limma

April 19, 2009

`01.Introduction` *Introduction to the LIMMA Package*

Description

LIMMA is a library for the analysis of gene expression microarray data, especially the use of linear models for analysing designed experiments and the assessment of differential expression. LIMMA provides the ability to analyse comparisons between many RNA targets simultaneously in arbitrary complicated designed experiments. Empirical Bayesian methods are used to provide stable results even when the number of arrays is small. The normalization and data analysis functions are for two-colour spotted microarrays. The linear model and differential expression functions apply to all microarrays including Affymetrix and other multi-array oligonucleotide experiments.

There are three types of documentation available. (1) The *LIMMA User's Guide* can be reached through the "User Guides and Package Vignettes" links at the top of the LIMMA contents page. The function `limmaUsersGuide` gives the file location of the User's Guide. (2) An overview of limma functions grouped by purpose is contained in the numbered chapters at the top of the LIMMA contents page, of which this page is the first. (3) The LIMMA contents page gives an alphabetical index of detailed help topics.

The function `changeLog` displays the record of changes to the package.

Author(s)

Gordon Smyth

References

Smyth, G. K., Yang, Y.-H., Speed, T. P. (2003). Statistical issues in microarray data analysis. *Methods in Molecular Biology* 224, 111-136.

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, 2005.

02.Classes *Classes Defined by this Package*

Description

This package defines the following data classes.

RGList A class used to store raw intensities as they are read in from an image analysis output file, usually by `read.maimages`.

MAList Intensities converted to M-values and A-values, i.e., to with-spot and whole-spot contrasts on the log-scale. Usually created from an `RGList` using `MA.RG` or `normalizeWithinArrays`. Objects of this class contain one row for each spot. There may be more than one spot and therefore more than one row for each probe.

MArrayLM Store the result of fitting gene-wise linear models to the normalized intensities or log-ratios. Usually created by `lmFit`. Objects of this class normally contain only one row for each unique probe.

TestResults Store the results of testing a set of contrasts equal to zero for each probe. Usually created by `decideTests`. Objects of this class normally contain one row for each unique probe.

All these data classes obey many analogies with matrices. In the case of `RGList` and `MAList`, rows correspond to spots and columns to arrays. In the case of `MarrayLM`, rows correspond to unique probes and the columns to parameters or contrasts. The functions `summary`, `dim`, `length`, `ncol`, `nrow`, `dimnames`, `rownames`, `colnames` have methods for these classes. Objects of any of these classes may be subsetted. Multiple data objects may be combined by rows (to add extra probes) or by columns (to add extra arrays).

Furthermore all of these classes may be coerced to actually be of class `matrix` using `as.matrix`, although this entails loss of information. Fitted model objects of class `MArrayLM` can be coerced to class `data.frame` using `as.data.frame`.

The first three classes belong to the virtual class `LargeDataObject`. A `show` method is defined for `LargeDataOjects` which uses the utility function `printHead`.

Author(s)

Gordon Smyth

03.ReadingData *Reading Microarray Data from Files*

Description

This help page gives an overview of LIMMA functions used to read data from files.

Reading Target Information

The function `readTargets` is designed to help with organizing information about which RNA sample is hybridized to each channel on each array and which files store information for each array.

Reading Intensity Data

The first step in a microarray data analysis is to read into R the intensity data for each array provided by an image analysis program. This is done using the function `read.maimages`.

`read.maimages` optionally constructs quality weights for each spot using quality functions listed in QualityWeights.

`read.maimages` produces an `RGList` object and stores only the information required from each image analysis output file. `read.maimages` uses utility functions `removeExt`, `read.imagene` and `read.columns`. There are also a series of utility functions which read the header information from image output files including `readGPRHeader`, `readImaGeneHeader` and `readGenericHeader`.

The function as.MAList can be used to convert a `marrayNorm` object to an `MAList` object if the data was read and normalized using the marray and marrayNorm packages.

Reading the Gene List

Most image analysis software programs provide gene IDs as part of the intensity output files, for example GenePix, Imagene and the Stanford Microarray Database do this. In other cases the probe ID and annotation information may be in a separate file. The most common format for the probe annotation file is the GenePix Array List (GAL) file format. The function `readGAL` reads information from a GAL file and produces a data frame with standard column names.

The function `getLayout` extracts from the GAL-file data frame the print layout information for a spotted array. The functions `gridr`, `gridc`, `spotr` and `spotc` use the extracted layout to compute grid positions and spot positions within each grid for each spot. The function `printorder` calculates the printorder, plate number and plate row and column position for each spot given information about the printing process. The utility function `getSpacing` converts character strings specifying spacings of duplicate spots to numeric values.

The Australian Genome Research Facility in Australia often produces GAL files with composite probe IDs or names consisting of multiple strings separated by a delimiter. These can be separated into name and annotation information using `strsplit2`.

If each probe is printed more than once of the arrays in a regular pattern, then `uniquegenelist` will remove duplicate names from the gal-file or gene list.

Identifying Control Spots

The functions `readSpotTypes` and `controlStatus` assist with separating control spots from ordinary genes in the analysis and data exploration.

Manipulating Data Objects

`cbind`, `rbind`, `merge` allow different `RGList` or `MAList` objects to be combined. `cbind` combines data from different arrays assuming the layout of the arrays to be the same. `merge` can combine data even when the order of the probes on the arrays has changed. `merge` uses utility function `makeUnique`.

Author(s)

Gordon Smyth

04.Background *Background Correction*

Description

This page deals with background correction methods for two-color microarray data.

Usually one doesn't need to explicitly ask for background correction of the intensities because this is done by default by normalizeWithinArrays, which subtracts the background from the foreground intensities before applying the normalization method. This default background correction method can be over-ridden by using backgroundCorrect which offers a number of alternative background correct methods to simple subtraction. The function backgroundCorrect is used to correct the RGList before applying normalizeWithinArrays.

The movingmin method of backgroundCorrect uses utility functions ma3x3.matrix and ma3x3.spottedarray.

The normexp method of backgroundCorrect uses utility functions normexp.fit and normexp.signal.

kooperberg is a Bayesian background correction tool designed specifically for GenePix data. kooperberg is not currently used as the default method for GenePix data because it is computationally intensive. It requires several additional columns from the GenePix data files which can be read in using read.maimages and specifying the other.columns argument.

Author(s)

Gordon Smyth

05.Normalization *Normalization of Microarray Data*

Description

This page gives an overview of the LIMMA functions available to normalize data from spotted two-colour microarrays. Smyth and Speed (2003) give an overview of the normalization techniques implemented in the functions.

Usually data from spotted microarrays will be normalized using normalizeWithinArrays. A minority of data will also be normalized using normalizeBetweenArrays if diagnostic plots suggest a difference in scale between the arrays.

In rare circumstances, data might be normalized using normalizeForPrintorder before using normalizeWithinArrays.

All the normalization routines take account of spot quality weights which might be set in the data objects. The weights can be temporarily modified using modifyWeights to, for example, remove ratio control spots from the normalization process.

If one is planning analysis of single-channel information from the microarrays rather than analysis of differential expression based on log-ratios, then the data should be normalized using a single channel-normalization technique. Single channel normalization uses further options of the normalizeBetweenArrays function. For more details see the *LIMMA User's Guide* which includes a section on single-channel normalization.

normalizeWithinArrays uses utility functions MA.RG, loessFit and normalizeRobustSpline. normalizeBetweenArrays uses utility functions normalizeMedianAbsValues and normalizeQuantile none of which need to be called directly by users.

removeBatchEffect can be used to remove a batch effect, associated with hybridization time or some other technical variable, prior to unsupervised analysis.

Author(s)

Gordon Smyth

References

Smyth, G. K., and Speed, T. P. (2003). Normalization of cDNA microarray data. In: *METHODS: Selecting Candidate Genes from DNA Array Screens: Application to Neuroscience*, D. Carter (ed.). Methods Volume 31, Issue 4, December 2003, pages 265-273. http://www.statsci.org/smyth/pubs/normalize.pdf

---

06.LinearModels *Linear Models for Microarrays*

Description

This page gives an overview of the LIMMA functions available to fit linear models and to interpret the results. This page covers models for two color arrays in terms of log-ratios or for single-channel arrays in terms of log-intensities. If you wish to fit models to the individual channel log-intensities from two colour arrays, see 07.SingleChannel.

The core of this package is the fitting of gene-wise linear models to microarray data. The basic idea is to estimate log-ratios between two or more target RNA samples simultaneously. See the LIMMA User's Guide for several case studies.

Fitting Models

The main function for model fitting is lmFit. This is recommended interface for most users. lmFit produces a fitted model object of class MArrayLM containing coefficients, standard errors and residual standard errors for each gene. lmFit calls one of the following three functions to do the actual computations:

lm.series Straightforward least squares fitting of a linear model for each gene.

mrlm An alternative to lm.series using robust regression as implemented by the rlm function in the MASS package.

gls.series Generalized least squares taking into account correlations between duplicate spots (i.e., replicate spots on the same array) or related arrays. The function duplicateCorrelation is used to estimate the inter-duplicate or inter-block correlation before using gls.series.

All the functions which fit linear models use link{getEAW} to extract data from microarray data objects, and unwrapdups which provides an unified method for handling duplicate spots.

Forming the Design Matrix

`lmFit` has two main arguments, the expression data and the design matrix. The design matrix is essentially an indicator matrix which specifies which target RNA samples were applied to each channel on each array. There is considerable freedom in choosing the design matrix - there is always more than one choice which is correct provided it is interpreted correctly.

Design matrices for Affymetrix or single-color arrays can be created using the function `model.matrix` which is part of the R base package. The function `modelMatrix` is provided to assist with creation of an appropriate design matrix for two-color microarray experiments. For direct two-color designs, without a common reference, the design matrix often needs to be created by hand.

Making Comparisons of Interest

Once a linear model has been fit using an appropriate design matrix, the command `makeContrasts` may be used to form a contrast matrix to make comparisons of interest. The fit and the contrast matrix are used by `contrasts.fit` to compute fold changes and t-statistics for the contrasts of interest. This is a way to compute all possible pairwise comparisons between treatments for example in an experiment which compares many treatments to a common reference.

Assessing Differential Expression

After fitting a linear model, the standard errors are moderated using a simple empirical Bayes model using `eBayes` or `treat`. `ebayes` is an older version of `eBayes`. A moderated t-statistic and a log-odds of differential expression is computed for each contrast for each gene. `treat` tests whether log-fold-changes are greater than a threshold rather than merely different to zero.

`eBayes` and `eBayes` use internal functions `squeezeVar`, `fitFDist`, `tmixture.matrix` and `tmixture.vector`.

The function `zscoreT` is sometimes used for computing z-score equivalents for t-statistics so as to place t-statistics with different degrees of freedom on the same scale. `zscoreGamma` is used the same way with standard deviations instead of t-statistics. These functions are for research purposes rather than for routine use.

Summarizing Model Fits

After the above steps the results may be displayed or further processed using:

`toptable` or `topTable` Presents a list of the genes most likely to be differentially expressed for a given contrast.

`topTableF` Presents a list of the genes most likely to be differentially expressed for a given set of contrasts.

`volcanoplot` Volcano plot of fold change versus the B-statistic for any fitted coefficient.

`plotlines` Plots fitted coefficients or log-intensity values for time-course data.

`write.fit` Writes an `MarrayLM` object to a file. Note that if `fit` is an `MArrayLM` object, either `write.fit` or `write.table` can be used to write the results to a delimited text file.

For multiple testing functions which operate on linear model fits, see 08.Tests.

Model Selection

`selectModel` provides a means to choose between alternative linear models using AIC or BIC information criteria.

Author(s)

Gordon Smyth

References

Smyth, G. K. (2004). Linear models and empirical Bayes methods for assessing differential expression in microarray experiments. *Statistical Applications in Genetics and Molecular Biology*, **3**, No. 1, Article 3. http://www.bepress.com/sagmb/vol3/iss1/art3

Smyth, G. K., Michaud, J., and Scott, H. (2005). The use of within-array replicate spots for assessing differential expression in microarray experiments. Bioinformatics 21(9), 2067-2075.

07.SingleChannel *Individual Channel Analysis of Two-Color Microarrays*

Description

This page gives an overview of the LIMMA functions fit linear models to two-color microarray data in terms of the log-intensities rather than log-ratios.

The function `intraspotCorrelation` estimates the intra-spot correlation between the two channels. The regression function `lmscFit` takes the correlation as an argument and fits linear models to the two-color data in terms of the individual log-intensities. The output of `lmscFit` is an `MArrayLM` object just the same as from `lmFit`, so inference proceeds in the same way as for log-ratios once the linear model is fitted. See 06.LinearModels.

The function `targetsA2C` converts two-color format target data frames to single channel format, i.e, converts from array-per-line to channel-per-line, to facilitate the formulation of the design matrix.

Author(s)

Gordon Smyth

08.Tests *Hypothesis Testing for Linear Models*

Description

LIMMA provides a number of functions for multiple testing across both contrasts and genes. The starting point is an `MArrayLM` object, called `fit` say, resulting from fitting a linear model and running `eBayes` and, optionally, `contrasts.fit`. See 06.LinearModels or 07.SingleChannel for details.

Multiple testing across genes and contrasts

The key function is `decideTests`. This function writes an object of class `TestResults`, which is basically a matrix of `-1`, `0` or `1` elements, of the same dimension as `fit$coefficients`, indicating whether each coefficient is significantly different from zero. A number of different multiple testing strategies are provided. The function calls other functions `classifyTestsF`, `classifyTestsP` and `classifyTestsT` which implement particular strategies. The function `FStat` provides an alternative interface to `classifyTestsF` to extract only the overall moderated F-statistic.

A number of other functions are provided to display the results of `decideTests`. The functions `heatDiagram` (or the older version `heatdiagram` displays the results in a heat-map style display. This allows visual comparison of the results across many different conditions in the linear model.

The functions `vennCounts` and `vennDiagram` provide Venn diagram style summaries of the results.

Summary and `show` method exists for objects of class `TestResults`.

The results from `decideTests` can also be included when the results of a linear model fit are written to a file using `write.fit`.

Gene Set Tests

Competitive gene set testing is provided by `geneSetTest`, which permutes genes, while self-contained gene set testing is provided by `roast`, which randomly rotates arrays.

The function `alias2Symbol` is provided to help match gene sets with microarray probes by way of official gene symbols.

Other Functions

Given a set of p-values, the function `convest` can be used to estimate the proportion of true null hypotheses.

When evaluating test procedures with simulated or known results, the utility function `auROC` can be used to compute the area under the Receiver Operating Curve for the test results for a given probe.

Author(s)

Gordon Smyth

`09.Diagnostics` *Diagnostics and Quality Assessment*

Description

This page gives an overview of the LIMMA functions available for microarray quality assessment and diagnostic plots.

This package provides an `anova` method which is designed for assessing the quality of an array series or of a normalization method. It is not designed to assess differential expression of individual genes. `anova` uses utility functions `bwss` and `bwss.matrix`.

The function `arrayWeights` estimates the empirical reliability of each array following a linear model fit.

Diagnostic plots can be produced by

imageplot Produces a spatial picture of any spot-specific measure from an array image. If the log-ratios are plotted, then this produces an in-silico representation of the well known false-color TIFF image of an array. imageplot3by2 will write imageplots to files, six plots to a page.

plotFB Plots foreground versus background log-intensies for an array.

plotMA MA-plots. One of the most useful plots of a two-color array. plotMA3by2 will write MA-plots to files, six plots to a page. mdplot can also be useful for comparing two one-channel microarrays.

plotPrintTipLoess Produces a grid of MA-plots, one for each print-tip group on an array, together with the corresponding lowess curve. Intended to help visualize print-tip loess normalization.

plotPrintorder For an array, produces a scatter plot of log-ratios or log-intensities by print order.

plotDensities Individual channel densities for one or more arrays. An essential plot to accompany between array normalization, especially quantile normalization.

plotPrintTipLoess uses utility functions gridr and gridc. plotDensities uses utility function RG.MA.

Author(s)

Gordon Smyth

`10.Other` *Other Functions*

Description

This page describes some functions not covered in the previous numbered pages, so far only blockDiag and poolVar which are not used in the package yet but are part of the development of methods to handle technical and biological replicates.

Author(s)

Gordon Smyth

`LargeDataObject-class`
Large Data Object - class

Description

A virtual class including the data classes RGList, MAList and MArrayLM, all of which typically contain large quantities of numerical data in vector, matrices and data.frames.

Methods

A show method is defined for objects of class LargeDataObject which uses printHead to print only the leading elements or rows of components or slots which contain large quantities of data.

Author(s)

Gordon Smyth

See Also

02.Classes gives an overview of all the classes defined by this package.

Examples

```
#  see normalizeBetweenArrays
```

`PrintLayout` *Print Layout - class*

Description

A list-based class for storing information about the process used to print spots on a microarray.

`PrintLayout` objects can be created using `getLayout`. The `printer` component of an `RGList` or `MAList` object is of this class.

Slots/List Components

Objects of this class contains no slots but should contain the following list components:

`ngrid.r`:	number of grid rows on the arrays
`ngrid.c`:	number of grid columns on the arrays
`nspot.r`:	number of rows of spots in each grid
`nspot.c`:	number of columns of spots in each grid
`ndups`:	number of duplicates of each DNA clone, i.e., number of times print-head dips into each well of DNA
`spacing`:	number of spots between duplicate spots. Only applicable if `ndups>1`. `spacing=1` for side-by-side spots
`npins`:	actual number of pins or tips on the print-head
`start`:	character string giving position of the spot printed first in each grid. Choices are `"topleft"` or `"toprigh`

Author(s)

Gordon Smyth

See Also

02.Classes gives an overview of all the classes defined by this package.

Examples

```
#  Settings for Swirl and ApoAI example data sets in User's Guide

printer <- list(ngrid.r=4, ngrid.c=4, nspot.r=22, nspot.c=24, ndups=1, spacing=1, npins=1

#  Typical settings at the Australian Genome Research Facility

#  Full pin set, duplicates side-by-side on same row
printer <- list(ngrid.r=12, ngrid.c=4, nspot.r=20, nspot.c=20, ndups=2, spacing=1, npins=
```

```
#  Half pin set, duplicates in top and lower half of slide
printer <- list(ngrid.r=12, ngrid.c=4, nspot.r=20, nspot.c=20, ndups=2, spacing=9600, npi
```

TestResults-class *Matrix of Test Results - class*

Description

A matrix-based class for storing the results of simultanous tests. TestResults objects are normally created by classifyTestsF, classifyTestsT or classifyTestsP.

Usage

```
## S3 method for class 'TestResults':
summary(object, ...)
```

Arguments

object	object of class TestResults
...	other arguments are not used

Slots/List Components

TestResults objects can be created by new("TestResults",results) where results is a matrix. Objects of this class contain no slots (other than .Data), although the attributes dim and dimnames may be treated as slots.

Methods

This class inherits directly from class matrix so any operation appropriate for matrices will work on objects of this class. show and summary methods are also implemented.

Functions in LIMMA which operate on TestResults objects include heatDiagram, vennCounts, vennDiagram, write.fit.

Author(s)

Gordon Smyth

See Also

02.Classes gives an overview of all the classes defined by this package. 08.Tests gives an overview of multiple testing.

Examples

```
## Not run:
#  Assume a data object MA and a design matrix
fit <- lmFit(MA, design)
fit <- eBayes(fit)
results <- decideTests(fit)
summary(results)
## End(Not run)
```

alias2Symbol *Convert Gene Alias to Official Gene Symbols*

Description

Map a set of gene alias names to a set of gene symbols.

Usage

```
alias2Symbol(alias, species = "Hs")
```

Arguments

alias	character vector of gene aliases
species	character string specifying the species. Possible values are "Dm", "Hs", "Mm" or "Rn".

Details

Aliases are mapped via NCBI Entrez Gene identity numbers using Bioconductor organism packages.

Value

A character vector of gene symbols. May be longer or shorter than the original vector.

Author(s)

Gordon Smyth

See Also

unwrapdups

Examples

```
if(!require("org.Hs.eg.db")) alias2Symbol(c("PUMA","NOXA"))
```

anova.MAList-method *ANOVA Table - method*

Description

Analysis of variance method for objects of class MAList. Produces an ANOVA table useful for quality assessment by decomposing between and within gene sums of squares for a series of replicate arrays. This method produces a single ANOVA Table rather than one for each gene and is not used to identify differentially expressed genes.

Usage

```
anova(object,design=NULL,ndups=2,...)
```

Arguments

object	object of class MAList. Missing values in the M-values are not allowed.
design	numeric vector or single-column matrix containing the design matrix for linear model. The length of the vector or the number of rows of the matrix should agree with the number of columns of M.
ndups	number of duplicate spots. Each gene is printed ndups times in adjacent spots on each array.
...	other arguments are not used

Details

This function aids in quality assessment of microarray data and in the comparison of normalization methodologies. It applies only to replicated two-color experiments in which all the arrays are hybridized with the same RNA targets, possibly with dye-swaps, so the design matrix should have only one column. The function has not been heavily used and is somewhat experimental.

Value

An object of class anova containing rows for between genes, between arrays, gene x array interaction, and between duplicate with array sums of squares. Variance components are estimated for each source of variation.

Note

This function does not give valid results in the presence of missing M-values.

Author(s)

Gordon Smyth

See Also

MAList-class, bwss.matrix, anova.

An overview of quality assessment and diagnostic functions in LIMMA is given by 09.Diagnostics.

---

arrayWeights *Array Quality Weights*

Description

Estimates relative quality weights for each array in a multi-array experiment.

Usage

```
arrayWeights(object, design = NULL, weights = NULL, method = "genebygene", maxit
arrayWeightsSimple(object, design = NULL, maxiter = 100, tol = 1e-6, maxratio =
```

Arguments

object	object of class numeric, matrix, MAList, marrayNorm, ExpressionSet or PLMset containing log-ratios or log-values of expression for a series of microarrays.
design	the design matrix of the microarray experiment, with rows corresponding to arrays and columns to coefficients to be estimated. Defaults to the unit vector meaning that the arrays are treated as replicates.
weights	optional numeric matrix containing prior weights for each spot.
method	character string specifying the estimating algorithm to be used. Choices are "genebygene" and "reml".
maxiter	maximum number of iterations allowed.
tol	convergence tolerance.
maxratio	maximum ratio between largest and smallest weights before iteration stops
trace	logical variable. If true then output diagnostic information at each iteration of '"reml"' algorithm.

Details

The relative reliability of each array is estimated by measuring how well the expression values for that array follow the linear model.

The method is described in Ritchie et al (2006). A heteroscedastic model is fitted to the expression values for each gene by calling the function lm.wfit. The dispersion model is fitted to the squared residuals from the mean fit, and is set up to have array specific coefficients, which are updated in either full REML scoring iterations, or using an efficient gene-by-gene update algorithm. The final estimates of these array variances are converted to weights.

The data object object is interpreted as for lmFit. In particular, the arguments design and weights will be extracted from the data object if available and do not normally need to be set explicitly in the call; if any of these are set in the call then they will over-ride the slots or components in the data object.

arrayWeightsSimple is a fast version of arrayWeights with method="reml", no prior weights and no missing values.

Value

A vector of array weights.

Author(s)

Matthew Ritchie and Gordon Smyth

References

Ritchie, M. E., Diyagama, D., Neilson, van Laar, R., J., Dobrovic, A., Holloway, A., and Smyth, G. K. (2006). Empirical array quality weights in the analysis of microarray data. BMC Bioinformatics 7, 261. http://www.biomedcentral.com/1471-2105/7/261/abstract

See Also

An overview of linear model functions in limma is given by 06.LinearModels.

Examples

```
library(sma)
# Subset of data from ApoAI case study in Limma User's Guide
data(MouseArray)
# Avoid non-positive intensities
RG <- backgroundCorrect(mouse.data, method="half")
MA <- normalizeWithinArrays(RG, mouse.setup)
MA <- normalizeBetweenArrays(MA, method="Aq")
targets <- data.frame(Cy3=I(rep("Pool",6)),Cy5=I(c("WT","WT","WT","KO","KO","KO")))
design <- modelMatrix(targets, ref="Pool")
arrayw <- arrayWeightsSimple(MA, design)
fit <- lmFit(MA, design, weights=arrayw)
fit2 <- contrasts.fit(fit, contrasts=c(-1,1))
fit2 <- eBayes(fit2)
# Use of array weights increases the significance of the top genes
topTable(fit2)
```

---

arrayWeightsQuick *Array Quality Weights*

Description

Estimates relative quality weights for each array in a multi-array experiment with replication.

Usage

```
arrayWeightsQuick(y, fit)
```

Arguments

y	the data object used to estimate fit. Can be of any class which can be coerced to matrix, including matrix, MAList, marrayNorm or ExpressionSet.
fit	MArrayLM fitted model object

Details

Estimates the relative reliability of each array by measuring how well the expression values for that array follow the linear model.

This is a quick and dirty version of arrayWeights.

Value

Numeric vector of weights of length ncol(fit).

Author(s)

Gordon Smyth

References

Ritchie, M. E., Diyagama, D., Neilson, van Laar, R., J., Dobrovic, A., Holloway, A., and Smyth, G. K. (2006). Empirical array quality weights for microarray data. BMC Bioinformatics. (Accepted 11 April 2006)

See Also

See arrayWeights. An overview of LIMMA functions for reading data is given in 03.ReadingData.

Examples

```
## Not run:
fit <- lmFit(y, design)
arrayWeightsQuick(y, fit)
## End(Not run)
```

`asMatrixWeights` *asMatrixWeights*

Description

Convert probe-weights or array-weights to a matrix of weights.

Usage

```
asMatrixWeights(weights, dim)
```

Arguments

`weights`	numeric matrix of weights, rows corresponding to probes and columns to arrays. Or vector of probe weights. Or vector of array weights.
`dim`	numeric dimension vector of length 2, i.e., the number of probes and the number of arrays.

Details

This function converts a vector or probe-weights or a vector of array-weights to a matrix of the correct size. Probe-weights are repeated across rows while array-weights are repeated down the columns. If `weights` has length equal to the number of probes, it is assumed to contain probe-weights. If it has length equal to the number of arrays, it is assumed to contain array-weights. If the number of probes is equal to the number of arrays, then `weights` is assumed to contain array-weights if it is a row-vector of the correct size, i.e., if it is a matrix with one row.

This function is used internally by the linear model fitting functions in limma.

Value

Numeric matrix of dimension `dim`.

Author(s)

Gordon Smyth

See Also

`modifyWeights`.

An overview of functions in LIMMA used for fitting linear models is given in 06.LinearModels.

Examples

```
asMatrixWeights(1:3,c(4,3))
asMatrixWeights(1:4,c(4,3))
```

`as.data.frame` *Turn a Microarray Linear Model Object into a Dataframe*

Description

Turn a `MArrayLM` object into a `data.frame`.

Usage

```
## S3 method for class 'MArrayLM':
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

`x`	an object of class `MArrayLM`
`row.names`	`NULL` or a character vector giving the row names for the data frame. Missing values are not allowed.
`optional`	logical. If `TRUE`, setting row names and converting column names (to syntactic names) is optional.
`...`	additional arguments to be passed to or from methods.

Details

This method combines all the components of `x` which have a row for each probe on the array into a `data.frame`.

Value

A data.frame.

Author(s)

Gordon Smyth

See Also

`as.data.frame` in the base package.

02.Classes gives an overview of data classes used in LIMMA. 06.LinearModels gives an overview of linear model functions in LIMMA.

as.MAList *Convert marrayNorm Object to an MAList Object*

Description

Convert marrayNorm Object to an MAList Object

Usage

```
as.MAList(object)
```

Arguments

object	an marrayNorm object

Value

Object of class MAList

Author(s)

Gordon Smyth

See Also

02.Classes gives an overview of all the classes defined by this package.

as.matrix *Turn a Microarray Data Object into a Matrix*

Description

Turn a microarray data object into a numeric matrix by extracting the expression values.

Usage

```
## S3 method for class 'MAList':
as.matrix(x,...)
```

Arguments

x	an object of class RGList, MAList, MArrayLM, marrayNorm, PLMset, ExpressionSet, LumiBatch or vsn.
...	additional arguments, not used for these methods.

Details

These methods extract the matrix of log-ratios, for MAList or marrayNorm objects, or the matrix of expression values for other expression objects such as ExressionSet. For MArrayLM objects, the matrix of fitted coefficients is extracted.

These methods involve loss of information, so the original data object is not recoverable.

Value

A numeric matrix.

Author(s)

Gordon Smyth

See Also

`as.matrix` in the base package or `exprs` in the Biobase package.

02.Classes gives an overview of data classes used in LIMMA.

---

auROC *Area Under Receiver Operating Curve*

Description

Compute exact area under the ROC for empirical data.

Usage

```
auROC(truth, stat)
```

Arguments

`truth`	numeric vector of 0 and 1 indicating whether the null or alternative respectively is true for each case. If `stat` is missing then `truth` is assuming to be already sorted in decreasing test statistic order.
`stat`	numeric vector containing test statistics. Hypotheses are to be rejected if `stat` exceeds a given threshold.

Details

This function computes the exact area under an empirical ROC curve. The number of true and false discoveries are determined by how well the true states represented by `truth` match up with the observed statistics given by `stat`.

Value

Numeric vector giving area under the curve, 1 being perfect and 0 being the minimum, or `NULL` if `truth` has zero length.

Author(s)

Gordon Smyth

See Also

See 08.Tests for other functions for testing and processing p-values.

See also `AUC` in the `ROC` package.

Examples

```
auROC(c(1,1,0,0,0))
truth <- rbinom(30,size=1,prob=0.2)
stat <- rchisq(30,df=2)
auROC(truth,stat)
```

`avedups` *Average Over Duplicate Spots*

Description

Condense a microarray data object so that values for within-array replicate spots are replaced with their average.

Usage

```
## Default S3 method:
avedups(x, ndups=2, spacing=1, weights=NULL)
## S3 method for class 'MAList':
avedups(x, ndups=x$printer$ndups, spacing=x$printer$spacing, weights=x$weights)
```

Arguments

`x`	a matrix-like object, usually a matrix or an `MAList` object.
`ndups`	number of within-array replicates for each probe.
`spacing`	number of spots to step from a probe to its duplicate.
`weights`	numeric matrix of spot weights.

Details

A new data object is computed in which each probe is represented by the (weighted) average of its duplicate spots. For a `MAList` object, the components `M` and `A` are both averaged in this way.

If `x` is of mode `"character"`, then the duplicate values are assumed to be equal and the first is taken as the average.

Value

A data object of the same class as `x` with `1/ndups` as many rows.

Author(s)

Gordon Smyth

See Also

`dimnames` in the base package.

02.Classes gives an overview of data classes used in LIMMA.

avereps *Average Over Irregular Replicate Spots*

Description

Condense a microarray data object so that values for within-array replicate spots are replaced with their average.

Usage

```
## Default S3 method:
avereps(x, ID=rownames(x))
## S3 method for class 'MAList':
avereps(x, ID=NULL)
```

Arguments

x	a matrix-like object, usually a matrix or an MAList object.
ID	probe identifier.

Details

A new data object is computed in which each probe is represented by the average of its replicate spots. For an MAList object, the components M and A are both averaged in this way, as weights and any matrices found in object$other.

For an MAList object, ID defaults to MA$genes$ID is that exists, otherwise to rownames(MA$M).

If x is of mode "character", then the replicate values are assumed to be equal and the first is taken as the average.

Value

A data object of the same class as x with a row for each unique value of ID.

Author(s)

Gordon Smyth

See Also

dimnames in the base package.

02.Classes gives an overview of data classes used in LIMMA.

backgroundCorrect *Correct Intensities for Background*

Description

Background correct microarray expression intensities.

Usage

```
backgroundCorrect(RG, method="subtract", offset=0, printer=RG$printer, normexp.m
```

Arguments

RG	an RGList object or a numeric matrix.
method	character string specifying correction method. Possible values are "none", "subtract", "half", "minimum", "movingmin", "edwards" or "normexp". If RG is a matrix, possible values are restricted to "none" or "normexp".
offset	numeric value to add to intensities
printer	a list containing printer layout information, see PrintLayout-class. Ignored if RG is a matrix.
normexp.method	character string specifying parameter estimation strategy used by normexp, ignored for other methods. Possible values are "saddle", "mle" or "rma".
verbose	logical. If TRUE, progress messages are sent to standard output

Details

This function implements the background correction methods reviewed or developed in Ritchie et al (2007) and Silver at al (2009). Ritchie et al (2007) recommend method="normexp" whenever RG contains local background estimates. Silver et al (2009) shows that either normexp.method="mle" or normexp.method="saddle" are excellent options for normexp. If RG contains morphological background estimates instead (available from SPOT or GenePix image analysis software), then method="subtract" performs well.

If method="none" then no correction is done, i.e., the background intensities are treated as zero. If method="subtract" then the background intensities are subtracted from the foreground intensities. This is the traditional background correction method, but is not necessarily recommended. If method="movingmin" then the background estimates are replaced with the minimums of the backgrounds of the spot and its eight neighbors, i.e., the background is replaced by a moving minimum of 3x3 grids of spots.

The remaining methods are all designed to produce positive corrected intensities. If method="half" then any intensity which is less than 0.5 after background subtraction is reset to be equal to 0.5. If method="minimum" then any intensity which is zero or negative after background subtraction is set equal to half the minimum of the positive corrected intensities for that array. If method="edwards" a log-linear interpolation method is used to adjust lower intensities as in Edwards (2003). If method="normexp" a convolution of normal and exponential distributions is fitted to the foreground intensities using the background intensities as a covariate, and the expected signal given the observed foreground becomes the corrected intensity. This results in a smooth monotonic transformation of the background subtracted intensities such that all the corrected intensities are positive.

The normexp method is available in a number of variants depending on how the model parameters are estimated, and these are selected by `normexp.method`. Here `"saddle"` gives the saddle-point approximation to maximum likelihood from Ritchie et al (2007), `"mle"` gives exact maximum likelihood from Silver at al (2009), `"rma"` gives the background correction algorithm from the RMA-algorithm for Affymetrix microarray data as implemented in the affy package, and `"rma75"` gives the RMA-75 method from McGee and Chen (2006). In practice `"mle"` performs well and is nearly as fast as `"saddle"`, but `"saddle"` is the default for backward compatibility. See `normexp.fit` for more details.

The `offset` can be used to add a constant to the intensities before log-transforming, so that the log-ratios are shrunk towards zero at the lower intensities. This may eliminate or reverse the usual 'fanning' of log-ratios at low intensities associated with local background subtraction.

Background correction (background subtraction) is also performed by the `normalizeWithinArrays` method for `RGList` objects, so it is not necessary to call `backgroundCorrect` directly unless one wants to use a method other than simple subtraction. Calling `backgroundCorrect` before `normalizeWithinArrays` will over-ride the default background correction.

Value

An `RGList` object in which components `R` and `G` are background corrected and components `Rb` and `Gb` are removed.

Author(s)

Gordon Smyth

References

Edwards, D. E. (2003). Non-linear normalization and background correction in one-channel cDNA microarray studies *Bioinformatics* 19, 825-833.

McGee, M., and Chen, Z. (2006). Parameter estimation for the exponential-normal convolution model for background correction of Affymetrix GeneChip data. *Stat Appl Genet Mol Biol*, Volume 5, Article 24.

Ritchie, M. E., Silver, J., Oshlack, A., Silver, J., Holmes, M., Diyagama, D., Holloway, A., and Smyth, G. K. (2007). A comparison of background correction methods for two-colour microarrays. *Bioinformatics* 23, 2700-2707. http://bioinformatics.oxfordjournals.org/cgi/content/abstract/btm412

Silver, J., Ritchie, M. E., and Smyth, G. K. (2009). Microarray background correction: maximum likelihood estimation for the normal-exponential convolution model. *Biostatistics*. http://biostatistics.oxfordjournals.org/cgi/content/abstract/kxn042

See Also

An overview of background correction functions is given in `04.Background`.

Examples

```
RG <- new("RGList", list(R=c(1,2,3,4),G=c(1,2,3,4),Rb=c(2,2,2,2),Gb=c(2,2,2,2)))
backgroundCorrect(RG)
backgroundCorrect(RG, method="half")
backgroundCorrect(RG, method="minimum")
backgroundCorrect(RG, offset=5)
```

blockDiag *Block Diagonal Matrix*

Description

Form a block diagonal matrix from the given blocks.

Usage

```
blockDiag(...)
```

Arguments

...	numeric matrices

Value

A block diagonal matrix with dimensions equal to the sum of the input dimensions

Author(s)

Gordon Smyth

See Also

10.Other

Examples

```
a <- matrix(1,3,2)
b <- matrix(2,2,2)
blockDiag(a,b)
```

bwss *Between and within sums of squares*

Description

Sums of squares between and within groups. Allows for missing values.

Usage

```
bwss(x,group)
```

Arguments

x	a numeric vector giving the responses.
group	a vector or factor giving the grouping variable.

Details

This is equivalent to one-way analysis of variance.

Value

A list with components

bss	sums of squares between the group means.
wss	sums of squares within the groups.
bdf	degrees of freedom corresponding to bss.
wdf	degrees of freedom corresponding to wss.

Author(s)

Gordon Smyth

See Also

bwss.matrix

---

bwss.matrix *Between and within sums of squares for matrix*

Description

Sums of squares between and within the columns of a matrix. Allows for missing values. This function is called by the anova method for MAList objects.

Usage

```
bwss.matrix(x)
```

Arguments

x	a numeric matrix.

Details

This is equivalent to a one-way analysis of variance where the columns of the matrix are the groups. If x is a matrix then bwss.matrix(x) is the same as bwss(x,col(x)) except for speed of execution.

Value

A list with components

bss	sums of squares between the column means.
wss	sums of squares within the column means.
bdf	degrees of freedom corresponding to bss.
wdf	degrees of freedom corresponding to wss.

Author(s)

Gordon Smyth

See Also

bwss, anova.MAList

---

cbind *Combine RGList or MAList Objects*

Description

Combine a series of RGList objects or combine a series of MAList objects.

Usage

```
## S3 method for class 'RGList':
cbind(..., deparse.level=1)
## S3 method for class 'RGList':
rbind(..., deparse.level=1)
```

Arguments

...	RGList objects or MAList objects
deparse.level	not currently used, see cbind in the base package

Details

cbind combines data objects assuming the same gene lists but different arrays. rbind combines data objects assuming equivalent arrays, i.e., the same RNA targets, but different genes.

For cbind, the matrices of expression data from the individual objects are cbinded. The data.frames of target information, if they exist, are rbinded. The combined data object will preserve any additional components or attributes found in the first object to be combined. For rbind, the matrices of expression data are rbinded while the target information, in any, is unchanged.

Value

An RGList or MAList object holding data from all the arrays and all genes from the individual objects.

Author(s)

Gordon Smyth

See Also

cbind in the base package.

03.ReadingData gives an overview of data input and manipulation functions in LIMMA.

Examples

```
M <- A <- matrix(11:14,4,2)
rownames(M) <- rownames(A) <- c("a","b","c","d")
colnames(M) <- colnames(A) <- c("A1","A2")
MA1 <- new("MAList",list(M=M,A=A))

M <- A <- matrix(21:24,4,2)
rownames(M) <- rownames(A) <- c("a","b","c","d")
colnames(M) <- colnames(A) <- c("B1","B2")
MA2 <- new("MAList",list(M=M,A=A))

cbind(MA1,MA2)
```

---

changeLog *Limma Change Log*

Description

Write as text the most recent changes from the limma package changelog.

Usage

```
changeLog(n=20)
```

Arguments

n	integer, number of lines to write of changelog.

Value

No value is produced, but a number of lines of text are written to standard output.

Author(s)

Gordon Smyth

See Also

01.Introduction

designI2M *Convert Individual Channel Design Matrix to M-A Format*

Description

Convert a design matrix in terms of individual channels to ones in terms of M-values or A-values for two-color microarray data.

Usage

```
designI2M(design)
designI2A(design)
```

Arguments

design	numeric model matrix with one row for each channel observation, i.e., twice as many rows as arrays

Details

If design is a model matrix suitable for modelling individual log-intensities for two color microarray data, then designI2M computes the corresponding model matrix for modelling M-values (log-ratios) and designI2A computes the model matrix for modelling A-values (average log-intensities).

Note that the matrices designI2M(design) or designI2A(design) may be singular if not all of the coefficients are estimable from the M or A-values. In that case there will be columns containing entirely zeros.

Value

numeric model matrix with half as many rows as design

Author(s)

Gordon Smyth

See Also

model.matrix in the stats package.

An overview of individual channel linear model functions in limma is given by 07.SingleChannel.

Examples

```
X <- cbind(1,c(1,1,1,1,0,0,0,0),c(0,0,0,0,1,1,1,1))
designI2M(X)
designI2A(X)
```

`classifyTests` *Multiple Testing Genewise Across Contrasts*

Description

For each gene, classify a series of related t-statistics as up, down or not significant.

Usage

```
classifyTestsF(object, cor.matrix=NULL, df=Inf, p.value=0.01, fstat.only=FALSE)
classifyTestsT(object, t1=4, t2=3)
classifyTestsP(object, df=Inf, p.value=0.05, method="holm")
FStat(object, cor.matrix=NULL)
```

Arguments

`object`	numeric matrix of t-statistics or an `MArrayLM` object from which the t-statistics may be extracted.
`cor.matrix`	covariance matrix of each row of t-statistics. Defaults to the identity matrix.
`df`	numeric vector giving the degrees of freedom for the t-statistics. May have length 1 or length equal to the number of rows of `tstat`.
`p.value`	numeric value between 0 and 1 giving the desired size of the test
`fstat.only`	logical, if `TRUE` then return the overall F-statistic as for `FStat` instead of classifying the test results
`t1`	first critical value for absolute t-statistics
`t2`	second critical value for absolute t-statistics
`method`	character string specifying p-value adjustment method. See `p.adjust` for possible values.

Details

Note that these functions do not adjust for multiple testing across genes. The adjustment for multiple testing is across the contrasts rather than the more usual control across genes. The functions described here are called by `decideTests`. Most users should use `decideTests` rather than using these functions directly.

These functions implement multiple testing procedures for determining whether each statistic in a matrix of t-statistics should be considered significantly different from zero. Rows of `tstat` correspond to genes and columns to coefficients or contrasts.

`FStat` computes the gene-wise F-statistics for testing all the contrasts equal to zero. It is equivalent to `classifyTestsF` with `fstat.only=TRUE`.

`classifyTestsF` uses a nested F-test approach giving particular attention to correctly classifying genes which have two or more significant t-statistics, i.e., are differential expressed under two or more conditions. For each row of `tstat`, the overall F-statistics is constructed from the t-statistics as for `FStat`. At least one constrast will be classified as significant if and only if the overall F-statistic is significant. If the overall F-statistic is significant, then the function makes a best choice as to which t-statistics contributed to this result. The methodology is based on the principle that any t-statistic should be called significant if the F-test is still significant for that row when all the larger t-statistics are set to the same absolute size as the t-statistic in question.

classifyTestsT and classifyTestsP implement simpler classification schemes based on threshold or critical values for the individual t-statistics in the case of classifyTestsT or p-values obtained from the t-statistics in the case of classifyTestsP. For classifyTestsT, classifies any t-statistic with absolute greater than t2 as significant provided that at least one t-statistic for that gene is at least t1 in absolute value. classifyTestsP applied p-value adjustment from p.adjust to the p-values for each gene.

If tstat is an MArrayLM object, then all arguments except for p.value are extracted from it.

cor.matrix is the same as the correlation matrix of the coefficients from which the t-statistics are calculated. If cor.matrix is not specified, then it is calculated from design and contrasts if at least design is specified or else defaults to the identity matrix. In terms of design and contrasts, cor.matrix is obtained by standardizing the matrix t(contrasts) %*% solve(t(design) %*% design) %*% contrasts to a correlation matrix.

Value

An object of class TestResults. This is essentially a numeric matrix with elements -1, 0 or 1 depending on whether each t-statistic is classified as significantly negative, not significant or significantly positive respectively.

FStat produces a numeric vector of F-statistics with attributes df1 and df2 giving the corresponding degrees of freedom.

Author(s)

Gordon Smyth

See Also

An overview of multiple testing functions is given in 08.Tests.

Examples

```
tstat <- matrix(c(0,5,0, 0,2.5,0, -2,-2,2, 1,1,1), 4, 3, byrow=TRUE)
classifyTestsF(tstat)

# See also the examples for contrasts.fit and vennDiagram
```

---

contrasts.fit *Compute Contrasts from Linear Model Fit*

Description

Given a linear model fit to microarray data, compute estimated coefficients and standard errors for a given set of contrasts.

Usage

```
contrasts.fit(fit, contrasts=NULL, coefficients=NULL)
```

Arguments

fit	an MArrayLM object or a list object produced by the function lm.series or equivalent. Must contain components coefficients and stdev.unscaled.
contrasts	numeric matrix with row corresponding to coefficients in fit and columns containing contrasts. May be a vector if there is only one contrast.
coefficients	vector indicating which coefficients are to be kept in the revised fit object. An alternative way to specify the contrasts.

Details

This function accepts input from any of the functions lmFit, lm.series, mrlm, gls.series or lmscFit. The function re-orientates the fitted model object from the coefficients of the original design matrix to any set of contrasts of the original coefficients. The coefficients, unscaled standard deviations and correlation matrix are re-calculated in terms of the contrasts.

The idea of this function is to fit a full-rank model using lmFit or equivalent, then use contrasts.fit to obtain coefficients and standard errors for any number of contrasts of the coefficients of the original model. Unlike the design matrix input to lmFit, which normally has one column for each treatment in the experiment, the matrix contrasts may have any number of columns and these are not required to be linearly independent. Methods of assessing differential expression, such as eBayes or classifyTestsF, can then be applied to fitted model object.

The coefficients argument provides a simpler way to specify the contrasts matrix when the desired contrasts are just a subset of the original coefficients.

Warning. For efficiency reasons, this function does not re-factorize the design matrix for each probe. A consequence is that, if the design matrix is non-orthogonal and the original fit included quality weights or missing values, then the unscaled standard deviations produced by this function are approximate rather than exact. The approximation is usually acceptable. The results are always exact if the original fit was a oneway model.

Value

An list object of the same class as fit. This is a list with components components

coefficients	numeric matrix containing the estimated coefficients for each contrast for each probe.
stdev.unscaled	numeric matrix conformal with coef containing the unscaled standard deviations for the coefficient estimators.
cor.coefficients	numeric correlation matrix for the estimated coefficients
...	any other components input in fit

Author(s)

Gordon Smyth

See Also

An overview of linear model functions in limma is given by 06.LinearModels.

Examples

```
#  Simulate gene expression data: 6 microarrays and 100 genes
#  with one gene differentially expressed in first 3 arrays
M <- matrix(rnorm(100*6,sd=0.3),100,6)
M[1,1:3] <- M[1,1:3] + 2
#  Design matrix corresponds to oneway layout, columns are orthogonal
design <- cbind(First3Arrays=c(1,1,1,0,0,0),Last3Arrays=c(0,0,0,1,1,1))
fit <- lmFit(M,design=design)
#  Would like to consider original two estimates plus difference between first 3 and last
contrast.matrix <- cbind(First3=c(1,0),Last3=c(0,1),"Last3-First3"=c(-1,1))
fit2 <- contrasts.fit(fit,contrast.matrix)
fit2 <- eBayes(fit2)
#  Large values of eb$t indicate differential expression
results <- classifyTestsF(fit2)
vennCounts(results)
```

---

controlStatus *Set Status of each Spot from List of Spot Types*

Description

Determine the type (or status) of each spot in the gene list.

Usage

```
controlStatus(types, genes, spottypecol="SpotType", regexpcol, verbose=TRUE)
```

Arguments

types	dataframe containing spot type specifiers, usually input using readSpotTypes
genes	dataframe containing the microarray gene list, or an RGList, MAList or MArrayList containing genes
spottypecol	integer or name specifying column of types containing spot type names
regexpcol	vector of integers or column names specifying columns of types containing regular expressions. Defaults to any column names in common between types and genes.
verbose	logical, if TRUE then progess on pattern matching is reported to the standard output channel

Details

This function constructs a vector of status codes by searching for patterns in the gene list. The data frame genes contains gene IDs and should have as many rows as there are spots on the microarrays. Such a data frame is often read using readGAL. The data frame types has as many rows as you want to distinguish types of spots in the gene list. This data frame should contain a column or columns, the regexpcol columns, which have the same names as columns in genes and which contain patterns to match in the gene list. Another column, the spottypecol, contains the names of the spot types. Any other columns are assumed to contain plotting parameters, such as colors or symbols, to be associated with the spot types.

The patterns in the regexpcol columns are simplified regular expressions. For example, AA* means any string starting with AA, *AA means any code ending with AA, AA means exactly these

two letters, *AA* means any string containing AA, AA. means AA followed by exactly one other character and AA\. means exactly AA followed by a period and no other characters. Any other regular expressions are allowed but the codes ^ for beginning of string and $ for end of string should not be included.

Note that the patterns are matched sequentially from first to last, so more general patterns should be included first. For example, it is often a good idea to include a default spot-type as the first line in types with pattern * for all regexpcol columns and default plotting parameters.

Value

Character vector specifying the type (or status) of each spot on the array. Attributes contain plotting parameters associated with each spot type.

Author(s)

Gordon Smyth

See Also

An overview of LIMMA functions for reading data is given in 03.ReadingData.

Examples

```
genes <- data.frame(ID=c("Control","Control","Control","Control","AA1","AA2","AA3","AA4")
Name=c("Ratio 1","Ratio 2","House keeping 1","House keeping 2","Gene 1","Gene 2","Gene 3"
types <- data.frame(SpotType=c("Gene","Ratio","Housekeeping"),ID=c("*","Control","Control
status <- controlStatus(types,genes)
```

---

convest *Estimate Proportion of True Null Hypotheses*

Description

Returns an estimate of the proportion of true null hypotheses using a convex decreasing density estimate on a vector of p-values.

Usage

```
convest(p, niter = 100, doplot = FALSE, doreport = FALSE)
```

Arguments

p	numeric vector of p-values, calculated using any test of your choice. Missing values are not allowed
niter	number of iterations to be used in fitting the convex, decreasing density for the p-values. Default is 100.
doplot	logical, should updated plots of fitted convex decreasing p-value density be produced at each iteration? Default is FALSE.
doreport	logical, should the estimated proportion be printed at each iteration? Default is FALSE.

Details

The proportion of true null hypotheses is often denoted pi0.

Value

Numeric value in the interval [0,1] representing he estimated proportion of true null hypotheses.

Author(s)

Egil Ferkingstad and Mette Langaas

References

Ferkingstad, E., Langaas, M., and Lindqvist, B. (2005). Estimating the proportion of true null hypotheses, with application to DNA microarray data. *Journal of the Royal Statistical Society Series* B, 67, 555-572. Preprint at http://www.math.ntnu.no/~mettela/

See Also

See 08.Tests for other functions for producing or interpretting p-values.

Examples

```
# First simulate data, use no.genes genes and no.ind individuals,
# with given value of pi0. Draw from normal distribution with mean=0
# (true null) and mean=mean.diff (false null).

no.genes <- 1000
no.ind <- 20
pi0 <- 0.9
mean.diff <- 1
n1 <- round(pi0*no.ind*no.genes)
n2 <- round((1-pi0)*no.ind*no.genes)
x <- matrix(c(rnorm(n1,mean=0),rnorm(n2,mean=mean.diff)),ncol=no.ind,byrow=TRUE)

# calculate p-values using your favorite method, e.g.
pvals <- ebayes(lm.series(x))$p.value

# run the convex decreasing density estimator to estimate pi0
convest(pvals,niter=100,doplot=interactive())
```

---

decideTests *Multiple Testing Across Genes and Contrasts*

Description

Classify a series of related t-statistics as up, down or not significant. A number of different multiple testing schemes are offered which adjust for multiple testing down the genes as well as across contrasts for each gene.

Usage

```
decideTests(object,method="separate",adjust.method="BH",p.value=0.05,lfc=0)
```

Arguments

object	MArrayLM object output from eBayes from which the t-statistics may be extracted.
method	character string specify how probes and contrasts are to be combined in the multiple testing strategy. Choices are "separate", "global", "hierarchical", "nestedF" or any partial string.
adjust.method	character string specifying p-value adjustment method. Possible values are "none", "BH", "fdr" (equivalent to "BH"), "BY" and "holm". See p.adjust for details.
p.value	numeric value between 0 and 1 giving the desired size of the test
lfc	minimum log2-fold-change required

Details

These functions implement multiple testing procedures for determining whether each statistic in a matrix of t-statistics should be considered significantly different from zero. Rows of tstat correspond to genes and columns to coefficients or contrasts.

The setting method="separate" is equivalent to using topTable separately for each coefficient in the linear model fit, and will give the same lists of probes if adjust.method is the same. method="global" will treat the entire matrix of t-statistics as a single vector of unrelated tests. method="hierarchical" adjusts down genes and then across contrasts. method="nestedF" adjusts down genes and then uses classifyTestsF to classify contrasts as significant or not for the selected genes.

Value

An object of class TestResults. This is essentially a numeric matrix with elements -1, 0 or 1 depending on whether each t-statistic is classified as significantly negative, not significant or significantly positive respectively.

If lfc>0 then contrasts are judged significant only when the log2-fold change is at least this large in absolute value. For example, one might choose lfc=log2(1.5) to restrict to 50% changes or lfc=1 for 2-fold changes. In this case, contrasts must satisfy both the p-value and the fold-change cutoff to be judged significant.

Author(s)

Gordon Smyth

See Also

An overview of multiple testing functions is given in 08.Tests.

dim *Retrieve the Dimensions of an RGList, MAList or MArrayLM Object*

Description

Retrieve the number of rows (genes) and columns (arrays) for an RGList, MAList or MArrayLM object.

Usage

```
## S3 method for class 'RGList':
dim(x)
## S3 method for class 'RGList':
length(x)
```

Arguments

x	an object of class RGList, MAList or MArrayLM

Details

Microarray data objects share many analogies with ordinary matrices in which the rows correspond to spots or genes and the columns to arrays. These methods allow one to extract the size of microarray data objects in the same way that one would do for ordinary matrices.

A consequence is that row and column commands nrow(x), ncol(x) and so on also work.

Value

Numeric vector of length 2. The first element is the number of rows (genes) and the second is the number of columns (arrays).

Author(s)

Gordon Smyth

See Also

dim in the base package.

02.Classes gives an overview of data classes used in LIMMA.

Examples

```
M <- A <- matrix(11:14,4,2)
rownames(M) <- rownames(A) <- c("a","b","c","d")
colnames(M) <- colnames(A) <- c("A1","A2")
MA <- new("MAList",list(M=M,A=A))
dim(M)
ncol(M)
nrow(M)
length(M)
```

`dimnames` *Retrieve the Dimension Names of an RGList, MAList or MArrayLM Object*

Description

Retrieve the dimension names of a microarray data object.

Usage

```
## S3 method for class 'RGList':
dimnames(x)
## S3 replacement method for class 'RGList':
dimnames(x) <- value
```

Arguments

`x`	an object of class `RGList`, `MAList` or (not for assignment) `MArrayLM`
`value`	a possible value for `dimnames(x)`: see `dimnames`

Details

The dimension names of a microarray object are the same as those of the most important matrix component of that object.

A consequence is that `rownames` and `colnames` will work as expected.

Value

Either `NULL` or a list of length 2. If a list, its components are either 'NULL' or a character vector the length of the appropriate dimension of `x`.

Author(s)

Gordon Smyth

See Also

`dimnames` in the base package.

02.Classes gives an overview of data classes used in LIMMA.

dupcor *Correlation Between Duplicates*

Description

Estimate the correlation between duplicate spots (regularly spaced replicate spots on the same array) or between technical replicates from a series of arrays.

Usage

```
duplicateCorrelation(object,design=rep(1,ncol(as.matrix(object))),ndups=2,spacin
```

Arguments

object	a numeric matrix of expression values, or any data object from which as.matrix will extract a suitable matrix such as an MAList, marrayNorm or ExpressionSet object. If object is an MAList object then the arguments design, ndups, spacing and weights will be extracted from it if available and do not have to be specified as arguments. Specifying these arguments explicitly will over-rule any components found in the data object.
design	the design matrix of the microarray experiment, with rows corresponding to arrays and columns to comparisons to be estimated. The number of rows must match the number of columns of object. Defaults to the unit vector meaning that the arrays are treated as replicates.
ndups	a positive integer giving the number of times each gene is printed on an array. nrow(object) must be divisible by ndups. Will be ignored if block is specified.
spacing	the spacing between the rows of object corresponding to duplicate spots, spacing=1 for consecutive spots
block	vector or factor specifying a blocking variable
trim	the fraction of observations to be trimmed from each end of tanh(all.correlations) when computing the trimmed mean.
weights	an optional numeric matrix of the same dimension as object containing weights for each spot. If smaller than object then it will be filled out the same size.

Details

When block=NULL, this function estimates the correlation between duplicate spots (regularly spaced within-array replicate spots). If block is not null, this function estimates the correlation between repeated observations on the blocking variable. Typically the blocks are biological replicates and the repeated observations are technical replicates. In either case, the correlation is estimated by fitting a mixed linear model by REML individually for each gene. The function also returns a consensus correlation, which is a robust average of the individual correlations, which can be used as input for functions lmFit or gls.series.

At this time it is not possible to estimate correlations between duplicate spots and between technical replicates simultaneously. If block is not null, then the function will set ndups=1.

For this function to return statistically useful results, there must be at least two more arrays than the number of coefficients to be estimated, i.e., two more than the column rank of design.

The function may take long time to execute as it fits a mixed linear model for each gene for an iterative algorithm. It is not uncommon for the function to return a small number of warning messages that correlation estimates cannot be computed for some individual genes. This is not a serious concern providing that there are only a few such warnings and the total number of genes is large. The consensus estimator computed by this function will not be materially affected by a small number of genes.

Value

A list with components

`consensus.correlation`
: the average estimated inter-duplicate correlation. The average is the trimmed mean of the individual correlations on the atanh-transformed scale.

`cor`
: same as `consensus.correlation`, for compatibility with earlier versions of the software

`atanh.correlations`
: numeric vector of length `nrow(object)/ndups` giving the individual genewise atanh-transformed correlations.

Author(s)

Gordon Smyth

References

Smyth, G. K., Michaud, J., and Scott, H. (2005). The use of within-array replicate spots for assessing differential expression in microarray experiments. *Bioinformatics* 21(9), 2067-2075. http://www.statsci.org/smyth/pubs/dupcor.pdf

See Also

These functions use `mixedModel2Fit` from the statmod package.

An overview of linear model functions in limma is given by 06.LinearModels.

Examples

```
#  Also see lmFit examples

## Not run:
corfit <- duplicateCorrelation(MA, ndups=2, design)
all.correlations <- tanh(corfit$atanh.correlations)
boxplot(all.correlations)
fit <- lmFit(MA, design, ndups=2, correlation=corfit$consensus)
## End(Not run)
```

ebayes *Empirical Bayes Statistics for Differential Expression*

Description

Given a series of related parameter estimates and standard errors, compute moderated t-statistics, moderated F-statistic, and log-odds of differential expression by empirical Bayes shrinkage of the standard errors towards a common value.

Usage

```
ebayes(fit, proportion=0.01, stdev.coef.lim=c(0.1,4))
eBayes(fit, proportion=0.01, stdev.coef.lim=c(0.1,4))
treat(fit, lfc=0)
```

Arguments

fit	an MArrayLM fitted model object produced by lmFit or contrasts.fit, or an unclassed list produced by lm.series, gls.series or mrlm containing components coefficients, stdev.unscaled, sigma and df.residual
proportion	numeric value between 0 and 1, assumed proportion of genes which are differentially expressed
stdev.coef.lim	numeric vector of length 2, assumed lower and upper limits for the standard deviation of log2-fold-changes for differentially expressed genes
lfc	the minimum log2-fold-change which is considered material

Details

These functions is used to rank genes in order of evidence for differential expression. They use an empirical Bayes method to shrink the probe-wise sample variances towards a common value and to augmenting the degrees of freedom for the individual variances (Smyth, 2004). The functions accept as input argument fit a fitted model object from the functions lmFit, lm.series, mrlm or gls.series. The fitted model object may have been processed by contrasts.fit before being passed to eBayes to convert the coefficients of the design matrix into an arbitrary number of contrasts which are to be tested equal to zero. The columns of fit define a set of contrasts which are to be tested equal to zero.

The empirical Bayes moderated t-statistics test each individual contrast equal to zero. For each probe (row), the moderated F-statistic tests whether all the contrasts are zero. The F-statistic is an overall test computed from the set of t-statistics for that probe. This is exactly analogous the relationship between t-tests and F-statistics in conventional anova, except that the residual mean squares and residual degrees of freedom have been moderated between probes.

The estimates s2.prior and df.prior are computed by fitFDist. s2.post is the weighted average of s2.prior and sigma^2 with weights proportional to df.prior and df.residual respectively. The lods is sometimes known as the B-statistic. The F-statistics F are computed by classifyTestsF with fstat.only=TRUE.

eBayes doesn't compute ordinary (unmoderated) t-statistics by default, but these can be easily extracted from the linear model output, see the example below.

ebayes is the earlier and leaner function. eBayes is intended to have a more object-orientated flavor as it produces objects containing all the necessary components for downstream analysis.

treat computes empirical Bayes moderated-t p-values relative to a minimum required fold-change threshold. Instead of testing for genes which have log-fold-changes different from zero, it tests whether the log2-fold-change is greater than lfc in absolute value (McCarthy and Smyth, 2009). treat is concerned with p-values rather than posterior odds, so it does not compute the B-statistic lods. The idea of thresholding doesn't apply to F-statistics in a straightforward way, so moderated F-statistics are also not computed.

Value

eBayes produces an object of class MArrayLM with the following components, see MArrayLM-class. ebayes produces an ordinary list without F or F.p.value. treat produces an MArrayLM object, but without lods, var.prior, F or F.p.value.

t	numeric vector or matrix of moderated t-statistics
p.value	numeric vector of p-values corresponding to the t-statistics
s2.prior	estimated prior value for sigma^2
df.prior	degrees of freedom associated with s2.prior
s2.post	vector giving the posterior values for sigma^2
lods	numeric vector or matrix giving the log-odds of differential expression
var.prior	estimated prior value for the variance of the log2-fold-change for differentially expressed gene
F	numeric vector of moderated F-statistics for testing all contrasts defined by the columns of fit simultaneously equal to zero
F.p.value	numeric vector giving p-values corresponding to F

Author(s)

Gordon Smyth

References

McCarthy, D. J., and Smyth, G. K. (2009). Testing significance relative to a fold-change threshold is a TREAT. *Bioinformatics*. http://bioinformatics.oxfordjournals.org/cgi/content/abstract/btp053

Loennstedt, I., and Speed, T. P. (2002). Replicated microarray data. *Statistica Sinica* **12**, 31-46.

Smyth, G. K. (2004). Linear models and empirical Bayes methods for assessing differential expression in microarray experiments. *Statistical Applications in Genetics and Molecular Biology*, Volume **3**, Article 3. http://www.bepress.com/sagmb/vol3/iss1/art3

See Also

squeezeVar, fitFDist, tmixture.matrix.

An overview of linear model functions in limma is given by 06.LinearModels.

Examples

```
#  See also lmFit examples

#  Simulate gene expression data,
#  6 microarrays and 100 genes with one gene differentially expressed
set.seed(2004); invisible(runif(100))
M <- matrix(rnorm(100*6,sd=0.3),100,6)
M[1,] <- M[1,] + 1
fit <- lmFit(M)

#  Ordinary t-statistic
par(mfrow=c(1,2))
ordinary.t <- fit$coef / fit$stdev.unscaled / fit$sigma
qqt(ordinary.t,df=fit$df.residual,main="Ordinary t")
abline(0,1)

#  Moderated t-statistic
eb <- eBayes(fit)
qqt(eb$t,df=eb$df.prior+eb$df.residual,main="Moderated t")
abline(0,1)
#  Points off the line may be differentially expressed
par(mfrow=c(1,1))
```

---

exprs.MA *Extract Log-Expression Matrix from MAList*

Description

Extract the matrix of log-expression values from an MAList object.

Usage

```
exprs.MA(MA)
```

Arguments

MA	an MAList object.

Details

Converts M and A-values to log-expression values. The output matrix will have two columns for each array, in the order green, red for each array.

This contrasts with as.matrix.MAList which extracts the M-values only, or RG.MA which converts to expression values in RGList form.

Value

A numeric matrix with twice the columns of the input.

Author(s)

Gordon Smyth

See Also

02.Classes gives an overview of data classes used in LIMMA.

`fitFDist` *Moment Estimation of Scaled F-Distribution*

Description

Moment estimation of the parameters of a scaled F-distribution given one of the degrees of freedom. This function is called internally by `ebayes` and is not usually called directly by a user.

Usage

```
fitFDist(x,df1)
```

Arguments

`x`	numeric vector or array of positive values representing a sample from an F-distribution.
`df1`	the first degrees of freedom of the F-distribution. May be an integer or a vector of the same length as `x`.

Details

The function estimates `scale` and `df2` under the assumption that `x` is distributed as `scale` times an F-distributed random variable on `df1` and `df2` degrees of freedom.

Value

A list containing the components

`scale`	scale factor for F-distribution
`df2`	the second degrees of freedom of the F-distribution

Author(s)

Gordon Smyth

See Also

`ebayes`, `trigammaInverse`

fitted.MArrayLM *Fitted Values Method for MArrayLM Fits*

Description

Obtains fitted values from a fitted microarray linear model object.

Usage

```
## S3 method for class 'MArrayLM':
fitted(object, design = object$design, ...)
```

Arguments

object	a fitted object of class inheriting from "MArrayLM".
design	numeric design matrix.
...	further arguments passed to or from other methods.

Value

A numeric matrix of fitted values.

Author(s)

Gordon Smyth

See Also

fitted

geneSetTest *Gene Set Test*

Description

Test whether a set of genes is enriched for differential expression. Genes are assumed to be independent.

Usage

```
geneSetTest(selected,statistics,alternative="mixed",type="auto",ranks.only=TRUE,
barcodeplot(selected,statistics,type="auto",...)
```

Arguments

selected	vector specifying the elements of statistic in the test group. This can be a vector of indices, or a logical vector of the same length as statistics, or any vector such as statistic[selected] contains the statistic values for the selected group.
statistics	numeric vector giving the values of the test statistic for every gene or probe in the reference set, usually every probe on the microarray.
alternative	character string specifying the alternative hypothesis, must be one of "mixed" (default), "either", "up" or "down". two.sided, "greater" and "less" are also permitted as synonyms for "either", "up" and "down" respectively.
type	character string specifying whether the statistics are t-like ("t"), F-like "f" or whether the function should make an educated guess ("auto")
ranks.only	logical, if TRUE only the ranks of the statistics are used.
nsim	number of random samples to take in computing the p-value. Not used if ranks.only=TRUE.
...	other arguments are passed to plot.

Details

This function computes a p-value to test the hypothesis that the selected set of genes tends to be more highly ranked in differential expression compared to randomly selected genes. This function can be used to detect differential expression for a group of genes, even when the effects are too small or there is too little data to detect the genes individually. It is also provides a means to compare the results between different experiments.

The mean-rank gene set test of Michaud et al (2008) corresponds to geneSetTest with ranks.only=TRUE.

geneSetTest is a competitive test in the sense that genes in the test set are compared to other genes (Goeman and Buhlmann, 2007). A self-contained gene set test is performed by roast.

Because it is based on permuting genes, geneSetTest assumes that the different genes (or probes) are independent. (Strictly speaking, it assumes that the genes in the set are no more correlated on average than randomly selected genes.) This assumption may be reasonable if the gene set is relatively small and if there is relatively little genotypic variation in the data, for example if the data is obtained from genetically identical inbred mice. The independence assumption may be misleading if the gene set is large or if the data contains a lot of genotypic variation, for example for human cancer samples. These assumptions, when valid, permit a much quicker and more powerful significance test to be conducted.

The statistics are usually a set of probe-wise statistics arising for some comparison from a microarray experiment. They may be t-statistics, meaning that the genewise null hypotheses would be rejected for large positive or negative values, or they may be F-statistics, meaning that only large values are significant. Any set of signed statistics, such as log-ratios, M-values or moderated t-statistics, are treated as t-like. Any set of unsigned statistics, such as F-statistics, posterior probabilities or chi-square tests are treated as F-like. If type="auto" then the statistics will be taken to be t-like if they take both positive and negative values and otherwise will be taken to be F-like.

There are four possible alternatives to test for. alternative=="up" means the genes in the set tend to be up-regulated, with positive t-statistics. alternative=="down" means the genes in the set tend to be down-regulated, with negative t-statistics. alternative=="either" means the set is either up or down-regulated as a whole. alternative=="mixed" test whether the genes in the set tend to be differentially expressed, without regard for direction. In this case, the test will be significant if the set contains mostly large test statistics, even if some are positive and some are negative.

The latter three alternatives are appropriate if you have a prior expection that all the genes in the set will react in the same direction. The "mixed" alternative is appropriate if you know only that the genes are involved in the relevant pathways, possibly in different directions. The "mixed" alternative is the only one possible with F-like statistics.

The test statistic used for the gene-set-test is the mean of the statistics in the set. If ranks.only is TRUE the only the ranks of the statistics are used. In this case the p-value is obtained from a Wilcoxon test. If ranks.only is FALSE, then the p-value is obtained by simulation using nsim random selected sets of genes.

The barcodeplot is a graphical representation of the gene set test using ranks.

Value

Numeric value giving the estimated p-value.

Author(s)

Gordon Smyth

References

Goeman, JJ, and Buhlmann P (2007). Analyzing gene expression data in terms of gene sets: methodological issues. *Bioinformatics* 23, 980-987.

Michaud, J, Simpson, KM, Escher, R, Buchet-Poyau, K, Beissbarth, T, Carmichael, C, Ritchie, ME, Schutz, F, Cannon, P, Liu, M, Shen, X, Ito, Y, Raskind, WH, Horwitz, MS, Osato, M, Turner, DR, Speed, TP, Kavallaris, M, Smyth, GK, and Scott, HS (2008). Integrative analysis of RUNX1 downstream pathways and target genes. *BMC Genomics* 9, 363. http://www.biomedcentral.com/1471-2164/9/363

See Also

roast, wilcox.test

Examples

```
stat <- -9:9
sel <- c(2,4,5)
geneSetTest(sel,stat,alternative="down")
geneSetTest(sel,stat,alternative="either")
geneSetTest(sel,stat,alternative="down",ranks=FALSE)
sel <- c(1,19)
geneSetTest(sel,stat,alternative="mixed")
geneSetTest(sel,stat,alternative="mixed",ranks=FALSE)
```

---

getEAWP *Extract Basic Data from Microarray Data Objects*

Description

Given a microarray data object of any known class, get the expression values, weights, probe annotation and A-values, which are needed for linear modelling. This function is called by the linear modelling functions in LIMMA.

Usage

```
getEAWP(object)
```

Arguments

object	a microarray data object. An object of class list, MAList, marrayNorm, PLMset, vsn, or any class inheriting from ExpressionSet, or any object that can be coerced to a numeric matrix.

Details

In the case of two-color objects, the Amean is computed from the matrix of A-values. For single-channel objects, Amean is computed from the matrix of expression vales. PLMset, vsn and ExpressionSet are assumed to be single-channel for this purpose.

If object is a matrix, it is assumed to contain log-intensities if the values are all positive and log-ratios otherwise. Amean is computed in the former case but not the latter.

Value

A list with components

exprs	numeric matrix of log-ratios or log-intensities
weights	numeric matrix of weights
probes	data.frame of probe-annotation
Amean	numeric vector of average log-expression for each probe

Author(s)

Gordon Smyth

See Also

02.Classes gives an overview of data classes used in LIMMA.

---

getSpacing *Get Numerical Spacing*

Description

Convert character to numerical spacing measure for within-array replicate spots.

Usage

```
getSpacing(spacing, layout)
```

Arguments

spacing	character string or integer. Acceptable character strings are "columns", "rows", "subarrays" or "topbottom". Integer values are simply passed through.
layout	list containing printer layout information

Details

"rows" means that duplicate spots are printed side-by-side by rows. These will be recorded in consecutive rows in the data object.

"columns" means that duplicate spots are printed side-by-sidy by columns. These will be separated in the data object by layout$nspot.r rows.

"subarrays" means that a number of sub-arrays, with identical probes in the same arrangement, are printed on each array. The spacing therefore will be the size of a sub-array.

"topbottom" is the same as "subarrays" when there are two sub-arrays.

Value

Integer giving spacing between replicate spots in the gene list.

Author(s)

Gordon Smyth

See Also

An overview of LIMMA functions for reading data is given in 03.ReadingData.

Examples

```
getSpacing("columns",list(ngrid.r=2,ngrid.c=2,nspot.r=20,nspot.c=19))
getSpacing("rows",list(ngrid.r=2,ngrid.c=2,nspot.r=20,nspot.c=19))
getSpacing("topbottom",list(ngrid.r=2,ngrid.c=2,nspot.r=20,nspot.c=19))
```

---

getLayout *Extract the Print Layout of an Array from the GAL File*

Description

From the Block, Row and Column information in a genelist, determine the number of grid rows and columns on the array and the number of spot rows and columns within each grid.

Usage

```
getLayout(gal, guessdups=FALSE)
getLayout2(galfile)
getDupSpacing(ID)
```

Arguments

gal	data.frame containing the GAL, i.e., giving the position and gene identifier of each spot
galfile	name or path of GAL file
guessdups	logical, if TRUE then try to determine number and spacing of duplicate spots, i.e., within-array replicates
ID	vector or factor of gene IDs

Details

A GenePix Array List (GAL) file is a list of genes and associated information produced by an Axon microarray scanner. The function `getLayout` determines the print layout from a data frame created from a GAL file or gene list. The data.frame must contain columns `Block`, `Column` and `Row`. (The number of tip columns is assumed to be either one or four.)

On some arrays, each probe may be duplicated a number of times (`ndups`) at regular intervals (`spacing`) in the GAL file. `getDupSpacing` determines valid values for `ndups` and `spacing` from a vector of IDs. If `guessdups=TRUE`, then `getLayout` calls `getDupSpacing`.

The function `getLayout2` attempts to determine the print layout from the header information of an actual GAL file.

Value

A `printlayout` object, which is a list with the following components. The last two components are present only if `guessdups=TRUE`.

`ngrid.r`	integer, number of grid rows on the arrays
`ngrid.c`	integer, number of grid columns on the arrays
`nspot.r`	integer, number of rows of spots in each grid
`nspot.c`	integer, number of columns of spots in each grid
`ndups`	integer, number of times each probe is printed on the array
`spacing`	integer, spacing between multiple printings of each probe

Author(s)

Gordon Smyth and James Wettenhall

See Also

`gpTools`.

An overview of LIMMA functions for reading data is given in 03.ReadingData.

Examples

```
# gal <- readGAL()
# layout <- getLayout(gal)
```

---

gls.series *Fit Linear Model to Microarray Data by Generalized Least Squares*

Description

Fit a linear model genewise to expression data from a series of microarrays. The fit is by generalized least squares allowing for correlation between duplicate spots or related arrays. This is a utility function for `lmFit`.

Usage

```
gls.series(M,design=NULL,ndups=2,spacing=1,block=NULL,correlation=NULL,weights=N
```

Arguments

M	numeric matrix containing log-ratio or log-expression values for a series of microarrays, rows correspond to genes and columns to arrays.
design	numeric design matrix defining the linear model, with rows corresponding to arrays and columns to comparisons to be estimated. The number of rows must match the number of columns of M. Defaults to the unit vector meaning that the arrays are treated as replicates.
ndups	positive integer giving the number of times each gene is printed on an array. nrow(M) must be divisible by ndups.
spacing	the spacing between the rows of M corresponding to duplicate spots, spacing=1 for consecutive spots
block	vector or factor specifying a blocking variable on the arrays. Same length as ncol(M).
correlation	numeric value specifying the inter-duplicate or inter-block correlation.
weights	an optional numeric matrix of the same dimension as M containing weights for each spot. If it is of different dimension to M, it will be filled out to the same size.
...	other optional arguments to be passed to dupcor.series.

Details

This is a utility function used by the higher level function lmFit. Most users should not use this function directly but should use lmFit instead.

This function is for fitting gene-wise linear models when some of the expression values are correlated. The correlated groups may arise from replicate spots on the same array (duplicate spots) or from a biological or technical replicate grouping of the arrays. This function is normally called by lmFit and is not normally called directly by users.

Note that the correlation is assumed to be constant across genes. If correlation=NULL then a call is made to duplicateCorrelation to estimated the correlation.

Value

A list with components

coefficients	numeric matrix containing the estimated coefficients for each linear model. Same number of rows as M, same number of columns as design.
stdev.unscaled	numeric matrix conformal with coef containing the unscaled standard deviations for the coefficient estimators. The standard errors are given by stdev.unscaled * sigma.
sigma	numeric vector containing the residual standard deviation for each gene.
df.residual	numeric vector giving the degrees of freedom corresponding to sigma
correlation	inter-duplicate or inter-block correlation
qr	QR decomposition of the generalized linear squares problem, i.e., the decomposition of design standardized by the Choleski-root of the correlation matrix defined by correlation

Author(s)

Gordon Smyth

See Also

duplicateCorrelation.

An overview of linear model functions in limma is given by 06.LinearModels.

`gridr` *Row and Column Positions on Microarray*

Description

Grid and spot row and column positions.

Usage

```
gridr(layout)
gridc(layout)
spotr(layout)
spotc(layout)
```

Arguments

`layout`	list with the components `ngrid.r`, `ngrid.c`, `nspot.r` and `nspot.c`

Value

Vector of length `prod(unlist(layout))` giving the grid rows (`gridr`), grid columns (`gridc`), spot rows (`spotr`) or spot columns (`spotc`).

Author(s)

Gordon Smyth

`heatdiagram` *Stemmed Heat Diagram*

Description

Creates a heat diagram showing the co-regulation of genes under one condition with a range of other conditions.

Usage

```
heatDiagram(results,coef,primary=1,names=NULL,treatments=colnames(coef),limit=NU
heatdiagram(stat,coef,primary=1,names=NULL,treatments=colnames(stat),critical.pr
```

Arguments

results	TestResults matrix, containing elements -1, 0 or 1, from decideTests
stat	numeric matrix of test statistics. Rows correspond to genes and columns to treatments or contrasts between treatments.
coef	numeric matrix of the same size as stat. Holds the coefficients to be displayed in the plot.
primary	number or name of the column to be compared to the others. Genes are included in the diagram according to this column of stat and are sorted according to this column of coef. If primary is a name, then stat and coef must have the same column names.
names	optional character vector of gene names
treatments	optional character vector of treatment names
critical.primary	critical value above which the test statistics for the primary column are considered significant and included in the plot
critical.other	critical value above which the other test statistics are considered significant. Should usually be no larger than critical.primary although larger values are permitted.
limit	optional value for coef above which values will be plotted in extreme color. Defaults to max(abs(coef)).
orientation	"portrait" for upright plot or "landscape" for plot orientated to be wider than high. "portrait" is likely to be appropriate for inclusion in printed document while "landscape" may be appropriate for a presentation on a computer screen.
low	color associated with repressed gene regulation
high	color associated with induced gene regulation
ncolors	number of distinct colors used for each of up and down regulation
cex	factor to increase or decrease size of column and row text
mar	numeric vector of length four giving the size of the margin widths. Default is cex*c(5,6,1,1) for landscape and cex*c(1,1,4,3) for portrait.
...	any other arguments will be passed to the image function

Details

Users are encouraged to use heatDiagram rather than heatdiagram as the later function may be removed in future versions of limma.

This function plots an image of gene expression profiles in which rows (or columns for portrait orientation) correspond to treatment conditions and columns (or rows) correspond to genes. Only genes which are significantly differentially expressed in the primary condition are included. Genes are sorted by differential expression under the primary condition.

Note: the plot produced by this function is unique to the limma package. It should not be confused with "heatmaps" often used to display results from cluster analyses.

Value

An image is created on the current graphics device. A matrix with named rows containing the coefficients used in the plot is also invisibly returned.

Author(s)

Gordon Smyth

See Also

image.

Examples

```
library(sma)
data(MouseArray)
MA <- normalizeWithinArrays(mouse.data,layout=mouse.setup)
design <- cbind(c(1,1,1,0,0,0),c(0,0,0,1,1,1))
fit <- lmFit(MA,design=design)
contrasts.mouse <- cbind(Control=c(1,0),Mutant=c(0,1),Difference=c(-1,1))
fit <- eBayes(contrasts.fit(fit,contrasts=contrasts.mouse))
results <- decideTests(fit,method="global",p=0.1)
heatDiagram(results,fit$coef,primary="Difference")
```

---

helpMethods *Prompt for Method Help Topics*

Description

For any S4 generic function, find all methods defined in currently loaded packages. Prompt the user to choose one of these to display the help document.

Usage

```
helpMethods(genericFunction)
```

Arguments

genericFunction	a generic function or a character string giving the name of a generic function

Author(s)

Gordon Smyth

See Also

showMethods

Examples

```
## Not run: helpMethods(show)
```

imageplot *Image Plot of Microarray Statistics*

Description

Creates an image of colors or shades of gray that represent the values of a statistic for each spot on a spotted microarray. This function can be used to explore any spatial effects across the microarray.

Usage

```
imageplot(z, layout, low = NULL, high = NULL, ncolors = 123, zerocenter = NULL,
zlim = NULL, mar=c(2,1,1,1), legend=TRUE, ...)
```

Arguments

z	numeric vector or array. This vector can contain any spot statistics, such as log intensity ratios, spot sizes or shapes, or t-statistics. Missing values are allowed and will result in blank spots on the image. Infinite values are not allowed.
layout	a list specifying the dimensions of the spot matrix and the grid matrix.
low	color associated with low values of z. May be specified as a character string such as "green", "white" etc, or as a rgb vector in which c(1,0,0) is red, c(0,1,0) is green and c(0,0,1) is blue. The default value is "green" if zerocenter=T or "white" if zerocenter=F.
high	color associated with high values of z. The default value is "red" if zerocenter=T or "blue" if zerocenter=F.
ncolors	number of color shades used in the image including low and high.
zerocenter	should zero values of z correspond to a shade exactly halfway between the colors low and high? The default is TRUE if z takes positive and negative values, otherwise FALSE.
zlim	numerical vector of length 2 giving the extreme values of z to associate with colors low and high. By default zlim is the range of z. Any values of z outside the interval zlim will be truncated to the relevant limit.
mar	numeric vector of length 4 specifying the width of the margin around the plot. This argument is passed to par.
legend	logical, if TRUE the range of z and zlim is shown in the bottom margin
...	any other arguments will be passed to the function image

Details

This function may be used to plot the values of any spot-specific statistic, such as the log intensity ratio, background intensity or a quality measure such as spot size or shape. The image follows the layout of an actual microarray slide with the bottom left corner representing the spot (1,1,1,1). The color range is used to represent the range of values for the statistic. When this function is used to plot the red/green log-ratios, it is intended to be an in silico version of the classic false-colored red-yellow-green image of a scanned two-color microarray.

This function is related to the earlier plot.spatial function in the sma package and to the later maImage function in the marray package. It differs from plot.spatial most noticeably in that all the spots are plotted and the image is plotted from bottom left rather than from top left. It is

intended to display spatial patterns and artefacts rather than to highlight only the extreme values as does `plot.spatial`. It differs from `maImage` in that any statistic may be plotted and in its use of a red-yellow-green color scheme for log-ratios, similar to the classic false-colored jpeg image, rather than the red-black-green color scheme associated with heat maps.

Value

An plot is created on the current graphics device.

Author(s)

Gordon Smyth

See Also

`maImage`, `image`.

An overview of diagnostic functions available in LIMMA is given in 09.Diagnostics.

Examples

```
M <- rnorm(8*4*16*16)
imageplot(M,layout=list(ngrid.r=8,ngrid.c=4,nspot.r=16,nspot.c=16))
```

---

imageplot3by2 *Write Imageplots to Files*

Description

Write imageplots to files in PNG format, six plots to a file in a 3 by 2 grid arrangement.

Usage

```
imageplot3by2(RG, z="Gb", prefix=paste("image",z,sep="-"), path=NULL, zlim=NULL,
```

Arguments

`RG`	an `RGList` or `MAList` object, or any list with component named by `z`
`z`	character string giving name of component of `RG` to plot
`prefix`	character string giving prefix to attach to file names
`path`	character string specifying directory for output files
`zlim`	numeric vector of length 2, giving limits of response vector to be associated with saturated colors
`common.lim`	logical, should all plots on a page use the same axis limits
`...`	any other arguments are passed to `imageplot`

Details

At the time of writing, this function writes plots in PNG format in an arrangement optimized for A4-sized paper.

Value

No value is returned, but one or more files are written to the working directory. The number of files is determined by the number of columns of RG.

Author(s)

Gordon Smyth

See Also

An overview of diagnostic functions available in LIMMA is given in 09.Diagnostics.

---

intraspotCorrelation

Intra-Spot Correlation for Two Color Data

Description

Estimate the within-block correlation associated with spots for spotted two color microarray data.

Usage

```
intraspotCorrelation(object, design, trim=0.15)
```

Arguments

object	an MAList object or a list from which M and A values may be extracted
design	a numeric matrix containing the design matrix for linear model in terms of the individual channels. The number of rows should be twice the number of arrays. The number of columns will determine the number of coefficients estimated for each gene.
trim	the fraction of observations to be trimmed from each end of the atanh-correlations when computing the consensus correlation. See mean.

Details

This function estimates the correlation between two channels observed on each spot. The correlation is estimated by fitting a heteroscedastic regression model to the M and A-values of each gene. The function also returns a consensus correlation, which is a robust average of the individual correlations, which can be used as input for functions lmscFit.

The function may take long time to execute.

Value

A list with components

consensus.correlation
: robust average of the estimated inter-duplicate correlations. The average is the trimmed mean of the correlations for individual genes on the atanh-transformed scale.

`atanh.correlations`
: a numeric vector giving the individual genewise correlations on the atanh scale

`df`
: numeric matrix of degrees of freedom associated with the correlations. The first column gives the degrees of freedom for estimating the within-spot or M-value mean square while the second gives the degrees of freedom for estimating the between spot or A-value mean square.

Author(s)

Gordon Smyth

See Also

This function uses `remlscore` from the statmod package.

An overview of methods for single channel analysis in limma is given by 07.SingleChannel.

Examples

```
#  See lmscFit
## Not run:
corfit <- dupclicateCorrelation(MA, design)
all.correlations <- tanh(corfit$atanh.correlations)
boxplot(all.correlations)
## End(Not run)
```

`is.fullrank` *Check for Full Column Rank*

Description

Test whether a numeric matrix has full column rank.

Usage

```
is.fullrank(x)
nonEstimable(x)
```

Arguments

`x`
: a numeric matrix or vector

Details

`is.fullrank` is used to check the integrity of design matrices in limma, for example after subsetting operations.

`nonEstimable` is used by `lmFit` to report which coefficients in a linear model cannot be estimated.

Value

`is.fullrank` returns `TRUE` or `FALSE`.

`nonEstimable` returns a character vector of names for the columns of `x` which are linearly dependent on previous columns. If `x` has full column rank, then the value is `NULL`.

Author(s)

Gordon Smyth

Examples

```
# TRUE
is.fullrank(1)
is.fullrank(cbind(1,0:1))

# FALSE
is.fullrank(0)
is.fullrank(matrix(1,2,2))
nonEstimable(matrix(1,2,2))
```

---

isNumeric *Test for Numeric Argument*

Description

Test whether argument is numeric or a data.frame with numeric columns.

Usage

```
isNumeric(x)
```

Arguments

x	any object

Details

This function is used to check the validity of arguments for numeric functions. It is an attempt to emulate the behavior of internal generic math functions.

`isNumeric` differs from `is.numeric` in that data.frames with all columns numeric are accepted as numeric.

Value

`TRUE` or `FALSE`

Author(s)

Gordon Smyth

See Also

`is.numeric`, `Math`

Examples

```
isNumeric(3)
isNumeric("a")
x <- data.frame(a=c(1,1),b=c(0,1))
isNumeric(x)    # TRUE
is.numeric(x)   # FALSE
```

---

kooperberg *Kooperberg Model-Based Background Correction*

Description

This function uses a Bayesian model to background correct GenePix microarray data.

Usage

```
kooperberg(RG, a=TRUE, layout=RG$printer, verbose=TRUE)
```

Arguments

RG	an RGList of GenePix data, read in using `read.maimages`, with `other.columns=c("F635 SD","B635 SD","F532 SD","B532 SD","B532 Mean","B635 Mean","F Pixels","B Pixels")`.
a	logical. If `TRUE`, the 'a' parameters in the model (equation 3 and 4) are estimated for each slide. If `FALSE` the 'a' parameters are set to unity.
layout	list containing print layout with components `ngrid.r`, `ngrid.c`, `nspot.r` and `nspot.c`. Defaults to `RG$printer`.
verbose	logical. If `TRUE`, progress is reported to standard output.

Details

This function is for use with GenePix data and is designed to cope with the problem of large numbers of negative intensities and hence missing values on the log-intensity scale. It avoids missing values in most cases and at the same time dampens down the variability of log-ratios for low intensity spots. See Kooperberg et al (2002) for more details.

`kooperberg` uses the foreground and background intensities, standard deviations and number of pixels to compute empirical estimates of the model parameters as described in equation 2 of Kooperberg et al (2002).

Value

An `RGList` containing the components

R	matrix containing the background adjusted intensities for the red channel for each spot for each array
G	matrix containing the background adjusted intensities for the green channel for each spot for each array
printer	list containing print layout

Author(s)

Matthew Ritchie

References

Kooperberg, C., Fazzio, T. G., Delrow, J. J., and Tsukiyama, T. (2002) Improved background correction for spotted DNA microarrays. *Journal of Computational Biology* **9**, 55-66.

See Also

04.Background gives an overview of background correction functions defined in the LIMMA package.

Examples

```
#  This is example code for reading and background correcting GenePix data
#  given GenePix Results (gpr) files in the working directory (data not
#  provided).
## Not run:
genepixFiles <- dir(pattern="*\\.gpr$") # get the names of the GenePix image analysis out
RG <- read.maimages(genepixFiles, source="genepix", other.columns=c("F635 SD","B635 SD","
RGmodel <- kooperberg(RG)
MA <- normalizeWithinArrays(RGmodel)
## End(Not run)
```

---

limmaUsersGuide *View Limma User's Guide*

Description

Finds the location of the Limma User's Guide and optionally opens it.

Usage

```
limmaUsersGuide(view=TRUE)
```

Arguments

view	logical, should the document be opened using the default PDF document reader?

Details

The function `vignette("limma")` will find the short limma Vignette which describes how to obtain the Limma User's Guide. The User's Guide is not itself a true vignette because it is not automatically generated using `Sweave` during the package build process. This means that it cannot be found using `vignette`, hence the need for this special function.

If the operating system is other than Windows, then the PDF viewer used is that given by `Sys.getenv("R_PDFVIEWE`
The PDF viewer can be changed using `Sys.putenv(R_PDFVIEWER=)`.

This function is used by drop-down Vignettes menu when the Rgui interface for Windows is used.

Value

Character string giving the file location.

Author(s)

Gordon Smyth

See Also

vignette, openPDF, openVignette, Sys.getenv, Sys.putenv

Examples

```
limmaUsersGuide(view=FALSE)
```

---

lm.series *Fit Linear Model to Microrray Data by Ordinary Least Squares*

Description

Fit a linear model genewise to expression data from a series of arrays. This function uses ordinary least squares and is a utility function for lmFit.

Usage

```
lm.series(M,design=NULL,ndups=1,spacing=1,weights=NULL)
```

Arguments

M	numeric matrix containing log-ratio or log-expression values for a series of microarrays, rows correspond to genes and columns to arrays
design	numeric design matrix defining the linear model. The number of rows should agree with the number of columns of M. The number of columns will determine the number of coefficients estimated for each gene.
ndups	number of duplicate spots. Each gene is printed ndups times in adjacent spots on each array.
spacing	the spacing between the rows of M corresponding to duplicate spots, spacing=1 for consecutive spots
weights	an optional numeric matrix of the same dimension as M containing weights for each spot. If it is of different dimension to M, it will be filled out to the same size.

Details

This is a utility function used by the higher level function lmFit. Most users should not use this function directly but should use lmFit instead.

The linear model is fit for each gene by calling the function lm.fit or lm.wfit from the base library.

Value

A list with components

coefficients	numeric matrix containing the estimated coefficients for each linear model. Same number of rows as M, same number of columns as design.
stdev.unscaled	numeric matrix conformal with coef containing the unscaled standard deviations for the coefficient estimators. The standard errors are given by stdev.unscaled * sigma.
sigma	numeric vector containing the residual standard deviation for each gene.
df.residual	numeric vector giving the degrees of freedom corresponding to sigma.
qr	QR-decomposition of design

Author(s)

Gordon Smyth

See Also

lm.fit.

An overview of linear model functions in limma is given by 06.LinearModels.

---

lmFit *Linear Model for Series of Arrays*

Description

Fit linear model for each gene given a series of arrays

Usage

```
lmFit(object,design=NULL,ndups=1,spacing=1,block=NULL,correlation,weights=NULL,m
```

Arguments

object	object of class numeric, matrix, MAList, marrayNorm, ExpressionSet or PLMset containing log-ratios or log-values of expression for a series of microarrays
design	the design matrix of the microarray experiment, with rows corresponding to arrays and columns to coefficients to be estimated. Defaults to the unit vector meaning that the arrays are treated as replicates.
ndups	positive integer giving the number of times each gene is printed on an array
spacing	positive integer giving the spacing between duplicate spots, spacing=1 for consecutive spots
block	vector or factor specifying a blocking variable on the arrays. Has length equal to the number of arrays. Must be NULL if ndups>2.
correlation	the inter-duplicate or inter-technical replicate correlation
weights	optional numeric matrix containing weights for each spot
method	character string, "ls" for least squares or "robust" for robust regression
...	other optional arguments to be passed to lm.series, gls.series or mrlm

Details

This function fits multiple linear models. It accepts data from a experiment involving a series of microarrays with the same set of probes. A linear model is fitted to the expression data for each probe. The expression data should be log-ratios for two-color array platforms or log-expression values for one-channel platforms. (To fit linear models to the individual channels of two-color array data, see `lmscFit`.) The coefficients of the fitted models describe the differences between the RNA sources hybridized to the arrays. The probe-wise fitted model results are stored in a compact form suitable for further processing by other functions in the limma package.

The function allows for missing values and accepts quantitative weights through the `weights` argument. It also supports two different correlation structures. If `block` is not `NULL` then different arrays are assumed to be correlated. If `block` is `NULL` and `ndups` is greater than one then replicate spots on the same array are assumed to be correlated. It is not possible at this time to fit models with both a block structure and a duplicate-spot correlation structure simultaneously.

If `object` is a matrix then it should contain log-ratios or log-expression data with rows corresponding to probes and columns to arrays. (A numeric vector is treated the same as a matrix with one column.) For objects of other classes, a matrix of expression values is taken from the appropriate component or slot of the object. If `object` is of class `MAList` or `marrayNorm`, then the matrix of log-ratios (M-values) is extracted. If `object` is of class `ExpressionSet`, then the expression matrix is extracted. (This may contain log-expression or log-ratio values, depending on the platform.) If `object` is of class `PLMset` then the matrix of chip coefficients `chip.coefs` is extracted.

The arguments `design`, `ndups`, `spacing` and `weights` will be extracted from the data `object` if available and do not normally need to set explicitly in the call. On the other hand, if any of these are set in the function call then they will over-ride the slots or components in the data `object`. If `object` is an `PLMset`, then weights are computed as `1/pmax(object@se.chip.coefs, 1e-05)^2`. If `object` is an `ExpressionSet` object, then weights are not computed.

If the argument `block` is used, then it is assumed that `ndups=1`.

The `correlation` argument has a default value of `0.75`, but in normal use this default value should not be relied on and the correlation value should be estimated using the function `duplicateCorrelation`. The default value is likely to be too high in particular if used with the `block` argument.

The actual linear model computations are done by passing the data to one the lower-level functions `lm.series`, `gls.series` or `mrlm`. The function `mrlm` is used if `method="robust"`. If `method="ls"`, then `gls.series` is used if a correlation structure has been specified, i.e., if `ndups>1` or `block` is non-null and `correlation` is different from zero. If `method="ls"` and there is no correlation structure, `lm.series` is used.

Value

Object of class `MArrayLM`

Author(s)

Gordon Smyth

See Also

An overview of linear model functions in limma is given by 06.LinearModels.

Examples

```
# Simulate gene expression data for 100 probes and 6 microarrays
# Microarray are in two groups
# First two probes are differentially expressed in second group
# Std deviations vary between genes with prior df=4
sd <- 0.3*sqrt(4/rchisq(100,df=4))
y <- matrix(rnorm(100*6,sd=sd),100,6)
rownames(y) <- paste("Gene",1:100)
y[1:2,4:6] <- y[1:2,4:6] + 2
design <- cbind(Grp1=1,Grp2vs1=c(0,0,0,1,1,1))
options(digit=3)

# Ordinary fit
fit <- lmFit(y,design)
fit <- eBayes(fit)
fit
as.data.frame(fit[1:10,2])

# Various ways of summarising or plotting the results
topTable(fit,coef=2)
qqt(fit$t[,2],df=fit$df.residual+fit$df.prior)
abline(0,1)
volcanoplot(fit,coef=2,highlight=2)

# Various ways of writing results to file
## Not run: write.fit(fit,file="exampleresults.txt")
## Not run: write.table(fit,file="exampleresults2.txt")

# Robust fit
# (There may be some warning messages)
fit2 <- lmFit(y,design,method="robust")

# Fit with correlated arrays
# Suppose each pair of arrays is a block
block <- c(1,1,2,2,3,3)
dupcor <- duplicateCorrelation(y,design,block=block)
dupcor$consensus.correlation
fit3 <- lmFit(y,design,block=block,correlation=dupcor$consensus)

# Fit with duplicate probes
# Suppose two side-by-side duplicates of each gene
rownames(y) <- paste("Gene",rep(1:50,each=2))
dupcor <- duplicateCorrelation(y,design,ndups=2)
dupcor$consensus.correlation
fit4 <- lmFit(y,design,ndups=2,correlation=dupcor$consensus)
fit4 <- eBayes(fit3)
dim(fit4)
topTable(fit4,coef=2)
```

`lmscFit` *Fit Linear Model to Individual Channels of Two-Color Data*

Description

Fit a linear model to the individual log-intensities for each gene given a series of two-color arrays

Usage

```
lmscFit(object, design, correlation)
```

Arguments

object	an MAList object or a list from which M and A values may be extracted
design	a numeric matrix containing the design matrix for linear model in terms of the individual channels. The number of rows should be twice the number of arrays. The number of columns will determine the number of coefficients estimated for each gene.
correlation	numeric value giving the intra-spot correlation

Details

For two color arrays, the channels measured on the same set of arrays are correlated. The M and A however are uncorrelated for each gene. This function fits a linear model to the set of M and A-values for each gene after re-scaling the M and A-values to have equal variances. The input correlation determines the scaling required. The input correlation is usually estimated using intraspotCorrelation before using lmscFit.

Missing values in M or A are not allowed.

Value

An object of class MArrayLM

Author(s)

Gordon Smyth

References

Smyth, G. K. (2005). Individual channel analysis of two-colour microarray data. Invited Session IPM 11: Computational Tools For Microarray Analysis, 55th Session of the International Statistics Institute, Sydney, 12 April 2005. (Four-page paper distributed on CD at the conference.)

See Also

lm.fit.

An overview of methods for single channel analysis in limma is given by 07.SingleChannel.

Examples

```
library(sma)
# Subset of data from ApoAI case study in Limma User's Guide
data(MouseArray)
# Avoid non-positive intensities
RG <- backgroundCorrect(mouse.data,method="normexp")
MA <- normalizeWithinArrays(RG,mouse.setup)
MA <- normalizeBetweenArrays(MA,method="Aq")
# Randomly choose 500 genes for this example
i <- sample(1:nrow(MA),500)
MA <- MA[i,]
targets <- data.frame(Cy3=I(rep("Pool",6)),Cy5=I(c("WT","WT","WT","KO","KO","KO")))
```

```
targets.sc <- targetsA2C(targets)
targets.sc$Target <- factor(targets.sc$Target,levels=c("Pool","WT","KO"))
design <- model.matrix(~Target,data=targets.sc)
corfit <- intraspotCorrelation(MA,design)
fit <- lmscFit(MA,design,correlation=corfit$consensus)
cont.matrix <- cbind(KOvsWT=c(0,-1,1))
fit2 <- contrasts.fit(fit,cont.matrix)
fit2 <- eBayes(fit2)
topTable(fit2,adjust="fdr")
```

`loessFit` *Fast Simple Loess*

Description

A fast version of locally weighted regression when there is only one x-variable and only the fitted values and residuals are required.

Usage

```
loessFit(y, x, weights=NULL, span=0.3, bin=0.01/(2-is.null(weights)), iterations
```

Arguments

`y`	numeric vector of response values. Missing values are allowed.
`x`	numeric vector of predictor values Missing values are allowed.
`weights`	numeric vector of non-negative weights. Missing values are allowed.
`span`	numeric parameter between 0 and 1 specifying proportion of data to be used in the local regression moving window. Larger numbers give smoother fits.
`bin`	numeric value between 0 and 1 giving the proportion of the data which can be grouped in a single bin when doing local regression fit. `bin=0` forces an exact local regression fit with no interpolation.
`iterations`	number of iterations of loess fit

Details

This is a wrapper function to the Fortran and C code in the stats package which underlies the `lowess` and `loess` functions. Its is to give a streamlined common interface to `lowess` and `loess` for use in `normalizeWithinArrays`. When `weights` is null, this function is in effect a call to `lowess` in the stats package, with appropropriate choice of tuning parameters. When `weights` is non-null, it is in effect a call to `loess`. See the help pages for those functions for references and credits.

Note tha `lowess` is faster, needs less memory and is able to use a more accurate interpolation scheme than `loess`, so it is desirable to use `lowess` whenever `loess` is not needed to handle quantitative weights.

The arguments `span`, `cell` and `iterations` here have the same meaning as in `loess`. `span` is equivalent to the argument `f` of `lowess` and `iterations` is equivalent to `iter+1`. The parameter `bin` is intended to give a simple uniform interface to the `delta` argument of `lowess` and the `cell` argument of `loess`. `bin` translates to `delta=bin*diff(range(x))` in a call to `lowess` or to `cell=bin/span` in a call to `loess`.

Unlike lowess, loessFit returns values in original rather than sorted order. Also unlike lowess, loessFit allows missing values, the treatment being analogous to na.exclude. Unlike loess, loessFit returns a linear regression fit if there are insufficient observations to estimate the loess curve.

Value

A list with components

fitted	numeric vector of same length as y giving the loess fit
residuals	numeric vector of same length as x giving residuals from the fit

Author(s)

Gordon Smyth, based on code from lowess and loess by BD Ripley

See Also

See lowess and loess in the stats package.

See 05.Normalization for an outline of the limma package normalization functions.

Examples

```
y <- rnorm(1000)
x <- rnorm(1000)
w <- rep(1,1000)
# The following are equivalent apart from execution time
# and interpolation inaccuracies
system.time(fit <- loessFit(y,x)$fitted)
system.time(fit <- loessFit(y,x,w)$fitted)
system.time(fit <- fitted(loess(y~x,weights=w,span=0.3,family="symmetric",iterations=4)))
# The same but with sorted x-values
system.time(fit <- lowess(x,y,f=0.3)$y)
```

---

ma3x3 *Two dimensional Moving Averages with 3x3 Window*

Description

Apply a specified function to each to each value of a matrix and its immediate neighbors.

Usage

```
ma3x3.matrix(x,FUN=mean,na.rm=TRUE,...)
ma3x3.spottedarray(x,printer,FUN=mean,na.rm=TRUE,...)
```

Arguments

x	numeric matrix
FUN	function to apply to each window of values
na.rm	logical value, should missing values be removed when applying FUN
...	other arguments are passed to FUN
printer	list giving the printer layout, see PrintLayout-class

Details

For `ma3x3.matrix`, `x` is an arbitrary function. for `ma3x3.spotted`, each column of `x` is assumed to contain the expression values of a spotted array in standard order. The printer layout information is used to re-arrange the values of each column as a spatial matrix before applying `ma3x3.matrix`.

Value

Numeric matrix of same dimension as `x` containing smoothed values

Author(s)

Gordon Smyth

See Also

An overview of functions for background correction are given in `04.Background`.

Examples

```
x <- matrix(c(2,5,3,1,6,3,10,12,4,6,4,8,2,1,9,0),4,4)
ma3x3.matrix(x,FUN="mean")
ma3x3.matrix(x,FUN="min")
```

`makeContrasts` *Construct Matrix of Custom Contrasts*

Description

Construct the contrast matrix corresponding to specified contrasts of a set of parameters.

Usage

```
makeContrasts(..., contrasts=NULL, levels)
```

Arguments

`...`	expressions, or character strings which can be parsed to expressions, specifying contrasts
`contrasts`	character vector specifying contrasts
`levels`	character vector or factor giving the names of the parameters of which contrasts are desired, or a design matrix or other object with the parameter names as column names.

Details

This function expresses contrasts between a set of parameters as a numeric matrix. The parameters are usually the coefficients from a linear model fit, so the matrix specifies which comparisons between the coefficients are to be extracted from the fit. The output from this function is usually used as input to `contrasts.fit`. The contrasts can be specified either as expressions using `...` or as a character vector through `contrasts`. (Trying to specify contrasts both ways will cause an error.)

The parameter names must be syntactically valid variable names in R and so, for example, must begin with a letter rather than a numeral. See `make.names` for a complete specification of what is a valid name.

Value

Matrix which columns corresponding to contrasts.

Author(s)

Gordon Smyth

See Also

An overview of linear model functions in limma is given by the help page 06.LinearModels.

Examples

```
makeContrasts(B-A,C-B,C-A,levels=c("A","B","C"))
makeContrasts(contrasts="A-(B+C)/2",levels=c("A","B","C"))
x <- c("B-A","C-B","C-A")
makeContrasts(contrasts=x,levels=c("A","B","C"))
```

`makeUnique` *Make Values of Character Vector Unique*

Description

Paste characters on to values of a character vector to make them unique.

Usage

```
makeUnique(x)
```

Arguments

`x`	object to be coerced to a character vector

Details

Repeat values of `x` are labelled with suffixes "1", "2" etc.

Value

A character vector of the same length as `x`

Author(s)

Gordon Smyth

See Also

`makeUnique` is called by `merge.RGList`. Compare with `make.unique` in the base package.

Examples

```
x <- c("a","a","b")
makeUnique(x)
```

MAList-class *M-value, A-value Expression List - class*

Description

A simple list-based class for storing M-values and A-values for a batch of spotted microarrays. `MAList` objects are usually created during normalization by the functions `normalizeWithinArrays` or `MA.RG`.

Slots/List Components

`MAList` objects can be created by `new("MAList",MA)` where `MA` is a list. This class contains no slots (other than `.Data`), but objects should contain the following components:

`M`: numeric matrix containing the M-values (log-2 expression ratios). Rows correspond to spots and columns to arrays.
`A`: numeric matrix containing the A-values (average log-2 expression values).

Optional components include:

`weights`: numeric matrix of same dimensions as `M` containing relative spot quality weights. Elements should be non-ne
`other`: list containing numeric matrices of other spot-specific information. All matrices must have the same dimensi
`genes`: data.frame containing probe information. Should have one row for each spot. May have any number of colun
`targets`: data.frame containing information on the target RNA samples. Rows correspond to arrays. May have any nu
`printer`: list containing information on the process used to print the spots on the arrays. See PrintLayout.

Valid `MAList` objects may contain other optional components, but all probe or array information should be contained in the above components.

Methods

This class inherits directly from class `list` so any operation appropriate for lists will work on objects of this class. In addition, `MAList` objects can be subsetted and combined. `RGList` objects will return dimensions and hence functions such as `dim`, `nrow` and `ncol` are defined. `MALists` also inherit a `show` method from the virtual class `LargeDataObject`, which means that `RGLists` will print in a compact way.

Other functions in LIMMA which operate on `MAList` objects include `normalizeWithinArrays`, `normalizeBetweenArrays`, `normalizeForPrintorder`, `plotMA` and `plotPrintTipLoess`.

Author(s)

Gordon Smyth

See Also

02.Classes gives an overview of all the classes defined by this package.

`marrayNorm-class` is the corresponding class in the marrayClasses package.

MArrayLM-class *Microarray Linear Model Fit - class*

Description

A list-based class for storing the results of fitting gene-wise linear models to a batch of microarrays. Objects are normally created by `lmFit`.

Slots/Components

`MArrayLM` objects do not contain any slots (apart from `.Data`) but they should contain the following list components:

`coefficients`: `matrix` containing fitted coefficients or contrasts

`stdev.unscaled`: `matrix` containing unscaled standard deviations of the coefficients or contrasts

`sigma`: `numeric` vector containing residual standard deviations for each gene

`df.residual`: `numeric` vector containing residual degrees of freedom for each gene

Objects may also contain the following optional components:

`Amean`: `numeric` vector containing the average log-intensity for each probe over all the arrays in the original linear model fit. Note this vector does not change when a contrast is applied to the fit using `contrasts.fit`.

`genes`: `data.frame` containing gene names and annotation

`design`: design `matrix` of full column rank

`contrasts`: `matrix` defining contrasts of coefficients for which results are desired

`F`: `numeric` vector giving moderated F-statistics for testing all contrasts equal to zero

`F.p.value`: `numeric` vector giving p-value corresponding to `F.stat`

`s2.prior`: `numeric` value giving empirical Bayes estimated prior value for residual variances

`df.prior`: `numeric` vector giving empirical Bayes estimated degrees of freedom associated with `s2.prior` for each gene

`s2.post`: `numeric` vector giving posterior residual variances

`t`: `matrix` containing empirical Bayes t-statistics

`var.prior`: `numeric` vector giving empirical Bayes estimated prior variance for each true coefficient

`cov.coefficients`: numeric `matrix` giving the unscaled covariance matrix of the estimable coefficients

pivot: integer vector giving the order of coefficients in cov.coefficients. Is computed by the QR-decomposition of the design matrix.

If there are no weights and no missing values, then the MArrayLM objects returned by lmFit will also contain the QR-decomposition of the design matrix, and any other components returned by lm.fit.

Methods

RGList objects will return dimensions and hence functions such as dim, nrow and ncol are defined. MArrayLM objects inherit a show method from the virtual class LargeDataObject.

The functions ebayes and classifyTestsF accept MArrayLM objects as arguments.

Author(s)

Gordon Smyth

See Also

02.Classes gives an overview of all the classes defined by this package.

---

mdplot *mdplot*

Description

Creates a mean-difference plot.

Usage

```
mdplot(x, ...)
```

Arguments

x	numeric matrix with at least two columns
...	any other arguments are passed to plot

Details

Plots differences vs means for a set of bivariate values. This is useful to contrast expression values for two microarrays.

Note that an MA-plot plotMA is a type of mean-difference plot.

Value

A plot is created on the current graphics device.

Author(s)

Gordon Smyth

References

Chambers, J. M., Cleveland, W. S., Kleiner, B., and Tukey, P. A. (1983). Graphical Methods of Data Analysis. Wadsworth (pp. 48-57).

Cleveland, W. S., (1993). Visualizing Data. Hobart Press.

Bland, J. M., and Altman, D. G. (1986). Statistical methods for assessing agreement between two methods of clinical measurement. Lancet i, 307-310.

See also http://www.statsci.org/micrarra/refs/maplots.html

See Also

An overview of diagnostic functions available in LIMMA is given in 09.Diagnostics.

`merge` *Merge RGList or MAList Data Objects*

Description

Merge two microarray data sets represented by RGLists in possibly irregular order.

Usage

```
## S3 method for class 'RGList':
merge(x,y,...)
```

Arguments

`x`	`RGList-class` or `MAList-class` object
`y`	`RGList` object, corresponding to the same genes as for `x`, possibly in a different order, but with different arrays.
`...`	other arguments are accepted but not used at present

Details

`RGList` and `MAList` objects are list objects containing numeric matrices all of the same dimensions. The RGLists are merged by merging each of the components by row names or, if there are no row names, by IDs in the `genes` component. Unlike when using `cbind`, row names are not required to be in the same order or to be unique. In the case of repeated row names, the order of the rows with repeated names in preserved. This means that the first occurrence of each name in `x$R` is matched with the first occurrence of the same name in `y$R`, the second with the second, and so on. The final vector of row names is the same as in `x`.

Note: if the `RGList` objects contain the same number of genes in the same order then the appropriate function to combine them is `cbind` rather than `merge`.

Value

An merged object of the same class as `x` and `y` with the same components as `x`. Component matrices have the same rows names as in `x` but columns from `y` as well as `x`.

Author(s)

Gordon Smyth

See Also

R base provides a `merge` method for merging data.frames.

An overview of limma commands for reading, subsetting and merging data is given in 03.Reading-Data.

Examples

```
M <- A <- matrix(11:14,4,2)
rownames(M) <- rownames(A) <- c("a","a","b","c")
MA1 <- new("MAList",list(M=M,A=A))

M <- A <- matrix(21:24,4,2)
rownames(M) <- rownames(A) <- c("b","a","a","c")
MA2 <- new("MAList",list(M=M,A=A))

merge(MA1,MA2)
merge(MA2,MA1)
```

---

mergeScans *Merge two scans of two-color arrays*

Description

Merge two sets of intensities of two-color arrays that are scanned twice at two different scanner settings, one at a lower gain setting with no saturated spot intensities and the other at a higher gain setting with a higher signal-to-noise ratio and some saturated spot intensities.

Usage

```
mergeScansRG(RGlow, RGhigh, AboveNoiseLowG=NULL, AboveNoiseLowR=NULL, outlierp=0
```

Arguments

`RGlow`	object of class `RGList` containing red and green intensities constituting two-color microarray data scanned at a lower gain setting.
`RGhigh`	object of class `RGList` containing red and green intensities constituting two-color microarray data scanned at a higher gain setting.
`AboveNoiseLowG`	matrix of 1 or 0 for low scan intensities of green color, 1 for spots above noise level or 0 otherwise. One column per array.
`AboveNoiseLowR`	matrix of 1 or 0 for low scan intensities of red color, 1 for spots above noise level or 0 otherwise. One column per array.
`outlierp`	p-value for outliers. 0 for no outlier detection or any value between 0 and 1. Default p-value is 0.01.

Details

This function merges two separate scans of each fluorescent label on a two-color array scanned at two different scanner settings by using a nonlinear regression model consisting of two linear regression lines and a quadratic function connecting the two, which looks like a hockey stick. The changing point, i.e. the saturation point, in high scan is also estimated as part of model. Signals produced for certain spots can sometimes be very low (below noise) or too high (saturated) to be accurately read by the scanner. The proportions of spots that are below noise or above saturation are affected by the settings of the laser scanner used to read the arrays, with low scans minimizing saturation effects and high scans maximizing signal-to-noise ratios. Saturated spots can cause bias in intensity ratios that cannot be corrected for using conventional normalization methods.

Each fluorescent label on a two-color array can be scanned twice: for example, a high scan targeted at reaching saturation level for the brightest 1 percent of the spots on the array, and a low scan targeted at the lowest level of intensity which still allowed accurate grid placement on the arrays. By merging data from two separate laser scans of each fluorescent label on an array, we can avoid the potential bias in signal intensities due to below noise or above saturation and, thus provide better estimates of true differential expression as well as increase usable spots.

The merging process is designed to retain signal intensities from the high scan except when scanner saturation causes the high scan signal to be under-measured. The saturated spots are predicted from the corresponding low scans by the fitted regression model. It also checks any inconsistency between low and high scans.

Value

An object of class `RGList-class` with the following components:

`G`	numeric matrix containing the merged green (cy3) foreground intensities. Rows correspond to spots and columns to arrays.
`R`	numeric matrix containing the merged red (cy5) foreground intensities. Rows correspond to spots and columns to arrays.
`Gb`	numeric matrix containing the green (cy3) background intensities from high scan.
`Rb`	numeric matrix containing the red (cy5) background intensities from high scan.
`other`	list numeric matrices `Gsaturated`, `Rsatured`, `Goutlier` and `Routlier`. The first two contain saturation flags (1=saturated, 0=otherwise) for the green (cy3) and red (Cy5) channels of the high scan. The second two contain outlier flags (1=outlier, 0=otherwise) for the green (cy3) and red (Cy5) channels.

Author(s)

Dongseok Choi ⟨choid@ohsu.edu⟩.

References

Choi D, O'Malley JP, Lasarev MR, Lapidus J, Lu X, Pattee P, Nagalla SR (2006). Extending the Dynamic Range of Signal Intensities in DNA Microarrays. *Online Journal of Bioinformatics*, **7**, 46-56.

Examples

```
## Not run:
#RG1: An RGList from low scan
```

```
#RG2: An RGList from high scan
RGmerged <- mergeScansRG(RG1,RG2,AboveNoiseLowG=ANc3,AboveNoiseLowR=ANc5)

#merge two scans when all spots are above noise in low scan and no outlier detection.
RGmerged <- mergeScansRG(RG1,RG2,outlierp=0)
## End(Not run)
```

`modelMatrix` *Construct Design Matrix*

Description

Construct design matrix from RNA target information for a two colour microarray experiment.

Usage

```
modelMatrix(targets, parameters, ref, verbose=TRUE)
uniqueTargets(targets)
```

Arguments

`targets`	matrix or data.frame with columns `Cy3` and `Cy5` specifying which RNA was hybridized to each array
`parameters`	matrix specifying contrasts between RNA samples which should correspond to regression coefficients. Row names should correspond to unique RNA sample names found in `targets`.
`ref`	character string giving name of one of the RNA sources to be treated as reference. Exactly one argument of `parameters` or `ref` should be specified.
`verbose`	logical, if `TRUE` then unique names found in `targets` will be printed to standard output

Details

This function computes a design matrix for input to `lmFit` when analysing two-color microarray experiments in terms of log-ratios.

If the argument `ref` is used, then the experiment is treated as a one-way layout and the coefficients measure expression changes relative to the RNA source specified by `ref`. The RNA source `ref` is often a common reference which appears on every array or is a control sample to which all the others are compared. There is no restriction however. One can choose `ref` to be any of the RNA sources appearing the `Cy3` or `Cy5` columns of `targets`.

If the `parameters` argument is set, then the columns of this matrix specify the comparisons between the RNA sources which are of interest. This matrix must be of size n by (n-1), where n is the number of unique RNA sources found in `Cy3` and `Cy5`, and must have row names which correspond to the RNA sources.

Value

`modelMatrix` produces a numeric design matrix with row names as in `targets` and column names as in `parameters`.

`uniqueTargets` produces a character vector of unique target names from the columns `Cy3` and `Cy5` of `targets`.

Author(s)

Gordon Smyth

See Also

model.matrix in the stats package.

An overview of linear model functions in limma is given by 06.LinearModels.

Examples

```
targets <- cbind(Cy3=c("Ref","Control","Ref","Treatment"),Cy5=c("Control","Ref","Treatmen
rownames(targets) <- paste("Array",1:4)

parameters <- cbind(C=c(-1,1,0),T=c(-1,0,1))
rownames(parameters) <- c("Ref","Control","Treatment")

modelMatrix(targets, parameters)
modelMatrix(targets, ref="Ref")
```

---

modifyWeights *modifyWeights*

Description

Modify weights matrix for given gene status values.

Usage

```
modifyWeights(weights=rep(1,length(status)), status, values, multipliers)
```

Arguments

weights	numeric matrix of relative weights, rows corresponding to genes and columns to arrays
status	character vector giving the control status of each spot on the array, of same length as the number of rows of weights
values	character vector giving subset of the unique values of status
multipliers	numeric vector of same length as values giving factor by which weights will be modified

Details

The function is usually used to temporarily modify the weights matrix during normalization of data. The function can be used for example to give zero weight to spike-in ratio control spots during normalization.

Value

Numeric matrix of same dimensions as weights with rows corresponding to values in status modified by the specified multipliers.

Author(s)

Gordon Smyth

See Also

An overview of normalization functions available in LIMMA is given in 05.Normalization.

Examples

```
w <- matrix(runif(6*3),6,3)
status <- c("Gene","Gene","Ratio_Control","Ratio_Control","Gene","Gene")
modifyWeights(w,status,values="Ratio_Control",multipliers=0)
```

---

mrlm *Fit Linear Model to Microrray Data by Robust Regression*

Description

Fit a linear model genewise to expression data from a series of arrays. The fit is by robust M-estimation allowing for a small proportion of outliers. This is a utility function for `lmFit`.

Usage

```
mrlm(M,design=NULL,ndups=1,spacing=1,weights=NULL,...)
```

Arguments

`M`	numeric matrix containing log-ratio or log-expression values for a series of microarrays, rows correspond to genes and columns to arrays.
`design`	numeric design matrix defining the linear model, with rows corresponding to arrays and columns to comparisons to be estimated. The number of rows must match the number of columns of `M`. Defaults to the unit vector meaning that the arrays are treated as replicates.
`ndups`	a positive integer giving the number of times each gene is printed on an array. `nrow(M)` must be divisible by `ndups`.
`spacing`	the spacing between the rows of `M` corresponding to duplicate spots, `spacing=1` for consecutive spots.
`weights`	numeric matrix of the same dimension as `M` containing weights. If it is of different dimension to `M`, it will be filled out to the same size. `NULL` is equivalent to equal weights.
`...`	any other arguments are passed to `rlm.default`.

Details

This is a utility function used by the higher level function `lmFit`. Most users should not use this function directly but should use `lmFit` instead.

This function fits a linear model for each gene by calling the function `rlm` from the MASS library.

Warning: don't use weights with this function unless you understand how `rlm` treats weights. The treatment of weights is somewhat different from that of `lm.series` and `gls.series`.

Value

A list with components

coefficients	numeric matrix containing the estimated coefficients for each linear model. Same number of rows as M, same number of columns as design.
stdev.unscaled	numeric matrix conformal with coef containing the unscaled standard deviations for the coefficient estimators. The standard errors are given by stdev.unscaled * sigma.
sigma	numeric vector containing the residual standard deviation for each gene.
df.residual	numeric vector giving the degrees of freedom corresponding to sigma.
qr	QR decomposition of design.

Author(s)

Gordon Smyth

See Also

rlm.

An overview of linear model functions in limma is given by 06.LinearModels.

---

normalizeMedianAbsValues

Normalize Columns of a Matrix to have the Median Absolute Value

---

Description

Performs scale normalization of an M-value matrix or an A-value matrix across a series of arrays. Users do not normally need to call these functions directly - use normalizeBetweenArrays instead.

Usage

```
normalizeMedianAbsValues(x)
```

Arguments

x	numeric matrix

Details

If x is a matrix of log-ratios of expression (M-values) then normalizeMedianAbsValues is very similar to scaling to equalize the median absolute deviation (MAD) as in Yang et al (2001, 2002). Here the median-absolute value is used for preference to as to not re-center the M-values.

normalizeMedianAbsValues is also used to scale the A-values when scale-normalization is applied to an MAList object.

Value

A numeric matrix of the same size as that input which has been scaled so that each column as the same median-absolute value.

Author(s)

Gordon Smyth

See Also

An overview of LIMMA functions for normalization is given in 05.Normalization.

Examples

```
M <- cbind(Array1=rnorm(10),Array2=2*rnorm(10))
normalizeMedianAbsValues(M)
```

---

normalizeRobustSpline

Normalize Single Microarray Using Shrunk Robust Splines

Description

Normalize the M-values for a single microarray using robustly fitted regression splines and empirical Bayes shrinkage.

Usage

```
normalizeRobustSpline(M,A,layout,df=5,method="M")
```

Arguments

M	numeric vector of M-values
A	numeric vector of A-values
layout	list specifying the dimensions of the spot matrix and the grid matrix
df	degrees of freedom for regression spline, i.e., the number of regression coefficients and the number of knots
method	choices are "M" for M-estimation or "MM" for high breakdown point regression

Details

This function implements an idea similar to print-tip loess normalization but uses regression splines in place of the loess curves and uses empirical Bayes ideas to shrink the individual prtin-tip curves towards a common value. This allows the technique to introduce less noise into good quality arrays with little spatial variation while still giving good results on arrays with strong spatial variation.

Value

Numeric vector containing normalized M-values.

Author(s)

Gordon Smyth

References

The function is based on unpublished work by the author.

See Also

An overview of LIMMA functions for normalization is given in 05.Normalization.

Examples

```
library(sma)
data(MouseArray)
MA <- MA.RG(mouse.data)
normM <- normalizeRobustSpline(MA$M[,1],MA$A[,1],mouse.setup)
```

---

normalizeWithinArrays
Normalize Within Arrays

Description

Normalize the expression log-ratios for one or more two-colour spotted microarray experiments so that the log-ratios average to zero within each array or sub-array.

Usage

```
normalizeWithinArrays(object, layout, method="printtiploess", weights=object$wei
MA.RG(object, bc.method="subtract", offset=0)
RG.MA(object)
```

Arguments

`object`	object of class `list`, `RGList` or `MAList` containing red and green intensities constituting two-color microarray data.
`layout`	list specifying the dimensions of the spot matrix and the grid matrix. For details see `PrintLayout-class`.
`method`	character string specifying the normalization method. Choices are `"none"`, `"median"`, `"loess"`, `"printtiploess"`, `"composite"`, `"control"` and `"robustspline"`. A partial string sufficient to uniquely identify the choice is permitted.
`weights`	numeric matrix or vector of the same size and shape as the components of `object` containing spot quality weights.
`span`	numeric scalar giving the smoothing parameter for the `loess` fit
`iterations`	number of iterations used in loess fitting. More iterations give a more robust fit.
`controlspots`	numeric or logical vector specifying the subset of spots which are non-differentially-expressed control spots, for use with `method="composite"` or `method="control"`.

df	degrees of freedom for spline if method="robustspline".
robust	robust regression method if method="robustspline". Choices are "M" or "MM".
bc.method	character string specifying background correct method, see backgroundCorrect for options.
offset	numeric value, intensity offset used when computing log-ratios, see backgroundCorrect.

Details

Normalization is intended to remove from the expression measures any systematic trends which arise from the microarray technology rather than from differences between the probes or between the target RNA samples hybridized to the arrays.

This function normalizes M-values (log-ratios) for dye-bias within each array. Apart from method="none" and method="median", all the normalization methods make use of the relationship between dye-bias and intensity. Method "none" computes M-values and A-values but does no normalization. Method "median" subtracts the weighted median from the M-values for each array.

The loess normalization methods ("loess", "printtiploess" and "composite") were proposed by Yang et al (2001, 2002). Smyth and Speed (2003) review these methods and describe how the methods are implemented in the limma package, including choices of tuning parameters. More information on the loess control parameters span and iterations can be found under loessFit. The default values used here are equivalent to those for the older function stat.ma in the sma package.

The "control" method fits a global loess curve through a set of control spots, such as a whole-library titration series, and applies that curve to all the other spots.

The "robustspline" method calls normalizeRobustSpline. See that function for more documentation.

MA.RG converts an unlogged RGList object into an MAList object. MA.RG(object) is equivalent to normalizeWithinArrays(object,method="none").

RG.MA(object) converts back from an MAList object to a RGList object with unlogged intensities.

weights is normally a matrix giving a quality weight for every spot on every array. If weights is instead a vector or a matrix with only one column, then the weights will be assumed to be the same for every array, i.e., the weights will be probe-specific rather than spot-specific.

Value

An object of class MAList. Any components found in object will preserved except for R, G, Rb, Gb and other.

Author(s)

Gordon Smyth

References

Yang, Y. H., Dudoit, S., Luu, P., and Speed, T. P. (2001). Normalization for cDNA microarray data. In *Microarrays: Optical Technologies and Informatics*, M. L. Bittner, Y. Chen, A. N. Dorsel, and E. R. Dougherty (eds), Proceedings of SPIE, Vol. 4266, pp. 141-152.

Yang, Y. H., Dudoit, S., Luu, P., Lin, D. M., Peng, V., Ngai, J., and Speed, T. P. (2002). Normalization for cDNA microarray data: a robust composite method addressing single and multiple slide systematic variation. *Nucleic Acids Research* **30**(4):e15.

Smyth, G. K., and Speed, T. P. (2003). Normalization of cDNA microarray data. *Methods* **31**, 265-273.

See Also

An overview of limma functions for normalization is given in 05.Normalization. In particular, see `normalizeBetweenArrays` for between-array normalization.

The original loess normalization function was the `statma` funtion in the sma package. `normalizeWithinArrays` is a direct generalization of that function, with more options and with support for quantitative spot quality weights.

A different implementation of loess normalization methods is provided by the `maNorm` in the marray package.

An alternative wrapper for loess normalization, using different data classes but calling the limma functions, is provided by the `normalise` function in the arrayMagic package.

Examples

```
#  See normalizeBetweenArrays
```

---

normalizeBetweenArrays
Normalize Between Arrays

Description

Normalizes expression intensities so that the intensities or log-ratios have similar distributions across a series of arrays.

Usage

```
normalizeBetweenArrays(object, method="Aquantile", targets=NULL, ...)
```

Arguments

`object`	a `matrix`, `RGList` or `MAList` object containing expression ratios for a series of arrays
`method`	character string specifying the normalization method to be used. Choices are `"none"`, `"scale"`, `"quantile"`, `"Aquantile"`, `"Gquantile"`, `"Rquantile"`, `"Tquantile"` or `"vsn"`. A partial string sufficient to uniquely identify the choice is permitted.
`targets`	vector, factor or matrix of length twice the number of arrays, used to indicate target groups if `method="Tquantile"`
`...`	other arguments are passed to `normalizeQuantiles` if one of the quantile methods are used or to `vsn` if `method="vsn"`

Details

normalizeWithinArrays normalizes expression values to make intensities consistent within each array. normalizeBetweenArrays normalizes expression values to achieve consistency between arrays. Normalization between arrays is usually, but not necessarily, applied after normalization within arrays. An exception is method="vsn", see below.

The scale normalization method was proposed by Yang et al (2001, 2002) and is further explained by Smyth and Speed (2003). The idea is simply to scale the log-ratios to have the same median-abolute-deviation (MAD) across arrays. This idea has also been implemented by the maNormScale function in the marrayNorm package. The implementation here is slightly different in that the MAD scale estimator is replaced with the median-absolute-value and the A-values are normalized as well as the M-values.

Quantile normalization was proposed by Bolstad et al (2003) for Affymetrix-style single-channel arrays and by Yang and Thorne (2003) for two-color cDNA arrays. method="quantile" ensures that the intensities have the same empirical distribution across arrays and across channels. method="Aquantile" ensures that the A-values (average intensities) have the same empirical distribution across arrays leaving the M-values (log-ratios) unchanged. These two methods are called "q" and "Aq" respectively in Yang and Thorne (2003).

method="Tquantile" performs quantile normalization separately for the groups indicated by targets. targets may be a target matrix such as read by readTargets or can be a vector indicating green channel groups followed by red channel groups.

method="Gquantile" ensures that the green (first) channel has the same empirical distribution across arrays, leaving the M-values (log-ratios) unchanged. This method might be used when the green channel is a common reference throughout the experiment. In such a case the green channel represents the same target throughout, so it makes compelling sense to force the distribution of intensities to be same for the green channel on all the arrays, and to adjust to the red channel accordingly. method="Rquantile" ensures that the red (second) channel has the same empirical distribution across arrays, leaving the M-values (log-ratios) unchanged. Both Gquantile and Rquantile normalization have the implicit effect of changing the red and green log-intensities by equal amounts.

If object is a matrix then the scale, quantile or vsn normalization will be applied to the columns. Applying method="Aquantile" when object is a matrix will produce an error.

method="vsn" uses the vsn function from the vsn package. For this option the input object should contain raw intensities, i.e., prior to background correction, log-transformation or any normalization. Note that the normalized intensities are on the log-2 scale, not the log-e scale output by the vsn function in the vsn package.

Value

If object is a matrix then normalizeBetweenArrays produces a matrix of the same size. Otherwise, normalizeBetweenArrays produces an MAList object with M and A-values on the log-2 scale.

Author(s)

Gordon Smyth

References

Bolstad, B. M., Irizarry R. A., Astrand, M., and Speed, T. P. (2003), A comparison of normalization methods for high density oligonucleotide array data based on bias and variance. *Bioinformatics* **19**, 185-193.

Smyth, G. K., and Speed, T. P. (2003). Normalization of cDNA microarray data. *Methods* **31**, 265-273.

Yang, Y. H., Dudoit, S., Luu, P., and Speed, T. P. (2001). Normalization for cDNA microarray data. In *Microarrays: Optical Technologies and Informatics*, M. L. Bittner, Y. Chen, A. N. Dorsel, and E. R. Dougherty (eds), Proceedings of SPIE, Volume 4266, pp. 141-152.

Yang, Y. H., Dudoit, S., Luu, P., Lin, D. M., Peng, V., Ngai, J., and Speed, T. P. (2002). Normalization for cDNA microarray data: a robust composite method addressing single and multiple slide systematic variation. *Nucleic Acids Research* **30**(4):e15.

Yang, Y. H., and Thorne, N. P. (2003). Normalization for two-color cDNA microarray data. In: D. R. Goldstein (ed.), *Science and Statistics: A Festschrift for Terry Speed*, IMS Lecture Notes - Monograph Series, Volume 40, pp. 403-418.

See Also

An overview of LIMMA functions for normalization is given in 05.Normalization.

See also `maNormScale` in the marrayNorm package, `normalize` in the affy package and `vsn` in the vsn package.

Examples

```
library(sma)
data(MouseArray)
MA <- normalizeWithinArrays(mouse.data, mouse.setup)
plot.scale.box(MA$M)

#  Between array scale normalization as in Yang et al (2001):
MA <- normalizeBetweenArrays(MA,method="scale")
print(MA)
show(MA)
plot.scale.box(MA$M)

#  One can get the same results using the matrix method:
M <- normalizeBetweenArrays(MA$M,method="scale")
plot.scale.box(M)

#  MpAq normalization as in Yang and Thorne (2003):
MpAq <- normalizeWithinArrays(mouse.data, mouse.setup)
MpAq <- normalizeBetweenArrays(MpAq, method="Aq")
plotDensities(MpAq)
```

---

normalizeForPrintorder

Print-Order Normalization

Description

Normalize intensity values on one or more spotted microarrays to adjust for print-order effects.

Usage

```
normalizeForPrintorder(object, layout, start="topleft", method = "loess", separa
normalizeForPrintorder.rg(R, G, printorder, method = "loess", separate.channels
plotPrintorder(object, layout, start="topleft", slide = 1, method = "loess", sep
```

Arguments

object	an RGList or list object containing components R and G which are matrices containing the red and green channel intensities for a series of arrays
R	numeric vector containing red channel intensities for a single microarray
G	numeric vector containing the green channel intensities for a single microarray
layout	list specifying the printer layout, see PrintLayout-class
start	character string specifying where printing starts in each pin group. Choices are "topleft" or "topright".
printorder	numeric vector specifying order in which spots are printed. Can be computed from printorder(layout,start=start).
slide	positive integer giving the column number of the array for which a plot is required
method	character string, "loess" if a smooth loess curve should be fitted through the print-order trend or "plate" if plate effects are to be estimated
separate.channels	logical, TRUE if normalization should be done separately for the red and green channel and FALSE if the normalization should be proportional for the two channels
span	numerical constant between 0 and 1 giving the smoothing span for the loess the curve. Ignored if method="plate".
plate.size	positive integer giving the number of consecutive spots corresponding to one plate or plate pack. Ignored if method="loess".
plot	logical. If TRUE then a scatter plot of the print order effect is sent to the current graphics device.

Details

Print-order is associated with the 384-well plates used in the printing of spotted microarrays. There may be variations in DNA concentration or quality between the different plates. The may be variations in ambient conditions during the time the array is printed.

This function is intended to pre-process the intensities before other normalization methods are applied to adjust for variations in DNA quality or concentration and other print-order effects.

Printorder means the order in which spots are printed on a microarray. Spotted arrays are printed using a print head with an array of print-tips. Spots in the various tip-groups are printed in parallel. Printing is assumed to start in the top left hand corner of each tip-groups and to proceed right and down by rows, or else to start in the top right hand and to proceed left and down by rows. See printorder for more details. (WARNING: this is not always the case.) This is true for microarrays printed at the Australian Genome Research Facility but might not be true for arrays from other sources.

If object is an RGList then printorder is performed for each intensity in each array.

plotPrintorder is a non-generic function which calls normalizeForPrintorder with plot=TRUE.

Value

normalizeForPrintorder produces an RGList containing normalized intensities.

The function plotPrintorder or normalizeForPrintorder.rg with plot=TRUE returns no value but produces a plot as a side-effect.

`normalizeForPrintorder.rg` with `plot=FALSE` returns a list with the following components:

`R`	numeric vector containing the normalized red channel intensities
`G`	numeric vector containing the normalized red channel intensites
`R.trend`	numeric vector containing the fitted printorder trend for the red channel
`G.trend`	numeric vector containing the fitted printorder trend for the green channe

Author(s)

Gordon Smyth

References

Smyth, G. K. Print-order normalization of cDNA microarrays. March 2002. http://www.statsci.org/smyth/pubs/porder/porder.html

See Also

`printorder`.

An overview of LIMMA functions for normalization is given in 05.Normalization.

Examples

```
library(sma)
data(MouseArray)
plotPrintorder(mouse.data,mouse.setup,slide=1,separate=TRUE)
RG <- normalizeForPrintorder(mouse.data,mouse.setup)
```

`normalizeQuantiles` *Normalize Columns of a Matrix to have the same Quantiles*

Description

Normalize the columns of a matrix to have the same quantiles, allowing for missing values. Users do not normally need to call this function directly - use `normalizeBetweenArrays` instead.

Usage

```
normalizeQuantiles(A, ties=TRUE)
```

Arguments

`A`	numeric matrix. Missing values are allowed.
`ties`	logical. If `TRUE`, ties in each column of `A` are treated in careful way. tied values will be normalized to the mean of the corresponding pooled quantiles.

Details

This function is intended to normalize single channel or A-value microarray intensities between arrays. Each quantile of each column is set to the mean of that quantile across arrays. The intention is to make all the normalized columns have the same empirical distribution. This will be exactly true if there are no missing values and no ties within the columns: the normalized columns are then simply permutations of one another.

If there are ties amongst the intensities for a particular array, then with `ties=FALSE` the ties are broken in an unpredictable order. If `ties=TRUE`, all the tied values for that array will be normalized to the same value, the average of the quantiles for the tied values.

Value

A matrix of the same dimensions as `A` containing the normalized values.

Author(s)

Gordon Smyth

References

Bolstad, B. M., Irizarry R. A., Astrand, M., and Speed, T. P. (2003), A comparison of normalization methods for high density oligonucleotide array data based on bias and variance. *Bioinformatics* **19**, 185-193.

See Also

An overview of LIMMA functions for normalization is given in 05.Normalization.

`normexp.fit` *Fit Normal+Exp Convolution Model to Observed Intensities*

Description

Fit the normal+exponential convolution model to a vector of observed intensities. The normal part represents the background and the exponential part represents the signal intensities. This function is called by `backgroundCorrect` and is not normally called directly by users.

Usage

```
normexp.fit(x, method="saddle", n.pts=NULL, trace=FALSE)
```

Arguments

`x`	numeric vector of (background corrected) intensities
`method`	method used to estimate the three parameters. Choices for `normexp.fit` are `"mle"`, `"saddle"`, `"rma"` and `"rma75"`.
`n.pts`	number of quantiles of `x` to use for the fit. If `NULL` then all values of `x` will be used.
`trace`	logical, if `TRUE`, tracing information on the progress of the optimization is given.

Details

The Normal+Exp (normexp) convolution model is a mathematical model representing microarray intensity data for the purposes of background correction. It was proposed originally as part of the RMA algorithm for Affymetrix microarray data. For two-color microarry data, the normexp background correction method was introduced and compared with other methods by Ritchie et al (2007).

This function uses maximum likelihood estimation to fit the normexp model to background-corrected intensities. The model assumes that the observed intensities are the sum of background and signal components, the background being normal and the signal being exponential distributed.

The likelihood may be computed exactly (`method="mle"`) or approximated using a saddle-point approximation (`method="saddle"`). The saddle-point approximation was proposed by Ritchie et al (2007). Silver et al (2008) added some computational refinements to the saddle-point approximation, making it more reliable in practice, and developed the exact likelihood maximization algorithm. The `"mle"` method uses the best performing algorithm from Silver et al (2008), which calls the optimization function `nlminb` with analytic first and second derivatives. Derivatives are computed with respect to the normal-mean, the log-normal-variance and the log-exponential-mean.

Two ad-hoc estimators are also available which do not require iterative estimation. `"rma"` results in a call to the `bg.parameters` function of the affy package. This provides the kernel estimation method that is part of the RMA algorithm for Affymetrix data. `"rma75"` uses the similar but less biased RMA-75 method from McGee and Chen (2006).

If the length `x` is very large, it may be worth saving computation time by setting `n.pts` to a value less than the total number of probes, for example `n.pts=2^14`.

Value

A list containing the components

`par`	numeric vector giving estimated values of the mean and log-standard-deviation of the background-normal part and the log-mean of the signal-exponential part.
`m2loglik`	numeric scalar giving minus twice the maximized log-likelihood
`convergence`	integer code indicating successful convergence or otherwise of the optimization.

Author(s)

Gordon Smyth and Jeremy Silver

References

McGee, M., and Chen, Z. (2006). Parameter estimation for the exponential-normal convolution model for background correction of Affymetrix GeneChip data. *Stat Appl Genet Mol Biol*, 5(1), Article 24.

Ritchie, M. E., Silver, J., Oshlack, A., Silver, J., Holmes, M., Diyagama, D., Holloway, A., and Smyth, G. K. (2007). A comparison of background correction methods for two-colour microarrays. *Bioinformatics* http://bioinformatics.oxfordjournals.org/cgi/content/abstract/btm412

Silver, J., Ritchie, M. E., and Smyth, G. K. (2008). Microarray background correction: maximum likelihood estimation for the normal-exponential convolution model. *Biostatistics*. To appear. http://www.statsci.org/smyth/pubs/normexp_28_Jul_2008.pdf

See Also

normexp.signal, \code{normexp.signal}

An overview of background correction functions is given in 04.Background.

Examples

```
x <- c(2,3,1,10,3,20,5,6)
out <- normexp.fit(x)
normexp.signal(out$par, x=x)
```

---

normexp.signal *Expected Signal Given Observed Foreground Under Normal+Exp Model*

Description

Adjust foreground intensities for observed background using Normal+Exp Model. This function is called by backgroundCorrect and is not normally called directly by the user.

Usage

```
normexp.signal(par, x)
```

Arguments

par	numeric vector containing the parameters of the Normal+Exp distribution, see normexp.fit for details.
x	numeric vector of (background corrected) intensities

Details

In general the vector normmean is computed conditional on background at each spot.

Value

Numeric vector containing adjusted intensities.

Author(s)

Gordon Smyth

References

McGee, M., and Chen, Z. (2006). Parameter estimation for the exponential-normal convolution model for background correction of Affymetrix GeneChip data. *Stat Appl Genet Mol Biol*, 5(1), Article 24.

Ritchie, M. E., Silver, J., Oshlack, A., Silver, J., Holmes, M., Diyagama, D., Holloway, A., and Smyth, G. K. (2007). A comparison of background correction methods for two-colour microarrays. *Bioinformatics* http://bioinformatics.oxfordjournals.org/cgi/content/abstract/btm412

See Also

normexp.fit

An overview of background correction functions is given in 04.Background.

Examples

```
# See normexp.fit
```

---

plotDensities *Individual-channel Densities Plot*

Description

Plots the densities of individual-channel intensities for two-color microarray data.

Usage

```
plotDensities(object, log=TRUE, arrays=NULL, singlechannels=NULL, groups=NULL, c
```

Arguments

object	an RGList or MAList object. RGList objects containing logged or unlogged intensities can be accommodated using the log.transform argument.
log	logical, should densities be formed and plotted for the log-intensities (TRUE) or raw intensities (FALSE)?
arrays	vector of integers giving the arrays from which the individual-channels will be selected to be plotted. Corresponds to columns of M and A (or R and G). Defaults to all arrays.
singlechannels	vector of integers indicating which individual-channels will be selected to be plotted. Values correspond to the columns of the matrix of cbind(R,G) and range between 1:ncol(R) for red channels and ((ncol(R)+1):(ncol(R)+ncol(G))) for the green channels in object. Defaults to all channels.
groups	vector of consecutive integers beginning at 1 indicating the groups of arrays or individual-channels (depending on which of arrays or singlechannels are non NULL). This is used to color any groups of the individual-channel densities. If NULL (default), groups correspond to the red and green channels. If both arrays and singlechannels are NULL all arrays are selected and groups (if specified) must correspond to the arrays.
col	vector of colors of the same length as the number of different groups. If NULL (default) the col equals c("red","green"). See details for more specifications.

Details

This function is used as a data display technique associated with between-array normalization, especially individual-channel normalization methods such as quantile-normalization. See the section on between-array normalization in the LIMMA User's Guide.

If no `col` is specified, the default is to color individual channels according to red and green. If both `arrays` and `groups` are non-`NULL`, then the length of `groups` must equal the length of `arrays` and the maximum of `groups` (i.e. the number of groups) must equal the length of `col` otherwise the default color of black will be used for all individual-channels. If `arrays` is `NULL` and both `singlechannels` and `groups` are non-`NULL`, then the length of `groups` must equal the length of `singlechannels` and the maximum of `groups` (i.e. the number of groups) must equal the length of `col` otherwise the default color of black will be used for all individual-channels.

Value

A plot is created on the current graphics device.

Author(s)

Natalie Thorne

See Also

An overview of diagnostic plots in LIMMA is given in 09.Diagnostics. There is a section using `plotDensities` in conjunction with between-array normalization in the *LIMMA User's Guide*.

Examples

```
library(sma)
data(MouseArray)

#  no normalization but background correction is done
MA.n <- MA.RG(mouse.data)

#  Default settings for plotDensities.
plotDensities(MA.n)

#  One can reproduce the default settings.
plotDensities(MA.n,arrays=c(1:6),groups=c(rep(1,6),rep(2,6)),
col=c("red","green"))

#  Color R and G individual-channels by blue and purple.
plotDensities(MA.n,arrays=NULL,groups=NULL,col=c("blue","purple"))

#  Indexing individual-channels using singlechannels (arrays=NULL).
plotDensities(MA.n,singlechannels=c(1,2,7))

#  Change the default colors from c("red","green") to c("pink","purple")
plotDensities(MA.n,singlechannels=c(1,2,7),col=c("pink","purple"))

#  Specified too many colors since groups=NULL defaults to two groups.
plotDensities(MA.n,singlechannels=c(1,2,7),col=c("pink","purple","blue"))

#  Three individual-channels, three groups, three colors.
plotDensities(MA.n,singlechannels=c(1,2,7),groups=c(1,2,3),
col=c("pink","purple","blue"))
```

```
#  Three individual-channels, one group, one color.
plotDensities(MA.n,singlechannels=c(1,2,7),groups=c(1,1,1),
col=c("purple"))

#  All individual-channels, three groups (ctl,tmt,reference), three colors.
plotDensities(MA.n,singlechannels=c(1:12),
groups=c(rep(1,3),rep(2,3),rep(3,6)),col=c("darkred","red","green"))
```

---

plotFB *FB-Plot*

Description

Creates foreground-background plots.

Usage

```
plotFB(RG, array=1, lim="separate", pch=16, cex=0.2, ...)
```

Arguments

RG	an RGList object.
array	integer giving the array to be plotted. Corresponds to columns of R, G, Rb and Gb.
lim	character string indicating whether the red and green plots should have "separate" or "common" x- and y- co-ordinate limits.
pch	vector or list of plotting characters. Defaults to integer code 16.
cex	numeric vector of plot symbol expansions.
...	any other arguments are passed to plot

Details

A foreground-background plot is a plot of log2-foreground vs log2-background for a particular channel on a particular two-color array. This function produces a pair of plots, one for green and one for red, for a specified array.

See points for possible values for pch, col and cex.

Value

A plot is created on the current graphics device.

Author(s)

Gordon Smyth

See Also

An overview of diagnostic functions available in LIMMA is given in 09.Diagnostics.

plotlines *plotlines*

Description

Time course style plot of expression data.

Usage

```
plotlines(x,first.column.origin=FALSE,xlab="Column",ylab="x",col="black",lwd=1,.
```

Arguments

x	numeric matrix or object containing expression data.
first.column.origin	logical, should the lines be started from zero?
xlab	x-axis label
ylab	y-axis label
col	vector of colors for lines
lwd	line width multiplier
...	any other arguments are passed to plot

Details

Plots a line for each probe.

Value

A plot is created on the current graphics device.

Author(s)

Gordon Smyth

See Also

An overview of modeling functions and associated plots available in LIMMA is given in 06.Linear-Models.

plotMA *MA-Plot*

Description

Creates an MA-plot with color coding for control spots.

Usage

```
plotMA(MA, array=1, xlab="A", ylab="M", main=colnames(MA)[array], xlim=NULL, yli
```

Arguments

MA	an RGList, MAList or MArrayLM object, or any list with components M containing log-ratios and A containing average intensities. Alternatively a matrix or ExpressionSet object.
array	integer giving the array to be plotted. Corresponds to columns of M and A.
xlab	character string giving label for x-axis
ylab	character string giving label for y-axis
main	character string giving title for plot
xlim	numeric vector of length 2 giving limits for x-axis, defaults to min and max of the data
ylim	numeric vector of length 2 giving limits for y-axis, defaults to min and max of the data
status	character vector giving the control status of each spot on the array, of same length as the number of rows of MA$M. If omitted, all points are plotted in the default color, symbol and size.
values	character vector giving values of status to be highlighted on the plot. Defaults to unique values of status. Ignored if there is no status vector.
pch	vector or list of plotting characters. Default is integer code 16 which gives a solid circle. Ignored is there is no status vector.
col	numeric or character vector of colors, of the same length as values. Defaults to 1:length(values). Ignored if there is no status vector.
cex	numeric vector of plot symbol expansions, of the the same length as values. Defaults to 0.2 for the most common status value and 1 for the others. Ignored if there is no status vector.
legend	logical, should a legend of plotting symbols and colors be included. Ignored if there is no status vector.
zero.weights	logical, should spots with zero or negative weights be plotted?
...	any other arguments are passed to plot

Details

An MA-plot is a plot of log-intensity ratios (M-values) versus log-intensity averages (A-values). If MA is an RGList or MAList then this function produces an ordinary within-array MA-plot. If MA is an MArrayLM object, then the plot is an fitted model MA-plot in which the estimated coefficient is on the y-axis and the average A-value is on the x-axis.

If MA is a matrix or ExpressionSet object, then this function produces a between-array MA-plot. In this case the A-values in the plot are the average log-intensities across the arrays and the M-values are the deviations of the log-intensities for the specified array from the average. If there are more than five arays, then the average is computed robustly using medians. With five or fewer arrays, it is computed by means.

The status vector is intended to specify the control status of each spot, for example "gene", "ratio control", "house keeping gene", "buffer" and so on. The vector is usually computed using the function controlStatus and a spot-types file. However the function may be used to highlight any subset of spots.

The status can be included as the component MA$genes$Status instead of being passed as an argument to plotMA. The arguments values, pch, col and cex can be included as attributes to status instead of being passed as arguments to plotMA.

See points for possible values for pch, col and cex.

Value

A plot is created on the current graphics device.

Author(s)

Gordon Smyth

References

See http://www.statsci.org/micrarra/refs/maplots.html

See Also

An overview of diagnostic functions available in LIMMA is given in 09.Diagnostics.

Examples

```
MA <- new("MAList")
MA$A <- runif(300,4,16)
MA$M <- rt(300,df=3)
status <- rep("Gene",300)
status[1:3] <- "M=0"
MA$M[1:3] <- 0
status[4:6] <- "M=3"
MA$M[4:6] <- 3
status[7:9] <- "M=-3"
MA$M[7:9] <- -3
plotMA(MA,main="MA-Plot with Simulated Data",status=status,values=c("M=0","M=3","M=-3"),c

#  Same as above
attr(status,"values") <- c("M=0","M=3","M=-3")
attr(status,"col") <- c("blue","red","green")
plotMA(MA,main="MA-Plot with Simulated Data",status=status)
```

```
#  Same as above
MA$genes$Status <- status
plotMA(MA,main="MA-Plot with Simulated Data")
```

`plotMA3by2` *Write MA-Plots to Files*

Description

Write MA-plots to files in PNG format, six plots to a file in a 3 by 2 grid arrangement.

Usage

```
plotMA3by2(MA, prefix="MA", path=NULL, main=colnames(MA), zero.weights=FALSE, co
```

Arguments

`MA`	an `MAList` or `RGList` object, or any list with components `M` containing log-ratios and `A` containing average intensities
`prefix`	character string giving prefix to attach to file names
`path`	character string specifying directory for output files
`main`	character vector giving titles for plots
`zero.weights`	logical, should points with non-positive weights be plotted
`common.lim`	logical, should all plots on a page use the same axis limits
`device`	device driver for the plot. Choices are `"png"`, `"jpeg"`, `"pdf"`, `"postscript"`.
`...`	any other arguments are passed to `plotMA`

Details

This function writes a series of graphic files to disk. Each file contains six MA-plots in three rows and two columns. The layout is optimized for A4-sized paper.

The graph format can be `"png"` or `"jpeg"`, which are screen-resolution formats, or `"pdf"` or `"postscript"`, which are loss-less formats. `"png"` is not available on every R platform. Note that `"pdf"` or `"postscript"` may produce very large files.

Value

No value is returned, but one or more files are written to the working directory. The number of files is determined by the number of columns of `MA`.

Author(s)

Gordon Smyth

See Also

An overview of diagnostic functions available in LIMMA is given in 09.Diagnostics.

plotPrintTipLoess *MA Plots by Print-Tip Group*

Description

Creates a coplot giving MA-plots with loess curves by print-tip groups.

Usage

```
plotPrintTipLoess(object,layout,array=1,span=0.4,...)
```

Arguments

object	MAList or RGList object or list with components M containing log-ratios and A containing average intensities
layout	a list specifying the number of tip rows and columns and the number of spot rows and columns printed by each tip. Defaults to MA$printer if that is non-null.
array	integer giving the array to be plotted. Corresponds to columns of M and A.
span	span of window for lowess curve
...	other arguments passed to panel.smooth

Details

Note that spot quality weights in object are not used for computing the loess curves for this plot even though such weights would be used for loess normalization using normalizeWithinArrays.

Value

A plot is created on the current graphics device. If there are missing values in the data, then the vector of row numbers for spots with missing values is invisibly returned, as for coplot.

Author(s)

Gordon Smyth

See Also

An overview of diagnostic functions available in LIMMA is given in 09.Diagnostics.

poolVar *Pool Sample Variances with Unequal Variances*

Description

Compute the Satterthwaite (1946) approximation to the distribution of a weighted sum of sample variances.

Usage

```
poolVar(var, df=n-1, multiplier=1/n, n)
```

Arguments

var	numeric vector of independent sample variances
df	numeric vector of degrees of freedom for the sample variances
multiplier	numeric vector giving multipliers for the sample variances
n	numeric vector of sample sizes

Details

The sample variances `var` are assumed to follow scaled chi-square distributions. A scaled chi-square approximation is found for the distribution of `sum(multiplier * var)` by equating first and second moments. On output the sum to be approximated is equal to `multiplier * var` which follows approximately a scaled chisquare distribution on `df` degrees of freedom. The approximation was proposed by Satterthwaite (1946).

If there are only two groups and the degrees of freedom are one less than the sample sizes then this gives the denominator of Welch's t-test for unequal variances.

Value

A list with components

var	effective pooled sample variance
df	effective pooled degrees of freedom
multiplier	pooled multiplier

Author(s)

Gordon Smyth

References

Welch, B. L. (1938). The significance of the difference between two means when the population variances are unequal. *Biometrika* **29**, 350-362.

Satterthwaite, F. E. (1946). An approximate distribution of estimates of variance components. *Biometrics Bulletin* **2**, 110-114.

Welch, B. L. (1947). The generalization of 'Student's' problem when several different population variances are involved. *Biometrika* **34**, 28-35.

Welch, B. L. (1949). Further note on Mrs. Aspin's tables and on certain approximations to the tabled function. *Biometrika* **36**, 293-296.

See Also

10.Other

Examples

```
#  Welch's t-test with unequal variances
x <- rnorm(10,mean=1,sd=2)
y <- rnorm(20,mean=2,sd=1)
s2 <- c(var(x),var(y))
n <- c(10,20)
out <- poolVar(var=s2,n=n)
tstat <- (mean(x)-mean(y)) / sqrt(out$var*out$multiplier)
pvalue <- 2*pt(-abs(tstat),df=out$df)
#  Equivalent to t.test(x,y)
```

---

printHead *Print Leading Rows of Large Objects*

Description

Print the leading rows of a large vector, matrix or data.frame. This function is used by `show` methods for data classes defined in LIMMA.

Usage

```
printHead(x)
```

Arguments

`x`	any object

Details

If `x` is a vector with more than 20 elements, then `printHead(x)` prints only the first 5 elements. If `x` is a matrix or data.frame with more than 10 rows, then `printHead(x)` prints only the first 5 rows. Any other type of object is printed normally.

Author(s)

Gordon Smyth

See Also

An overview of classes defined in LIMMA is given in 02.Classes

printorder *Identify Order in which Spots were Printed*

Description

Identify order in which spots were printed and the 384-well plate from which they were printed.

Usage

```
printorder(layout, ndups=1, spacing="columns", npins, start="topleft")
```

Arguments

layout	list with the components ngrid.r, ngrid.c, nspot.r and nspot.c, or an RGList or MAList object from which the printer layout may be extracted.
ndups	number of duplicate spots, i.e., number of times print-head dips into each well
spacing	character string indicating layout of duplicate spots. Choices are "columns", "rows" or "topbottom".
npins	actual number of pins or tips on the print-head
start	character string giving position of the spot printed first in each grid. Choices are "topleft" or "topright" and partial matches are accepted.

Details

In most cases the printer-head contains the layout$ngrid.r times layout$ngrid.c pins or tips and the array is printed using layout$nspot.r times layout$npot.c dips of the head. The plate holding the DNA to be printed is assumed to have 384 wells in 16 rows and 24 columns.

ndups indicates the number of spots printed from each well. The replicate spots from multiple dips into the same wells are assumed to be side-by-side by columns (spacing="columns"), by rows (spacing="rows") or in the top and bottom halves of the array (spacing="topbottom").

In some cases a smaller number of physical pins is used and the total number of grids is built up by effectively printing two or more sub-arrays on the same slide. In this case the number of grids should be a multiple of the number of pins.

Printing is assumed to proceed by rows within in each grid starting either from the top-left or the top-right.

Value

List with components

printorder	numeric vector giving printorder of each spot, i.e., which dip of the print-head was used to print it
plate	numeric vector giving plate number from which each spot was printed
plate.r	numeric vector giving plate-row number of the well from which each spot was printed
plate.c	numeric vector giving plate-column number of the well from which each spot was printed

plateposition
: character vector summarizing plate number and plate position of the well from which each spot was printed with letters for plate rows and number for columns. For example 02B13 is second row, 13th column, of the second plate.

Author(s)

Gordon Smyth

See Also

normalizeForPrintorder.

An overview of LIMMA functions for reading data is given in 03.ReadingData.

Examples

```
printorder(list(ngrid.r=2,ngrid.c=2,nspot.r=12,nspot.c=8))
```

---

printtipWeights *Sub-array Quality Weights*

Description

Estimates relative quality weights for each sub-array in a multi-array experiment.

Usage

```
printtipWeights(object, design = NULL, weights = NULL, method = "genebygene", la
```

Arguments

object	object of class numeric, matrix, MAList, marrayNorm, or ExpressionSet containing log-ratios or log-values of expression for a series of spotted microarrays.
design	the design matrix of the microarray experiment, with rows corresponding to arrays and columns to coefficients to be estimated. Defaults to the unit vector meaning that the arrays are treated as replicates.
weights	optional numeric matrix containing prior weights for each spot.
method	character string specifying the estimating algorithm to be used. Choices are "genebygene" and "reml".
layout	list specifying the dimensions of the spot matrix and the grid matrix. For details see PrintLayout-class.
maxiter	maximum number of iterations allowed.
tol	convergence tolerance.
trace	logical variable. If true then output diagnostic information at each iteration of '"reml"' algorithm.

Details

The relative reliability of each sub-array (print-tip group) is estimated by measuring how well the expression values for that sub-array follow the linear model.

The method described in Ritchie et al (2006) and implemented in the `arrayWeights` function is adapted for this purpose.A heteroscedastic model is fitted to the expression values for each gene by calling the function `lm.wfit`. The dispersion model is fitted to the squared residuals from the mean fit, and is set up to have sub-array specific coefficients, which are updated in either full REML scoring iterations, or using an efficient gene-by-gene update algorithm. The final estimates of the sub-array variances are converted to weights.

The data object `object` is interpreted as for `lmFit`. In particular, the arguments `design`, `weights` and `layout` will be extracted from the data `object` if available and do not normally need to be set explicitly in the call; if any of these are set in the call then they will over-ride the slots or components in the data `object`.

Value

A matrix of sub-array weights which can be passed to `lmFit`.

Author(s)

Matthew Ritchie and Gordon Smyth

References

Ritchie, M. E., Diyagama, D., Neilson, van Laar, R., J., Dobrovic, A., Holloway, A., and Smyth, G. K. (2006). Empirical array quality weights in the analysis of microarray data. BMC Bioinformatics 7, 261. http://www.biomedcentral.com/1471-2105/7/261/abstract

See Also

An overview of linear model functions in limma is given by 06.LinearModels.

Examples

```
library(sma)
# Subset of data from ApoAI case study in Limma User's Guide
data(MouseArray)
# Avoid non-positive intensities
RG <- backgroundCorrect(mouse.data, method="half")
MA <- normalizeWithinArrays(RG, mouse.setup)
MA <- normalizeBetweenArrays(MA, method="Aq")
targets <- data.frame(Cy3=I(rep("Pool",6)),Cy5=I(c("WT","WT","WT","KO","KO","KO")))
design <- modelMatrix(targets, ref="Pool")
subarrayw <- printtipWeights(MA, design, layout=mouse.setup)
fit <- lmFit(MA, design, weights=subarrayw)
fit2 <- contrasts.fit(fit, contrasts=c(-1,1))
fit2 <- eBayes(fit2)
# Use of sub-array weights increases the significance of the top genes
topTable(fit2)
# Create an image plot of sub-array weights from each array
zlim <- c(min(subarrayw), max(subarrayw))
par(mfrow=c(3,2), mai=c(0.1,0.1,0.3,0.1))
for(i in 1:6)
        imageplot(subarrayw[,i], layout=mouse.setup, zlim=zlim, main=paste("Array", i))
```

protectMetachar *Protect Metacharacters*

Description

Add backslashes before any metacharacters found in a string.

Usage

```
protectMetachar(x)
```

Arguments

x	character vector

Details

This function is used to protect strings containing metacharacters so that the metacharacters can be treated as ordinary characters in string matching functions operations.

Value

A character vector of the same length as x in which two backslashes have been inserted before any metacharacter.

Author(s)

Gordon Smyth

See Also

An overview of LIMMA functions for reading data is given in 03.ReadingData.

Examples

```
# without protectMetachar, this would be no match
grep(protectMetachar("Ch1 (mean)"),"Ch1 (mean)")
```

qqt *Student's t Quantile-Quantile Plot*

Description

Plots the quantiles of a data sample against the theoretical quantiles of a Student's t distribution.

Usage

```
qqt(y, df = Inf, ylim = range(y), main = "Student's t Q-Q Plot",
    xlab = "Theoretical Quantiles", ylab = "Sample Quantiles", plot.it = TRUE, .
```

Arguments

y	a numeric vector or array containing the data sample
df	degrees of freedom for the t-distribution. The default df=Inf represents the normal distribution.
ylim	plotting range for y
main	main title for the plot
xlab	x-axis title for the plot
ylab	y-axis title for the plot
plot.it	whether or not to produce a plot
...	other arguments to be passed to plot

Details

This function is analogous to qqnorm for normal probability plots. In fact qqt(y,df=Inf) is identical to qqnorm(y) in all respects except the default title on the plot.

Value

A list is invisibly returned containing the values plotted in the QQ-plot:

x	theoretical quantiles of the t-distribution
y	the data sample, same as input y

Author(s)

Gordon Smyth

See Also

qqnorm

Examples

```
#  See also the lmFit examples

y <- rt(50,df=4)
qqt(y,df=4)
abline(0,1)
```

QualityWeights *Spot Quality Weights*

Description

Functions to calculate quality weights for individual spots based on image analyis output file.

Usage

```
wtarea(ideal=c(160,170))
wtflags(weight=0,cutoff=0)
wtIgnore.Filter
```

Arguments

`ideal`	numeric vector giving the ideal area or range of areas for a spot in pixels
`weight`	weight to be given to flagged spots
`cutoff`	cutoff value for `Flags` below which spots will be downweighted

Details

These functions can be passed as an argument to `read.maimages` to construct quality weights as the microarray data is read in.

`wtarea` downweights unusually small or large spots and is designed for SPOT output. It gives weight 1 to spots which have areas in the ideal range, given in pixels, and linearly downweights spots which are smaller or larger than this range.

`wtflags` is designed for GenePix output and gives the specified weight to spots with `Flags` value less than the `cutoff` value. Choose `cutoff=0` to downweight all flagged spots. Choose `cutoff=-50` to downweight bad or absent spots or `cutoff=-75` to downweight only spots which have been manually flagged as bad.

`wtIgnore.Filter` is designed for QuantArray output and sets the weights equal to the column `Ignore Filter` produced by QuantArray. These weights are 0 for spots to be ignored and 1 otherwise.

Value

A function which takes a dataframe or matrix as argument and produces a numeric vector of weights between 0 and 1

Author(s)

Gordon Smyth

See Also

An overview of LIMMA functions for reading data is given in 03.ReadingData.

Examples

```
#  Read in spot output files from current directory and give full weight to 165
#  pixel spots.  Note: for this example to run you must set fnames to the names
#  of actual spot output files (data not provided).
## Not run:
RG <- read.maimages(fnames,source="spot",wt.fun=wtarea(165))
#  Spot will be downweighted according to weights found in RG
MA <- normalizeWithinArrays(RG,layout)
## End(Not run)
```

`read.columns` *Read specified columns from a file*

Description

Reads specified columns from a file in table format and creates a data frame from it, with cases corresponding to lines and variables to fields in the file.

Usage

```
read.columns(file,required.col=NULL,text.to.search="",sep="\t",quote="\"",skip=0
```

Arguments

`file`	the name of the file which the data are to be read from.
`required.col`	character vector of names of the required columns
`text.to.search`	character string. If any column names can be found in this string, those columns will also be read.
`sep`	the field separator character
`quote`	character string of characters to be treated as quote marks
`skip`	the number of lines of the data file to skip before beginning to read data.
`fill`	logical: if `TRUE` then in case the rows have unequal length, blank fields are implicitly added.
`blank.lines.skip`	logical: if `TRUE` blank lines in the input are ignored.
`comment.char`	character: a character vector of length one containing a single character or an empty string.
`allowEscapes`	logical. Should C-style escapes such as `\n` be processed or read verbatim (the default)?
`...`	other arguments are passed to `read.table`, excluding the following which are reserved and cannot be set by the user: `header`, `col.names`, `check.names` and `colClasses`.

Details

This function is an interface to `read.table` in the base package. It uses `required.col` and `text.to.search` to set up the `colClasses` argument of `read.table`.

Note the following arguments of `read.table` are used by `read.columns` and therefore cannot be set by the user: `header`, `col.names`, `check.names` and `colClasses`.

This function is used by `read.maimages`.

Value

A data frame (data.frame) containing a representation of the data in the file.

Author(s)

Gordon Smyth

See Also

read.maimages, read.table.

An overview of LIMMA functions for reading data is given in 03.ReadingData.

---

read.maimages *Read RGList from Image Analysis Output Files*

Description

Reads an RGList from a series of two-color microarray image analysis output files

Usage

```
read.maimages(files=NULL,source="generic",path=NULL,ext=NULL,names=NULL,columns=
read.imagene(files,path=NULL,ext=NULL,names=NULL,columns=NULL,other.columns=NULL
```

Arguments

files	character vector giving the names of the files containing image analysis output or, for Imagene data, a character matrix of names of files. If omitted, then all files with extension ext in the specified directory will be read in alphabetical order.
source	character string specifying the image analysis program which produced the output files. Choices are "generic", "agilent", "arrayvision", arrayvision.ARM", arrayvision.MTM", "bluefuse", "genepix", "genepix.custom", "genepix.median", "imagene", "quantarray", "scanarrayexpress", "smd.old", "smd", "spot" or "spot.close.open".
path	character string giving the directory containing the files. The default is the current working directory.
ext	character string giving optional extension to be added to each file name
names	character vector of names to be associated with each array as column name. Defaults to removeExt(files).
columns	list with fields R, G, Rb and Gb giving the column names to be used for red and green foreground and background or, in the case of Imagene data, a list with fields f and b. This argument is optional if source is specified, otherwise it is required.
other.columns	character vector of names of other columns to be read containing spot-specific information
annotation	character vector of names of columns containing annotation information about the probes
wt.fun	function to calculate spot quality weights
verbose	logical, TRUE to report each time a file is read
sep	the field separator character
quote	character string of characters to be treated as quote marks
...	any other arguments are passed to read.table

Details

This is the main data input function for the LIMMA package for two-color microarray data. It extracts the foreground and background intensities from a series of files, produced by an image analysis program, and assembles them into the components of one list. The image analysis programs Agilent Feature Extraction, ArrayVision, BlueFuse, GenePix, ImaGene, QuantArray (Version 3 or later), Stanford Microarray Database (SMD) and SPOT are supported explicitly. Data from some other image analysis programs can be read if the appropriate column names containing the foreground and background intensities are specified using the `columns` argument. (This will work if the column names are unique and if there are no rows in the file after the last line of data. Header lines are ok.)

SMD data should consist of raw data files from the database, in tab-delimited text form. There are two possible sets of column names depending on whether the data was entered into the database before or after September 2003. `source="smd.old"` indicates that column headings in use prior to September 2003 should be used. In the case of GenePix, two possible foreground estimators are supported: `source="genepix"` uses the mean foreground estimates while `source="genepix.median"` uses median foreground estimates. GenePix 6.0 and later also supplies some custom background options, notably morphological background. If the GPR files have been written using a custom background, you may read it using `source="genepix.custom"`. In the case of SPOT, two possible background estimators are supported: if `source="spot.close.open"` then background intensities are estimated from `morph.close.open` rather than `morph`.

Spot quality weights may be extracted from the image analysis files using a weight function wt.fun. `wt.fun` may be any user-supplied function which accepts a data.frame argument and returns a vector of non-negative weights. The columns of the data.frame are as in the image analysis output files. There is one restriction, which is that the column names should be refered to in full form in the weight function, i.e., do not rely on name expansion for partial matches when refering to the names of the columns. See `QualityWeights` for suggested weight functions.

For Imagene image data the argument `files` should be a matrix with two columns. The first column should contain the names of the files containing green channel (cy3) data and the second column should contain names of files containing red channel (cy5) data. If `source="imagene"` and `files` is a vector of even length instead of a matrix, then each consecutive pair of file names is assumed to correspond to the same array. The function `read.imagene` is called by `read.maimages` when `source="imagene"`. It does not need to be called directly by users.

ArrayVision reports spot intensities in a number of different ways. `read.maimages` caters for ArrayVision's Artifact-removed (ARM) density values as `"arrayvision.ARM"` or for Median-based Trimmed Mean (MTM) density values as `"arrayvision.MTM"`. ArrayVision users may find it useful to read the top two lines of their data file to check which version of density values they have.

The argument `other.columns` allows arbitrary columns of the image analysis output files to be preserved in the data object. These become matrices in the component `other` component. For ImaGene data, the other column headings with be prefixed with `"R "` or `"G "` as appropriate.

Value

An `RGList` object containing the components

`R`	matrix containing the red channel foreground intensities for each spot for each array.
`Rb`	matrix containing the red channel background intensities for each spot for each array.
`G`	matrix containing the green channel foreground intensities for each spot for each array.

Gb	matrix containing the green channel background intensities for each spot for each array.
weights	spot quality weights, if wt.fun is given
other	list containing matrices corresponding to other.columns if given
genes	data frame containing annotation information about the probes, for example gene names and IDs and spatial positions on the array, currently set only if source is "agilent", "genepix" or source="imagene" or if the annotation argument is set
targets	data frame with column FileName giving the names of the files read
source	character string giving the image analysis program name
printer	list of class PrintLayout, currently set only if source="imagene"

Warnings

All image analysis files being read are assumed to contain data for the same genelist in the same order. No checking is done to confirm that this is true. Probe annotation information is read from the first file only.

Author(s)

Gordon Smyth, with speed improvements by Marcus Davy

References

Web pages for the image analysis software packages mentioned here are listed at http://www.statsci.org/micrarra/image.html

See Also

read.maimages is based on read.table in the base package. read.marrayRaw is the corresponding function in the marray package.

An overview of LIMMA functions for reading data is given in 03.ReadingData.

Examples

```
#  Read all .gpr files from current working directory
#  and give weight 0.1 to spots with negative flags

## Not run:
files <- dir(pattern="*\\.gpr$")
RG <- read.maimages(files,"genepix",wt.fun=wtflags(0.1))
## End(Not run)

#  Read all .spot files from current working director and down-weight
#  spots smaller or larger than 150 pixels

## Not run:
files <- dir(pattern="*\\.spot$")
RG <- read.maimages(files,"spot",wt.fun=wtarea(150))
## End(Not run)
```

`readHeader` *Read Header Information from Image Analysis Raw Data File*

Description

Read the header information from a GenePix Results (GPR) file or from an SMD raw data file. These functions are used internally by `read.maimages` and are not usually called directly by users.

Usage

```
readGenericHeader(file, columns, sep="\t")
readGPRHeader(file)
readSMDHeader(file)
```

Arguments

`file`	character string giving file name. If it does not contain an absolute path, the file name is relative to the current working directory.
`columns`	character vector specifying data column headings expected to be in file
`sep`	the character string separating column names

Details

Raw data files exported by image analysis programs include a number of header lines which contain information about the scanning process. This function extracts that information and locates the line where the intensity data begins. `readGPRHeader` is for GenePix output and `readSMDHeader` is for files from the Stanford Microarray Database (SMD). `readGenericHeader` finds the line in the file on which the data begins by searching for specified column headings.

Value

A list with components corresponds to lines of header information. A key component is `NHeaderRecords` which gives the number of lines in the file before the intensity data begins. All other components are character vectors.

Author(s)

Gordon Smyth

References

See `http://www.axon.com/gn_GenePix_File_Formats.html` for GenePix formats.

See `http://www.bluegnome.co.uk` for information on BlueFuse.

See `http://genome-www.stanford.edu/Microarray` for the SMD.

See Also

`read.maimages`

An overview of LIMMA functions to read data is given in 03.ReadingData.

`readImaGeneHeader` *Read ImaGene Header Information*

Description

Read the header information from an ImaGene image analysis output file. This function is used internally by `read.maimages` and is not usually called directly by users.

Usage

```
readImaGeneHeader(file)
```

Arguments

`file`	character string giving file name or path

Details

The raw data files exported by the image analysis software ImaGene include a number of header lines which contain information about the printing and scanning processes. This function extracts that information and locates the line where the intensity data begins.

Value

A list containing information read from the header of the ImaGene file. Each Begin-End environment found in the file header will become a recursive list in the output object, with components corresponding to fields in the file. See the ImaGene documentation for further information. The output object will also contain a component `NHeaderRecords`giving the number of lines in the file before the intensity data begins.

Author(s)

Gordon Smyth

References

http://www.biodiscovery.com/imagene.asp

See Also

`read.imagene`

An overview of LIMMA functions to read data is given in 03.ReadingData.

Examples

```
## Not run:
h <- readImaGeneHeader("myImaGenefile.txt")
names(h)
h$NHeaderRecords
h[["Field Dimensions"]]
## End(Not run)
```

readSpotTypes *Read Spot Types File*

Description

Read a table giving regular expressions to identify different types of spots in the gene-dataframe.

Usage

```
readSpotTypes(file="SpotTypes.txt",path=NULL,sep="\t",check.names=FALSE,...)
```

Arguments

file	character string giving the name of the file specifying the spot types.
path	character string giving the directory containing the file. Can be omitted if the file is in the current working irectory.
sep	the field separator character
check.names	logical, if FALSE column names will not be converted to valid variable names, for example spaces in column names will not be left as is
...	any other arguments are passed to read.table

Details

The file is a text file with rows corresponding to types of spots and the following columns: SpotType gives the name for the spot type, ID is a regular expression matching the ID column, Name is a regular expression matching the Name column, and Color is the R name for the color to be associated with this type.

Value

A data frame with columns

SpotType	character vector giving names of the spot types
ID	character vector giving regular expressions
Name	character vector giving regular expressions
Color	character vector giving names of colors

Author(s)

Gordon Smyth following idea of James Wettenhall

See Also

An overview of LIMMA functions for reading data is given in 03.ReadingData.

`readTargets` *Read Targets File*

Description

Read targets file for a microarray experiment into a dataframe.

Usage

```
readTargets(file="Targets.txt", path=NULL, sep="\t", row.names=NULL, quote="\"",
```

Arguments

`file`	character string giving the name of the targets file.
`path`	character string giving the directory containing the file. Can be omitted if the file is in the current working irectory.
`sep`	field separator character
`row.names`	character string giving the name of a column from which to obtain row names
`quote`	the set of quoting characters
`...`	other arguments are passed to `read.table`

Details

The targets file is a text file containing information about the RNA samples used as targets in the microarray experiment. Rows correspond to arrays and columns to covariates associated with the targets. For a two-color experiment, the targets file will normally include columns labelled `Cy3` and `Cy5` or similar specifying which RNA samples are hybridized to each channel of each array. Other columns may contain any other covariate information associated with the arrays or targets used in the experiment.

If `row.names` is non-null and there is a column by that name with unique values, then those values will be used as row names for the dataframe. If `row.names` is null, then the column `Labels` will be used if such exists or, failing that, the column `FileName`.

See the Limma User's Guide for examples of this function.

Value

A dataframe. Character columns are not converted into factors.

Author(s)

Gordon Smyth

See Also

An overview of LIMMA functions for reading data is given in 03.ReadingData.

readGAL *Read a GAL file*

Description

Read a GenePix Array List (GAL) file into a dataframe.

Usage

```
readGAL(galfile=NULL,path=NULL,header=TRUE,sep="\t",quote="\"",skip=NULL,as.is=T
```

Arguments

galfile	character string giving the name of the GAL file. If NULL then a file with extension .gal is found in the directory specified by path.
path	character string giving the directory containing the files. If NULL then assumed to be the current working directory.
header	logical variable, if TRUE then the first line after skip is assumed to contain column headings. If FALSE then a value should specified for skip.
sep	the field separator character
quote	the set of quoting characters
skip	number of lines of the GAL file to skip before reading data. If NULL then this number is determined by searching the file for column headings.
as.is	logical variable, if TRUE then read in character columns as vectors rather than factors.
...	any other arguments are passed to read.table

Details

A GAL file is a list of genes IDs and associated information produced by an Axon microarray scanner. Apart from header information, the file must contain data columns labeled Block, Column, Row and ID. A Name column is usually included as well. Other columns are optional. See the Axon URL below for a detaile description of the GAL file format.

This function reads in the data columns with a minimum of user information. In most cases the function can be used without specifying any of the arguments.

Value

A data frame with columns

Block	numeric vector containing the print tip indices
Column	numeric vector containing the spot columns
Row	numeric vector containing the spot rows
ID	character vector, for factor if as.is=FALSE, containing gene library identifiers
Name	character vector, for factor if as.is=FALSE, containing gene names

The data frame will be sorted so that Column is the fastest moving index, then Row, then Block.

Author(s)

Gordon Smyth

References

http://www.axon.com/gn_GenePix_File_Formats.html

See Also

An overview of LIMMA functions for reading data is given in 03.ReadingData.

Examples

```
# readGAL()
# will read in the first GAL file (with suffix ".gal")
# found in the current working directory
```

---

removeBatchEffect *Remove Batch Effect*

Description

Remove a batch effect from expression data.

Usage

```
removeBatchEffect(x,batch,design=NULL)
```

Arguments

x	numeric matrix containing log-expression intensity for a series of microarrays. Each column corresponds to an array.
batch	a factor or vector indicating batches.
design	an optional design matrix

Details

This function is useful for removing batch effects, associated with hybridization time or other technical variables, prior to clustering or unsupervised analysis.

This function is intended for use with clustering or PCA, not for use prior to linear modelling. If linear modelling is intended, it is better to include the batch effect as part of the linear model.

Value

A numeric matrix of log-expression values with batch effects removed.

Author(s)

Gordon Smyth and Carolyn de Graaf

See Also

05.Normalization

removeExt *Remove Common Extension from File Names*

Description

Finds and removes any common extension from a vector of file names.

Usage

```
removeExt(x)
```

Arguments

x	character vector

Value

A character vector of the same length as x in which any common extension has been stripped off.

Author(s)

Gordon Smyth

See Also

An overview of LIMMA functions for reading data is given in 03.ReadingData.

Examples

```
x <- c("slide1.spot","slide2.spot","slide3.spot")
removeExt(x)
```

residuals.MArrayLM *Extract Residuals from MArrayLM Fit*

Description

This method extracts the residuals from all the probewise linear model fits and returns them in a matrix.

Usage

```
## S3 method for class 'MArrayLM':
residuals(object, y, ...)
```

Arguments

object	a fitted model object inheriting from class MarrayLM.
y	a data object containing the response data used to compute the fit. This can be of any class for which as.matrix is defined, including MAList, ExpressionSet, marrayNorm etc.
...	other arguments are not used

Value

Numeric matrix of residuals.

See Also

residuals.

RGList-class *Red, Green Intensity List - class*

Description

A simple list-based class for storing red and green channel foreground and background intensities for a batch of spotted microarrays. RGList objects are normally created by read.maimages.

Slots/List Components

RGList objects can be created by new("RGList",RG) where RG is a list. Objects of this class contains no slots (other than .Data), but objects should contain the following list components:

R: numeric matrix containing the red (cy5) foreground intensities. Rows correspond to spots and columns to arrays.
G: numeric matrix containing the green (cy3) foreground intensities. Rows correspond to spots and columns to arrays.

Optional components include

Rb: numeric matrix containing the red (cy5) background intensities
Gb: numeric matrix containing the green (cy3) background intensities
weights: numeric matrix of same dimension as R containing relative spot quality weights. Elements should be non-neg
other: list containing numeric matrices of other spot-specific information.
genes: data.frame containing probe information. Should have one row for each spot. May have any number of colun
targets: data.frame containing information on the target RNA samples. Rows correspond to arrays. May have any nu
printer: list containing information on the process used to print the spots on the arrays. See PrintLayout.

Valid RGList objects may contain other optional components, but all probe or array information should be contained in the above components.

Methods

This class inherits directly from class list so any operation appropriate for lists will work on objects of this class. In addition, RGList objects can be subsetted, combined and merged. RGList objects will return dimensions and hence functions such as dim, nrow and ncol are defined. RGLists also inherit a show method from the virtual class LargeDataObject, which means that RGLists will print in a compact way.

RGList objects can be converted to exprSet2 objects by as(RG,"exprSet2").

Other functions in LIMMA which operate on RGList objects include normalizeBetweenArrays, normalizeForPrintorder, normalizeWithinArrays.

Author(s)

Gordon Smyth

See Also

02.Classes gives an overview of all the classes defined by this package.

marrayRaw-class is the corresponding class in the marrayClasses package.

---

roast *roast*

Description

Rotation gene set testing for linear models.

Usage

```
roast(iset=NULL,y,design,contrast=ncol(design),gene.weights=NULL,array.weights=N
```

Arguments

iset	vector specifying the rows of y in the test set. This can be a vector of indices, or a logical vector of the same length as statistics, or any vector such as y[selected,] contains the values for the gene set to be tested.
y	numeric matrix giving log-expression values. If var.prior or df.prior are null, then y should contain values for all genes on the arrays. If both prior parameters are given, then only y values for the test set are required.
design	design matrix
contrast	contrast for which the test is required. Can be an integer specifying a column of design, or else a contrast vector of length equal to the number of columns of design.
gene.weights	optional numeric vector of weights for genes in the set.
array.weights	optional numeric vector of array weights.
block	optional vector of blocks.
correlation	correlation between blocks.
var.prior	prior value for residual variances. If not provided, this is estimated from all the data using squeezeVar.
df.prior	prior degrees of freedom for residual variances. If not provided, this is estimated using squeezeVar.
nrot	number of rotations used to estimate the p-values.

Details

This function tests whether any of the genes in the set are differentially expressed. It uses rotation, which is a smoothed version of permutation suitable for linear models (Langsrud, 2005). It can be used for any linear model with replication. and negative values and otherwise will be taken to be F-like.

This is a self-contained test is the sense that genes outside the test set do not play a role (Goeman, JJ, and Buhlmann P, 2007). A competitive gene set test is performed by geneSetTest.

p-values are given for four possible alternative hypotheses. `alternative=="up"` means the genes in the set tend to be up-regulated, with positive t-statistics. `alternative=="down"` means the genes in the set tend to be down-regulated, with negative t-statistics. `alternative=="either"` means the set is either up or down-regulated as a whole. `alternative=="mixed"` test whether the genes in the set tend to be differentially expressed, without regard for direction. In this case, the test will be significant if the set contains mostly large test statistics, even if some are positive and some are negative.

The first three alternatives are appropriate if you have a prior expection that all the genes in the set will react in the same direction. The `"mixed"` alternative is appropriate if you know only that the genes are involved in the relevant pathways, without knowing the direction of effect for each gene. The `"mixed"` alternative is the only one possible with F-like statistics.

Note that `roast` estimates p-values by simulation, specifically by random rotations of the orthogonalized residuals. This means that the p-values will vary slightly from run to run. To get more precise p-values, increase the number of rotations `nrot`. The strategy of random rotations is due to Langsrud (2005).

Value

data.frame with columns `Z`, `Active` and `P.Value`. The `Z` column gives average (root mean square) z-statistics for the genes in the set. The `Active` gives the proportion of genes in the set contributing meaningfully to significance, defined as those with squared z-values greater than 2. The `P.Value` gives estimated p-values. The rows correspond to the alternative hypotheses mixed, up, down or either.

Author(s)

Gordon Smyth and Di Wu

References

Goeman, JJ, and Buhlmann P, 2007. Analyzing gene expression data in terms of gene sets: methodological issues. *Bioinformatics* 23, 980-987.

Langsrud, O. (2005). Rotation tests. *Statistics and Computing* 15, 53-60

See Also

`geneSetTest`

Examples

```
y <- matrix(rnorm(100*4),100,4)
design <- cbind(Intercept=1,Group=c(0,0,1,1))
iset <- 1:5
y[iset,3:4] <- y[iset,3:4]+3
roast(iset,y,design,contrast=2)

# Alternative approach useful if multiple gene sets are tested:
fit <- lmFit(y,design)
sv <- squeezeVar(fit$sigma^2,df=fit$df.residual)
iset1 <- 1:5
iset2 <- 6:10
roast(y=y[iset1,],design=design,contrast=2,var.prior=sv$var.prior,df.prior=sv$var.prior)
roast(y=y[iset2,],design=design,contrast=2,var.prior=sv$var.prior,df.prior=sv$var.prior)
```

`selectModel` *Select Appropriate Linear Model*

Description

Select the best fitting linear model for each gene by minimizing an information criterion.

Usage

```
selectModel(y, designlist, criterion="aic", df.prior=0, s2.prior=NULL, s2.true=N
```

Arguments

`y`	a matrix-like data object, containing log-ratios or log-values of expression for a series of microarrays. Any object class which can be coerced to matrix is acceptable including `numeric`, `matrix`, `MAList`, `marrayNorm`, `ExpressionSet` or `PLMset`.
`designlist`	list of design matrices
`criterion`	information criterion to be used for model selection, `"aic"`, `"bic"` or `"mallowscp"`.
`df.prior`	prior degrees of freedom for residual variances. See `squeezeVar`
`s2.prior`	prior value for residual variances, to be used if `df.prior`>0.
`s2.true`	numeric vector of true variances, to be used if `criterion="mallowscp"`.
`...`	other optional arguments to be passed to `lmFit`

Details

This function chooses, for each probe, the best fitting model out of a set of alternative models represented by a list of design matrices. Selection is by Akaike's Information Criterion (AIC), Bayesian Information Criterion (BIC) or by Mallow's Cp.

The criteria have been generalized slightly to accommodate an information prior on the variances represented by `s2.prior` and `df.prior` or by `s2.post`. Suitable values for these parameters can be estimated using `squeezeVar`.

Value

List with components

`IC`	matrix of information criterion scores, rows for probes and columns for models
`pref`	factor indicating the model with best (lowest) information criterion score

Author(s)

Alicia Oshlack and Gordon Smyth

See Also

An overview of linear model functions in limma is given by 06.LinearModels.

Examples

```
nprobes <- 100
narrays <- 5
y <- matrix(rnorm(nprobes*narrays),nprobes,narrays)
A <- c(0,0,1,1,1)
B <- c(0,1,0,1,1)
designlist <- list(
  None=cbind(Int=c(1,1,1,1,1)),
  A=cbind(Int=1,A=A),
  B=cbind(Int=1,B=B),
  Both=cbind(Int=1,AB=A*B),
  Add=cbind(Int=1,A=A,B=B),
  Full=cbind(Int=1,A=A,B=B,AB=A*B)
)
out <- selectModel(y,designlist)
table(out$pref)
```

---

squeezeVar *Smooth Sample Variances*

Description

Smooth a set of sample variances by computing empirical Bayes posterior means.

Usage

```
squeezeVar(var, df)
```

Arguments

var	numeric vector of independent sample variances
df	numeric vector of degrees of freedom for the sample variances

Details

The sample variances var are assumed to follow scaled chi-squared distributions. An inverse chi-squared prior is assumed for the true variances. The scale and degrees of freedom for the prior distribution are estimated from the data.

The effect of this function is to smooth or shrink the variances towards a common value. The smoothed variances have a smaller expected mean square error to the true variances than do the sample variances themselves.

This function is called by eBayes, but beware a possible confusion with the output from that function. The values var.prior and *var.post* output by squeezeVar correspond to the quantities s2.prior and s2.post output by eBayes, whereas var.prior output by eBayes relates to a different parameter.

Value

A list with components

var.post	numeric vector of posterior variances
var.prior	location of prior distribution
df.prior	degrees of freedom of prior distribution

Author(s)

Gordon Smyth

References

Smyth, G. K. (2004). Linear models and empirical Bayes methods for assessing differential expression in microarray experiments. *Statistical Applications in Genetics and Molecular Biology*, **3**, No. 1, Article 3. http://www.bepress.com/sagmb/vol3/iss1/art3

Examples

```
s2 <- rchisq(20,df=5)/5
squeezeVar(s2, df=5)
```

---

strsplit2 *Split Composite Names*

Description

Split a vector of composite names into a matrix of simple names.

Usage

```
strsplit2(x, split, extended = TRUE, fixed = FALSE, perl = FALSE)
```

Arguments

x	character vector
split	character to split each element of vector on, see strsplit
extended	logical. If TRUE, extended regular expression matching is used, see strsplit.
fixed	logical. If TRUE match string exactly, otherwise use regular expressions. Has priority over perl and extended.
perl	logical. Should perl-compatible regexps be used? Has priority over extended.

Details

This function is the same as strsplit except that the output value is a matrix instead of a list. The first column of the matrix contains the first component from each element of x, the second column contains the second components etc. The number of columns is equal to the maximum number of components for any element of x.

Value

A list containing components

Name	character vector of the same length as x contain first splits of each element
Annotation	character vector of the same length as x contain second splits of each element

Author(s)

Gordon Smyth

See Also

strsplit.

An overview of LIMMA functions for reading data is given in 03.ReadingData.

Examples

```
x <- c("AA196000;actinin, alpha 3",
"AA464163;acyl-Coenzyme A dehydrogenase, very long chain",
"3E7;W15277;No Annotation")
strsplit2(x,split=";")
```

---

subsetting *Subset RGList, MAList or MArrayLM Objects*

Description

Extract a subset of an RGList, MAList or MArrayLM object.

Usage

```
## S3 method for class 'RGList':
object[i, j, ...]
```

Arguments

object	object of class RGList, MAList or MArrayLM
i,j	elements to extract. i subsets the genes or spots while j subsets the arrays
...	not used

Details

i,j may take any values acceptable for the matrix components of object. See the Extract help entry for more details on subsetting matrices.

Value

An object of the same class as object holding data from the specified subset of genes and arrays.

Author(s)

Gordon Smyth

See Also

Extract in the base package.

03.ReadingData gives an overview of data input and manipulation functions in LIMMA.

Examples

```
M <- A <- matrix(11:14,4,2)
rownames(M) <- rownames(A) <- c("a","b","c","d")
colnames(M) <- colnames(A) <- c("A","B")
MA <- new("MAList",list(M=M,A=A))
MA[1:2,]
MA[1:2,2]
MA[,2]
```

`summary` *Summaries of Microarray Data Objects*

Description

Briefly summarize microarray data objects.

Usage

```
## S3 method for class 'RGList':
summary(object, ...)
```

Arguments

`object`	an object of class `RGList`, `MAList` or `MArrayLM`
`...`	other arguments are not used

Details

The data objects are summarized as if they were lists, i.e., brief information about the length and type of the components is given.

Value

A table.

Author(s)

Gordon Smyth

See Also

`summary` in the base package.

02.Classes gives an overview of data classes used in LIMMA.

`targetsA2C` *Convert Two-Color Targets Dataframe from One-Row-Per-Array to One-Row-Per-Channel*

Description

Convert a two-color targets dataframe with one row per array to one with one row per channel.

Usage

```
targetsA2C(targets, channel.codes=c(1,2), channel.columns=list(Target=c("Cy3","C
```

Arguments

`targets`	data.frame with one row per array giving information about target samples associated covariates.
`channel.codes`	numeric or character vector of length 2 giving codes for the channels
`channel.columns`	named list of character vectors of length 2. Each entry gives a pair of names of columns in `targets` which contain channel-specific information. This pair of columns should be assembled into one column in the output.
`grep`	logical, if `TRUE` then the channel column names are found by `grep`ing, i.e., the actual column names need only contain the names given by `channel.columns` as substrings

Details

The `targets` dataframe holds information about the RNA samples used as targets in the microarray experiment. It is often read from a file using `readTargets`. This function is used to convert the dataframe from an array-orientated format with one row for each array and two columns for the two channels into a channel-orientated format with one row for each individual channel observations. In statistical terms, the first format treats the arrays as cases and treats the channels as repeated measurements. The second format treats the individual channel observations as cases. The second format may be more appropriate if the data is to be analyzed in terms of individual log-intensities.

Value

data.frame with twice as many rows as `targets`. Any pair of columns named by `channel.columns` will now be one column.

Author(s)

Gordon Smyth

See Also

`targetsA2C` is used by the `coerce` method from `RGList` for `ExpressionSet` in the convert package.

An overview of methods for single channel analysis in limma is given by 07.SingleChannel.

Examples

```
targets <- data.frame(FileName=c("file1.gpr","file2.gpr"),Cy3=c("WT","KO"),Cy5=c("KO","WT
targetsA2C(targets)
```

`tmixture` *Estimate Scale Factor in Mixture of t-Distributions*

Description

This function estimates the unscaled standard deviation of the log fold change for differentially expressed genes. It is called by the function `ebayes` and is not intended to be called by users.

Usage

```
tmixture.vector(tstat,stdev.unscaled,df,proportion,v0.lim=NULL)
tmixture.matrix(tstat,stdev.unscaled,df,proportion,v0.lim=NULL)
```

Arguments

`tstat`	numeric vector or matrix of t-statistics
`stdev.unscaled`	numeric matrix conformal with `tstatf` containing the unscaled standard deviations for the coefficient estimators
`df`	numeric vector giving the degrees of freedom associated with `tstat`
`proportion`	assumed proportion of genes which are differentially expressed
`v0.lim`	numeric vector of length 2, assumed lower and upper limits for the estimated unscaled standard deviation

Details

The values in each column of `tstat` are assumed to follow a mixture of an ordinary t-distribution, with mixing proportion `1-proportion`, and `(v0+v1)/v1` times a t-distribution, with mixing proportion `proportion`. Here `v1=stdev.unscaled^2` and `v0` is the value to be estimated.

Value

Numeric vector of length equal to the number of columns of `tstat` and `stdev.unscaled`.

Author(s)

Gordon Smyth

See Also

`ebayes`

toptable *Table of Top Genes from Linear Model Fit*

Description

Extract a table of the top-ranked genes from a linear model fit.

Usage

```
topTable(fit,coef=NULL,number=10,genelist=fit$genes,adjust.method="BH",sort.by="
toptable(fit,coef=1,number=10,genelist=NULL,A=NULL,eb=NULL,adjust.method="BH",so
topTableF(fit,number=10,genelist=fit$genes,adjust.method="BH",sort.by="F",p.valu
```

Arguments

fit	list containing a linear model fit produced by lmFit, lm.series, gls.series or mrlm. For topTable, fit should be an object of class MArrayLM as produced by lmFit and eBayes.
coef	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.
number	maximum number of genes to list
genelist	data frame or character vector containing gene information. For topTable only, this defaults to fit$genes.
A	matrix of A-values or vector of average A-values. For topTable only, this defaults to fit$Amean.
eb	output list from ebayes(fit). If NULL, this will be automatically generated.
adjust.method	method used to adjust the p-values for multiple testing. Options, in increasing conservatism, include "none", "BH", "BY" and "holm". See p.adjust for the complete list of options. A NULL value will result in the default adjustment method, which is "BH".
sort.by	character string specifying statistic to rank genes by. Possibilities for topTable and toptable are "logFC", "AveExpr", "t", "P", "p", "B" or "none". "M" is allowed as a synonym for "logFC" for backward compatibility. Other permitted synonyms are "A" or "Amean" for "AveExpr", "T" for "t" and "p" for "P". Possibilities for topTableF are "F" or "none".
resort.by	character string specifying statistic to sort the selected genes by in the output data.frame. Possibilities are the same as for sort.by.
p.value	cutoff value for adjusted p-values. Only genes with lower p-values are listed.
lfc	cutoff value for log2-fold-change. Only genes with larger fold changes are listed.
...	any other arguments are passed to ebayes if eb is NULL

Details

toptable is an earlier interface and is retained only for backward compatibility.

This function summarizes a linear model fit object produced by lmFit, lm.series, gls.series or mrlm by selecting the top-ranked genes for any given contrast. topTable assumes that the linear model fit has already been processed by eBayes.

The p-values for the coefficient/contrast of interest are adjusted for multiple testing by a call to p.adjust. The "BH" method, which controls the expected false discovery rate (FDR) below the specified value, is the default adjustment method because it is the most likely to be appropriate for microarray studies. Note that the adjusted p-values from this method are bounds on the FDR rather than p-values in the usual sense. Because they relate to FDRs rather than rejection probabilities, they are sometimes called q-values. See help("p.adjust") for more information.

Note, if there is no good evidence for differential expression in the experiment, that it is quite possible for all the adjusted p-values to be large, even for all of them to be equal to one. It is quite possible for all the adjusted p-values to be equal to one if the smallest p-value is no smaller than 1/ngenes where ngenes is the number of genes with non-missing p-values.

The sort.by argument specifies the criterion used to select the top genes. The choices are: "logFC" to sort by the (absolute) coefficient representing the log-fold-change; "A" to sort by average expression level (over all arrays) in descending order; "T" or "t" for absolute t-statistic; "P" or "p" for p-values; or "B" for the lods or B-statistic.

Normally the genes appear in order of selection in the output table. If a different order is wanted, then the resort.by argument may be useful. For example, topTable(fit, sort.by="B", resort.by="logFC") selects the top genes according to log-odds of differential expression and then orders the selected genes by log-ratio in decreasing order. Or topTable(fit, sort.by="logFC", resort.by="logFC") would select the genes by absolute log-fold-change and then sort them from most positive to most negative.

topTableF ranks genes on the basis of moderated F-statistics for one or more coefficients. If topTable is called with coef has length greater than 1, then the specified columns will be extracted from fit and topTableF called on the result. topTable with coef=NULL is the same as topTableF, unless the fitted model fit has only one column.

Toptable output for all probes in original (unsorted) order can be obtained by topTable(fit,sort="none",n=Ini
However write.fit or write may be preferable if the intention is to write the results to a file. A related method is as.data.frame(fit) which coerces an MArrayLM object to a data.frame.

Value

A dataframe with a row for the number top genes and the following columns:

genelist	one or more columns of probe annotation, if genelist was included as input
logFC	estimate of the log2-fold-change corresponding to the effect or contrast
AveExpr	average log2-expression for the probe over all arrays and channels, same as Amean in the MarrayLM object
t	moderated t-statistic
P.Value	raw p-value
adj.P.Value	adjusted p-value or q-value
B	log odds that the gene is differentially expressed

Author(s)

Gordon Smyth

See Also

An overview of linear model and testing functions is given in 06.LinearModels. See also `p.adjust` in the `stats` package.

Examples

```
#  See lmFit examples
```

`trigammaInverse` *Inverse Trigamma Function*

Description

The inverse of the trigamma function.

Usage

```
trigammaInverse(x)
```

Arguments

`x`	numeric vector or array

Details

The function uses Newton's method with a clever starting value to ensure monotonic convergence.

Value

Numeric vector or array `y` satisfying `trigamma(y)==x`.

Note

This function does not accept a data.frame as argument although the internal function `trigamma` does.

Author(s)

Gordon Smyth

See Also

`trigamma`

Examples

```
y <- trigammaInverse(5)
trigamma(y)
```

trimWhiteSpace *Trim Leading and Trailing White Space*

Description

Trims leading and trailing white space from character strings.

Usage

```
trimWhiteSpace(x)
```

Arguments

x	character vector

Value

A character vector of the same length as x in which leading and trailing white space has been stripped off each value.

Author(s)

Tim Beissbarth and Gordon Smyth

See Also

An overview of LIMMA functions for reading data is given in 03.ReadingData.

Examples

```
x <- c("a "," b ")
trimWhiteSpace(x)
```

uniquegenelist *Eliminate Duplicate Names from the Gene List*

Description

Eliminate duplicate names from the gene list. The new list is shorter than the full list by a factor of ndups.

Usage

```
uniquegenelist(genelist,ndups=2,spacing=1)
```

Arguments

genelist	vector of gene names
ndups	number of duplicate spots. The number of rows of genelist must be divisible by ndups.
spacing	the spacing between duplicate names in genelist

Value

A vector of length `length(genelist)/ndups` containing each gene name once only.

Author(s)

Gordon Smyth

See Also

`unwrapdups`

Examples

```
genelist <- c("A","A","B","B","C","C","D","D")
uniquegenelist(genelist,ndups=2)
genelist <- c("A","B","A","B","C","D","C","D")
uniquegenelist(genelist,ndups=2,spacing=2)
```

---

unwrapdups *Unwrap Duplicate Spot Values from Rows into Columns*

Description

Reshape a matrix so that a set of consecutive rows becomes a single row in the output.

Usage

```
unwrapdups(M,ndups=2,spacing=1)
```

Arguments

`M`	a matrix.
`ndups`	number of duplicate spots. The number of rows of M must be divisible by `ndups`.
`spacing`	the spacing between the rows of `M` corresponding to duplicate spots, `spacing=1` for consecutive spots

Details

This function is used on matrices corresponding to a series of microarray experiments. Rows corresponding to duplicate spots are re-arranged to that all values corresponding to a single gene are on the same row. This facilitates fitting models or computing statistics for each gene.

Value

A matrix containing the same values as `M` but with fewer rows and more columns by a factor of `ndups`. Each set of `ndups` rows in `M` is strung out to a single row so that duplicate values originally in consecutive rows in the same column are in consecutive columns in the output.

Author(s)

Gordon Smyth

Examples

```
M <- matrix(1:12,6,2)
unwrapdups(M,ndups=2)
unwrapdups(M,ndups=3)
unwrapdups(M,ndups=2,spacing=3)
```

---

venn *Venn Diagrams*

Description

Compute classification counts or plot classification counts in a Venn diagram.

Usage

```
vennCounts(x, include="both")
vennDiagram(object, include="both", names, mar=rep(1,4), cex=1.5, lwd=1,
circle.col, counts.col, show.include, ...)
```

Arguments

x	numeric matrix of 0's and 1's indicating significance of a test. Usually created by decideTests.
object	either a TestResults matrix or a VennCounts object produced by vennCounts.
include	character string specifying whether to counts genes up-regulated, down-regulated or both. See details. Choices are "both", "up" or "down".
names	optional character vector giving names for the sets or contrasts
mar	numeric vector of length 4 specifying the width of the margins around the plot. This argument is passed to par.
cex	numerical value giving the amount by which the contrast names should be scaled on the plot relative to the default.plotting text. See par.
lwd	numerical value giving the amount by which the circles should be scaled on the plot. See par.
circle.col	optional vector of color specifications defining the colors by which the circles should be drawn. See par.
counts.col	optional vector of color specifications defining the colors by which the circles should be drawn. See par.
show.include	logical value whether the value of include should be printed on the plot. Defaults to FALSE if include is a single value and TRUE otherwise
...	any other arguments are passed to plot

Details

If a vennCounts object is given to vennDiagram, the include parameter is ignored. If a TestResults object is given, then it is possible to set include as a vector of 2 character strings and both will be shown.

Value

`vennCounts` produces a `VennCounts` object, which is a numeric matrix with last column `"Counts"` giving counts for each possible vector outcome. `vennDiagram` causes a plot to be produced on the current graphical device. For `venDiagram`, the number of columns of `object` should be three or fewer.

Author(s)

Gordon Smyth James Wettenhall and Francois Pepin

See Also

An overview of linear model functions in limma is given by 06.LinearModels.

Examples

```
Y <- matrix(rnorm(100*6),100,6)
Y[1:10,3:4] <- Y[1:10,3:4]+3
Y[1:20,5:6] <- Y[1:20,5:6]+3
design <- cbind(1,c(0,0,1,1,0,0),c(0,0,0,0,1,1))
fit <- eBayes(lmFit(Y,design))
results <- decideTests(fit)
a <- vennCounts(results)
print(a)
vennDiagram(a)
vennDiagram(results,include=c("up","down"),counts.col=c("red","green"))
```

---

volcanoplot *Volcano Plot*

Description

Creates a volcano plot of log-fold changes versus log-odds of differential expression.

Usage

```
volcanoplot(fit, coef=1, highlight=0, names=fit$genes$ID, xlab="Log Fold Change"
```

Arguments

`fit`	an `MArrayLM` fitted linear model object
`coef`	integer giving the coefficient
`highlight`	number of top genes to be highlighted
`names`	character vector giving text labels for the probes to be used in highlighting
`xlab`	character string giving label for x-axis
`ylab`	character string giving label for y-axis
`pch`	vector or list of plotting characters. Default is integer code 16 which gives a solid circle.
`cex`	numeric vector of plot symbol expansions. Default is 0.35.
`...`	any other arguments are passed to `plot`

Details

A volcano plot is any plot which displays fold changes versus a measure of statistical significance of the change.

Value

A plot is created on the current graphics device.

Author(s)

Gordon Smyth

See Also

An overview of presentation plots following the fitting of a linear model in LIMMA is given in 06.LinearModels.

Examples

```
#  See lmFit examples
```

---

weighted.median *Weighted Median*

Description

Compute a weighted median of a numeric vector.

Usage

```
weighted.median(x, w, na.rm = FALSE)
```

Arguments

x	a numeric vector containing the values whose mean is to be computed.
w	a vector of weights the same length as x giving the weights to use for each element of x.
na.rm	a logical value indicating whether NA values in x should be stripped before the computation proceeds.

Details

If w is missing then all elements of x are given the same weight.

Missing values in w are not handled.

The weighted median is the median of the discrete distribution with values given by x and probabilities given by w/sum(w).

Value

numeric value giving the weighted median

See Also

median, weighted.mean

Examples

```
## GPA from Siegel 1994
wt <- c(5,  5,  4,  1)/15
x <- c(3.7,3.3,3.5,2.8)
xm <- weighted.median(x,wt)
```

`write.fit` *Write MArrayLM Object to a File*

Description

Write a microarray linear model fit to a file.

Usage

```
write.fit(fit, results=NULL, file, digits=3, adjust="none", method="separate", F
```

Arguments

`fit`	object of class `MArrayLM` containing the results of a linear model fit
`results`	object of class `TestResults`
`file`	character string giving name of file
`digits`	integer indicating precision to be used
`adjust`	character string specifying multiple-testing adjustment method for the t-statistic P-values, e.g., `"BH"`. See `p.adjust` for the available options. If `NULL` or `"none"` then the P-values are not adjusted.
`method`	character string, should the P-value adjustment be `"global"` or `"separate"` for each contrast.
`F.adjust`	character string specifying adjustment method for the F-statistic P-values.
`sep`	the field separator string. Values in the output file will be separated by this string.
`...`	other arguments are passed to `write.table`

Details

This function writes a tab-delimited text file containing for each gene (1) the average log-intensity, (2) the log-ratios, (3) moderated t-statistics, (4) t-statistic P-values, (5) F-statistic if available, (6) F-statistic P-values if available, (7) classification if available and (8) gene names and annotation.

Value

No value is produced but a file is written to the current working directory.

Author(s)

Gordon Smyth

See Also

write in the base library.

An overview of linear model functions in limma is given by 06.LinearModels.

`zscore` *Z-score Equivalents*

Description

Compute z-score equivalents of for gamma or t-distribution random deviates.

Usage

```
zscoreGamma(q, shape, rate = 1, scale = 1/rate)
zscoreT(x, df)
tZscore(x, df)
```

Arguments

`q, x`	numeric matrix for vector giving deviates of a random variaable
`shape`	gamma shape parameter (>0)
`rate`	gamma rate parameter (>0)
`scale`	gamma scale parameter (>0)
`df`	degrees of freedom (>0 for `zscoreT` or >=1 for `tZscore`)

Details

These functions compute the standard normal deviates which have the same quantiles as the given values in the specified distribution. For example, if `z <- zscoreT(x,df=df)` then `pnorm(z)` equals `pt(x,df=df)`. `tZscore` is the inverse of `zscoreT`.

Care is taken to do the computations accurately in both tails of the distributions.

Value

Numeric vector giving equivalent deviates from a standard normal distribution (`zscoreGamma` and `zscoreT`) or deviates from a t-distribution (`tZscore`).

Author(s)

Gordon Smyth

See Also

qnorm, pgamma, pt

Examples

```
zscoreGamma(1, shape=1, scale=1)
zscoreT(2, df=3)
tZscore(2, df=3)
```

Index

∗Topic **IO**
controlStatus, 32
getLayout, 48
getSpacing, 47
gridr, 51
printorder, 101
readGAL, 115
readSpotTypes, 113
readTargets, 114
write.fit, 136
∗Topic **algebra**
is.fullrank, 57
∗Topic **array**
as.data.frame, 17
as.matrix, 18
avedups, 20
avereps, 21
blockDiag, 24
dim, 36
dimnames, 37
exprs.MA, 42
getEAWP, 46
normalizeMedianAbsValues, 79
uniquegenelist, 131
unwrapdups, 132
∗Topic **character**
alias2Symbol, 12
makeUnique, 69
protectMetachar, 104
removeExt, 117
strsplit2, 123
trimWhiteSpace, 131
∗Topic **classes**
as.MAList, 18
LargeDataObject-class, 9
MAList-class, 70
MArrayLM-class, 71
PrintLayout, 10
RGList-class, 118
TestResults-class, 11
∗Topic **data**
as.MAList, 18
LargeDataObject-class, 9
MAList-class, 70
PrintLayout, 10
RGList-class, 118
∗Topic **distribution**
fitFDist, 43
qqt, 104
zscore, 137
∗Topic **documentation**
01.Introduction, 1
02.Classes, 2
03.ReadingData, 2
04.Background, 4
05.Normalization, 4
06.LinearModels, 5
07.SingleChannel, 7
08.Tests, 7
09.Diagnostics, 8
10.Other, 9
changeLog, 27
limmaUsersGuide, 60
∗Topic **file**
read.columns, 107
read.maimages, 108
readHeader, 111
readImaGeneHeader, 112
removeExt, 117
∗Topic **hplot**
asMatrixWeights, 16
heatdiagram, 51
imageplot, 54
imageplot3by2, 55
mdplot, 72
modifyWeights, 77
plotDensities, 91
plotFB, 93
plotlines, 94
plotMA, 95
plotMA3by2, 97
plotPrintTipLoess, 98
printHead, 100
volcanoplot, 134
∗Topic **htest**
auROC, 19

classifyTests, 29
contrasts.fit, 30
convest, 33
decideTests, 34
ebayes, 40
geneSetTest, 44
poolVar, 99
roast, 119
squeezeVar, 122
targetsA2C, 126
TestResults-class, 11
tmixture, 127
toptable, 128
venn, 133
∗Topic **manip**
cbind, 26
merge, 73
subsetting, 124
∗Topic **math**
trigammaInverse, 130
∗Topic **methods**
helpMethods, 53
summary, 125
∗Topic **models**
anova.MAList-method, 12
arrayWeights, 13
backgroundCorrect, 22
bwss, 24
bwss.matrix, 25
fitted.MArrayLM, 44
gls.series, 49
kooperberg, 59
lm.series, 61
lmFit, 62
lmscFit, 64
loessFit, 66
mergeScans, 74
mrlm, 78
normalizeBetweenArrays, 83
normalizeForPrintorder, 85
normalizeQuantiles, 87
normalizeRobustSpline, 80
normalizeWithinArrays, 81
normexp.fit, 88
normexp.signal, 90
printtipWeights, 102
residuals.MArrayLM, 117
selectModel, 121
∗Topic **multivariate**
dupcor, 38
intraspotCorrelation, 56
normalizeBetweenArrays, 83
∗Topic **programming**
isNumeric, 58
∗Topic **regression**
arrayWeights, 13
arrayWeightsQuick, 15
designI2M, 28
fitted.MArrayLM, 44
gls.series, 49
lm.series, 61
lmFit, 62
lmscFit, 64
makeContrasts, 68
MArrayLM-class, 71
modelMatrix, 76
mrlm, 78
printtipWeights, 102
QualityWeights, 105
removeBatchEffect, 116
residuals.MArrayLM, 117
selectModel, 121
∗Topic **smooth**
ma3x3, 67
∗Topic **univar**
weighted.median, 135
[.MAList *(subsetting)*, 124
[.MArrayLM *(subsetting)*, 124
[.RGList *(subsetting)*, 124
01.Introduction, 1, *27*
02.Classes, 2, *10, 11*, *17–21*, *36, 37*, *43*, *47*, *71, 72*, *100*, *119*, *125*
03.ReadingData, 2, *16*, *26*, *33*, *48, 49*, *74*, *102*, *104*, *106*, *108*, *110–114*, *116, 117*, *124*, *131*
04.Background, 4, *23*, *60*, *68*, *90, 91*
05.Normalization, 4, *67*, *78*, *80, 81*, *83*, *85*, *87, 88*, *116*
06.LinearModels, 5, *7*, *14*, *16, 17*, *31*, *39*, *41*, *51*, *62, 63*, *69*, *77*, *79*, *94*, *103*, *121*, *130*, *134, 135*, *137*
07.SingleChannel, *5*, 7, *7*, *28*, *57*, *65*, *126*
08.Tests, *6*, 7, *11*, *19*, *30*, *34, 35*
09.Diagnostics, 8, *13*, *55, 56*, *73*, *92, 93*, *96–98*
10.Other, 9, *24*, *100*

alias2Symbol, *8*, 12
anova, *8*, *13*, *25*
anova.MAList, *26*
anova.MAList *(anova.MAList-method)*, 12
anova.MAList-method, 12
array2channel *(targetsA2C)*, 126

arrayWeights, *8*, 13, *15, 16*
arrayWeightsQuick, 15
arrayWeightsSimple *(arrayWeights)*, 13
as.data.frame, *2*, 17, *17*
as.MAList, *3*, 18
as.matrix, 18, *19*
asMatrixWeights, 16
AUC, *19*
auROC, *8*, 19
avedups, 20
avereps, 21

backgroundCorrect, *4*, 22, *82*
barcodeplot *(geneSetTest)*, 44
blockDiag, *9*, 24
bwss, *8*, 24, *26*
bwss.matrix, *8*, *13*, 25, *25*

cbind, *3*, 26, *26*, *73*
changeLog, *1*, 27
classifyTests, 29
classifyTestsF, *8*, *11*, *72*
classifyTestsF *(classifyTests)*, 29
classifyTestsP, *8*, *11*
classifyTestsP *(classifyTests)*, 29
classifyTestsT, *8*, *11*
classifyTestsT *(classifyTests)*, 29
coerce, *126*
coerce,RGList,exprSet2-method *(RGList-class)*, 118
combined, *2*, *70*, *118*
contrasts.fit, *6*, 30, *69*
controlStatus, *3*, 32, *96*
convest, *8*, 33

decideTests, *8*, *29*, 34, *52*, *133*
designI2A *(designI2M)*, 28
designI2M, 28
dim, 36, *36*, *70*, *72*, *118*
dimnames, *20, 21*, 37, *37*
dimnames<-.MAList *(dimnames)*, 37
dimnames<-.RGList *(dimnames)*, 37
dupcor, 38
duplicateCorrelation, *5*, *51*
duplicateCorrelation *(dupcor)*, 38

eBayes, *6*
eBayes *(ebayes)*, 40
ebayes, *6*, 40, *43*, *72*, *127*
exprs, *19*
exprs.MA, 42
Extract, *124*

fitFDist, *6*, *41*, 43
fitted, *44*
fitted.MArrayLM, 44
FStat, *8*
FStat *(classifyTests)*, 29

geneSetTest, *8*, 44, *119, 120*
getDupSpacing *(getLayout)*, 48
getEAWP, 46
getLayout, *3*, *10*, 48
getLayout2 *(getLayout)*, 48
getSpacing, *3*, 47
gls.series, *5*, 49
gpTools, *49*
gridc, *3*, *9*
gridc *(gridr)*, 51
gridr, *3*, *9*, 51

heatDiagram, *8*, *11*
heatDiagram *(heatdiagram)*, 51
heatdiagram, *8*, 51
helpMethods, 53

image, *53*, *55*
imageplot, *9*, 54
imageplot3by2, *9*, 55
intraspotCorrelation, *7*, 56, *65*
is.fullrank, 57
is.numeric, *58*
isNumeric, 58

kooperberg, *4*, 59

LargeDataObject, *2*, *70*, *118*
LargeDataObject-class, 9
length.MAList *(dim)*, 36
length.MArrayLM *(dim)*, 36
length.RGList *(dim)*, 36
limma *(01.Introduction)*, 1
LIMMA contents page, *1*
LIMMA User's Guide, *4*, *92*
limmaUsersGuide, *1*, 60
lm.fit, *62*, *65*
lm.series, *5*, 61
lmFit, *5*, *31*, *50*, *57*, *61*, 62, *71*, *78*
lmscFit, *7*, *63*, 64
loess, *67*
loessFit, *5*, 66, *82*
lowess, *67*

MA.RG, *5*, *70*
MA.RG *(normalizeWithinArrays)*, 81
ma3x3, 67
ma3x3.matrix, *4*

ma3x3.spottedarray, *4*
maImage, *55*
make.names, *69*
make.unique, *70*
makeContrasts, *6*, 68
makeUnique, *3*, 69
MAList, *2*, *18*, *26*, *38*, *56*, *65*, *82–84*
MAList-class, *13*, *73*
MAList-class, 70
maNorm, *83*
maNormScale, *85*
MArrayLM, *2*, *5*, *63*, *65*
MArrayLM-class, *41*
MArrayLM-class, 71
marrayNorm, *18*
marrayNorm-class, *71*
marrayRaw-class, *119*
Math, *58*
mdplot, *9*, 72
mean, *56*
median, *136*
merge, *3*, 73, *74*
merge.RGList, *70*
merged, *118*
mergeScans, 74
mergeScansRG *(mergeScans)*, 74
mixedModel2Fit, *39*
model.matrix, *6*, *28*, *77*
modelMatrix, *6*, 76
modifyWeights, *4*, *16*, 77
mrlm, *5*, 78

ncol, *70*, *72*, *118*
nlminb, *89*
nonEstimable *(is.fullrank)*, 57
normalise, *83*
normalize, *85*
normalizeBetweenArrays, *4*, *70*, 83, *83*, *87*, *118*
normalizeForPrintorder, *4*, *70*, 85, *102*, *118*
normalizeMedianAbsValues, *5*, 79
normalizeQuantiles, *5*, 87
normalizeRobustSpline, *5*, 80, *82*
normalizeWithinArrays, *4*, *23*, *66*, *70*, 81, *118*
normexp.fit, *4*, *23*, 88, *90*, *91*
normexp.signal, *4*, 90, *90*
nrow, *70*, *72*, *118*

openPDF, *61*
openVignette, *61*

p.adjust, *29*, *35*, *128–130*, *136*
par, *54*
pgamma, *137*
plotDensities, *9*, 91
plotFB, *9*, 93
plotlines, *6*, 94
plotMA, *9*, *70*, *72*, 95
plotMA3by2, *9*, 97
plotPrintorder, *9*
plotPrintorder *(normalizeForPrintorder)*, 85
plotPrintTipLoess, *9*, *70*, 98
points, *93*, *96*
poolVar, *9*, 99
printHead, *2*, 100
PrintLayout, 10, *70*, *110*, *118*
PrintLayout-class, *22*, *67*, *81*, *86*, *102*
PrintLayout-class *(PrintLayout)*, 10
printorder, *3*, *86, 87*, 101
printtipWeights, 102
printtipWeightsSimple *(printtipWeights)*, 102
protectMetachar, 104
pt, *137*

qnorm, *137*
qqnorm, *105*
qqt, 104
QualityWeights, *3*, 105, *109*

rbind, *3*
rbind.MAList *(cbind)*, 26
rbind.RGList *(cbind)*, 26
read.columns, *3*, 107
read.imagene, *3*, *112*
read.imagene *(read.maimages)*, 108
read.maimages, *3*, *107*, 108, *108*, *111*, *118*
read.marrayRaw, *110*
read.table, *108*, *110*, *114*
readGAL, *3*, *32*, 115
readGenericHeader, *3*
readGenericHeader *(readHeader)*, 111
readGPRHeader, *3*
readGPRHeader *(readHeader)*, 111
readHeader, 111
readImaGeneHeader, *3*, 112
readSMDHeader *(readHeader)*, 111
readSpotTypes, *3*, 113
readTargets, *2*, 114, *126*
remlscore, *57*

removeBatchEffect, 116
removeExt, *3*, 117
residuals, *118*
residuals.MArrayLM, 117
RG.MA, *9*
RG.MA *(normalizeWithinArrays)*, 81
RGList, *2*, *22*, *26*, *83*, *109*
RGList-class, *73*, *75*
RGList-class, 118
rlm, *79*
roast, *8*, *45*, *46*, 119

selectModel, *6*, 121
show, *11*, *70*, *118*
show,LargeDataObject-method *(LargeDataObject-class)*, 9
show,TestResults-method *(TestResults-class)*, 11
showMethods, *53*
spotc, *3*
spotc *(gridr)*, 51
spotr, *3*
spotr *(gridr)*, 51
squeezeVar, *41*, *121*, 122
statma, *83*
strsplit, *123*, *124*
strsplit2, *3*, 123
subsetted, *2*, *70*, *118*
subsetting, *57*, 124
summary, 125, *125*
summary.TestResults *(TestResults-class)*, 11
Sweave, *60*
Sys.getenv, *61*
Sys.putenv, *61*

targetsA2C, *7*, 126
TestResults, *2*, *8*, *35*
TestResults-class, 11
tmixture, 127
tmixture.matrix, *6*, *41*
tmixture.vector, *6*
topTable, *6*
topTable *(toptable)*, 128
toptable, *6*, 128
topTableF, *6*
topTableF *(toptable)*, 128
treat, *6*
treat *(ebayes)*, 40
trigamma, *130*
trigammaInverse, *43*, 130
trimWhiteSpace, 131
tZscore *(zscore)*, 137

uniquegenelist, *3*, 131
uniqueTargets *(modelMatrix)*, 76
unwrapdups, *5*, *12*, 132, *132*

venn, 133
vennCounts, *8*, *11*
vennCounts *(venn)*, 133
vennDiagram, *8*, *11*
vennDiagram *(venn)*, 133
vignette, *61*
volcanoplot, *6*, 134
vsn, *84*, *85*

weighted.mean, *136*
weighted.median, 135
wilcox.test, *46*
write, *129*, *137*
write.fit, *6*, *8*, *11*, *129*, 136
wtarea *(QualityWeights)*, 105
wtflags *(QualityWeights)*, 105
wtIgnore.Filter *(QualityWeights)*, 105

zscore, 137
zscoreGamma, *6*
zscoreGamma *(zscore)*, 137
zscoreT, *6*
zscoreT *(zscore)*, 137