the DEMIdiff object which is a DEMIClust object.

### Author(s)

Sten Ilmjarv

### See Also

DEMIClust, DEMIdiff
Examples

## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitesterdata/"

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.
download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ))
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ))
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ))
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ))
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR05_GSM247700.CEL.gz", sep = "" ))
download.file( paste( ftpaddress, "GSM2477001/suppl/GSM2477001.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR06_GSM2477001.CEL.gz", sep = "" ))

download.file( paste( ftpaddress, "GSM2477002/suppl/GSM2477002.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR07_GSM2477002.CEL.gz", sep = "" ))
download.file( paste( ftpaddress, "GSM2477003/suppl/GSM2477003.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR08_GSM2477003.CEL.gz", sep = "" ))
download.file( paste( ftpaddress, "GSM2477004/suppl/GSM2477004.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR09_GSM2477004.CEL.gz", sep = "" ))

# We need the gunzip function (located in the R.utils package) to unpack the.gz files.
# Also we will remove the original unpacked files for we won't need them.
library(R.utils)
for( i in list.files( destfolder ) ) {
gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function getDEMIClust

# Set up an experiment
demiexp <- DEMIExperiment( analysis = 'gene', celpath = destfolder,
experiment = 'myexperiment', organism = 'homo_sapiens' )

# Create clusters with an optimized wilcoxon's rank sum test incorporated within demi that
# precalculates the probabilities
demiclust <- DEMIClust( demiexp, group = c( "BRAIN", "UHR" ), clust.method = demi.wilcox.test.fast )
getExperiment

Description

Gets the experiment parameter of the specified object. For object of class DEMIExperiment it returns the name given to the experiment. For objects of class DEMICluster or DEMIDiff it returns the initial DEMIExperiment object. This function can be useful if the user wants to access metadata, such as annotations and alignments from other DElM analysis objects. As well as accessing the name of the analysis.

Usage

getExperiment(object)

## S4 method for signature 'DEMICluster'
getExperiment(object)

## S4 method for signature 'DEMDiff'
getExperiment(object)

## S4 method for signature 'DEMIExperiment'
getExperiment(object)

Arguments

object A DEMIExperiment, DEMICluster or DEMIDiff object.

Value

Returns the experiment parameter. If the input object is DEMIExperiment it returns a character, if the input object is either DEMICluster or DEMIDiff it returns a DEMIExperiment object.

Author(s)

Sten Ilmarv

See Also

DEMIExperiment, DEMICluster, DEMIDiff
getExperiment

Examples

## Not run:

```
# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/"

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.
download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN02_GSM247697.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN03_GSM247700.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN04_GSM247701.CEL.gz", sep = "" ) )
```

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library(R.utils)
for( i in list.files( destfolder ) ) {
  gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function getExperiment

# Set up an experiment
demiexp <- DEMIExperiment( analysis = 'gene', celpath = destfolder,
experiment = 'myexperiment', organism = 'homo_sapiens' )

# Create clusters with an optimized wilcoxon's rank sum test incorporated within demi that
# precalculates the probabilities
demiclust <- DEMIClust( demiexp, group = c( "BRAIN", "UHR" ), clust.method = demi.wilcox.test.fast )
# Calculate differential expression
demidiff <- DEMIDiff( demiclust )

# Retrieve the 'experiment' parameter
getExperiment( demiexp )
getExperiment( demiclust )
getExperiment( demidiff )

## End(Not run)

---

**getGroup**  
*Returns the group parameter*

**Description**

Returns the group parameter which is a DEMIGroup object.

**Usage**

getGroup(object)

## S4 method for signature 'DEMIClust'
getGroup(object)

## S4 method for signature 'DEMIDiff'
getGroup(object)

## S4 method for signature 'DEMIResult'
getGroup(object)

**Arguments**

- **object**  
  A DEMIClust, DEMIDiff or DEMIResult object.

**Value**

Returns the group parameter that is a DEMIGroup object.

**Author(s)**

Sten Ilmjarv

**See Also**

DEMIClust, DEMIDiff, DEMIResult
getGroup

Examples

## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/"

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN02_GSM247697.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN03_GSM247700.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN04_GSM247701.CEL.gz", sep = "" ) )

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library(R.utils)
for( i in list.files( destfolder ) ) {
  gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function getGroup

# Set up an experiment
demiexp <- DEMIExperiment( analysis = 'gene', celpath = destfolder,
experiment = 'myexperiment', organism = 'homo_sapiens' )

# Create clusters with an optimized wilcoxon's rank sum test incorporated within demi that
# precalculates the probabilities
demiclust <- DEMIClust( demiexp, group = c( "BRAIN", "UHR" ), clust.method = demi.wilcox.test.fast )
getGroupA

returns the groupA parameter

Description

Returns the groupA parameter of the DEMIGroup object. It is a character that represents the name of group A.

Usage

groupA(object)

## S4 method for signature 'DEMIGroup'

groupA(object)

Arguments

object A DEMIGroup object.

Value

Returns the groupA parameter of the DEMIGroup object that is a character.

Author(s)

Sten Ilmjarv

See Also

DEMIGroup
getGroupA

Examples

```r
# Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/"

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.
download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN02_GSM247697.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN03_GSM247700.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN04_GSM247701.CEL.gz", sep = "" ) )

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library(R.utils)
for( i in list.files( destfolder ) ) {
  gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function getGroupA

# Set up an experiment
demiexp <- DEMIExperiment( analysis = 'gene', celpath = destfolder,
experiment = 'myexperiment', organism = 'homo_sapiens' )

# Create clusters with an optimized wilcoxon's rank sum test incorporated within demi that
# precalculates the probabilities
demiclust <- DEMIClust( demiexp, group = c( "BRAIN", "UHR" ), clust.method = demi.wilcox.test.fast )
```
getGroupB

Description

Returns the groupB parameter of the DEMIGroup object. It is a character that represents the name of group B.

Usage

getGroupB(object)

## S4 method for signature 'DEMIGroup'
getGroupB(object)

Arguments

object A DEMIGroup object.

Value

Returns the groupB parameter of the DEMIGroup object that is a character.

Author(s)

Sten Ilmjarv

See Also

DEMIGroup

Examples

## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.
# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.
download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN02_GSM247697.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN03_GSM247700.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN04_GSM247701.CEL.gz", sep = "" ) )

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library( R.utils )
for( i in list.files( destfolder ) ) {
  gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function getGroupB

# Set up an experiment
demiexp <- DEMIExperiment( analysis = 'gene', celpath = destfolder,
experiment = 'myexperiment', organism = 'homo_sapiens' )

# Create clusters with an optimized wilcoxon's rank sum test incorporated within demi that
# precalculates the probabilities
demiclust <- DEMIClust( demiexp, group = c( "BRAIN", "UHR" ), clust.method = demi.wilcox.test.fast )

# Calculate differential expression
demidiff <- DEMIDiff( demiclust )

# Get group B name
getGroupB( getGroup( demiclust ) )
getGroupB( getGroup( demidiff ) )
### Description

Returns the groupNames parameter of the DEMIGroup object. It is a character that represent the custom group names. The groupNames parameter is only stored when the user defines his/her own clusters.

### Usage

```r
getGroupNames(object)
```

```
# S4 method for signature 'DEMIGroup'
getGroupNames(object)
```

### Arguments

- `object` A DEMIGroup object.

### Value

Returns the groupNames parameter of the DEMIGroup object that is a character.

### Author(s)

Sten Ilmjarv

### See Also

DEMIGroup

### Examples

```r
## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files fill be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/"

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
```
# It is good practice to name the files according to their features which
# allows easier identification of the files later.

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN02_GSM247697.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN03_GSM247700.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN04_GSM247701.CEL.gz", sep = "" )

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library( R.utils )
for( i in list.files( destfolder ) ) {
  gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function getGroupNames

# Set up an experiment
demiexp <- DEMIExperiment( analysis = 'gene', celpath = destfolder,
experiment = 'myexperiment', organism = 'homo_sapiens' )

# Define a custom 'DEMIClust' object with user defined clusters
demiclust_custom <- DEMIClust( demiexp, cluster = list( customcluster = c(1190, 1998, 2007) ) )

# Calcuate differential expression
demidiff_custom <- DEMIDiff( demiclust_custom )

# Get group B name
getGroupNames( getGroup( demiclust_custom ) )
getGroupNames( getGroup( demidiff_custom ) )

## End(Not run)
getIndexA

Description

Returns the indexA parameter of the DEMIGroup object. It is a numeric that represents the column indexes of group A.

Usage

getIndexA(object)

## S4 method for signature 'DEMIGroup'
getIndexA(object)

Arguments

object A DEMIGroup object.

Value

Returns the indexA parameter of the DEMIGroup object that is a numeric.

Author(s)

Sten Ilmjarv

See Also

DEMIGroup

Examples

## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/"

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
C we need the gunzip function (located in the R.utils package) to unpack the gz files.
C also we will remove the original unpacked files for we won't need them.
library(R.utils)
for( i in list.files( destfolder ) ) {
gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function getIndexA

# Set up an experiment
demiexp <- DEMIExperiment( analysis = 'gene', celpath = destfolder, 
experiment = 'myexperiment', organism = 'homo_sapiens' )

# Create clusters with an optimized wilcoxon's rank sum test incorporated within demi that 
# precalculates the probabilities
demiclust <- DEMIClust( demiexp, group = c("BRAIN", "UHR"), clust.method = demi.wilcox.test.fast )

# Calcluate differential expression
demidiff <- DEMIDiff( demiclust )

# Get group A indexes
groupA <- getGroup( demiclust )
groupB <- getGroup( demidiff )

## End(Not run)

---

**getIndexB**

Returns the `indexB` parameter

---

**Description**

Returns the `indexB` parameter of the `DEMIGroup` object. It is a numeric that represents the column indexes of group B.

**Usage**

`getIndexB(object)`
## `getIndexB`

### Arguments

- **object**: A DEMIGroup object.

### Value

Returns the `indexB` parameter of the DEMIGroup object that is a numeric.

### Author(s)

Sten Ilmjarv

### See Also

DEMIGroup

### Examples

```r
## S4 method for signature 'DEMIGroup'
getIndexB(object)

Arguments

- **object**: A DEMIGroup object.

Value

Returns the `indexB` parameter of the DEMIGroup object that is a numeric.

Author(s)

Sten Ilmjarv

See Also

DEMIGroup

Examples

```r
## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/"),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/"),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/"),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/"),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/"),
destfile = paste( destfolder, "BRAIN01_GSM247700.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/"),
destfile = paste( destfolder, "BRAIN02_GSM247701.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247702/suppl/GSM247702.CEL.gz", sep = "/"),
destfile = paste( destfolder, "BRAIN03_GSM247702.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247703/suppl/GSM247703.CEL.gz", sep = "/"),
destfile = paste( destfolder, "BRAIN04_GSM247703.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247704/suppl/GSM247704.CEL.gz", sep = "/"),
destfile = paste( destfolder, "BRAIN05_GSM247704.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247705/suppl/GSM247705.CEL.gz", sep = "/"),
destfile = paste( destfolder, "BRAIN06_GSM247705.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247706/suppl/GSM247706.CEL.gz", sep = "/"),
destfile = paste( destfolder, "BRAIN07_GSM247706.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247707/suppl/GSM247707.CEL.gz", sep = "/"),
destfile = paste( destfolder, "BRAIN08_GSM247707.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247708/suppl/GSM247708.CEL.gz", sep = "/"),
destfile = paste( destfolder, "BRAIN09_GSM247708.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247709/suppl/GSM247709.CEL.gz", sep = "/"),
destfile = paste( destfolder, "BRAIN10_GSM247709.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247710/suppl/GSM247710.CEL.gz", sep = "/"),
destfile = paste( destfolder, "BRAIN11_GSM247710.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247711/suppl/GSM247711.CEL.gz", sep = "/"),
destfile = paste( destfolder, "BRAIN12_GSM247711.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247712/suppl/GSM247712.CEL.gz", sep = "/"),
destfile = paste( destfolder, "BRAIN13_GSM247712.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247713/suppl/GSM247713.CEL.gz", sep = "/"),
destfile = paste( destfolder, "BRAIN14_GSM247713.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247714/suppl/GSM247714.CEL.gz", sep = "/"),
destfile = paste( destfolder, "BRAIN15_GSM247714.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247715/suppl/GSM247715.CEL.gz", sep = "/"),
destfile = paste( destfolder, "BRAIN16_GSM247715.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247716/suppl/GSM247716.CEL.gz", sep = "/"),
destfile = paste( destfolder, "BRAIN17_GSM247716.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247717/suppl/GSM247717.CEL.gz", sep = "/"),
destfile = paste( destfolder, "BRAIN18_GSM247717.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247718/suppl/GSM247718.CEL.gz", sep = "/"),
destfile = paste( destfolder, "BRAIN19_GSM247718.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247719/suppl/GSM247719.CEL.gz", sep = "/"),
destfile = paste( destfolder, "BRAIN20_GSM247719.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247720/suppl/GSM247720.CEL.gz", sep = "/")
```
```
getMaxprobes

Returns the maxprobes parameter

Description

Returns the maxprobes of the DEMIExperiment object. It is a character that represents the maximum number of probes a target is allowed to have a match against in DEMI analysis.

Usage

getMaxprobes(object)

## S4 method for signature 'DEMIExperiment'
getMaxprobes(object)

Arguments

object A DEMIExperiment object.
Details

If the maxprobes in DEMIExperiment object is set to 'median' or some integer larger than 0, then all targets that yield more alignments to different probes then defined by maxprobes will be scaled down to the number set in the maxprobes parameter.

Value

Returns the maxprobes parameter of the DEMIExperiment object.

Author(s)

Sten Ilmjarv

See Also

DEMIExperiment

Examples

```r
## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/"

destfolder <- "demitest/testdata/"

destfolder <- "demitest/testdata/"

destfolder <- "demitest/testdata/"

destfolder <- "demitest/testdata/"

destfolder <- "demitest/testdata/"

destfolder <- "demitest/testdata/"

destfolder <- "demitest/testdata/"

destfolder <- "demitest/testdata/"

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN02_GSM247697.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN03_GSM247700.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN04_GSM247701.CEL.gz", sep = "" ) )
```
# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library( R.utils )
for( i in list.files( destfolder ) ) {
  gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function maxprobes

demiexp <- DEMIExperiment( analysis = 'gene', celpath = destfolder,
                           experiment = 'myexperiment', organism = 'homo_sapiens' )

getMaxtargets( demiexp )

## End(Not run)

---

**getMaxtargets**

**Returns the maxtargets parameter**

**Description**

Returns the maxtargets parameter of the DEMIExperiment object. It is a numeric that represents the maximum number of allowed targets (e.g. genes or transcripts) a probe can have a match against.

**Usage**

getMaxtargets(object)

## S4 method for signature 'DEMIExperiment'

getMaxtargets(object)

**Arguments**

object A DEMIExperiment object.

**Details**

If the analysis in DEMIExperiment object is set to 'transcript' the program still calculates the number of matches on genes. Hence a probe matching two transcripts on the same gene would be included but a probe matching two transcripts on different genes would not be included if maxtargets would be set to 1.

**Value**

Returns the maxtargets parameter of the DEMIExperiment object that is a numeric.
Example

## Not run:

```r
# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/"

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.
download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN02_GSM247697.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN03_GSM247700.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN04_GSM247701.CEL.gz", sep = "" ) )

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library( R.utils )
for( i in list.files( destfolder ) ) {
  gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function getMaxtargets

# Set up an experiment
demiexp <- DEMIExperiment( analysis = 'gene', celpath = destfolder,
experiment = 'myexperiment', organism = 'homo_sapiens' )
```
getName

# Retrieve the 'maxtargets' parameter
getMaxtargets( demiexp )

## End(Not run)

getName

*Returns the name parameter*

**Description**

Returns the name parameter of the DEMIDiff object. It is a character that represents the name of the differential expression analysis stored in the DEMIDiff object.

**Usage**

getName(object)

## S4 method for signature 'DEMIDiff'

getName(object)

**Arguments**

- **object**: A DEMIDiff object.

**Value**

Returns the name parameter of the DEMIDiff object which is a character.

**Author(s)**

Sten Ilmjarv

**See Also**

DEMIDiff

**Examples**

## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files fill be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/"
# Download packed CEL files and change the names according to the feature they represent (for example to include UHR or BRAIN in them to denote the features).
# It is good practice to name the files according to their features which allows easier identification of the files later.

```r
download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247696.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN01_GSM247697.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN02_GSM247700.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN03_GSM247701.CEL.gz", sep = "" ) )

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library( R.utils )
for( i in list.files( destfolder ) ) {
gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function getNormMatrix

# Set up an experiment
demiexp <- DEMIExperiment( analysis = 'gene', celpath = destfolder,
experiment = 'myexperiment', organism = 'homo_sapiens' )

# Create clusters with an optimized wilcoxon's rank sum test incorporated within demi that precalculates the probabilities
demiclust <- DEMIClustering( demiexp, group = c( "BRAIN", "UHR" ), clust.method = demi.wilcox.test.fast )

# Calculate differential expression
demidiff <- DEMIDifferential( demiclust )

# Retrieve the 'cluster' parameter that is 'DEMIClustering' object
getName( demidiff )

## End(Not run)
```

getNormMatrix

Returns the normalized expression matrix
Description

Returns the normalized expression matrix of the DEMIExperiment object. It is a matrix where column names indicate different file names and row names indicate probe ID’s.

Usage

getNormMatrix(object)

## S4 method for signature 'DEMIExperiment'
getNormMatrix(object)

Arguments

object
A DEMIExperiment object.

Value

Returns the normalized expression matrix of the DEMIExperiment object that is a matrix.

Author(s)

Sten Ilmjarv

See Also

DEMIExperiment, DEMICel

Examples

## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/"

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
getOrganism

Returns the organism parameter

Description

Returns the organism parameter of the DEMIExperiment object. It is a character that represents the species used in DEMI analysis.

Usage

getOrganism(object)

Arguments

object       A DEMIExperiment object.
getOrganism

Value

Returns the organism parameter of the DEMIExperiment object that is a character.

Author(s)

Sten Ilmjarv

See Also

DEMIExperiment

Examples

```r
## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files fill be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/"

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.
download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR05_GSM247700.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR06_GSM247701.CEL.gz", sep = "" ) )

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library( R.utils )
for( i in list.files( destfolder ) ) {
  gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}
```
getPathway

# Now we can continue the example of the function getOrganism

# Set up an experiment
demiexp <- DEMIExperiment( analysis = 'gene', celpath = destfolder, experiment = 'myexperiment', organism = 'homo_sapiens' )

# Retrieve the 'organism' parameter
getOrganism( demiexp )

## End(Not run)

---

**getPathway**

*Returns the pathway parameter representing functional annotation information*

**Description**

Returns the pathway parameter of the DEMIExperiment object. It is a data.frame that stores information about gene ontology categories.

**Usage**

getPathway(object)

## S4 method for signature 'DEMIExperiment'
ggetPathway(object)

**Arguments**

- **object**
  - A DEMIExperiment object.

**Details**

The information about gene ontology categories is used when the user runs pathway analysis on DEMI differential expression results with the function DEMIPathway.

**Value**

Returns the pathway parameter of the DEMIExperiment object that is a data.frame.

**Author(s)**

Sten Ilmjarv

**See Also**

DEMIExperiment, DEMIPathway
**Examples**

```r
## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

default <- ~demitest/testdata/

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.
download(file= paste(ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfolder = paste(destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ))
download(file= paste(ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfolder = paste(destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ))
download(file= paste(ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfolder = paste(destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ))
download(file= paste(ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfolder = paste(destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ))
download(file= paste(ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/" ),
destfolder = paste(destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "" ))
download(file= paste(ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/" ),
destfolder = paste(destfolder, "BRAIN02_GSM247697.CEL.gz", sep = "" ))
download(file= paste(ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfolder = paste(destfolder, "BRAIN03_GSM247700.CEL.gz", sep = "" ))
download(file= paste(ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/" ),
destfolder = paste(destfolder, "BRAIN04_GSM247701.CEL.gz", sep = "" ))

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library(R.utils)
for( i in list.files( destfolder ) ){
gunzip(paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function getPathway. Note that pathway can only
# be retrieved if the analysis is set to gene or transcript.

# Set up an experiment
demiexp <- DEMIExperiment( analysis = 'gene', celpath = destfolder,
experiment = 'myexperiment', organism = 'homo_sapiens' )

# Retrieve the 'pathway' parameter representing functional annotation information
head( getPathway( demiexp ) )
```
getProbeLevel

Returns the probe levels from the normalized expression matrix for the specified probes

Description

The function getProbeLevel returns the probe levels in the normalized expression matrix specified by the probe ID’s.

Usage

getProbeLevel(object, probes, verbose)

## S4 method for signature 'DEMIDiff,vector,logical'
getProbeLevel(object, probes, verbose)

## S4 method for signature 'DEMIExperiment,vector,logical'
getProbeLevel(object, probes,
   verbose = TRUE)

Arguments

object         A DEMIExperiment or DEMIDiff object.
probes         A vector. A vector of probe ID’s whose expression levels should be returned.
verbose        A logical. If TRUE it will print out the probe ID’s that were not found in normalized expression matrix.

Details

To see what are the available probes in the normalized expression matrix you can try row.names(getNormMatrix(x)) where x is an object of class DEMIExperiment.

Value

Returns the probe levels in the normalized expression matrix for the specified probes.

Author(s)

Sten Ilmjarv

See Also

DEMIExperiment, DEMIDiff
getProbeLevel

Examples

```r
## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN02_GSM247697.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN03_GSM247700.CEL.gz", sep = "" )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN04_GSM247701.CEL.gz", sep = "" )

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library(R.utils)
for( i in list.files( destfolder ) ) {
  gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function getProbeLevel

# Set up an experiment
demiexp <- DEMIExperiment( analysis = 'gene', celpath = destfolder,
experiment = 'myexperiment', organism = 'homo_sapiens' )

# Create clusters with an optimized wilcoxon's rank sum test incorporated within demi that
# precalculates the probabilities
demiclust <- DEMIClust( demiexp, group = c("BRAIN", "UHR"), clust.method = demi.wilcoxon.test.fast )
```
# Calculate differential expression
demidiff <- DEMIDiff( demiclust )

# Retrieve the probe levels specified by probe ID's of the normalized expression matrix
getProbeLevel( demiexp, c( 1171, 1182 ), TRUE )
getProbeLevel( demidiff, c( 1171, 1182 ), TRUE )

### End(Not run)

---

**getResult**

*Returns the result parameter*

---

**Description**

Returns the result parameter stored in the specified object. If the object is of class DEMIExperiment then it returns a list of DEMIResult objects. If the object is of class DEMIDiff then it returns only one DEMIResult object. But if the object is of class DEMIResult then the function returns a list that contains the results for every cluster in a data.frame. However instead of using this function it maybe easier to use the function getResultTable that returns the result parameter as a data.frame.

**Usage**

```r
getResult(object)
```

### S4 method for signature 'DEMIDiff'

```r
getResult(object)
```

### S4 method for signature 'DEMIExperiment'

```r
getResult(object)
```

### S4 method for signature 'DEMIResult'

```r
getResult(object)
```

**Arguments**

- `object` A DEMIExperiment, DEMIDiff or DEMIResult object.

**Value**

Returns the result parameter. For objects of class DEMIExperiment it returns a list of DEMIResult objects. For objects of class DEMIDiff it returns a single DEMIResult object and for objects of class DEMIResult it returns a list.

**Author(s)**

Sten Ilmjarv
See Also

DEMIExperiment, DEMIDiff, DEMIResult, getResultTable

Examples

```r
## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/"

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = " " )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = " " )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = " " )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = " " )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN01_GSM247696.CEL.gz", sep = " " )
download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN02_GSM247697.CEL.gz", sep = " " )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN03_GSM247700.CEL.gz", sep = " " )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN04_GSM247701.CEL.gz", sep = " " )

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library( R.utils )
for( i in list.files( destfolder ) ) {
  gunzip( paste( destfolder, i, sep = " " ), remove = TRUE )
}

# Now we can continue the example of the function getResult

# Set up an experiment
demiexp <- DEMIExperiment( analysis = 'gene', celpath = destfolder,
  experiment = 'myexperiment', organism = 'homo_sapiens' )
```
# Create clusters with an optimized wilcoxon's rank sum test incorporated within demi that # precalculates the probabilities
demiclust <- DEMIClust( demiexp, group = c( "BRAIN", "UHR" ), clust.method = demi.wilcox.test.fast )

# Calculcate differential expression
demidiff <- DEMIDiff( demiclust )

# Attach results from 'DEМIDiff' object to 'DEМIExperiment' object
demiexp_attached <- attachResult( demiexp, demidiff )

# Retrieve the 'result' parameter
getResult( demiexp_attached )
getResult( demidiff )

### End(Not run)

---

getResultTable

**Retruns the DEMI analysis results as a data.frame**

### Description

The function getResultTable returns the DEMI analysis results as a data.frame. It retrieves the result parameter of the specified object with the function getResult and converts it into a data.frame for convenient viewing.

### Usage

getResultTable(object)

### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>A DEMIExperiment or DEMIDiff object.</td>
</tr>
</tbody>
</table>

### Value

Returns the result parameter of the specified object as a data.frame.

### Author(s)

Sten Ilmjarv
See Also
demiexperiment, DEMIDiff, getResult

Examples

## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files fill be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.


download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694 CEL.gz", sep = "\" " ),
destfile = paste( destfolder, "UHR01_GSM247694. CEL.gz", sep = "\" " )

download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695. CEL.gz", sep = "\" " ),
destfile = paste( destfolder, "UHR02_GSM247695. CEL.gz", sep = "\" " )

download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698. CEL.gz", sep = "\" " ),
destfile = paste( destfolder, "UHR03_GSM247698. CEL.gz", sep = "\" " )

download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699. CEL.gz", sep = "\" " ),
destfile = paste( destfolder, "UHR04_GSM247699. CEL.gz", sep = "\" " )

download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696. CEL.gz", sep = "\" " ),
destfile = paste( destfolder, "UHR01_GSM247696. CEL.gz", sep = "\" " )

download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697. CEL.gz", sep = "\" " ),
destfile = paste( destfolder, "UHR02_GSM247697. CEL.gz", sep = "\" " )

download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700. CEL.gz", sep = "\" " ),
destfile = paste( destfolder, "UHR03_GSM247700. CEL.gz", sep = "\" " )

download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701. CEL.gz", sep = "\" " ),
destfile = paste( destfolder, "UHR04_GSM247701. CEL.gz", sep = "\" " )

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library( R.utils )
for( i in list.files( destfolder ) ) {
  gunzip( paste( destfolder, i, sep = "\" " ), remove = TRUE )
}

# Now we can continue the example of the function getResultTable

# Set up an experiment
demiexp <- DEMIExperiment( analysis = 'gene', celpath = destfolder,
experiment = 'myexperiment', organism = 'homo_sapiens' )
getTargetAnnotation

getTargetAnnotation(object, target)

--

Description

Returns annotation information for the specified targets from a DEMIExperiment object. Depending on the analysis parameter in the DEMIExperiment object the target parameter can be an ensembl gene ID or gene symbol (e.g. 'MAOB'), ensembl transcript ID, ensembl peptide ID or genomic region ID.

Usage

gtargetAnnotation(object, target)

## S4 method for signature 'DEMIExperiment,vector'
gtargetAnnotation(object, target)

Arguments

- **object**: A DEMIExperiment object.
- **target**: A vector. A vector of targets whose annotation information should be returned. Depending on the analysis the target can be ensembl gene ID or gene symbol (e.g. 'MAOB'), ensembl transcript ID, ensembl peptide ID or genomic region ID.

Details

To see available targets used in the analysis you can try head(getAnnotation(x)) where x is an object of class DEMIExperiment.
**getTargetAnnotation**

**Value**

Returns annotation information from DEMIExperiment object specified by the targets.

**Author(s)**

Sten Ilmjarv

**See Also**

DEMIExperiment

**Examples**

```r
## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

destfolder <- "demitest/testdata/"

destfile <- "uhr1.gz"

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEIL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEIL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEIL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEIL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEIL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEIL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEIL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEIL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEIL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN01_GSM247700.CEIL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEIL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN02_GSM247701.CEIL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247702/suppl/GSM247702.CEIL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN03_GSM247702.CEIL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247703/suppl/GSM247703.CEIL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN04_GSM247703.CEIL.gz", sep = "" ) )

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library( R.utils )
for( i in list.files( destfolder ) ) {
  gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
```
getTargetProbes

Returns the probe ID's of the specified targets

Description

The function `getTargetProbes` returns the probe ID's of the specified targets. Depending on the analysis parameter in the underlying DEMIExperiment object the target parameter can be an ensembl gene ID or gene symbol (e.g. 'MAOB'), ensembl transcript ID, ensembl peptide ID or genomic region ID.

Usage

`getTargetProbes(object, target)`

### S4 method for signature 'DEMIExperiment,vector'

`getTargetProbes(object, target)`

### S4 method for signature 'DEMIExperiment,vector'

`getTargetProbes(object, target)`

Arguments

- **object**
  - A DEMIExperiment or DE MiDiff object.
- **target**
  - A vector. A vector of targets whose probe ID's should be returned. Depending on the analysis the target can be ensembl gene ID or gene symbol (e.g. 'MAOB'), ensembl transcript ID, ensembl peptide ID or genomic region ID.

Details

To see available targets used in the analysis you can try `head(getAnnotation(x))` where `x` is an object of class DEMIExperiment. Alternatively you could use `head(getAnnotation(getExperiment(y)))` where `y` is of class DE MiDiff.
**getTargetProbes**

**Value**

Returns the probes ID's specified by the targets.

**Author(s)**

Sten Ilmjarv

**See Also**

DEMIEperiment, DEMIDiff

**Examples**

```r
## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/")
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ))
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" )
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ))
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" )
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ))
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/")
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ))
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/"

destfile = paste( destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "" ))
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/

destfile = paste( destfolder, "BRAIN02_GSM247700.CEL.gz", sep = "" ))
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/

destfile = paste( destfolder, "BRAIN03_GSM247701.CEL.gz", sep = "" ))

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library( R.utils )
for( i in list.files( destfolder ) ) {
gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}
```
# Now we can continue the example of the function `getTargetProbes`

# Set up an experiment
demiexp <- DEMIExperiment( analysis = 'gene', celpath = destfolder, experiment = 'myexperiment', organism = 'homo_sapiens' )

# Create clusters with an optimized wilcoxon's rank sum test incorporated within demi that
# precalculates the probabilities
demicluster <- DEMIClust( demiexp, group = c( "BRAIN", "UHR" ), clust.method = demi.wilcox.test.fast )

# Calculate differential expression
demidiff <- DEMIDiff( demicluster )

# Retrieve the probe ID's of the specified targets
getTargetProbes( demiexp, "MAOB" )
getTargetProbes( demidiff, "MAOB" )

### End(Not run)

---

**initialize.DEMICel**  
*Initializes a DEMICel object*

**Description**

Initializes a DEMICel object.

**Usage**

```r
initialize.DEMICel(.Object, ...)
```

**Arguments**

- `.Object`  
  A DEMICel object.

- `...`  
  Additional arguments that may never be used.

**Value**

Returns a DEMICel object

**Author(s)**

Sten Ilmjarv
initialize.DEMIClust  

Initializes the DEMIClust object.

Usage

initialize.DEMIClust(.Object, ...)

Arguments

:Object  A DEMIClust object.
...      Additional arguments that may never be used.

Value

Returns a DEMIClust object.

Author(s)

Sten Ilmjarv

initialize.DEMIDiff  

Initializes the DEMIDiff object.

Usage

initialize.DEMIDiff(.Object, ...)

Arguments

:Object  A DEMIDiff object.
...      Additional arguments that may never be used.

Value

Returns a 'DEMDiff' object.

Author(s)

Sten Ilmjarv
initialize.DEMIExperiment

Initializes the DEMIExperiment object

Description

Initializes the DEMIExperiment object.

Usage

initialize.DEMIExperiment(.Object, ...)

Arguments

:Object A DEMIExperiment object.

... Additional arguments that may never be used.

Value

Returns a 'DEMIExperiment' object.

Author(s)

Sten Ilmjarv

initialize.DEMIGroup

Initializes the DEMIGroup object

Description

Initializes the DEMIGroup object.

Usage

initialize.DEMIGroup(.Object, ...)

Arguments

:Object A DEMIGroup object.

... Additional arguments that may never be used.

Value

Returns a 'DEMIGroup' object.

Author(s)

Sten Ilmjarv
initialize.DEMIResult  Initializes the DEMIResult object

Description
Initialize the DEMIResult object.

Usage
initialize.DEMIResult(.Object, ...)

Arguments
/Object A DEMIResult object.
... Additional arguments that may never be used.

Value
Returns a 'DEMIResult' object.

Author(s)
Sten Ilmjarv

loadAnnotation  Loads the annotation information specified by the DEMIExperiment object

Description
The function loadAnnotation loads the annotation information for the specified DEMIExperiment object. It is used internally in DEMI analysis.

Usage
loadAnnotation(object, pkg)

## S4 method for signature 'DEMIExperiment,environment'
loadAnnotation(object, pkg)

Arguments
/object A DEMIExperiment object.
pkg An environment. Specifies the environment where to load the data from.
Value

Returns a data.frame with annotation information.

Author(s)

Sten Ilmjarv

See Also

DEMIExperiment, environment

Description

The function `loadBlat` loads the alignment information for the specified DEMIExperiment object. It is used internally in DEMI analysis.

Usage

```r
loadBlat(object, pkg)
```

```r
## S4 method for signature 'DEMIExperiment,environment'
loadBlat(object, pkg)
```

Arguments

- `object`: A DEMIExperiment object.
- `pkg`: An environment. Specifies the environment where to load the data from.

Value

Returns a data.frame with annotation information.

Author(s)

Sten Ilmjarv

See Also

DEMIExperiment, environment
Description

The function `loadCel` loads the raw expression matrix from CEL files into a DEMIExperiment object. It is used internally in DEMI analysis.

Usage

```r
loadCel(object)
```

## S4 method for signature 'DEMIExperiment'
loadCel(object)

Arguments

- `object` A DEMIExperiment object.

Value

Returns a DEMIExperiment object updated with a DEMICel object attached to the slot `exprsData` that contains the raw expression matrix loaded from the CEL files.

Author(s)

Sten Ilmjarv

See Also

DEMIExperiment, DEMICel

Description

The function `loadCytoband` loads the karyotype information for the specified DEMIExperiment object. It is used internally in DEMI analysis.

Usage

```r
loadCytoband(object, pkg)
```

## S4 method for signature 'DEMIExperiment,environment'
loadCytoband(object, pkg)
loadDEMILibrary

Arguments

object A DEMIExperiment object.

pkg An environment. Specifies the environment where to load the data from.

Value

Returns a data.frame with karyotype information.

Author(s)

Sten Ilmjarv

See Also

DEMIExperiment, environment

Description

The function loadDEMILibrary loads the DEMI annotation packages specified by the organism parameter in the DEMIExperiment object. It is used internally in DEMI analysis.

Usage

loadDEMILibrary(object)

## S4 method for signature 'DEMIExperiment'
loadDEMILibrary(object)

Arguments

object A DEMIExperiment object.

Value

Returns a DEMIExperiment object updated with annotation and alignment information for the specified microarray platform and species. If the analysis parameter of the DEMIExperiment object is set to 'genome' it also attaches the cytoband information and if the analysis parameter of the DEMIExperiment object is set to 'gene' or 'transcript' it additionally loads the pathway information.

Author(s)

Sten Ilmjarv
The `loadPathway` function loads the pathway information specified by the `DEMIExperiment` object. It is used internally in DEMI analysis.

### Usage

```r
loadPathway(object, pkg)
```

---

#### Arguments

- **object**: A `DEMIExperiment` object.
- **pkg**: An environment. Specifies the environment where to load the data from.

#### Value

Returns a `data.frame` with pathway information.

#### Author(s)

Sten Ilmjarv

#### See Also

- `DEMIExperiment`
- `environment`
makeDEMIResultsTable  Returns a data.frame of the differential expression results

Description

Returns a data.frame of the differential expression results stored in the DEMIResult object. It is used internally by DEMI methods.

Usage

code{makeDEMIResultsTable(input = "list")}

Arguments

input  A list. Represents a list of DEMIResult object.

Value

Returns a data.frame of the differential expression analysis results stored in the DEMIResult objects.

Author(s)

Sten Ilmjarv

See Also

DEMIResult

makeUCSCLink  Make UCSC link

Description

The function makeUCSCLink makes a UCSC link of every genomic region in the specified data.frame. It is used internally in DEMI analysis.

Usage

code{makeUCSCLink(result)}

Arguments

result  A data.frame. A data.frame that consists of chromosome name and start and end coordinates to be used to make the UCSC link.
**matchExonGene**

**Value**

The input data.frame with added UCSC link as the last column.

**Author(s)**

Sten Ilmjarv

---

**matchExonGene**

*Matches exons to their corresponding transcripts.*

---

**Description**

The function `matchExonGene` matches exons to their corresponding transcripts. It is used internally in DEMI analysis.

**Usage**

`matchExonGene(cluster, blatTable, annoTable)`

**Arguments**

- `cluster` A vector. A vector of probe ID’s in the cluster.
- `blatTable` A data.frame. A data.frame with alignment information.
- `annoTable` A data.frame. A data.frame with annotation information.

**Value**

A data.frame where exons are matched to transcript.

**Author(s)**

Sten Ilmjarv
norm.quantile

Quantile normalization function

Description

A function for normalizing the expression matrix with quantiles. In the current state it tries to mimic rma quantile normalization. In the current state it is not used in DEMI analysis.

Usage

norm.quantile(object)

## S4 method for signature 'matrix'
norm.quantile(object)

Arguments

object A matrix. The raw expression matrix.

Value

A data.frame representing the normalized expression matrix.

Author(s)

Sten Ilmjarv

Examples

## Not run:

# Create a matrix with 1000 values that represents raw expression values
rawmatrix <- matrix(rexp(1000, rate=1), ncol=8)

# Normalize the raw expression matrix
normmatrix <- norm.quantile( rawmatrix )

## End(Not run)
norm.rank

Relative rank normalization function

Description

The function norm.rank normalizes the raw expression matrix by relative ranking. It is used internally in DEMI analysis.

Usage

```r
norm.rank(object)
```

## S4 method for signature 'matrix'
`norm.rank(object)`

## S4 method for signature 'numeric'
`norm.rank(object)`

Arguments

- `object` A matrix or numeric. The raw expression matrix or a single expression vector.

Value

A `data.frame` representing the normalized expression matrix.

Author(s)

Sten Ilmjarv

Examples

```r
## Not run:

# Create a matrix with 1000 values that represents raw expression values
rawmatrix <- matrix(rexp(1000, rate=1), ncol=8)

# Normalize the raw expression matrix
normmatrix <- norm.rank( rawmatrix )

## End(Not run)
```
probe.levels  

**Draws a histogram of the normalized expression levels of the specified targets**

**Description**

The function `probe.levels` draws a histogram of the normalized expression levels for the specified targets. Depending on the analysis the target can be ensembl gene ID or gene symbol (e.g. 'MAOB'), ensembl transcript ID, ensembl peptide ID or genomic region ID.

**Usage**

```r
probe.levels(object, target)
```

```
## S4 method for signature 'DEMIExperiment, character'
probe.levels(object, target)
```

**Arguments**

- `object`: A `DEMIExperiment` object.
- `target`: A vector. Depending on the analysis the target can be ensembl gene ID or gene symbol (e.g. 'MAOB'), ensembl transcript ID, ensembl peptide ID or genomic region ID.

**Value**

A ggplot object.

**Author(s)**

Sten Ilmjarv

**See Also**

DEMIExperiment

**Examples**

```r
## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files will be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/"
```
# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.

download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247696/suppl/GSM247696.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN01_GSM247696.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN02_GSM247697.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN03_GSM247700.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN04_GSM247701.CEL.gz", sep = "" ) )

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library( R.utils )
for( i in list.files( destfolder ) ) {
gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function probe.level

# Set up an experiment
demiexp <- DEMIExperiment( analysis = 'gene', celpath = destfolder,
experiment = 'myexperiment', organism = 'homo_sapiens' )

# Draw probes levels measuring the gene 'MAOB'
pdf( "MAOB_probe_levels.pdf", width=8, height=8 )
probe.levels( demiexp, "MAOB" )
dev.off()

## End(Not run)
Description
The function `probe.plot` draws a plot of the normalized expression levels for the specified targets. Depending on the analysis the target can be ensembl gene ID or gene symbol (e.g. 'MAOB'), ensembl transcript ID, ensembl peptide ID or genomic region ID.

Usage

```r
probe.plot(object, target)
```

## S4 method for signature 'DEMIExperiment,character'

```r
probe.plot(object, target)
```

Arguments

- `object`: A `DEMIExperiment` object.
- `target`: A vector. Depending on the analysis the target can be ensembl gene ID or gene symbol (e.g. 'MAOB'), ensembl transcript ID, ensembl peptide ID or genomic region ID.

Value

A `ggplot` object.

Author(s)

Sten Ilmjarv

See Also

`DEMIExperiment`

Examples

```r
## Not run:

# To use the example we need to download a subset of CEL files from
# by Pradervand et al. 2008.

# Set the destination folder where the downloaded files fill be located.
# It can be any folder of your choosing.
destfolder <- "demitest/testdata/"

# Download packed CEL files and change the names according to the feature
# they represent (for example to include UHR or BRAIN in them to denote the
# features).
# It is good practice to name the files according to their features which
# allows easier identification of the files later.
```
download.file( paste( ftpaddress, "GSM247694/suppl/GSM247694.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR01_GSM247694.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247695/suppl/GSM247695.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR02_GSM247695.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247698/suppl/GSM247698.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR03_GSM247698.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247699/suppl/GSM247699.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "UHR04_GSM247699.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247697/suppl/GSM247697.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN01_GSM247697.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247700/suppl/GSM247700.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN02_GSM247700.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247701/suppl/GSM247701.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN03_GSM247701.CEL.gz", sep = "" ) )
download.file( paste( ftpaddress, "GSM247704/suppl/GSM247704.CEL.gz", sep = "/" ),
destfile = paste( destfolder, "BRAIN04_GSM247704.CEL.gz", sep = "" ) )

# We need the gunzip function (located in the R.utils package) to unpack the gz files.
# Also we will remove the original unpacked files for we won't need them.
library( R.utils )
for( i in list.files( destfolder ) ) {
gunzip( paste( destfolder, i, sep = "" ), remove = TRUE )
}

# Now we can continue the example of the function probe.plot

# Set up an experiment
demiexp <- DEMIExperiment( analysis = 'gene', celpath = destfolder,
experiment = 'myexperiment', organism = 'homo_sapiens' )

# Draw probes levels measuring the gene 'MAOB'

pdf( "MAOB_probe_plot.pdf", width=8, height=8 )
probe.plot( demiexp, "MAOB" )
dev.off()

## End(Not run)

---

**totalMatches_all**

*Calculates the number of matches over all probes*

**Description**

The function `totalMatches_all` calculates the number of matches for all probes in DEMI analysis. It is used internally in DEMI analysis.

**Usage**

`totalMatches_all(blatTable)`
totalMatches_cluster

Arguments

- **blatTable**: A `data.frame`. A `data.frame` with alignment information.

Value

- A `data.frame` that represents the number of probes for each target.

Author(s)

- Sten Ilmjarv

---

**totalMatches_cluster**  
*Calculates the number of matches in the cluster*

Description

The function `totalMatches_cluster` calculates the number of matches in the cluster in DEMI analysis. It is used internally in DEMI analysis.

Usage

```
totalMatches_cluster(cluster, blatTable)
```

Arguments

- **cluster**: A vector. A vector of probe ID's in the cluster.
- **blatTable**: A `data.frame`. A `data.frame` with alignment information.

Value

- A `data.frame` that represents the number of probes for each target in the cluster.

Author(s)

- Sten Ilmjarv
validDEMIClust

Validates the DEMIClust object

Description
Validates the DEMIClust object.

Usage
validDEMIClust(object)

Arguments
object A DEMIClust object.

Value
Returns a validated DEMIClust object.

Author(s)
Sten Ilmjarv

validDEMIExperiment

Validates the DEMIExperiment object

Description
Validates the DEMIExperiment object

Usage
validDEMIExperiment(object)

Arguments
object A DEMIExperiment object.

Value
Returns a validated DEMIExperiment object.
Calculates wilcoxon's upper and lower probabilities

Description

Calculates the wilcoxon's upper and lower probabilities for each possible rank sum defined by the size of the test and reference samples.

Usage

wprob(m, n)

Arguments

m  
An integer. Defines the test sample size.

n  
An integer. Defines the reference sample size.

Value

A list. Returns a list of all possible lower and upper tail p-values defined by the sum of the possible rank combinations.

Author(s)

Sten Ilmjarv

Examples

# For test sample 4 and reference sample 6 calculate wilcoxon's upper and lower probabilities
wprob( 4, 6 )
Index

+Topic *data*

- DEMIMessages, 51

- addCustomTargets, 4
- addCytoband, 6
- adjust4maxprobes, 7
- attachResult, 7
- attachResult, DEMIExperiment, DEMIDiff-method (attachResult), 7

- calcHypergeoExon, 9
- calcHypergeoProb, 10
- celMatrixNormalize, 11
- celMatrixNormalize, DEMIExperiment, function-method (celMatrixNormalize), 11
- check4probe, 11
- check4probe, DEMIExperiment, vector-method (check4probe), 11
- check4target, 13
- check4target, DEMIExperiment, vector-method (check4target), 13
- checkDEMIExperiment_analysis, 15
- checkDEMIExperiment_celpath, 16
- checkDEMIExperiment_experiment, 16
- checkDEMIExperiment_maxprobes, 17
- checkDEMIExperiment_maxtargets, 17
- checkDEMIExperiment_normalization, 18
- checkDEMIExperiment_psize, 19
- cleanorganismname, 19
- cluster, 20
- cluster, DEMIClust-method (cluster), 20
- createGroup, 20
- createGroup, DEMIClust-method (createGroup), 20
- customObject, 21
- customObject, DEMIClust-method (customObject), 21

- demi, 23
- demi.comp.test, 27

- demi.t.test, 29
- demi.wilcox.test, 31
- demi.wilcox.test.fast, 33
- DEMICel, 35
- DEMICel-class, 36
- DEMIClust, 36
- DEMIClust-class, 39
- DEMIDiff, 40
- DEMIDiff-class, 42
- demiequal, 42
- DEMIExperiment, 44
- DEMIExperiment-class, 48
- DEMIGroup, 50
- DEMIGroup-class, 51
- DEMIMessages, 51
- DEMIPathway, 56
- DEMIResult-class, 58
- demisummary, 58
- demisummary, DEMIDiff-method (demisummary), 58
- demisummary, DEMIExperiment-method (demisummary), 58
- diffexp, 61
- diffexp, DEMIDiff-method (diffexp), 61
- diffSpliceScore, 62

- findCytoband, 6, 62

- getAlignment, 63
- getAlignment, DEMIExperiment-method (getAlignment), 63
- getAnalysis, 64
- getAnalysis, DEMIExperiment-method (getAnalysis), 64
- getAnnotation, 66
- getAnnotation, DEMIExperiment-method (getAnnotation), 66
- getArrayType, 68
- getArrayType, DEMIExperiment-method (getArrayType), 68
getCelMatrix, 69
getCelMatrix, DEMIExperiment-method (getCelMatrix), 69
getCelpath, 71
getCelpath, DEMIExperiment-method (getCelpath), 71
getCluster, 73
getCluster, DEMIClust-method (getCluster), 73
getClustMethod, 75
getClustMethod, DEMIClust-method (getClustMethod), 75
getCutoffPvalue, 76
getCutoffPvalue, DEMIClust-method (getCutoffPvalue), 76
getCytoband, 78
getCytoband, DEMIExperiment-method (getCytoband), 78
getDEMIClust, 80
getDEMIClust, DEMIDiff-method (getDEMIClust), 80
getExperiment, 82
getExperiment, DEMIClust-method (getExperiment), 82
getExperiment, DEMIDiff-method (getExperiment), 82
getExperiment, DEMIExperiment-method (getExperiment), 82
getGroup, 84
getGroup, DEMIClust-method (getGroup), 84
getGroup, DEMIDiff-method (getGroup), 84
getGroup, DEMIResult-method (getGroup), 84
getGroupA, 86
getGroupA, DEMIGroup-method (getGroupA), 86
getGroupB, 88
groupB, DEMIGroup-method (getGroupB), 88
groupB, DEMIExperiment-method (getGroupB), 88
groupNames, 90
groupNames, DEMIGroup-method (groupNames), 90
getIndexA, 91
getIndexA, DEMIGroup-method (getIndexA), 91
getIndexB, 93
getIndexB, DEMIGroup-method (getIndexB), 93
getIndexB, DEMIExperiment-method (getIndexB), 93
getMaxprobes, 95
getMaxprobes, DEMIExperiment-method (getMaxprobes), 95
getMaxtargets, 97
getMaxtargets, DEMIExperiment-method (getMaxtargets), 97
getName, 99
getName, DEMIDiff-method (getName), 99
gNormMatrix, 100
getNormMatrix, DEMIExperiment-method (getNormMatrix), 101
getOrganism, 102
getOrganism, DEMIExperiment-method (getOrganism), 102
gPathway, 104
gPathway, DEMIExperiment-method (gPathway), 104
gProbeLevel, 106
gProbeLevel, DEMIDiff, vector, logical-method (gProbeLevel), 106
gProbeLevel, DEMIExperiment, vector, logical-method (gProbeLevel), 106
gResult, 108
gResult, DEMIDiff-method (gResult), 108
gResult, DEMIExperiment-method (gResult), 108
gResult, DEMIResult-method (gResult), 108
gResultTable, 110
gResultTable, DEMIDiff-method (gResultTable), 110
gResultTable, DEMIExperiment-method (gResultTable), 110
gTargetAnnotation, 112
gTargetAnnotation, DEMIExperiment, vector-method (gTargetAnnotation), 112
gTargetProbes, 114
gTargetProbes, DEMIDiff, vector-method (gTargetProbes), 114
gTargetProbes, DEMIExperiment, vector-method (gTargetProbes), 114
initialize.DEMICel, 116
initialize.DEMIClust, 117
initialize.DEMIDiff, 117
initialize.DEMIExperiment, 118
initialize.DEMIGroup, 118
initialize.DEMIResult, 119
loadAnnotation, 119
loadAnnotation, DEMIExperiment, environment-method
(loadAnnotation), 119
loadBlat, 120
loadBlat, DEMIExperiment, environment-method
(loadBlat), 120
loadCel, 121
loadCel, DEMIExperiment-method
(loadCel), 121
loadCytoband, 121
loadCytoband, DEMIExperiment, environment-method
(loadCytoband), 121
loadDEMIlibrary, 122
loadDEMIlibrary, DEMIExperiment-method
(loadDEMIlibrary), 122
loadPathway, 123
loadPathway, DEMIExperiment, environment-method
(loadPathway), 123
makeDEMIResultsTable, 124
makeUCSCLink, 124
matchExonGene, 125
norm.quantile, 126
norm.quantile, matrix-method
(norm.quantile), 126
norm.rrank, 127
norm.rrank, matrix-method(norm.rrank), 127
norm.rrank, numeric-method(norm.rrank), 127
probe.levels, 128
probe.levels, DEMIExperiment, character-method
(probe.levels), 128
probe.plot, 129
probe.plot, DEMIExperiment, character-method
(probe.plot), 129
t.test, 30
totalMatches_all, 131
totalMatches_cluster, 132
validDEMIClust, 133
validDEMIExperiment, 133
wilcox.test, 32, 34, 43
wprob, 34, 134