

CALIB

March 24, 2012

ParameterList-class

Class "ParameterList" - List to store all the parameters

Description

A simple list-based class for storing all the model parameters of a batch of microarray.

Objects from the Class

Objects can be created by calls of the form `new("ParameterList", parameter)` where `parameter` is a list. In the CALIB package, `ParameterList` objects are normally generated by function `estimateParameter`.

List Components

Objects should contain the following list components:

MuS: Spot parameter. Mean of spot capacity.

Ka: Hybridization constant.

P1: Respective slope of the linear dye saturation function. It is different for different dyes.

P2: Respective intercept of the linear dye saturation function. It is different for different dyes

SigmaA: Standard deviation of additive error. It is different for different dyes.

SigmaM: Standard deviation of multiplicative error.

SigmaS: Spot parameter. Standard deviation of spot capacity.

SpotError: numeric matrix containing spot error of all external control spikes on arrays

Method: boolean values indicating the way to calculate the measured intensities. It contains two subfields which are both logical value: `BC` and `Area`. `BC` indicates whether background corrected measured intensities are used. `Area` indicates whether spot area is used to calculate measured intensities.

ErrorModel: a character to represent the distribution of spot capacity. "L" means spot capacity follows `log normal` distribution. "N" means spot capacity follows `normal` distribution.

genes: `data.frame` containing information on spikes spotted on the arrays. Should include a character column `Name` containing names for all the spikes.

See reference for more detailed explanation for these list components.

Extends

Class "list", from data part. Class "vector", by class "list".

Methods

This class inherits directly from class List. However since it represents the parameters of the calibration model for arrays, it makes on sense to run functions like [dim](#), [dimnames](#), or [merge](#) on this class.

Therefore, only some operations appropriate for list will work on objects of this class. ParameterList objects can be [cbind](#) and [show](#) in a compact way.

ParameterList objects are used on functions such as [normalizeData](#) or on some other data visualization functions like [plotSpikeHI](#) in the CALIB package.

Author(s)

Hui Zhao

References

Engelen, K., Naudts, B., DeMoor, B., Marchal, K. (2006) A calibration method for estimating absolute expression levels from microarray data. *Bioinformatics* 22: 1251-1258.

RG

Experiment Data: RGList_CALIB Example

Description

This [RGList_CALIB](#) object represents microarray data of two arrays.

It is generated by function [read.rg](#) from raw data files.

Usage

data (RG)

Format

RG is an [RGList_CALIB](#) object containing the following list components: `\$R`, `\$G`, `\$Rb`, `\$Gb`, `\$RArea`, `\$GArea`, `\$targets`, `\$source` and `\$genes`. It represents two microarrays and 19749 clones.

Source

This comes from a publicly available dataset, consisting 14 hybridizations. From this RG, two out of these fourteen are chosen. The experiment design of these two are color flip. Except for cDNA probes, external control spikes are also spotted on the arrays. There are 192 ratio controls, 480 calibration controls, 24 negative controls and 72 utility controls.

For more information, see the reference.

References

Engelen, K., Naudts, B., DeMoor, B., Marchal, K. (2006) A calibration method for estimating absolute expression levels from microarray data. *Bioinformatics* 22: 1251-1258.

Hilson, P., et al. (2004) Versatile gene-specific sequence tags for Arabidopsis functional genomics: transcript profiling and reverse genetics applications. *Genome Res.* 14, 2176-2189.

Examples

```
data(RG)
```

RGList_CALIB-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. It is an extension of the RGList in the LIMMA package.

Objects from the Class

Objects can be created by calls of the form `new("RGList_CALIB", RG)`, where RG is a list. In the CALIB package, RGList_CALIB objects are normally generated by `read.rg`.

List Components

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.

Rb: numeric matrix containing the red(Cy5) background intensities.

Gb: numeric matrix containing the green(Cy3) background intensities.

RArea: numeric matrix containing the red(Cy5) spot areas.

GArea: numeric matrix containing the green(Cy3) spot areas.

Optional components include:

weights: numeric matrix containing relative spot quality weights. Should be non-negative.

printer: list containing information on the process used to print the spots on the arrays. See [read.rg](#).

genes: data.frame containing information on the genes spotted on the arrays. Should include a character column Name containing names for the genes or controls.

targets: data.frame containing information on the target RNA samples. Should include factor or character columns Cy3 and Cy5 specifying which RNA was hybridized to each array.

other: list containing numeric matrices of other spot-specific information.

All of the matrices should have the same dimensions. The row dimension of targets should match the column dimension of the matrices.

Extends

Class "list", from data part. Class "LargeDataObject", directly. Class "vector", by class "list".

Methods

This class inherits directly from class List so any operation appropriate for lists will work on objects of this class. In addition, RGList\CALIB objects can be [subsetting](#), [combined](#) and [merge](#). RGList\CALIB objects will return dimensions and hence functions such as [dim](#), [dimnames](#), [nrow](#) and [ncol](#) are defined. RGList\CALIB also inherit a [show](#) method from the virtual class [LargeDataObject](#), which means that RGList\CALIB objects will print in a compact way.

In the CALIB package, RGList\CALIB objects are mainly for storing microarray data and they are used to pass microarray data into functions such as [estimateParameter](#) or [normalizeData](#).

Author(s)

Hui Zhao

References

RGList in the limma package

See Also

[RGList](#) and [LargeDataObject](#) in limma package.

SpikeList-class *Class "SpikeList" - Spike Intensity and Concentration List*

Description

A simple list-based class for storing red and green channel foreground and background intensities, spot area and concentrations for external control spike on spotted microarray.

Objects from the Class

Objects can be created by calls of the form `new("SpikeList", spike)` where `spike` is a list. In the CALIB package, SpikeList objects are normally generated by function `read.spike`.

List Components

Objects should contain the following list components:

- R: numeric matrix containing the red(Cy5) foreground intensities of all external control spikes on arrays. Rows correspond to spikes and columns to arrays.
- G: numeric matrix containing the green(Cy3) foreground intensities of all external control spikes on arrays.
- Rb: numeric matrix containing the red(Cy5) background intensities of all external control spikes on arrays.
- Gb: numeric matrix containing the green(Cy3) background intensities of all external control spikes on arrays.

RArea: numeric matrix containing the red(Cy5) spot areas of all external control spikes on arrays.

GArea: numeric matrix containing the green(Cy3) spot areas of all external control spikes on arrays.

RConc: numeric matrix containing the red(Cy5) known concentrations of all external control spikes on arrays.

GConc: numeric matrix containing the green(Cy3) know concentrations of all external control spikes on arrays.

genes: data.frame containing information on spikes spotted on the arrays. Should include a character column `Name` containing names for all the spikes.

All of the matrices should have the same dimensions.

Extends

Class "list", from data part. Class "vector", by class "list".

Methods

This class inherits directly from class `List`, so any operation appropriate for lists will work on objects of this class. In addition, `SpikeList` objects can be [subsetting](#), [combined](#) and [merged](#). `SpikeList` objects will return dimensions and hence functions such as `dim`, `dimnames`, `nrow` and `ncol` are defined. Generic method `show` is applied on `SpikeList`, so `SpikeList` will print in a compact way.

`SpikeList` objects are used on functions such as `estimateParameter` or on some other data visualization functions like `plotSpikeHI` in the `CALIB` package.

Author(s)

Hui Zhao

References

the `limma` package

See Also

[RGList_CALIB](#).

[RGList](#) in `limma` package.

adjustP2

Adjust model parameter P2

Description

Adjust the calibration model parameter P2 according to the measured intensities of all clones spotted on the array.

Usage

```
adjustP2(RG, parameter, arrayindex = arrayindex, colorindex = colorindex)
```

Arguments

RG	an RGList object.
parameter	a ParameterList object.
arrayindex	integer vector specifying the index of the arrays of whose the parameter P2 needed to be adjusted.
colorindex	integer vector specifying which color needed to be adjusted.

Details

RG is an `RGList_CALIB` object which contains all the experimental data. `parameter` is the return result of function `estimateParameter`. `arrayindex` is an integer vector. It gives the index of the arrays whose P2 is needed to be adjusted. `colorindex` is an integer vector. It gives the color needed to be adjusted. 1 means red P2 and 2 means green P2.

Value

It returns a `ParameterList` object with a adjusted P2 compared to the input argument `parameter`.

Note

The user should decide on which array and which color the adjustment is needed. Therefore it is important to specify the right array index and color index. There is no check on this in the function.

Author(s)

Hui Zhao

Examples

```
# load data: RG and parameter:
data(RG)
data(parameter)

# adjust P2
parameter_new <- adjustP2(RG,parameter,arrayindex=c(1,2),colorindex=c(2,2))
```

calibReadMe

View CALIB readme file

Description

Finds the location of the CALIB readme file and optionally opens it.

Usage

```
calibReadMe(view = TRUE)
```

Arguments

view	logical, TRUE means open the readme file and FALSE means finds out the location of the file only.
------	---------------------------------------------------------------------------------------------------

Details

The function `vignette("limma")` will find the short CALIB vignette which describes how to obtain the CALIB readme file. The readme file is not 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 the one given by `Sys.getenv("R_PDFVIEWER")`. The PDF viewer can be changed using `Sys.putenv("R_PDFVIEWER")`.

Value

Character string giving the file location.

Author(s)

Hui Zhao

References

[limmaUsersGuide](#) in the limma package

Examples

```
calibReadMe(view=FALSE)
```

cbind

Combine RGList_CALIB, SpikeList or ParameterList objects

Description

Combine a series of `RGList_CALIB` objects or a series of `SpikeList` objects or a series of `ParameterList` objects.

Usage

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

Arguments

... `RGList_CALIB` objects, `SpikeList` objects or `ParameterList` objects
.
deparse.level
see [cbind](#) in 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 `ParameterList` objects, only `cbind` is available, because it makes no sense to `rbind` parameter.

For `RGList_CALIB` objects and `SpikeList` objects, `cbind` and `rbind` are both available. 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_CALIB`, a `SpikeList` or a `ParameterList` object holding data from all the arrays and all genes from the individual objects.

Author(s)

Hui Zhao

References

`cbind` in limma package

See Also

`cbind` in the base package

`cbind` in the limma package

Examples

```
R1 <- G1 <- matrix(1:8, 4, 2)
rownames(R1) <- rownames(G1) <- c("g1", "g2", "g3", "g4")
colnames(R1) <- colnames(G1) <- c("a1", "a2")
RG1 <- new("RGList_CALIB", list(R=R1, G=G1))

R2 <- G2 <- matrix(9:16, 4, 2)
rownames(R2) <- rownames(G2) <- c("g1", "g2", "g3", "g4")
colnames(R2) <- colnames(G2) <- c("a3", "a4")
RG2 <- new("RGList_CALIB", list(R=R2, G=G2))

RG <- cbind(RG1, RG2)
```

dim *Retrieve the Dimensions of an RGList_CALIB or SpikeList object*

Description

Retrieve the number of rows (genes) and columns (arrays) for an RGList_CALIB or SpikeList

Usage

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

Arguments

x an object of class RGList_CALIB or SpikeList.

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)

Hui Zhao

References

[dim](#) in limma package

See Also

[dim](#) in the base package

[dim](#) in the limma package

Examples

```
# for RGList_CALIB  
R <- G <- matrix(1:8, 4, 2)  
rownames(R) <- rownames(G) <- c("g1", "g2", "g3", "g4")  
colnames(R) <- colnames(G) <- c("a1", "a2")  
RG <- new("RGList_CALIB", list(R=R, G=G))
```

```

dim(RG)

# for SpikeList
SR <- SG <- matrix(1:8, 4, 2)
rownames(SR) <- rownames(SG) <- c("s1", "s2", "s3", "s4")
colnames(SR) <- colnames(SG) <- c("a1", "a2")
spike <- new("SpikeList", list(R=SR, G=SG))

dim(spike)

```

dimnames	<i>Retrieve the Dimension Names of an RGList_CALIB or SpikeList object.</i>
----------	-----------------------------------------------------------------------------

Description

Retrieve the the dimension names of an RGList_CALIB object or an SpikeList object

Usage

```

## S3 method for class 'RGList_CALIB'
dimnames(x)

```

Arguments

x an object of class RGList_CALIB or SpikeList.

Details

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

A consequence is that row and column command `rownames` and `colnames` also work.

Value

Either NULL or a list of length 2. If the value is a list, its componets are either NULL or a character vector with the length of the appropriate dimension of `x`. If the list component is not NULL, the first field of the list indicates rownames and the second field indicates colnames.

Author(s)

Hui Zhao

References

[dimnames](#) in limma package

See Also

[dimnames](#) in the base package

[dimnames](#) in the limma package

Examples

```
# for RGList_CALIB
R <- G <- matrix(1:8,4,2)
rownames(R) <- rownames(G) <- c("g1", "g2", "g3", "g4")
colnames(R) <- colnames(G) <- c("a1", "a2")
RG <- new("RGList_CALIB", list(R=R,G=G))

dimnames(RG)

# for SpikeList
SR <- SG <- matrix(1:8,4,2)
rownames(SR) <- rownames(SG) <- c("s1", "s2", "s3", "s4")
colnames(SR) <- colnames(SG) <- c("a1", "a2")
spike <- new("SpikeList", list(R=SR,G=SG))

dimnames(spike)
```

```
estimateParameter Estimate model parameter from spikes
```

Description

Estimate the calibration model parameters according to the known concentration and the measured intensities of external control spikes on each array.

Usage

```
estimateParameter(spike, RG, bc = FALSE, area = TRUE, errormodel = "M")
```

Arguments

<code>spike</code>	a <code>SpikeList</code> object.
<code>RG</code>	a <code>RGList_CALIB</code> object.
<code>bc</code>	a logical value. <code>TRUE</code> means background corrected measured intensities are used. Default is <code>FALSE</code> .
<code>area</code>	a logical value. <code>TRUE</code> means spot area is used to calculate measured intensities. Namly, measured intensities are calculated by foreground intensities(or background corrected intensities, if <code>bc</code> is <code>TRUE</code>) multiply spot area. <code>FALSE</code> means spot area is not used. Default is <code>TRUE</code> .
<code>errormodel</code>	a character to indicate the distribution of spot capacity. "A" means spot capacity is additive. "M" means spot capacity is multiplicative. Default is "M".

Details

This function estimates calibration model parameters. In this function, the model parameters are estimated separately for each microarray, based on the measured intensities of the external control spikes and their known concentration in the hybridization solution. It accepts spike measured intensities and concentration from `spike` argument, which is an object of `SpikeList` class.

It supports different ways to calculate the measured intensities. Arguments `bc` and `area` are logical and their combinations is used for specifying four different ways. `bc` indicates using background

correction or not. `area` indicates multiplying spot area or not. The default value of these two arguments are `bc = FALSE` and `area = TRUE`.

The argument `errormodel` is to specify the distribution of spot capacity of each array. The spot capacity is either additive or multiplicative. Whichever distribution is more appropriate will depend largely on the type of microarray slide and spotting procedure used. The spot parameters `mus` and `sigmas` can be considered equal for all measurements of a single array.

The argument `RG` is for calculating the maximum intensity of each array. These maximum intensities are used to estimate the upper saturation level of each array.

More details please refer to the reference literature.

Value

An `ParameterList` object containing the components:

<code>MuS</code>	matrix containing <code>MuS</code> for each array.
<code>Ka</code>	matrix containing <code>Ka</code> for each array.
<code>P1</code>	matrix containing <code>P1</code> of each dye for each array.
<code>P2</code>	matrix containing <code>P2</code> of each dye for each array.
<code>SigmaA</code>	matrix containing sigma additive for each array.
<code>SigmaM</code>	matrix containing sigma multiplicative for each array.
<code>SigmaS</code>	matrix containing sigma spoterror for each array.
<code>SpotError</code>	matrix containing the spot error of each spot for each array.
<code>Method</code>	boolean values indicating the way to calculate the measured intensities.
<code>ErrorModel</code>	character "M" or "A" to indicate the type of spot capacity distribution.

Author(s)

Hui Zhao

References

Engelen, K., Naudts, B., DeMoor, B., Marchal, K. (2006) A calibration method for estimating absolute expression levels from microarray data. *Bioinformatics* 22: 1251-1258.

Examples

```
# load data: RG and spike
data(RG)
data(spike)

# for the measured intensities, take the default bc=FALSE and area=TRUE.
# use multiplicative spot error model
parameter <- estimateParameter(spike, RG)
```

getColClasses	<i>Construct colClasses vector for use within read.rg function</i>
---------------	--------------------------------------------------------------------

Description

Construct a colClasses vector, an argument in read.table used in read.rg

Usage

```
getColClasses(cols, ...)
```

Arguments

cols	a character vector of all columns to search against
...	accepts any character lists, character vectors or weight functions to match wanted columns against cols

Details

This is an internally called function by read.rg to create a colClasses vector used in read.table for fast loading of only required columns in read.rg

Value

Character. A named vector of classes to assumed for the columns. Possible values are NA (when [type.convert](#) is used), NULL (when the column is skipped).

Author(s)

Hui Zhao

References

getColClasses in limma package

Examples

```
allnames <- c("Block", "Column", "Row", "Name", "ID", "F635 Mean", "F532 Mean", "Flag", "Autofl")
Annotation <- c("Block", "Column")
Columns <- list(R="F635 Mean", G="F532 Mean")

getColClasses(allnames, Annotation, Columns)
```

`merge`*Merge RGList_CALIB or SpikeList objects*

Description

Merge two `RGList_CALIB` objects or two `SpikeList` objects in possibly irregular order.

Usage

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

Arguments

<code>x</code>	an <code>RGList_CALIB</code> object or an <code>SpikeList</code> object.
<code>y</code>	corresponding <code>RGList_CALIB</code> object or <code>SpikeList</code> object. Has the same genes or spikes as <code>x</code> , possibly in a different order, but with different arrays.
<code>...</code>	other arguments can be used in merge in the base package.

Details

[RGList_CALIB](#) and [SpikeList](#)

objects are list objects containing numeric matrices with the same dimensions. The `RGLists_CALIB` or `SpikeLists` 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 is 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`.

Value

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

Note

If the `RGList_CALIB` or `SpikeList` objects contain the same number of genes or spikes in the same order then the appropriate function to combine them is [cbind](#) rather than `merge`.

Author(s)

Hui Zhao

References

[merge](#) in limma package

See Also

[merge](#) in the base package

[merge](#) in the limma package

Examples

```

R1 <- G1 <- matrix(1:8,4,2)
rownames(R1) <- rownames(G1) <- c("g1", "g1", "g2", "g3")
colnames(R1) <- colnames(G1) <- c("a1", "a2")
RG1 <- new("RGList_CALIB", list(R=R1, G=G1))

R2 <- G2 <- matrix(9:16,4,2)
rownames(R2) <- rownames(G2) <- c("g2", "g3", "g1", "g1")
colnames(R2) <- colnames(G2) <- c("a3", "a4")
RG2 <- new("RGList_CALIB", list(R=R2, G=G2))

RG12 <- merge(RG1, RG2)
RG21 <- merge(RG2, RG1)

```

normalizeData

*Normalization: estimation of absolute expression levels***Description**

estimates absolute expression levels for each combination of a gene and a tested biological condition.

Usage

```
normalizeData(RG, parameter, array = array, condition = condition, dye = dye,
             cloneid = cloneid, idcol = idcol)
```

Arguments

RG	an <code>RGList_CALIB</code> object
parameter	a <code>ParameterList</code> object
array	integer vector specifying the index of the arrays. Has length equal to two times of the number of arrays.
condition	integer vector specifying the index of the conditions. Has length equal to two times of the number of arrays.
dye	integer vector specifying the index of the dyes. Has length equal to two times of the number of arrays.
cloneid	string vector specifying the clone ids of the clones to be normalized. If missing, normalize all the clones.
idcol	string specifying the column name of clone ids in the genes field of RG.

Details

This function estimates absolute expression levels for each combination of a gene and a tested biological condition from the measured intensity. It accepts measured intensities from RG.

The argument `parameter` is an object of `ParameterList`. The function accepts model parameters from this argument.

By using this function, for each combination of a gene and a tested biological condition, a single absolute expression level for target is estimated. Therefore, specifying the design of experiment is necessary. Namely, the design of array, condition and dye is needed. The three arguments `array`, `condition` and `dye` are three numeric vectors to indicate the design of array, condition and dye respectively. How to specify these three arguments refer to the example below.

The function is able to not only estimate all the genes on the slides but also estimate any gene on the slides separately. The argument `cloneid` accepts the clone ids of which the genes are interested by the user. If `cloneid` argument is missing, the function will estimate all the genes on the slides. In order to match clone id in the RG, column name which indicates clone ids in `RG$genes` should be specified by argument `idcol`.

Value

a numeric matrix containing the absolute expression levels. Columns indicate different conditions and rows indicate different genes.

Warning

The function doesn't allow missing clone id. So please check before run the function.

Note

The main calculation part in this function is done by c++ code.

Author(s)

Hui Zhao

References

Engelen, K., Naudts, B., DeMoor, B., Marchal, K. (2006) A calibration method for estimating absolute expression levels from microarray data. *Bioinformatics* 22: 1251-1258.

Examples

```
# load data: RG and parameter
data(RG)
data(parameter)

# define design matrix: two arrays, two condition and color-flip design
array <- c(1,1,2,2)
condition <- c(1,2,2,1)
dye <- c(1,2,1,2)

# specify clone-id column
idcol <- "CLONE_ID"

#data <- normalizeData(RG,parameter,array=array,condition=condition,dye=dye,idcol=idcol)

## only normalize a group of genes
cloneid_interested <- c("250001", "250002", "250003", "250004", "250005")
data <- normalizeData(RG,parameter,array=array,condition=condition,dye=dye,cloneid=cloneid_interested)
```

`normdata`*Example of normalized data*

Description

Normalized data of data set RG. It is generated by function `normalizeData` in the CALIB package by using calibration model parameter `parameter`.

Usage

```
data(normdata)
```

Format

This is a numeric matrix. Row corresponds to unique clone on the arrays and column corresponds to the two different conditions.

References

dataset `RG`.

dataset `parameter`.

Engelen, K., Naudts, B., DeMoor, B., Marchal, K. (2006) A calibration method for estimating absolute expression levels from microarray data. *Bioinformatics* 22: 1251-1258.

Hilson, P., et al. (2004) Versatile gene-specific sequence tags for Arabidopsis functional genomics: transcript profiling and reverse genetics applications. *Genome Res.* 14, 2176-2189.

Examples

```
data(normdata)
```

```
plotNormalizedData(normdata, condition = c(1,2))
```

`parameter`*Calibration Model parameter: ParameterList Example*

Description

This `ParameterList` object represents calibration model parameters of two arrays. It was estimated from the dataset `spike` by the function `estimateParameter` in the CALIB package.

Usage

```
data(parameter)
```

Format

parameter is a ParameterList object containing the following list components: μS , K_a , P_1 , P_2 , σ_A , σ_M , σ_S , SpotError , Method with two subfields $\text{Method}\backslash\text{BC}$ and $\text{Method}\backslash\text{Area}$, ErrorModel and genes . Among these parameters, P_1 , P_2 and σ_A are different for different dyes.

References

dataset spike.

Engelen, K., Naudts, B., DeMoor, B., Marchal, K. (2006) A calibration method for estimating absolute expression levels from microarray data. *Bioinformatics* 22: 1251-1258.

Hilson, P., et al. (2004) Versatile gene-specific sequence tags for Arabidopsis functional genomics: transcript profiling and reverse genetics applications. *Genome Res.* 14, 2176-2189.

Examples

```
data(parameter)
```

```
plotNormalizedData plot estimated absolute levels of two conditions
```

Description

plot estimated absolute levels of any two user-specified conditions. The values in the plot are in log scale.

Usage

```
plotNormalizedData(data, condition = c(1, 2), xlab = NULL, ylab = NULL,
  main = NULL, xlim = NULL, ylim = NULL, pch = 19,
  cex = 0.2, col = "black", diag=TRUE, diagcol="blue",
  diaglwd=1.5, ...)
```

Arguments

data	matrix containing estimated absolute levels. columns are conditions and rows are genes.
condition	integer vector giving the two conditions to be plotted.
xlab	a title for the x axis.
ylab	a title for the y axis.
main	an overall title for the plot.
xlim	the x limits (min,max) of the plot.
ylim	the y limits of the plot.
pch	an integer code for one of a set of plotting characters or symbols for the spike data set. Default is 19.
cex	a numerical value giving the amount by which points should be scaled relative to the default. Default is 0.2.
col	the color of the points. Default is black.

diag	a logical value. Add diagonal on the plot if it is TRUE. Default is TRUE.
diagcol	the color of the diagonal. Default is blue.
diaglwd	the width of the diagonal. Default is 1.5.
...	other graphical parameters can be used in function <code>plot</code> .

Details

The function plots estimated absolute expression levels of two conditions. It accepts expression levels from the argument 'data', which should have the same data format as the output value of the function `normalizeData`.

The two conditions to be plotted should be specified by the argument `condition`. The `condition` should be a numeric vector with length two and it should be subset of condition vector of the design matrix. see function `NormalizeData`.

see other graphic functions for the other arguments.

Value

A plot is created on the current graphics device.

Author(s)

Hui Zhao

Examples

```
# load data: normalized data
data(normdata)

# specify the two conditions to be plotted.
cond <- c(1,2)

# use the default values for other parameters.
plotNormalizedData(normdata,condition = cond)
```

plotSpikeCI *plot spike concentration vs measured intensity*

Description

plot spike known concentration and measured intensity of one array.

Usage

```
plotSpikeCI(spike, parameter,array = 1, bc = FALSE, area = TRUE,
            meanpoint = TRUE,xlab = "log(Concentration)",
            ylab = "log(Intensity)", main = colnames(spike$R)[array],
            onlycalib = TRUE, xlim = NULL, ylim = NULL, pch = 19,
            cex = 0.2, meanpch = 21, meancex = 1, lwd = 1.5,
            cy5col = "red", cy3col = "green", ...)
```

Arguments

<code>spike</code>	a <code>SpikeList</code> object.
<code>parameter</code>	a <code>ParameterList</code> object.
<code>array</code>	integer giving the array to be plotted.
<code>bc</code>	a logical value. <code>TRUE</code> means background corrected measured intensities are used. Default is <code>FALSE</code> .
<code>area</code>	a logical value. <code>TRUE</code> means spot area is used to calculate measured intensities. Namly, measured intensities are calculated by foreground intensities(or background corrected intensities, if <code>bc</code> is <code>TRUE</code>) multiply spot area. <code>FALSE</code> means spot area is not used. Default is <code>TRUE</code> .
<code>meanpoint</code>	a logical value. <code>TRUE</code> is to show meanpoint of measured intensities with the same concentration on the plot. <code>FALSE</code> means not show.
<code>xlab</code>	a title for the x axis.
<code>ylab</code>	a title for the y axis.
<code>main</code>	an overall title for the plot.
<code>onlycalib</code>	a logical value. <code>TRUE</code> means only the calibration controls are on the plot. <code>FALSE</code> means to plot all the spikes
<code>xlim</code>	the x limits (min,max) of the plot.
<code>ylim</code>	the y limits of the plot.
<code>pch</code>	a integer code for one of plotting characters or symbols for the spike data set. Default is 21.
<code>cex</code>	a numerical value giving the amount by which the points which indicate spike data set should be scaled relative to the default. Default is 0.4.
<code>meanpch</code>	a integer code for one of plotting characters or symbols for the meanpoints. Default is 21.
<code>meancex</code>	a numerical value giving the amount by which the meanpoints should be scaled relative to the default value. Default is 1.
<code>lwd</code>	width of the model curves. Default is 1.5.
<code>cy5col</code>	color of all symbols for cy5. Default is red.
<code>cy3col</code>	color of all symbols for cy3. Default is green.
<code>...</code>	other graphical parameters can be used in function <code>plot</code> .

Details

The function plots spike concentration and measured intensity of one array. array number is specified by the argument `array`. It accepts the concentration of given array from the argument `spike`, which is a `SpikeList` object. The measured intensities are calculated from `spike`. Four different ways can be used to calculate the measured intensities. Arguments `bc` and `area` are logical and their combinations are used for specifying the four different ways. `bc` indicates using background correction or not. `area` indicates multiplying spot area or not. The default value of these two arguments are `bc = FALSE` and `area = TRUE`.

In order to help data visualization, meanpoints and model curve can be added on the plot. And the arguments `meanpoint` and `parameter` are correspond to these. The meadians of every group of measured intensities which have the same concentration are shown on the polt if `meanpoint` is true. Model curves of both dye are shown if the arguement `parameter` is specified after parameter estimation.

Value

A plot is created on the current graphics device.

Author(s)

Hui Zhao

See Also

see graphic functions [plot](#), [par](#)

Examples

```
# load data: spike
data(spike)

# specify the array to be plotted.
array <- 1

# use the default values for other parameters.
plotSpikeCI(spike, array=array)

# after parameter estimation, the model curves can be shown on the plot.
data(parameter)
plotSpikeCI(spike, parameter, array=array)
```

plotSpikeHI

plot hybridized target vs intensity

Description

With final parameter setting, plot the amount of hybridized targets and intensities of calibration controls.

Usage

```
plotSpikeHI(spike, parameter, array = 1, xlab = "log(Hybridized)",
            ylab = "log(Intensity)", main = colnames(spike$R)[array],
            xlim = NULL, ylim = NULL, pch = 19, cex = 0.2,
            cy5col = "black", cy3col = "black", noerror = TRUE,
            noepch = 19, noecex = 0.1, noecy5col = "lightpink",
            noecy3col = "lightblue", curve = TRUE, lwd = 1.5,
            curvecy5col = "red", curvecy3col = "green", ...)
```

Arguments

spike	a SpikeList object.
parameter	a ParameterList object.
array	integer giving the array to be plotted.
xlab	a title for the x axis.
ylab	a title for the y axis.

<code>main</code>	an overall title for the plot.
<code>xlim</code>	the x limits (min,max) of the plot.
<code>ylim</code>	the y limits of the plot.
<code>pch</code>	an integer code for one of plotting characters or symbols for the spike data set. Default is 19.
<code>cex</code>	a numerical value giving the amount by which the points which indicate spike data set should be scaled relative to the default. Default is 0.2.
<code>cy5col</code>	color of points for cy5. Default is black.
<code>cy3col</code>	color of points for cy3. Default is black.
<code>noerror</code>	a logical value. If it is <code>TRUE</code> , plot the amount of hybridized targets assuming equal spot capacities. Default is <code>TRUE</code> .
<code>noepch</code>	<code>pch</code> for the points with equal spot capacities. Default is 19.
<code>noecex</code>	<code>cex</code> for the points with equal spot capacities. Default is 0.1.
<code>noecy5col</code>	color for the points with equal spot capacities of cy5. Default is lightpink.
<code>noecy3col</code>	color for the points with equal spot capacities of cy3. Default is lightblue.
<code>curve</code>	a logical value. If it is <code>TRUE</code> , plot final parameter setting. Default is <code>TRUE</code> .
<code>lwd</code>	width of the parameter curves. Default is 1.5.
<code>curvecy5col</code>	color of the parameter curves for cy5. Default is red.
<code>curvecy3col</code>	color of the parameter curves for cy3. Default is green.
<code>...</code>	other graphical parameters can be used in function <code>'plot'</code> .

Details

The function plots hybridized targets vs measured intensities of one array. The argument `array` gives the array index to be plotted. The function accepts the spike concentrations from the argument `spike` and the estimated spot error for each spot from the argument `parameter`. The hybridized targets for each spot can be calculated by the following formula: formula.

The argument `noerror` says whether or not the hybridized targets, which are calculated by the above mentioned formula assuming equal spot capacities, are plotted. If they are plotted, other arguments like `noepch`, `noecex`, `noecy5col` and `noecy3col` are used to specify the type, the size and the color of the points.

Estimated parameter curves can be shown on the plot. Since model parameters are different for two colors, two parameter curves are expected for one array. The function accepts parameters of both colors from the argument `parameter`. If the curves are plotted, the arguments `lwd`, `curvecy3col` and `curvecy5col` are used to specify the width and color of the curves.

Details for the graphical parameters can be seen in function `plot`, `points` and `curve`.

Value

A plot is created on the current graphics device.

Author(s)

Hui Zhao

References

Engelen, K., Naudts, B., DeMoor, B., Marchal, K. (2006) A calibration method for estimating absolute expression levels from microarray data. *Bioinformatics* 22: 1251-1258.

Examples

```
# load data: spike and parameter
data(spike)
data(parameter)

# specify the array to be plotted.
array <- 1

# use the default values for other parameters
plotSpikeHI(spike,parameter,array=array)
```

plotSpikeRG	<i>plot spike intensity R vs G</i>
-------------	------------------------------------

Description

plot red intensity vs green intensity of spikes.

Usage

```
plotSpikeRG(spike,parameter,RG,array = 1, bc = FALSE, area = TRUE,
            xlab = "log(Rintensity)", ylab = "log(Gintensity)",
            main = colnames(spike$R)[array], onlycalib = FALSE,
            xlim = NULL, ylim = NULL, pch = 19, cex = 0.3, col = "black",
            allpch = 19, allcex = 0.05, allcol = "lightgrey", diag = TRUE,
            diagcol = "grey", diaglwd = 1, curvecol = "blue",
            curvelwd = 1.5, calibtype = 1, adjusttype = 4, ...)
```

Arguments

spike	a SpikeList object.
parameter	a ParameterList object.. If parameter argument is sepcified, model curves are shown on the plot.
RG	a RGList_CALIB object. If parameter argument is specified, this argument is obligated. More description in Detail section.
array	integer giving the array to be plotted.
bc	a logical value. TRUE means background corrected measured intensities are used. Default is FALSE.
area	a logical value. TRUE means spot area is used to calculate measured intensities. Namly, measured intensities are calculated by foreground intensities(or background corrected intensities, if bc is TRUE) multiply spot area. FALSE means spot area is not used. Default is TRUE.
xlab	a title for the x axis.
ylab	a title for the y axis.
main	an overall title for the plot.
onlycalib	a logical value. TRUE means only the calibration controls are on the plot. FALSE means to plot all the spikes

xlim	the x limits (min,max) of the plot.
ylim	the y limits (min,max) of the plot.
pch	an integer code for one of a set of plotting characters or symbols for the spike data set. Default is 21.
cex	a numerical value giving the amount by which the points which indicate spike data set should be scaled relative to the default. Default is 0.3.
col	the color of the points indicating spike data set. Default is black.
allpch	an integer code for one of a set of plotting characters or symbols for the spike data set. Default is 19.
allcex	a numerical value giving the amount by which the points which indicate all data set should be scaled relative to the default. Default is 0.05.
allcol	the color of the points indicating all data set. Default is lightgrey.
diag	a logical value. Add diagonal on the plot if it is TRUE. Default is TRUE.
diagcol	the color of the diagonal. Default is grey.
diaglwd	the width of the diagonal. Default is 1.
curvecol	the color of the model curves. Default is blue.
curvelwd	the width of the model curves for calibration control spikes. Default is 1.5.
calibtype	the line type of the model curves for calibration control spikes. Default is 1.
adjusttype	the line type of the model curves (using parameter after adjustment) for calibration control spikes. Default is 4.
...	other graphical parameters can be used in function <code>plot</code> .

Details

The function plots red vs green measured intensities of spikes of one array. The argument `array` gives the array index to be plotted.

If parameter estimation is done, the model curves can be plotted by giving the argument `parameter`. And if the argument `parameter` is specified, the argument `RG` is obligated. The combination of these two arguments is used to compare how the model fits to the spike and to the whole data set. More details about the usage of this function refers to the readme file of this package.

It supports different ways to calculate the measured intensities. Arguments `bc` and `area` are logical and their combinations are used for specifying four different ways. `bc` indicates using background correction or not. `area` indicates multiplying spot area or not. The default value of these two arguments are `bc = FALSE` and `area = TRUE`.

see other graphic functions for the other arguments.

Value

A plot is created on the current graphics device.

Author(s)

Hui Zhao

Examples

```
# load data: spike
data(spike)

# specify the array to be plotted.
array <- 1

# use the default values for other arguments
plotSpikeRG(spike,array=array)

# after parameter estimation, the model curves can be shown on the plot.
data(parameter)
data(RG)
plotSpikeRG(spike,parameter, RG)
```

plotSpikeSpotError *plot spot error of spikes.*

Description

Plot spot error of spikes. Three types of plots are available: histogram, boxplot and density.

Usage

```
plotSpikeSpotError(parameter, array = 1, plottype = "hist", width = 1,
                   plotnames = NULL, main = NULL, ...)
```

Arguments

parameter	a ParameterList object.
array	integer giving the array to be plotted.
plottype	string giving the type of plot.
width	needed for density plot. This exists for compatibility with S; if given, and <code>bw</code> is not, will set <code>bw</code> to <code>width</code> if this is a character string, or to a kernel-dependent multiple of <code>width</code> if this is numeric. Default is 1.
plotnames	needed for boxplot. group labels which will be printed under each boxplot.
main	an overall title for the plot.
...	other parameters can be used according to the plottype user specified.

Details

The function plots spot error of one array on different types of plots. Three types, which are histogram, boxplot and density function, are available now. The argument `plottype` is used for giving the plot type. It should be one of the following three types: "hist", "boxplot" and "dens". The argument `array` gives the array index to be plotted.

The function accepts estimated spot error from the argument `parameter`.

Value

A plot is created on the current graphics device.

Author(s)

Hui Zhao

See Also

[hist](#), [boxplot](#) and [plot](#) in the graphics package.
[density](#) in the stats package.

Examples

```
# load data: parameter
data(parameter)

# specify the array to be plotted.
array <- 1

# plot histogram
plotSpikeSpotError(parameter, array=array, plottype="hist")
# plot boxplot
plotSpikeSpotError(parameter, array=array, plottype="boxplot", plotnames=NULL)
# plot density function
plotSpikeSpotError(parameter, array=array, plottype="dens", width=1)
```

read.rg

Read RGList_CALIB from Image Analysis Output Files

Description

Reads an RGList\CALIB from a series of microarray image analysis output files

Usage

```
read.rg(files = NULL, source = "generic", path = NULL, ext = NULL,
        names = NULL, columns = NULL, other.columns = NULL,
        annotation = NULL, wt.fun = NULL, verbose = TRUE, sep = "\t",
        quote = NULL, DEBUG = FALSE, ...)
```

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 <code>ext</code> 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", "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.

<code>ext</code>	character string giving optional extension to be added to each file name
<code>names</code>	character vector of names to be associated with each array as column name. Defaults to <code>removeExt(files)</code> .
<code>columns</code>	list with fields <code>R</code> , <code>G</code> , <code>Rb</code> , <code>Gb</code> , <code>RArea</code> and <code>GArea</code> giving the column names to be used for red foreground, green foreground, red background, green background, red area and green area respectively. Or, in the case of Imagen data, a list with fields <code>f</code> and <code>b</code> . This argument is optional if <code>source</code> is specified, otherwise it is required.
<code>other.columns</code>	character vector of names of other columns to be read containing spot-specific information
<code>annotation</code>	character vector of names of columns containing annotation information about the probes
<code>wt.fun</code>	function to calculate spot quality weights
<code>verbose</code>	logical, TRUE to report each time a file is read
<code>sep</code>	the field separator character
<code>quote</code>	character string of characters to be treated as quote marks
<code>DEBUG</code>	a logical value, if TRUE, a series of echo statements will be printed for each file. Details on the file, skip, and selected columns in a <code>colClasses</code> format for <code>read.table</code> will be displayed.
<code>...</code>	any other arguments are passed to <code>read.table</code> .

Details

This is the main data input function for CALIB package. It has the similar usage as the `read.maimages` function in `limma` package. The output of the function is an `RGList_CALIB` object. However, there are two more fields - `$RArea` and `$GArea` than `RGList` object in `limma` package. These two fields contain spot area of each color. More details see `read.maimages` in `limma` package.

Value

An `RGList_CALIB` object containing the components

<code>R</code>	matrix containing the red channel foreground intensities for each spot for each array.
<code>G</code>	matrix containing the green channel foreground intensities for each spot for each array.
<code>Rb</code>	matrix containing the red channel background intensities for each spot for each array.
<code>Gb</code>	matrix containing the green channel background intensities for each spot for each array.
<code>RArea</code>	matrix containing the red spot area for each spot for each array.
<code>GArea</code>	matrix containing the green spot area for each spot for each array.
<code>weights</code>	spot quality weights, if <code>wt.fun</code> is given
<code>other</code>	list containing matrices corresponding to <code>other.columns</code> 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"

Author(s)

Hui Zhao

References[read.maimages](#) in limma package**See Also**'read.rg' is based on [read.table](#) in the base package**Examples**

```
# Read all .gpr files from current working directory.
# files <- dir(pattern="*\.gpr$")
# RG <- read.rg(files, "genepix")
```

read.spike*Read SpikeList from a RGList_CALIB and Concentration files*

Description

Reads a SpikeList from a given RGList\CALIB object and user specified concentration files

Usage

```
read.spike(RG, file = NULL, path = NULL, ext = NULL, sep = "\t", conccol,
           regexpcol, different = FALSE, ...)
```

Arguments

RG	a RGList_CALIB object.
file	character string giving the name of the concentration file.
path	character string giving the directory containing the file. Can be omitted if the file is in the current working directory.
ext	character string giving optional extension to be added to each file name.
sep	field separator character.
conccol	list with fields RConc, GConc giving the column names to be used for red and green concentration in the concentration file.

regexprcol	character vector giving regular expressions in the concentration file.
different	a logical value. TRUE means that different arrays use different spikes. So every array should have one concentration file. FALSE means that different arrays use the same spike and only one concentration file is needed. Default is FALSE.
...	any other arguments are passed to <code>read.table</code> .

Details

This is the function to generate `SpikeList` in the CALIB package. `SpikeList` contains all the information for the spikes. This function extracts foreground and background intensities and spot area from `RGList_CALIB`, which is generated from function `read.rg`. Also this function reads concentrations for each spike from a user-specified concentration file (or more than one concentration files if different arrays use different spikes).

For the concentration file, it should contain the following columns: regular expression, red and green concentrations and spike type. Spike type should be in the set of `Calibration`, `Ratio` and `Negative`.

See the CALIB User's Guide for the example of this function.

Value

An `SpikeList` object containing the components

R	matrix containing the red channel foreground intensities for each spot for each array.
G	matrix containing the green channel foreground intensities for each spot for each array.
Rb	matrix containing the red channel background intensities for each spike for each array.
Gb	matrix containing the green channel background intensities for each spike for each array.
RArea	matrix containing the red spot area for each spike for each array.
GArea	matrix containing the green spot area for each spike for each array.
RConc	matrix containing the red concentration for each spike for each array.
GConc	matrix containing the green concentration for each spike for each array.
genes	data frame containing annotation information about the probes, for example, spike types and IDs and spatial positions on the array

Author(s)

Hui Zhao

References

~put references to the literature/web site here ~

 spike

Experiment Data: SpikeList Example

Description

This `SpikeList` object contains all the spikes in dataset `RG`. It is generated by the function `read.spike` from `RG` and user-specified concentration file in the `CALIB` package.

Usage

```
data(spike)
```

Format

`spike` is an `SpikeList` object containing the following list components: `\$R`, `\$G`, `\$Rb`, `\$Gb`, `\$RArea`, `\$GArea`, `\$RConc`, `\$GConc` and `\$genes`. It represents two microarrays and 600 spikes, including 480 calibration controls, 96 ratio controls and 24 negative controls.

Source

For the source information, see the introduction of dataset `RG`.

References

dataset `RG`.

Engelen, K., Naudts, B., DeMoor, B., Marchal, K. (2006) A calibration method for estimating absolute expression levels from microarray data. *Bioinformatics* 22: 1251-1258.

Hilson, P., et al. (2004) Versatile gene-specific sequence tags for Arabidopsis functional genomics: transcript profiling and reverse genetics applications. *Genome Res.* 14, 2176-2189.

Examples

```
data(spike)
```

```
plotSpikeRG(spike, array=1)
```

 subsetting

Subset of RGList_CALIB, SpikeList or ParameterList object

Description

Exact a subset of an `RGList_CALIB`, `SpikeList` or `ParameterList` object.

Usage

```
## S3 method for class 'RGList_CALIB'
object[i, j, ...]
```

Arguments

`object` an object of class `RGList_CALIB` or `SpikeList`.
`i, j` subscripts. `i` is the subscripts of the spots and `j` is the subscripts of the arrays.
`...` not used

Details

`i, j` may take any values acceptable for the matrix components of `object`.

Value

An object of the same class as `object` containing the data with specified subset of spots and arrays.

Author(s)

Hui Zhao

References

[subsetting](#) in limma package

See Also

[subsetting](#) in the limma package

Examples

```
# for RGList_CALIB
R <- G <- matrix(1:8, 4, 2)
rownames(R) <- rownames(G) <- c("g1", "g2", "g3", "g4")
colnames(R) <- colnames(G) <- c("a1", "a2")
RG <- new("RGList_CALIB", list(R=R, G=G))

RG[1:2, ]
RG[, 1:2]
RG[1:2, 1:2]
```

Index

- *Topic **array**
 - dim, 9
 - dimnames, 10
 - subsetting, 30
- *Topic **classes**
 - ParameterList-class, 1
 - RGList_CALIB-class, 3
 - SpikeList-class, 4
- *Topic **datasets**
 - normdata, 17
 - parameter, 17
 - RG, 2
 - spike, 30
- *Topic **documentation**
 - calibReadMe, 6
- *Topic **file**
 - getColClasses, 13
 - read.rg, 26
 - read.spike, 28
- *Topic **hplot**
 - plotNormalizedData, 18
 - plotSpikeCI, 19
 - plotSpikeHI, 21
 - plotSpikeRG, 23
 - plotSpikeSpotError, 25
- *Topic **manip**
 - cbind, 7
 - merge, 14
- *Topic **optimize**
 - adjustP2, 5
 - estimateParameter, 11
 - normalizeData, 15
- [.ParameterList (subsetting), 30
- [.RGList_CALIB (subsetting), 30
- [.SpikeList (subsetting), 30

- adjustP2, 5

- boxplot, 26

- calibReadMe, 6
- cbind, 2, 7, 7, 8, 14
- coerce, RGList_CALIB, exprSet2-method (RGList_CALIB-class), 3

- combined, 4, 5
- curve, 22

- density, 26
- dim, 2, 4, 5, 9, 9
- dimnames, 2, 4, 5, 10, 10

- estimateParameter, 1, 4–6, 11, 17

- getColClasses, 13

- hist, 26

- LagerDataObject, 4
- LargeDataObject, 4
- length.RGList_CALIB (dim), 9
- length.SpikeList (dim), 9
- limmaUsersGuide, 7

- merge, 2, 4, 14, 14
- merged, 5

- ncol, 4, 5
- ncol(x), 9
- normalizeData, 2, 4, 15, 17, 19
- normdata, 17
- nrow, 4, 5
- nrow(x), 9

- par, 21
- parameter, 17, 17
- ParameterList, 6, 8, 15, 17, 21, 23, 25
- ParameterList-class, 1
- plot, 19–22, 24, 26
- plotNormalizedData, 18
- plotSpikeCI, 19
- plotSpikeHI, 2, 5, 21
- plotSpikeRG, 23
- plotSpikeSpotError, 25
- points, 22

- rbind.RGList_CALIB (cbind), 7
- rbind.SpikeList (cbind), 7
- read.maimages, 28
- read.rg, 2, 3, 26, 29

`read.spike`, 4, 28, 30
`read.table`, 27–29
`RG`, 2, 17, 30
`RGList`, 4, 5
`RGList_CAILB`, 8
`RGList_CALIB`, 2, 5, 6, 14, 15, 23, 27, 28
`RGList_CALIB-class`, 3

`show`, 2, 4, 5
`show`, `ParameterList`-method
 (`ParameterList-class`), 1
`show`, `RGList_CALIB`-method
 (`RGList_CALIB-class`), 3
`show`, `SpikeList`-method
 (`SpikeList-class`), 4
`spike`, 17, 30
`SpikeList`, 8, 11, 14, 20, 21, 23, 29, 30
`SpikeList-class`, 4
`subsetting`, 4, 5
`subsetting`, 30, 31

`type.convert`, 13