Package ‘a4Base’

May 2, 2024

Type Package

Title Automated Affymetrix Array Analysis Base Package

Version 1.52.0

Date 2020-10-26

Description
Base utility functions are available for the Automated Affymetrix Array Analysis set of packages.

Imports methods, graphics, grid, Biobase, annaffy, mpm, genefilter,
limma, multtest, glmnet, gplots

Depends a4Preproc, a4Core

Suggests Cairo, ALL, hgu95av2.db, nlcv

Enhances gridSVG, JavaGD

License GPL-3

biocViews Microarray

RoxygenNote 7.1.1

git_url https://git.bioconductor.org/packages/a4Base

git_branch RELEASE_3_19

git_last_commit 7312ef9

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-02

Author Willem Talloen [aut],
Tine Casneuf [aut],
An De Bondt [aut],
Steven Osselaer [aut],
Hinrich Goehlmann [aut],
Willem Ligtenberg [aut],
Tobias Verbeke [aut],
Laure Cougnaud [cre]

Maintainer Laure Cougnaud <laure.cougnaud@openanalytics.eu>
Contents

a4palette .................................................. 3
addQuantilesColors ..................................... 4
boxPlot ..................................................... 5
combineTwoExpressionSet ................................ 6
computeLogRatio .......................................... 7
createExpressionSet ...................................... 9
ExpressionSetWithComputation-class .................... 10
filterVarInt .............................................. 11
heatmap.expressionSet .................................. 12
histPvalue .................................................. 20
histpvalueplotter ........................................ 21
lassoReg .................................................... 22
limmaReg .................................................... 23
limmaTwoLevels .......................................... 24
logReg ...................................................... 25
nlcvTT ...................................................... 26
oaColors ................................................... 27
oaPalette .................................................. 28
plot1gene ................................................... 28
plotComb2Samples ........................................ 30
plotCombination2genes ................................... 31
plotCombMultSamples .................................... 33
plotLogRatio ............................................... 34
probabilitiesPlot ......................................... 38
probe2gene ................................................ 40
profilesPlot ............................................... 41
propDEgenes ................................................ 42
propDEgenes-methods ..................................... 43
propdegenescalculation .................................. 44
replicates .................................................. 45
spectralMap ............................................... 46
spectralMap-methods ..................................... 47
topTable,limma-method ................................... 48
tTest ......................................................... 50
volcanoPlot ............................................... 51
volcanoPlot-methods ..................................... 53
volcanoplotter ............................................ 56

Index ......................................................... 58
Utility function that defines a color palette for use in a4

Usage

a4palette(n, alpha = 1, Janssen = FALSE)

Arguments

- **n**: Number of color levels the palette should provide.
- **alpha**: Alpha transparency level of the colors.
- **Janssen**: Logical. If TRUE, Janssen Pharmaceutical colors are used (with a maximum of 6 possible colors).

Details

For n = 1, "blue" is returned; for n = 2 c("red", "blue") is returned; for n = 3 c("red", "green", "blue") is returned; for n = 4 c("red", "green", "blue", "purple") is returned and for n > 2, the output of rainbow(n) is returned.

Value

A character vector of colors.

Author(s)

Steven Osselaer, Tobias Verbeke

See Also

rainbow palette in palettes

Examples

```r
op <- par(mfrow = c(2, 3))
for (nGroups in 1:6)
  pie(rep(1, nGroups), a4palette(nGroups))
par(op)
```
addQuantilesColors  
Compute quantiles for plotGeneDE function

Description

Compute quantiles on mean expression level for plotGeneDE function. Colors of bars in the plot could then be allocated using buckets defined by those quantiles.

Usage

addQuantilesColors(e, ngroups = 3)

Arguments

e  ExpressionSet object to use for computation
ngroups  Number of groups to be created

Details

Number of computed quantiles is equal to (ngroups - 1).

Value

The ExpressionSet object e is returned, with a new column called colorsQuantilesVector in its slot featureData

Author(s)

Eric Lecoutre

See Also

plotLogRatio

Examples

if (require(ALL)){
  data(ALL, package = "ALL")
  ALLQ <- addQuantilesColors(ALL)
  fData(ALLQ)
  }

Description

Create a boxplot for a given gene. The boxplot displays the expression values (y-axis) by groupss (x-axis). The raw data are superimposed as dots, jittered for readability of the plot. Optionally, the dots can be colored by another variable.

Usage

```r
boxPlot(
  probesetId = NULL,
  geneSymbol = NULL,
  object,
  groups,
  main = NULL,
  colvec = NULL,
  colgroups = NULL,
  probe2gene = TRUE,
  addLegend = TRUE,
  legendPos = "topleft",
  ...
)
```

Arguments

- `probesetId`: The probeset ID. These should be stored in the `featureNames` of the `expressionSet` object.
- `geneSymbol`: The gene symbol. These should be stored in the column `Gene Symbol` in the `featureData` of the `expressionSet` object.
- `object`: ExpressionSet object for the experiment.
- `groups`: String containing the name of the grouping variable. This should be a the name of a column in the `pData` of the `expressionSet` object.
- `main`: Main title on top of the graph.
- `colvec`: Vector of colors to be used for the groups. If not specified, the default colors of `a4palette` are used.
- `colgroups`: String containing the name of the variable to color the superimposed dots. This should be a the name of a column in the `pData` of the `expressionSet` object.
- `probe2gene`: Boolean indicating whether the probeset should be translated to a gene symbol (used for the default title of the plot).
- `addLegend`: Boolean indicating whether a legend for the colors of the dots should be added.
- `legendPos`: Specify where the legend should be placed. Typically either `topleft`, `bottomright`, `topleft` (the default) or `bottomleft`.
- `...`: Possibility to add extra plot options. See `par`.
combineTwoExpressionSet

Combine two ExpressionSet objects

Description
Merge two ExpressionSet objects, checking their attributes.

Usage
combineTwoExpressionSet(x, y)

Arguments

x  An object of class ExpressionSet
y  An object of class ExpressionSet

Details
exprs and pData are merged. Other data (such as MIAME or annotation) are those of x.

Value
An object of class ExpressionSet
computeLogRatio

Author(s)
Eric Lecoutre

See Also
ExpressionSet

Examples

```r
## Not run:
# prepare and combine two ExpressionSet
data(data.H2009); data(phenoData.H2009)
data(data.SKOV3); data(phenoData.SKOV3)
eH2009 <- prepareExpressionSet(exprs = data.H2009, phenoData = phenoData.H2009, changeColumnsNames = TRUE)
eSKOV3 <- prepareExpressionSet(exprs = data.SKOV3, phenoData = phenoData.SKOV3, changeColumnsNames = TRUE)
newE <- combineTwoExpressionSet(eH2009, eSKOV3)
## End(Not run)
```

computeLogRatio  Summary statistics for gene expression

Description

Compute summary statistics per gene of expression data in a ExpressionSet object.

Usage

```r
computeLogRatio(
  e,
  reference,
  within = NULL,
  across = NULL,
  nReplicatesVar = 3,
  ...
)
```

Arguments

- `e`  
  An object of class ExpressionSet
- `reference`  
  A list with two items: var and level - See details
- `within`  
  Character vector - names of pData columns - See details
- `across`  
  Character vector - names of pData columns - See details
- `nReplicatesVar`  
  Integer - Minimum number of replicates to compute variance
- `...`  
  ...
computeLogRatio

Details

Summary statistics (mean, variances and difference to reference or control) will be computed on the 'exprs' slot of the ExpressionSet object. The parameters of the computation are specified by the parameters 'reference', 'within' and 'across'. The design of the computations is such that the differences and pooled variances are calculated against the sample(s) that was(were) chosen as reference. The reference is specified by the level of a certain variable in the phenoData slot (e.g.: column 'control' and level 'WT' of the phenoData slot or a boolean ('ref') variable with 0 or 1) – the list object of 'var' and 'level' together determine the reference group.

All groups determined by combining the reference$var and across variables will be compared to the reference group. Two different approaches to obtain necessary computations:

- Prepare a boolean variable that reflects only the reference group and specify all groupings in the across arguments. E.g.: reference=list(var = 'boolean', level = 1), across = c('compound', 'dose')

- Add an extra column to the phenoData slot that contains all combinations, with a specific one for the reference group: for example, pData(e)['refvar'] <- paste(pData(e)['compound'], pData(e)['dose'], sep='.') so as to use reference = list(var = 'refvar', level = 'comp1.dose1') as argument for reference.

Sometimes computations need to be conducted within groups, and are thus nested. For example, when comparing treatment values of different cell lines, each will have gene expression values for its own reference. The parameter 'within' allows to define such subgroups, for which computations will be done separately and combined afterwards. Both parameters 'within' and 'across' can be a vector of column names, whose unique combinations will be used for groupings.

Value

Returns an object of class ExpressionSet with pData inherited from the submitted ExpressionSet object, supplemented by the computed statistics in the 'exprs' slot and info thereof in the 'phenoData' slot.

Author(s)

Eric Lecoutre

See Also

plotLogRatio

Examples

if (require(ALL)){
data(ALL, package = "ALL")
ALL <- addGeneInfo(ALL)
ALL$BTtype <- as.factor(substr(ALL$BT,0,1))
ALL2 <- ALL[,ALL$BT != 'T1'] # omit subtype T1 as it only contains one sample
ALL2$BTtype <- as.factor(substr(ALL2$BT,0,1)) # create a vector with only T and B
createExpressionSet

# Test for differential expression between B and T cells
tTestResult <- tTest(ALL, "BTtype", probe2gene = FALSE)
topGenes <- rownames(tTestResult)[1:20]

# plot the log ratios versus subtype B of the top genes
LogRatioALL <- computeLogRatio(ALL2, reference=list(var='BT', level='B'))
a <- plotLogRatio(e=LogRatioALL[topGenes,], openFile=FALSE, tooltipvalues=FALSE, device='pdf', colorsColumnsBy=c('BTtype'), main = 'Top 20 genes most differentially between T- and B-cells', orderBy = list(rows = "hclust"), probe2gene = TRUE)
Details

If changeColumnsNames is TRUE, then the procedure is the following: first one checks if phenoData contains a column named 'colNames'. If so, content will be used to rename exprs columns. On the other case, one uses combinations of phenoData columns to create new names. In any case, old columns names are stored within a column named 'oldcolnames' in the pData.

Value

An object of class ExpressionSet

Author(s)

Eric Lecoutre

See Also

ExpressionSet

Examples

# simulate expression data of 10 features (genes) measured in 4 samples
x <- matrix(rnorm(40), ncol = 4)
colnames(x) <- paste("sample", 1:4, sep = ",")
rownames(x) <- paste("feature", 1:10, sep = ",")
# simulate a phenodata with two variables
ToBePheno <- data.frame(Gender = rep(c('Male', 'Female'), 2),
Treatment = rep(c('Trt', 'Control'), each=2))
rownames(ToBePheno) <- paste("sample", 1:4, sep = ",")
eset <- createExpressionSet(exprs = x, phenoData = ToBePheno)
filterVarInt

Objects from the Class

Objects can be created by calls of the form `new("ExpressionSetWithComputation", assayData, phenoData, featureData, experimentData, annotation, exprs, ...)`.

Extends

- Class `ExpressionSet`, directly.
- Class `eSet`, by class "ExpressionSet", distance 2.
- Class `VersionedBiobase`, by class "ExpressionSet", distance 3.
- Class `Versioned`, by class "ExpressionSet", distance 4.

Methods

No methods defined with class "ExpressionSetWithComputation" in the signature.

Author(s)

Tobias Verbeke

See Also

`ExpressionSet`, `computeLogRatio`

---

**filterVarInt**  
*Filtering on Intensity and Variance*

Description

Function to filter on intensity and variance as typically used in gene expression studies.

Usage

```r
filterVarInt(
  object,
  IntCutOff = log2(100),
  IntPropSamples = 0.25,
  VarCutOff = 0.5
)
```

Arguments

- `object`  
  ExpressionSet object
- `IntCutOff`  
  cut-off value used for the intensity filter
- `IntPropSamples`  
  proportion of samples used by the intensity filter; by default `IntPropSamples` is set to 0.25.
- `VarCutOff`  
  cut-off value used for the variance filter
Details

The intensity filter implies that (by default) the intensity levels must be greater than \(\log_2(100)\) in at least 25 percent of the samples. The variance filter requires that the features have an interquartile range (IQR) greater than 0.5. Note that the IQR is quite insensitive to outliers such that genes with outlying expression values in a few samples are excluded as long as their overall variation is small.

Value

Object of class ExpressionSet containing only the features that pass the variance and intensity filter.

Author(s)

Willem Talloen

References


See Also

pOverA, filterfun

Examples

```r
if (require(ALL)){
  data(ALL, package = "ALL")
  fALL <- filterVarInt(ALL)
  fALL
}
```

heatmap.expressionSet  

**Image plot of an expressionSet**

Description

Grid version of heatmap function adapted to expressionSet objects with some specific requirements such as the possibility to display subgroups, define colors, adapt text graphical parameters (sizes...). The function also suggests a size appropriate for a device to generate a complete plot with all elements.
heatmap.expressionSet

**Usage**

```r
heatmap.expressionSet(
  eset,
  col.groups = pData(phenoData(eset))[, "subGroup"],
  col.orderBy = order(pData(phenoData(eset))[, "subGroup"]),
  col.groups.sep.width = unit(8, "points"),
  col.labels = sampleNames(eset),
  col.labels.sep.width = unit(10, "points"),
  col.labels.gpar = gpar(cex = 1),
  col.labels.max.nchar = 20,
  colors.pergroup = FALSE,
  colors.groups = NULL,
  colors.groups.min = rgb(1, 1, 1),
  colors.max = rgb(1, 0, 0),
  colors.min = rgb(1, 1, 1),
  colors.nbbreaks = 128,
  colors.palette = NULL,
  cell.gpar = gpar(lty = 0),
  row.groups.sep.height = unit(15, "points"),
  row.labels.sep.height = unit(10, "points"),
  row.col.groups.display = ifelse(length(unique(col.groups)) > 1, TRUE, FALSE),
  row.col.groups.display.height = unit(6, "points"),
  row.labels.gpar = gpar(cex = 1, col = "black"),
  row.labels.max.nchar = 45,
  row.labels = list("SYMBOL", "GENENAME"),
  row.labels.sep = " - ",
  row.groups = rep(1, nrow(exprs(eset))),
  row.order = "none",
  row.groups.hclust = FALSE,
  row.groups.hclust.n = 4,
  distfun = dist,
  hclustfun = function(d) { hclust(d, method = "ward") },
  values.min = 0,
  values.max = 16,
  title.gpar = gpar(cex = 1.4),
  title.main = "This is the title possibly being very long -- it will be split on several lines or even",
  title.just = c("right", "top"),
  title.maxlines = 4,
  title.cutpoint = 40,
  subtitle.gpar = gpar(cex = 1),
  subtitle.main = "This is subtitle",
  subtitle.maxlines = 4,
  subtitle.just = title.just,
  subtitle.cutpoint = 40,
  margin.top = unit(2, "lines"),
  margin.left = unit(2, "lines"),
  margin.right = unit(2, "lines"),
```

```
Arguments

**eset** expressionSet object

**col.groups** Vector specifying sub-groups for individual. Sub-groups are treated separately and can thus on plot have different colors.

**col.orderBy** Vector specifying ordering for individual. In case there are sub-groups, individual must first be ordered by sub-groups, but an additional variable gives a way to sort individual within sub-groups.

**col.groups.sep.width** Object of class unit (grid package). Width used to visually separate sub-groups of individuals. This can be unit(0,"points") for example for no separation.

**col.labels** Character vector for columns labels (individuals), by default taken from phenoData.

**col.labels.sep.width** Object of class gpar. Parameters to be used for labels (cex,...).

**col.labels.gpar** Object of class gpar. Parameters to be used for labels (cex,...).

**col.labels.max.nchar** Integer. Number of maximum characters to be used for labels truncation

**colors.perGroup** Boolean. If TRUE, separate colors are used to color image matrix. Colors defined for groups are used.

**colors.groups** Vector. Colors to be used for each group of individual. If NULL (default), colors are taken from column "sampleColor" of expressionSet phenoData.

**colors.groups.min** Character vector of length 1 corresponding to a valid color. If colors.groups are provided, a shading if done between color.group and this color (default: white).

**colors.max** Character vector of length 1 corresponding to a valid color. See colors details.

**colors.min** Character vector of length 1 corresponding to a valid color. See colors details.

**colors.nbreaks** Integer. Number of cutpoints used to split the color palette/shading.

**colors.palette** Character vector of valid color names.

**cell.gpar** Object of class gpar (grid package). Parameters used to format cells, for example to add border (gpar(lty=1)).

**row.groups.sep.height** Object of class unit (grid package). Height between rows sub-groups.
row.labels.sep.height
Object of class unit (grid package). Height between image plot zone and rows labels.

row.col.groups.display
Boolean. Display or not colored band for subgroups of individuals.

row.col.groups.display.height
Object of class unit (grid package). If row.col.groups.display is TRUE then height used for the displayed band.

row.labels.gpar
Object of class gpar (grid package). Parameters to be used for labels (cex,...).

row.labels.max.nchar
Integer. Number of maximum characters to be used for labels truncation.

row.labels
Character vector or list. If vector, direct labels to be used. If list, elements of the list will be taken from featureData and collapsed using row.labels.sep.

row.labels.sep
In case labels are taken from featureData (list for row.labels), separator used to paste the provided columns.

row.groups
Boolean specifying whether rows are split into sub-groups.

row.order
Either a vector of indices to be used to reorder features (rows) or "none" or "hclust" to use clustering.

row.groups.hclust
Boolean. If row.order equals "hclust", one can ask to split features into sub-groups based on a cut of the clustering dendogram.

row.groups.hclust.n
Integer. If row.order equals "hclust" and row.groups.hclust is TRUE, number of sub-groups.

distfun
Function. For row.order equals "hclust", metric function.

hclustfun
Function. For row.order equals "hclust", clustering function.

values.min
Minimum value for the data range. Values that are inferior are assigned to that value. That ensures a maximal cutpoint for the coloring scale.

values.max
Maximum value for the data range. Values that are superior are assigned to that value. That ensures a maximal cutpoint for the coloring scale.

title.gpar
Object of class gpar (grid package). Parameters to be used for the main title (cex,...).

title.main
Character vector. Main title to be displayed.

title.just
Title justification, one of "center","left","right" (first letter of the word can also be used).

title.maxlines
Maximum number of lines for the title split.

title.cutpoint
Integer. Maximum number of characters a line must have. Title is split into lines according to that cutpoint.

subtitle.gpar
Object of class gpar (grid package). Parameters to be used for the subtitle (cex, col,....).

subtitle.main
Character vector. Subtitle. The subtitle will be split into lines following same rules as used for main title.
subtitle.maxlines
  Maximum number of lines for the subtitle split.

subtitle.just
  Subtitle justification, one of "center","left","right" (first letter of the word can also be used).

subtitle.cutpoint
  Integer. Maximum number of characters a line must have. Subtitle is split into lines according to that cutpoint.

margin.top
  Object of class unit (grid package). Top margin.

margin.left
  Object of class unit (grid package). Left margin.

margin.right
  Object of class unit (grid package). Right margin.

margin.bottom
  Object of class unit (grid package). Bottom margin.

legend.display
  Boolean. Display or not the legend. Legend is positionned in upper right corner.

legend.range
  Character: "full" (default) or "data". If full, color scale legend ranges from values.min to values.max. If "data", range is c(min(data),max(data)).

legend.data.display
  Boolean. Display or not color scale legend.

legend.gpar
  Object of class gpar (grid package). Parameters to be used for color scale legend axis (cex,....).

legend.width
  Object of class unit (grid package). Width for the color scale legend.

legend.height
  Object of class unit (grid package). Height for the color scale legend.

... Additional parameters the function may have. Not used currently

Value

The function suggests a size (width, height) for the graphic returned as a vector. A typical usage will be to call the function a first time to get those values and call it again with an output device.

Colors

There are several ways to specify colors used for the image zone. The usual way is to have a shading from colors.groups.min to a color per group (typically the same). By default, a shading is indeed proposed between white (for colors.groups.min) and a same color shared by groups (red for colors.groups.max). The number of possible colors in the shading is determined by colors.nbbreaks. In case one asks for distinct colors for groups, only a single value for colors.groups.min is allowed. By default, subgroups colors are taken from phenoData ("sampleColor" column), consequence of colors.groups being NULL. Colors for groups are overided by providing a vector of valid colors for this colors.groups argument. An additional and flexible way to determine colors is to provide a complete palette of possible colors, as a character vector of valid colors (argument colors.palette). Note that in this case the argument colors.nbbreaks has no effect as the number of possible values is the length of the palette.

Author(s)

Eric Lecoutre <eric.lecoutre@gmail.com>
Examples

```r
# Not run:
library(RColorBrewer)
library(dichromat)

library(Biobase)
library(grid)
pdf.directory=getwd()

load(file.path(getwd(),"expressionSetRma.Rda"))  #expressionSetRma

eset <- expressionSetRma[100:130,pData(phenoData(expressionSetRma))[,"sample"]%in%c(1:10,41:50)]  # ARG
##### !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
exprs(eset)[1,5] <- 13.8
exprs(eset)[10,7] <- 0.5
eset2 <- expressionSetRma[200:250,]  # ARG
eset3 <- expressionSetRma[1000:1009,pData(phenoData(expressionSetRma))[,"sample"]%in%c(1:3,41:46)]  # ARG
eset4 <- expressionSetRma[100:230,pData(phenoData(expressionSetRma))[,"sample"]%in%c(1:20,31:50)]  # ARG
eset5 <- expressionSetRma[1:400,]  # ARG

# eset <- eset2

pdf(file.path(pdf.directory,"eset.pdf"))
size <- heatmap.expressionSet(eset,subtitle.main="")
dev.off()

pdf(file.path(pdf.directory,"eset.pdf"),width=size[1],height=size[2])
heatmap.expressionSet(eset,subtitle.main="")
dev.off()

pdf(file.path(pdf.directory,"eset2.pdf"))
size <- heatmap.expressionSet(
eset2,
colors.nbreaks = 20,
colors.pergroup=TRUE,
legend.range="data",
row.col.groups.display=FALSE,
cell.gpar=gpar(lwd=0.5),
legend.height=unit(50,"points"),
title.just=c("center","center"),
title.maxlines=2,
col.groups.sep.width=unit(0,"points"),
row.labels=featureNames(eset),
subtitle.main="This is subtitle",
row.order="hclust",row.groups.hclust=FALSE,
title.gpar=gpar(cex=2),
subtitle.gpar=gpar(cex=1.5)
)
dev.off()
```
pdf(file.path(pdf.directory,"eset2.pdf"),width=size[1],height=size[2])
size <- heatmap.expressionSet(
eset2,
colors.nbreaks = 20,
colors.pergroup=TRUE,
legend.range="data",
row.col.groups.display=FALSE,
cell.gpar=gpar(lwd=0.5),
legend.height=unit(50,"points"),
title.just=c("center","center"),
title.maxlines=2,
col.groups.sep.width=unit(0,"points"),
row.labels=featureNames(eset),
subtitle.main="This is subtitle",
row.order="hclust",row.groups.hclust=FALSE,
title.gpar=gpar(cex=2),
subtitle.gpar=gpar(cex=1.5)
)
dev.off()

pdf(file.path(pdf.directory,"eset3.pdf"))
size <- heatmap.expressionSet(
eset3,
row.labels.gpar=gpar(cex=0.4,col=c(rep("red",2),rep("black",49)) ), # col will correctly be a vector only if no group
col.labels.gpar=gpar(cex=0.6),
colors.nbreaks = 20,
colors.pergroup=TRUE,
legend.range="data",
row.col.groups.display=FALSE,
cell.gpar=gpar(lwd=0.5),
legend.height=unit(50,"points"),
title.just=c("center","center"),
title.maxlines=2,
col.groups.sep.width=unit(0,"points"),
row.labels=featureNames(eset),
subtitle.main="Essai subtitle",
row.order="hclust",row.groups.hclust=FALSE,
interactive=FALSE)
dev.off()

pdf(file.path(pdf.directory,"eset3.pdf"),width=size[1],height=size[2])
size <- heatmap.expressionSet(
eset3,
row.labels.gpar=gpar(cex=0.4,col=c(rep("red",2),rep("black",49)) ), # col will correctly be a vector only if no group
col.labels.gpar=gpar(cex=0.6),
colors.nbreaks = 20,
colors.pergroup=TRUE,
heatmap.expressionSet

legend.range="data",
row.col.groups.display=FALSE,
cell.gpar=gpar(lwd=0.5),
legend.height=unit(50,"points"),
title.just=c("center","center"),
title.maxlines=2,
col.groups.sep.width=unit(0,"points"),
row.labels=featureNames(eset),
subtitle.main="Essai subtitle",
row.order="hclust",row.groups.hclust=FALSE,
interactive=FALSE
)
dev.off()

pdf(file.path(pdf.directory,"eset4.pdf"))
size <- heatmap.expressionSet(
eset4,
legend.range="data",
colors.palette = dichromat(rich.colors(190)[1:128]),
row.col.groups.display=TRUE,
title.just=c("left","top"),
title.maxlines=2,
row.labels=featureNames(eset),
subtitle.main="",
row.order="hclust",row.groups.hclust=FALSE,
)
dev.off()

pdf(file.path(pdf.directory,"eset4.pdf"),width=size[1],height=size[2])
size <- heatmap.expressionSet(
eset4,
legend.range="data",
colors.palette = dichromat(rich.colors(190)[1:128]),
row.col.groups.display=TRUE,
title.just=c("left","top"),
title.maxlines=2,
row.labels=featureNames(eset),
subtitle.main="",
row.order="hclust",row.groups.hclust=FALSE,
)
dev.off()

pdf(file.path(pdf.directory,"eset5.pdf"))
size <- heatmap.expressionSet(eset5,row.order="hclust",row.groups.hclust=FALSE)
dev.off()

pdf(file.path(pdf.directory,"eset5.pdf"),width=size[1],height=size[2])
heatmap.expressionSet(eset5,row.order="hclust",row.groups.hclust=FALSE)
dev.off()
histPvalue

Plot the Distribution of P Values

Description
This function displays the distribution of the p values using a histogram; the horizontal line represents a uniform distribution based on the p value distribution between 0.5 and 1. This represents the hypothetical p value distribution arising just by chance. This uniform distribution is used to estimate the proportion of differentially expressed genes.

Usage
histPvalue(object, ...)

## S4 method for signature 'limma'
histPvalue(object, ...)

## S4 method for signature 'MArrayLM'
histPvalue(object, coef, ...)

## S4 method for signature 'numeric'
histPvalue(object, ...)

Arguments

object either a numeric vector of p-values, or an object of class tTest, limma or MArrayLM

... further arguments passed to the method

coef index of the coefficient for which the p values should be plotted; only applies to the MArrayLM method

Value
The histogram is displayed on the current device.

Author(s)
Willem Talloen and Tobias Verbeke

References
Examples

```r
if (require(ALL)){
data(ALL, package = "ALL")
ALL <- addGeneInfo(ALL)
ALL$BTtype <- as.factor(substr(ALL$BT,0,1))
tTestResult <- tTest(ALL, "BTtype")
histPvalue(tTestResult[,"p"], addLegend = TRUE)
propDEgenesRes <- propDEgenes(tTestResult[,"p"])
}
```

histpvalueplotter  
Workhorse function for the histPvalue function

Description

Workhorse function for the histPvalue function. This function displays the distribution of the p values using a histogram; the horizontal line represents a uniform distribution based on the p value distribution between 0.5 and 1. This represents the hypothetical p value distribution arising just by chance. This uniform distribution is used to estimate the proportion of differentially expressed genes.

Usage

```r
histpvalueplotter(
pValue,
addLegend = FALSE,
xlab = NULL,
ylab = NULL,
main = NULL,
...
)
```

Arguments

- `pValue` numeric vector of p values
- `addLegend` logical; should a legend be added (TRUE) or not (FALSE; default
- `xlab` label for the x axis; defaults to NULL (no label)
- `ylab` label for the y axis; defaults to NULL (no label)
- `main` main title for the plot; if NULL (default) no main title is displayed
- `...` further arguments for the hist call; currently none are used

Value

no returned value, a plot is drawn to the current device.
lassoReg

Multiple regression using the Lasso algorithm as implemented in the glmnet package

Description

Multiple regression using the Lasso algorithm as implemented in the glmnet package. This is a theoretically nice approach to see which combination of genes predict best a continuous response. Empirical evidence that this actually works with high-dimensional data is however scarce.

Usage

lassoReg(object, covariate)

Arguments

object object containing the expression measurements; currently the only method supported is one for ExpressionSet objects

covariate character string indicating the column containing the continuous covariate.

Value

object of class glmnet

Author(s)

Willem Talloen
References

See Also
lassoClass

Examples
if (require(ALL)){
  data(ALL, package = "ALL")
  ALL <- addGeneInfo(ALL)
  ALL$BTtype <- as.factor(substr(ALL$BT,0,1))
  resultLasso <- lassoReg(object = ALL[1:100,], covariate = "age")
  plot(resultLasso, label = TRUE,
       main = "Lasso coefficients in relation to degree of penalization.")
  featResultLasso <- topTable(resultLasso, n = 15)
}

limmaReg

Wrapper for the limma function for the comparison of two groups (two factor levels)

Description
Wrapper for the limma function for the comparison of two groups (two factor levels)

Usage
limmaReg(object, covariable, probe2gene = TRUE)

Arguments
object object of class ExpressionSet
covariable string indicating the variable defining the continuous covariate
probe2gene logical; if TRUE Affymetrix probeset IDs are translated into gene symbols; if FALSE no such translation is done
Description

Wrapper for the limma function for the comparison of two groups (two factor levels)

Usage

limmaTwoLevels(object, group, probe2gene = TRUE)

Arguments

object object of class ExpressionSet

(group string indicating the variable defining the two groups to be compared

probe2gene logical; if TRUE Affymetrix probeset IDs are translated into gene symbols; if FALSE no such translation is done

Value

S4 object of class ‘limma’ with the following two components:

MArrayLM S4 object of class MArrayLM as returned by the limma function of the limma package

geneSymbols character vector of gene symbols; this slot is only populated if probe2gene=TRUE (and if the ExpressionSet object is appropriately annotated by addGeneInfo for gene symbols to be extracted)

Note

A 'topTable' method is defined for 'limma' objects.

Author(s)

Tobias Verbeke and Willem Talloen
logReg

Logistic regression for predicting the probability to belong to a certain class in binary classification problems

Description

Logistic regression for predicting the probability to belong to a certain class in binary classification problems.

Usage

```r
logReg(
  object,
  groups,
  probesetId = NULL,
  geneSymbol = NULL,
  main = NULL,
  probe2gene = TRUE,
  ...
)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>object</code></td>
<td>ExpressionSet object for the experiment</td>
</tr>
<tr>
<td><code>groups</code></td>
<td>String containing the name of the grouping variable. This should be a name</td>
</tr>
<tr>
<td></td>
<td>of a column in the <code>pData</code> of the <code>expressionSet</code> object.</td>
</tr>
<tr>
<td><code>probesetId</code></td>
<td>The probeset ID. These should be stored in the <code>featureNames</code> of the</td>
</tr>
<tr>
<td></td>
<td><code>expressionSet</code> object.</td>
</tr>
<tr>
<td><code>geneSymbol</code></td>
<td>The gene symbol. These should be stored in the column <code>Gene Symbol</code> in the</td>
</tr>
<tr>
<td></td>
<td><code>featureData</code> of the <code>expressionSet</code> object.</td>
</tr>
<tr>
<td><code>main</code></td>
<td>Main title on top of the graph</td>
</tr>
<tr>
<td><code>probe2gene</code></td>
<td>Boolean indicating whether the probeset should be translated to a gene symbol</td>
</tr>
<tr>
<td></td>
<td>(used for the default title of the plot)</td>
</tr>
<tr>
<td><code>...</code></td>
<td>Possibility to add extra plot options. See <code>plot</code></td>
</tr>
</tbody>
</table>

Details

It will always estimate probability scores to belong to the second level of the factor variable. If a probability score to other level is preferred, then you need to change the order of the levels of the factor.
Value

A data.frame object with three columns and rownames

rownames The 'sampleNames' of the expressionSet object
x The expression values for the specified gene for all samples
y The labels of the samples
fit The fitted probability score to belong to one of the two classes.

Author(s)

Willem Talloen

See Also

ROCcurve, probabilitiesPlot

Examples

```r
## Not run:
if (require(ALL)) {
  data(ALL, package = "ALL")
  ALL <- addGeneInfo(ALL)
  ALL$BTtype <- as.factor(substr(ALL$BT, 0, 1))
  logRegRes <- logReg(geneSymbol = "HLA-DPB1", object = ALL, groups = "BTtype")
  # scoresplot
  probabilitiesPlot(proportions = logRegRes$fit, classVar = logRegRes$y,
                   sampleNames = rownames(logRegRes), main = 'Probability of being a T-cell type ALL')
  # barplot
  probabilitiesPlot(proportions = logRegRes$fit, classVar = logRegRes$y, barPlot = TRUE,
                  sampleNames = rownames(logRegRes), main = 'Probability of being a T-cell type ALL')
}
## End(Not run)
```

nlcvTT

Data to Demonstrate nlcv and Co Functions

Description

Simulated data set used to demonstrate nlcv and accompanying plot functions to study classification problems

Usage

nlcvTT

Format

The object is of class "nlcv", an object as produced by the nlcv function.
Source

data simulated using: nlcvTT <- nlcv(selBcrAblOrNeg, classVar = 'mol.biol', classdist = "unbalanced", nRuns = 10, fsMethod = "t.test", verbose = TRUE)

See Also

nlcv

Examples

## Not run:
data(nlcvTT)
if (require(nlcv)) # on R-Forge
  scoresPlot(nlcvTT, tech = 'svm', nfeat = 25)

## End(Not run)

oaColors

Pick One or More OA Colors

Description

Pick One or More OA Colors

Usage

oaColors(color = NULL, alpha = 1)

Arguments

color  a character vector of color names; possible values are "red", "orange", "yellow", "green", "cyan", "blue", "pink", "limegreen", "purple", "black", "white", "grey" or "gray"

alpha transparency level for the color(s)

Value

character vector of colors

Author(s)

Tobias Verbeke
oaPalette  

Generate a Palette of OA Colors

Description

Generate a Palette of OA Colors

Usage

```
oaPalette(numColors = NULL, alpha = 1)
```

Arguments

- `numColors`: number of colors to be contained in the palette
- `alpha`: transparency level of the colors

Value

vector of colors

Author(s)

Jason Waddell

plot1gene  

Create a Profile Plot for a given Gene

Description

Create a profile plot for a given gene. A profile plot displays the expression values (y-axis) by samples (x-axis), sorted by group. This is a useful working graph as samples can be directly identified. For presentation purposes, a boxPlot can also be considered. with jittered for readability of the plot.

Usage

```
plot1gene(
    probesetId = NULL,
    geneSymbol = NULL,
    object,
    groups,
    main = NULL,
    colvec = NULL,
    colgroups = NULL,
    probe2gene = TRUE,
```

sampleIDs = TRUE,
addLegend = TRUE,
legendPos = "topleft",
cex = 1.5,
...)

Arguments

probesetId The probeset ID. These should be stored in the featureNames of the expressionSet object.
geneSymbol The gene symbol. These should be stored in the column `Gene Symbol` in the featureData of the expressionSet object
object ExpressionSet object for the experiment
groups String containing the name of the grouping variable. This should be a name of a column in the pData of the expressionSet object.
main Main title on top of the graph
colvec Vector of colors to be used for the groups. If not specified, the default colors of a4palette are used.
colgroups String containing the name of the variable to color the superimposed dots. This should be a the name of a column in the pData of the expressionSet object
probe2gene Boolean indicating whether the probeset should be translated to a gene symbol (used for the default title of the plot)
sampleIDs A boolean or a string to determine the labels on the x-axis. Setting it to FALSE results in no labels (interesting when the labels are unreadable due to large sample sizes). Setting it to a string will put the values of that particular pData column as labels. The string should be a name of a column in the pData of the expressionSet object.
addLegend Boolean indicating whether a legend for the colors of the dots should be added.
legendPos Specify where the legend should be placed. Typically either topright, bottomright, topleft (the default) or bottomleft
cex character expansion used for the plot symbols; defaults to 1.5
...

Value

If a geneSymbol is given that has more than one probeSet, the plots for only the first probeSet is displayed. A character vector of corresponding probeset IDs is returned invisibly, so that one can check the profiles of the other related probeset IDs with an extra plot1gene statement If a probesetId is given, one single profile plot for the probeset is displayed.

Author(s)

S. Osselaer, W. Talloen, T. Verbeke
plotComb2Samples

Plots the correlation in gene expression between two samples

Description

Plots the correlation in gene expression between two samples. Each dot represents a gene, and the dots have a density-dependent coloring. Genes with exceptional behavior can be highlighted by showing their gene symbol.

Usage

plotComb2Samples(
    object,
    x,
    y,
    trsholdX = NULL,
    trsholdY = NULL,
    probe2gene = TRUE,
...
)

Arguments

object         ExpressionSet object for the experiment
x              String containing the name of the first sample. This should be a name of a column in the exprs data of the expressionSet object.
y              String containing the name of the second sample. See x
trsholdX       Vector of two values specifying the X-axis thresholds within which genes should be highlighted by their gene symbol.
plotCombination2genes

Plot a Combination of Two Genes

Description
Plot a Combination of Two Genes

Usage
plotCombination2genes(
  probesetId1 = NULL,
  probesetId2 = NULL,
  geneSymbol1 = NULL,
  geneSymbol2 = NULL,
  object,
  groups,
  addLegend = TRUE,
  legendPos = "topleft",
  probe2gene = TRUE,
  colvec = NULL,
  ...}

Value
No returned value, a plot is drawn to the current device.

Author(s)
W. Talloen

See Also
plotCombMultSamples

Examples
if (require(ALL)){
data(ALL, package = "ALL")
ALL <- addGeneInfo(ALL)
plotComb2Samples(ALL,"84004", "01003",
  trsholdX = c(10,12), trsholdY = c(4,6),
  xlab = "a B-cell", ylab = "a T-cell")
}

plotCombination2genes  Plot a Combination of Two Genes
plotCombination2genes

Arguments

probesetId1 First probeset id, plotted in the x-axis
probesetId2 Second probeset id, plotted in the y-axis
geneSymbol1 First gene symbol, plotted in the x-axis
geneSymbol2 Second gene symbol, plotted in the y-axis
object ExpressionSet object for the experiment
groups string containing the name of the grouping variable
addLegend Logical value to indicate whether a legend needs to be draw
legendPos Position on the graph where to put the legend
probe2gene should the probeset be translated to a gene symbol (used for the default title of the plot
colvec a character vector of colors. If not specified it will be automatically generated by a4palette
...

Value

If a gene id is given, the plots for only the first probeset is displayed and a character vector of corresponding probeset IDs is returned invisibly. It is a list containing

probeset1 Probeset ids measuring 'gene1'
probeset1 Probeset ids measuring 'gene1'

If a probeset id is given, one single profile plot for the probeset is displayed.

Author(s)

W. Talloen, T. Verbeke

See Also

plot1gene

Examples

if (require(ALL)){
  data(ALL, package = "ALL")
  ALL <- addGeneInfo(ALL)
  aa <- plotCombination2genes(geneSymbol1 = 'HLA-DPB1', geneSymbol2 = 'CD3D',
      object = ALL, groups = "BT",
      addLegend = TRUE, legendPos = 'topright')
  aa
}
plotCombMultSamples

Plots the correlation in gene expression between more than 2 samples

Description

Plots the correlation in gene expression between more than 2 samples

Usage

plotCombMultSamples(exprsMatrix, ...)

Arguments

exprsMatrix ExpressionSet object to plot. For larger datasets, this will typically be a subset of the data.
...
Further arguments, e.g. to add extra plot options. See pairs

Value

no returned value, a plots is drawn in the current device

Author(s)

Willem Talloen

See Also

plotComb2Samples

Examples

if (require(ALL)){
data(ALL, package = "ALL")
ALL <- addGeneInfo(ALL)
plotCombMultSamples(exprs(ALL)[,c("84004", "11002", "01003")])
}

plotLogRatio

- **Plot a summary gene expression graph**

**Description**
Plot ratios of expression values observed in a treatment versus those of a reference. First the ratios and variances are computed on the gene expression data.

**Usage**

```r
plotLogRatio(
  e,  
  reference,  
  within = NULL,  
  across = NULL,  
  nReplicatesVar = 3,  
  filename = "Rplots",  
  device = "svg",  
  orderBy = list(rows = "hclust", cols = NULL),  
  colorsColumns = NULL,  
  colorsColumnsBy = NULL,  
  colorsColumnsByPalette = c("#1B9E77", "#D95F02", "#7570B3", "#E7298A", "#66A61E",  
                           "#E6AB02", "#A6761D", "#666666"),  
  colorsUseMeanQuantiles = FALSE,  
  colorsMeanQuantilesPalette = c("orange", "red", "darkred"),  
  colorsBarsMatrix = NULL,  
  colorsGenesNames = c("black"),  
  main = paste("log2 ratio's"),  
  shortvarnames = NULL,  
  longvarnames = NULL,  
  gene.length = 50,  
  gene.fontsize = 6,  
  main.fontsize = 9,  
  columnhead.fontsize = 8,  
  mx = 1.5,  
  exp.width = 1.8,  
  exp.height = 0.2,  
  log2l.show = TRUE,  
  log4l.show = FALSE,  
  quantiles.show = FALSE,  
  quantiles.compute = c(0.9),  
  error.show = TRUE,  
  view.psid = FALSE,  
  errorLabel = "Error bars show the pooled standard deviation",  
  closeX11 = FALSE,  
  openFile = FALSE,  
  tooltipvalues = FALSE,
```
probes2gene = TRUE,
...
)

Arguments

- **e**: ExpressionSet object to use
- **reference**: A list with two items: var and level - See details
- **within**: Character vector - names of pData columns - See details
- **across**: Character vector - names of pData columns - See details
- **nReplicatesVar**: Integer - Minimum number of replicates to compute variance
- **filename**: Name of the filename to use. No need to specify extension which will be added according to device.
- **device**: One of 'pdf', 'X11', 'png', 'svg'. For svg device, one X11 device is also opened.
- **orderBy**: See details
- **colorsColumns**: A vector of colors to be used for plotting columns; default value is NULL which ends up with red – see Colors section
- **colorsColumnsBy**: A vector of pData columns which combinations specify different colors to be used – see Colors section
- **colorsColumnsByPalette**: If colorsColumns is NULL, vector of colors to be used for coloring columns potentially splitted by colorsColumnsBy
- **colorsUseMeanQuantiles**: Boolean to indicate if the quantile groups computed on averages over all treatments should be used for coloring – see Colors section
- **colorsMeanQuantilesPalette**: if colorsUseMeanQuantiles is TRUE, these colors will be used for the different groups – see Colors section
- **colorsBarsMatrix**: Matrix of colors to be used for each individual bar; colors are provided for genes in data order and thus are possibly reordered according to orderBy – see Colors section
- **colorsGenesNames**: Vector of colors to be used for gene names; will be recycled if necessary; colors are provided for genes in data order and thus are possibly reordered according to orderBy
- **main**: Main title
- **shortvarnames**: vector or pData column to be used to display in graph columns. If NULL, those names will be used from the coded names added to pData during computations (list of columns values pasted with a dot). Warning: shortvarnames must be defined in the order columns are present in the ExpressionSet object so that they will be reordered if one asks to order columns.
- **longvarnames**: pData column to be used in SVG tooltip title. If NULL, shortvarnames will be used. Same warning than shortvarnames about ordering
gene.length  Maximum number of characters that will be printed of the gene names

gene.fontsize  Font size for the gene names, default =

main.fontsize  Font size for the main, default = 9

columnhead.fontsize  Font size for the column headers, default = 8

mx  Expansion factor for the width of the bars that represent the expression ratios

exp.width  Expansion factor for global graph width, and the space between the plotted column

exp.height  Expansion factor for global graph height, and the space between the plotted row

log2l.show  A logical value. If 'TRUE', the line for log2 values on each column (when max(data) > 2) is drawn

log4l.show  A logical value. If 'TRUE', the line for log4 values on each column (when max(data) > 4) is drawn

quantiles.show  A logical value. If 'TRUE', a line is drawn for quantiles computed separately on each column

quantiles.compute  A logical value. If 'TRUE', the vector quantiles will be computed and displayed provided that quantile.show is TRUE

error.show  A logical value. If 'TRUE', errors bars are displayed on the graph (only for those columns for which they are available

view.psid  A logical value. If 'TRUE', the genes psid is displayed on the gene name

errorLabel  A character vector describing the error bars, printed at the bottom of the figure

closeX11  If device is SVG, do we close the required X11 device at the end?

openFile  A logical value. If 'TRUE', the produced output file is opened

tooltipvalues  If device is SVG, one can choose to display each bar separately, with data values as tooltips. Note however that each bar will be considered as a distinct object instead of a column, which will takes much more time to create the graph and produces a much bigger SVG file

probe2gene  Boolean indicating whether the probeset should be translated to a gene symbol (used for the default title of the plot

... ...

Value

The ExpressionSet object with the computated variables is returned.

Ordering

orderBy: A list with two components, rows and cols, each one possibly being NULL (no ordering on the specific dimension). Ordering on cols can be done according to (a) pData column(s) (for example: c('cellline', 'compound', 'dose'). Ordering on rows can be done using of the following values:

- NULL no reordering on rows
• numeric vector use the vector values to sort rows
• alphause genes names alphabetice order
• effecttry to assess global gene expression level by taking sum(abs(values)) on specified exprs columns
• hclustuse the ordering returned by hclust invoked on specified exprs columns

**Colors**

The management of colors is very flexible but is a little bit tricky, as a variety of parameters are available to the user. Basically, combinations of arguments allow to set colors for columns headers (text), columns as a whole (different colors for the different columns) or for each of the individual horizontal bars. By default, everything is red. There are four main different arguments that can be used and that are applied in a consecutive order. Each one may override a previous argument value. Below is a list of arguments and their consecutive actions:

• colorsColumns The first way to assign colors is to provide a vector of colors that will be used for each column (headers and its horizontal bars). This vector is recycled so that providing one unique value will color all columns, whereas providing a vector of length 2 will alternate columns colors.
• colorsColumnsByTo be used when the experiment involves groupings for pData, for example dose, cellline or treatment. In order to see the effects of such variables, one can color columns using combinations of those. The argument is a vector of pData columns such as c('cellline', 'dose'). Unique combinations will be computed and a color will be assigned for each group of columns. The vector that is provided with the argument colorsColumnsByPalette is used to assign colors. If the argument colorsColumnsBy is not NULL then it overrides the previous argument colorsColumns.
• colorsUseMeanQuantiles A logical value. The default plotGeneDE displays for each gene the expression value difference between treatment and reference, but does not reveal any information about the expression levels in these conditions. Parameter colorsUseMeanQuantiles allows to color the horizontal bars according to expression level that is derived from quantiles computed on averages of the complete ExpressionSet object. As it involves the expression data of all probesets, computations must be done before subsetting the ExpressionSet object and the plotGeneDEting. The function addQuantilesColors computes quantiles and corresponding mean expression level intervals. If colorsUseMeanQuantiles 'TRUE', previous coloring parameters are overriden. The parameter colorsMeanQuantilesPalette is used to assign colors for average-quantiles-groups. Note that columns headers are still given by previous arguments.
• colorsBarsMatrix The most flexible way to assign colors as the matrix will be used to color each bar of the plot individually. A check is done to ensure that the number of rows and columns are not less than the number of probesets and columns. If not NULL, this parameter overrides the previous ones.

**Author(s)**

Hinrich Goehlmann and Eric Lecoutre

**See Also**

computeLogRatio, addQuantilesColors
Examples

```r
if (require(ALL)) {
  data(ALL, package = "ALL")
  ALL <- addGeneInfo(ALL)
  ALL$BTtype <- as.factor(substr(ALL$BT, 0, 1))
  ALL2 <- ALL[, ALL$BT != 'T1'] # omit subtype T1 as it only contains one sample
  ALL2$BTtype <- as.factor(substr(ALL2$BT, 0, 1)) # create a vector with only T and B

  # Test for differential expression between B and T cells
  tTestResult <- tTest(ALL, "BTtype", probe2gene = FALSE)
  topGenes <- rownames(tTestResult)[1:20]

  # plot the log ratios versus subtype B of the top genes
  LogRatioALL <- computeLogRatio(ALL2, reference=list(var='BT', level='B'))
  a <- plotLogRatio(e=LogRatioALL[topGenes,], openFile=FALSE, tooltipvalues=FALSE, device='pdf',
                   colorsColumnsBy=c('BTtype'), main = 'Top 20 genes most differentially between T- and B-cells',
                   orderBy = list(rows = "hclust"), probe2gene = TRUE)
  ## Not run:
  a <- plotLogRatio(e=LogRatioALL[topGenes,], openFile=TRUE, tooltipvalues=FALSE, device='pdf',
                   colorsColumnsBy=c('BTtype'), main = 'Top 20 genes most differentially between T- and B-cells',
                   orderBy = list(rows = "hclust", cols = "sex"), probe2gene = TRUE)
  ## End(Not run)
}
```

probabilitiesPlot

Function to plot the probabilities to belong to a certain class in binary classification problems.

Description

Function to plot the probabilities to belong to a certain class in binary classification problems. These probabilities are often calculated using a logistic regression model. The class membership of the samples is displayed using a colored strip (with legend below the plot).

Usage

```r
probabilitiesPlot(
  proportions,  # required
  classVar,     # required
  sampleNames,  # required
  plot = TRUE,
  barPlot = FALSE,
  layout = TRUE,
  main = NULL,
  sub = NULL,
  ...)
```
Arguments

proportions A vector containing the calculated probabilities to belong to a certain class in binary classification problems. These probabilities are often calculated using a logistic regression model.

classVar A vector containing the class where the sample belongs to.

sampleNames A vector with the names of the sample.

plot logical. If FALSE, nothing is plotted.

barPlot Should a barplot be drawn (TRUE) or a scatterplot like MCREstimate-type scores plot (the default, FALSE).

layout boolean indicating whether mcrPlot should prespecify a layout for a single plot (default, TRUE) or whether the user takes care of the layout (FALSE).

main Main title for the scores plot; if not supplied, 'Scores Plot' is used as a default.

sub Subtitle for the scores plot; if not supplied, the classification technique and the chosen number of features are displayed.

... Additional graphical parameters to pass to the plot function.

Value

No returned value, a plot is drawn in the current device.

Author(s)

Willem Talloen and Tobias Verbeke

See Also

logReg

Examples

## Not run:
if (require(ALL)){
data(ALL, package = "ALL")
  ALL <- addGeneInfo(ALL)
  ALL$BTtype <- as.factor(substr(ALL$BT,0,1))
  logRegRes <- logReg(geneSymbol = "HLA-DRB1", object = ALL, groups = "BTtype")
  # scoresplot
  probabilitiesPlot(proportions = logRegRes$fit, classVar = logRegRes$y,
                   sampleNames = rownames(logRegRes), main = 'Probability of being a T-cell type ALL')
  # barplot
  probabilitiesPlot(proportions = logRegRes$fit, classVar = logRegRes$y, barPlot=TRUE,
                   sampleNames = rownames(logRegRes), main = 'Probability of being a T-cell type ALL')
}

## End(Not run)
probe2gene

Translate Affymetrix probeset IDs into gene symbols

Description

Auxiliary function for (currently) spectralMap allowing the conversion of Affy probeset IDs to gene symbols

Usage

probe2gene(probesetIds, chipPkg)

Arguments

- probesetIds: Affymetrix probeset IDs
- chipPkg: string indicating the annotation package for the chip

Value

Vector containing the respective gene symbols

Author(s)

Tobias Verbeke

See Also

spectralMap, lassoClass, ...

Examples

```r
if (require(ALL)){
  data(ALL, package = "ALL")
  chip <- annotation(ALL)
  chipAnnotationPkg <- paste(chip, "db", sep = ".")
  res <- probe2gene(featureNames(ALL), chipAnnotationPkg)
  head(res)
}
```
profilesPlot

Plot expression profiles of multiple genes or probesets. Each line depicts a gene, and the color legend can be used to identify the gene.

Description

Plot expression profiles of multiple genes or probesets. Each line depicts a gene, and the color legend can be used to identify the gene.

Usage

profilesPlot(
  object, 
  probesetIds, 
  sampleIDs = TRUE, 
  addLegend = TRUE, 
  legendPos = "topleft", 
  colvec = NULL, 
  orderGroups = NULL, 
  ...
)

Arguments

- **object**: ExpressionSet object for the experiment
- **probesetIds**: The probeset ID. These should be stored in the featureNames of the expressionSet object.
- **sampleIDs**: A boolean or a string to determine the labels on the x-axis. Setting it to FALSE results in no labels (interesting when the labels are unreadable due to large sample sizes). Setting it to a string will put the values of that particular pData column as labels. The string should be a name of a column in the pData of the expressionSet object.
- **addLegend**: Boolean indicating whether a legend for the colors of the dots should be added.
- **legendPos**: Specify where the legend should be placed. Typically either topright, bottomright, topleft (the default) or bottomleft.
- **colvec**: Vector of colors to be used for the groups. If not specified, the default colors of a4palette are used.
- **orderGroups**: String containing the name of the grouping variable to order the samples in the x-axis accordingly. This should be a name of a column in the pData of the expressionSet object.
- **...**: Possibility to add extra plot options. See par.

Value

No returned value, a plot is drawn in the current device.
propDEgenes

Generic function to compute the proportion of differentially expressed genes that are present

Description

Generic function to compute the proportion of differentially expressed genes that are present

Usage

propDEgenes(object, ...)

Arguments

- **object**: object of class propDEgene
- **...**: further arguments for the method (currently none implemented)

Value

numeric of length one giving the proportion of differentially expressed genes

Author(s)

Willem Talloen and Tobias Verbeke
propDEgenes-methods

Generic function to compute the proportion of differentially expressed genes that are present

Description

Generic function to compute the proportion of differentially expressed genes that are present

Usage

## S4 method for signature 'limma'
propDEgenes(object, ...)

## S4 method for signature 'numeric'
propDEgenes(object, ...)

Arguments

object object of class propDEgene

... further arguments for the method (currently none implemented)

Value

numeric of length one giving the proportion of differentially expressed genes

Methods

limma

propDEgenes method for a limma object numeric

object = "limma" propDEgenes method for a numeric vector, i.e. a vector of P Values

Author(s)

Willem Talloen and Tobias Verbeke
propdegenescalculation

*Estimation of proportion of differentially expressed genes*

**Description**

Estimation of proportion of differentially expressed genes. This estimation is based on a histogram of the p-values. More specifically, based on the horizontal line representing a uniform distribution based on the p value distribution between 0.5 and 1. This represents the hypothetical p value distribution arising just by chance. All genes with small p-values above this line reflect the expected number of differentially expressed genes not by chance.

**Usage**

`propdegenescalculation(pValue)`

**Arguments**

- `pValue`: a vector of p-values

**Value**

proportion of differential genes

**Author(s)**

Willem Talloen and Tobias Verbeke

**See Also**

`histPvalue`

**Examples**

```r
if (require(ALL)){
data(ALL, package = "ALL")
ALL <- addGeneInfo(ALL)
ALL$BTtype <- as.factor(substr(ALL$BT,0,1))

tTestResult <- tTest(ALL, "BTtype")
histPvalue(tTestResult[,"p"], addLegend = TRUE)
propDEgenesRes <- propDEgenes(tTestResult[,"p"])
}
```
**replicates**  
*computes replicates across a vector*

**Description**  
Given a vector, returns the replicates in order

**Usage**  
`replicates(x)`

**Arguments**  
`x`  
character or numeric vector

**Value**  
numeric vector

**Author(s)**  
Henrique Dallazuanna

**References**  
R-help mailing list

**See Also**  
rle

**Examples**  
```r  
x <- c('a', 'b', 'a', 'a', 'b', 'a', 'c', 'c', 'c')  
data.frame(val=x, rep=replicates(x))```
spectralMap

Generic function to draw a spectral map, according to JnJ Standards

Description

Generic function to draw a spectral map, according to JnJ Standards

Usage

spectralMap(object, groups, ...)

Arguments

object 
object of class ExpressionSet

groups 
string indicating the name of the column in the phenoData that defines the groups

... 
further arguments to be passed to the methods

Value

Object of class plot.mpm, i.e. the S3 output object of the plot.mpm function of the mpm package

Note

Coloring of groups on the spectralMap uses the a4 palette as produced by a4palette

Author(s)

Tobias Verbeke

References


See Also

plot.mpm
Examples

```r
if (require(ALL)){
  data(ALL, package = "ALL")
  ALL <- addGeneInfo(ALL)
  spectralMap(object = ALL, groups = "BT", legendPos = 'bottomright')
  spectralMap(object = ALL, groups = "BT",
    plot.mpm.args = list(label.tol = 10, rot = c(-1, 1), sub = "", lab.size = 0.65,
    dim = c(1,2), sampleNames = FALSE, zoom = c(1,5), col.size = 2,
    do.smoothScatter = TRUE))
  spectralMap(object = ALL, groups = "BT",
    plot.mpm.args = list(label.tol = 10, rot = c(-1, 1), sub = "", lab.size = 0.65,
    dim = c(1,2), sampleNames = as.character(pData(ALL)$BT),
    zoom = c(1,5), col.size = 2, do.smoothScatter = TRUE))
}
```

Description

Methods for spectralMap

Usage

```r
## S4 method for signature 'ExpressionSet,character'
spectralMap(
  object,
  groups,
  makeLognormal = TRUE,
  mpm.args = list(row.weight = "mean", col.weight = "constant", logtrans = TRUE),
  plot.mpm.args = list(zoom = c(1, 2), label.tol = 10, rot = c(-1, 1), sub = "",
    lab.size = 0.85, col.group = pData(object)[[, groups], colors = c("wheat", "darkgrey",
    a4palette(nlevels(pData(object)[[, groups])), col.size = 2, do.smoothScatter = TRUE),
    probe2gene = TRUE,
    addLegend = TRUE,
    legendPos = "topleft",
    ...
  )
)
```

Arguments

- **object**: object of class ExpressionSet
- **groups**: string indicating the name of the column in the phenoData that defines the groups
- **makeLognormal**: boolean indicating whether one wants to exponentiate the data to make them lognormally shaped (TRUE; the default) or not (FALSE)
- **mpm.args**: list of arguments that can be passed to the mpm function
Methods for `topTable`

Methods for `topTable`. `topTable` extracts the top n most important features for a given classification or regression procedure.

Usage

```r
## S4 method for signature 'limma'

## S4 method for signature 'MArrayLM'
```
\texttt{eb = fit[c("t", "p.value", "lods")],}
\texttt{adjust.method = "BH",}
\texttt{sort.by = "B",}
\texttt{resort.by = NULL,}
\texttt{p.value = 1,}
\texttt{lfc = 0}
\texttt{)}

## S4 method for signature 'tTest'
\texttt{topTable(fit, n)}

## S4 method for signature 'fTest'
\texttt{topTable(fit, n)}

### Arguments

\texttt{fit} \hspace{1cm} \text{object resulting from a classification or regression procedure}

\texttt{n} \hspace{1cm} \text{number of features that one wants to extract from a table that ranks all features according to their importance in the classification or regression model; defaults to 10 for limma objects}

\texttt{coef} \hspace{1cm} \text{column number or column name specifying which coefficient or contrast of the linear model is of interest. For topTable, can also be a vector of column subscripts, in which case the gene ranking is by F-statistic for that set of contrasts.}

\texttt{genelist} \hspace{1cm} \text{data frame or character vector containing gene information. For topTable only, this defaults to \texttt{fit$genes}.}

\texttt{eb} \hspace{1cm} \text{subset of \texttt{fit} containing Empirical Bayesian estimates, so columns: 't', 'p-value' and 'lods' by default. For expert use only.}

\texttt{adjust.method} \hspace{1cm} \text{method used to adjust the p-values for multiple testing. Options, in increasing conservatism, include "none", "BH", "BY" and "holm". See \texttt{p.adjust} for the complete list of options. A NULL value will result in the default adjustment method, which is "BH".}

\texttt{sort.by} \hspace{1cm} \text{character string specifying which statistic to rank the genes by. Possible values for topTable are "logFC", "AveExpr", "t", "P", "p", "B" or "none". (Permitted synonyms are "M" for "logFC", "A" or "Amean" for "AveExpr", "T" for "t" and "p" for "P".) Possible values for topTableF are "F" or "none". topTreat accepts the same values as topTable except for "B".}

\texttt{resort.by} \hspace{1cm} \text{character string specifying statistic to sort the selected genes by in the output data.frame. Possibilities are the same as for sort.by.}

\texttt{p.value} \hspace{1cm} \text{cutoff value for adjusted p-values. Only genes with lower p-values are listed.}

\texttt{lfc} \hspace{1cm} \text{minimum absolute log2-fold-change required. topTable and topTableF include only genes with (at least one) absolute log-fold-change greater than lfc. topTreat does not remove genes but ranks genes by evidence that their log-fold-change exceeds lfc.}
Methods

glmnet

glmnet objects are produced by lassoClass or lassoReg

fit = "glmnet", n = "numeric"  limma objects are produced by limma2Groups

MarrayLM

fit = "limma", n = "numeric"  MarrayLM objects are produced by lmFit of the limma package

pamClass

fit = "pamClass", n = "numeric"  pamClass objects are produced by pamClass

rfClass

fit = "rfClass", n = "numeric"  rfClass objects are produced by rfClass

tTest

fit = "tTest", n = "numeric"  tTest objects are produced by tTest

fit = "fTest", n = "numeric"  fTest objects are produced by fTest

See Also

• topTable-methods for: glmnet, lognet and elnet

• topTable,pamClass-method

• topTable,rfClass-method

---

tTest  

Use t Test to Compare Two Groups

---

Description

Use a (modified) t test to compare two groups

Usage

tTest(object, groups, probe2gene = TRUE)

Arguments

object  
ExpressionSet object

groups  
string indicating the name of the variable of the phenoData containing the group information

probe2gene  
logical; if TRUE Affymetrix probeset IDs are translated into gene symbols; if FALSE no such translation is conducted

Details

For multiple testing the mt.raw2adjp function of package multtest is used.
Volcano Plot

Value

Object of class "tTest", a data frame with the following columns

- **gSymbol**: Gene Symbol
- **p**: p-value of the difference between the groups
- **logRatio**: Log ratio of the expression between the groups
- **pBH**: p-value of the difference between the groups, with Benjamini-Hochberg multiplicity correction
- **tStat**: Student t-statistic of the difference between groups

Author(s)

Willem Talloen, Tobias Verbeke

See Also

- `rowttests` in `rowFtests`

Examples

```r
if (require(ALL)){
  data(ALL, package = "ALL")
  ALL <- addGeneInfo(ALL)
  ALL$BTtype <- as.factor(substr(ALL$BT,0,1))
  tTestRes <- tTest(object = ALL, groups = "BTtype", probe2gene = TRUE)
  volcanoPlot(tTestRes)
}
```

---

**volcanoPlot**  
*Draw a Volcano Plot*

Description

Generic function to draw a volcano plot. A volcano plot is a graph that allows to simultaneously assess the P values (statistical significance) and log ratios (biological difference) of differential expression for the given genes.

Usage

```r
volcanoPlot(x, y, pointLabels, ...)
```
Arguments

x either an object of class 'tTest', of class 'limma' or a numeric vector of log ratios, i.e. the log of the fold change values; the names of the logRatio vector will be used to display the names of the most interesting gene

y should not be given if an object of class 'tTest' or 'limma' is passed as argument 'x'; if 'x' is a numeric vector of log ratios, 'y' should be given and should be a numeric vector of P-values indicating the statistical significance

pointLabels Labels for points on the volcano plot that are interesting taking into account both the x and y dimensions; typically this is a vector of gene symbols; most methods can access the gene symbols directly from the object passed as 'x' argument; the argument allows for custom labels if needed

... further arguments to specific methods

Value

The volcano plot is drawn to the current device.

Author(s)

Tobias Verbeke, based on code by Willem Talloen

References


See Also

See volcanoplotter

Examples

```r
if (require(ALL)){
  data(ALL, package = "ALL")
  ALL <- addGeneInfo(ALL)
  ALL$BTtype <- as.factor(substr(ALL$BT,0,1))
  tTestRes <- tTest(object = ALL, groups = "BTtype", probe2gene = TRUE)
  volcanoPlot(tTestRes)
}
```
Description

This function draws a volcano plot, a graph that allows to simultaneously assess the statistical and biological significance of differential expression for the given genes.

Usage

```r
## S4 method for signature 'tTest,missing,missing'
volcanoPlot(x, y, pointLabels, topPValues = 10, topLogRatios = 10, smoothScatter = TRUE, xlab = NULL, ylab = NULL, main = NULL, sub = NULL, newpage = TRUE, additionalPointsToLabel = NULL, additionalLabelColor = "red")

## S4 method for signature 'tTest,missing,character'
volcanoPlot(x, y, pointLabels, topPValues = 10, topLogRatios = 10, smoothScatter = TRUE, xlab = NULL, ylab = NULL, main = NULL, sub = NULL, newpage = TRUE, additionalPointsToLabel = NULL, additionalLabelColor = "red")

## S4 method for signature 'limma,missing,missing'
volcanoPlot(x,)
```
y, pointLabels, topPValues = 10, topLogRatios = 10, smoothScatter = TRUE, xlab = NULL, ylab = NULL, main = NULL, sub = NULL, newpage = TRUE, additionalPointsToLabel = NULL, additionalLabelColor = "red"
}

## S4 method for signature 'limma,missing,character'
volcanoPlot(
  x, y, pointLabels, topPValues = 10, topLogRatios = 10, smoothScatter = TRUE, xlab = NULL, ylab = NULL, main = NULL, sub = NULL, newpage = TRUE, additionalPointsToLabel = NULL, additionalLabelColor = "red"
)

## S4 method for signature 'numeric,numeric,character'
volcanoPlot(
  x, y, pointLabels, topPValues = 10, topLogRatios = 10, smoothScatter = TRUE, xlab = NULL, ylab = NULL, main = NULL, sub = NULL, newpage = TRUE, additionalPointsToLabel = NULL, additionalLabelColor = "red"
)
## S4 method for signature 'numeric,numeric,missing'

```r
def volcanoPlot(
    x,
    y,
    pointLabels,
    topPValues = 10,
    topLogRatios = 10,
    smoothScatter = TRUE,
    xlab = NULL,
    ylab = NULL,
    main = NULL,
    sub = NULL,
    newpage = TRUE,
    additionalPointsToLabel = NULL,
    additionalLabelColor = "red"
)
```

**Arguments**

- `x` either an object of class 'tTest', of class 'limma' or a numeric vector of log ratios, i.e. the log of the fold change values; the names of the logRatio vector will be used to display the names of the most interesting gene.

- `y` should not be given if an object of class 'tTest' or 'limma' is passed as argument 'x'; if 'x' is a numeric vector of log ratios, 'y' should be given and should be a numeric vector of P-values indicating the statistical significance.

- `pointLabels` Labels for points on the volcano plot that are interesting taking into account both the x and y dimensions; typically this is a vector of gene symbols; most methods can access the gene symbols directly from the object passed as 'x' argument; the argument allows for custom labels if needed.

- `topPValues` top n points that will be included in the points to label based on their low P Values.

- `topLogRatios` top n points that will be included in the points to label based on their high absolute values of the log ratio.

- `smoothScatter` use color saturation to indicate dots that are in densely populated regions of the graph; defaults to TRUE.

- `xlab` label for the x axis (string).

- `ylab` label for the y axis (string).

- `main` main title for the graph (string).

- `sub` subtitle for the graph (string).

- `newpage` should the graph be drawn to a new grid page? Defaults to TRUE. This argument is useful for including several volcano plots in one layout.

- `additionalPointsToLabel` Entrez IDs of genes of interest, that will be highlighted on the plot; the color of highlighting is determined by the 'additionalLabelColor' argument.

- `additionalLabelColor` Color used to highlight the 'additionalPointsToLabel'; defaults to "red".
Details

The set of genes for which labels are displayed is the union of the set of genes that have lowest P-values (topPValues) and the set of genes that display the highest absolute values for the log ratios (topLogRatios).

Value

The volcano plot is drawn to the current device.

Author(s)

Tobias Verbeke, based on code by Willem Talloen

Description

Workhorse function for the different volcanoPlot methods. A volcano plot is a graph that allows to simultaneously assess the P values (statistical significance) and log ratios (biological difference) of differential expression for the given genes.

Usage

volcanoplotter(
  logRatio,
  pValue,
  pointLabels,
  topPValues = 10,
  topLogRatios = 10,
  logTransformP = TRUE,
  smoothScatter = TRUE,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  sub = NULL,
  newpage = TRUE,
  additionalPointsToLabel = NULL,
  additionalLabelColor = "red"
)

Arguments

- **logRatio**: numeric vector of log ratios
- **pValue**: numeric vector of P values
pointLabels: Labels for points on the volcano plot that are interesting taking into account both the x and y dimensions; typically this is a vector of gene symbols; most methods can access the gene symbols directly from the object passed as 'x' argument; the argument allows for custom labels if needed.

topPValues: top n points that will be included in the points to label based on their low P Values.

topLogRatios: top n points that will be included in the points to label based on their high absolute values of the log ratio.

logTransformP: if TRUE (default) -log10(pValue) is used for the plot instead of the raw P values.

smoothScatter: use color saturation to indicate dots that are in densely populated regions of the graph; defaults to TRUE.

xlab: label for the x axis (string).

ylab: label for the y axis (string).

main: main title for the graph (string).

sub: subtitle for the graph (string).

newpage: should the graph be drawn to a new grid page? Defaults to TRUE. This argument is useful for including several volcano plots in one layout.

additionalPointsToLabel: Entrez IDs of genes of interest, that will be highlighted on the plot; the color of highlighting is determined by the 'additionalLabelColor' argument.

additionalLabelColor: Color used to highlight the 'additionalPointsToLabel'; defaults to "red".

Value:
a volcanoplot is drawn to the current device.

Author(s):
Tobias Verbeke.
Index

* classes
  ExpressionSetWithComputation-class, 10
* datasets
  nlcvTT, 26
* data
  combineTwoExpressionSet, 6
  computeLogRatio, 7
  createExpressionSet, 9
* dplot
  a4palette, 3
  computeLogRatio, 7
  histPvalue, 20
  plot1gene, 28
  volcanoPlot, 51
  volcanoPlot-methods, 53
  volcanoplotter, 56
* hplot
  spectralMap, 46
  spectralMap-methods, 47
* htest
  propDEgenes, 42
  propDEgenes-methods, 43
  tTest, 50
* manip
  computeLogRatio, 7
  filterVarInt, 11
  probe2gene, 40
  replicates, 45
  topTable, limma-method, 48
* methods
  spectralMap-methods, 47
  topTable, limma-method, 48
* models
  limmaTwoLevels, 24

a4palette, 3
addQuantilesColors, 4, 37
boxPlot, 5, 30, 42
combineTwoExpressionSet, 6
computeLogRatio, 7, 11, 37
createExpressionSet, 9
eSet, 11
ExpressionSet, 7, 9–11
ExpressionSetWithComputation-class, 10
filterfun, 12
filterVarInt, 11
gpar, 14
heatmap.expressionSet, 12
histPvalue, 20, 22, 44
histPvalue, limma-method (histPvalue), 20
histPvalue, MArrayLM-method (histPvalue), 20
histPvalue, numeric-method (histPvalue), 20
histPvalueplotter, 21
lassoClass, 23, 40
lassoReg, 22
limmaReg, 23
limmaTwoLevels, 24
logReg, 25, 39
nlcv, 27
nlcvTT, 26
oaColors, 27
oaPalette, 28
p.adjust, 49
pairs, 33
palettes, 3
par, 5, 29, 31, 41
plot, 25
plot.mpm, 46
plot1gene, 6, 28, 32, 42