Package 'AffyExpress'

April 4, 2014

Version 1.28.0

Date 2009-07-22

Title Affymetrix Quality Assessment and Analysis Tool

Author Xiwei Wu <xwu@coh.org>, Xuejun Arthur Li <xueli@coh.org>

Maintainer Xuejun Arthur Li <xueli@coh.org>

Depends R (>= 2.10), affy (>= 1.23.4), limma

Suggests simpleaffy, R2HTML, affyPLM, hgu95av2cdf, hgu95av2, test3cdf,genefilter, estrogen, annaffy, gcrma

Description The purpose of this package is to provide a comprehensive and easy-to-use tool for quality assessment and to identify differentially expressed genes in the Affymetrix gene expression data.

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

License LGPL

R topics documented:

`AffyInteraction` *Analyze interaction effect and produce output*

Description

This wrapper function will create a design matrix and contrast matrix for the interaction test. Then it will fit linear model to test for interaction effect for each gene and identify genes for which interaction test is being significant. For genes having the interaction effect, it fits linear model for each genes in each level of strata.var. For genes don't have interaction effect, it fits linear model for each genes without stratifying the strata.var. In the end, it will output signficant result.

Usage

```
AffyInteraction(object, method, main.var, strata.var, compare1, compare2,
  covariates=NULL, p.int=0.05, m.int=0, adj.int="none", p.value=0.05,
  m.value=0, adj="none", filename1="result", filename2="inter_result")
```

Arguments

`object`	an "ExpressionSet"
`method`	Three methods are supported by this function: "L" for using LIMMA method - compute moderated t-statistics and log-odds of differential expression by empirical Bayes shrinkage of the standard errors towards a common value; "F" for using ordinary linear regression; "P" for permuation test by resampling the phenotype
`main.var`	the variable of your main interest
`strata.var`	a categorical variable serves as a potential effect modifier. An effect modifier is a variable that modifies the association between outcome variable and the main variable. If the interaction exits, the association between the outcome and main.var will be analyzed separately within each stratum of strata.var
`compare1`	first value of the variable of main interest. Suppose the main variable is "estrogen", and its has two values: "present" and "absent". You would like to compare "present" versus "absent". Then you will use compare1 = "present"
`compare2`	second value of the variable of main interest. Follow from the same example above, you will set compare2 = "absent"
`covariates`	a list of covariates, not including main.var and strata.var, the default value is NULL
`p.int`	p value for the interaction test
`m.int`	fold change cut-off value for the interaction test
`adj.int`	adjustment method for multiple comparison for testing interaction, including "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none". Type help(p.adjust) for more detail.
`p.value`	p value for main effect test

m.value	fold change cut-off value for main effect test
adj	adjustment method for multiple comparison for testing main effect
filename1	name of the output file for the main effect
filename2	name of the output file for the interaction test

Value

a list of data frame: The first data frame contains results for genes with interaction effect. The second data frame contains results for genes don't have interaction effect. The rest of the data frames contain information for main effects for each stratum of strata.var.

Author(s)

Xiwei Wu <xwu@coh.org>, Xuejun Arthur Li <xueli@coh.org>

Examples

```
## Suppose that you would like to test the interaction effect between "gender"
## and "group" variables, "gender" is the main effect variable.  For genes in
## which the interaction test are being significant, you would like to compare
## "M" vs. "F" within each level of "group".  For genes in which the interaction
## test are not being significant, you would like to compare "M" vs. "F" without
## stratifying "group".

data(testData)
normaldata<-pre.process("rma",testData)
result<-AffyInteraction(normaldata, "L", "gender", "group","M", "F", p.int=0.05,
  m.int=0, adj.int="none", p.value=0.05, m.value=0, adj="none",
  filename1="result1", filename2="result2")
```

AffyQA *Microarray quality control and assessment*

Description

Create quality control report in an HTML file that contains a set of assessment plots, including Affymetrix recommended quality assessment, RNA quality assessment, sample quality assessment, quality diagnostic using PLM (pseudo-chip images and NUSE and RLE plots) in your current working directory.

Usage

```
AffyQA(parameters, raw, Output="AffyQA.html")
```

Arguments

parameters	a list of variables
raw	an 'AffyBatch'
Output	name of the output file

Value

This function only create an html file

Author(s)

Xiwei Wu <xwu@coh.org>, Xuejun Arthur Li <xueli@coh.org>

Examples

```
## Not run:
data(testData)
AffyQA(parameters=c("group", "gender"), testData)

## End(Not run)
```

AffyRegress *Select differentially expressed genes, and output the result to a html file*

Description

This is a wrapper function complete the following tasks: 1. Create a design matrix 2. Create a contrast 3. Run regression 4. Select differentaly expressed gene 5. Output the differentally expressed gene to a html file

Usage

```
AffyRegress(normal.data, cov, compare1, compare2, method,
    int=NULL, level=NULL, adj="none", p.value=0.05, m.value=0,
filename="result")
```

Arguments

normal.data	an 'ExpressionSet'
cov	a list of 1-n covariates
compare1	the first 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

compare2	the second value of the main covariate. Based on the previoius example, if you would like to compare "drug1" vs "drug2", then you would use "drug2" as compare2
method	Three methods are supported by this function: "L" for using LIMMA method - compute moderated t-statistics and log-odds of differential expression by empirical Bayes shrinkage of the standard errors towards a common value; "F" for using ordinary linear regression; "P" for permuation test by resampling the phenotype
int	if int=NULL, the interaction effect is not considered; otherwise, use two integers to indicate which covariates are considered for interaction effect. For example, if cov<-c("estrogen","drug","time") and you are considering the interaction between "estrogen" and "time", then you would write int=c(1,3)
level	you only specify this term 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.
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.
p.value	p value, the default value is 0.05
m.value	fold change cut-off value, default value is 0
filename	name of the output file

Value

A dataframe which has the same format as the one created by select.sig.gene function.

Author(s)

Xiwei Wu <xwu@coh.org>, Xuejun Arthur Li <xueli@coh.org>

Examples

```
data(testData)
normaldata<-pre.process("rma",testData)
result<-AffyRegress(normaldata, "group", "A", "C", "L")
```

Filter *filter an 'ExpressionSet' using different methods*

Description

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

Usage

```
Filter(object, numChip=1, bg=0, range=0, iqrPct=0)
```

Arguments

object	an 'ExpressionSet'
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.

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. Calulating the interquatile 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 <xwu@coh.org>, Xuejun Arthur Li <xueli@coh.org>

Examples

```
data(testData)
normaldata<-pre.process("rma",testData)

##At least one chip is greater than 4
filtered.1<-Filter(normaldata, numChip=1, bg=4)

##At least one chip is greater than 4 and range >0.5
filtered.2<-Filter(normaldata, numChip=1, bg=4, range=0.5)

##range >0.5 and IQR > 20percentile
filtered.3<-Filter(normaldata, range=0.5, iqrPct=0.2)
```

`import.data` *Importing Affy Cel File*

Description

This function imports cel files and create an 'AffyBatch' based on the imported cel

Usage

```
import.data(phenotype.file, path=getwd(), ...)
```

Arguments

`phenotype.file`	the name of the phenotype file - text file
`path`	the name of the directory storing phenotype.file and the cel files, the default value is the current working directory
`...`	Refers to "read.AnnotatedDataFrame" (Biobase)

Value

an 'AffyBatch'

Author(s)

Xiwei Wu <xwu@coh.org>, Xuejun Arthur Li <xueli@coh.org>

Examples

```
## Not run:
datadir <- system.file("extdata", package = "estrogen")
raw <- import.data("phenodata.txt", path=datadir, header=TRUE, sep="",
    row.names="filename")

## End(Not run)
```

`interaction.result2html` *output differentially expressed genes for the interaction model to a HTML file*

Description

output differentially expressed genes for the interaction model to a HTML file. It contais the following columns: ProbeID, Symbol, Description, GenBank, LocusLink, Log2ratio for each stratum, p value for each stratum, and interaction p value.

Usage

```
interaction.result2html(cdf.name, result, inter.result,filename="inter_result")
```

Arguments

cdf.name	cdf name which can be obtained from annotation function
result	a list of data frame returned from post.interaction function
inter.result	a data frame returned from select.sig.gene function, this is the result based on testing the interaction effect.
filename	the name of the output file

Author(s)

Xiwei Wu <xwu@coh.org>, Xuejun Arthur Li <xueli@coh.org>

Examples

```
data(testData)
normaldata<-pre.process("rma",testData)

## Create design matrix for interaction effect between "group"
## and "gender"
design.int<-make.design(pData(normaldata), c("group", "gender"), int=c(1,2))

## Create the interaction contrast
contrast.int<-make.contrast(design.int, interaction=TRUE)

## Run Regression to detect interaction effect
result.int<-regress(normaldata, design.int, contrast.int, "L")

## Select differentally expressed genes based on p.value
select.int<-select.sig.gene(result.int, p.value=0.05)

## Identify genes with the interaction effect
sig.ID<-select.int$ID[select.int$significant==TRUE]
sig.index<-match(sig.ID, rownames(exprs(normaldata)))

## Create separate tables for each level of effect modifier
result<-post.interaction("group","M", "F", design.int, normaldata[sig.index,],
  "L","none", 0.05, log2(1.5))

## Output significant result for the interaction model
interaction.result2html(annotation(normaldata), result, result.int, filename="interaction")
```

make.contrast *Create a contrast matrix*

Description

Create a contrast matrix based on a given design matrix

Usage

```
make.contrast(design.matrix, compare1=NULL, compare2=NULL, level=NULL,
  interaction=FALSE)
```

Arguments

design.matrix	a design matrix returned from the make.design funcion
compare1	the first 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. If interaction==TRUE, do not specify this value.
compare2	the second value of the main covariate. Based on the example above, if you would like to compare "drug1" vs "drug2", then you would use "drug2" as compare2. If interaction ==TRUE, do not specify this value
level	you only specify this term 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
interaction	you only specify interaction=TRUE when you would like to detect the interaction effect between two covariates. When you specify interaction=TRUE, do not provide values for compare1, compare2, and level

Value

contrast matrix for the linear model

Author(s)

Xiwei Wu <xwu@coh.org>, Xuejun Arthur Li <xueli@coh.org>

See Also

make.design

Examples

```
target<-data.frame(drug=(c(rep("A",4),rep("B",4),rep("C",4))),
gender=factor(c(rep("M",6),rep("F",6))),
group=factor(rep(c(1,2,3),4)))

# Example1:  Compare drug "A" vs. "B"
design1<-make.design(target, "drug")
contrast1<-make.contrast(design1, "A", "B")

# Example2:  Compare drug "A" vs. "B", adjusting for "group" variable
design2<-make.design(target, c("drug","group"))
contrast2<-make.contrast(design2, "A", "B")

# Example3:  Suppose you are interested in "drug", "group" interaction
design3<-make.design(target, c("drug","group"), int=c(1,2))
contrast3<-make.contrast(design3, interaction=TRUE)

# Example4:  Compare drug "A" vs. "B" among "male"
# Notice that you must use an design matrix containing an interaction term
design4<-make.design(target, c("drug","gender"), int=c(1,2))
contrast4<-make.contrast(design4, "A", "B", "M")
```

`make.design` *Create a Design Matrix*

Description

Create a design matrix for a linear model

Usage

```
make.design(target, cov, int=NULL)
```

Arguments

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

Value

a matrix containing design matrix for the linear model

Author(s)

Xiwei Wu <xwu@coh.org>, Xuejun Arthur Li <xueli@coh.org>

See Also

make.contrast

Examples

```
target<-data.frame(drug=(c(rep("A",4),rep("B",4),rep("C",4))),
gender=factor(c(rep("M",6),rep("F",6))),
group=factor(rep(c(1,2,3),4)))

#To create a design matrix using "drug", "gender" as covariates
design1<-make.design(target, c("drug","gender"))

#To create a design matrix by using "drug","gender","group" as covariates,
#and consider the interaction effect of "drug" and "group"
design2<-make.design(target, c("drug","gender", "group"), int=c(1,3))
```

---

post.interaction *Create a list of tables for the result based on the main effect variable*

Description

Once the interaction effect is being detected for a list of genes, that means that for these group of genes, the main effect is different across different level of another variable (call it effect modifier). This function will create a list of data frame for each level of the effect modifier.

Usage

```
post.interaction(strata.var, compare1, compare2, design.int, object, method, adj="none",
  p.value=0.05, m.value=0)
```

Arguments

strata.var	a categorical variable serves as a potential effect modifier
compare1	the first 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
compare2	the second value of the main covariate. Based on the example above, if you would like to compare "drug1" vs "drug2", then you would use "drug2" as compare2
design.int	the design matrix of the interaction effect
object	an "ExpressionSet"

method	It is used to run regression within each level of the effect modifier. Three methods are supported by this function: "L" for using LIMMA method - compute moderated t-statistics and log-odds of differential expression by empirical Bayes shrinkage of the standard errors towards a common value, "F" for using ordinary linear regression, "P" for permuation test by resampling the phenotype
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.
p.value	a p-value used to select significant gene within each level of the effect modifier
m.value	a fold change value used to select significant gene within each level of the effect modifier

Value

a list of data frame. The length of the data frame equals on number of levels of effect modifier.

Each dataframe contains rows for all the genes from object and the following columns: ID (probeid); Log2Ratio (estimate of the effect or the contrast, on the log2 scale); F (F statistics); P.Value (raw p-value); adj.P.Value (adjusted p-value or q-value); significant (either TRUE or FALSE based on p.value and m.value)

Author(s)

Xiwei Wu <xwu@coh.org>, Xuejun Arthur Li <xueli@coh.org>

Examples

```
data(testData)
normaldata<-pre.process("rma",testData)

## Create design matrix for interaction effect between "group"
## and "gender"
design.int<-make.design(pData(normaldata), c("group", "gender"), int=c(1,2))

## Create the interaction contrast
contrast.int<-make.contrast(design.int, interaction=TRUE)

## Run Regression to detect interaction effect
result.int<-regress(normaldata, design.int, contrast.int, "L")

## Select differentally expressed genes based on p.value
select.int<-select.sig.gene(result.int, p.value=0.05)

## Identify genes with the interaction effect
sig.ID<-select.int$ID[select.int$significant==TRUE]
sig.index<-match(sig.ID, rownames(exprs(normaldata)))

## Create separate tables for each level of effect modifier
result<-post.interaction("group","M", "F", design.int, normaldata[sig.index,],
  "L","none", 0.05, log2(1.5))
```

pre.process *Data Preprocessing*

Description

This function converts an 'AffyBatch' into an 'ExpressionSet' using either RMA or GCRMA methods

Usage

```
pre.process(method, raw, plot=FALSE, output=FALSE)
```

Arguments

method	either "rma" or "gcrma"
raw	an 'AffyBatch'
plot	if plot = TRUE, it will plot gene expression for each chip
output	if output = TRUE, it will output 'ExpressionSet' to the current directory

Value

an 'ExpressionSet'

Author(s)

Xiwei Wu <xwu@coh.org>, Xuejun Arthur Li <xueli@coh.org>

Examples

```
data(testData)
normaldata<-pre.process("rma",testData)
```

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, design, contrast, method, adj="none", permute.time=1000)
```

Arguments

object	an "ExpressionSet"
design	design matrix from the make.design function
contrast	contrast matrix from the make.contrast function
method	Three methods are supported by this function: "L" for using LIMMA method - compute moderated t-statistics and log-odds of differential expression by empirical Bayes shrinkage of the standard errors towards a common value, "F" for using ordinary linear regression, "P" for permuation test by resampling the phenotype
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

A dataframe contains rows for all the genes from object and the following columns: ID(probeid); Log2Ratio (estimate of the effect or the contrast, on the log2 scale); F (F statistics); P.Value (raw p-value); adj.P.Value (adjusted p-value or q-value)

Author(s)

Xiwei Wu <xwu@coh.org>, Xuejun Arthur Li <xueli@coh.org>

References

Smyth, G.K. (2005) Limma: linear models for microarray data. In: Bioinformatics and Computational Biology Solutions using R and Bioconductor, R. Gentleman, V. Carey, S. Dudoit, R. Irizarry, W. Huber (eds.), Springer, New York, pages 397-420

Examples

```
data(testData)
normaldata<-pre.process("rma",testData)

## Create design matrix
design<-make.design(pData(normaldata), "group")

## Create contrast matrix - Compare group "A" vs. "C"
contrast<-make.contrast(design, "A", "C")

## Identify differentially expressed gene by using LIMMA method
result<-regress(normaldata, design, contrast, "L")
```

result2html *output differentially expressed genes to a HTML file*

Description

output differentially expressed genes to a HTML file based on a result table from the select.sig.gene function. It contais the following columns: Probe, Symbol, Description, GenBank, LocusLink, Log2ratio, and p value.

Usage

```
result2html(cdf.name, result, filename="result")
```

Arguments

cdf.name	cdf name which can be obtained from the annotation function
result	a data frame returned from the gene.select function
filename	a file name for the output

Author(s)

Xiwei Wu <xwu@coh.org>, Xuejun Arthur Li <xueli@coh.org>

Examples

```
data(testData)
normaldata<-pre.process("rma",testData)
## Create design matrix
design<-make.design(pData(normaldata), "group")

## Create contrast matrix - Compare group "A" vs. "C"
contrast<-make.contrast(design, "A", "C")

## Identify differentially expressed gene by using LIMMA method
result<-regress(normaldata, design, contrast, "L")

## Select differentially expressed gene based on p <0.05 and
## fold change >=log2(1.5)
select<-select.sig.gene(result, p.value=0.05, m.value=log2(1.5))

## Output differentially expressed gene to a example.html
result2html(annotation(normaldata), select, "example")
```

`select.sig.gene` *select differentially expressed genes based on p value and/or fold change*

Description

select differentially expressed genes based on p value and/or fold change

Usage

```
select.sig.gene(top.table, p.value =0.05, m.value =0)
```

Arguments

`top.table`	an data frame returned from the regress function
`p.value`	p value, the default value is 0.05
`m.value`	fold change cut-off value, default value is 0

Value

A dataframe which is the similar to the one returned from regress function. An additional column 'significant' is added to the table from the "regress" function. If p value < p.value and absolute of fold change value >=m.value then signiicant = TRUE, otherwise, significant = FALSE.

Author(s)

Xiwei Wu <xwu@coh.org>, Xuejun Arthur Li <xueli@coh.org>

Examples

```
data(testData)
normaldata<-pre.process("rma",testData)
## Create design matrix
design<-make.design(pData(normaldata), "group")

## Create contrast matrix - Compare group "A" vs. "C"
contrast<-make.contrast(design, "A", "C")

## Identify differentially expressed gene by using LIMMA method
result<-regress(normaldata, design, contrast, "L")

## Select differentially expressed gene based on p <0.05 and
## fold change >=log2(1.5)
select<-select.sig.gene(result, p.value=0.05, m.value=log2(1.5))
```

testData *AffyBatch instance testData*

Description

This hypothetical AffyBatch instance testData has two phenotype: group and gender.

Usage

```
data(testData)
```

Index