

ArrayTools

April 20, 2011

adjustment	<i>Access the multiple comparison adjustment method from the regressResult or interactionResult class</i>
------------	---

Description

Access the multiple comparison adjustment method from the regressResult class or interactionResult class

Usage

```
adjustment (object)
```

Arguments

object a regressResult or interactionResult class

Value

a character vector

Author(s)

Xiwei Wu, Arthur Li

See Also

[regressResult](#) [interactionResult](#)

contrastMatrix *Class to Contain the Contrast Matrix that Used for Linear Regression*

Description

Class to Contain the Contrast Matrix that Used for Linear Regression, inherited from the designMatrix class

Creating Objects

```
new("contrastMatrix", ..., design.matrix=[designMatrix], compare1=[character],
    compare2=[character], level=[character], interaction==[numeric]).
```

This creates a contrast matrix class. `design.matrix` is a `designMatrix` class. `compare1` the first value of the main covariate, and `compare2` is the second value of the main covariate. For example, suppose that the main covariate is "drug", and there are three unique values: "drug1", "drug2", and "placebo". You would like to compare "drug1" to "drug2". Then you would use "drug1" as `compare1` and "drug2" as `compare2`. If `interaction==TRUE`, do not specify `compare1` and `compare2`. You only specify `level` when the design matrix contains an interaction term. Suppose that you would like to compare "drug1" to "drug2" only when estrogen is "present", where "present" is one of the values of the estrogen variable. You will use "present" as `level`. If `interaction==TRUE`, do not specify this value as well. You only specify `interaction=TRUE` when you would like to detect the interaction effect between two covariates. In this case, do not provide values for `compare1`, `compare2`, and `level`

Slots

contrast: Object of class "matrix" contains the contrast matrix
compare1: Object of class "character" contains `compare1`
compare2: Object of class "character" contains `compare2`
level: Object of class "character" contains `level`
interaction: Object of class "logical" contains `interaction`
design: Object of class "matrix" contain the design matrix
target: Object of class "data.frame" contains `target`
covariates: Object of class "character" contains `covariates`
intIndex: Object of class "numeric" contains `intIndex`

Extends

Class "`designMatrix`", directly.

Methods

getCompare1 signature(object = "contrastMatrix"): access the `compare1` slot
getCompare2 signature(object = "contrastMatrix"): access the `compare2` slot
getContrast signature(object = "contrastMatrix"): access the `contrast` slot
getInteraction signature(object = "contrastMatrix"): access the `interaction` slot

getLevel signature(object = "contrastMatrix"): access the level slot
initialize signature(.Object = "contrastMatrix"): create a new contrast matrix class
show signature(object = "contrastMatrix"): print the contrast matrix

Author(s)

Xiwei Wu, Arthur Li

See Also

[designMatrix](#)

Examples

```
data(eSetExample)
## One-way Anova
(design1<- new("designMatrix", target=pData(eSetExample), covariates = "Treatment"))
(contrast1<- new("contrastMatrix", design.matrix = design1,
  compare1 = "Treated", compare2 = "Control"))

## Randomized block design
(design2<- new("designMatrix", target=pData(eSetExample),
  covariates = c("Treatment", "Group")))
(contrast2<- new("contrastMatrix", design.matrix = design2,
  compare1 = "Treated", compare2 = "Control"))

## Interaction design
(design3<- new("designMatrix", target=pData(eSetExample),
  covariates = c("Treatment", "Group"), intIndex=c(1,2)))
# Test for interaction:
(contrast.int<- new("contrastMatrix", design.matrix = design3,
  interaction=TRUE))
# Compare Treated vs Control among group A
(contrast.a<- new("contrastMatrix", design.matrix = design3,
  compare1 = "Treated", compare2 = "Control", level="A"))
```

createExpressionSet

Creating an ExpressionSet

Description

Create an ExpressionSet based on phenotype data and expression data

Usage

```
createExpressionSet(pData, exprs, ...)
```

Arguments

pData	a data frame contains the phenotype data
exprs	a data frame contains the expression data
...	additional arguments passed to new("ExpressionSet", exprs, phenoData, ...) if needed

Value

an ExpressionSet

Author(s)

Xiwei Wu, Arthur Li

References

Bioconductor: Open software development for computational biology and bioinformatics R. Gentleman, V. J. Carey, D. M. Bates, B. Bolstad, M. Dettling, S. Dudoit, B. Ellis, L. Gautier, Y. Ge, and others 2004, Genome Biology, Vol. 5, R80

See Also

[ExpressionSet](#)

Examples

```
data(pDataExample)
data(exprsExample)
eSet <- createExpressionSet (pDataExample, exprsExample,
  annotation = "hugene10sttranscriptcluster")
```

createGSEAFiles *A Wrapper Function to create *.GCT and *.CLS for GSEA analysis*

Description

A Wrapper Function to create *.GCT and *.CLS for GSEA analysis

Usage

```
createGSEAFiles(mydir = getwd(), eSet, catVar)
```

Arguments

mydir	directory where you would like to store the files
eSet	an ExpressionSet
catVar	variable of interest

Value

Creating *.GCT and *.CLS for GSEA

Author(s)

Xiwei Wu, Arthur Li

References

<http://www.broad.mit.edu/gsea/>

See Also

[output.cls](#), [output.gct](#)

Examples

```
data(eSetExample)
## Not run: createGSEAFfiles (mydir, eSetExample, "Treatment")
```

createIndex	<i>Creating an HTML index file</i>
-------------	------------------------------------

Description

This HTML index file will link all the ouputted result, including Quality Assessment Report, differentially expressed genes, etc...

Usage

```
createIndex(..., mydir = getwd(), index.file = "index.html", createHeader = NULL)
```

Arguments

... regressionResults or interactionResult
 mydir the directory to contain the index file
 index.file name of the index file
 createHeader If want to want to create an Header, such as your name, company names, etc...

Value

creating an HTML index-file in your directory

Author(s)

Xiwei Wu, Arthur Li

Examples

```
data(eSetExample)
design<- new("designMatrix", target=pData(eSetExample), covariates = "Treatment")
contrast<- new("contrastMatrix", design.matrix = design,
  compare1 = "Treated", compare2 = "Control")
result<- regress(eSetExample, contrast)
sigResult<- selectSigGene(result, fc.value=log2(2))
## Not run: Output2HTML(sigResult)

design.int<- new("designMatrix", target=pData(eSetExample), covariates = c("Treatment", "
  intIndex = c(1, 2))
contrast.int<- new("contrastMatrix", design.matrix = design.int, interaction=TRUE)
result.int<- regress(eSetExample, contrast.int)
sigResult.int <- selectSigGene(result.int)
intResult <- postInteraction(eSetExample, sigResult.int, mainVar ="Treatment",
  compare1 = "Treated", compare2 = "Control")
```

```
sigResultInt <- selectSigGeneInt(intResult)
## Not run: Output2HTML(sigResultInt)

## Not run: createIndex(sigResult, sigResultInt, createHeader = c("Arthur Li", "COH"))
```

```
createIngenuityFile
```

A Wrapper Function to Create Files for Ingenuity Analysis

Description

A Wrapper Function to Create Files for Ingenuity Analysis

Usage

```
createIngenuityFile(..., mydir = getwd(), eSet, filename = "IngenuityFile")
```

Arguments

...	a list of regressResult class
mydir	the directory where you would like to store the file
eSet	an ExpressionSet
filename	file name

Details

This function enable to create the ingenuity upload file based on a list of regressResult

Value

create an Ingenuity upload file

Author(s)

Xiwei Wu, Arthur Li

References

<http://www.ingenuity.com/>

Examples

```
data(eSetExample)
design<- new("designMatrix", target=pData(eSetExample), covariates = "Treatment")
contrast<- new("contrastMatrix", design.matrix = design,
  compare1 = "Treated", compare2 = "Control")
result<- regress(eSetExample, contrast)
## Not run: createIngenuityFile(result, eSet = eSetExample)
```

`designMatrix`*Class to Contain the Design Matrix that Used for Linear Regression*

Description

Class to Contain the Design Matrix that Used for Linear Regression

Creating Objects

```
new("designMatrix", ..., target, covariates, intIndex=0)
```

This create as design matrix class. `target` is a data frame that contains chip and covariate information, or experimental phenotypes recorded in `eSet` and `ExpressionSet`-derived classes. `covariates` is a list of 1-n covariates. If `intIndex=0`, the interaction effect is not considered; otherwise, use two integers to indicate which covariates are considered for interaction effect. For example, if `covariates <- c("estrogen", "drug", "time")` and you are considering the interaction between "estrogen" and "time", then you would write `intIndex=c(1, 3)`

Slots

`design`: contains the design matrix

`target`: contains the target data

`covariates`: contains the covariates

`intIndex`: contains the `intIndex`

Methods

getCovariates signature(object = "designMatrix"): access the covariates slot

getDesign signature(object = "designMatrix"): access the design slot

getIntIndex signature(object = "designMatrix"): access the intIndex slot

getTarget signature(object = "designMatrix"): access the target slot

initialize signature(.Object = "designMatrix"): create a new designMatrix class

show signature(object = "designMatrix"): print the designMatrix class

Author(s)

Xiwei Wu, Arthur Li

See Also

[contrastMatrix](#)

Examples

```
data(eSetExample)
## One-way Anova
(design1<- new("designMatrix", target=pData(eSetExample), covariates = "Treatment"))

## Randomized block design
(design2<- new("designMatrix", target=pData(eSetExample),
  covariates = c("Treatment", "Group")))
```

```
## Interaction design
(design3<- new("designMatrix", target=pData(eSetExample),
  covariates = c("Treatment", "Group"), intIndex=c(1,2)))
```

eSetExample *An ExpressionSet example*

Description

An ExpressionSet example

Usage

```
data(eSetExample)
```

Format

The format is: Formal class 'ExpressionSet' [package "Biobase"] with 6 slots

Examples

```
data(eSetExample)
```

exprsExample *a data.frame contains expression data*

Description

a data.frame contains expression data

Usage

```
data(exprsExample)
```

Format

A data frame with 1000 observations on the following 17 variables.

```
probeset_id a numeric vector
H1.CEL a numeric vector
H2.CEL a numeric vector
H3.CEL a numeric vector
H4.CEL a numeric vector
H5.CEL a numeric vector
H6.CEL a numeric vector
H7.CEL a numeric vector
```


H8.CEL a numeric vector
 H9.CEL a numeric vector
 H10.CEL a numeric vector
 H11.CEL a numeric vector
 H12.CEL a numeric vector
 H13.CEL a numeric vector
 H14.CEL a numeric vector
 H15.CEL a numeric vector
 H16.CEL a numeric vector

Examples

```
data(exprsExample)
```

geneFilter	<i>filter an ExpressionSet using different methods</i>
------------	--

Description

Create a filtered 'ExpressionSet' based on background, range, or interquartile range

Usage

```
geneFilter(object, pct = 0.1, numChip = ceiling(ncol(exprs(object)) * pct), bg =
```

Arguments

object	an ExpressionSet
pct	percentage
numChip	number of chips. If you would like to filter the ExpressionSet based on at least 3 chips greater than 1 (bg=1), then set numChip = 3
bg	background value. If you would like to filter the ExpressionSet based on at least 3 chips greater than 1, then set bg=1
range	range = max value - min value of each gene
iqrPct	interquartile percentage
output	if output = TRUE, output filtered data in the sepecified directory
mydir	the directory containing the filtered data

Details

There are three filtering methods. The User can use either one, two, or three. 1). At least a certain number of chips (numChip) are greater than a given background (bg) 2). The range of the gene have to be greater than a given value (range) 3). Calculating the interquartile range (IQR) of each gene to create an IQR vector. Based on the given percentage (e.g. iqrPct=0.2), find the corresponding percentile. If IQR is less than percentile, the gene will be filtered

Value

a filtered `ExpressionSet`

Author(s)

Xiwei Wu, Arthur Li

Examples

```
data(eSetExample)
filtered <- geneFilter(eSetExample)
```

getAdjP

access the adjPVal slot from regressResult or interactionResult class

Description

access the adjPVal slot from regressResult or interactionResult class

Usage

```
getAdjP(object)
```

Arguments

object a regressResult class or interactionResult class

Value

a numeric vector

Author(s)

Xiwei Wu, Arthur Li

See Also

[regressResult](#) [interactionResult](#)

getAnnotation	<i>access the annotation slot from the regressResult or interactionResult slot</i>
---------------	--

Description

access the annotation slot from the regressResult or interactionResult slot

Usage

```
getAnnotation(object)
```

Arguments

object a regressResult class or interactionResult class

Value

a character vector

Author(s)

Xiwei Wu, Arthur Li

See Also

[regressResult](#) [interactionResult](#)

getCompare1	<i>Access the Compare1 slot from the contrastMatrix</i>
-------------	---

Description

Access the Compare1 slot from the contrastMatrix

Usage

```
getCompare1(object)
```

Arguments

object a contrastMatrix class

Value

a character vector

Author(s)

Xiwei Wu, Arthur Li

See Also[contrastMatrix](#)

getCompare2	<i>Access the compare2 slot from the contrastMatrix class</i>
-------------	---

Description

Access the compare2 slot from the contrastMatrix class

Usage

```
getCompare2(object)
```

Arguments

object a contrastMatrix class

Value

a character vector

Author(s)

Xiwei Wu, Arthur Li

See Also[contrastMatrix](#)

getContrast	<i>Access the contrast matrix from the contrastMatrix class</i>
-------------	---

Description

Access the contrast matrix from the contrastMatrix class

Usage

```
getContrast(object)
```

Arguments

object a contrastMatrix class

Value

a numeric matrix

Author(s)

Xiwei Wu, Arthur Li

See Also

[contrastMatrix](#)

getCovariates *Accessing the covariates from the designMatrix class*

Description

Accessing the covariates from the designMatrix class

Usage

```
getCovariates(object)
```

Arguments

object a designMatrix class

Value

a character vector containing covariates

Author(s)

Xiwei Wu, Arthur Li

See Also

[designMatrix](#)

getDesign *Access the design matrix from the designMatrix class*

Description

Access the design matrix from the designMatrix class

Usage

```
getDesign(object)
```

Arguments

object a designMatrix class

Value

a matrix containing the `designMatrix`

Author(s)

Xiwei Wu, Arthur Li

See Also

[designMatrix](#)

<code>getFCCutoff</code>	<i>Access the significantFCCutoff slot from the regressResult or interactionResult class</i>
--------------------------	--

Description

Access the significantFCCutoff slot from the regressResult or interactionResult class

Usage

```
getFCCutoff(object)
```

Arguments

`object` a regressResult or interactionResult class

Value

a numeric vector

Author(s)

Xiwei Wu, Arthur Li

See Also

[regressResult](#) [interactionResult](#)

getFC	<i>Access the foldChange slot from the regressResult or interactionResult class</i>
-------	---

Description

Access the foldChange slot from the regressResult or interactionResult class

Usage

```
getFC(object)
```

Arguments

object a regressResult class or interactionResult class

Value

a numeric vector

Author(s)

Xiwei Wu, Arthur Li

See Also

[regressResult](#) [interactionResult](#)

getFilterMethod	<i>Access the filterMethod slot from the regressResult or interactionResult class</i>
-----------------	---

Description

Access the filterMethod slot from the regressResult or interactionResult class

Usage

```
getFilterMethod(object)
```

Arguments

object a regressResult or interactionResult class

Value

a list

Author(s)

Xiwei Wu, Arthur Li

References

~put references to the literature/web site here ~

See Also

[regressResult](#) [interactionResult](#)

getF	<i>access the foldChange slot from regressionResult or interactionResult class</i>
------	--

Description

access the foldChange slot from regressionResult or interactionResult class

Usage

```
getF(object)
```

Arguments

object a regressResult or interactionResult class

Value

a numeric vector

Author(s)

Xiwei Wu, Arthur Li

See Also

[regressResult](#) [interactionResult](#)

getID	<i>access the ID slot from the regressResult or interactionResult class</i>
-------	---

Description

access the ID slot from the regressResult or interactionResult class

Usage

```
getID(object)
```

Arguments

object a regressResult class or interactionResult class

Value

a character vector

Author(s)

Xiwei Wu, Arthur Li

See Also

[regressResult](#) [interactionResult](#)

`getIndex`

Access the SignificantIndex slot from the regressResult or interactionResult class

Description

Access the SignificantIndex slot from the regressResult or interactionResult class

Usage

`getIndex(object)`

Arguments

`object` a regressResult or interactionResult class

Value

a logical vector

Author(s)

Xiwei Wu, Arthur Li

See Also

[regressResult](#) [interactionResult](#)

`getInteraction` *Access the interaction slot from the contrastMatrix class*

Description

Access the interaction slot from the contrastMatrix class

Usage

```
getInteraction(object)
```

Arguments

`object` a contrastMatrix class

Value

a logical vector

Author(s)

Xiwei Wu, Arthur Li

See Also

[contrastMatrix](#)

`getIntIndex` *Access the IntIndex slot from the designMatrix class*

Description

Access the IntIndex slot from the designMatrix class

Usage

```
getIntIndex(object)
```

Arguments

`object` an designMatrix class

Value

a numeric vector

Author(s)

Xiwei Wu, Arthur Li

See Also

[designMatrix](#)

`getLength` *Calculate the Length of interactionResult class*

Description

Calculate the Length of interactionResult class

Usage

```
getLength(object)
```

Arguments

`object` an interactionResult class

Value

a numeric value

Author(s)

Xiwei Wu, Arthur Li

See Also

[interactionResult](#)

`getLevel` *Access the level slot from the contrastMatrix class*

Description

Access the level slot from the contrastMatrix class

Usage

```
getLevel(object)
```

Arguments

`object` a contrastMatrix class

Value

a character vector

Author(s)

Xiwei Wu, Arthur Li

See Also

[contrastMatrix](#)

getNormalizationMethod

Access the significantIndex slot from the regressResult or interactionResult class

Description

Access the significantIndex slot from the regressResult or interactionResult class

Usage

```
getNormalizationMethod(object)
```

Arguments

object a regressResult or interactionResult class

Value

a character vector

Author(s)

Xiwei Wu, Arthur Li

See Also

[regressResult](#) [interactionResult](#)

getPCutoff

Access the significantPvalueCutoff slot from regressResult or interactionResult class

Description

Access the significantPvalueCutoff slot from regressResult or interactionResult class

Usage

```
getPCutoff(object)
```

Arguments

object a regressResult or interactionResult class

Value

a numeric vector

Author(s)

Xiwei Wu, Arthur Li

See Also

[regressResult](#) [interactionResult](#)

getP

Access the pValue slot from regressResult or interactionResult class

Description

Access the pValue slot from regressResult or interactionResult class

Usage

getP (object)

Arguments

object a regressResult or interactionResult class

Value

a character vector

Author(s)

Xiwei Wu, Arthur Li

See Also

[regressResult](#) [interactionResult](#)

getTarget

Access the target slots from the designMatrix class

Description

Access the target slots from the designMatrix class

Usage

getTarget (object)

Arguments

object a designMatrix class

Value

a data frame contains the target file

Author(s)

Xiwei Wu, Arthur Li

See Also

[designMatrix](#)

hugene10stCONTROL *hugene10stCONTROL*

Description

It is used to remove "normgene" and "control" genes for hugene10st array in the `preProcessGeneST` function. It is not intended to be used by the user.

Usage

```
data(hugene10stCONTROL)
```

Format

A data frame with 4201 observations on the following 2 variables.

interactionResult-class
Class to Contain the Regression Result Based on An Interaction Model

Description

Class to Contain the Regression Result Based on An Interaction Model. Interaction is a statistical term referring to a situation when the relationship between the outcome and the variable of the main interest differs at different levels of the extraneous variable

Creating Objects

`interactionResult` object is generally created from the `postInteraction` function See [postInteraction](#)

Object Components

A list of four or more components. Each component is a `reggressResult` class. The first component contains all the genes. The second component contains genes with the interaction effect The rest components contains genes with the interaction effect across different levels. Each component contains the result for each level.

Extends

Class "[list](#)", from data part. Class "[vector](#)", by class "[list](#)", distance 2.

Methods

adjustment signature(object = "regressResult"): access the adjustment slot

getAdjP signature(object = "regressResult"): access the adjPVal slot

getAnnotation signature(object = "regressResult"): access the annotation slot

getContrast signature(object = "regressResult"): access the contrast slot

getF signature(object = "regressResult"): access the FValue slot

getFC signature(object = "regressResult"): access the foldChange slot

getFCCutoff signature(object = "regressResult"): access the significantFCCutoff slot

getFileName signature(object = "regressResult"): access the fileName slot

getFilterMethod signature(object = "regressResult"): access the filterMethod slot

getID signature(object = "regressResult"): access the ID slot

getIndex signature(object = "regressResult"): access the significantIndex slot

getNormalizationMethod signature(object = "regressResult"): access the normalizationMethod slot

getP signature(object = "regressResult"): access the pValue slot

getPCutoff signature(object = "regressResult"): access the significantPvalueCutoff slot

Output2HTML signature(object = "regressResult"): create HTML file for significant genes in regressionResult

regressionMethod signature(object = "regressResult"): access the regressionMethod slot

selectSigGene signature(object = "regressResult"): select significant genes for regressionResult class

show signature(object = "regressResult"): print regressResult

Sort signature(x = "regressResult"): sort regressResult

summary signature(object = "regressResult"): print the summary for regressResult

getLength signature(object = "interactionResult"): calculate the length of the interactionResult class

Author(s)

Xiwei Wu, Arthur Li

See Also

[regressResult](#)

Examples

```
## Creating the interactionResult takes a few steps:
data(eSetExample)
design.int<- new("designMatrix", target=pData(eSetExample), covariates = c("Treatment", "
  intIndex = c(1, 2))
contrast.int<- new("contrastMatrix", design.matrix = design.int, interaction=TRUE)
result.int<- regress(eSetExample, contrast.int)
sigResult.int <- selectSigGene(result.int)
intResult <- postInteraction(eSetExample, sigResult.int, mainVar ="Treatment",
  compare1 = "Treated", compare2 = "Control")
```

mogene10stCONTROL *mogene10stCONTROL*

Description

It is used to remove "normgene" and "control" genes for mogene10st array in the `preProcessGeneST` function. It is not intended to be used by the user.

Usage

```
data(mogene10stCONTROL)
```

Format

A data frame with 6613 observations on the following 2 variables.

Output2HTML *Creating HTML file for regressResult or interactionResult class*

Description

Creating HTML file for regressResult or interactionResult class

Usage

```
Output2HTML(object, ...)
```

Arguments

<code>object</code>	an <code>regressResult</code> or <code>interactionResult</code> class
<code>...</code>	you can specify the directory to store the result by using the <code>mydir</code> argument. The default value of <code>mydir</code> is the current working directory

Value

creating an HTML file

Author(s)

Xiwei Wu, Arthur Li

Examples

```
data(eSetExample)
design<- new("designMatrix", target=pData(eSetExample), covariates = "Treatment")
contrast<- new("contrastMatrix", design.matrix = design,
  compare1 = "Treated", compare2 = "Control")
result<- regress(eSetExample, contrast)
sigResult<- selectSigGene(result, fc.value=log2(2))
## Not run: Output2HTML(sigResult)
```

output.cls	<i>Create *.CLS file for GSEA analysis</i>
------------	--

Description

Create *.CLS file for GSEA analysis

Usage

```
output.cls(target, variable, filename = "phenotype")
```

Arguments

target	pheno Data file
variable	variable of interest
filename	file name

Value

create a *.CLS file

Author(s)

Xiwei, Wu, Arthur Li

References

<http://www.broad.mit.edu/gsea/>

See Also

[output.gct](#), [createGSEAFiles](#)

`output.gct`*Create an *.GCT file for GSEA analysis*

Description

Create an *.GCT file for GSEA analysis

Usage

```
output.gct(normal, filename = "probe")
```

Arguments

<code>normal</code>	an ExpressionSet
<code>filename</code>	file name

Value

create an *.GCT file

Author(s)

Xiwei Wu, Arthur Li

References

<http://www.broad.mit.edu/gsea/>

See Also

[output.cls](#), [createGSEAFiles](#)

`output.ing`*Create an Ingenuity File for Ingenuity Analysis*

Description

Create an Ingenuity File for Ingenuity Analysis

Usage

```
output.ing(allfile, eSet, filename = "IngenuityFile")
```

Arguments

<code>allfile</code>	a list of regressResult class
<code>eSet</code>	an ExpressionSet
<code>filename</code>	file name

Value

create an txt file for Ingenuity Analysis

Author(s)

Xiwei Wu, Arthur Li

References

<http://www.ingenuity.com/>

See Also

[createIngenuityFile](#)

`pDataExample`

a phenoData example

Description

a data frame contains the phenotype data

Usage

```
data(pDataExample)
```

Format

A data frame with 16 observations on the following 2 variables.

Treatment a character vector

Group a character vector

Examples

```
data(pDataExample)
```

postInteraction *Create an Object of InteractionResult Class for Testing Interaction*

Description

Based on the result from the interaction test by looking at the result from the regressResult object, this function partitions the original data, an ExpressionSet into groups, one contains the genes without the interaction and others contains the genes with the interaction across different level of covariates.

Usage

```
postInteraction(eSet, regressObject, mainVar, compare1, compare2, method = regre
```

Arguments

eSet	an ExpressionSet
regressObject	a regressResult
mainVar	variable of main interest
compare1	the first value of the mainVar. For example, suppose that mainVar is "drug", and there are three unique values: "drug1", "drug2", and "placebo". You would like to compare "drug1" to "drug2". Then you would use "drug1" as compare1
compare2	Based on the example for compare1, "drug2" will be the compare2
method	It is used to run regression within each level of the effect modifier. choose the following three options: "limma" (LIMMA), "regression" (ordinary linear regression), "permutation" (permutation test)
adj	adjustment method for multiple comparison test, including "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none". The default value is "none". Type help(p.adjust) for more detail.

Value

an interactionResult class. The first component contains all the result for all the genes. The second component contains the genes without the interaction effect. The rest of the components contains genes with the interactions.

Author(s)

Xiwei Wu, Arthur Li

Examples

```
data(eSetExample)
design.int<- new("designMatrix", target=pData(eSetExample), covariates = c("Treatment", "
  intIndex = c(1, 2))
contrast.int<- new("contrastMatrix", design.matrix = design.int, interaction=TRUE)
result.int<- regress(eSetExample, contrast.int)
sigResult.int <- selectSigGene(result.int)
intResult <- postInteraction(eSetExample, sigResult.int, mainVar ="Treatment",
  compare1 = "Treated", compare2 = "Control")
```

```
preProcess3prime
```

A wrapper function to normalize the the 3 prime array

Description

A wrapper function to normalize the 3 prime array by using either RMA or GCRMA method

Usage

```
preProcess3prime(object, method = c("rma", "gcrma"), output = FALSE, mydir = get
```

Arguments

<code>object</code>	an AffyBatch.
<code>method</code>	either rma or gcrma
<code>output</code>	If <code>output = TRUE</code> , it will output the preprocessed data in the specified directory from the <code>mydir</code> argument
<code>mydir</code>	specified directory to contain the output

Value

an ExpressionSet

Author(s)

Xiwei Wu, Arthur Li

See Also

~~objects to See Also as [help](#), ~~~

Examples

```
if (require(affydata)) {
  data(Dilution)
  eset <- preProcess3prime(Dilution)
}
```

```
preProcessGeneST
```

Proprocess genechip ST array

Description

Proprocess genechip ST array by taking the log2 of the expression value.

Usage

```
preProcessGeneST(object, offset = 1, rmControl = TRUE, output = FALSE, mydir = g
```

Arguments

object	an ExpressionSet.
offset	The offset is added to the expression value to avoid $\log_2(0) = -\text{Inf}$.
rmControl	Setting <code>rmControl = TRUE</code> to remove control probes.
output	If <code>output = TRUE</code> , it will output the preprocessed data in the specified directory from the <code>mydir</code> argument.
mydir	specified directory to contain the output

Value

an ExpressionSet

Author(s)

Xiwei Wu, Arthur Li

Examples

```
data(eSetExample)
processedData <- preProcessGeneST(eSetExample)
```

qa3prime

Creating Quality Assessment Report for 3 Prime Array

Description

Creating Quality Assessment Report for 3 Prime Array in HTML file

Usage

```
qa3prime(object, parameters, outputFile = "QA.html", mydir = getwd())
```

Arguments

object	an AffyBatch object
parameters	The names of the variables to be included in the report
outputFile	The name of the outputfile. Make sure write ".html"
mydir	The name of the directory containing the report

Details

This function creates quality control report in an HTML file that contains a set of 9 assessment figures.

Figure1: The Raw Intensity Plot. The raw intensity should be similar across all chips

Figure2: The Average Background/Percentage Present Plot. The Average Background should be similar across all chips. The Percentage Present should be similar across all chips, except that in rare situations transcription is globally shut down or turned on under some conditions

Figure3: The Scaling Factor Plot. The scaling factor should be within 3-fold across all chips

Figure4: The Hybridization Controls Plot. BioB, BioC, BioD, CreX should be called present, except that it is acceptable if BioB is absent sometimes.

Figure5: The Housekeeping Controls Plot. The GAPDH ratio should be around 1 and the actin ratio should be less than 3. Note that if two-cycle amplification or NuGen amplification is used, this ratio could be much higher.

Figure6: The RNA Degradation Plot. On Affymetrix GeneChips, individual probes in a probeset are ordered by location relative to the 5' end of the targeted RNA molecule. On each chip, probe intensities are averaged by location in the probeset, with the average taken over probesets. In an RNA digestion plot, these means are plotted side-by-side, making it easy to notice any 5' to 3' trend. The trend can be due to RNA degradation or 3'-biased amplification. Since RNA degradation typically starts from the 5' end of the molecule and amplification starts at the 3' end, we would expect probe intensities to be systematically lowered at the 5' end of a probeset when compared to the 3' end.

Figure7: The Hierarchical Clustering of Samples. Samples will be grouped using hierarchical clustering and principal component analysis (PCA). If the sample preparation steps introduced bigger variation than biological variation, treatment groups will be mixed up in the plot. This could also happen when the samples between groups were mixed up accidentally when the samples were prepared. We acknowledge that clinical samples are harder to collect and sometimes impossible to control. Therefore, sample QC criteria will be much looser when dealing with clinical samples.

Figure8: The Pseudo-chip Images. A Pseudo-chip image plots the weights and residuals from the model fit. The image plot allows detection of artifacts on the chip.

Figure9: The Normalized Unscaled Standard Error (NUSE) and Relative Log Expression (RLE) Plots. The NUSE is fitted robustly by iteratively reweighted least squares (IRLS) so that the standard error of the estimated log₂ scale expression can be estimated. The boxplots of the NUSE show the differences in hybridization quality most clearly, in magnitude as well as variability. A high NUSE likely corresponds to a low signal. The RLE plot is a boxplot showing the distribution of Log₂ ratio of each chip relative to a median chip. A discordant distribution infers a problem with the chip.

Value

no value is returned

Author(s)

Xiwei Wu, Arthur Li

References

<http://www.affymetrix.com>

Examples

```
## Not run: qa3prime(AffyBatchExample, c("var1", "var2"))
```

Description

Creating Quality Assessment Report for Gene ST Array in HTML file

Usage

```
qaGeneST(object, parameters, QC, mydir = getwd(), outputFile = "QA.html")
```

Arguments

object	an ExpressionSet
parameters	The names of the variables to be included in the report
QC	The QC report generated from Affymetrix Expression Console
mydir	The name of the directory containing the report
outputFile	The name of the outputfile. Make sure write ".html"

Details

This function creates quality control report in an HTML file that contains a set of 8 assessment figures.

Figure1: The intensity distributio Plot. The raw intensity should be similar across all chips

Figure2: The Mean Signal Plot. The mean signal of each group should be consistant across the samples. The positive control should be higher than the negative controls.

Figure3: BAC SPIKE plot. The mean signal of each group should be consistant across the samples. The signal for BioB should be the lowest, follows by BioC, BioD, and CreX (the highest).

Figure4: POLYA SPIKE plot. The mean signal of each group should be consistant across the samples. The signal for Lys should be the lowest, follows by Thr, Phe, and Dap.

Figure5: POS VS NEG AUC plot. Pos vs neg auc is the area under the curve (AUC) for a receiver operating characteristic (ROC) plot comparing signal values for the positive controls to the negative controls. In practice the expected value for this metric is tissue type specific and may be sensitive to the quality of the RNA sample. Values between 0.80 and 0.90 are typical.

Figure6: MAD RESIDUAL MEAN plot. A measure of how well or poor all of the probes on a given chip fit the RMA or PLIER model. An unusually high mean absolute deviation of the residuals from the median suggests problematic data for that chip.

Figure7: RLE MEAN plot. This metric is generated by taking the signal estimate for a given probeset on a given chip and calculating the difference in log base 2 from the median signal value of that probeset over all the chips. When just the replicates are analyzed together the mean absolute RLE should be consistently low, reflecting the low biological variability of the replicates.

Figure8: Hierarchical Clustering of Samples . Samples will be grouped using hierarchical clustering and principal component analysis (PCA). If the sample preparation steps introduced bigger variation than biological variation, treatment groups will be mixed up in the plot. This could also happen when the samples between groups were mixed up accidentally when the samples were prepared. We acknowledge that clinical samples are harder to collect and sometimes impossible to control. Therefore, sample QC criteria will be much looser when dealing with clinical samples.

Value

no value is returned

Author(s)

Xiwei Wu, Arthur Li

References

http://www.affymetrix.com/support/technical/whitepapers/exon_gene_arrays_qa_whitepaper.pdf

Examples

```
data(eSetExample)
logdata <- preprocessGeneST(eSetExample)
data(QC)
## Not run: qaGeneST(logdata, c("Treatment", "Group"), QC)
```

QC

sample QC result from Affy Expression Console

Description

quality assessment result sample data generated from Affy Expression Console

Usage

```
data(QC)
```

Examples

```
data(QC)
```

regressionMethod

Access the regressionMethod slot from the regressResult or interactionResult class

Description

Access the regressionMethod slot from the regressResult or interactionResult class

Usage

```
regressionMethod(object)
```

Arguments

object a regressResult or interactionResult class

Value

a character vector

Author(s)

Xiwei Wu, Arthur Li

See Also

[regressResult](#) [interactionResult](#)

regress

Run regression to fit genewise linear model

Description

Fit genewise linear model using LIMMA package, ordinary linear regression, or permutation method.

Usage

```
regress(object, contrast, method = c("limma", "regression", "permutation"), adj
```

Arguments

<code>object</code>	an ExpressionSet
<code>contrast</code>	a contrastMatrix
<code>method</code>	choose the following three options: "limma" (LIMMA), "regression" (ordinary linear regression), "permutation" (permutation test)
<code>adj</code>	adjustment method for multiple comparison test, including "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none". The default value is "none". Type <code>help(p.adjust)</code> for more detail.
<code>permute.time</code>	number of permutation times, only used for the "permutation" method

Value

an object of `regressResult`

Author(s)

Xiwei Wu, Arthur Li

Examples

```
data(eSetExample)
design<- new("designMatrix", target=pData(eSetExample), covariates = "Treatment")
contrast<- new("contrastMatrix", design.matrix = design,
  compare1 = "Treated", compare2 = "Control")
result<- regress(eSetExample, contrast)
```

 regressResult-class

Class to Contain the Regression Result

Description

Class to Contain the Regression Result

Creating Objects

regressResult object is generally created from the regress function See [regress](#)

Slots

ID: contains probe ID/gene ID

foldChange: contains fold change value

FValue: contains F statistics

pValue: contains p value

adjPVal: contains adjusted p value

contrast: contains class "contrastMatrix"

regressionMethod: contains regression method: "limma", "regression", or "permutation"

adjustment: contains method for multiple comparison adjustment

significantIndex: contains a logical index indicating significant genes

significantPvalueCutoff: contains a cutoff p-value for choosing significant genes

significantFCCutoff: contains a fold change cutoff value for choosing significant genes

fileName: contains a file name for output purpose

annotation: contains annotation

normalizationMethod: contains normalization method - for output purpose

filterMethod: contains filtered method - for output purpose

Methods

adjustment signature(object = "regressResult"): access the adjustment slot

getAdjP signature(object = "regressResult"): access the adjPVal slot

getAnnotation signature(object = "regressResult"): access the annotation slot

getContrast signature(object = "regressResult"): access the contrast slot

getF signature(object = "regressResult"): access the FValue slot

getFC signature(object = "regressResult"): access the foldChange slot

getFCCutoff signature(object = "regressResult"): access the significantFCCutoff slot

getFileName signature(object = "regressResult"): access the fileName slot

getFilterMethod signature(object = "regressResult"): access the filterMethod slot

getID signature(object = "regressResult"): access the ID slot

getIndex signature(object = "regressResult"): access the significantIndex slot

getNormalizationMethod signature(object = "regressResult"): access the normalizationMethod slot

getP signature(object = "regressResult"): access the pValue slot

getPCutoff signature(object = "regressResult"): access the significantPvalueCutoff slot

Output2HTML signature(object = "regressResult"): create HTML file for significant genes in regressionResult

regressionMethod signature(object = "regressResult"): access the regressionMethod slot

selectSigGene signature(object = "regressResult"): select significant genes for regressionResult class

show signature(object = "regressResult"): print regressResult

Sort signature(x = "regressResult"): sort regressResult

summary signature(object = "regressResult"): print the summary for regressResult

Author(s)

Xiwei Wu, Arthur Li

Examples

```
data(eSetExample)
design<- new("designMatrix", target=pData(eSetExample), covariates = "Treatment")
contrast<- new("contrastMatrix", design.matrix = design,
  compare1 = "Treated", compare2 = "Control")
result<- regress(eSetExample, contrast)
```

selectSigGeneInt *select differentially expressed genes from the interactionResult class*

Description

select differentially expressed genes based on p value and/or fold change from the interactionResult class

Usage

```
selectSigGeneInt(object, pGroup = 0.05, fcGroup = 0, pMain = 0.05, fcMain = 0)
```

Arguments

object	an interactionResult class
pGroup	the p value that used to select significant genes at each level of the covariate
fcGroup	the fold change value that used to select significant genes at each level of the covariate
pMain	the p values that used to select significant genes among genes without any interaction effect
fcMain	the fold change values that used to select significant genes among genes without any interaction effect

Value

an interactionResult

Author(s)

Xiwei Wu, Arthur Li

Examples

```
data(eSetExample)
design.int<- new("designMatrix", target=pData(eSetExample), covariates = c("Treatment", "
  intIndex = c(1, 2))
contrast.int<- new("contrastMatrix", design.matrix = design.int, interaction=TRUE)
result.int<- regress(eSetExample, contrast.int)
sigResult.int <- selectSigGene(result.int)
intResult <- postInteraction(eSetExample, sigResult.int, mainVar ="Treatment",
  compare1 = "Treated", compare2 = "Control")
sigResultInt <- selectSigGeneInt(intResult)
```

selectSigGene

select differentially expressed genes from the regressResult class

Description

select differentially expressed genes based on p value and/or fold change from the regressResult class

Usage

```
selectSigGene(object, p.value = 0.05, fc.value = 0)
```

Arguments

object	an regressResult class
p.value	p value
fc.value	fold change cut-off value

Value

an regressResult

Author(s)

Xiwei Wu, Arthur Li

Examples

```
data(eSetExample)
design<- new("designMatrix", target=pData(eSetExample), covariates = "Treatment")
contrast<- new("contrastMatrix", design.matrix = design,
  compare1 = "Treated", compare2 = "Control")
result<- regress(eSetExample, contrast)
sigResult<- selectSigGene(result, fc.value=log2(2))
```

Sort

Sort a regressionResult or an interactionResult

Description

Sort a regressionResult or an interactionResult based on p-value, fold-change, or F statistics

Usage

```
Sort(x, ...)
```

Arguments

`x` a regressionResult or an interactionResult class
`...` any other arguments. See below...

Value

if sorting a regressionResult, returned value is a data frame if sorting a interactionResult, returned value is a list of data frames

Sort a regressionResult or an interactionResult class

```
Sort(x, sorted.by = c("pValue", "log2Ratio", "F"), top=20)
```

`x` is a regressionResult class or an interactionResult class. `sorted.by` can be specified by using "pValue" (p value), "log2Ratio" (log2 of fold-change value) or "F" (F statistics). `top` is used to specified number of genes being printed

Author(s)

Xiwei Wu, Arthur Li

See Also

[regressResult](#) [interactionResult](#)

Examples

```
data(eSetExample)
design<- new("designMatrix", target=pData(eSetExample), covariates = "Treatment")
contrast<- new("contrastMatrix", design.matrix = design,
  compare1 = "Treated", compare2 = "Control")
result<- regress(eSetExample, contrast)
Sort(result)
```

Index

*Topic classes

contrastMatrix, 2
designMatrix, 7
interactionResult-class, 22
regressResult-class, 35

*Topic datasets

eSetExample, 8
exprsExample, 8
pDataExample, 27
QC, 33

adjustment, 1

adjustment, interactionResult-method
(*interactionResult-class*),
22

adjustment, regressResult-method
(*regressResult-class*), 35

class:contrastMatrix
(*contrastMatrix*), 2

class:designMatrix
(*designMatrix*), 7

class:interactionResult
(*interactionResult-class*),
22

class:regressResult
(*regressResult-class*), 35

contrastMatrix, 2, 7, 12, 13, 18, 19

contrastMatrix-class
(*contrastMatrix*), 2

createExpressionSet, 3

createGSEAFfiles, 4, 25, 26

createIndex, 5

createIngenuityFile, 6, 27

designMatrix, 2, 3, 7, 13, 14, 18, 22

designMatrix-class
(*designMatrix*), 7

eSetExample, 8

ExpressionSet, 4

exprsExample, 8

geneFilter, 9

getAdjP, 10

getAdjP, interactionResult-method
(*interactionResult-class*),
22

getAdjP, regressResult-method
(*regressResult-class*), 35

getAnnotation, 11

getAnnotation, interactionResult-method
(*interactionResult-class*),
22

getAnnotation, regressResult-method
(*regressResult-class*), 35

getCompare1, 11

getCompare1, contrastMatrix-method
(*contrastMatrix*), 2

getCompare2, 12

getCompare2, contrastMatrix-method
(*contrastMatrix*), 2

getContrast, 12

getContrast, contrastMatrix-method
(*contrastMatrix*), 2

getContrast, interactionResult-method
(*interactionResult-class*),
22

getContrast, regressResult-method
(*regressResult-class*), 35

getCovariates, 13

getCovariates, designMatrix-method
(*designMatrix*), 7

getDesign, 13

getDesign, designMatrix-method
(*designMatrix*), 7

getF, 16

getF, interactionResult-method
(*interactionResult-class*),
22

getF, regressResult-method
(*regressResult-class*), 35

getFC, 15

getFC, interactionResult-method
(*interactionResult-class*),
22

getFC, regressResult-method
(*regressResult-class*), 35

- getFCCutoff, [14](#)
- getFCCutoff, interactionResult-method
(*interactionResult-class*),
[22](#)
- getFCCutoff, regressResult-method
(*regressResult-class*), [35](#)
- getFileName, interactionResult-method
(*interactionResult-class*),
[22](#)
- getFileName, regressResult-method
(*regressResult-class*), [35](#)
- getFilterMethod, [15](#)
- getFilterMethod, interactionResult-method
(*interactionResult-class*),
[22](#)
- getFilterMethod, regressResult-method
(*regressResult-class*), [35](#)
- getID, [16](#)
- getID, interactionResult-method
(*interactionResult-class*),
[22](#)
- getID, regressResult-method
(*regressResult-class*), [35](#)
- getIndex, [17](#)
- getIndex, interactionResult-method
(*interactionResult-class*),
[22](#)
- getIndex, regressResult-method
(*regressResult-class*), [35](#)
- getInteraction, [18](#)
- getInteraction, contrastMatrix-method
(*contrastMatrix*), [2](#)
- getIntIndex, [18](#)
- getIntIndex, designMatrix-method
(*designMatrix*), [7](#)
- getLength, [19](#)
- getLength, interactionResult-method
(*interactionResult-class*),
[22](#)
- getLevel, [19](#)
- getLevel, contrastMatrix-method
(*contrastMatrix*), [2](#)
- getNormalizationMethod, [20](#)
- getNormalizationMethod, interactionResult-method
(*interactionResult-class*),
[22](#)
- getNormalizationMethod, regressResult-method
(*regressResult-class*), [35](#)
- getP, [21](#)
- getP, interactionResult-method
(*interactionResult-class*),
[22](#)
- getP, regressResult-method
(*regressResult-class*), [35](#)
- getPCutoff, [20](#)
- getPCutoff, interactionResult-method
(*interactionResult-class*),
[22](#)
- getPCutoff, regressResult-method
(*regressResult-class*), [35](#)
- getTarget, [21](#)
- getTarget, designMatrix-method
(*designMatrix*), [7](#)
- help, [29](#)
- hugene10stCONTROL, [22](#)
- initialize, contrastMatrix-method
(*contrastMatrix*), [2](#)
- initialize, designMatrix-method
(*designMatrix*), [7](#)
- interactionResult, [1](#), [10](#), [11](#), [14–17](#),
[19–21](#), [34](#), [38](#)
- interactionResult
(*interactionResult-class*),
[22](#)
- interactionResult-class, [22](#)
- list, [23](#)
- mogene10stCONTROL, [24](#)
- output.cls, [5](#), [25](#), [26](#)
- output.gct, [5](#), [25](#), [26](#)
- output.ing, [26](#)
- Output2HTML, [24](#)
- Output2HTML, interactionResult-method
(*interactionResult-class*),
[22](#)
- Output2HTML, regressResult-method
(*regressResult-class*), [35](#)
- pDataExample, [27](#)
- postInteraction, [22](#), [28](#)
- preProcess3prime, [29](#)
- preProcessGeneST, [29](#)
- qa3prime, [30](#)
- qaGeneST, [32](#)
- QC, [33](#)
- regress, [34](#), [35](#)
- regressionMethod, [33](#)
- regressionMethod, interactionResult-method
(*interactionResult-class*),
[22](#)

regressionMethod, regressResult-method
 (*regressResult-class*), 35
regressResult, 1, 10, 11, 14–17, 20, 21,
 23, 34, 38
regressResult
 (*regressResult-class*), 35
regressResult-class, 35

selectSigGene, 37
selectSigGene, interactionResult-method
 (*interactionResult-class*),
 22
selectSigGene, regressResult-method
 (*regressResult-class*), 35
selectSigGeneInt, 36
show, contrastMatrix-method
 (*contrastMatrix*), 2
show, designMatrix-method
 (*designMatrix*), 7
show, interactionResult-method
 (*interactionResult-class*),
 22
show, regressResult-method
 (*regressResult-class*), 35
Sort, 38
Sort, interactionResult-method
 (*interactionResult-class*),
 22
Sort, regressResult-method
 (*regressResult-class*), 35
summary, interactionResult-method
 (*interactionResult-class*),
 22
summary, regressResult-method
 (*regressResult-class*), 35

vector, 23