Package ‘RPA’

April 23, 2016

Type Package

Title RPA: Robust Probabilistic Averaging for probe-level analysis

Version 1.26.0

Date 2014-09-14

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

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

Depends R (>= 3.1.1), parallel, affy, methods

Suggests affydata

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 BSD_2_clause + file LICENSE

URL `https://github.com/antagomir/RPA`

BugReports `https://github.com/antagomir/RPA`

LazyLoad yes

NeedsCompilation no

R topics documented:

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

Description

Brief summary of the RPA package

Details

Package:	RPA
Type:	Package
Version:	See sessionInfo() or DESCRIPTION file
Date:	2008-2013
License:	FreeBSD
LazyLoad:	yes

Author(s)

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

References

See citation("RPA")

Examples

```
#
```

`collect.hyperparameters` *collect.hyperparameters*

Description

Collect probe-level parameters during online-learning from the batch files.

Usage

```
collect.hyperparameters(batches, unique.run.identifier, save.batches.dir,
  save.batches, verbose = TRUE)
```

Arguments

`batches`	batch list
`unique.run.identifier`	Batch file identifier string
`save.batches.dir`	Batch file directory
`save.batches`	Logical. Determines whether batches are available.
`verbose`	verbose

Author(s)

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

References

See citation("RPA")

Examples

```
# hpe <- collect.hyperparameters(batches, unique.run.identifier, save.batches.dir, save.batches)
```

`d.update.fast` *d.update.fast*

Description

Description: computes weighted average over the probes, weighted by their inverse probe-specific variances.

Usage

```
d.update.fast(St, s2)
```

Arguments

`St`	probes x samples data matrix
`s2`	variances for the probes

Details

Returns summarized probeset-level weighted average

Author(s)

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

References

See citation("RPA")

Examples

```
#
```

`estimate.affinities` *estimate.affinities*

Description

Probe affinity estimation. Estimates probe-specific affinity parameters.

Usage

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

Arguments

`dat`	Input data set: probes x samples.
`a`	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 + v + noise, where x and d are vectors over samples, v is a scalar (vector with identical elements) noise is Gaussian with zero mean and probe-specific variance parameters tau2 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 tau2 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 tau2 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.

Author(s)

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

References

See citation("RPA")

See Also

rpa.fit

Examples

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

`estimate.hyperparameters`

estimate.hyperparameters Hyperparameter estimation

Description

estimate.hyperparameters Hyperparameter estimation

Usage

```
estimate.hyperparameters(sets = NULL, probe.parameters = list(alpha = 2,
  beta = 1), batches, cdf = NULL, bg.method = "rma", epsilon = 0.01,
  load.batches = FALSE, save.hyperparameter.batches = FALSE, mc.cores = 1,
  verbose = TRUE, normalization.method = "quantiles",
  save.batches.dir = ".", unique.run.identifier = NULL,
  set.inds = set.inds)
```

Arguments

`sets`	Probesets to handle. All probesets by default.
`probe.parameters`	User-defined priors. May also include quantile.basis
`batches`	Data batches for online learning
`cdf`	CDF probeset definition file
`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
`save.batches.dir`	Specify the output directory for temporary batch saves.
`unique.run.identifier`	Define identifier for this run for naming the temporary batch files. By default, a random id is generated.
`set.inds`	Probeset indices

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

```
#
```

`frpa` *frpa frozen-RPA preprocessing using precalculated probe parameters*

Description

frpa frozen-RPA preprocessing using precalculated probe parameters

Usage

```
frpa(abatch = NULL, probe.parameters = NULL, verbose = FALSE,
  cdf = NULL, cel.files = NULL, cel.path = NULL, mc.cores = 1,
  summarize.with.affinities = FALSE)
```

Arguments

`abatch`	An AffyBatch object.
`probe.parameters`	A list with tau2 (probe variance), quantile.basis (basis for quantile normalization in log2 domain), and optionally affinity (probe affinities). The probe.parameters$tau2 and probe.parameters$affinity are lists, each element corresponding to a probeset and containing a parameter vector over the probes. The quantile.basis is a vector over the probes, the probes need to be listed in the same order as in tau2 and affinity. probe.parameters can be optionally provided as a data frame.
`verbose`	Print progress information during computation.
`cdf`	Specify an alternative CDF environment. Default: none.
`cel.files`	List of CEL files to preprocess.
`cel.path`	Path to CEL file directory.
`mc.cores`	Number of cores for parallelized processing.
`summarize.with.affinities`	Use affinity estimates in probe summarization step. Default: FALSE.

Details

fRPA function to preprocess Affymetrix CEL files with RPA using precalculated (frozen) probe parameters.

Value

Preprocessed expression matrix in expressionSet format

Author(s)

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

References

See citation("RPA")

See Also

rpa, AffyBatch, ExpressionSet

Examples

```
# eset <- frpa(abatch, probe.parameters)
```

`get.batches` *get.batches Split data into batches*

Description

get.batches Split data into batches

Usage

```
get.batches(items, batch.size = NULL, 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")

Examples

```
#
```

`get.probe.matrix` *get.probe.matrix*

Description

get.probe.matrix

Usage

```
get.probe.matrix(cels, cdf = NULL, quantile.basis, bg.method = "rma",
  normalization.method = "quantiles", batch = NULL, verbose = TRUE)
```

Arguments

`cels`	List of CEL files to preprocess
`cdf`	Specify an alternative CDF environment
`quantile.basis`	Pre-calculated basis for quantile normalization in log2 domain
`bg.method`	Specify background correction method. See bgcorrect.methods() for options.
`normalization.method`	normalization method
`batch`	batch
`verbose`	Print progress information during computation

Details

Returns background-corrected, quantile normalized log2 probes x samples matrix

Author(s)

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

References

See citation("RPA")

Examples

```
#
```

`get.probe.parameters` *get.probe.parameters*

Description

Get probe-level hyperparameter from batch files

Usage

```
get.probe.parameters(affinities, unique.run.identifier,
  save.batches.dir = ".", mode = "list")
```

Arguments

`affinities`	probe affinities
`unique.run.identifier`	Batch file identifier string
`save.batches.dir`	Batch file directory
`mode`	"list" or "table"

Author(s)

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

References

See citation("RPA")

Examples

```
# df <- get.probe.parameters(unique.run.identifier, save.batches.dir = ".", mode = "list")
```

`hyperparameter.update` *hyperparameter.update Update hyperparameters Update shape (alpha) and scale (beta) parameters of the inverse gamma distribution.*

Description

hyperparameter.update Update hyperparameters 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`	Shape parameter of inverse gamma density for the probe variances.
`beta`	Scale parameter 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$tau2), sqrt(s2), main = cor(sqrt(real$tau2), sqrt(s2))); abline(0,1)
```

`online.quantile` *online.quantile Quantile normalization tools for online preprocessing. Estimate quantiles for quantile normalization based on subset of the data (random, or specified by the user).*

Description

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

Usage

```
online.quantile(abatch, n)
```

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.

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")

Examples

```
#
```

`probe.parameters.tolist` *probe.parameters.tolist*

Description

Convert probe parameter table into a list format

Usage

```
probe.parameters.tolist(probe.parameters)
```

Arguments

`probe.parameters`	A data.frame with alpha, betas, tau2, affinities, quantile.basis

Author(s)

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

References

See citation("RPA")

Examples

```
# df <- probe.parameters.tolist(probe.parameters.table)
```

`probe.performance` *probe.performance*

Description

Provide a table of probe-level parameter estimates (affinity and stochastic noise) for RPA output.

Usage

```
probe.performance(probe.parameters, abatch, sets = NULL)
```

Arguments

`probe.parameters`	List with affinities and variances for the probesets
`abatch`	Affybatch used in the analysis
`sets`	Specify the probesets to include in the output. Default: All probesets

Value

Data frame of probe-level parameter estimates

Author(s)

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

References

See citation("RPA")

Examples

```
# library(affydata); data(Dilution); rpa.results <- rpa.complete(Dilution); tab <- probe.parameters(rpa.results)
```

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

Description

probeplot Plot RPA results and probe-level data for a specified probeset.

Usage

```
probeplot(dat, highlight.probes = NULL, pcol = "darkgrey", hcol = "red",
  cex.lab = 1.5, cex.axis = 1, cex.main = 1, cex.names = 1, main = "",
  ...)
```

Arguments

`dat`	Background-corrected and normalized data: probes x samples.
`highlight.probes`	Optionally highlight some of the probes (with dashed line)
`pcol`	Color for probe signal visualization.
`hcol`	Color for probe highlight
`cex.lab`	Label size adjustment parameters.
`cex.axis`	Axis size adjustment parameters.
`cex.main`	Title size adjustment parameters.
`cex.names`	Names size adjustment parameters.
`main`	Title text.
`...`	Other parameters to pass for plot function.

Details

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.

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

See citation("RPA")

Examples

```
#
```

`probetable` *probetable*

Description

Convert probe-level hyperparameter lists into a table format.

Usage

```
probetable(probe.parameters)
```

Arguments

`probe.parameters`	A list with alpha, betas, variances and affinities

Author(s)

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

References

See citation("RPA")

Examples

```
# df <- probetable(probe.parameters)
```

`rpa` *rpa Wrapper for RPA preprocessing.*

Description

rpa Wrapper for RPA preprocessing.

Usage

```
rpa(abatch = NULL, verbose = FALSE, bg.method = "rma",
  normalization.method = "quantiles.robust", cdf = NULL, cel.files = NULL,
  cel.path = NULL, probe.parameters = NULL, mc.cores = 1,
  summarize.with.affinities = FALSE)
```

Arguments

`abatch`	An AffyBatch object.
`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.
`probe.parameters`	A list, each element corresponding to a probe set. Each probeset element has the following optional elements: mu (affinity), tau2 (variance), alpha (shape prior), beta (scale prior). Each of these elements contains a vector over the probeset probes, specifying the probe parameters according to the RPA model. If variance is given, it overrides the priors. Can be also used to set user-specified priors for the model parameters. Not used tau2.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 tau2.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. May also include quantile.basis
`mc.cores`	Number of cores for parallelized processing.
`summarize.with.affinities`	Use affinity estimates in probe summarization step. Default: FALSE.

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 choices tau2.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.

Value

Preprocessed expression matrix in expressionSet format

Author(s)

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

References

See citation("RPA")

See Also

rpa.online, AffyBatch, ExpressionSet, estimate.affinities, rpa.fit

Examples

```
# eset <- rpa(abatch)
```

`rpa.complete` *rpa.complete RPA preprocessing, also returns probe parameters*

Description

rpa.complete RPA preprocessing, also returns probe parameters

Usage

```
rpa.complete(abatch = NULL, sets = NULL, epsilon = 0.01,
  tau2.method = "robust", d.method = "fast", verbose = FALSE,
  bg.method = "rma", normalization.method = "quantiles.robust",
  cdf = NULL, cel.files = NULL, cel.path = NULL,
  probe.parameters = list(), mc.cores = 1,
  summarize.with.affinities = FALSE)
```

Arguments

`abatch`	An AffyBatch object.
`sets`	Probesets for which RPA will be computed.
`epsilon`	Convergence tolerance. The iteration is deemed converged when the change in all parameters is < epsilon.
`tau2.method`	Optimization method for tau2 (probe-specific variances). This parameter is denoted by tau^2 in the vignette and manuscript "robust": (default) update tau2 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 tau2 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 tau2 with posterior mean "mean": update tau2 with posterior mean "var": update tau2 with variance around d. Applies the fact that tau2 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.
`probe.parameters`	A list, each element corresponding to a probe set. Each probeset element has the following optional elements: affinity (affinity), tau2 (variance), alpha (shape prior), betas (scale prior). Each of these elements contains a vector over the probeset probes, specifying the probe parameters according to the RPA model. If variance is given, it overrides the priors. Can be also used to set user-specified priors for the model parameters. Not used tau2.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 tau2.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. Can also include quantile.basis

`mc.cores`	Number of cores for parallelized processing.
`summarize.with.affinities`	Use affinity estimates in probe summarization step. Default: FALSE.

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 choices tau2.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.

Value

List with preprocessed expression matrix, corresponding probe parameters, AffyBatch and CDF

Author(s)

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

References

See citation("RPA")

Examples

```
# eset <- rpa(abatch)
```

`rpa.fit` *rpa.fit*

Description

rpa.fit

Usage

```
rpa.fit(dat, epsilon = 0.01, alpha = NULL, beta = NULL,
  tau2.method = "robust", d.method = "fast",
  summarize.with.affinities = FALSE)
```

Arguments

`dat`	Original data: probes x samples.
`epsilon`	Convergence tolerance. The iteration is deemed converged when the change in all parameters is < epsilon.

`alpha`	alpha prior for inverse Gamma distribution of probe-specific variances. Non-informative prior is obtained with alpha, beta -> 0. Not used with tau2.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.
`beta`	beta prior for inverse Gamma distribution of probe-specific variances. Noninformative prior is obtained with alpha, beta -> 0. Not used with tau2.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.
`tau2.method`	Optimization method for tau2 (probe-specific variances); "robust": (default) update tau2 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 tau2 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 tau2 with posterior mean "mean": update tau2 with posterior mean "var": update tau2 with variance around d. Applies the fact that tau2 cost function converges to variance with large sample sizes.
`d.method`	Method used to optimize d. Options: "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.
`summarize.with.affinities`	Use affinity estimates in probe summarization step. Default: FALSE.

Details

Fits the RPA model, including estimation of probe-specific affinity parameters. 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; tau2: Probe-specific stochastic noise; affinity: Probe-specific affinities; data: Probeset data matrix; alpha, beta: prior parameters

Author(s)

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

References

See citation("RPA")

See Also

rpa, estimate.affinities

Examples

```
# res <- rpa.fit(dat, epsilon, alpha, beta, tau2.method, d.method, affinity.method)
```

RPA.iteration *RPA.iteration Estimating model parameters d and tau2.*

Description

RPA.iteration Estimating model parameters d and tau2.

Usage

```
RPA.iteration(S, epsilon = 0.001, alpha = NULL, beta = NULL,
  tau2.method = "fast", d.method = "fast", maxloop = 1e+06)
```

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`	alpha prior for inverse Gamma distribution of probe-specific variances. Noninformative prior is obtained with alpha, beta -> 0. Not used with tau2.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.
`beta`	beta prior for inverse Gamma distribution of probe-specific variances. Noninformative prior is obtained with alpha, beta -> 0. Not used with tau2.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.
`tau2.method`	Optimization method for tau2 (probe-specific variances). "robust": (default) update tau2 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 tau2 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 tau2 with posterior mean "mean": update tau2 with posterior mean "var": update tau2 with variance around d. Applies the fact that tau2 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

Finds point estimates of the model parameters d (estimated true signal underlying probe-level observations), and tau2 (probe-specific variances). Assuming data set S with P observations of signal d with Gaussian noise that is specific for each observation (specified by a vector tau2 of length P), this method gives a point estimate of d and tau2. Probe-level variance priors alpha, beta can be used with tau2.methods 'robust', 'mode', and 'mean'. The d.method = "fast" is the recommended method for point computing point estimates with large samples size.

Value

A list with the following elements: d: A vector. Estimated 'true' signal underlying the noisy probe-level observations.; tau2: A vector. Estimated variances for each measurement (or probe).

Author(s)

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

References

See citation("RPA")

Examples

```
#
```

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

Description

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

Usage

```
rpa.online(cel.path = NULL, cel.files = NULL, sets = NULL, cdf = NULL,
  bg.method = "rma", probe.parameters = list(alpha = 1, beta = 1),
  epsilon = 0.01, mc.cores = 1, verbose = TRUE, shuffle = TRUE,
  batch.size = 100, batches = NULL, save.batches.dir = ".",
  keep.batch.files = FALSE, unique.run.identifier = paste("RPA-run-id-",
  rnorm(1), sep = ""), rseed = 23, speedup = TRUE,
  summarize.with.affinities = 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.
`probe.parameters`	Can be used to set user-specified priors for the model parameters alpha, beta. Not used tau2.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 tau2.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. May also include quantile.basis, which should be provided at log2 domain.
`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
`save.batches.dir`	Output directory for temporary batch saves.
`keep.batch.files`	Logical. Keep (TRUE) or remove (FALSE) the batch files after preprocessing.
`unique.run.identifier`	Define identifier for this run for naming the temporary batch files. By default, a random id is generated.
`rseed`	Random seed.
`speedup`	Speed up computations with approximations.
`summarize.with.affinities`	Use affinity estimates in probe summarization step. Default: FALSE.

Details

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. 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

List with two elements: an instance of the 'expressionSet' class and probe parameters. For probe.parameters contents, see the probe.parameters input argument.

Author(s)

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

References

See citation("RPA")

See Also

rpa, AffyBatch, ExpressionSet

Examples

```
# eset <- rpa.online(cel.file.path)
```

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

Description

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

Usage

```
rpa.plot(x, set, highlight.probes = NULL, pcol = "darkgrey",
  mucol = "black", ecol = "red", external.signal = NULL, main = NULL,
  plots = "all", ...)
```

Arguments

`x`	Output from rpa.complete function
`set`	probeset
`highlight.probes`	mark probes for highlight
`pcol`	probe color
`mucol`	probeset signal color
`ecol`	external signal color
`external.signal`	external signal to be plotted on top
`main`	title
`plots`	plot type
`...`	other arguments to be passed

Details

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.

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

See citation("RPA")

Examples

```
#
```

`RPA.preprocess` *RPA.preprocess Preprocess AffyBatch object for RPA.*

Description

RPA.preprocess Preprocess AffyBatch object for RPA.

Usage

```
RPA.preprocess(abatch, bg.method = "rma",
  normalization.method = "quantiles.robust", cdf = NULL, cel.files = NULL,
  cel.path = NULL, quantile.basis = 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.
`quantile.basis`	Optional. Basis for quantile normalization. NOTE: required in original, not log2 scale!

Details

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. 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")

Examples

```
#
```

`rpa.summarize` *rpa.summarize*

Description

rpa.summarize

Usage

```
rpa.summarize(dat, affinities, variances, summarize.with.affinities = FALSE)
```

Arguments

`dat`	Original data: probes x samples.
`affinities`	Probe affinities
`variances`	Probe variances
`summarize.with.affinities`	Use affinity estimates in probe summarization step. Default: FALSE.

Details

Summarizes the probes in a probe set according to the RPA model based on the given affinity and variance parameters.

Value

A vector. Probeset-level summary signal.

Author(s)

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

References

See citation("RPA")

See Also

rpa

Examples

```
# res <- rpa.summarize(dat, affinities, variances, summarize.with.affinities = FALSE)
```

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

Description

rpaplot Plot RPA results and probe-level data for a specified probeset.

Usage

```
rpaplot(dat, mu = NULL, tau2 = NULL, affinity = NULL,
  highlight.probes = NULL, pcol = "darkgrey", mucol = "black",
  ecol = "red", cex.lab = 1.5, cex.axis = 1, cex.main = 1,
  cex.names = 1, external.signal = NULL, main = "", plots = "all", ...)
```

Arguments

`dat`	Background-corrected and normalized data: probes x samples.
`mu`	probeset signal
`tau2`	probe variances
`affinity`	probe affinities
`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`	Label size adjustment parameters.

`cex.axis`	Axis size adjustment parameters.
`cex.main`	Title size adjustment parameters.
`cex.names`	Names 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.

Details

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.

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

See citation("RPA")

Examples

```
#
```

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

Description

sample.probeset Toydata generator for probeset data.

Usage

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

Arguments

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

Details

Generate random probeset with varying probe-specific affinities and variances. 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
`tau2`	Probe variances.
`affinity`	Probe affinities.
`d`	Probeset signal shape.
`mu.real`	Probeset signal level.
`mu`	Probeset-level total signal.

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.batch` *summarize.batch*

Description

summarize.batch

Usage

```
summarize.batch(q, set.inds, probe.parameters = list(), epsilon,
  verbose = FALSE, mc.cores = 1, summarize.with.affinities = FALSE)
```

Arguments

`q`	Background corrected, quantile-normalized, log2 probes x samples matrix
`set.inds`	Indices for each probeset, corresponding to q matrix
`probe.parameters`	A list, each element corresponding to a probe set. Each probeset element has the following elements: affinity, variance and optionally alpha and beta priors. Each of these elements contains a vector over the probeset probes, specifying the probe parameters according to the RPA model. If variances are given, that overrides the priors.
`epsilon`	Convergence tolerance. The iteration is deemed converged when the change in all parameters is < epsilon.
`verbose`	Print progress information during computation.
`mc.cores`	Number of cores for parallel processing
`summarize.with.affinities`	Use affinity estimates in probe summarization step. Default: FALSE.

Author(s)

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

References

See citation("RPA")

Examples

```
#
```

`summarize.batches` *summarize.batches*

Description

summarize.batches

Usage

```
summarize.batches(sets = NULL, probe.parameters = list(), batches,
  load.batches = FALSE, mc.cores = 1, cdf = NULL, bg.method = "rma",
  normalization.method = "quantiles", verbose = TRUE,
  save.batches.dir = ".", unique.run.identifier = NULL,
  save.batches = FALSE, set.inds, speedup = FALSE,
  summarize.with.affinities = FALSE)
```

Arguments

`sets`	Probesets to summarize
`probe.parameters`	Optional probe parameters, including priors.
`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
`save.batches.dir`	Specify the output directory for temporary batch saves.
`unique.run.identifier`	Define identifier for this run for naming the temporary batch files. By default, a random id is generated.
`save.batches`	Save batches?
`set.inds`	Probeset indices
`speedup`	Speed up calculations with approximations.
`summarize.with.affinities`	Use affinity estimates in probe summarization step. Default: FALSE.

Details

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

Value

Expression matrix: probesets x samples.

Author(s)

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

References

See citation("RPA")

Examples

```
#
```

`updating.hyperparameters`

updating.hyperparameters Hyperparameter update

Description

updating.hyperparameters Hyperparameter update

Usage

```
updating.hyperparameters(q, set.inds, verbose, mc.cores = 1, alpha, betas,
  epsilon)
```

Arguments

`q`	probes x samples matrix
`set.inds`	Probe set indices
`verbose`	Print progress information
`mc.cores`	Number of cores for parallel computation
`alpha`	alpha hyperparameter
`betas`	beta hyperparameters
`epsilon`	Convergence parameter

Value

List with the following elements: alpha, betas, s2s (variances)

Author(s)

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

References

See citation("RPA")

Examples

```
#
```

Index