affyPara

April 19, 2009

`affyParaIntern` *Internal affyPara objects / functions*

Description

Internal functions for the `affyPara` package.

Usage

```
initAffyBatchSF(object, object.type)

getAffyBatchSF()
getIntensitySF(rows, refindexname)
getCompIntensitySF(rows)
getCompIntensityMatrixSF(rows, drop=FALSE)
getFUNAffyBatchSF(FUN)

writeLinesSF(data, fileName)
ReadHeaderSF(object)

getObjectType(object)
checkPartSize(object, number.parts)
```

Arguments

`object`	An object of class AffyBatch OR a `character` vector with the names of CEL files OR a (partitioned) list of `character` vectors with CEL file names.
`object.type`	Declaration for the type of the argument `object`: "AffyBatch", "CELfileVec", "partCELfileList"
`rows`	Number of rows which have to be changed.
`refindexname`	The name of the array used as a reference.
`drop`	A logical value. If `TRUE` the dimensions of an array which have only one level will be deleted.
`FUN`	A function generating a value from a an AffyBatch. e.g. `dim`
`data`	A `character` vector, containing the data for the file.
`fileName`	A `character` string with the file name.
`number.parts`	Number of nodes in the computer cluster.

Details

Internal functions for the `affyPara` package. The functions have to be used in a cluster function (e.g. clusterApply) from the SNOW package.

`initAffyBatchSF` Slavefunction for initializing an AffyBatch at slaves. AffyBatch will be stored in the .GlobalEnv with the name 'AffyBatch'.

`getAffyBatchSF` Slavefunction to get an AffyBatch from slaves. Gets the object AffyBatch from the .GlobalEnv.

`getIntensitySF` Slavefunction to get special values from the intensity matrix from slaves.

`getCompIntensitySF` Slavefunction to get special rows from the intensity matrix from slaves.

`getCompIntensityMatrixSF` Slavefunction to get complete intensity matrix from slaves.

`getFUNAffyBatchSF` Slavefunction to get a value from an AffyBatch at slaves.

`writeLinesSF` Slavefunction to write data into a file at slaves.

`ReadHeaderSF` Slavefunction to return Header-Informations from CEL Files at slaves.

`getObjectType` Function to get type from object.

`checkPartSize` Function to check object for length.

Author(s)

Markus Schmidberger ⟨schmidb@ibe.med.uni-muenchen.de⟩, Ulrich Mansmann ⟨mansmann@ibe.med.uni-muenchen.de⟩

`bgCorrectPara` *Parallelized Background Correction*

Description

Parallelized functions for background correction of probe intensities.

Usage

```
bgCorrectPara(cluster,
        object, phenoData = new("AnnotatedDataFrame"), method,
        verbose = FALSE)

bgCorrectParaSF(method)
```

Arguments

`cluster`	A cluster object obtained from the function makeCluster in the `SNOW` package.
`object`	An object of class AffyBatch OR a `character` vector with the names of CEL files OR a (partitioned) list of `character` vectors with CEL file names.
`phenoData`	An AnnotatedDataFrame object
`method`	A `character` that defines what background correction method will be used. Available methods are given by `bg.correct.methods`. The name of the method to apply must be double-quoted.
`verbose`	A logical value. If `TRUE` it writes out some messages.

Details

`bgCorrectPara` is the parallelized function for background correction of probe intensities. For serial function an more details see bg.correct.

For using this function a computer cluster using the `snow` package has to be started.

`bgCorrectParaSF` is a internal function which will be executed at slaves.

`bgCorrectParaSF` Calls `bg.correct` at slaves.

Value

An `AffyBatch` for which the intensities have been background adjusted. For some methods (RMA), only PMs are corrected and the MMs remain the same.

Author(s)

Markus Schmidberger ⟨schmidb@ibe.med.uni-muenchen.de⟩, Ulrich Mansmann ⟨mansmann@ibe.med.uni-muenchen.de⟩

Examples

```
## Not run:
library(affyPara)
if (require(affydata)) {
  data(Dilution)

  c1 <- makeCluster(3)

  ##bgc will be the bg corrected version of Dilution
  bgc <- bgCorrectPara(c1, Dilution, method="rma", verbose=TRUE)

  stopCluster(c1)
}
## End(Not run)
```

`computeExprSetPara` *Parallel generate a set of expression values*

Description

Parallel generation of a set of expression values from the probe pair information. The set of expression is returned as an ExpressionSet object.

Usage

```
computeExprSetPara(cluster,
     object,
     ids = NULL,
     pmcorrect.method, summary.method,
     summary.param = list(), pmcorrect.param = list(),
     phenoData = new("AnnotatedDataFrame"), cdfname = NULL,
     verbose = TRUE)
```

Arguments

`cluster`	A cluster object obtained from the function makeCluster in the `SNOW` package.
`object`	An object of class AffyBatch OR a `character` vector with the names of CEL files OR a (partitioned) list of `character` vectors with CEL file names.
`pmcorrect.method`	The name of the PM adjustement method.
`pmcorrect.param`	A list of parameters for `pmcorrect.method` (if needed/wanted).
`summary.method`	The method used for the computation of expression values
`summary.param`	A list of parameters to be passed to the `summary.method` (if wanted).
`ids`	List of `ids` for summarization
`phenoData`	An AnnotatedDataFrame object.
`cdfname`	Used to specify the name of an alternative cdf package. If set to `NULL`, the usual cdf package based on Affymetrix' mappings will be used.
`verbose`	A logical value. If `TRUE` it writes out some messages.

Details

Parallelized preprocessing function, which goes from raw probe intensities to expression values in one steps: summarization

For the serial function and more details see the function `computeExprSet`.

For using this function a computer cluster using the `snow` package has to be started.

Value

An object of class ExpressionSet.

Author(s)

Markus Schmidberger ⟨schmidb@ibe.med.uni-muenchen.de⟩, Ulrich Mansmann ⟨mansmann@ibe.med.uni-muenchen.de⟩

Examples

```
## Not run:
library(affyPara)
if (require(affydata)) {
  data(Dilution)

  c1 <- makeCluster(3)

  esset <- computeExprSetPara(cluster,
      Dilution,
      pmcorrect.method = "pmonly",
      summary.method = "avgdiff",
      verbose = TRUE)

  stopCluster(c1)
}
## End(Not run)
```

`distributeFiles` *Distribute files to slaves*

Description

This function distributes files from the master node to the disk of the slaves in the computer cluster.

Usage

```
distributeFiles(cluster,
        files, to = tempdir(),
        protocol = c("R", "RCP", "SCP"), hierarchicallyDist = FALSE,
        master=TRUE, delExistTo=FALSE,
        full.names=FALSE, verbose = FALSE)

hirarchicalDistSF(to, nodes, protocol)
```

Arguments

`cluster`	A cluster object obtained from the function makeCluster in the `SNOW` package.
`files`	A `character` vector containing the names of the files.
`to`	A `character` that defines the path where the files should be stored at the slaves. Default: tempdir()
`protocol`	A `character` that defines the Copy-Protocol: "R", "RCP", "SCP"
`hierarchicallyDist`	A logical value. If `TRUE` data will be hierarchically distributed to all slaves. If `FALSE` at every slave only a part of data is available.
`master`	A logical value. If `TRUE` all data will be copied to the 'to' directory at the master node. Default = TRUE
`delExistTo`	A logical value. If `TRUE` directory 'to' will be deleted at master and all nodes first. Default = FALSE
`full.names`	A logical value. If `TRUE`, the directory path is prepended to the file names. If `FALSE`, only the file names are returned .
`verbose`	A logical value. If `TRUE` it writes out some messages.
`nodes`	A list of `character` with the node names where the files have to be send.

Details

This function distributes files from the master node to the disk of the slaves in the computer cluster. First the vector of files get partitioned by the number of slaves. Then the parts will be copied to the `to` directory at the slaves. If `hierarchicallyDist` is `TRUE`, all slaves change the files among each other and in the end at every slave all files are located. (But this is not necessary for distributed computing with the `affyPara` package.

For using this function a computer cluster using the `SNOW` package has to be started.

`hirarchicalDistSF` is a internal function which will be executed at slaves.

`hirarchicalDistSF` Slavefunction for exchanging the files between all slaves.

Value

A list of two objects

`to`	A `character` that defines the path where the files are located at the slaves.
`CELfiles`	A list of `characters`, how the files are distributed to the slaves.

Warning

For protocol "R" hierarchically distribution not yet available.

Author(s)

Markus Schmidberger ⟨schmidb@ibe.med.uni-muenchen.de⟩, Ulrich Mansmann ⟨mansmann@ibe.med.uni-muenchen.de⟩

Examples

```
## Not run:
library(affyPara)

c1 <- makeCluster(10)

path <- "tmp/CELfiles"
CELfiles <- list.files(path,full.names=TRUE)

distList <- distributeFiles(c1, CELfiles, protocol="RCP", verbose=TRUE)

stopCluster(c1)
## End(Not run)
```

`mergeSplitObjects` *Merge a list of split objects*

Description

Functions to merge or combine a `list` of split objects (AffyBatch, Matrix).

Usage

```
mergeAffyBatches(abatch.list, description = NULL, notes = character(0))
combineMatrices(matrix.list, verbose=FALSE)
```

Arguments

`abatch.list`	A `list` of objects of class AffyBatch.
`description`	A MIAME object.
`notes`	A `character` vector of explanatory text.
`matrix.list`	A `list` of objects of class matrix.
`verbose`	A logical value. If `TRUE` it writes out some messages.

Details

Functions to merge or combine a `list` of split objects.

`mergeAffyBatches` Merges a `list` of AffyBatches to one AffyBatch.

`combineMatrices` Combines a `list` of matrices by columns to one matrix.

Value

`this-is-escaped-codenormal-bracket36bracket-normal`
Returns ONE object of class AffyBatch.

`this-is-escaped-codenormal-bracket40bracket-normal`
Returns ONE object of class matrix.

Author(s)

Markus Schmidberger ⟨schmidb@ibe.med.uni-muenchen.de⟩, Ulrich Mansmann ⟨mansmann@ibe.med.uni-muenchen.de⟩

Examples

```
library(affyPara)
if (require(affydata)) {
  data(Dilution)

  #split AffyBatch
  abatch.list<- splitAffyBatch(Dilution, 2)

  #Merge AffyBatch
  AffyBatch <- mergeAffyBatches(abatch.list)

  # Create matrices
  a <- matrix(1:25, nrow=5)
  b <- matrix(101:125, nrow=5)
  matrix.list <- list(a,b)

  # Combine matrices
  combineMatrices(matrix.list)
}
```

`normalizeAffyBatchConstantPara`
Parallelized scaling normalization

Description

Parallelized scaling normalization of arrays.

Usage

```
normalizeAffyBatchConstantPara(cluster,
      object,
      refindex = 1, FUN = mean, na.rm = TRUE,
      phenoData = new("AnnotatedDataFrame"), cdfname = NULL,
      verbose = FALSE)

normalizeConstantParaSF1(refindexname)
normalizeConstantParaSF2(refconstant, refindexname, FUN = mean, na.rm = TRUE)
```

Arguments

`cluster`	A cluster object obtained from the function makeCluster in the `SNOW` package.
`object`	An object of class AffyBatch OR a `character` vector with the names of CEL files OR a (partitioned) list of `character` vectors with CEL file names.
`refindex`	The index of the array used as reference.
`FUN`	A function generating a value from the intensities on an array. Typically `mean` or `median`.
`na.rm`	Paramater passed to the function FUN. A logical value indicating whether `NA` values should be stripped before the computation proceeds.
`phenoData`	An AnnotatedDataFrame object.
`cdfname`	Used to specify the name of an alternative cdf package. If set to `NULL`, the usual cdf package based on Affymetrix' mappings will be used.
`verbose`	A logical value. If `TRUE` it writes out some messages.
`refindexname`	The name of the array used as reference.
`refconstant`	The constant calculated by `FUN` from the reference array. This value will be used for scaling.

Details

Parallelized scaling normalization of arrays. This means that all the array are scaled so that they have the same mean value.

For the serial function and more details see the function `normalize.constant`.

For using this function a computer cluster using the `snow` package has to be started.

`normalizeConstantParaSF1` and `normalizeConstantParaSF2` are internal function which will be executed at all slaves.

`normalizeConstantParaSF1` Calculates and returns intensities from the refindex array.

`normalizeConstantParaSF2` Normalizes the arrays at all slaves.

Value

An AffyBatch of normalized objects.

Author(s)

Markus Schmidberger ⟨schmidb@ibe.med.uni-muenchen.de⟩, Ulrich Mansmann ⟨mansmann@ibe.med.uni-muenchen.de⟩

Examples

```
## Not run:
library(affyPara)
if (require(affydata)) {
  data(Dilution)

  c1 <- makeCluster(3)

  AffyBatch <- normalizeAffyBatchConstantPara(c1, Dilution, verbose=TRUE)

  stopCluster(c1)
}
## End(Not run)
```

`normalizeAffyBatchInvariantsetPara`
Parallelized Invariante Set normalization

Description

Parallelized normalization of arrays using an invariant set.

Usage

```
normalizeAffyBatchInvariantsetPara(cluster,
    object,
    prd.td = c(0.003, 0.007), baseline.type = c("mean", "median", "pseudo-mean",
    type = c("separate", "pmonly", "mmonly", "together"),
    phenoData = new("AnnotatedDataFrame"), cdfname = NULL,
    verbose = FALSE)

    normalizeInvariantsetParaSF1(FUN, type)
    normalizeInvariantsetParaSF2(refindexname, rows, prd.td, baseline.chip)
```

Arguments

`cluster`	A cluster object obtained from the function makeCluster in the `SNOW` package.
`object`	An object of class AffyBatch OR a `character` vector with the names of CEL files OR a (partitioned) list of `character` vectors with CEL file names.
`prd.td`	A cutoff parameter for normalization.
`baseline.type`	Specify how to determine the baseline array (mean, median).
`type`	A string specifying how the normalization should be applied.
`phenoData`	A AnnotatedDataFrame object.
`cdfname`	Used to specify the name of an alternative cdf package. If set to `NULL`, the usual cdf package based on Affymetrix' mappings will be used.
`verbose`	A logical value. If `TRUE` it writes out some messages.
`FUN`	A function generating a value from the intensities on an array. Typically mean or median.

`refindexname`	The name of the array used as reference.
`rows`	Number of rows which have to be changed.
`baseline.chip`	The values of the array used as reference.

Details

Parallelized normalization of arrays using an invariant set. The set of invariant intensities between data and ref is found through an iterative process (based on the respective ranks the intensities). This set of intensities is used to generate a normalization curve by smoothing.

For the serial function and more details see the function `normalize.invariantset`.

For using this function a computer cluster using the `snow` package has to be started.

`normalizeInvariantsetParaSF1` and `normalizeInvariantsetParaSF2` are internal function which will be executed at all slaves.

`normalizeInvariantsetParaSF1`	Calculates and returns a value from the intensities of an array.
`normalizeInvariantsetParaSF2`	Normalizes the arrays at slaves.

Value

An AffyBatch of normalized objects.

Author(s)

Markus Schmidberger ⟨schmidb@ibe.med.uni-muenchen.de⟩, Ulrich Mansmann ⟨mansmann@ibe.med.uni-muenchen.de⟩

Examples

```
## Not run:
library(affyPara)
if (require(affydata)) {
  data(Dilution)

  c1 <- makeCluster(3)

  AffyBatch <- normalizeAffyBatchInvariantsetPara(c1, Dilution, verbose=TRUE)

  stopCluster(c1)
}
## End(Not run)
```

`normalizeAffyBatchLoessPara` *Parallelized loess normalization*

Description

Parallelized loess normalization of arrays.

Usage

```
normalizeAffyBatchLoessPara(cluster,
        object,
        phenoData = new("AnnotatedDataFrame"), cdfname = NULL,
        type=c("separate","pmonly","mmonly","together"),
        subset = NULL,
        epsilon = 10^-2, maxit = 1, log.it = TRUE,
        span = 2/3, family.loess ="symmetric",
        verbose=FALSE)
```

Arguments

`cluster`	A cluster object obtained from the function makeCluster in the `SNOW` package.
`object`	An object of class AffyBatch OR a `character` vector with the names of CEL files OR a (partitioned) list of `character` vectors with CEL file names.
`phenoData`	An AnnotatedDataFrame object.
`cdfname`	Used to specify the name of an alternative cdf package. If set to `NULL`, the usual cdf package based on Affymetrix' mappings will be used.
`type`	A string specifying how the normalization should be applied.
`subset`	a subset of the data to fit a loess to.
`epsilon`	a tolerance value (supposed to be a small value - used as a stopping criterium).
`maxit`	maximum number of iterations.
`log.it`	logical. If `TRUE` it takes the log2 of mat
`span`	parameter to be passed the function loess
`family.loess`	parameter to be passed the function loess. "gaussian" or "symmetric" are acceptable values for this parameter.
`verbose`	A logical value. If `TRUE` it writes out some messages.

Details

Parallelized loess normalization of arrays.

For the serial function and more details see the function `normalize.AffyBatch.loess`.

For using this function a computer cluster using the `snow` package has to be started. In the loess normalization the arrays will compared by pairs. Therefore at every node minimum two arrays have to be!

Value

An AffyBatch of normalized objects.

Author(s)

Markus Schmidberger ⟨schmidb@ibe.med.uni-muenchen.de⟩, Ulrich Mansmann ⟨mansmann@ibe.med.uni-muenchen.de⟩

Examples

```
## Not run:
library(affyPara)
if (require(affydata)) {
  data(Dilution)

  c1 <- makeCluster(3)

  AffyBatch <- normalizeAffyBatchLoessPara(c1, Dilution, verbose=TRUE)

  stopCluster(c1)
}
## End(Not run)
```

`normalizeAffyBatchLoessIterPara`

Parallelized partial loess normalization with permutation

Description

Parallelized partial cyclic loess normalization of arrays with permutation.

Usage

```
normalizeAffyBatchLoessIterPara(cluster,
                  object,
                  percentPerm = 0.75,
                  phenoData = new("AnnotatedDataFrame"), cdfname = NULL,
                  type=c("separate","pmonly","mmonly","together"),
                  subset = NULL,
                  epsilon = 10^-2, maxit = 1, log.it = TRUE,
                  span = 2/3, family.loess ="symmetric",
                  verbose=FALSE)
```

Arguments

`cluster`	A cluster object obtained from the function makeCluster in the `SNOW` package.
`object`	An object of class AffyBatch OR a `character` vector with the names of CEL files OR a (partitioned) list of `character` vectors with CEL file names.
`percentPerm`	Percent of permutations to do.
`phenoData`	An AnnotatedDataFrame object.
`cdfname`	Used to specify the name of an alternative cdf package. If set to `NULL`, the usual cdf package based on Affymetrix' mappings will be used.
`type`	A string specifying how the normalization should be applied.
`subset`	a subset of the data to fit a loess to.
`epsilon`	a tolerance value (supposed to be a small value - used as a stopping criterium).
`maxit`	maximum number of iterations.
`log.it`	logical. If `TRUE` it takes the log2 of mat

`span`	parameter to be passed the function loess
`family.loess`	parameter to be passed the function loess. "gaussian" or "symmetric" are acceptable values for this parameter.
`verbose`	A logical value. If `TRUE` it writes out some messages.

Details

Parallelized partial cyclic loess normalization of arrays with permutation. This is a new kind of normalization based on cyclic loess normalization.

In the partial cyclic loess normalization the loess normalization will be done only at the slaves with the arrays at the slaves. Therefore we only have to do loess normalization for some pairs and have a big saving of time. But this is no enough for good normalization. We have to do some interations of array permutation between the slaves and again loess normalization at the slaves. If we did about 75 percent of the complete cyclic loess normalization we can achieve same results and save computation time.

For the similar serial function and more details to loess normalization see the function `normalize.AffyBatch.loes`

For using this function a computer cluster using the `snow` package has to be started. In the loess normalization the arrays will compared by pairs. Therefore at every node minimum two arrays have to be!

Value

An AffyBatch of normalized objects.

Author(s)

Markus Schmidberger ⟨schmidb@ibe.med.uni-muenchen.de⟩, Ulrich Mansmann ⟨mansmann@ibe.med.uni-muenchen.de⟩

Examples

```
## Not run:
library(affyPara)
if (require(affydata)) {
  data(Dilution)

  c1 <- makeCluster(3)

  AffyBatch <- normalizeAffyBatchLoessIterPara(c1, percentPerm=0.75, Dilution, verbose=TR

  stopCluster(c1)
}
## End(Not run)
```

`normalizeAffyBatchQuantilesPara`
Parallelized quantile normalization

Description

Parallelized normalization of arrays based upon quantiles.

Usage

```
normalizeAffyBatchQuantilesPara(cluster,
    object,
    phenoData = new("AnnotatedDataFrame"), cdfname = NULL,
    type = c("separate", "pmonly", "mmonly", "together"), verbose = FALSE)

normalizeQuantilesPara(cluster, type, object.length, verbose=FALSE)
normalizeQuantilesParaSF1(type)
normalizeQuantilesParaSF2(row_mean)
```

Arguments

`cluster`	A cluster object obtained from the function makeCluster in the `SNOW` package.
`object`	An object of class AffyBatch OR a `character` vector with the names of CEL files OR a (partitioned) list of `character` vectors with CEL file names.
`phenoData`	An AnnotatedDataFrame object.
`cdfname`	Used to specify the name of an alternative cdf package. If set to `NULL`, the usual cdf package based on Affymetrix' mappings will be used.
`type`	A string specifying how the normalization should be applied.
`verbose`	A logical value. If `TRUE` it writes out some messages.
`object.length`	Number of samples, which should be normalized.
`row_mean`	Row mean used for normalization.

Details

Parallelized normalization of arrays based upon quantiles. This method is based upon the concept of a quantile-quantile plot extended to `n` dimensions. No special allowances are made for outliers.

For the serial function and more details see the function `normalize.AffyBatch.quantiles`.

For using this function a computer cluster using the `snow` package has to be started.

`normalizeQuantilesPara`, `normalizeQuantilesParaSF1` and `normalizeQuantilesParaSF2` are internal function which will be executed at all slaves.

`normalizeQuantilesPara` Function for quantil normalization.

`normalizeQuantilesParaSF1` Slavefuntion to calculate row means.

`normalizeQuantilesParaSF2` Slavefunction to do normalization.

Value

An AffyBatch of normalized objects.

Author(s)

Markus Schmidberger ⟨schmidb@ibe.med.uni-muenchen.de⟩, Ulrich Mansmann ⟨mansmann@ibe.med.uni-muenchen.de⟩

Examples

```
## Not run:
library(affyPara)
if (require(affydata)) {
  data(Dilution)

  c1 <- makeCluster(3)

  AffyBatch <- normalizeAffyBatchQuantilesPara(c1, Dilution, verbose=TRUE)

  stopCluster(c1)
}
## End(Not run)
```

`preproPara` *Parallelized preprocessing*

Description

Parallelized preprocessing function, which goes from raw probe intensities to expression values in three steps: Background correction, normalization and summarization

Usage

```
preproPara(cluster,
    object,
    bgcorrect = TRUE, bgcorrect.method = NULL, bgcorrect.param = list(),
    normalize = TRUE, normalize.method = NULL, normalize.param = list(),
    pmcorrect.method = NULL, pmcorrect.param = list(),
    summary.method = NULL, summary.param = list(),
    ids = NULL,phenoData = new("AnnotatedDataFrame"), cdfname = NULL,
    verbose = FALSE)
```

Arguments

`cluster`	A cluster object obtained from the function makeCluster in the `SNOW` package.
`object`	An object of class AffyBatch OR a `character` vector with the names of CEL files OR a (partitioned) list of `character` vectors with CEL file names.
`bgcorrect`	A boolean to express whether background correction is wanted or not.
`bgcorrect.method`	The name of the background adjustment method to use.
`bgcorrect.param`	A list of parameters for `bgcorrect.method` (if needed/wanted)
`normalize`	A boolean to express whether normalization is wanted or not.
`normalize.method`	The name of the normalization method to use.
`normalize.param`	A list of parameters to be passed to the normalization method (if wanted).
`pmcorrect.method`	The name of the PM adjustement method.

`pmcorrect.param`	A list of parameters for `pmcorrect.method` (if needed/wanted).
`summary.method`	The method used for the computation of expression values
`summary.param`	A list of parameters to be passed to the `summary.method` (if wanted).
`ids`	List of `ids` for summarization
`phenoData`	An AnnotatedDataFrame object.
`cdfname`	Used to specify the name of an alternative cdf package. If set to `NULL`, the usual cdf package based on Affymetrix' mappings will be used.
`verbose`	A logical value. If `TRUE` it writes out some messages.

Details

Parallelized preprocessing function, which goes from raw probe intensities to expression values in three steps: Background correction, normalization and summarization

For the serial function and more details see the function `expresso`.

For using this function a computer cluster using the `snow` package has to be started.

Available methods:

`bgcorrect.method`: see `bgcorrect.methods`

`normalize.method`: 'quantil', 'constant', 'invariantset','loess'

`summary.method`: see `generateExprSet.methods`

Value

An object of class ExpressionSet.

Author(s)

Markus Schmidberger ⟨schmidb@ibe.med.uni-muenchen.de⟩, Ulrich Mansmann ⟨mansmann@ibe.med.uni-muenchen.de⟩

Examples

```
## Not run:
library(affyPara)
if (require(affydata)) {
  data(Dilution)

  c1 <- makeCluster(3)

  esset <- preproPara(cluster,
      Dilution,
      bgcorrect = TRUE, bgcorrect.method = "rma2",
      normalize = TRUE, normalize.method = "quantil",
      pmcorrect.method = "pmonly",
      summary.method = "avgdiff",
      verbose = TRUE)

  stopCluster(c1)
}
## End(Not run)
```

`removeDistributedFiles` *Remove distributed files from slaves*

Description

This function removes distributed files from a special path at the disk at all slaves in a computer cluster.

Usage

```
removeDistributedFiles(cluster, path=tempdir(), verbose = FALSE)
```

Arguments

`cluster`	A cluster object obtained from the function makeCluster in the `SNOW` package.
`path`	A `character` that defines which path (inclusive files) should be removed at every slave. Default: tempdir()
`verbose`	A logical value. If `TRUE` it writes out some messages.

Details

This function removes distributed files from a special path at the disk at all slaves in a computer cluster.

For using this function a computer cluster using the `SNOW` package has to be started.

Value

If `verbose = TRUE`, result of removing (successfully / not successfully) will be noticed with a message.

Author(s)

Markus Schmidberger ⟨schmidb@ibe.med.uni-muenchen.de⟩, Ulrich Mansmann ⟨mansmann@ibe.med.uni-muenchen.de⟩

Examples

```
## Not run:
library(affyPara)

c1 <- makeCluster(10)

removeDistributedFiles(c1, verbose=TRUE)

stopCluster(c1)
## End(Not run)
```

rmaPara *Parallelized PMA preprocessing*

Description

Parallelized preprocessing function, which converts an AffyBatch into an ExpressionSet using the robust multi-array average (RMA) expression measure.

Usage

```
rmaPara(cluster,
        object,
        ids = NULL,
        phenoData = new("AnnotatedDataFrame"), cdfname = NULL,
        verbose=FALSE)
```

Arguments

`cluster`	A cluster object obtained from the function makeCluster in the `SNOW` package.
`object`	An object of class AffyBatch OR a `character` vector with the names of CEL files OR a (partitioned) list of `character` vectors with CEL file names.
`ids`	List of `ids` for summarization
`phenoData`	An AnnotatedDataFrame object.
`cdfname`	Used to specify the name of an alternative cdf package. If set to `NULL`, the usual cdf package based on Affymetrix' mappings will be used.
`verbose`	A logical value. If `TRUE` it writes out some messages.

Details

Parallelized preprocessing function, which goes from raw probe intensities to expression values using the robust multi-array average (RMA) expression measure: Background correction: rma; Normalization: quantile; Summarization: medianpolish

For the serial function and more details see the function `rma`.

For using this function a computer cluster using the `snow` package has to be started.

This is a wrapper function for `preproPara`.

Value

An object of class ExpressionSet.

Author(s)

Markus Schmidberger ⟨schmidb@ibe.med.uni-muenchen.de⟩, Ulrich Mansmann ⟨mansmann@ibe.med.uni-muenchen.de⟩

Examples

```
## Not run:
library(affyPara)
if (require(affydata)) {
  data(Dilution)

  c1 <- makeCluster(3)

  esset <- rmaPara(cluster, Dilution)

  stopCluster(c1)
}
## End(Not run)
```

`splitObjects` *Functions to split objects into parts*

Description

Functions to split an AffyBatch, a `list` of files and a `matrix` into several objects for distributed computing. If possible objects will be of the same size.

Usage

```
splitAffyBatch(abatch, number.part)
splitFileVector(fileVec, number.part)
splitMatrix(matrix, number.part)
```

Arguments

`abatch`	An object of class AffyBatch.
`fileVec`	A `character` vector containing the names of the files.
`matrix`	An object of class matrix.
`number.part`	Number of parts to split the object

Details

`splitAffyBatch` Splits an AffyBatch into a list of AffyBatches.

`splitFileVector` Splits a `character` vector of file names into a list of `character` vectors with file names.

`splitMatrix` Splits a `matrix` by columns into a `list` of matrices.

These functions use the functions splitIndices and splitCols from the `SNOW` package.

Value

A `list` of the split objects.

Author(s)

Markus Schmidberger ⟨schmidb@ibe.med.uni-muenchen.de⟩, Ulrich Mansmann ⟨mansmann@ibe.med.uni-muenchen.de⟩

Examples

```
library(affyPara)
if (require(affydata)) {
  data(Dilution)

  spAffyB <- splitAffyBatch(Dilution, 2)
}
```

Index