MAplot

Relative M vs. A plots

Description

Create boxplots of M or M vs A plots. Where M is determined relative to a specified chip or to a pseudo-median reference chip.

Arguments

... Additional parameters for the routine
A A vector to plot along the horizontal axis
M A vector to plot along vertical axis
subset A set of indices to use when drawing the loess curve
show.statistics If true some summary statistics of the M values are drawn
span span to be used for loess fit.
family.loess "guassian" or "symmetric" as in loess.
cex Size of text when writing summary statistics on plot

See Also

mva.pairs
**PLMset-class**

**Class** PLMset

**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.
- **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.
- **description**: 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.
- **notes**: Object of class "character" Vector of explanatory text.
- **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).
Methods

weights<- signature(object = "PLMset") : replaces the weights.
weights signature(object = "PLMset") : extracts the model fit weights.
coefs<- signature(object = "PLMset") : replaces the chip coefs.
coefs signature(object = "PLMset") : extracts the chip coefs.
se signature(object = "PLMset") : extracts the standard error estimates of the chip coefs.
se<- signature(object = "PLMset") : replaces the standard error estimates of the chip coefs.
coefs.probe signature(object = "PLMset") : extracts the probe coefs.
se.probe signature(object = "PLMset") : extracts the standard error estimates of the probe coefs.
coefs.const signature(object = "PLMset") : extracts the intercept coefs.
se.const signature(object = "PLMset") : extracts the standard error estimates of the intercept coefs.
getCdfInfo signature(object = "PLMset") : retrieve the environment that defines the location of probes by probe set.
image signature(x = "PLMset") : creates an image of the robust linear model fit weights for each sample.
indexProbes signature(object = "PLMset", which = "character") : returns a list with locations of the probes in each probe set. The list names defines the probe set names. which can be "pm", "mm", or "both". If "both" then perfect match locations are given followed by mismatch locations.
Mbox signature(object = "PLMset") : gives a boxplot of M's for each chip. The M’s are computed relative to a "median" chip.
normvec signature(x = "PLMset") : will return the normalization vector (if it has been stored).
residSE signature(x = "PLMset") : will return the residual SE (if it has been stored).
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.

Note
This class is better described in the vignette.

Author(s)
B. M. Bolstad (bmb@bmbolstad.com)

References
PLMset2exprSet  

*Convert a PLMset to an ExpressionSet*

**Description**

This function converts a PLMset to an ExpressionSet. This is often useful since many Bioconductor functions operate on ExpressionSet objects.

**Usage**

```r
PLMset2exprSet(pset)
pset2eset(pset)
```

**Arguments**

- `pset`  
The PLMset to convert to ExpressionSet.

**Details**

These functions convert PLMset objects to ExpressionSet objects. This is often useful since many Bioconductor functions operate on ExpressionSet objects. Note that the function `pset2eset` is a wrapper for `PLMset2exprSet`.

**Value**

returns a ExpressionSet

**Author(s)**

Ben Bolstad (bmb@bmbolstad.com)

**See Also**

ExpressionSet

**Examples**

```r
if (require(affydata)) {
  data(Dilution)
  Pset <- fitPLM(Dilution)
  eset <- pset2eset(Pset)
}
```
ReadRMAExpress

Read RMAExpress computed expression values

Description

Read RMAExpress computed binary output files into a matrix or ExpressionSet

Usage

ReadRMAExpress(filename, return.value=c("ExpressionSet","matrix"))

Arguments

filename The name of the file containing RMAExpress output to be read in
return.value should a matrix or an ExpressionSet be returned

Value

returns an ExpressionSet

Author(s)

Ben Bolstad (bmb@bmbolstad.com)

References

http://rmaexpress.bmbolstad.com

bg.correct.LESN

LESN - Low End Signal is Noise Background corrections

Description

This function background corrects PM probe data using LESN - Low End Signal is Noise concepts.

Usage

bg.correct.LESN(object, method = 2, baseline = 0.25, theta=4)

Arguments

object an AffyBatch
method an integer code specifying which method to use
baseline A baseline value to use
theta A parameter used in the background correction process
Details

This method will be more formally documented at a later date.

The basic concept is to consider that the lowest end of intensities is most likely just noise (and should be heavily corrected) and the highest end signals are most likely signal and should have little adjustment. Low end signals are made much smaller while high end signals get less adjustment relative adjustment.

Value

An AffyBatch

Author(s)

Ben Bolstad (bmb@bmbolstad.com)

References


Examples

```r
if (require(affydata)) {
  data(Dilution)
  Dilution.example.bgcorrect <- bg.correct.LESN(Dilution)
}
```

---

**fitPLM**

*Fit a Probe Level Model to Affymetrix Genechip Data.*

Description

This function converts an AffyBatch into an PLMset by fitting a specified robust linear model to the probe level data.

Usage

```r
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 = 
verify.output.param(), model.param = 
verify.model.param(object, model), 
verbosity.level=0)
```
Arguments

object an AffyBatch
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 AffyBatch. 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.

Value

An PLMset

Author(s)

Ben Bolstad (bmb@bmbolstad.com)

References


See Also

expresso, rma, threestep
Examples

```r
if (require(affydata)) {
  data(Dilution)
  Pset <- fitPLM(Dilution, model=PM ~ -1 + probes + samples)
  se(Pset)[1:5,]
  image(Pset)
  NUSE(Pset)

  # now lets try a wider class of models
  ## Not run: Pset <- fitPLM(Dilution, model=PM ~ -1 + probes + liver, normalize=FALSE, background=FALSE)
  ## Not run: coefs(Pset)[1:10,]

  ## Not run: Pset <- fitPLM(Dilution, model=PM ~ -1 + probes + liver + scanner, normalize=FALSE, background=FALSE)
  coefs(Pset)[1:10,]

  # try liver as a covariate
  logliver <- log2(c(20,20,10,10))
  ## Not run: Pset <- fitPLM(Dilution, model=PM ~ -1 + probes + logliver + scanner, normalize=FALSE, background=FALSE)
  coefs(Pset)[1:10,]

  # try a different se.type
  ## Not run: Pset <- fitPLM(Dilution, model=PM ~ -1 + probes + scanner, normalize=FALSE, background=FALSE)
  se(Pset)[1:10,]
}
```

### normalize.ExpressionSet

**Normalization applied to ExpressionSets**

#### Description

Allows the user to apply normalization routines to ExpressionSets.

#### Usage

```r
normalize.ExpressionSet.quantiles(eset, transfn=c("none","log","antilog"))
normalize.ExpressionSet.loess(eset, transfn=c("none","log","antilog"),...)
normalize.ExpressionSet.contrasts(eset, span = 2/3, choose.subset = TRUE, subset.size = 5000, verbose = TRUE, family = "symmetric", transfn=c("none","log","antilog"),...)
normalize.ExpressionSet.qspline(eset, transfn=c("none","log","antilog"),...)
normalize.ExpressionSet.invariantset(eset, prd.td = c(0.003, 0.007), verbose=FALSE, baselinetype=c("mean","median","pseudo-mean","pseudo-median"),...)
normalize.ExpressionSet.scaling(eset, trim=0.02, baseline=-1, transfn=c("none","log"),...)
```

#### Arguments

- `eset`: An `ExpressionSet`
- `span`: Parameter to be passed to the function `loess`.
- `choose.subset`: Parameter to be passed to the function `loess`.
- `subset.size`: Parameter to be passed to the function `loess`.
- `verbose`: Verbosity flag.
normalize.ExpressionSet

family parameter to be passed to the function loess.

prd.td cutoff parameter (details in the bibliographic reference)

trim How much to trim from the top and bottom before computing the mean when using the scaling normalization

baseline Index of array to use as baseline, negative values (-1,-2,-3,-4) control different baseline selection methods

transfn Transform the ExpressionSet before normalizing. Useful when dealing with expression values that are log-scale

baseline.type A method of selecting the baseline array

... Additional parameters that may be passed to the normalization routine

Details

This function carries out normalization of expression values. In general you should either normalize at the probe level or at the expression value level, not both.

Typing normalize.ExpressionSet.methods should give you a list of methods that you may use. Note that you can also use the normalize function on ExpressionSets. Use method to select the normalization method.

Value

A normalized ExpressionSet.

Author(s)

Ben Bolstad, ⟨bmb@bmbolstad.com⟩

References


See Also

normalize

Examples

if (require(affydata)) {
  data(Dilution)
  eset <- rma(Dilution, normalize=FALSE, background=FALSE)
  normalize(eset)
}
normalize.quantiles.probeset

Quantile Normalization applied to probesets

Description
Using a normalization based upon quantiles, this function normalizes a matrix of probe level intensities.

Usage

normalize.AffyBatch.quantiles.probeset(abatch,type=c("separate","pmonly","mmonly"),use.median=FALSE,use.log=TRUE)

Arguments

abatch
An AffyBatch
type
how should MM and PM values be handled
use.median
use median rather than mean
use.log
take logarithms, then normalize

Details
This function applies the quantile method in a probeset specific manner.

In particular a probeset summary is normalized using the quantile method and then the probes adjusted accordingly.

Value

A normalized AffyBatch.

Author(s)

Ben Bolstad, ⟨bmb@bmbolstad.com⟩

References


See Also

normalize.normalize.quantiles
normalize.scaling  Scaling normalization

Description

Allows the user to apply scaling normalization.

Usage

normalize.scaling(X, trim=0.02, baseline=-1, log.scalefactors=FALSE)
normalize.AffyBatch.scaling(abatch, type=c("together","pmonly","mmonly","separate"), trim=0.02, baseline=-1, log.scalefactors=FALSE)

Arguments

X  A matrix. The columns of which are to be normalized.

abatch  An AffyBatch

type  A parameter controlling how normalization is applied to the Affybatch.

trim  How much to trim from the top and bottom before computing the mean when using the scaling normalization.

baseline  Index of array to use as baseline, negative values (-1,-2,-3,-4) control different baseline selection methods.

log.scalefactors  Compute the scale factors based on log2 transformed data.

Details

These function carries out scaling normalization of expression values.

Value

A normalized ExpressionSet.

Author(s)

Ben Bolstad, ⟨bmb@bmbolstad.com⟩

See Also

normalize

Examples

if (require(affydata)) {
  data(Dilution)
  normalize.AffyBatch.scaling(Dilution)
}
Description

This function pre-processes an `AffyBatch`.

Usage

```r
preprocess(object, subset=NULL, normalize=TRUE, background=TRUE, background.method="RMA.2", normalize.method="quantile", background.param=list(), normalize.param=list(), verbosity.level=0)
```

Arguments

- `object`: an `AffyBatch`
- `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`: list of parameters for background correction methods
- `normalize.param`: list of parameters for normalization methods
- `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 carries out background correction and normalization pre-processing steps. It does not summarize to produce gene expression measures. All the same pre-processing methods supplied by `threestep` are supported by this function.

Value

An `AffyBatch`

Author(s)

Ben Bolstad (bmb@bmbolstad.com)

References

pseudo.coloring

See Also

expresso, rma

Examples

```r
if (require(affydata)) {
  data(Dilution)
  # should be equivalent to the bg and norm of rma()
  abatch.preprocessed <- preprocess(Dilution)
}
```

pseudo.coloring  Coloring pseudo chip images

Description

These are routines used for coloring pseudo chip images.

Usage

```r
pseudoPalette(low = "white", high = c("green", "red"), mid = NULL, k = 50)
pseudoColorBar(x, horizontal = TRUE, col = heat.colors(50), scale = 1:length(x),

Arguments

- **low**: color at low end of scale
- **high**: color at high end of scale
- **mid**: color at exact middle of scale
- **k**: number of colors to have
- **x**: A data series
- **horizontal**: If TRUE then color bar is to be draw horizontally
- **col**: colors for color bar
- **scale**: tickmarks for x if x is not numeric
- **log.ticks**: use a log type transformation to assign the colors
- **...**: additional parameters to plotting routine

Details

Adapted from similar tools in maPlots package.

Author(s)

Ben Bolstad (bmb@bmbolstad.com)

See Also

AffyBatch, read.affybatch
rmaPLM

Fit a RMA to Affymetrix Genechip Data as a PLMset

Description

This function converts an `AffyBatch` into an `PLMset` by fitting a multichip model. In particular we concentrate on the RMA model.

Usage

```r
rmaPLM(object, subset=NULL, normalize=TRUE, background=TRUE, background.method="RMA.2", normalize.method="quantile", background.param = list(), normalize.param=list(), output.param=list(), model.param=list(), verbosity.level=0)
```

Arguments

- `object` an `AffyBatch`
- `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 the RMA as a Probe Level Linear models to all the probesets in an `AffyBatch`.

Value

An `PLMset`

Author(s)

Ben Bolstad (bmb@bmbolstad.com)
threestep

References


See Also

expresso, rma, threestep.fitPLM, threestepPLM

Examples

if (require(affydata)) {
  # A larger example testing weight image function
  data(Dilution)
  ## Not run: Pset <- rmaPLM(Dilution, output.param=list(weights=TRUE))
  ## Not run: image(Pset)
}

threestep

Three Step expression measures

Description

This function converts an AffyBatch into an ExpressionSet using a three step expression measure.

Usage

threestep(object, subset=NULL, normalize=TRUE, background=TRUE, background.method="RMA.2", normalize.method="quantile", summary.method="median.polish", background.param = list(), normalize.param=list(), summary.param=list(), verbosity.level=0)

Arguments

object an AffyBatch.
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.
summary.method name of summary method to use.
background.param
list of parameters for background correction methods.
normalize.param
list of parameters for normalization methods.
summary.param
list of parameters for summary methods.
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 computes the expression measure using threestep methods. Greater details can be found in a vignette.

Value
An ExpressionSet

Author(s)
Ben Bolstad (bmb@bmbolstad.com)

References

See Also
despresso, rma

Examples
if (require(affydata)) {
  data(Dilution)

  # should be equivalent to rma()
  eset <- threestep(Dilution)

  # Using Tukey Biweight summarization
  eset <- threestep(Dilution, summary.method="tukey.biweight")

  # Using Average Log2 summarization
  eset <- threestep(Dilution, summary.method="average.log")

  # Using IdealMismatch background and Tukey Biweight and no normalization.
  eset <- threestep(Dilution, normalize=FALSE, background.method="IdealMM", summary.method="tukey.biweight")

  # Using average.log summarization and no background or normalization.
  eset <- threestep(Dilution, background=FALSE, normalize=FALSE,
                   background.method="IdealMM", summary.method="tukey.biweight")

  # Use threestep methodology with the rlm model fit
threestepPLM

```r
eset <- threestep(Dilution, summary.method="rlm")
# Use threestep methodology with the log of the average
eset <- threestep(Dilution, summary.method="log.average")
# Use threestep methodology with log 2nd largest method
eset <- threestep(Dilution, summary.method="log.2nd.largest")
eset <- threestep(Dilution, background.method="LESN2")
```

---

**threestepPLM**

*Three Step expression measures returned as a PLMset*

**Description**

This function converts an *AffyBatch* into an *PLMset* using a three step expression measure.

**Usage**

```r
threestepPLM(object, subset=NULL, normalize=TRUE, background=TRUE, background.method="LESN2")
```

**Arguments**

- `object` - an *AffyBatch*
- `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.
- `summary.method` - name of summary method to use.
- `background.param` - list of parameters for background correction methods
- `normalize.param` - list of parameters for normalization methods
- `output.param` - list of parameters for output methods
- `model.param` - list of parameters for model methods
- `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 computes the expression measure using threestep methods. It returns a *PLMset*. The most important difference is that the PLMset allows you to access the residuals which the *threestep* function does not do.
Value

An `PLMset`

Author(s)

Ben Bolstad (bmb@bmbolstad.com)

References


See Also

dexpreso, rma, threestep, rmaPLM, fitPLM

Examples

```r
if (require(affydata)) {
  data(Dilution)
  # should be equivalent to rma()
  ## Not run: eset <- threestepPLM(Dilution)
}
```
### Index

<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>classes</strong></td>
<td>PLMset-class, 2</td>
</tr>
<tr>
<td><strong>hplot</strong></td>
<td>MAplot, 1</td>
</tr>
<tr>
<td><strong>manip</strong></td>
<td>bg.correct.LESN, 5</td>
</tr>
<tr>
<td></td>
<td>fitPLM, 6</td>
</tr>
<tr>
<td></td>
<td>normalize.ExpressionSet, 8</td>
</tr>
<tr>
<td></td>
<td>normalize.quantiles.probeset, 10</td>
</tr>
<tr>
<td></td>
<td>normalize.scaling, 11</td>
</tr>
<tr>
<td></td>
<td>PLMset2exprSet, 4</td>
</tr>
<tr>
<td></td>
<td>preprocess, 12</td>
</tr>
<tr>
<td></td>
<td>pseudo.coloring, 13</td>
</tr>
<tr>
<td></td>
<td>ReadRMAExpress, 5</td>
</tr>
<tr>
<td></td>
<td>rmaPLM, 14</td>
</tr>
<tr>
<td></td>
<td>threestep, 15</td>
</tr>
<tr>
<td></td>
<td>threestepPLM, 17</td>
</tr>
<tr>
<td>AffyBatch</td>
<td>5–7, 10–15, 17</td>
</tr>
<tr>
<td>annotation, PLMset-method</td>
<td>(PLMset-class), 2</td>
</tr>
<tr>
<td>bg.correct.LESN, 5</td>
<td></td>
</tr>
<tr>
<td>boxplot, PLMset-method</td>
<td>(PLMset-class), 2</td>
</tr>
<tr>
<td>cdfName, PLMset-method</td>
<td>(PLMset-class), 2</td>
</tr>
<tr>
<td>coefs, PLMset-method</td>
<td>(PLMset-class), 2</td>
</tr>
<tr>
<td>coefs.const, PLMset-method</td>
<td>(PLMset-class), 2</td>
</tr>
<tr>
<td>coefs.probe, PLMset-method</td>
<td>(PLMset-class), 2</td>
</tr>
<tr>
<td>coefs&lt;- (PLMset-class), 2</td>
<td></td>
</tr>
<tr>
<td>description, PLMset-method</td>
<td>(PLMset-class), 2</td>
</tr>
<tr>
<td>ExpressionSet</td>
<td>4, 5, 8, 9, 11, 15, 16</td>
</tr>
<tr>
<td>espresso</td>
<td>7, 13, 15, 16, 18</td>
</tr>
<tr>
<td>fitPLM</td>
<td>2, 6, 15, 18</td>
</tr>
<tr>
<td>getCdfInfo, PLMset-method</td>
<td>(PLMset-class), 2</td>
</tr>
<tr>
<td>image, PLMset-method</td>
<td>(PLMset-class), 2</td>
</tr>
<tr>
<td>indexProbes, PLMset, character-method</td>
<td>(PLMset-class), 2</td>
</tr>
<tr>
<td>indexProbesProcessed, PLMset-method</td>
<td>(PLMset-class), 2</td>
</tr>
<tr>
<td>loess</td>
<td>1, 8, 9</td>
</tr>
<tr>
<td>MAplot</td>
<td>1</td>
</tr>
<tr>
<td>MAplot, PLMset-method (MAplot)</td>
<td>1</td>
</tr>
<tr>
<td>matrix</td>
<td>5</td>
</tr>
<tr>
<td>Mbox (PLMset-class)</td>
<td>2</td>
</tr>
<tr>
<td>Mbox, PLMset-method</td>
<td>(PLMset-class), 2</td>
</tr>
<tr>
<td>model.description (PLMset-class)</td>
<td>2</td>
</tr>
<tr>
<td>mva.pairs</td>
<td>1</td>
</tr>
<tr>
<td>normalize, 9–11</td>
<td></td>
</tr>
<tr>
<td>normalize.AffyBatch.quantiles.probeset</td>
<td>(normalize.quantiles.probeset), 10</td>
</tr>
<tr>
<td>normalize.AffyBatch.scaling</td>
<td>(normalize.scaling), 11</td>
</tr>
<tr>
<td>normalize.ExpressionSet</td>
<td>8</td>
</tr>
<tr>
<td>normalize.quantiles</td>
<td>10</td>
</tr>
<tr>
<td>normalize.quantiles.probeset</td>
<td>10</td>
</tr>
<tr>
<td>normalize.scaling</td>
<td>11</td>
</tr>
<tr>
<td>normvec (PLMset-class)</td>
<td>2</td>
</tr>
<tr>
<td>normvec, PLMset-method</td>
<td>(PLMset-class), 2</td>
</tr>
<tr>
<td>NUSE (PLMset-class)</td>
<td>2</td>
</tr>
</tbody>
</table>
nuse(PLMset-class), 2
NUSE, PLMset-method
  (PLMset-class), 2
nuse, PLMset-method
  (PLMset-class), 2
pData, PLMset-method
  (PLMset-class), 2
pData<-, PLMset, data.frame-method
  (PLMset-class), 2
phenoData, PLMset-method
  (PLMset-class), 2
phenoData<-, PLMset, AnnotatedDataFrame-method
  (PLMset-class), 2
PLMset, 4, 6, 7, 14, 17, 18
PLMset-class, 2
PLMset2exprSet, 4
preprocess, 12
pset2eset(PLMset2exprSet), 4
pseudo.coloring, 13
pseudoColorBar(pseudo.coloring), 13
pseudoPalette(pseudo.coloring), 13
quantile, 10
read.affybatch, 13
readRMAExpress, 5
resid, PLMset-method
  (PLMset-class), 2
resid<-(PLMset-class), 2
resid<-, PLMset-method
  (PLMset-class), 2
residSE(PLMset-class), 2
residSE, PLMset-method
  (PLMset-class), 2
residuals, PLMset-method
  (PLMset-class), 2
residuals<-(PLMset-class), 2
residuals<-, PLMset-method
  (PLMset-class), 2
RLE(PLMset-class), 2
RLE, PLMset-method(PLMset-class), 2
rma, 7, 13, 15, 16, 18
rmaPLM, 14, 18
sampleNames, PLMset-method
  (PLMset-class), 2
sampleNames<-(PLMset-class), 2
sampleNames<-, PLMset, character-method
  (PLMset-class), 2
se(PLMset-class), 2
se, PLMset-method(PLMset-class), 2
se.const(PLMset-class), 2
se.const, PLMset-method
  (PLMset-class), 2
se.probe(PLMset-class), 2
se.probe, PLMset-method
  (PLMset-class), 2
se<-(PLMset-class), 2
se<-, PLMset-method
  (PLMset-class), 2
show, PLMset-method
summary(PLMset-method
  (PLMset-class), 2
threestep, 7, 12, 15, 15, 17, 18
threestepPLM, 15, 17
varcov(PLMset-class), 2
varcov, PLMset-method
  (PLMset-class), 2
weights, PLMset-method
  (PLMset-class), 2
weights<-(PLMset-class), 2
weights<-, PLMset-method
  (PLMset-class), 2