Package ‘affyPLM’

April 25, 2017

Version 1.52.0
Title Methods for fitting probe-level models
Author Ben Bolstad <bmb@bmbolstad.com>
Maintainer Ben Bolstad <bmb@bmbolstad.com>
Depends R (>= 2.6.0), BiocGenerics (>= 0.3.2), affy (>= 1.11.0), Biobase (>= 2.17.8), gcrma, stats, preprocessCore (>= 1.5.1)
Imports zlibbioc, graphics, grDevices, methods
Suggests affydata, MASS
LinkingTo preprocessCore
Description A package that extends and improves the functionality of the base affy package. Routines that make heavy use of compiled code for speed. Central focus is on implementation of methods for fitting probe-level models and tools using these models. PLM based quality assessment tools.
License GPL (>= 2)
URL https://github.com/bmbolstad/affyPLM
biocViews Microarray, OneChannel, Preprocessing, QualityControl
LazyLoad yes
NeedsCompilation yes

R topics documented:

bg.correct.LESN ................................................. 2
fitPLM .......................................................... 3
MAplot .......................................................... 5
normalize.ExpressionSet ........................................ 5
normalize.quantiles.probeset .................................. 7
normalize.scaling ............................................. 8
PLMset-class .................................................. 9
PLMset2exprSet ................................................ 11
preprocess ..................................................... 12
pseudo.coloring .............................................. 13
ReadRMAExpress .............................................. 14
rmaPLM ......................................................... 14
threestep ....................................................... 16
threestepPLM ................................................ 18
**bg.correct.LESN**

**Index**

| 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**

```r
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 intensites 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 a 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 = 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

dexpeso, 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)
  ## End(Not run)
  ## Not run: coefs(Pset)[1:10,]
  ## End(Not run)

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

  # 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, variable.type=c(logliver="covariate"))
  ## End(Not run)
  coefs(Pset)[1:10,]

  # try a different se.type
  ## Not run: Pset <- fitPLM(Dilution, model=PM~-1+probes+scanner, normalize=FALSE, background=FALSE, model.param=list(se.type=2))
  ## End(Not run)
  se(Pset)[1:10,]
}
```
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

normalize.ExpressionSet

Normalization applied to ExpressionSets

Description

Allows the user to apply normalization routines to ExpressionSets.

Usage

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, transfn=c("none","log","antilog"),
baseline.type=c("mean","median","pseudo-mean","pseudo-median"))
normalize.ExpressionSet.scaling(eset, trim=0.02, baseline=-1,
transfn=c("none","log","antilog"))
Arguments

eset  An ExpressionSet
span  parameter to be passed to the function loess.
choose.subset use a subset of values to establish the normalization relationship
subset.size number to use for subset
verbose verbosity flag
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


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**

```r
normalize.AffyBatch.quantiles.probeset(abatch, type=c("separate","pmonly","mmonly","together"), 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.quantiles`
**normalize.scaling**  

*Scaling normalization*

**Description**

Allows the user to apply scaling normalization.

**Usage**

```r
normalize.scaling(X, trim=0.02, baseline=-1, log.scalefactors=FALSE)
```

```r
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>

**Examples**

```r
if (require(affydata)) {
    data(Dilution)
    normalize.AffyBatch.scaling(Dilution)
}
```
**PLMset-class**

**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.

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"
**Methods**

weights<-

signature(object = "PLMset"): replaces the weights.

weights

signature(object = "PLMset"): extracts the model fit weights.

coeffs<-

signature(object = "PLMset"): replaces the chip coefs.

coeffs

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.

coeffs.probe

signature(object = "PLMset"): extracts the probe coefs.

se.probe

signature(object = "PLMset"): extracts the standard error estimates of the probe coefs.

coeffs.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)
}
```
preprocess

**Background correct and Normalize**

**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

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

pseudoPalette(low = "white", high = c("green", "red"), mid = NULL, k = 50)
pseudoColorBar(x, horizontal = TRUE, col = heat.colors(50), scale = 1:length(x), k = 11, log.ticks = FALSE, ...)

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
ReadRMAExpress  
*Read RMAExpress computed expression values*

**Description**

Read RMAExpress computed binary output files into a matrix or ExpressionSet

**Usage**

```r
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

---

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>

References


See Also

expresso, rma, threestep, fitPLM, threestepPLM
Examples

```r
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**

```r
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

expresso, rma

Examples

```r
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
  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")
}
```
Description

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

Usage

threestepPLM(object, subset=NULL, normalize=TRUE, background=TRUE,
background.method="RMA.2", normalize.method="quantile",
summary.method="median.polish", 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.
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 ver-
bose. 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
expresso, rma, threestep, rmaPLM, fitPLM

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

  # should be equivalent to rma()
  ## Not run: eset <- threestepPLM(Dilution)
}
Index

*Topic classes
  PLMset-class, 9

*Topic hplot
  MAplot, 5

*Topic manip
  bg.correct.LESN, 2
  fitPLM, 3
  normalize.ExpressionSet, 5
  normalize.quantiles.probeset, 7
  normalize.scaling, 8
  PLMset2exprSet, 11
  preprocess, 12
  pseudo.coloring, 13
  ReadRMAExpress, 14
  rmaPLM, 14
  threestep, 16
  threestepPLM, 18

AffyBatch, 2–4, 7, 8, 12–16, 18
annotation, PLMset-method (PLMset-class), 9

cdfName, PLMset-method (PLMset-class), 9
coefs, PLMset-method (PLMset-class), 9
coefs, PLMset-method (PLMset-class), 9
coefs.const (PLMset-class), 9
coefs.const, PLMset-method (PLMset-class), 9
coefs.probe (PLMset-class), 9
coefs.probe, PLMset-method (PLMset-class), 9
coefs<-, PLMset-method (PLMset-class), 9
coefs<-, PLMset-method (PLMset-class), 9
description, PLMset-method (PLMset-class), 9

ExpressionSet, 6, 8, 11, 14, 16, 17
expression, 4, 13, 15, 17, 19

fitPLM, 3, 9, 15, 19

getCdfInfo, PLMset-method (PLMset-class), 9
image, PLMset-method (PLMset-class), 9
indexProbes, PLMset.character-method (PLMset-class), 9
indexProbesProcessed, PLMset-method (PLMset-class), 9
indexProbesProcessed, PLMset-method (PLMset-class), 9
loess, 5, 6

MAplot, 5
MAplot, PLMset-method (MAplot), 5
matrix, 14
Mbox (PLMset-class), 9
Mbox, PLMset-method (PLMset-class), 9
model.description, PLMset-method (PLMset-class), 9
model.description, PLMset-method (PLMset-class), 9
mva.pairs, 5

normalize.AffyBatch.quantiles.probeset
  (normalize.quantiles.probeset), 7
normalize.AffyBatch.scaling
  (normalize.scaling), 8
normalize.ExpressionSet, 5
normalize.quantiles, 7
normalize.quantiles.probeset, 7
normalize.scaling, 8
normvec (PLMset-class), 9
normvec, PLMset-method (PLMset-class), 9
NUSE (PLMset-class), 9
nuse (PLMset-class), 9
NUSE, PLMset-method (PLMset-class), 9
nuse, PLMset-method (PLMset-class), 9

pData, PLMset-method (PLMset-class), 9
pData<-, PLMset.data.frame-method (PLMset-class), 9
phenoData, PLMset-method (PLMset-class), 9

phenoData<-, PLMset.AnnotatedDataFrame-method
  (PLMset-class), 9

20
PLMset, 3, 4, 11, 14, 15, 18
PLMset (PLMset-class), 9
PLMset-class, 9
PLMset2exprSet, 11
preprocess, 12
pset2eset (PLMset2exprSet), 11
pseudo.coloring, 13
pseudoColorBar (pseudo.coloring), 13
pseudoPalette (pseudo.coloring), 13
quantile, 7
read.affybatch, 13
ReadRMAExpress, 14
resid (PLMset-class), 9
resid,PLMset-method (PLMset-class), 9
resid<-(PLMset-class), 9
residSE (PLMset-class), 9
residSE,PLMset-method (PLMset-class), 9
residuals (PLMset-class), 9
residuals,PLMset-method (PLMset-class), 9
residuals<-(PLMset-class), 9
RLE (PLMset-class), 9
RLE,PLMset-method (PLMset-class), 9
rma, 4, 13, 15, 17, 19
rmaPLM, 14, 19
sampleNames,PLMset-method (PLMset-class), 9
sampleNames<-(PLMset-class), 9
sampleNames<-,PLMset.character-method (PLMset-class), 9
se (PLMset-class), 9
se,PLMset-method (PLMset-class), 9
se.const (PLMset-class), 9
se.const,PLMset-method (PLMset-class), 9
se.probe (PLMset-class), 9
se.probe,PLMset-method (PLMset-class), 9
se<-(PLMset-class), 9
se<-,PLMset-method (PLMset-class), 9
show,PLMset-method (PLMset-class), 9
summary,PLMset-method (PLMset-class), 9
threestep, 4, 12, 15, 16, 18, 19
threestepPLM, 15, 18
varcov (PLMset-class), 9
varcov,PLMset-method (PLMset-class), 9
weights (PLMset-class), 9
weights,PLMset-method (PLMset-class), 9
weights<-(PLMset-class), 9
weights<-,PLMset-method (PLMset-class), 9