RPA

April 19, 2010

`d.update.fast` *Fast update for probeset-level signal d.*

Description

Weighted average over the columns of S, weighted by the inverse variances 1/sigma2.

Usage

```
d.update.fast(S, sigma2)
```

Arguments

`S`	Matrix of probe-level observations for a single probeset: samples x probes.
`sigma2`	A vector. Estimated variance for each measurement (probe).

Value

A vector giving the estimated true signal underlying the observations in S.

Author(s)

Leo Lahti <leo.lahti@tkk.fi>

References

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

Examples

```
set.seed(24)
vec <- seq(10)
S <- cbind(rnorm(length(vec),sd=1),
  rnorm(length(vec),sd=2),
  rnorm(length(vec),sd=3))
d.update.fast(S,sigma2=seq(3))
```

`initialize.priors` *Initialize prior parameter object for RPA.*

Description

Creates a template for prior parameters using non-informative (or user-specified) priors. After creating the template, probe-specific priors can be set by modifying the default values. Current version supports priors are only for the probe-specific variances (i.e. priors alpha, beta).

Usage

```
initialize.priors(abatch, sets, alpha = 1e-6, beta = 1e-6, d = NULL)
```

Arguments

`abatch`	An AffyBatch object.
`sets`	Vector listing the probesets for which the prior template is created.
`alpha`	Default template values for the alpha prior
`beta`	Default template values for the beta prior
`d`	Not used, included for later compatibility.

Details

Probesets that do not have predefined priors are analyzed using noninformative alpha, beta (1e-6). See the source code for RPA.iteration for details.

Value

An instance of 'rpa.priors' class.

Author(s)

Leo Lahti <leo.lahti@tkk.fi>

References

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

Examples

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

# List probesets to investigate
sets = geneNames(Dilution)[1:5]

# Create a template for prior parameters using default priors
my.priors = initialize.priors(Dilution, sets, alpha = 1e-6, beta = 1e-6)
```

```
# Modify the template to provide user-specified prior for one of the
# probes
# high value implies an unreliable probe
set = sets[[1]]
probe.idx = 5
my.priors[[set]]$beta[[probe.idx]] = 10
my.priors[[set]]$alpha[[probe.idx]] = 10

# Run RPA using the predefined priors
# NOTE: priors are only used with sigma2.method = "basic"
rpa.results <- RPA.pointestimate(Dilution,sets,
         priors=my.priors,
         sigma2.method = "basic",
 d.method = "basic")


# This toy example shows the probe reliability values for the probeset
# where the user-specified prior was set for one of the probes
barplot(rpa.results$sigma2[[set]])
```

`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.

Value

An 'ExpressionSet' object.

Author(s)

Leo Lahti <leo.lahti@tkk.fi>

References

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

Examples

```
require(RPA)
require(affy)
require(affydata)
data(Dilution)
# Compute RPA for specific probesets
sets = geneNames(Dilution)[1:5]
rpa.results <- RPA.pointestimate(Dilution,sets)
# Coerce the rpa object into an ExpressionSet
eset = rpa2eset(rpa.results)
```

---

rpa-class *Class "rpa"*

Description

Class for the RPA package.

Objects from the Class

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

The class instance is a list that contains the following elements:

d A matrix of probesets x arrays. Specifies the estimated 'true' underlying differential gene expression signal over the arrays (vs. the control array 'cind') for each investigated probeset. Note that the control 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.

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

sets A character vector listing the investigated probesets.

Slots

Slots are not utilized here.

Object of class `"list"`

Extends

`.Data`**:** 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@tkk.fi>

References

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

Examples

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

`RPA.iteration` *Finding a mode for the model parameters d and sigma2.*

Description

Finds a mode for the model parameters d (estimated true signal underlying probe-level observations), and sigma2 (probe-specific variances).

Usage

```
RPA.iteration(S, sigma2.guess, epsilon = 10^(-3),
                     alpha = NULL, beta = NULL,
                     sigma2.method = "var", d.method = "fast")
```

Arguments

`S`	Matrix of probe-level observations for a single probeset: samples x probes.
`sigma2.guess`	Initial values for probe-specific variances.
`epsilon`	Convergence tolerance. The iteration is deemed converged when the change in all parameters is < epsilon.
`alpha, beta`	Vectors that specify priors for inverse Gamma distribution of probe-specific variances. Noninformative prior is obtained with alpha, beta -> 0. Used only when sigma2.method = "basic".
`sigma2.method`	Optimization method for sigma2 (probe-specific variances). "basic": optimization using user-specified alpha, beta priors. Default: alpha, beta = 1e-6. "var": utilizes the fact that the cost function converges to variance with large sample sizes. Default method.
`d.method`	Method to optimize d. "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. "fast": weighted mean over the probes, weighted by probe variances The solution converges to this with large sample size. Default method.

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. Note that probe-level variance priors alpha, beta are used only when sigma2.method = "basic". The sigma2.method = "var" assumes non-informative priors. The d.method = "fast" is the preferred method for point computing point estimates when sample size is large. It computes the average over probe-level observations, weighted by the inverse probe-specific variances, and is expected to be more robust and faster than d.method="basic" that finds point estimate for d by directly optimizing the posterior distribution.

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@tkk.fi>

References

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

Examples

```
require(affydata)
data(Dilution) # load affybatch

# select array against which differential expression is computed the
# method has appeared to be robust for the choice of cind
cind <- 1

# Preprocess probe-level data
Smat <- RPA.preprocess(Dilution,cind)

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

# Set initial values for probe-specific variances
sigma2.guess <- rep(10,ncol(S))

# assume non-informative prior and large sample size
# and se fast optimization
res <- RPA.iteration(S, sigma2.guess,
                     epsilon = 10^(-3),
                     sigma2.method = "var",
                     d.method = "fast")

par(mfrow=c(1,2))
```

```
barplot(res$d,names.arg=paste("Array",seq(ncol(exprs(Dilution)))[-cind]),
        main="d",xlab="Arrays/samples",ylab="Differential expression",las=2)

barplot(res$sigma2,las=2,names.arg=paste(set,seq(ncol(S))),
        main="sigma2",xlab="Probes",ylab="Variance",las=2)

# Generate toy data where the ground truth is known
# 300 arrays x 11 probes

set.seed(24)
d.true <- rnorm(300,0,4)
sigma2.true <- c(1,1,1,1,2,2,3,3,3,4,6)
S <- NULL

for (s2 in sigma2.true) {
   S <- cbind(S,rnorm(length(d.true),mean=d.true,sd=sqrt(s2)))
}

# Initial guess for sigma2
sigma2.guess <- rep(1,ncol(S))

# assuming non-informative priors and large sample size
res <- RPA.iteration(S, sigma2.guess,
                     epsilon = 10^(-3),
                     sigma2.method = "var",
                     d.method = "fast")

barplot(res$sigma2,ylab="Estimated var", xlab="Probes",main="sigma2")
```

`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", ...)`.

An extended list containing the following information for one probeset.

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

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

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

set Probeset name.

Slots

Slots are not utilized in this class.

Object of class "list"

Extends

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

Methods

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

Author(s)

Leo Lahti <leo.lahti@tkk.fi>

References

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

Examples

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

RPA-package *RPA: probe reliability and differential expression analysis*

Description

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

Details

Package:	RPA
Type:	Package
Version:	1.1.2
Date:	2009-07-22
License:	GNU GPL 2 or any later version (at your option)
LazyLoad:	yes

RPA.pointestimate computes 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@tkk.fi>

References

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

Examples

```
        ## 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)

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

# Compute RPA for specific probesets only
sets = geneNames(Dilution)[1:5]
rpa.results <- RPA.pointestimate(Dilution,sets)

## Visualize the results for one of the probe sets
plot(rpa.results[sets[[1]],])
```

`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 = 10^(-2), cind = FALSE,
 sigma2.method = "var",
 d.method = "fast",
 verbose = TRUE)
```

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	An 'rpa.priors' object. Can be used to set user-specified priors for the model parameters. Used only with sigma2.method = "basic".
epsilon	Convergence tolerance. The iteration is deemed converged when the change in all parameters is < epsilon.
cind	Specifies which of the arrays in abatch is used as a control in computing probe-level differential expression.
sigma2.method	Optimization method for sigma2 (probe-specific variances). "basic": optimization using user-specified alpha, beta priors. Default: alpha, beta = 1e-6. "var": utilizes the fact that the cost function converges to variance with large sample sizes. Default method.
d.method	Method to optimize d. "basic": Finds a mode by directly optimizing the cost function of the probabilistic model. "fast": weighted mean over the probes, weighted by inverse probe-specific variances. The solution converges to this with large sample size. Default method.
verbose	Print progress information during computation. Default: TRUE.

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. Note that probe-level variance priors alpha, beta are used only when sigma2.method = "basic". The sigma2.method = "var" assumes non-informative priors. The d.method = "fast" is the preferred method for point computing point estimates when sample size is large. It computes the average over probe-level observations, weighted by the inverse probe-specific variances, and is expected to be more robust and faster than d.method="basic" that finds point estimate for d by directly optimizing the posterior distribution.

Value

An instance of the 'rpa' class. 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 control array 'cind') for each investigated probeset. Note that the control 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.
cind	Specifies which of the arrays in the abatch (the affybatch object to be analyzed) was used as a control in computing probe-level differential expression.
sets	A character vector listing the investigated probesets.

Note

With large sample size, sigma2.method="var" and d.method="fast" are recommended. With small sample size and informative prior, sigma2.method="basic" and d.method="basic" may be preferable.

Author(s)

Leo Lahti <leo.lahti@tkk.fi>

References

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

See Also

RPA.iteration, initialize.priors, AffyBatch

Examples

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

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

# Compute RPA for specific probesets only
sets <- geneNames(Dilution)[1:5]
rpa.results <- RPA.pointestimate(Dilution,sets)

# Visualize the results for one of the probe sets
plot(rpa.results[sets[[1]],])
```

RPA.preprocess *Preprocess AffyBatch object for RPA.*

Description

Quantile normalizes and log2-transforms the probe-level raw data in abatch. Then probe-level differential expression is computed between the specified 'control' array (cind) and the other arrays.

Usage

```
RPA.preprocess(abatch, cind)
```

Arguments

abatch	An AffyBatch object.
cind	Specifies which of the arrays in abatch is used as a control in computing probe-level differential expression.

Details

The probe-specific variance estimates are robust against the choice of cind; cind can be selected at random.

Value

`fcmat`	Probes x arrays preprocessed differential expression matrix.
`cind`	Specifies which array in abatch was selected as a control for computing probe-level differential expression.

Author(s)

Leo Lahti <leo.lahti@tkk.fi>

References

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

See Also

AffyBatch

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

Description

Class for defining prior parameters in the RPA package.

Objects from the Class

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

An extended list with elements alpha, beta, d.

alpha A list. Each element is a list that corresponds to one probeset and specified the shape parameters of the inverse Gamma distribution for the probe-specific variance priors.

beta A list. Each element is a list that corresponds to one probeset and specified the scale parameters of the inverse Gamma distribution for the probe-specific variance priors.

d Not implemented. Can be later used to set priors for d.

Slots

`.Data`**:** Object of class `"list"` ~~

Extends

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

Methods

No methods defined with class "rpa.priors" in the signature.

Author(s)

Leo Lahti <leo.lahti@tkk.fi>

References

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

Examples

```
showClass("rpa.priors")
```

Index

∗Topic **classes**
    rpa-class, 4
    rpa.list-class, 7
    rpa.priors-class, 12
∗Topic **iteration**
    d.update.fast, 1
    RPA.iteration, 5
∗Topic **methods**
    d.update.fast, 1
    initialize.priors, 2
    RPA.iteration, 5
    RPA.pointestimate, 9
    RPA.preprocess, 11
    rpa2eset, 3
∗Topic **package**
    RPA-package, 8
[,rpa-method *(rpa-class)*, 4
[[,rpa-method *(rpa-class)*, 4

d.update.fast, 1

initialize.priors, 2

list, *4*, *8*, *12*

plot,rpa.list-method
        (rpa.list-class), 7

RPA *(RPA-package)*, 8
rpa-class, 4
RPA-package, 8
RPA.iteration, 5
rpa.list-class, 7
RPA.pointestimate, 9
RPA.preprocess, 11
rpa.priors-class, 12
rpa2eset, 3

show,rpa-method *(rpa-class)*, 4

vector, *4*, *8*, *12*