Package ‘ArrayTools’

March 28, 2017

Type Package

Title geneChip Analysis Package

Version 1.34.0

Date 2009-07-22

Author Xiwei Wu, Arthur Li

Depends R (>= 2.7.0), affy (>= 1.23.4), Biobase (>= 2.5.5), methods

Imports affy, Biobase, graphics, grDevices, limma, methods, stats, utils, xtable

Suggests simpleaffy, R2HTML, affydata, affyPLM, genefilter, annaffy, gcma, hugene10sttranscriptcluster.db


Maintainer Arthur Li <xue11@coh.org>

Description This package is designed to provide solutions for quality assessment and to detect differentially expressed genes for the Affymetrix GeneChips, including both 3’-arrays and gene 1.0-ST arrays. The package generates comprehensive analysis reports in HTML format. Hyperlinks on the report page will lead to a series of QC plots, processed data, and differentially expressed gene lists. Differentially expressed genes are reported in tabular format with annotations hyperlinked to online biological databases.

License LGPL (>= 2.0)

LazyLoad yes

biocViews Microarray, OneChannel, QualityControl, Preprocessing, StatisticalMethod, DifferentialExpression, Annotation, ReportWriting, Visualization

NeedsCompilation no
R topics documented:

adjustment ....................................................... 3
contrastMatrix ............................................... 3
createExpressionSet ........................................ 5
createGSEAFiles ............................................ 6
createIndex .................................................... 7
createIngenuityFile ........................................ 8
designMatrix .................................................. 9
eSetExample ................................................... 10
exprsExample .................................................. 10
geneFilter ..................................................... 11
getAdjP ........................................................ 12
getAnnotation ............................................... 12
getCompare1 .................................................. 13
getCompare2 .................................................. 14
getContrast ................................................... 14
getCovariates ............................................... 15
getDesign ..................................................... 15
getF ........................................................... 16
getFC .......................................................... 17
getFCCutoff ................................................... 17
getFilterMethod ............................................. 18
getID .......................................................... 18
getIndex ......................................................... 19
getInteraction ................................................ 20
getIntIndex .................................................... 20
getLength ...................................................... 21
getLevel ........................................................ 21
getNormalizationMethod ................................. 22
getP ............................................................. 22
getPCutoff ..................................................... 23
getTarget ....................................................... 23
hugene10stCONTROL ........................................ 24
interactionResult-class .................................... 24
mogene10stCONTROL ......................................... 26
output.cls ..................................................... 26
output.gct ..................................................... 27
output.ing ..................................................... 27
Output2HTML ................................................... 28
pDataExample ................................................ 29
postInteraction .............................................. 29
preProcess3prime .......................................... 30
preProcessGeneST ........................................... 31
qa3prime ....................................................... 32
qaGeneST ....................................................... 33
QC .............................................................. 34
regress ........................................................ 35
regressionMethod .......................................... 36
regressResult-class ....................................... 36
selectSigGene ................................................ 38
selectSigGeneInt ............................................ 38
adjustment

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

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"
contrastMatrix

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")))
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


See Also

ExpressionSet
**Examples**

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

**Description**

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

**Usage**

```r
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**

```r
data(eSetExample)
## Not run: createGSEAFiles (mydir, eSetExample, "Treatment")
```
createIndex

**Creating an HTML index file**

**Description**

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

**Usage**

```r
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", "Group"),
    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)
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 covaraite 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

covariates signature(object = "designMatrix"): access the covariates slot
design signature(object = "designMatrix"): access the design slot
intIndex signature(object = "designMatrix"): access the intIndex slot
target 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")))
**exprsExample**

```r
## 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**

```r
data(eSetExample)
```

**Format**

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

**Examples**

```r
data(eSetExample)
```

---

**exprsExample**

*a data.frame contains expression data*

**Description**

a data.frame contains expression data

**Usage**

```r
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
geneFilter

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 filter an ExpressionSet using different methods

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 = 4, range = 0, iqrPct = 0, output = FALSE, mydir = getwd())

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 specified 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
getAdjP

Value

a filtered ExpressionSet

Author(s)

Xiwei Wu, Arthur Li

Examples

data(eSetExample)
filtered <- geneFilter(eSetExample)

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

access the annotation slot from the regressResult or interactionResult slot

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

Access the Compare1 slot from the contrastMatrix

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
getContrast

See Also

contrastMatrix

getCompare2

Access the compare2 slot from the contrastMatrix class

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

Access the contrast matrix from the contrastMatrix class

Description

Access the contrast matrix from the contrastMatrix class

Usage

getContrast(object)

Arguments

object

a contrastMatrix class

Value

a numeric matrix
**getCovariates**

**Author(s)**

Xiwei Wu, Arthur Li

**See Also**

`contrastMatrix`

---

**Description**

Accessing the covariates from the designMatrix class

**Usage**

```r
covariates <- 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**

```r
design <- getDesign(object)
```

**Arguments**

- `object`: a designMatrix class
getF

Value

a matrix containing the designMatrix

Author(s)

Xiwei Wu, Arthur Li

See Also

designMatrix

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
getFC

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

getFCCutoff

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
getFilterMethod

Access the filterMethod slot from the regressResult or interactionResult class

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

getID

access the ID slot from the regressResult or interactionResult class

Description
access the ID slot from the regressResult or interactionResult class

Usage
getID(object)

Arguments
object a regressResult class or interactionResult class
getIndex

Value

a character vector

Author(s)

Xiwei Wu, Arthur Li

See Also

regressResult interactionResult

getIndex(object)

Description

Access the SignificantIndex slot from the regressResult or interactionResult class

Usage

gIndex(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
c contrastMatrix

gIntIndex

Access the IntIndex slot from the designMatrix class

Description
Access the IntIndex slot from the designMatrix class

Usage
gIntIndex(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

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
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

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

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 regressResult class. The first component contains all the genes. The second component contains genes with the interaction effect. The rest components contain 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.
interactionResult-class

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
getNormalMethod 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", "Group"),
               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

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.

output.cls

Create *.CLS file for GSEA analysis

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
- normal: an ExpressionSet
- filename: 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
- allfile: a list of regressResult class
- eSet: an ExpressionSet
- filename: file name
Value

create an txt file for Ingenuity Analysis

Author(s)

Xiwei Wu, Arthur Li

References

http://www.ingenuity.com/

See Also

createIngenuityFile

Output2HTML

Creating HTML file for regressResult or interactionResult class

Description

Creating HTML file for regressResult or interactionResult class

Usage

Output2HTML(object, ...)

Arguments

object an regressResult or interactionResult class
...

you can specify the directory to store the result by using the mydir argument. The default value of mydir 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)
**pDataExample**

A phenoData example

**Description**

A data frame contains the phenotype data

**Usage**

```r
data(pDataExample)
```

**Format**

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

- **Treatment** a character vector
- **Group** a character vector

**Examples**

```r
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 contain the genes with the interaction across different levels of covariates.

**Usage**

```r
postInteraction(eSet, regressObject, mainVar, compare1, compare2, method = regressionMethod(regressObject), adj = adjustment(regressObject))
```

**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.
**preProcess3prime**

**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", "Group"), 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 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 = getwd())`

**Arguments**

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

**Value**

an ExpressionSet

**Author(s)**

Xiwei Wu, Arthur Li

**See Also**

~~~objects to See Also as help, ~~~
preProcessGeneST  

**Preprocess genechip ST array**

**Description**

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

**Usage**

```r
preProcessGeneST(object, offset = 1, rmControl = TRUE, output = FALSE, mydir = getwd())
```

**Arguments**

- `object`: an ExpressionSet.
- `offset`: The offset is added to the expression value to avoid log2(0) = -Inf.
- `rmControl`: Setting `rmControl = TRUE` to remove control probes.
- `output`: If `output = TRUE`, it will output the preprocessed data in the specified directory from the `mydir` argument.
- `mydir`: specified directory to contain the output

**Value**

an ExpressionSet

**Author(s)**

Xiwei Wu, Arthur Li

**Examples**

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

<table>
<thead>
<tr>
<th>argument</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>an AffyBatch object</td>
</tr>
<tr>
<td>parameters</td>
<td>The names of the variables to be included in the report</td>
</tr>
<tr>
<td>outputFile</td>
<td>The name of the output file. Make sure write &quot;.html&quot;</td>
</tr>
<tr>
<td>mydir</td>
<td>The name of the directory containing the report</td>
</tr>
</tbody>
</table>

Details

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

Figure 1: The Raw Intensity Plot. The raw intensity should be similar across all chips.

Figure 2: 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.

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

Figure 4: The Hybridization Controls Plot. BioB, BioC, BioD, CreX should be called present, except that it is acceptable if BioB is absent sometimes.

Figure 5: 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.

Figure 6: 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.

Figure 7: 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.

Figure 8: 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.
Figure 9: 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

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

qaGeneST

Creating Quality Assessment Report for Gene ST Array

Description

Creating Quality Assessment Report for Gene ST Array in HTML file

Usage

```r
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 output file. Make sure write ".html"

Details

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

Figure 1: The intensity distribution Plot. The raw intensity should be similar across all chips

Figure 2: The Mean Signal Plot. The mean signal of each group should be consistent across the samples. The positive control should be higher than the negative controls.
Figure 3: BAC SPIKE plot. The mean signal of each group should be consistent across the samples. The signal for BioB should be the lowest, followed by BioC, BioD, and CreX (the highest).

Figure 4: POLYA SPIKE plot. The mean signal of each group should be consistent across the samples. The signal for Lys should be the lowest, followed by Thr, Phe, and Dap.

Figure 5: 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.

Figure 6: 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.

Figure 7: 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.

Figure 8: 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


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**

```r
data(QC)
```

**Examples**

```r
data(QC)
```

---

**regress**  \hspace{1cm} \textit{Run regression to fit genewise linear model}

**Description**

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

**Usage**

```r
regress(object, contrast, method = c("limma", "regression", "permutation"), adj = "none", permute.time = 1000)
```

**Arguments**

- `object`: an ExpressionSet
- `contrast`: a contrastMatrix
- `method`: 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.
- `permute.time`: number of permutation times, only used for the "permutation" method

**Value**

an object of `regressResult`

**Author(s)**

Xiwei Wu, Arthur Li

**Examples**

```r
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

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

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
significantFCutOff: 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
getAddress 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)
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))

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)
Sort

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", "Group"),
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)

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 regressResult or an interactionResult class
... any other arguments. See below...

Value

if sorting a regressResult, returned value is a data frame if sorting a interactionResult, returned value is a list of data frames
Sort a regressResult or an interactionResult class

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

x is a regressResult 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, 3
designMatrix, 9
 interactionResult-class, 24
 regressResult-class, 36
*Topic datasets
eSetExample, 10
eexprsExample, 10
 pDataExample, 29
 QC, 34
adjustment, 3
adjustment, interactionResult-method (interactionResult-class), 24
adjustment, regressResult-method (regressResult-class), 36
class: contrastMatrix (contrastMatrix), 3
class: designMatrix (designMatrix), 9
class: interactionResult (interactionResult-class), 24
class: regressResult (regressResult-class), 36
contrastMatrix, 3, 9, 14, 15, 20, 21
contrastMatrix-class (contrastMatrix), 3
createExpressionSet, 5
createGSEAFiles, 6, 26, 27
createIndex, 7
createIngenuityFile, 8, 28
designMatrix, 4, 9, 15, 16, 20, 24
designMatrix-class (designMatrix), 9
eSetExample, 10
ExpressionSet, 5
exprsExample, 10
geneFilter, 11
getAdjP, 12
getAdjP, interactionResult-method (interactionResult-class), 24
getAdjP, regressResult-method (regressResult-class), 36
getAnnotation, 13
getAnnotation, interactionResult-method (interactionResult-class), 24
getAnnotation, regressResult-method (regressResult-class), 36
getCompare1, 13
getCompare1, contrastMatrix-method (contrastMatrix), 3
getCompare2, 14
getCompare2, contrastMatrix-method (contrastMatrix), 3
getContrast, 14
getContrast, contrastMatrix-method (contrastMatrix), 3
getContrast, interactionResult-method (interactionResult-class), 24
getContrast, regressResult-method (regressResult-class), 36
getCovariates, 15
getCovariates, designMatrix-method (designMatrix), 9
getDesign, 15
getDesign, designMatrix-method (designMatrix), 9
getF, 16
getF, interactionResult-method (interactionResult-class), 24
getF, regressResult-method (regressResult-class), 36
getFC, 17
getFC, interactionResult-method (interactionResult-class), 24
getFC, regressResult-method (regressResult-class), 36
getFCCutoff, 17
getFCCutoff, interactionResult-method (interactionResult-class), 24
getFCCutoff, regressResult-method (regressResult-class), 36
getFileName, interactionResult-method (interactionResult-class), 24
getFileName, regressResult-method (regressResult-class), 36
getFilterMethod, 18
INDEX

getFilterMethod, interactionResult-method (interactionResult-class), 24
getFilterMethod, regressResult-method (regressResult-class), 36
getID, 18
getID, interactionResult-method (interactionResult-class), 24
getID, regressResult-method (regressResult-class), 36
getIndex, 19
getIndex, interactionResult-method (interactionResult-class), 24
getIndex, regressResult-method (regressResult-class), 36
getInteraction, 20
getInteraction, contrastMatrix-method (contrastMatrix), 3
getIntIndex, 20
getIntIndex, designMatrix-method (designMatrix), 9
getLength, 21
getLength, interactionResult-method (interactionResult-class), 24
getLevel, 21
getLevel, contrastMatrix-method (contrastMatrix), 3
getNormalizationMethod, 22
getNormalizationMethod, interactionResult-method (interactionResult-class), 24
getNormalizationMethod, regressResult-method (regressResult-class), 36
getP, 22
getP, interactionResult-method (interactionResult-class), 24
getP, regressResult-method (regressResult-class), 36
getPCutoff, 23
getPCutoff, interactionResult-method (interactionResult-class), 24
getPCutoff, regressResult-method (regressResult-class), 36
getTarget, 23
getTarget, designMatrix-method (designMatrix), 9
help, 30
hugene10stCONTROL, 24
initialize, contrastMatrix-method (contrastMatrix), 3
initialize, designMatrix-method (designMatrix), 9
interactionResult, 3, 12, 13, 16–19, 21–23, 36, 40
interactionResult (interactionResult-class), 24
interactionResult-class, 24
list, 24
mogene10stCONTROL, 26
output.cls, 6, 26, 27
output.gct, 6, 26, 27
output.ing, 27
Output2HTML, 28
Output2HTML, interactionResult-method (interactionResult-class), 24
Output2HTML, regressResult-method (regressResult-class), 36
pDataExample, 29
postInteraction, 24, 29
preProcess3prime, 30
preProcessGeneST, 31
qa3prime, 32
qaGeneST, 33
QC, 34
ingress, 35, 36
regressionMethod, 36
regressionMethod, interactionResult-method (interactionResult-class), 24
regressionMethod, regressResult-method (regressResult-class), 36
regressResult, 3, 12, 13, 16–19, 22, 23, 25, 36, 40
regressResult (regressResult-class), 36
regressResult-class, 36
selectSigGene, 38
selectSigGene, interactionResult-method (interactionResult-class), 24
selectSigGene, regressResult-method (regressResult-class), 36
selectSigGeneInt, 38
show, contrastMatrix-method (contrastMatrix), 3
show, designMatrix-method (designMatrix), 9
show, interactionResult-method (interactionResult-class), 24
show, regressResult-method (regressResult-class), 36
Sort, 39
INDEX

Sort, interactionResult-method
  (interactionResult-class), 24
Sort, regressResult-method
  (regressResult-class), 36
summary, interactionResult-method
  (interactionResult-class), 24
summary, regressResult-method
  (regressResult-class), 36

vector, 24