

oligo

February 9, 2012

mmindex *Accessors for PM, MM or background probes indices.*

Description

Extracts the indexes for PM, MM or background probes.

Usage

```
mmindex(object, ...)  
pmindex(object, ...)  
bgindex(object, ...)
```

Arguments

object	FeatureSet or DBPDInfo object
...	Extra arguments, not yet implemented

Details

The indices are ordered by 'fid', i.e. they follow the order that the probes appear in the CEL/XYS files.

Value

A vector of integers representing the rows of the intensity matrix that correspond to PM, MM or background probes.

Examples

```
## How pm() works  
## Not run:  
x <- read.celfiles(list.celfiles())  
pms0 <- pm(x)  
pmi <- pmindex(x)  
pms1 <- exprs(x)[pmi,]  
identical(pms0, pms1)  
  
## End(Not run)
```

mm	<i>Accessors and replacement methods for the intensity/PM/MM/BG matrices.</i>
----	-------------------------------------------------------------------------------

Description

Accessors and replacement methods for the PM/MM/BG matrices.

Usage

```
intensity(object)
mm(object, subset = NULL)
pm(object, subset = NULL, ...)
bg(object, subset = NULL)
mm(object) <-value
pm(object) <-value
bg(object) <-value
```

Arguments

object	FeatureSet object.
subset	Not implemented yet.
value	matrix object.
...	Extra arguments for future implementation.

Details

For all objects but `TilingFeatureSet`, these methods will return matrices. In case of `TilingFeatureSet` objects, the value is a 3-dimensional array (probes x samples x channels).

`intensity` will return the whole intensity matrix associated to the object. `pm`, `mm`, `bg` will return the respective PM/MM/BG matrix.

Examples

```
if (require(maqcExpression4plex) & require(pd.hg18.60mer.expr)) {
  xysPath <- system.file("extdata", package="maqcExpression4plex")
  xysFiles <- list.xysfiles(xysPath, full.name=TRUE)
  ngsExpressionFeatureSet <- read.xysfiles(xysFiles)
  pm(ngsExpressionFeatureSet) [1:10, ]
}
```

MAplot

*MA plots***Description**

Create MA plots using a reference array (if one channel) or using channel2 as reference (if two channel).

Usage

```
MAplot(object, ...)

## S4 method for signature 'FeatureSet'
MAplot(object, what=pm, transfo=log2, groups,
        refSamples, which, pch=".", summaryFun=rowMedians,
        plotFun=smoothScatter, main="vs pseudo-median reference chip",
        pairs=FALSE, ...)

## S4 method for signature 'TilingFeatureSet'
MAplot(object, what=pm, transfo=log2, groups,
        refSamples, which, pch=".", summaryFun=rowMedians,
        plotFun=smoothScatter, main="vs pseudo-median reference chip",
        pairs=FALSE, ...)

## S4 method for signature 'PLMset'
MAplot(object, what=coefs, transfo=identity, groups,
        refSamples, which, pch=".", summaryFun=rowMedians,
        plotFun=smoothScatter, main="vs pseudo-median reference chip",
        pairs=FALSE, ...)

## S4 method for signature 'matrix'
MAplot(object, what=identity, transfo=identity,
        groups, refSamples, which, pch=".", summaryFun=rowMedians,
        plotFun=smoothScatter, main="vs pseudo-median reference chip",
        pairs=FALSE, ...)

## S4 method for signature 'ExpressionSet'
MAplot(object, what=exprs, transfo=identity,
        groups, refSamples, which, pch=".", summaryFun=rowMedians,
        plotFun=smoothScatter, main="vs pseudo-median reference chip",
        pairs=FALSE, ...)
```

Arguments

object	FeatureSet, PLMset or ExpressionSet object.
what	function to be applied on object that will extract the statistics of interest, from which log-ratios and average log-intensities will be computed.
transfo	function to transform the data prior to plotting.

<code>groups</code>	factor describing groups of samples that will be combined prior to plotting. If missing, MvA plots are done per sample.
<code>refSamples</code>	integers (indexing samples) to define which subjects will be used to compute the reference set. If missing, a pseudo-reference chip is estimated using <code>summaryFun</code> .
<code>which</code>	integer (indexing samples) describing which samples are to be plotted.
<code>pch</code>	same as <code>pch</code> in <code>plot</code>
<code>summaryFun</code>	function that operates on a matrix and returns a vector that will be used to summarize data belonging to the same group (or reference) on the computation of grouped-stats.
<code>plotFun</code>	function to be used for plotting. Usually <code>smoothScatter</code> , <code>plot</code> or <code>points</code> .
<code>main</code>	string to be used in title.
<code>pairs</code>	logical flag to determine if a matrix of MvA plots is to be generated
<code>...</code>	Other arguments to be passed downstream, like <code>plot</code> arguments.

Details

MAplot will take the following extra arguments:

1. `subset`: indices of elements to be plotted to reduce impact of plotting 100's thousands points (if `pairs=FALSE` only);
2. `span`: see `loess`;
3. `family.loess`: see `loess`;
4. `addLoess`: logical flag (default TRUE) to add a loess estimate;
5. `parParams`: list of params to be passed to `par()` (if `pairs=TRUE` only);

Value

Plot

Author(s)

Benilton Carvalho - based on Ben Bolstad's original MAplot function.

See Also

`plot`, `smoothScatter`

Examples

```
if(require(oligoData) & require(pd.hg18.60mer.expr)){
  data(nimbleExpressionFS)
  nimbleExpressionFS
  groups <- factor(rep(c('brain', 'UnivRef'), each=3))
  data.frame(sampleNames(nimbleExpressionFS), groups)
  MAplot(nimbleExpressionFS, pairs=TRUE, ylim=c(-.5, .5), groups=groups)
}
```

PLMset-class	<i>Class PLMset</i>
--------------	---------------------

Description

This is a class representation for Probe level Linear Models fitted to Affymetrix GeneChip probe level data.

Objects from the Class

Objects can be created using the function `fitPLM`

Slots

`probe.coefs`: Object of class "matrix". Contains model coefficients related to probe effects.

`se.probe.coefs`: Object of class "matrix". Contains standard error estimates for the probe coefficients.

`chip.coefs`: Object of class "matrix". Contains model coefficients related to chip (or chip level) effects for each fit.

`se.chip.coefs`: Object of class "matrix". Contains standard error estimates for the chip coefficients.

`const.coefs`: Object of class "matrix". Contains model coefficients related to intercept effects for each fit.

`se.const.coefs`: Object of class "matrix". Contains standard error estimates for the intercept estimates

`model.description`: Object of class "character". This string describes the probe level model fitted.

`weights`: List of objects of class "matrix". Contains probe weights for each fit. The matrix has columns for chips and rows are probes.

`phenoData`: Object of class "phenoData" This is an instance of class `phenoData` containing the patient (or case) level data. The columns of the `pData` slot of this entity represent variables and the rows represent patients or cases.

`annotation` A character string identifying the annotation that may be used for the `ExpressionSet` instance.

`experimentData`: Object of class "MIAME". For compatibility with previous version of this class description can also be a "character". The class `characterOrMIAME` has been defined just for this.

`cdfName`: A character string giving the name of the `cdfFile`.

`nrow`: Object of class "numeric". Number of rows in chip.

`ncol`: Object of class "numeric". Number of cols in chip.

`narrays`: Object of class "numeric". Number of arrays used in model fit.

`normVec`: Object of class "matrix". For storing normalization vector(s). Not currently used

`varcov`: Object of class "list". A list of variance/covariance matrices.

`residualSE`: Object of class "matrix". Contains residual standard error and df.

`residuals`: List of objects of class "matrix". Contains residuals from model fit (if stored).

`model.call`: Object of class "call"

`manufacturer`: Manufacturer string (affymetrix/nimblegen)

Methods

boxplot signature(x = "PLMset"): Boxplot of Normalized Unscaled Standard Errors (NUSE).

NUSE signature(x = "PLMset"): Boxplot of Normalized Unscaled Standard Errors (NUSE) or NUSE values.

RLE signature(x = "PLMset"): Relative Log Expression boxplot or values.

coefs signature(object = "PLMset"): Extractor for chip coefficients

se signature(object = "PLMset"): Extractor for the standard errors for chip coefficients

Note

This class is better described in the vignette.

Author(s)

B. M. Bolstad <bmb@bmbolstad.com>

References

Bolstad, BM (2004) *Low Level Analysis of High-density Oligonucleotide Array Data: Background, Normalization and Summarization*. PhD Dissertation. University of California, Berkeley.

 mmSequence

Probe Sequences

Description

Accessor to the (PM/MM/background) probe sequences.

Usage

```
mmSequence(object)
pmSequence(object, ...)
bgSequence(object, ...)
```

Arguments

```
object      FeatureSet, AffySNPPDInfo or DBPDInfo object
...         additional arguments
```

Value

A DNASringSet containing the PM/MM/background probe sequence associated to the array.

basecontent	<i>Sequence Base Contents</i>
-------------	-------------------------------

Description

Function to compute the amounts of each nucleotide in a sequence.

Usage

```
basecontent(seq)
```

Arguments

`seq` character vector of length `n` containing a valid sequence (A/T/C/G)

Value

matrix with `n` rows and 4 columns with the counts for each base.

Examples

```
sequences <- c("ATATATCCCCG", "TTTCCGAGC")
basecontent(sequences)
```

basicPLM	<i>Simplified interface to PLM.</i>
----------	-------------------------------------

Description

Simplified interface to PLM.

Usage

```
basicPLM(pmMat, pnVec, normalize = TRUE, background = TRUE, transfo =
  log2, method = c('plm', 'plmr', 'plmrr', 'plmrc'), verbose = TRUE)
```

Arguments

<code>pmMat</code>	Matrix of intensities to be processed.
<code>pnVec</code>	Probeset names
<code>normalize</code>	Logical flag: normalize?
<code>background</code>	Logical flag: background adjustment?
<code>transfo</code>	function: function to be used for data transformation prior to summarization.
<code>method</code>	Name of the method to be used for normalization. 'plm' is the usual PLM model; 'plmr' is the (row and column) robust version of PLM; 'plmrr' is the row-robust version of PLM; 'plmrc' is the column-robust version of PLM.
<code>verbose</code>	Logical flag: verbose.

Value

A list with the following components:

Estimates	A (length(pnVec) x ncol(pmMat)) matrix with probeset summaries.
StdErrors	A (length(pnVec) x ncol(pmMat)) matrix with standard errors of 'Estimates'.
Residuals	A (nrow(pmMat) x ncol(pmMat)) matrix of residuals.

Note

Currently, only RMA-bg-correction and quantile normalization are allowed.

Author(s)

Benilton Carvalho

See Also

[rcModelPLM](#), [rcModelPLMr](#), [rcModelPLMrr](#), [rcModelPLMrc](#), [basicRMA](#)

Examples

```
set.seed(1)
pms <- 2^matrix(rnorm(1000), nc=20)
colnames(pms) <- paste("sample", 1:20, sep="")
pns <- rep(letters[1:10], each=5)
res <- basicPLM(pms, pns, TRUE, TRUE)
res[['Estimates']][1:4, 1:3]
res[['StdErrors']][1:4, 1:3]
res[['Residuals']][1:20, 1:3]
```

basicRMA

Simplified interface to RMA.

Description

Simple interface to RMA.

Usage

```
basicRMA(pmMat, pnVec, normalize = TRUE, background = TRUE, bgversion = 2, destr
```

Arguments

pmMat	Matrix of intensities to be processed.
pnVec	Probeset names.
normalize	Logical flag: normalize?
background	Logical flag: background adjustment?
bgversion	Version of background correction.
destructive	Logical flag: use destructive methods?
verbose	Logical flag: verbose.
...	Not currently used.

Value

Matrix.

Examples

```
set.seed(1)
pms <- 2^matrix(rnorm(1000), nc=20)
colnames(pms) <- paste("sample", 1:20, sep="")
pns <- rep(letters[1:10], each=5)
res <- basicRMA(pms, pns, TRUE, TRUE)
res[, 1:3]
```

boxplot

Boxplot

Description

Boxplot for observed (log-)intensities in a FeatureSet-like object (ExpressionFeatureSet, ExonFeatureSet, SnpFeatureSet, TilingFeatureSet) and ExpressionSet.

Usage

```
## S4 method for signature 'FeatureSet'
boxplot(x, which=c("pm", "mm", "bg", "both", "all"), transfo=log2, nsample=10000)

## S4 method for signature 'ExpressionSet'
boxplot(x, which, transfo=identity, nsample=10000, ...)
```

Arguments

<code>x</code>	a FeatureSet-like object or ExpressionSet object.
<code>which</code>	character defining what probe types are to be used in the plot.
<code>transfo</code>	a function to transform the data before plotting. See 'Details'.
<code>nsample</code>	number of units to sample and build the plot.
<code>...</code>	arguments to be passed to the default boxplot method.

Details

The 'transfo' argument will set the transformation to be used. For raw data, 'transfo=log2' is a common practice. For summarized data (which are often in log2-scale), no transformation is needed (therefore 'transfo=identity').

Note

The boxplot methods for FeatureSet and Expression use a sample (via sample) of the probes/probesets to produce the plot. Therefore, the user interested in reproducibility is advised to use set.seed.

See Also

[hist](#), [image](#), [sample](#), [set.seed](#)

chromosome *Accessor for chromosome information*

Description

Returns chromosome information.

Usage

```
pmChr(object)
```

Arguments

object TilingFeatureSet or SnpCallSet object

Details

chromosome() returns the chromosomal information for all probes and pmChr() subsets the output to the PM probes only (if a TilingFeatureSet object).

Value

Vector with chromosome information.

darkColors *Create set of colors, interpolating through a set of preferred colors.*

Description

Create set of colors, interpolating through a set of preferred colors.

Usage

```
darkColors(n)
seqColors(n)
seqColors2(n)
divColors(n)
```

Arguments

n integer determining number of colors to be generated

Details

darkColors is based on the Dark2 palette in RColorBrewer, therefore useful to describe qualitative features of the data.

seqColors is based on Blues and generates a gradient of blues, therefore useful to describe quantitative features of the data. seqColors2 behaves similarly, but it is based on OrRd (white-orange-red).

divColors is based on the RdBu palette in RColorBrewer, therefore useful to describe quantitative features ranging on two extremes.

Examples

```
x <- 1:10
y <- 1:10
cols1 <- darkColors(10)
cols2 <- seqColors(10)
cols3 <- divColors(10)
cols4 <- seqColors2(10)
plot(x, y, col=cols1, xlim=c(1, 13), pch=19, cex=3)
points(x+1, y, col=cols2, pch=19, cex=3)
points(x+2, y, col=cols3, pch=19, cex=3)
points(x+3, y, col=cols4, pch=19, cex=3)
abline(0, 1, lty=2)
abline(-1, 1, lty=2)
abline(-2, 1, lty=2)
abline(-3, 1, lty=2)
```

getX

Accessors for physical array coordinates.

Description

Accessors for physical array coordinates.

Usage

```
getX(object, type)
getY(object, type)
```

Arguments

object	FeatureSet object
type	'character' defining the type of the probes to be queried. Valid options are 'pm', 'mm', 'bg'

Value

A vector with the requested coordinates.

Examples

```
## Not run:
x <- read.celfiles(list.celfiles())
theXpm <- getX(x, "pm")
theYpm <- getY(x, "pm")

## End(Not run)
```

 crlmm

Genotype Calls

Description

Performs genotype calls via CRLMM (Corrected Robust Linear Model with Maximum-likelihood based distances).

Usage

```
crlmm(filenamees, outdir, batch_size=40000, balance=1.5,
        minLLRforCalls=c(5, 1, 5), recalibrate=TRUE,
        verbose=TRUE, pkgname, reference=TRUE)
justCRLMM(filenamees, batch_size = 40000, minLLRforCalls = c(5, 1, 5),
           recalibrate = TRUE, balance = 1.5, phenoData = NULL, verbose = TRUE,
           pkgname = NULL, tmpdir=tmpdir())
```

Arguments

filenamees	character vector with the filenames.
outdir	directory where the output (and some tmp files) files will be saved.
batch_size	integer defining how many SNPs should be processed at a time.
recalibrate	Logical - should recalibration be performed?
balance	Control parameter to balance homozygotes and heterozygotes calls.
minLLRforCalls	Minimum thresholds for genotype calls.
verbose	Logical.
phenoData	phenoData object or NULL
pkgname	alt. pdInfo package to be used
reference	logical, defaulting to TRUE ...
tmpdir	Directory where temporary files are going to be stored at.

Value

SnpCallSetPlus object.

 fitPLM

Fit a Probe Level Model to Affymetrix Genechip Data.

Description

This function converts an [FeatureSet](#) into an [PLMset](#) by fitting a specified robust linear model to the probe level data.

Usage

```
fitPLM(object, model=PM ~ -1 + probes + samples,
       variable.type=c(default="factor"),
       constraint.type=c(default="contr.treatment"),
       subset=NULL,
       background=TRUE, normalize=TRUE, background.method="RMA.2",
       normalize.method="quantile", background.param=list(),
       normalize.param=list(), output.param=NULL,
       model.param=NULL,
       verbosity.level=0)
```

Arguments

object	an FeatureSet object
model	A formula describing the model to fit. This is slightly different from the standard method of specifying formulae in R. Read the description below
variable.type	a way to specify whether variables in the model are factors or standard variables
constraint.type	should factor variables sum to zero or have first variable set to zero (endpoint constraint)
subset	a vector with the names of probesets to be used. If NULL then all probesets are used.
normalize	logical value. If TRUE normalize data using quantile normalization
background	logical value. If TRUE background correct using RMA background correction
background.method	name of background method to use.
normalize.method	name of normalization method to use.
background.param	A list of parameters for background routines
normalize.param	A list of parameters for normalization routines
output.param	A list of parameters controlling optional output from the routine.
model.param	A list of parameters controlling model procedure
verbosity.level	An integer specifying how much to print out. Higher values indicate more verbose. A value of 0 will print nothing

Details

This function fits robust Probe Level linear Models to all the probesets in an [FeatureSet](#). This is carried out on a probeset by probeset basis. The user has quite a lot of control over which model is used and what outputs are stored. For more details please read the vignette and the original documentation in the [affyPLM](#) package.

Value

An [PLMset](#)

Note

This is the initial port of `fitPLM` to `oligo`. Some features found on the original work by Ben Bolstad (in the `affyPLM` package) may not be yet available. If you found one of this missing characteristics, please contact Benilton Carvalho.

Author(s)

This is a port from Ben Bolstad's work implemented in the `affyPLM` package. Problems with the implementation in `oligo` should be reported to Benilton Carvalho <Benilton.Carvalho@cancer.org.uk>

References

Bolstad, BM (2004) *Low Level Analysis of High-density Oligonucleotide Array Data: Background, Normalization and Summarization*. PhD Dissertation. University of California, Berkeley.

See Also

`rma`, `summarize`

Examples

```
if (require(oligoData)) {  
  data(nimbleExpressionFS)  
  fit <- fitPLM(nimbleExpressionFS)  
  image(fit)  
  NUSE(fit)  
  RLE(fit)  
}
```

```
getAffinitySplineCoefficients  
  Estimate affinity coefficients.
```

Description

Estimate affinity coefficients using sequence information and splines.

Usage

```
getAffinitySplineCoefficients(intensities, sequences)
```

Arguments

```
intensities  Intensity matrix  
sequences    Probe sequences
```

Value

Matrix with estimated coefficients.

See Also

`getBaseProfile`

getBaseProfile *Compute and plot nucleotide profile.*

Description

Computes and, optionally, plots nucleotide profile, describing the sequence effect on intensities.

Usage

```
getBaseProfile(coefs, probeLength = 25, plot = FALSE, ...)
```

Arguments

coefs	affinity spline coefficients.
probeLength	length of probes
plot	logical. Plots profile?
...	arguments to be passed to matplot.

Value

Invisibly returns a matrix with estimated effects.

getContainer *Get container information for NimbleGen Tiling Arrays.*

Description

Get container information for NimbleGen Tiling Arrays. This is useful for better identification of control probes.

Usage

```
getContainer(object, probeType)
```

Arguments

object	A TilingFeatureSet or TilingFeatureSet object.
probeType	String describing which probes to query ('pm', 'bg')

Value

'character' vector with container information.

`getCrlmmSummaries` *Function to get CRLMM summaries saved to disk*

Description

This will read the summaries written to disk and return them to the user as a `SnpCallSetPlus` or `SnpCnvCallSetPlus` object.

Usage

```
getCrlmmSummaries(tmpdir)
```

Arguments

`tmpdir` directory where CRLMM saved the results to.

Value

If the data were from SNP 5.0 or 6.0 arrays, the function will return a `SnpCnvCallSetPlus` object. It will return a `SnpCallSetPlus` object, otherwise.

`getNetAffx` *NetAffx Biological Annotations*

Description

Gets NetAffx Biological Annotations saved in the annotation package (Exon and Gene ST Affymetrix arrays).

Usage

```
getNetAffx(object, type = "probeset")
```

Arguments

`object` 'ExpressionSet' object (eg., result of `rma()`)
`type` Either 'probeset' or 'transcript', depending on what type of summaries were obtained.

Details

This retrieves NetAffx annotation saved in the (pd) annotation package - `annotation(object)`. It is only available for Exon ST and Gene ST arrays.

The 'type' argument should match the summarization target used to generate 'object'. The 'rma' method allows for two targets: 'probeset' (target='probeset') and 'transcript' (target='core', target='full', target='extended').

Value

'AnnotatedDataFrame' that can be used as `featureData(object)`

Author(s)

Benilton Carvalho

getNgsColorsInfo *Helper function to extract color information for filenames on Nimble-Gen arrays.*

Description

This function will (try to) extract the color information for NimbleGen arrays. This is useful when using `read.xysfiles2` to parse XYS files for Tiling applications.

Usage

```
getNgsColorsInfo(path = ".", pattern1 = "_532", pattern2 = "_635", ...)
```

Arguments

<code>path</code>	path where to look for files
<code>pattern1</code>	pattern to match files supposed to go to the first channel
<code>pattern2</code>	pattern to match files supposed to go to the second channel
<code>...</code>	extra arguments for <code>list.xysfiles</code>

Details

Many NimbleGen samples are identified following the pattern `sampleID_532.XYS / sampleID_635.XYS`.

The function suggests sample names if all the filenames follow the standard above.

Value

A `data.frame` with, at least, two columns: `'channel1'` and `'channel2'`. A third column, `'sample-Names'`, is returned if the filenames follow the `sampleID_532.XYS / sampleID_635.XYS` standard.

Author(s)

Benilton Carvalho <bcarvalh@jhsp.edu>

Details

The 'transfo' argument will set the transformation to be used. For raw data, 'transfo=log2' is a common practice. For summarized data (which are often in log2-scale), no transformation is needed (therefore 'transfo=identity').

Note

The hist methods for `FeatureSet` and `Expression` use a sample (via `sample`) of the probes/probesets to produce the plot (unless `nsample > nrow(x)`). Therefore, the user interested in reproducibility is advised to use `set.seed`.

image	<i>Display a pseudo-image of a microarray chip</i>
-------	----------------------------------------------------

Description

Produces a pseudo-image (`graphics::image`) for each sample.

Usage

```
## S4 method for signature 'FeatureSet'
image(x, which, transfo=log2, ...)

## S4 method for signature 'PLMset'
image(x, which=0,
      type=c("weights", "resids", "pos.resids", "neg.resids", "sign.resids"),
      use.log=TRUE, add.legend=FALSE, standardize=FALSE,
      col=NULL, main, ...)
```

Arguments

x	FeatureSet object
which	integer indices of samples to be plotted (optional).
transfo	function to be applied to the data prior to plotting.
type	Type of statistics to be used.
use.log	Use log.
add.legend	Add legend.
standardize	Standardize residuals.
col	Colors to be used.
main	Main title.
...	parameters to be passed to <code>image</code>

Examples

```

if(require(oligoData) & require(pd.hg18.60mer.expr)){
  data(nimbleExpressionFS)
  par(mfrow=c(1, 2))
  image(nimbleExpressionFS, which=4)
  fit <- fitPLM(nimbleExpressionFS)
  image(fit, which=4)
}

```

justSNPRMA

Summarization of SNP data

Description

This function implements the SNPRMA method for summarization of SNP data. It works directly with the CEL files, saving memory.

Usage

```
justSNPRMA(filenamees, verbose = TRUE, phenoData = NULL, normalizeToHapmap = TRUE)
```

Arguments

filenamees character vector with the filenames.
verbose logical flag for verbosity.
phenoData a phenoData object or NULL
normalizeToHapmap
 Normalize to Hapmap? Should always be TRUE, but it's kept here for future use.

Value

SnpQSet or a SnpCnvQSet, depending on the array type.

Examples

```
## snprmaResults <- justSNPRMA(list.celfiles())
```

list.xysfiles

List XYS files

Description

Lists the XYS files.

Usage

```
list.xysfiles(...)
```

Arguments

... parameters to be passed to `list.files`

Details

The functions interface `list.files` and the user is asked to check that function for further details.

Value

Character vector with the filenames.

See Also

`list.files`

Examples

```
list.xysfiles()
```

oligo-package	<i>The oligo package: a tool for low-level analysis of oligonucleotide arrays</i>
---------------	-----------------------------------------------------------------------------------

Description

The **oligo** package provides tools to preprocess different oligonucleotide arrays types: expression, tiling, SNP and exon chips. The supported manufacturers are Affymetrix and NimbleGen.

It offers support to large datasets (when the **bigmemory** is loaded) and can execute preprocessing tasks in parallel (if, in addition to **bigmemory**, the **snow** package is also loaded).

Details

The package will read the raw intensity files (CEL for Affymetrix; XYS for NimbleGen) and allow the user to perform analyses starting at the feature-level.

Reading in the intensity files require the existence of data packages that contain the chip specific information (X/Y coordinates; feature types; sequence). These data packages packages are built using the **pdInfoBuilder** package.

For Affymetrix SNP arrays, users are asked to download the already built annotation packages from BioConductor. This is because these packages contain metadata that are not automatically created. The following annotation packages are available:

50K Xba - pd.mapping50kxba.240 50K Hind - pd.mapping50khind.240 250K Sty - pd.mapping250k.sty
250K Nsp - pd.mapping250k.nsp GenomeWideSnp 5 (SNP 5.0) - pd.genomewidesnp.5 GenomeWideSnp
6 (SNP 6.0) - pd.genomewidesnp.6

For users interested in genotype calls for SNP 5.0 and 6.0 arrays, we strongly recommend the use use the **crlmm** package, which implements a more efficient version of CRLMM.

Author(s)

Benilton Carvalho - <carvalho@bclab.org>

References

Carvalho, B.; Bengtsson, H.; Speed, T. P. & Irizarry, R. A. Exploration, Normalization, and Genotype Calls of High Density Oligonucleotide SNP Array Data. Biostatistics, 2006.

paCalls

Methods for P/A Calls

Description

Methods for Present/Absent Calls are meant to provide means of assessing whether or not each of the (PM) intensities are compatible with observations generated by background probes.

Usage

```
paCalls(object, method, ..., verbose=TRUE)
## S4 method for signature 'ExonFeatureSet'
paCalls(object, method, verbose = TRUE)
## S4 method for signature 'GeneFeatureSet'
paCalls(object, method, verbose = TRUE)
## S4 method for signature 'ExpressionFeatureSet'
paCalls(object, method, ..., verbose = TRUE)
```

Arguments

object	Exon/Gene/Expression-FeatureSet object.
method	String defining what method to use. See 'Details'.
...	Additional arguments passed to MAS5. See 'Details'
verbose	Logical flag for verbosity.

Details

For Whole Transcript arrays (Exon/Gene) the valid options for `method` are 'DABG' (p-values for each probe) and 'PSDABG' (p-values for each probeset). For Expression arrays, the only option currently available for `method` is 'MAS5'.

ABOUT MAS5 CALLS:

The additional arguments that can be passed to MAS5 are:

1. `alpha1`: a significance threshold in (0, alpha2);
2. `alpha2`: a significance threshold in (alpha1, 0.5);
3. `tau`: a small positive constant;
4. `ignore.saturated`: if TRUE, do the saturation correction described in the paper, with a saturation level of 46000;

This function performs the hypothesis test:

H0: $\text{median}(R_i) = \tau$, corresponding to absence of transcript
 H1: $\text{median}(R_i) > \tau$, corresponding to presence of transcript

where $R_i = (PM_i - MM_i) / (PM_i + MM_i)$ for each i a probe-pair in the probe-set represented by data.

The p-value that is returned estimates the usual quantity:

Pr(observing a more "present looking" probe-set than data | data is absent)

So that small p-values imply presence while large ones imply absence of transcript. The detection call is computed by thresholding the p-value as in:

call "P" if p-value < alpha1 call "M" if alpha1 <= p-value < alpha2 call "A" if alpha2 <= p-value

Value

A matrix (of dimension dim(PM) if method="DABG" or "MAS5"; of dimension length(unique(probeNames(object))) x ncol(object) if method="PSDABG") with p-values for P/A Calls.

Author(s)

Benilton Carvalho

References

Clark et al. Discovery of tissue-specific exons using comprehensive human exon microarrays. *Genome Biol* (2007) vol. 8 (4) pp. R64

Liu, W. M. and Mei, R. and Di, X. and Ryder, T. B. and Hubbell, E. and Dee, S. and Webster, T. A. and Harrington, C. A. and Ho, M. H. and Baid, J. and Smeekens, S. P. (2002) Analysis of high density expression microarrays with signed-rank call algorithms, *Bioinformatics*, 18(12), pp. 1593–1599.

Liu, W. and Mei, R. and Bartell, D. M. and Di, X. and Webster, T. A. and Ryder, T. (2001) Rank-based algorithms for analysis of microarrays, *Proceedings of SPIE, Microarrays: Optical Technologies and Informatics*, 4266.

Affymetrix (2002) Statistical Algorithms Description Document, Affymetrix Inc., Santa Clara, CA, [whitepaper](http://www.affymetrix.com/support/technical/whitepapers/sadd_whitepaper.pdf). http://www.affymetrix.com/support/technical/whitepapers/sadd_whitepaper.pdf

Examples

```
## Not run:
if (require(oligoData) & require(pd.huex.1.0.st.v2)){
  data(affyExonFS)
  ## Get only 2 samples for example
  dabgP = paCalls(affyExonFS[, 1:2])
  dabgPS = paCalls(affyExonFS[, 1:2], "PSDABG")
  head(dabgP) ## for probe
  head(dabgPS) ## for probeset
}

## End(Not run)
```

Description

The plotM methods are meant to plot log-ratios for different classes of data.

Methods

object = "SnpQSet", i = "character" Plot log-ratio for SNP data for sample i.
object = "SnpQSet", i = "integer" Plot log-ratio for SNP data for sample i.
object = "SnpQSet", i = "numeric" Plot log-ratio for SNP data for sample i.
object = "TilingQSet", i = "missing" Plot log-ratio for Tiling data for sample i.

pmAllele *Access the allele information for PM probes.*

Description

Accessor to the allelic information for PM probes.

Usage

```
pmAllele(object)
```

Arguments

object SnpFeatureSet or PDInfo object.

pmFragmentLength *Access the fragment length for PM probes.*

Description

Accessor to the fragment length for PM probes.

Usage

```
pmFragmentLength(object, enzyme, type=c('snp', 'cn'))
```

Arguments

object PDInfo or SnpFeatureSet object.
enzyme Enzyme to be used for query. If missing, all enzymes are used.
type Type of probes to be used: 'snp' for SNP probes; 'cn' for Copy Number probes.

Value

A list of length equal to the number of enzymes used for digestion. Each element of the list is a data.frame containing:

- row: the row used to link to the PM matrix;
- length: expected fragment length.

Note

There is not a 1:1 relationship between probes and expected fragment length. For one enzyme, a given probe may be associated to multiple fragment lengths. Therefore, the number of rows in the data.frame may not match the number of PM probes and the row column should be used to match the fragment length with the PM matrix.

pmPosition	<i>Accessor to position information</i>
------------	-----------------------------------------

Description

pmPosition will return the genomic position for the (PM) probes.

Usage

```
pmPosition(object)
pmOffset(object)
```

Arguments

object AffySNPPDInfo, TilingFeatureSet or SnpCallSet object

Details

pmPosition will return genomic position for PM probes on a tiling array.

pmOffset will return the offset information for PM probes on SNP arrays.

pmStrand	<i>Accessor to the strand information</i>
----------	-------------------------------------------

Description

Returns the strand information on SNP arrays for PM probes (0 - sense / 1 - antisense).

Usage

```
pmStrand(object)
```

Arguments

object AffySNPPDInfo object

summarize

Tools for microarray preprocessing.

Description

These are tools to preprocess microarray data. They include background correction, normalization and summarization methods.

Usage

```
backgroundCorrectionMethods()
normalizationMethods()
backgroundCorrect(object, method=backgroundCorrectionMethods(), copy=TRUE, extra
summarize(object, probes=rownames(object), method="medianpolish", verbose=TRUE,
normalize(object, method=normalizationMethods(), copy=TRUE, verbose=TRUE, ...)
normalizeToTarget(object, target, method="quantile", copy=TRUE, verbose=TRUE)
```

Arguments

object	Object containing probe intensities to be preprocessed.
method	String determining which method to use at that preprocessing step.
target	Vector with the target distribution
probes	Character vector that identifies the name of the probes represented by the rows of object.
copy	Logical flag determining if data must be copied before processing (TRUE), or if data can be overwritten (FALSE).
extra	Extra arguments to be passed to other methods.
verbose	Logical flag for verbosity.
...	Arguments to be passed to methods.

Details

Number of rows of `object` must match the length of `probes`.

Value

`backgroundCorrectionMethods` and `normalizationMethods` will return a character vector with the methods implemented currently.

`backgroundCorrect`, `normalize` and `normalizeToTarget` will return a matrix with same dimensions as the input matrix. If they are applied to a `FeatureSet` object, the PM matrix will be used as input.

The `summarize` method will return a matrix with `length(unique(probes))` rows and `ncol(object)` columns.

Examples

```

ns <- 100
nps <- 1000
np <- 10
intensities <- matrix(rnorm(ns*nps*np, 8000, 400), nc=ns)
ids <- rep(as.character(1:nps), each=np)
bgCorrected <- backgroundCorrect(intensities)
normalized <- normalize(bgCorrected)
expression <- summarize(normalized, probes=ids)
intensities[1:20, 1:3]
expression[1:20, 1:3]
target <- rnorm(np*nps)
normalizedToTarget <- normalizeToTarget(intensities, target)

if (require(oligoData) & require(pd.hg18.60mer.expr)){
  ## Example of normalization with real data
  data(nimbleExpressionFS)
  boxplot(nimbleExpressionFS, main='Original')
  for (mtd in normalizationMethods()){
    message('Normalizing with ', mtd)
    res <- normalize(nimbleExpressionFS, method=mtd, verbose=FALSE)
    boxplot(res, main=mtd)
  }
}

```

probeNames

Accessor to feature names

Description

Accessors to featureset names.

Usage

```

probeNames(object, subset = NULL)
probesetNames(object)

```

Arguments

object	FeatureSet or DBPDInfo
subset	not implemented yet.

Value

probeNames returns a string with the probeset names for *each probe* on the array. probesetNames, on the other hand, returns the *unique probeset names*.

read.celfiles *Parser to CEL files*

Description

Reads CEL files.

Usage

```
read.celfiles(..., filenames, pkgname, phenoData, featureData,
experimentData, protocolData, notes, verbose=TRUE, sampleNames,
rm.mask=FALSE, rm.outliers=FALSE, rm.extra=FALSE, checkType=TRUE)
```

```
read.celfiles2(channel1, channel2, pkgname, phenoData, featureData,
experimentData, protocolData, notes, verbose=TRUE, sampleNames,
rm.mask=FALSE, rm.outliers=FALSE, rm.extra=FALSE, checkType=TRUE)
```

Arguments

...	names of files to be read.
filenames	a character vector with the CEL filenames.
channel1	a character vector with the CEL filenames for the first 'channel' on a Tiling application
channel2	a character vector with the CEL filenames for the second 'channel' on a Tiling application
pkgname	alternative data package to be loaded.
phenoData	phenoData
featureData	featureData
experimentData	experimentData
protocolData	protocolData
notes	notes
verbose	logical
sampleNames	character vector with sample names (usually better descriptors than the file-names)
rm.mask	logical. Read masked?
rm.outliers	logical. Remove outliers?
rm.extra	logical. Remove extra?
checkType	logical. Check type of each file? This can be time consuming.

Details

When using 'affyio' to read in CEL files, the user can read compressed CEL files (CEL.gz). Additionally, 'affyio' is much faster than 'affxparser'.

The function guesses which annotation package to use from the header of the CEL file. The user can also provide the name of the annotation package to be used (via the pkgname argument). If the annotation package cannot be loaded, the function returns an error. If the annotation package is not available from BioConductor, one can use the pdInfoBuilder package to build one.

Value

```

ExpressionFeatureSet
    if Expression arrays
ExonFeatureSet
    if Exon arrays
SnpFeatureSet
    if SNP arrays
TilingFeatureSet
    if Tiling arrays

```

See Also

[list.celfiles](#), [read.xysfiles](#)

Examples

```

if(require(pd.mapping50k.xba240) & require(hapmap100kxba)){
celPath <- system.file("celFiles", package="hapmap100kxba")
celFiles <- list.celfiles(celPath, full.name=TRUE)
affySnpFeatureSet <- read.celfiles(celFiles)
}

```

read.xysfiles *Parser to XYS files*

Description

NimbleGen provides XYS files which are read by this function.

Usage

```

read.xysfiles(..., filenames, pkgname, phenoData, featureData,
experimentData, protocolData, notes, verbose=TRUE, sampleNames,
checkType=TRUE)

```

```

read.xysfiles2(channel1, channel2, pkgname, phenoData, featureData,
experimentData, protocolData, notes, verbose=TRUE, sampleNames,
checkType=TRUE)

```

Arguments

...	file names
filenames	character vector with filenames.
channel1	a character vector with the XYS filenames for the first 'channel' on a Tiling application
channel2	a character vector with the XYS filenames for the second 'channel' on a Tiling application
pkgname	character vector with alternative PD Info package name
phenoData	phenoData

```

featureData  featureData
experimentData
              experimentData

protocolData protocolData
notes        notes
verbose      verbose
sampleNames  character vector with sample names (usually better descriptors than the file-
              names)
checkType    logical. Check type of each file? This can be time consuming.

```

Details

The function will read the XYS files provided by NimbleGen Systems and return an object of class FeatureSet.

The function guesses which annotation package to use from the header of the XYS file. The user can also provide the name of the annotation package to be used (via the `pkgname` argument). If the annotation package cannot be loaded, the function returns an error. If the annotation package is not available from BioConductor, one can use the `pdInfoBuilder` package to build one.

Value

```

ExpressionFeatureSet
              if Expression arrays
TilingFeatureSet
              if Tiling arrays

```

See Also

[list.xysfiles](#), [read.celfiles](#)

Examples

```

if (require(maqcExpression4plex) & require(pd.hg18.60mer.expr)) {
xysPath <- system.file("extdata", package="maqcExpression4plex")
xysFiles <- list.xysfiles(xysPath, full.name=TRUE)
ngsExpressionFeatureSet <- read.xysfiles(xysFiles)
}

```

readSummaries *Read summaries generated by crlmm*

Description

This function read the different summaries generated by crlmm.

Usage

```
readSummaries(type, tmpdir)
```

Arguments

`type` type of summary of character class: 'alleleA', 'alleleB', 'alleleA-sense', 'alleleA-antisense', 'alleleB-sense', 'alleleB-antisense', 'calls', 'llr', 'conf'.

`tmpdir` directory containing the output saved by crlmm

Details

On the 50K and 250K arrays, given a SNP, there are probes on both strands (sense and antisense). For this reason, the options 'alleleA-sense', 'alleleA-antisense', 'alleleB-sense' and 'alleleB-antisense' should be used **only** with such arrays (XBA, HIND, NSP or STY).

On the SNP 5.0 and SNP 6.0 platforms, this distinction does not exist in terms of algorithm (note that the actual strand could be queried from the annotation package). For these arrays, options 'alleleA', 'alleleB' are the ones to be used.

The options `calls`, `llr` and `conf` will return, respectively, the CRLMM calls, log-likelihood ratios (for devel purpose **only**) and CRLMM confidence calls matrices.

Value

Matrix with values of summaries.

 rma-methods

RMA - Robust Multichip Average algorithm

Description

Robust Multichip Average preprocessing methodology. This strategy allows background subtraction, quantile normalization and summarization (via median-polish).

Usage

```
## S4 method for signature 'ExonFeatureSet'
rma(object, background=TRUE, normalize=TRUE, subset=NULL, target="core")
## S4 method for signature 'ExpressionFeatureSet'
rma(object, background=TRUE, normalize=TRUE, subset=NULL)
## S4 method for signature 'GeneFeatureSet'
rma(object, background=TRUE, normalize=TRUE, subset=NULL, target="core")
## S4 method for signature 'SnpCnvFeatureSet'
rma(object, background=TRUE, normalize=TRUE, subset=NULL)
```

Arguments

`object` Exon/Expression/Gene/SnpCnv-FeatureSet object.

`background` Logical - perform RMA background correction?

`normalize` Logical - perform quantile normalization?

`subset` To be implemented.

`target` Level of summarization (only for Exon/Gene arrays)

Methods

`signature(object = "ExonFeatureSet")` When applied to an `ExonFeatureSet` object, `rma` can produce summaries at different levels: probeset (as defined in the PGF), core genes (as defined in the `core.mps` file), full genes (as defined in the `full.mps` file) or extended genes (as defined in the `extended.mps` file). To determine the level for summarization, use the `target` argument.

`signature(object = "ExpressionFeatureSet")` When used on an `ExpressionFeatureSet` object, `rma` produces summaries at the probeset level (as defined in the CDF or NDF files, depending on the manufacturer).

`signature(object = "GeneFeatureSet")` When applied to a `GeneFeatureSet` object, `rma` can produce summaries at different levels: probeset (as defined in the PGF) and 'core genes' (as defined in the `core.mps` file). To determine the level for summarization, use the `target` argument.

`signature(object = "SnpCnvFeatureSet")` If used on a `SnpCnvFeatureSet` object (ie., SNP 5.0 or SNP 6.0 arrays), `rma` will produce summaries for the CNV probes. Note that this is an experimental feature for internal (and quick) assessment of CNV probes. We recommend the use of the 'crlmm' package, which contains a Copy Number tool specifically designed for these data.

References

Rafael. A. Irizarry, Benjamin M. Bolstad, Francois Collin, Leslie M. Cope, Bridget Hobbs and Terence P. Speed (2003), Summaries of Affymetrix GeneChip probe level data *Nucleic Acids Research* 31(4):e15

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(2):185-193

Irizarry, RA, Hobbs, B, Collin, F, Beazer-Barclay, YD, Antonellis, KJ, Scherf, U, Speed, TP (2003) Exploration, Normalization, and Summaries of High Density Oligonucleotide Array Probe Level Data. *Biostatistics*. Vol. 4, Number 2: 249-264

See Also

[snprma](#)

Examples

```
if (require(maqcExpression4plex) & require(pd.hg18.60mer.expr)) {
  xysPath <- system.file("extdata", package="maqcExpression4plex")
  xysFiles <- list.xysfiles(xysPath, full.name=TRUE)
  ngsExpressionFeatureSet <- read.xysfiles(xysFiles)
  summarized <- rma(ngsExpressionFeatureSet)
  show(summarized)
}
```

runDate	<i>Date of scan</i>
---------	---------------------

Description

Retrieves date information in CEL/XYS files.

Usage

```
runDate(object)
```

Arguments

object 'FeatureSet' object.

sequenceDesignMatrix	<i>Create design matrix for sequences</i>
----------------------	-------------------------------------------

Description

Creates design matrix for sequences.

Usage

```
sequenceDesignMatrix(seqs)
```

Arguments

seqs character vector of 25-mers.

Details

This assumes all sequences are 25bp long.

The design matrix is often used when the objective is to adjust intensities by sequence.

Value

Matrix with length(seqs) rows and 75 columns.

Examples

```
genSequence <- function(x)
  paste(sample(c("A", "T", "C", "G"), 25, rep=TRUE), collapse="", sep="")
seqs <- sapply(1:10, genSequence)
X <- sequenceDesignMatrix(seqs)
Y <- rnorm(10, mean=12, sd=2)
Ydemean <- Y-mean(Y)
X[1:10, 1:3]
fit <- lm(Ydemean~X)
coef(fit)
```

`snprma`*Preprocessing SNP Arrays*

Description

This function preprocess SNP arrays.

Usage

```
snprma(object, verbose = TRUE, normalizeToHapmap = TRUE)
```

Arguments

<code>object</code>	<code>SnpFeatureSet</code> object
<code>verbose</code>	Verbosity flag. logical
<code>normalizeToHapmap</code>	internal

Value

A `SnpQSet` object.

Index

*Topic **IO**

read.celfiles, 28
read.xysfiles, 29

*Topic **classes**

PLMset-class, 5

*Topic **classif**

crlmm, 12
getNetAffx, 16
runDate, 33

*Topic **file**

list.xysfiles, 20

*Topic **hplot**

boxplot, 9
darkColors, 10
hist, 18
image, 19
MAplot, 3

*Topic **loess**

MAplot, 3

*Topic **manip**

basecontent, 7
basicPLM, 7
basicRMA, 8
chromosome, 10
fitPLM, 12
getAffinitySplineCoefficients,
14
getBaseProfile, 15
getContainer, 15
getCrlmmSummaries, 16
getNgsColorsInfo, 17
getPlatformDesign, 18
getX, 11
justSNPRMA, 20
mm, 2
mmindex, 1
mmSequence, 6
paCalls, 22
pmAllele, 24
pmFragmentLength, 24
pmPosition, 25
pmStrand, 25
probeNames, 27

readSummaries, 30
sequenceDesignMatrix, 33
snprma, 34
summarize, 26

*Topic **methods**

boxplot, 9
hist, 18
MAplot, 3
plotM-methods, 23
rma-methods, 31

*Topic **package**

oligo-package, 21

*Topic **smooth**

MAplot, 3

backgroundCorrect (*summarize*), 26
backgroundCorrect, FeatureSet-method
(*summarize*), 26
backgroundCorrect, ff_matrix-method
(*summarize*), 26
backgroundCorrect, matrix-method
(*summarize*), 26
backgroundCorrect-methods
(*summarize*), 26
backgroundCorrectionMethods
(*summarize*), 26
basecontent, 7
basicPLM, 7
basicRMA, 8, 8
bg (*mm*), 2
bg, FeatureSet-method (*mm*), 2
bg, TilingFeatureSet-method (*mm*), 2
bg<- (*mm*), 2
bg<-, FeatureSet, ff_matrix-method
(*mm*), 2
bg<-, FeatureSet, matrix-method
(*mm*), 2
bg<-, TilingFeatureSet, array-method
(*mm*), 2
bgindex (*mmindex*), 1
bgindex, DBPDInfo-method
(*mmindex*), 1
bgindex, FeatureSet-method
(*mmindex*), 1

- bgSequence (*mmSequence*), 6
- bgSequence, DBPDInfo-method (*mmSequence*), 6
- bgSequence, ExonFeatureSet-method (*mmSequence*), 6
- bgSequence, FeatureSet-method (*mmSequence*), 6
- bgSequence, GeneFeatureSet-method (*mmSequence*), 6
- boxplot, 9
- boxplot, ExpressionSet-method (*boxplot*), 9
- boxplot, FeatureSet-method (*boxplot*), 9
- boxplot, PLMset-method (*boxplot*), 9
- boxplot-methods (*boxplot*), 9
- chromosome, 10
- chromosome<- (*chromosome*), 10
- chromosome<-, AnnotatedDataFrame, character-method (*chromosome*), 10
- cleanPlatformName (*read.celfiles*), 28
- coefs (*PLMset-class*), 5
- coefs, PLMset-method (*PLMset-class*), 5
- crlmm, 12
- darkColors, 10
- divColors (*darkColors*), 10
- FeatureSet, 12, 13
- fitPLM, 5, 12
- getAffinitySplineCoefficients, 14
- getBaseProfile, 15
- getContainer, 15
- getContainer, TilingFeatureSet-method (*getContainer*), 15
- getContainer-methods (*getContainer*), 15
- getCrlmmSummaries, 16
- getNetAffx, 16
- getNetAffx, ExpressionSet-method (*getNetAffx*), 16
- getNetAffx-methods (*getNetAffx*), 16
- getNgsColorsInfo, 17
- getPD (*getPlatformDesign*), 18
- getPlatformDesign, 18
- getPlatformDesign, FeatureSet-method (*getPlatformDesign*), 18
- getX, 11
- getX, DBPDInfo-method (*getX*), 11
- getX, FeatureSet-method (*getX*), 11
- getX-methods (*getX*), 11
- getY (*getX*), 11
- getY, DBPDInfo-method (*getX*), 11
- getY, FeatureSet-method (*getX*), 11
- getY-methods (*getX*), 11
- hist, 9, 18
- hist, ExpressionSet-method (*hist*), 18
- hist, FeatureSet-method (*hist*), 18
- hist-methods (*hist*), 18
- image, 9, 19
- image, FeatureSet-method (*image*), 19
- image, PLMset-method (*image*), 19
- image-methods (*image*), 19
- intensity (*mm*), 2
- intensity, FeatureSet-method (*mm*), 2
- intensity<- (*mm*), 2
- intensity<-, FeatureSet-method (*mm*), 2
- justCRLMM (*crlmm*), 12
- justSNPRMA, 20
- list.celfiles, 29
- list.files, 21
- list.xysfiles, 20, 30
- loess, 4
- MAplot, 3
- MAplot, ExpressionSet-method (*MAplot*), 3
- MAplot, FeatureSet-method (*MAplot*), 3
- MAplot, matrix-method (*MAplot*), 3
- MAplot, PLMset-method (*MAplot*), 3
- MAplot, TilingFeatureSet-method (*MAplot*), 3
- MAplot-methods (*MAplot*), 3
- mm, 2
- mm, FeatureSet-method (*mm*), 2
- mm, TilingFeatureSet-method (*mm*), 2
- mm<- (*mm*), 2
- mm<-, FeatureSet, ff_matrix-method (*mm*), 2
- mm<-, FeatureSet, matrix-method (*mm*), 2
- mm<-, TilingFeatureSet, array-method (*mm*), 2

- mmindex, 1
- mmindex, DBPDInfo-method
(*mmindex*), 1
- mmindex, FeatureSet-method
(*mmindex*), 1
- mmSequence, 6
- mmSequence, AffySNPPDInfo-method
(*mmSequence*), 6
- mmSequence, DBPDInfo-method
(*mmSequence*), 6
- mmSequence, FeatureSet-method
(*mmSequence*), 6
- normalizationMethods (*summarize*),
26
- normalize (*summarize*), 26
- normalize, FeatureSet-method
(*summarize*), 26
- normalize, ff_matrix-method
(*summarize*), 26
- normalize, matrix-method
(*summarize*), 26
- normalize-methods (*summarize*), 26
- normalizeToTarget (*summarize*), 26
- normalizeToTarget, ff_matrix-method
(*summarize*), 26
- normalizeToTarget, matrix-method
(*summarize*), 26
- normalizeToTarget-methods
(*summarize*), 26
- NUSE (*PLMset-class*), 5
- NUSE, PLMset-method
(*PLMset-class*), 5
- oligo-package, 21
- paCalls, 22
- paCalls, ExonFeatureSet-method
(*paCalls*), 22
- paCalls, ExpressionFeatureSet-method
(*paCalls*), 22
- paCalls, GeneFeatureSet-method
(*paCalls*), 22
- PLMset, 12, 13
- PLMset (*PLMset-class*), 5
- PLMset-class, 5
- plot, 4
- plotM (*plotM-methods*), 23
- plotM, SnpQSet, character-method
(*plotM-methods*), 23
- plotM, SnpQSet, integer-method
(*plotM-methods*), 23
- plotM, SnpQSet, numeric-method
(*plotM-methods*), 23
- plotM, TilingQSet, missing-method
(*plotM-methods*), 23
- plotM-methods, 23
- pm (*mm*), 2
- pm, FeatureSet-method (*mm*), 2
- pm, TilingFeatureSet-method (*mm*), 2
- pm<- (*mm*), 2
- pm<-, FeatureSet, ff_matrix-method
(*mm*), 2
- pm<-, FeatureSet, matrix-method
(*mm*), 2
- pm<-, TilingFeatureSet, array-method
(*mm*), 2
- pmAllele, 24
- pmAllele, AffySNPPDInfo-method
(*pmAllele*), 24
- pmAllele, SnpFeatureSet-method
(*pmAllele*), 24
- pmChr (*chromosome*), 10
- pmChr, ExonFeatureSet-method
(*chromosome*), 10
- pmChr, FeatureSet-method
(*chromosome*), 10
- pmChr, GeneFeatureSet-method
(*chromosome*), 10
- pmFragmentLength, 24
- pmFragmentLength, AffySNPPDInfo-method
(*pmFragmentLength*), 24
- pmFragmentLength, SnpFeatureSet-method
(*pmFragmentLength*), 24
- pmindex (*mmindex*), 1
- pmindex, DBPDInfo-method
(*mmindex*), 1
- pmindex, FeatureSet-method
(*mmindex*), 1
- pmOffset (*pmPosition*), 25
- pmOffset, AffySNPPDInfo-method
(*pmPosition*), 25
- pmPosition, 25
- pmPosition, ExpressionPDInfo-method
(*pmPosition*), 25
- pmPosition, FeatureSet-method
(*pmPosition*), 25
- pmPosition, TilingFeatureSet-method
(*pmPosition*), 25
- pmPosition, TilingPDInfo-method
(*pmPosition*), 25
- pmSequence (*mmSequence*), 6
- pmSequence, AffySNPPDInfo-method
(*mmSequence*), 6

- pmSequence, DBPDInfo-method
(*mmSequence*), 6
- pmSequence, FeatureSet-method
(*mmSequence*), 6
- pmStrand, 25
- pmStrand, AffySNPPDInfo-method
(*pmStrand*), 25
- probeNames, 27
- probeNames, DBPDInfo-method
(*probeNames*), 27
- probeNames, ExonFeatureSet-method
(*probeNames*), 27
- probeNames, FeatureSet-method
(*probeNames*), 27
- probeNames, GeneFeatureSet-method
(*probeNames*), 27
- probesetNames (*probeNames*), 27
- probesetNames, FeatureSet-method
(*probeNames*), 27

- rcModelPLM, 8
- rcModelPLMr, 8
- rcModelPLMrc, 8
- rcModelPLMrr, 8
- read.celfiles, 28, 30
- read.celfiles2 (*read.celfiles*), 28
- read.xysfiles, 29, 29
- read.xysfiles2 (*read.xysfiles*), 29
- readSummaries, 30
- RLE (*PLMset-class*), 5
- RLE, PLMset-method (*PLMset-class*),
5
- rma, 14
- rma (*rma-methods*), 31
- rma, ExonFeatureSet-method
(*rma-methods*), 31
- rma, ExpressionFeatureSet-method
(*rma-methods*), 31
- rma, GeneFeatureSet-method
(*rma-methods*), 31
- rma, SnpCnvFeatureSet-method
(*rma-methods*), 31
- rma-methods, 31
- runDate, 33
- runDate, FeatureSet-method
(*runDate*), 33
- runDate-methods (*runDate*), 33

- sample, 9
- se (*PLMset-class*), 5
- se, PLMset-method (*PLMset-class*), 5
- seqColors (*darkColors*), 10
- seqColors2 (*darkColors*), 10
- sequenceDesignMatrix, 33
- set.seed, 9
- show, PLMset-method
(*PLMset-class*), 5
- smoothScatter, 4
- snprma, 32, 34
- summarize, 14, 26
- summarize, ff_matrix-method
(*summarize*), 26
- summarize, matrix-method
(*summarize*), 26
- summarize-methods (*summarize*), 26