Package ‘oligo’

November 20, 2016

Version 1.38.0

Title Preprocessing tools for oligonucleotide arrays

Author Benilton Carvalho and Rafael Irizarry

Contributors Ben Bolstad, Vincent Carey, Wolfgang Huber, Harris Jaffee, Jim MacDonald, Matt Settles

Maintainer Benilton Carvalho <benilton@unicamp.br>

Depends R (>= 3.2.0), BiocGenerics (>= 0.13.11), oligoClasses (>= 1.29.6), Biobase (>= 2.27.3), Biostrings (>= 2.35.12)

Imports affyio (>= 1.35.0), affxparser (>= 1.39.4), DBI (>= 0.3.1), ff, graphics, methods, preprocessCore (>= 1.29.0), RSQLite (>= 1.0.0), splines, stats, stats4, utils, zlibbioc

Enhances ff, doMC, doMPI

LinkingTo preprocessCore

Suggests BSgenome.Hsapiens.UCSC.hg18, hapmap100kxba, pd.hg.u95av2, pd.mapping50k.xba240, pd.huex.1.0.st.v2, pd.hg18.60mer.expr, pd.hugene.1.0.st.v1, maqcExpression4plex, genefilter, limma, RColorBrewer, oligoData, BiocStyle, knitr, RUnit, biomaRt, AnnotationDbi, GenomeGraphs, RCurl, ACME, biomaRt, AnnotationDbi, GenomeGraphs, RCurl

VignetteBuilder knitr

Description A package to analyze oligonucleotide arrays (expression/SNP/tiling/exon) at probe-level. It currently supports Affymetrix (CEL files) and NimbleGen arrays (XYS files).

License LGPL (>= 2)


LazyLoad Yes
biocViews Microarray, OneChannel, TwoChannel, Preprocessing, SNP, DifferentialExpression, ExonArray, GeneExpression, DataImport

NeedsCompilation yes

R topics documented:

oligo-package ................................................................. 3
basecontent ................................................................. 3
basicPLM ................................................................. 4
basicRMA ................................................................. 5
boxplot ................................................................. 6
chromosome ............................................................... 7
crlmm ................................................................. 7
darkColors .............................................................. 8
fitProbeLevelModel .................................................. 9
getAffinitySplineCoefficients .................................. 10
getBaseProfile ...................................................... 11
getContainer .......................................................... 11
getCrlmmSummaries .................................................. 12
getNetAffx .............................................................. 12
getNgsColorsInfo .................................................... 13
getPlatformDesign .................................................... 14
getProbeInfo ............................................................ 14
getX ................................................................. 15
hist ................................................................. 16
image ................................................................. 16
justSNPRMA ............................................................. 17
list.xysfiles ............................................................. 18
MAplot ................................................................. 19
mm ................................................................. 21
mmindex .............................................................. 22
mmSequence ........................................................... 22
oligo-defunct .......................................................... 23
oligoPLM-class .......................................................... 23
paCalls ................................................................. 25
plotM-methods ........................................................... 27
pmAllele ................................................................. 27
pmFragmentLength ..................................................... 28
pmPosition .............................................................. 28
pmStrand ............................................................... 29
probeNames ............................................................. 29
read.celfiles ............................................................ 30
read.xysfiles ............................................................ 31
readSummaries .......................................................... 32
rma-methods ............................................................. 33
runDate ................................................................. 35
sequenceDesignMatrix ........................................ 35
snprma ................................................................. 36
summarize ............................................................... 36

Index ................................................................. 39
The oligo package: a tool for low-level analysis of oligonucleotide arrays

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 of the crlmm package, which implements a more efficient version of CRLMM.

Author(s)
Benilton Carvalho - <carvalho@bclab.org>

References

Sequence Base Contents

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

Usage
basecontent(seq)
basicPLM

Simplified interface to PLM.

Description
Simplified interface to PLM.

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

Arguments

pmMat Matrix of intensities to be processed.

pnVec Probeset names

normalize Logical flag: normalize?

background Logical flag: background adjustment?

transfo function: function to be used for data transformation prior to summarization.

method 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.

verbose 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

```r
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[['VarEstimates']][1:4, 1:3]
res[['VarStdErrors']][1:4, 1:3]
res[['VarResiduals']][1:20, 1:3]
```

---

**basicRMA**

Simplified interface to RMA.

**Description**

Simple interface to RMA.

**Usage**

```r
basicRMA(pmMat, pnVec, normalize = TRUE, background = TRUE, bgversion = 2, destructive = FALSE, verbose = TRUE, ...)
```

**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**

```r
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]
```
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

- **x**: a FeatureSet-like object or ExpressionSet object.
- **which**: character defining what probe types are to be used in the plot.
- **transfo**: a function to transform the data before plotting. See 'Details'.
- **nsample**: number of units to sample and build the plot.
- **...**: 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
**Access 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.

---

**CRLMM Genotype Calls**

**Description**

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

**Usage**

```
crlmm(filenames, outdir, batch_size=40000, balance=1.5, 
       minLLRforCalls=c(5, 1, 5), recalibrate=TRUE, 
       verbose=TRUE, pkgname, reference=TRUE)
```

```
justCRLMM(filenames, batch_size = 40000, minLLRforCalls = c(5, 1, 5), 
          recalibrate = TRUE, balance = 1.5, phenoData = NULL, verbose = TRUE, 
          pkgname = NULL, tmpdir=tempdir())
```
darkColors

Arguments

filenames 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.

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 pallete in RColorBrewer, therefore useful to describe quantitative features ranging on two extremes.
**Examples**

```r
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)
```

**Description**

Fits robust Probe Level linear Models to all the (meta)probesets in an `FeatureSet`. This is carried out on a (meta)probeset by (meta)probeset basis.

**Usage**

```r
fitProbeLevelModel(object, background=TRUE, normalize=TRUE, target="core", method="plm", verbose=TRUE, S4=TRUE, ...)
```

**Arguments**

- `object` `FeatureSet` object.
- `background` Do background correction?
- `normalize` Do normalization?
- `target` character vector describing the summarization target. Valid values are: 'probeset', 'core' (Gene/Exon), 'full' (Exon), 'extended' (Exon).
- `method` summarization method to be used.
- `verbose` verbosity flag.
- `S4` return final value as an S4 object (oligoPLM) if TRUE. If FALSE, final value is returned as a list.
- `...` subset to be passed down to `getProbeInfo` for subsetting. See `subset` for details.

**Value**

`fitProbeLevelModel` returns an `oligoPLM` object, if `S4=TRUE`; otherwise, it will return a list.

**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.
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

Author(s)

This is a simplified port from Ben Bolstad’s work implemented in the affyPLM package. Problems with the implementation in oligo should be reported to Benilton Carvalho.

References


See Also

rma, summarizationMethods, subset

Examples

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

getAffinitySplineCoefficients

Estimate affinity coefficients.
**getBaseProfile**

*Compute and plot nucleotide profile.*

**Description**

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

**Usage**

```r
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**

```r
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**

```r
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**

```r
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)
getNgsColorsInfo

Author(s)
Benilton Carvalho

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

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, ‘sampleNames’, is returned if the filenames follow the sampleID_532.XYS / sampleID_635.XYS standard.

Author(s)
Benilton Carvalho <bcarvalh@jhsph.edu>
getPlatformDesign

Retrieve Platform Design object

Description
Retrieve platform design object.

Usage
getPlatformDesign(object)
getPD(object)

Arguments
object FeatureSet object

Details
Retrieve platform design object.

Value
platformDesign or PDInfo object.

getProbeInfo

Probe information selector.

Description
A tool to simplify the selection of probe information, so user does not need to use the SQL approaches.

Usage
getProbeInfo(object, field, probeType = "pm", target = "core", sortBy = c("fid", "man_fsetid", "none"), ...)

Arguments
object FeatureSet object.
field character string with names of field(s) of interest to be obtained from database.
probeType character string: ’pm’ or ’mm’
target Used only for Exon or Gene ST arrays: ’core’, ’full’, ’extended’, ’probeset’.
sortBy Field to be used for sorting.
... Arguments to be passed to subset

Value
A data.frame with the probe level information.
Note

The code allows for querying info on MM probes, however it has been used mostly on PM probes.

Author(s)

Benilton Carvalho

Examples

```r
if (require(oligoData)){
  data(affyGeneFS)
  availProbeInfo(affyGeneFS)
  probeInfo <- getProbeInfo(affyGeneFS, c('fid', 'x', 'y', 'chrom'))
  head(probeInfo)
  ## Selecting antigenomic background probes
  agenGene <- getProbeInfo(affyGeneFS, field=c('fid', 'fsetid', 'type'), target='probeset', subset= type == 'control->bgp->antigenomic')
  head(agenGene)
}
```

---

**getX**

Accessors for physical array coordinates.

**Description**

Accessors for physical array coordinates.

**Usage**

```r
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**

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

**Density estimate**

**Description**

Plot the density estimates for each sample

**Usage**

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

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

**Arguments**

- `x` FeatureSet or ExpressionSet object
- `transfo` a function to transform the data before plotting. See 'Details'.
- `nsample` number of units to sample and build the plot.
- `which` set of probes to be plotted ("pm", "mm", "bg", "both", "all").
- `...` arguments to be passed to `matplot`

**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

**Display a pseudo-image of a microarray chip**

**Description**

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

```r
## 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 `image`

Examples

```r
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)
  plot(1) ## while fixing fitPLM TODO
}
```

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**

```r
justSNPRMA(filenames, verbose = TRUE, phenoData = NULL, normalizeToHapmap = TRUE)
```
Arguments

filenames  character vector with the filenames.
verbose    logical flag for verbosity.
phenoData  a phenoData object or NULL
normalizeToHapmap

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(...)  ...
 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()
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.
MAplot

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**

```r
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)
}
```
Accessors and replacement methods for the intensity/PM/MM/BG matrices.

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

**Usage**

- `intensity(object)`
- `mm(object, subset = NULL, target='core')`
- `pm(object, subset = NULL, target='core')`
- `bg(object, subset = NULL)`
- `mm(object, subset = NULL, target='core')<-value`
- `pm(object, subset = NULL, target='core')<-value`
- `bg(object)<-value`

**Arguments**
- **object** FeatureSet object.
- **subset** Not implemented yet.
- **value** matrix object.
- **target** One of 'probeset', 'core', 'full', 'extended'. This is ignored if the array design is something other than Gene ST or Exon ST.

**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.

When applied to ExonFeatureSet or GeneFeatureSet objects, `pm` will return the PM matrix at the transcript level ('core' probes) by default. The user should set the `target` argument accordingly if something else is desired. The valid values are: 'probeset' (Exon and Gene arrays), 'core' (Exon and Gene arrays), 'full' (Exon arrays) and 'extended' (Exon arrays).

The `target` argument has no effects when used on designs other than Gene and Exon ST.

**Examples**

```r
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,]
}
```
**mmSequence**

**Description**

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

**Usage**

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

**Arguments**

- `object`: FeatureSet or DBPDInfo object
- `...`: Extra arguments, not yet implemented
Arguments

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

Value

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

Description

The functions or variables listed here are no longer part of `oligo`

Usage

fitPLM(...) coefs(...) resids(...)  
  Arguments.

Details

fitPLM was replaced by fitProbeLevelModel, allowing faster execution and providing more specific models. fitPLM was based in the code written by Ben Bolstad in the affyPLM package. However, all the model-fitting functions are now in the package preprocessCore, on which fitProbeLevelModel depends. coefs and resids, like fitPLM, were inherited from the affyPLM package. They were replaced respectively by coef and residuals, because this is how these statistics are called everywhere else in R.

Description

A class to represent Probe Level Models.

Objects from the Class

Objects can be created by calls of the form fitProbeLevelModel(FeatureSetObject), where FeatureSetObject is an object obtained through read.celfiles or read.xysfiles, representing intensities observed for different probes (which are grouped in probesets or meta-probesets) across distinct samples.
Slots

- chip.coefs: "matrix" with chip/sample effects - probeset-level
- probe.coefs: "numeric" vector with probe effects
- weights: "matrix" with weights - probe-level
- residuals: "matrix" with residuals - probe-level
- se.chip.coefs: "matrix" with standard errors for chip/sample coefficients
- se.probe.coefs: "numeric" vector with standard errors for probe effects
- residualSE: scale - residual standard error
- geometry: array geometry used for plots
- method: "character" string describing method used for PLM
- manufacturer: "character" string with manufacturer name
- annotation: "character" string with the name of the annotation package
- narrays: "integer" describing the number of arrays
- nprobes: "integer" describing the number of probes before summarization
- nprobesets: "integer" describing the number of probesets after summarization

Methods

- annotation signature(object = "oligoPLM"): accessor/replacement method to annotation slot
- boxplot signature(x = "oligoPLM"): boxplot method
- coef signature(object = "oligoPLM"): accessor/replacement method to coef slot
- coefs.probe signature(object = "oligoPLM"): accessor/replacement method to coefs.probe slot
- geometry signature(object = "oligoPLM"): accessor/replacement method to geometry slot
- image signature(x = "oligoPLM"): image method
- manufacturer signature(object = "oligoPLM"): accessor/replacement method to manufacturer slot
- method signature(object = "oligoPLM"): accessor/replacement method to method slot
- ncol signature(x = "oligoPLM"): accessor/replacement method to ncol slot
- nprobes signature(object = "oligoPLM"): accessor/replacement method to nprobes slot
- nprobesets signature(object = "oligoPLM"): accessor/replacement method to nprobesets slot
- residuals signature(object = "oligoPLM"): accessor/replacement method to residuals slot
- residualSE signature(object = "oligoPLM"): accessor/replacement method to residualSE slot
- se signature(object = "oligoPLM"): accessor/replacement method to se slot
- se.probe signature(object = "oligoPLM"): accessor/replacement method to se.probe slot
- show signature(object = "oligoPLM"): show method
- weights signature(object = "oligoPLM"): accessor/replacement method to weights slot
- NUSE signature(x = "oligoPLM"): Boxplot of Normalized Unscaled Standard Errors (NUSE) or NUSE values.
- RLE signature(x = "oligoPLM"): Relative Log Expression boxplot or values.
paCalls

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 the package’s maintainer.

References


See Also

rma, summarize

Examples

## TODO: review code and fix broken
## Not run:
if (require(oligoData)){
  data(nimbleExpressionFS)
  fit <- fitProbeLevelModel(nimbleExpressionFS)
  image(fit)
  NUSE(fit)
  RLE(fit)
}
## End(Not run)

---

**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: median(Ri) = tau, corresponding to absence of transcript
H1: median(Ri) > tau, corresponding to presence of transcript

where Ri = (PMi - MMi) / (PMi + MMi) 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


Examples

```r
## 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  Access to position information

Description

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

Usage

pmPosition(object)

pmOffset(object)

Arguments

object  AffySNPDBInfo, TilingFeatureSet or SnpCallSet object
**pmStrand**

**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.

<table>
<thead>
<tr>
<th>pmStrand</th>
<th>Accessor to the strand information</th>
</tr>
</thead>
</table>

**Description**

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

**Usage**

`pmStrand(object)`

**Arguments**

| object | AffySNPPDInfo or TilingFeatureSet object |

<table>
<thead>
<tr>
<th>probeNames</th>
<th>Accessor to feature names</th>
</tr>
</thead>
</table>

**Description**

Accessors to featureset names.

**Usage**

`probeNames(object, subset = NULL, ...)`

`probesetNames(object, ...)`

**Arguments**

| object | FeatureSet or DBPDInfo |
| subset | not implemented yet. |
| ... | Arguments (like 'target') passed to downstream methods. |

**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

...  
filenames  
channel1  
channel2  
pkgname  
phenoData  
featureData  
experimentData  
protocolData  
notes  
verbose  
sampleNames  
rm.mask  
rm.outliers  
rm.extra  
checkType

Details

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

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.
NimbleGen provides XYS files which are read by this function.

Usage

```r
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

Examples

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

Description

This function reads 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

Examples

```r
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)
}
```
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 'HTAFeatureSet'
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/HTA/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 = "HTAFeatureSet") When applied to a HTAFeatureSet 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


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

<table>
<thead>
<tr>
<th>runDate</th>
<th>Date of scan</th>
</tr>
</thead>
</table>

**Description**

Retrieves date information in CEL/XYS files.

**Usage**

```r
grunDate(object)
```

**Arguments**

- `object`: 'FeatureSet' object.

---

**sequenceDesignMatrix**

*Create design matrix for sequences*

**Description**

Creates design matrix for sequences.

**Usage**

```r
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**

```r
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)
```
Preprocessing SNP Arrays

Description

This function preprocess SNP arrays.

Usage

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

Arguments

- **object**: SnpFeatureSet object
- **verbose**: Verbosity flag. logical
- **normalizeToHapmap**: internal

Value

A SnpQSet object.

Tools for microarray preprocessing.

Description

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

Usage

backgroundCorrectionMethods()
normalizationMethods()
summarizationMethods()

backgroundCorrect(object, method=backgroundCorrectionMethods(), copy=TRUE, extra, subset=NULL, target=QuotescoreQuotescore, verbose=TRUE, ...)

## S4 method for signature 'FeatureSet'

summarize(object, probes=rownames(object), method="medianpolish", verbose=TRUE, ...)

## S4 method for signature 'FeatureSet'

normalize(object, method=normalizationMethods(), copy=TRUE, subset=NULL, target=QuotescoreQuotescore, verbose=TRUE, ...)

## S4 method for signature 'FeatureSet'

normalizeToTarget(object, targetDist, method="quantile", copy=TRUE, verbose=TRUE)
summarize

Arguments

- **object**: Object containing probe intensities to be preprocessed.
- **method**: String determining which method to use at that preprocessing step.
- **targetDist**: 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).
- **subset**: Not yet implemented.
- **target**: One of the following values: 'core', 'full', 'extended', 'probeset'. Used only with Gene ST and Exon ST designs.
- **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

```r
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)
summarizationMethods()
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=’Var

for (mtd in normalizationMethods()){  
  message(’Normalizing with ‘, mtd)

```

```
`res <- normalize(nimbleExpressionFS, method=mtd, verbose=FALSE)`
`boxplot(res, main=mtd)}`
Index

*Topic **IO**
  read.celfiles, 30
  read.xysfiles, 31
*Topic **classes**
  oligoPLM-class, 23
*Topic **classif**
  crlmm, 7
  getNetAffx, 12
  runDate, 35
*Topic **file**
  list.xysfiles, 18
*Topic **hplot**
  boxplot, 6
  darkColors, 8
  hist, 16
  image, 16
  MAplot, 19
*Topic **loess**
  MAplot, 19
*Topic **manip**
  basecontent, 3
  basicPLM, 4
  basicRMA, 5
  chromosome, 7
  fitProbeLevelModel, 9
  getAffinitySplineCoefficients, 10
  getBaseProfile, 11
  getContainer, 11
  getCrlmmSummaries, 12
  getNgsColorsInfo, 13
  getPlatformDesign, 14
  getProbeInfo, 14
  getX, 15
  justSNPRMA, 17
  mm, 21
  mmindex, 22
  mmSequence, 22
  oligo-defunct, 23
  paCalls, 25
  pmAllele, 27
  pmFragmentLength, 28
  pmPosition, 28
  pmStrand, 29
  probeNames, 29
  readSummaries, 32
  sequenceDesignMatrix, 35
  snprma, 36
  summarize, 36
*Topic **methods**
  boxplot, 6
  hist, 16
  MAplot, 19
  plotM-methods, 27
  rma-methods, 33
*Topic **package**
  oligo-package, 3
*Topic **smooth**
  MAplot, 19
  annotation, oligoPLM-method
    (oligoPLM-class), 23
  availProbeInfo (getProbeInfo), 14
  backgroundCorrect (summarize), 36
  backgroundCorrect, FeatureSet-method
    (summarize), 36
  backgroundCorrect, ff_matrix-method
    (summarize), 36
  backgroundCorrect, matrix-method
    (summarize), 36
  backgroundCorrect-methods (summarize), 36
  backgroundCorrectionMethods
    (summarize), 36
  basecontent, 3
  basicPLM, 4
  basicRMA, 5, 5
  bg (mm), 21
  bg, FeatureSet-method (mm), 21
  bg, TilingFeatureSet-method (mm), 21
  bg<-(mm), 21
  bg<-, FeatureSet, ff_matrix-method (mm),
    21
  bg<-, FeatureSet, matrix-method (mm), 21
  bg<-, TilingFeatureSet, array-method
    (mm), 21
  bgindex (mmindex), 22
INDEX

mm, FeatureSet-method (mm), 21
mm, TilingFeatureSet-method (mm), 21
mm<->, FeatureSet, ANY, ANY, ff_matrix-method (mm), 21
mm<-, FeatureSet, ANY, ANY, matrix-method (mm), 21
mm<-, TilingFeatureSet, ANY, ANY, array-method (mm), 21
mmindex, FeatureSet-method (mmindex), 22
mmindex, DBPDInfo-method (mmindex), 22
mmSequence, AffySNPPDInfo-method (mmSequence), 22
mmSequence, DBPDInfo-method (mmSequence), 22
mmSequence, FeatureSet-method (mmSequence), 22

ncol, oligoPLM-method (oligoPLM-class), 23
normalizationMethods (summarize), 36
normalize, FeatureSet-method (summarize), 36
normalize, ff_matrix-method (summarize), 36
normalize, matrix-method (summarize), 36
normalizeToTarget (summarize), 36
normalizeToTarget, ff_matrix-method (summarize), 36
normalizeToTarget, matrix-method (summarize), 36
normalizeToTarget-methods (summarize), 36
nprobes (oligoPLM-class), 23
nprobes, oligoPLM-method (oligoPLM-class), 23
nprobesets, oligoPLM-method (oligoPLM-class), 23
NUSE (oligoPLM-class), 23
NUSE, oligoPLM-method (oligoPLM-class), 23

oligo-defunct, 23
oligo-package, 3
oligoPLM, 9
oligoPLM (oligoPLM-class), 23
oligoPLM-class, 23

paCalls, 25
paCalls, ExonFeatureSet-method (paCalls), 25
paCalls, ExpressionFeatureSet-method (paCalls), 25
paCalls, GeneFeatureSet-method (paCalls), 25
plot, 20
plotM (plotM-methods), 27
plotM, SnpQSet, character-method (plotM-methods), 27
plotM, SnpQSet, integer-method (plotM-methods), 27
plotM, SnpQSet, numeric-method (plotM-methods), 27
plotM, TilingQSet, missing-method (plotM-methods), 27
plotM-methods, 27
pm (mm), 21
pm, FeatureSet-method (mm), 21
pm, GenericFeatureSet-method (mm), 21
pm, TilingFeatureSet-method (mm), 21
pm<-> (mm), 21
pm<-, FeatureSet, ANY, ANY, ff_matrix-method (mm), 21
pm<-, FeatureSet, ANY, ANY, matrix-method (mm), 21
pm<-, GenericFeatureSet, ANY, ANY, ff_matrix-method (mm), 21
pm<-, GenericFeatureSet, ANY, ANY, matrix-method (mm), 21
pm<-, TilingFeatureSet, ANY, ANY, array-method (mm), 21
pmAllele, 27
pmAllele, AffySNPPDInfo-method (pmAllele), 27
pmAllele, SnpFeatureSet-method (pmAllele), 27
pmChr (chromosome), 7
pmChr, ExonFeatureSet-method (chromosome), 7
pmChr, FeatureSet-method (chromosome), 7
pmChr, GeneFeatureSet-method (chromosome), 7
pmFragmentLength, 28
pmFragmentLength, AffySNPPDInfo-method (pmFragmentLength), 28
pmFragmentLength, SnpFeatureSet-method (pmFragmentLength), 28
pmindex (mmindex), 22
pmindex, DBPDInfo-method (mmindex), 22
pmindex, FeatureSet-method (mmindex), 22
pmindex, GenericFeatureSet-method
rcModelPLMr, 5
rcModelPLMrc, 5
rcModelPLMrr, 5
read.celfiles, 30, 32
read.celfiles2 (read.celfiles), 30
read.xysfiles, 31
read.xysfiles2 (read.xysfiles), 31
readSummaries, 32
resids (oligo-defunct), 23
residuals, oligoPLM-method
(oligoPLM-class), 23
residualSE (oligoPLM-class), 23
residualSE, oligoPLM-method
(oligoPLM-class), 23
RLE (oligoPLM-class), 23
RLE, oligoPLM-method (oligoPLM-class), 23
rma, 10, 25
rma (rma-methods), 33
rma, ExonFeatureSet-method
(rma-methods), 33
rma, ExpressionFeatureSet-method
(rma-methods), 33
rma, GeneFeatureSet-method
(rma-methods), 33
rma, GenericFeatureSet-method
(rma-methods), 33
rma, HTAFeatureSet-method (rma-methods), 33
rma, SnpCnvFeatureSet-method
(rma-methods), 33
rma-methods, 33
runDate, 35
runDate, FeatureSet-method (runDate), 35
runDate-methods (runDate), 35
sample, 6
se (oligoPLM-class), 23
se, oligoPLM-method (oligoPLM-class), 23
se.probe (oligoPLM-class), 23
se.probe, oligoPLM-method
(oligoPLM-class), 23
seqColors (darkColors), 8
seqColors2 (darkColors), 8
sequenceDesignMatrix, 35
set.seed, 6
show, oligoPLM-method (oligoPLM-class), 23
smoothScatter, 20
snprma, 34, 36
subset, 9, 14
summarizationMethods, 10
summarizationMethods (summarize), 36
summarize, 25, 36

rcModelPLM, 5
pmindex, 22
pmindex, GenericPDInfo-method (mmindex), 22
pmindex, stArrayDBPDInfo-method
(mmindex), 22
pmOffset (pmPosition), 28
pmOffset, AffySNPPDInfo-method
(pmPosition), 28
pmPosition, 28
pmPosition, ExpressionPDInfo-method
(pmPosition), 28
pmPosition, FeatureSet-method
(pmPosition), 28
pmPosition, TilingFeatureSet-method
(pmPosition), 28
pmPosition, TilingPDInfo-method
(pmPosition), 28
pmSequence (mmSequence), 22
pmSequence, AffyGenePDInfo-method
(mmSequence), 22
pmSequence, AffySNPPDInfo-method
(mmSequence), 22
pmSequence, DBPDInfo-method
(mmSequence), 22
pmSequence, ExonFeatureSet-method
(mmSequence), 22
pmSequence, FeatureSet-method
(mmSequence), 22
pmSequence, GeneFeatureSet-method
(mmSequence), 22
pmSequence, stArrayDBPDInfo-method
(mmSequence), 22
pmStrand, 29
pmStrand, AffySNPPDInfo-method
(pmStrand), 29
pmStrand, TilingFeatureSet-method
(pmStrand), 29
probeNames, 29
probeNames, DBPDInfo-method
(probeNames), 29
probeNames, ExonFeatureSet-method
(probeNames), 29
probeNames, FeatureSet-method
(probeNames), 29
probeNames, GeneFeatureSet-method
(probeNames), 29
probeNames, stArrayDBPDInfo-method
(probeNames), 29
probeSetNames (probeNames), 29
probeSetNames, FeatureSet-method
(probeNames), 29
summarize,ff_matrix-method (summarize), 36
summarize, matrix-method (summarize), 36
summarize-methods (summarize), 36

weights, oligoPLM-method (oligoPLM-class), 23