Package 'RPA'

September 24, 2012

Type Package

Title RPA: Robust Probabilistic Averaging for probe-level analysis

Version 1.12.0

Date 2012-02-25

Author Leo Lahti <`leo.lahti@iki.fi`>

Maintainer Leo Lahti <`leo.lahti@iki.fi`>

Depends R (>= 2.14.1), affy, affydata, methods, parallel

biocViews GeneExpression, Microarray, Preprocessing, QualityControl

Description Probabilistic analysis of probe reliability and
differential gene expression on short oligonucleotide
arrays. Lahti et al. ''Probabilistic Analysis of Probe
Reliability in Differential Gene Expression Studies with
Short Oligonucleotide Arrays'', TCBB/IEEE, 2011.
http://doi.ieeecomputersociety.org/10.1109/TCBB.2009.38

License FreeBSD

LazyLoad yes

R topics documented:

`RPA-package` *RPA: probabilistic analysis of probe reliability and gene expression*

Description

RPA estimates probe-specific variances and differential gene expression using probe-level observations of differential gene expression.

Details

Package:	RPA
Type:	Package
Version:	1.7.32
Date:	2011-03-12
License:	FreeBSD
LazyLoad:	yes

RPA.pointestimate is used to calculate probe reliability and differential expression estimates: 'rpa.results <- RPA.pointestimate(affybatch)'. The other functions are provided for users who wish to investigate the details of the algorithm more closely.

Author(s)

Leo Lahti <`leo.lahti@iki.fi`>

References

Probabilistic Analysis of Probe Reliability in Differential Gene Expression Studies with Short Oligonucleotide Arrays. Lahti et al., TCBB/IEEE 2011.

Examples

```
## Not run:

## Load example data set (Dilution affybatch).
## This is a toy example with a small example dataset
## for probe reliability analysis (4 arrays).
## For practical applications, a larger sample size is
## recommended.
```

```
#require(affy)
#require(affydata)
#data(Dilution)

## Compute RPA for whole data set
## ... slow, not executed here
#rpa.results <- RPA.pointestimate(Dilution)
```

`estimate.affinities` *Probe affinity estimation*

Description

Estimates probe-specific affinity parameters.

Usage

```
estimate.affinities(dat, mu)
```

Arguments

`dat`	Input data set: probes x samples.
`mu`	Estimated expression signal from RPA model.

Details

To estimate means in the original data domain let us assume that each probe-level observation x is of the following form: x = d + mu + noise, where x and d are vectors over samples, mu is a scalar (vector with identical elements) noise is Gaussian with zero mean and probe-specific variance parameters sigma2 Then the parameter mu will indicate how much probe-level observation deviates from the estimated signal shape d. This deviation is further decomposed as mu = mu.real + mu.probe, where mu.real describes the 'real' signal level, common for all probes mu.probe describes probe affinity effect Let us now assume that mu.probe ~ N(0, sigma.probe). This encodes the assumption that in general the affinity effect of each probe tends to be close to zero. Then we just calculate ML estimates of mu.real and mu.probe based on particular assumptions. Note that this part of the algorithm has not been defined in full probabilistic terms yet, just calculating the point estimates.

Note that while sigma2 in RPA measures stochastic noise, and NOT the affinity effect, we use it here as a heuristic solution to weigh the probes according to how much they contribute to the overall signal shape. Intuitively, probes that have little effect on the signal shape (i.e. are very noisy and likely to be contaminated by many unrelated signals) should also contribute less to the absolute signal estimate. If no other prior information is available, using stochastic parameters sigma2 to determine probe weights is likely to work better than simple averaging of the probes without weights. Also in this case the probe affinities sum close to zero but there is some flexibility, and more noisy probes can be downweighted.

Value

A vector with probe-specific affinities.

Note

Affinity estimation is not part of the original RPA procedure in TCBB/IEEE 2011 paper. It is added here since estimates of the absolute levels are often needed in microarray applications. Note that affinity parameters are unidentifiable in the model if no prior assumptions are given. We assume that affinity effects are zero on average, but allow flexibility through probe-specific weights.

Author(s)

Leo Lahti <leo.lahti@iki.fi>

References

See citation("RPA").

See Also

rpa.fit

Examples

```
##  mu <- estimate.affinities(dat, mu)
```

`estimate.hyperparameters` *Hyperparameter estimation*

Description

Hyperparameter estimation

Usage

```
estimate.hyperparameters(sets = NULL, priors = list(alpha = 2, beta =
1), batches, cdf = NULL, quantile.basis, bg.method = "rma", epsilon =
1e-2, load.batches = FALSE, save.hyperparameter.batches =
FALSE, mc.cores = 1, verbose = TRUE, normalization.method = "quantiles")
```

Arguments

`sets`	Probesets to handle. All probesets by default.
`priors`	User-defined priors
`batches`	Data batches for online learning
`cdf`	CDF probeset definition file
`quantile.basis`	Basis for quantile normalization
`bg.method`	Background correction method
`epsilon`	Convergence parameter
`load.batches`	Logical. Load preprocessed data whose identifiers are picked from names(batches). Assuming that the same batch list (batches) was used to create the files in online.quantiles function.

`save.hyperparameter.batches`	Save hyperparameters for each batch into files using the identifiers with batch name with -hyper.RData suffix.
`mc.cores`	Number of cores for parallel computation
`verbose`	Print progress information
`normalization.method`	Normalization method

Details

Sweeps through the batches. Updates hyperparameters at each batch.

Value

`alpha`	Hyperparameter alpha (same for all probesets)
`betas`	Hyperparameter beta (probe-specific)
`variances`	Probe-specific variances (beta/alpha)

Author(s)

Leo Lahti <`leo.lahti@iki.fi`>

References

See citation("RPA").

Examples

```
#
```

`get.batches` *Split data into batches*

Description

Splits data into batches

Usage

```
get.batches(items, batch.size, shuffle = FALSE)
```

Arguments

`items`	A vector of items to be splitted into batches.
`batch.size`	Batch size. The last batch may contain less elements than the other batches which have batch.size elements each.
`shuffle`	Split the elements randomly in the batches.

Value

A list. Each element corresponds to one batch and contains a vector listing the elements in that batch.

Author(s)

Leo Lahti <leo.lahti@iki.fi>

References

See citation("RPA").

See Also

RPA.preprocess

Examples

```
#
```

`get.probe.noise.estimates` *Fetch probe-level noise estimates from an rpa object*

Description

Provides estimates of probe-level noise, given by the RPA algorithm.

Usage

```
get.probe.noise.estimates(rpa.res, sets = NULL, normalization = NULL, verbose = FALSE)
```

Arguments

`rpa.res`	An rpa object.
`sets`	Probesets to check.
`normalization`	Optional normalization for probe noise estimates. The higher the value, the higher the probe-level noise. By default, probe-level variances of the RPA model are returned. Other options include: "withinset.weights": The relative weight of a probe within probeset is determined by the relative noise of the probe with respect to the other probes in the same probeset. This option returns the inverse of probe-specific weights within each probeset. This can be used to normalize probe-level weights to improve comparability across probesets. "withinset.relative": The detected probe-level noise can be coupled with overall signal levels of the probeset. This option provides an estimate of probe-wise standard deviation normalized by the standard deviation of the probeset-level signal d. "withinset.categorical": In some applications it can be sufficient to investigate the relative order of the probes, ignoring the parameter estimates. This option indexes the probes according to their reliability within each probeset. Probes with higher indices are more noisy.
`verbose`	Print progress information during computation.

Details

The normalization options are included to improve comparability across probesets. The higher the variance, the more noisy the probe. Inverse of the variance, can be used to quantitate probe reliability. Note that the relative weight of a probe within probeset is determined by the relative noise of the probe with respect to the other probes in the same probeset. Comparison of probe-specific variances across probesets may benefit from normalization of this effect. Therefore optional normalizations for probe noise estimation are provided.

Value

A list. Each element corresponds to one probeset (of the input object). The element lists noise estimates for each probe within the probeset.

Author(s)

Leo Lahti <`leo.lahti@iki.fi`>

References

Probabilistic Analysis of Probe Reliability in Differential Gene Expression Studies with Short Oligonucleotide Arrays. Lahti et al., TCBB/IEEE 2011. See http://bioconductor.org/packages/release/bioc/html/RPA.htm

See Also

RPA.pointestimate

Examples

```
## Load example data set
require(affydata)
data(Dilution)
## Compute RPA
rpa.results <- RPA.pointestimate(Dilution, set = "1000_at")
noise <- get.probe.noise.estimates(rpa.results)
```

`hyperparameter.update` *Update hyperparameters.*

Description

Update shape (alpha) and scale (beta) parameters of the inverse gamma distribution.

Usage

```
hyperparameter.update(dat, alpha, beta, th = 0.01)
```

Arguments

`dat`	A probes x samples matrix (probeset).
`alpha, beta`	Shape and scale parameters of inverse gamma density for the probe variances.
`th`	Convergence threshold.

Details

Shape update: alpha <- alpha + T/2; Scale update: beta <- alpha * s2 where s2 is the updated variance for each probe (the mode of variances is given by beta/alpha). The variances (s2) are updated by EM type algorithm, see s2.update.

Value

A list with elements alpha, beta (corresponding to the shape and scale parameters of inverse gamma distribution, respectively).

Author(s)

Leo Lahti <leo.lahti@iki.fi>

References

See citation("RPA").

See Also

s2.update, rpa.online

Examples

```
# Generate and fit toydata, learn hyperparameters
set.seed(11122)
P <- 11   # number of probes
N <- 5000 # number of arrays
real <- sample.probeset(P = P, n = N, shape = 3, scale = 1, mu.real = 4)
dat <- real$dat # probes x samples

# Set priors
alpha <- 1e-2
beta  <- rep(1e-2, P)

# Operate in batches
step <- 1000
for (ni in seq(1, N, step)) {
  batch <- ni:(ni+step-1)
  hp <- hyperparameter.update(dat[,batch], alpha, beta, th = 1e-2)
  alpha <- hp$alpha
  beta <- hp$beta
}

# Final variance estimate
s2 <- beta/alpha

# Compare real and estimated variances
plot(sqrt(real$variance), sqrt(s2), main = cor(sqrt(real$variance), sqrt(s2))); abline(0,1)
```

`online.quantile` *Quantile normalization tools for online preprocessing.*

Description

Estimate quantiles for quantile normalization based on subset of the data (random, or specified by the user).

Usage

```
online.quantile(abatch, n)
qnorm.basis.online(cel.files, bg.method = "rma", cdf = NULL,
save.batches = FALSE, batch.size = 2, verbose = TRUE)
```

Arguments

`abatch`	AffyBatch
`n`	Numeric: number of random samples to use to define quantile basis. Vector: specify samples to be used in quantile basis calculation.
`cel.files`	List or vector of CEL files to process. Can also be a list of batches, each batch containing a vector of CEL files
`bg.method`	Background correction method
`cdf`	Optional. CDF file for alternative probeset definitions
`save.batches`	Logical. If TRUE, save results for each batch into file with batchname.RData where 'batchname' is picked from the batch identifiers in the batch list (as given by get.batches function).
`batch.size`	Batch size
`verbose`	Print progress information

Details

"online.quantile": Ordinary quantile normalization is exhaustively memory-consuming in alrge data sets. Then the quantiles can be calculated based on subset of the data to allow efficient normalization. This function can also be used to investigate effect of subset size to convergence of the quantile estimates; "qnorm.basis.online": sweeps through the data in batches to calculate the basis for quantile normalization (average over sorted profiles).

Value

"online.quantile": AffyBatch; "qnorm.basis.online": a vector containing the basis for quantile normalization.

Author(s)

Leo Lahti <`leo.lahti@iki.fi`>

References

See citation("RPA").

See Also

RPA.preprocess

Examples

```
#
```

`plot` *Plot RPA results and probe-level data for a specific probeset.*

Description

Plots the preprocessed probe-level observations, estimated probeset-level signal, and probe-specific variances. It is also possible to highlight individual probes and external summary measures.

Usage

```
## S3 method for class 'rpa'
plot(x, y = NULL, set, highlight.probes = NULL, pcol =
"darkgrey", mucol = "black", ecol = "red", cex.lab = 1.5, cex.axis = 1, cex.names = 1, cex.main
```

Arguments

`x,y`	Instances of the 'rpa class; y is optional and never used. Provided for consistency.
`set`	Probeset to plot.
`highlight.probes`	Optionally highlight some of the probes (with dashed line)
`pcol`	Color for probe signal visualization.
`mucol`	Color for summary estimate.
`ecol`	Color for external signal.
`cex.lab, cex.axis, cex.names, cex.main`	Axis adjustment parameters.
`external.signal`	Plot external signal on the probeset. For instance, an alternative summary estimate from another preprocessing methods.
`main`	Title for plot.
`plots`	Type of plot. See help(rpa.plot).
`...`	Other arguments to be passed.

Value

Used for side-effects. Returns probes x samples matrix of probe-level data plotted on the image.

Author(s)

Leo Lahti <leo.lahti@iki.fi>

References

Probabilistic Analysis of Probe Reliability in Differential Gene Expression Studies with Short Oligonucleotide Arrays. Lahti et al., TCBB/IEEE. See http://bioconductor.org/packages/release/bioc/html/RPA.html

See Also

RPA.pointestimate

Examples

```
# Not run:

## Load example data set
#require(affydata)
#data(Dilution)

## Compute RPA for specific probesets only
#set <- "1000_at"
#rpa.results <- RPA.pointestimate(Dilution, set)

## Visualize the results for one of the probe sets
#plot(rpa.results, set)
```

`rpa` *RPA for preprocessing.*

Description

Returns an expressionSet object preprocessed with RPA. If 'cind' is not specified, uses the first array of affybatch as the reference.

Usage

```
rpa(abatch = NULL, sets = NULL, priors = NULL, epsilon = 1e-2, cind = 1, sigma2.method = "robust
```

Arguments

`abatch`	An AffyBatch object.
`sets`	Probesets for which RPA will be computed. Default: all probe sets.
`priors`	An 'rpa.priors' object. Can be used to set user-specified priors for the model parameters. Not used sigma2.method = "var". The prior parameters alpha and beta are prior parameters for inverse Gamma distribution of probe-specific variances. Noninformative prior is obtained with alpha, beta -> 0. Not used with sigma2.method 'var'. Scalar alpha and beta specify an identical inverse Gamma prior for all probes, which regularizes the solution. Can be also specified as lists, each element corresponding to one probeset.
`epsilon`	Convergence tolerance. The iteration is deemed converged when the change in all parameters is < epsilon.

`cind`	Specify reference array for computing probe-level differential expression. Default: cind = 1. Note that if exclude.reference.array = TRUE the expression value for the reference array (cind) will be excluded in the output. Note that all values of the reference array are 0 since they indicate the differential expression of the reference array against itself.
`sigma2.method`	Optimization method for sigma2 (probe-specific variances). This parameter is denoted by tau^2 in the vignette and manuscript. "robust": (default) update sigma2 by posterior mean, regularized by informative priors that are identical for all probes (user-specified by setting scalar values for alpha, beta). This regularizes the solution, and avoids overfitting where a single probe obtains infinite reliability. This is a potential problem in the other sigma2 update methods with non-informative variance priors. The default values alpha = 2; beta = 1 are used if alpha and beta are not specified. "mode": update sigma2 with posterior mean "mean": update sigma2 with posterior mean "var": update sigma2 with variance around d. Applies the fact that sigma2 cost function converges to variance with large sample sizes.
`d.method`	Method to optimize d. "fast": (default) weighted mean over the probes, weighted by probe variances The solution converges to this with large sample size. "basic": optimization scheme to find a mode used in Lahti et al. TCBB/IEEE; relatively slow; this is the preferred method with small sample sizes.
`verbose`	Print progress information during computation.
`bg.method`	Specify background correction method. Default: "rma". See bgcorrect.methods() for other options.
`normalization.method`	Specify quantile normalization method. Default: "pmonly". See normalize.methods(Dilution) for other options.
`cdf`	Specify an alternative CDF environment. Default: none.
`cel.files`	List of CEL files to preprocess.
`cel.path`	Path to CEL file directory.

Details

RPA preprocessing function. Gives an estimate of the probeset-level mean parameter d of the RPA model, and returns these in an expressionSet object.

Value

An instance of the 'expressionSet' class.

Note

The choices sigma2.method = "robust" and d.method = "fast" are recommended. With small sample size and informative prior, d.method = "basic" may be preferable. For very large expression data collections, see rpa.online function.

Author(s)

Leo Lahti <`leo.lahti@iki.fi`>

References

See citation("RPA").

See Also

rpa.online, RPA.pointestimate, set.priors, AffyBatch, ExpressionSet, estimate.affinities, rpa.fit

Examples

```
# Not run:
## Load example data set
#require(affydata)
#data(Dilution)
## eset <- rpa(Dilution)
```

`rpa-class` *Class "rpa"*

Description

Class for the RPA package.

Objects from the Class

Returned by `RPA.pointestimate` function. Objects can be created by calls of the form `new("rpa", ...)`.

Slots

Contains the following information for analyzed probesets and data:

Object of class `"list"`

`.Data`: A matrix of probesets x arrays. Specifies the estimated 'true' underlying gene expression signal over the arrays for each investigated probeset.

mu.real Numeric. Used for technical purposes. Indicates the difference between original signal levels and the signal levels in the differential signal domain used to learn the probe-specific variance parameters. See model description in vignette for details.

sigma2 A list. Each element corresponds to a probeset, and contains a vector that gives the estimated variance (noise level) for each probe in that probeset.

affinity A list. Each element corresponds to a probeset, and contains a vector that gives the estimated affinity effect for each probe in that probeset.

cind Specifies which of the arrays in abatch was used as a reference for computing probe-level differential expression.

sets A character vector listing the investigated probesets.

cdf Alternative CDF that was used in the analysis.

data Preprocessed probe-level data on which the model was fitted.

abatch The associated affybatch object.

Extends

Class `"list"`, from data part. Class `"vector"`, by class "list", distance 2.

Methods

[`signature(x = "rpa")`: ...

[[`signature(x = "rpa")`: ...

show `signature(x = "rpa")`: ...

Author(s)

Leo Lahti `<leo.lahti@iki.fi>`

References

Probabilistic Analysis of Probe Reliability in Differential Gene Expression Studies with Short Oligonucleotide Arrays. Lahti et al., TCBB/IEEE. http://www.roihu.info/publications/preprints/TCBB10.pdf

Examples

```
showClass("rpa")
```

`rpa.fit` *rpa.fit*

Description

Fits the RPA model, including estimation of probe-specific affinity parameters.

Usage

```
rpa.fit(dat, cind = 1, epsilon = 1e-2, alpha = NULL, beta = NULL,
sigma2.method = "robust", d.method = "fast")
```

Arguments

`dat`	Original data: probes x samples.
`cind`	Index of reference array.
`epsilon`	Convergence tolerance. The iteration is deemed converged when the change in all parameters is < epsilon.
`alpha, beta`	Priors for inverse Gamma distribution of probe-specific variances. Noninformative prior is obtained with alpha, beta -> 0. Not used with sigma2.method 'var'. Scalar alpha and beta are specify equal inverse Gamma prior for all probes to regularize the solution. The defaults depend on the method.
`sigma2.method`	Optimization method for sigma2 (probe-specific variances). "robust": (default) update sigma2 by posterior mean, regularized by informative priors that are identical for all probes (user-specified by setting scalar values for alpha, beta). This regularizes the solution, and avoids overfitting where a single probe obtains infinite reliability. This is a potential problem in the other sigma2 update methods with non-informative variance priors. The default values alpha = 2; beta = 1 are used if alpha and beta are not specified. "mode": update sigma2 with posterior mean "mean": update sigma2 with posterior mean "var": update sigma2 with variance around d. Applies the fact that sigma2 cost function converges to variance with large sample sizes.

`d.method`	Method to optimize d. "fast": (default) weighted mean over the probes, weighted by probe variances The solution converges to this with large sample size. "basic": optimization scheme to find a mode used in Lahti et al. TCBB/IEEE; relatively slow; preferred with small sample size.

Details

First learns a point estimate for the RPA model in terms of differential expression values w.r.t. reference sample. After this, probe affinities are estimated by comparing original data and differential expression shape, and setting prior assumptions concerning probe affinities.

Value

`mu`	Fitted signal in original data: mu.real + d
`mu.real`	Shifting parameter of the reference sample
`sigma2`	Probe-specific stochastic noise
`affinity`	Probe-specific affinities
`data`	Probeset data matrix
`alpha, beta`	prior parameters

Note

Affinity estimation is not part of the original RPA procedure in TCBB/IEEE 2011 paper. It is added here since estimates of the absolute levels are often needed in microarray applications. Note that affinity parameters are unidentifiable in the model if no prior assumptions are given. We assume that affinity effects are zero on average, but allow some flexibility through probe-specific weights.

Author(s)

Leo Lahti <`leo.lahti@iki.fi`>

References

Probabilistic Analysis of Probe Reliability in Differential Gene Expression Studies with Short Oligonucleotide Arrays. Lahti et al., TCBB/IEEE 2011. See http://www.roihu.info/publications/preprints/TCBB10.pdf

See Also

rpa, RPA.pointestimate, estimate.affinities

Examples

```
## res <- rpa.fit(dat, cind, epsilon, alpha, beta, sigma2.method, d.method, affinity.method)
```

`rpa.fit-class` *Class "rpa.fit"*

Description

Class for the RPA package.

Objects from the Class

Returned by `rpa.fit` function. Objects can be created by calls of the form `new("rpa.fit", ...)`.

Slots

Contains the following information for analyzed probesets and data:

Object of class `"list"`

`.Data` **mu:** A matrix of probes x samples. Gives the summary estimate corresponding to the 'true' underlying signal over the samples.

mu.real Numeric. Used for technical purposes. Indicates the difference between original signal levels and the signal levels in the differential signal domain used to learn the probe-specific variance parameters. See model description in vignette for details.

sigma2 A list. Each element corresponds to a probeset, and contains a vector that gives the estimated variance (noise level) for each probe in that probeset.

affinity A list. Each element corresponds to a probeset, and contains a vector that gives the estimated affinity effect for each probe in that probeset.

data Original probe-level data.

alpha, beta Inverse Gamma prior parameters for variance (sigma2).

Extends

Class `"list"`, from data part. Class `"vector"`, by class "list", distance 2.

Author(s)

Leo Lahti `<leo.lahti@iki.fi>`

References

Probabilistic Analysis of Probe Reliability in Differential Gene Expression Studies with Short Oligonucleotide Arrays. Lahti et al., TCBB/IEEE 2011. http://www.roihu.info/publications/preprints/TCBB10.pdf

Examples

```
showClass("rpa.fit")
```

`RPA.iteration` *Estimating model parameters d and sigma2.*

Description

Finds point estimates of the model parameters d (estimated true signal underlying probe-level observations), and sigma2 (probe-specific variances).

Usage

```
RPA.iteration(S, epsilon = 1e-3,
                        alpha = NULL, beta = NULL,
                        sigma2.method = "fast", d.method = "fast",
                        maxloop = 1e6)
```

Arguments

`S`	Matrix of probe-level observations for a single probeset: samples x probes.
`epsilon`	Convergence tolerance. The iteration is deemed converged when the change in all parameters is < epsilon.
`alpha, beta`	Priors for inverse Gamma distribution of probe-specific variances. Noninformative prior is obtained with alpha, beta -> 0. Not used with sigma2.method 'var'. Scalar alpha and beta are specify equal inverse Gamma prior for all probes to regularize the solution. The defaults depend on the method.
`sigma2.method`	Optimization method for sigma2 (probe-specific variances). "robust": (default) update sigma2 by posterior mean, regularized by informative priors that are identical for all probes (user-specified by setting scalar values for alpha, beta). This regularizes the solution, and avoids overfitting where a single probe obtains infinite reliability. This is a potential problem in the other sigma2 update methods with non-informative variance priors. The default values alpha = 2; beta = 1 are used if alpha and beta are not specified. "mode": update sigma2 with posterior mean "mean": update sigma2 with posterior mean "var": update sigma2 with variance around d. Applies the fact that sigma2 cost function converges to variance with large sample sizes.
`d.method`	Method to optimize d. "fast": (default) weighted mean over the probes, weighted by probe variances The solution converges to this with large sample size. "basic": optimization scheme to find a mode used in Lahti et al. TCBB/IEEE; relatively slow; this is the preferred method with small sample sizes.
`maxloop`	Maximum number of iterations in the estimation process.

Details

Assuming data set S with P observations of signal d with Gaussian noise that is specific for each observation (specified by a vector sigma2 of length P), this method gives a point estimate of d and sigma2. Probe-level variance priors alpha, beta can be used with sigma2.methods 'robust', 'mode', and 'mean'. The d.method = "fast" is the recommended method for point computing point estimates with large sample size.

Value

A list with the following elements:

`d`	A vector. Estimated 'true' signal underlying the noisy probe-level observations.
`sigma2`	A vector. Estimated variances for each measurement (or probe).

Author(s)

Leo Lahti <`leo.lahti@iki.fi`>

References

Probabilistic Analysis of Probe Reliability in Differential Gene Expression Studies with Short Oligonucleotide Arrays. Lahti et al., TCBB/IEEE 2011.

Examples

```
## Not run:

## Preprocess probe-level data
## cind determines the 'reference' array
#Smat <- RPA.preprocess(Dilution, cind = 1)

## Pick probe-level data for one probe set
#pmindices <- pmindex(Dilution, "1000_at")[[1]]
#S <- t(Smat$fcmat[pmindices, ])

## RPA with default parameters:
#res <- RPA.iteration(S)
```

`rpa.list-class` *Class "rpa.list"*

Description

Class for the RPA package.

Objects from the Class

Objects can be created by calls of the form `new("rpa.list", ...)`.

Slots

An extended list. Contains the following information for one probeset.

Object of class `"list"`

`.Data`, `d`: A vector (probeset x arrays). Specifies the estimated 'true' underlying differential gene expression signal over the arrays (vs. the reference array 'cind') for the investigated probeset. Note that the reference array is not included.

mu.real Deviation between the signal shape and the real signal level. Here d is the sum of mu.real and shape; affinities are calculated around d = mu.real + shape. Note that in Lahti11 paper d denotes the shape; we will polish the notation later.

sigma2 Contains a vector that gives the estimated variance for each probe in the investigated probeset.

affinity A list. Each element corresponds to a probeset, and contains a vector that gives the estimated affinity effect for each probe in that probeset.

cind Specifies which of the arrays in abatch was used as the reference for computing probe-level differential expression.

set Probeset name.

data Preprocessed probe-level data on which the model was fitted.

Extends

Class "`list`", from data part. Class "`vector`", by class "list", distance 2.

Methods

plot `signature(x = "rpa.list")`: ...

Author(s)

Leo Lahti <`leo.lahti@iki.fi`>

References

Probabilistic Analysis of Probe Reliability in Differential Gene Expression Studies with Short Oligonucleotide Arrays. Lahti et al., TCBB/IEEE 2011.

Examples

```
showClass("rpa.list")
```

`rpa.online` *RPA-online for preprocessing very large expression data sets.*

Description

rpa.online is used to preprocess very large expression data collections based on a Bayesian hyperparameter update procedure. Returns an expressionSet object preprocessed with RPA.

Usage

```
rpa.online(cel.path = NULL, cel.files = NULL, sets = NULL, cdf =
                 NULL, bg.method = "rma", priors = list(alpha = 2, beta
                 = 1), epsilon = 1e-2, mc.cores = 1,
                 verbose = TRUE, shuffle = TRUE, batch.size = 10,
                 batches = NULL, quantile.basis = NULL, save.batches = FALSE)
```

Arguments

`cel.path`	Path to CEL file directory
`cel.files`	List of CEL files to preprocess
`sets`	Probesets for which RPA will be computed
`cdf`	Specify an alternative CDF environment
`bg.method`	Specify background correction method. See bgcorrect.methods() for options.
`priors`	An 'rpa.priors' object. Can be used to set user-specified priors for the model parameters. Not used sigma2.method = "var". The prior parameters alpha and beta are prior parameters for inverse Gamma distribution of probe-specific variances. Noninformative prior is obtained with alpha, beta -> 0. Not used with sigma2.method 'var'. Scalar alpha and beta specify an identical inverse Gamma prior for all probes, which regularizes the solution. Can be also specified as lists, each element corresponding to one probeset.
`epsilon`	Convergence tolerance. The iteration is deemed converged when the change in all parameters is < epsilon.
`mc.cores`	Number of cores for parallel computation
`verbose`	Print progress information during computation
`shuffle`	Form random batches
`batch.size`	Batch size for online mode (rpa.online); the complete list of CEL files will be preprocessed in batches with this size using Bayesian online-updates for probe-specific parameters.
`batches`	User-defined CEL file batches
`quantile.basis`	Pre-calculated basis for quantile normalization
`save.batches`	Save batches to speed up preprocessing.

Details

RPA preprocessing function. Gives an estimate of the probeset-level mean parameter d of the RPA model, and returns these in an expressionSet object. The CEL files are handled in batches to obtain Bayesian updates for probe-specific hyperpriors; after sweeping through the database in batches the results are combined. The online mode is useful for preprocessing very large expression data sets where ordinary preprocessing algorithms fail, without compromises in modelling stage.

Value

An instance of the 'expressionSet' class.

Note

Development version

Author(s)

Leo Lahti <`leo.lahti@iki.fi`>

References

See citation("RPA").

See Also

rpa, AffyBatch, ExpressionSet

Examples

```
# Not run:
# eset <- rpa.online(cel.file.path)
```

`rpa.plot` *Plot RPA results and probe-level data for a specified probeset.*

Description

Plots the preprocessed probe-level observations, estimated probeset-level signal, and probe-specific variances. It is also possible to highlight individual probes and external summary measures.

Usage

```
rpa.plot(dat, rpa.fit.object = NULL, toydata.object = NULL, highlight.probes = NULL, pcol =
"darkgrey", mucol = "black", ecol = "red", cex.lab = 1.5, cex.axis = 1, cex.main = 1, cex.names
```

Arguments

`dat`	Original data: probes x samples.
`rpa.fit.object`	An instance of the 'rpa.fit' class.
`toydata.object`	Optional. Output from sample.probeset toydata generator function. Can be used to compare (toy)data with known ground truth to RPA estimates from rpa.fit.object.
`highlight.probes`	Optionally highlight some of the probes (with dashed line)
`pcol`	Color for probe signal visualization.
`mucol`	Color for summary estimate.
`ecol`	Color for external signal.
`cex.lab, cex.axis, cex.main, cex.names`	Font size adjustment parameters.
`external.signal`	Plot external signal on the probeset. For instance, an alternative summary estimate from another preprocessing methods
`main`	Title text.
`plots`	"all": plot data and summary, noise and affinity "data": plot data and summary
`...`	Other parameters to pass for plot function.

Value

Used for its side-effects. Returns probes x samples matrix of probe-level data plotted on the image.

Author(s)

Leo Lahti <leo.lahti@iki.fi>

References

Probabilistic Analysis of Probe Reliability in Differential Gene Expression Studies with Short Oligonucleotide Arrays. Lahti et al., TCBB/IEEE. See http://www.cis.hut.fi/projects/mi/software/RPA/

See Also

RPA.pointestimate

Examples

```
# Not run:

## Load example data set
#require(affydata)
#data(Dilution)

## Compute RPA for specific probesets only
#set <- "1000_at"
#rpa.results <- RPA.pointestimate(Dilution, set)

## Visualize the results for one of the probe sets
#rpa.plot(set, rpa.results)
```

RPA.pointestimate *Computing point estimate for the model parameters for all probe sets.*

Description

Computes point estimate

Usage

```
RPA.pointestimate(abatch, sets = NULL, myseed = 101, priors =
NULL, epsilon = 1e-2, cind = 1, sigma2.method = "robust", d.method =
"fast", verbose = TRUE, bg.method = "rma", normalization.method =
"quantiles.robust", cdf = NULL)
```

Arguments

`abatch`	An AffyBatch object.
`sets`	Specifies the probesets for which RPA estimates will be computed. Default: all probe sets.
`myseed`	Specifies the random seed.
`priors`	Optional list containing hyperparameters alpha and beta of the inverse Gamma prior of the probe-specific variances; alpha is a scalar, common for all probes and probeset; beta is a list where each element is a vector corresponding to one probeset, specifying beta for each probe. Can be used to set user-specified priors for the model parameters. Not applicable for sigma2.method = "var". Noninformative prior is obtained with alpha, beta -> 0. Not used with sigma2.method 'var'. Can be used to regularize the solution with small sample size, or in batch-wise online-updates with large sample size. If priors are not provided for certain probesets (NULL), default priors are used.
`epsilon`	Convergence tolerance. The iteration is deemed converged when the change in the d parameter is < epsilon.
`cind`	Specifies which array in abatch is used as a reference in computing probe-level differential expression.
`sigma2.method`	Optimization method for sigma2 (probe-specific variances). "robust": (default) update sigma2 by posterior mean, regularized by informative priors that are identical for all probes (user-specified by setting scalar values for alpha, beta). This regularizes the solution and avoids overfitting where a single probe obtains infinite reliability. This is a potential problem in the other sigma2 update methods with non-informative variance priors. The default values alpha = 2; beta = 1 are used if alpha and beta are not specified. "mode": update sigma2 with posterior mean "mean": update sigma2 with posterior mean "var": update sigma2 with variance around d. Applies the fact that sigma2 cost function converges to variance with large sample sizes.
`d.method`	Method to optimize d. "fast": (default) weighted mean over the probes, weighted by probe variances The solution converges to this with large sample size. "basic": optimization scheme to find a mode used in Lahti et al. TCBB/IEEE; relatively slow; this is the preferred method with small sample sizes.
`verbose`	Print progress information during computation. Default: TRUE.
`bg.method`	Specify background correction method. Default: "rma". See bgcorrect.methods() for other options.
`normalization.method`	Specify quantile normalization method. Default: "pmonly". See normalize.methods(Dilution) for other options.
`cdf`	Specify an alternative CDF environment. Default: none.

Details

Calculates RPA estimates of probe reliability and differential expression between the user-specified reference array (cind) and the other arrays in the data set. The model assumes P observations for each transcript target (i.e. a probeset) with Gaussian noise which is specific for each probe (variance is specified by sigma2). The mean (affinity) parameters of the Gaussian noise model cancel out in calculating probe-level differential expression. RPA.pointestimate gives a point estimate for d and sigma2. The 'prior' parameter is not applicable with sigma2.method = "var". The d.method = "fast" is recommended with large sample size.

Value

An instance of class 'rpa'. This is an extended list containing the following elements:

`d`	A matrix of probesets x arrays. Specifies the estimated 'true' underlying differential gene expression signal over the arrays (vs. the reference array 'cind') for each investigated probeset. Note that the reference array is not included.
`sigma2`	A list. Each element corresponds to a probeset, and contains a vector that gives the estimated variance for each probe in that probeset. This corresponds to the parameter tau^2 in the vignette and manuscript.
`cind`	Specifies which of the arrays in the abatch (the affybatch object to be analyzed) has been used as the reference for computing probe-level differential expression.
`affinity`	Probe affinity effects.
`sets`	A character vector listing the investigated probesets.

Note

sigma2.method = "robust" and d.method = "fast" are recommended. With small sample size and informative priors, d.method = "basic" may be preferable, with large sample size d.method = "fast" should have considerable speedups and comparable accuracy.

Author(s)

Leo Lahti <`leo.lahti@iki.fi`>

References

See citation("RPA").

See Also

rpa.plot, rpa, set.priors, rpa2eset, RPA.preprocess, AffyBatch, rpa.fit, estimate.affinities

Examples

```
## Load example data set
#require(affydata)
#data(Dilution)

## Compute RPA for whole data set
## ... slow, not executed here
# rpa.results <- RPA.pointestimate(Dilution)

## Visualize the results for one of the probe sets
#rpa.plot(set, rpa.results)
```

RPA.preprocess *Preprocess AffyBatch object for RPA.*

Description

Background correction, quantile normalization and log2-transformation for probe-level raw data in abatch. Then probe-level differential expression is computed between the specified 'reference' array (cind) and the other arrays.

Usage

```
RPA.preprocess(abatch, bg.method = "rma",
                      normalization.method = "quantiles.robust",
      cdf = NULL,
      cel.files = NULL, cel.path = NULL)
```

Arguments

`abatch`	An AffyBatch object.
`bg.method`	Specify background correction method. See bgcorrect.methods(abatch) for options.
`normalization.method`	Specify normalization method. See normalize.methods(abatch) for options. For memory-efficient online version, use "quantiles.online".
`cdf`	The CDF environment used in the analysis.
`cel.files`	List of CEL files to preprocess.
`cel.path`	Path to CEL file directory.

Details

Probe-specific variance estimates are robust against the choice of reference array.

Value

`fcmat`	Probes x arrays preprocessed differential expression matrix.
`cind`	Specifies which array in abatch was selected as a reference in calculating probe-level differential expression.
`cdf`	The CDF environment used in the analysis.
`set.inds`	Indices for probes in each probeset, corresponding to the rows of fcmat.

Author(s)

Leo Lahti <leo.lahti@iki.fi>

References

See citation("RPA").

See Also

AffyBatch

rpa2eset *Coerce 'rpa' object into an 'ExpressionSet'*

Description

An instance of 'rpa' class contains differential gene expression estimates in the variable 'd'. The function 'rpa2eset' coerces this into an ExpressionSet object to allow downstream analysis of the results using standard R/BioC tools for gene expression data.

Usage

```
rpa2eset(x)
```

Arguments

x	An instance of the rpa class (obtained as output from RPA.pointestimate)

Value

An 'ExpressionSet' object.

Author(s)

Leo Lahti <leo.lahti@iki.fi>

References

Probabilistic Analysis of Probe Reliability in Differential Gene Expression Studies with Short Oligonucleotide Arrays. Lahti et al., TCBB/IEEE. See http://bioconductor.org/packages/release/bioc/html/RPA.html

Examples

```
# Not run:

#require(RPA)
#require(affy)
#require(affydata)
#data(Dilution)

## Compute RPA for specific probesets
#sets <- geneNames(Dilution)[1:2]
#rpa.results <- RPA.pointestimate(Dilution,sets)

## Coerce the rpa object into an ExpressionSet
#eset <- rpa2eset(rpa.results)
```

`s2.update` *Variance update.*

Description

Update variance parameters in the Bayesian RPA model with an EM-type optimization procedure and potentially informative priors.

Usage

```
s2.update(dat, alpha = 1e-2, beta = 1e-2, s2.init = NULL, th = 0.01)
```

Arguments

`dat`	A probes x samples matrix (probeset).
`alpha, beta`	Shape and scale parameters of the inverse gamma prior for the variances.
`s2.init`	Initial values for the variances for optimization.
`th`	Optimization convergence threshold.

Details

Updates the variance parameters in the Bayesian RPA model with a (generalized) EM-type optimization procedure and potentially informative priors. The variances are updated by their mode at each step until convergence. Priors (alpha, beta) are taken into account if provided.

Value

A vector of variance parameters, one for each probe.

Author(s)

Leo Lahti <`leo.lahti@iki.fi`>

References

See citation("RPA").

See Also

rpa.online

Examples

```
# dat: probes x samples matrix (probeset)
# s2 <- s2.update(dat)
```

`sample.probeset` *Toydata generator for probeset data.*

Description

Generate random probeset with varying probe-specific affinities and variances.

Usage

```
sample.probeset(P = 10, n = 20, shape = 1, scale = 1, mu.real = 2)
```

Arguments

`P`	Number of probes.
`n`	Number of samples.
`shape, scale`	Shape and scale parameters of the inverse Gamma function used to generate the probe-specific variances.
`mu.real`	Absolute signal level of the probeset.

Details

The toy data generator follows distributional assumptions of the RPA model and allows quantitative estimation of model accuracy with different options, noise levels and sample sizes. Probeset-level summary estimate is obtained as mu.real + d.

Value

A list with the following elements:

`dat`	Probeset data: probes x samples
`variance`	Probe variances.
`affinity`	Probe affinities.
`d`	Probeset-level signal shape.
`mu.real`	Probeset-level absolute signal level.

Author(s)

Leo Lahti <`leo.lahti@iki.fi`>

References

See citation("RPA").

Examples

```
real <- sample.probeset(P = 10, n = 20, shape = 1, scale = 1, mu.real = 2)
```

`summarize.batches` *Summarize batches*

Description

Probeset summarization in batches

Usage

```
summarize.batches(sets, variances, batches, load.batches = FALSE,
mc.cores = 1, cdf = NULL, bg.method = "rma", normalization.method =
"quantiles", verbose = TRUE, quantile.basis)
```

Arguments

`sets`	Probesets to summarize
`variances`	Precalculated probe-specific variances
`batches`	Data batches for online learning
`load.batches`	Logical. Load precalculated data for the batches.
`mc.cores`	Number of cores for parallel computation
`cdf`	CDF for alternative probeset definitions
`bg.method`	Background correction method
`normalization.method`	Normalization method
`verbose`	Print progress information
`quantile.basis`	Basis for quantile normalization

Details

Sweeps through the batches. Summarizes the probesets within each batch based on the precalculated model parameters using point estimates of the model.

Value

Expression matrix: probesets x samples.

Author(s)

Leo Lahti <`leo.lahti@iki.fi`>

References

See citation("RPA").

Examples

```
#
```

Index