Package ‘cn.farms’

May 29, 2024

Title cn.FARMS - factor analysis for copy number estimation
Version 1.52.0
Date 2020-03-09
Type Package
License LGPL (>= 2.0)
Author Andreas Mitterecker, Djork-Arne Clevert
Maintainer Andreas Mitterecker <mitterecker@ml.jku.at>
Description This package implements the cn.FARMS algorithm for copy number variation (CNV) analysis. cn.FARMS allows to analyze the most common Affymetrix (250K-SNP6.0) array types, supports high-performance computing using snow and ff.
URL http://www.bioinf.jku.at/software/cnfarms/cnfarms.html
Depends R (>= 3.0), Biobase, methods, ff, oligoClasses, snow
Imports DBI, affxparser, oligo, DNAcopy, preprocessCore, lattice
Suggests pd.mapping250k.sty, pd.mapping250k.nsp, pd.genomewidesnp.5, pd.genomewidesnp.6
Collate 'callSummarize.R' 'combineData.R' 'correctPkgname.R'
'cnFarms.R' 'createAnnotation.R' 'createMatrix.R'
'determineBaselineArray.R' 'distributionDistance.R'
'dnaCopySt.R' 'doCnFarms.R' 'fragLengthCorr.R' 'normAdd.R'
'normalizeAverage.R' 'normalizeCels.R' 'normalizeNpData.R'
'normalizeQuantiles.R' 'normalizeSor.R' 'plotDendrogram.R'
'plotDensity.R' 'plotEvalClc.R' 'plotSmoothScatter.R'
'plotsRegions.R' 'plotViolines.R' 'sparseFarmsC.R'
'summarizationML.R' 'summarizationSl.R'
'summarizeFarmsGaussian.R' 'summarizeFarmsLaplaceExact.R'
'summarizeFarmsLaplaceVar.R' 'summarizeFarmsMethods.R'
'summarizeStatistics.R' 'windowFunctions.R' 'windowMethods.R'
'normalizeProbeSequence.R' 'snowfallExt.R'
'summarizeFarmsLaplaceExact2.R' 'summarizeFarmsLaplaceExact3.R'
'normalizeNone.R' 'utils-llds.R' 'zzz.R' 'sFclusterFunctions.R'
'sFinit.R' 'sFsnowfall-internal.R' 'sFsnowWrappers.R'
'sFsocketRequest.R' 'vanillaIce.R'

1
**Contents**

- `callSummarize` ................................................................. 3
- `cn.farms` ............................................................................. 4
- `cnLibrary` ............................................................................ 4
- `combineData` ........................................................................ 5
- `createAnnotation` ................................................................. 6
- `createMatrix` ........................................................................ 7
- `distributionDistance` ............................................................ 8
- `dnaCopySf` ............................................................................ 9
- `doCnFarmsSingle` ................................................................. 10
- `flcSnp6Std` .......................................................................... 10
- `flcStd` ................................................................................. 11
- `fragLengCorr` ....................................................................... 12
- `getFragmentSet` ................................................................... 13
- `getSingleProbeSetSize` ......................................................... 13
- `mlSummarization` ................................................................. 14
- `normAdd` ............................................................................. 15
- `normalizeAverage` ............................................................... 15
- `normalizeCels` ..................................................................... 16
- `normalizeNone` .................................................................... 17
- `normalizeNpData` ................................................................. 18
- `normalizeQuantiles` ............................................................. 19
- `normalizeSequenceEffect` ..................................................... 19
- `normalizeSor` ....................................................................... 20
- `plotDendrogram` ................................................................. 21
- `plotDensity` ......................................................................... 21
- `plotEvalIc` .......................................................................... 22
- `plotRegions` ........................................................................ 23
- `plotSmoothScatter` ............................................................... 24
- `plotViolines` ....................................................................... 25
- `slSummarization` ................................................................. 26
- `sparseFarmsC` ...................................................................... 27
- `summarizeFarmsExact` ......................................................... 28
- `summarizeFarmsExact2` ....................................................... 29
- `summarizeFarmsExact3` ........................................................ 31
**callSummarize**

Defines which variables should be written back when calling a cn.farms run

**Description**

Defines which variables should be written back when calling a cn.farms run

**Usage**

```r
callSummarize(object, psInfo, summaryMethod, summaryParam, batchList = NULL,
  cores = 1, runtype = "ff", returnValues, saveFile = "summData")
```

**Arguments**

- **object**: an matrix with normalized intensity values.
- **psInfo**: a data frame stating the physical position.
- **summaryMethod**: the summarization method.
- **summaryParam**: a list with the parameters of the summarization method.
- **batchList**: batchList
- **cores**: cores
- **runtype**: mode how the results are saved. Possible values are ff or bm. If ff is chosen the data will not be saved automatically. With bm the results will be saved permanently.
- **returnValues**: list with return values. For possible values see summaryMethod.
- **saveFile**: name of the file to save.

**Value**

Results of FARMS run with specified parameters - exact FARMS version

**Author(s)**

Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>
Description

Wrapper for the cn.farms algorithm

Usage

```
cn.farms(filenames, cores = 1, runtype = "bm")
```

Arguments

- `filenames`: the absolute filepaths of the CEL files.
- `cores`: number of parallel instances.
- `runtype`: either ff or bm.

Value

An instance of `ExpressionSet` containing the results of the analysis.

Author(s)

Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>

Examples

```r
## Not run:
require('hapmapsnp6')
celDir <- system.file('celFiles', package = 'hapmapsnp6')
filenames <- dir(path = celDir, full.names = TRUE)
cn.farms(filenames = filenames)

## End(Not run)
```

Description

This function was taken from snowfall and edited due to some deprecated function calls.
Usage

cnLibrary(package, pos = 2, lib.loc = NULL, character.only = FALSE,
    warn.conflicts = TRUE, keep.source = getOption("keep.source.pkgs"),
    verbose = getOption("verbose"), version, stopOnError = TRUE)

Arguments

package name of the package. Check 'library' for details.
pos position in search path to load library.
lib.loc a character vector describing the location of the R library trees to search through,
    or 'NULL'. Check 'library' for details.
character.only a logical indicating package can be assumed to be a character string. Check
    'library' for details.
warn.conflicts warn on conflicts (see "library").
keep.source DEPRECATED (see "library").
verbose enable verbose messages.
version version of library to load (see "library").
stopOnError logical.

Value

for more information see "library".

Author(s)

xxx

---

combineData Combine two ExpressionSet objects

Description

Suitable for SNP or non-polymorphic data which were already processed with single locus FARMS

Usage

combineData(object01, object02, obj01Var = "intensity",
            obj02Var = "intensity", runtype = "ff", saveFile = "combData")
createAnnotation

**Arguments**

- **object01**: An instance of `ExpressionSet` either with SNP or non-polymorphic data
- **object02**: An instance of `ExpressionSet` either with SNP or non-polymorphic data
- **obj01Var**: States the variable which should be combined from the assayData slot. Default is intensity.
- **obj02Var**: States the variable which should be combined from the assayData slot. Default is intensity.
- **runtype**: Mode how the results are saved. Possible values are ff or bm. If ff is chosen the data will not be saved automatically. With bm the results will be saved permanently.
- **saveFile**: Name of the file to save.

**Value**

An instance of `ExpressionSet`.

**Author(s)**

Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>

**Examples**

```r
load(system.file("exampleData/normData.RData", package = "cn.farms"))
notes(experimentData(normData))$annotDir <-
  system.file("exampleData/annotation/pd.genomewidesnp.6/1.1.0",
    package = "cn.farms")
summaryMethod <- "Variational"
summaryParam <- list()
summaryParam$cyc <- c(10)
slData <- slSummarization(normData,
    summaryMethod = summaryMethod,
    summaryParam = summaryParam)
assayData(slData)$L_z[1:10, ]
combData <- combineData(slData, slData)
combData
```

**Description**

Annotation files for cn.farms are created

**Usage**

```r
createAnnotation(filenames = NULL, annotation = NULL, annotDir = NULL,
    checks = TRUE)
```
createMatrix

Arguments

- filenames: An absolute path of the CEL files to process.
- annotation: Optional parameter stating the annotation from a pd-mapping.
- annotDir: Optional parameter stating where the annotation should go.
- checks: States if sanity checks should be done.

Value

NULL

Note

The annotation files used for cn.farms will be placed in the current work directory under annotations.

Author(s)

Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>

Examples

```r
## Not run:
library("hapmapsnp6")
celDir <- system.file("celFiles", package = "hapmapsnp6")
filenames <- dir(path = celDir, full.names = TRUE)
createAnnotation(filenames = filenames)
## End(Not run)
```

createMatrix  

Creates the needed matrix

Description

Creates the needed matrix

Usage

```r
createMatrix(runtype, nrow, ncol, type = "double", bmName = "NA")
```

Arguments

- runtype: Mode how the results are saved. Possible values are ff or bm. If ff is chosen the data will not be saved automatically. With bm the results will be saved permanently.
- nrow: nrow
- ncol: ncol
- type: type
- bmName: Identifier for ff name
distributionDistance

Value

a matrix

Author(s)

Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>

distributionDistance  Computes the distribution distance

Description

Be aware that this function is implemented quite slow.

Usage

distributionDistance(intensityData, method = c("JSDiv", "KLDiv", "KLInf"),
useSubset = T, subsetFraction = 0.25, useQuantileReference = FALSE)

Arguments

intensityData  A matrix or an AffyBatch object.
method  The method you want to use.
useSubset  Logical. States if only a subset should be used.
subsetFraction  The fraction of the subset.
useQuantileReference  Logical for a quantile reference.

Value

Computes the distribution distance

Author(s)

Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>

Examples

load(system.file("exampleData/normData.RData", package = "cn.farms"))
x <- assayData(normData)$intensity[, 1:3]
y <- distributionDistance(x)
attr(y, "Labels") <- substr(sampleNames(normData), 1, 7)
plotDendrogram(y)
Runs DNAcopy in parallel mode

Description

This function even works very well with ff matrices.

Usage

dnaCopySf(x, chrom, maploc, cores = 1, smoothing, ...)

Arguments

x A matrix with data of the copy number experiments
chrom The chromosomes (or other group identifier) from which the markers came
maploc The locations of marker on the genome
cores Number of cores to use
smoothing States if smoothing of the data should be done
... Further parameter for the function segment of DNAcopy

Value

An instance of ExpressionSet containing the segments.

Author(s)

Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>

Examples

```r
load(system.file("exampleData/mlData.RData", package = "cn.farms"))
mlData <- mlData[, 1:3]
colnames(assayData(mlData)$L_z) <- sampleNames(mlData)
segments <- dnaCopySf(
  x = assayData(mlData)$L_z,
  chrom = fData(mlData)$chrom,
  maploc = fData(mlData)$start,
  cores = 1,
  smoothing = FALSE)
fData(segments)
```
**doCnFarmsSingle**

*Does the whole cn.farms process in one call*

**Description**

Works for all kind of Affymetrix SNP arrays

**Usage**

```
doCnFarmsSingle(celfiles, samplenames, normalization)
```

**Arguments**

- `celfiles`: The celfiles which you want to process with the whole path. Either a vector or a matrix with two columns for combined analysis e.g. 500K Array.
- `samplenames`: An optional vector with the same dimension as the number of cel files
- `normalization`: The normalization method you want to use.

**Value**

The ready cn.FARMS results.

**Author(s)**

Andreas Mitterecker

---

**flcSnsp6Std**

*Does a fragment length correction on intensities*

**Description**

Does a fragment length correction on intensities

**Usage**

```
flcSnsp6Std(y, fragmentLengths, targetFcn = NULL, subsetToFit = NULL, runtype = "ff", cores = 1, saveFile = "flc", ...)
```
**flcStd**  

*Does a fragment length correction on intensities*

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>fragmentLengths</td>
<td>fragmentLengths</td>
</tr>
<tr>
<td>targetFcn</td>
<td>targetFcn</td>
</tr>
<tr>
<td>subsetToFit</td>
<td>subsetToFit</td>
</tr>
<tr>
<td>runtype</td>
<td>Mode how the results are saved. Possible values are ff or bm. If ff is chosen the data will not be saved automatically. With bm the results will be saved permanently.</td>
</tr>
<tr>
<td>cores</td>
<td>cores</td>
</tr>
<tr>
<td>saveFile</td>
<td>Name of the file to save.</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Value**

data frame

**Author(s)**

Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>

---

**Description**

Does a fragment length correction on intensities

**Usage**

```r
flcStd(y, fragmentLengths, targetFcn = NULL, subsetToFit = NULL, runtype = "ff", cores = 1, saveFile = "flc", ...)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>fragmentLengths</td>
<td>fragmentLengths</td>
</tr>
<tr>
<td>targetFcn</td>
<td>targetFcn</td>
</tr>
<tr>
<td>subsetToFit</td>
<td>subsetToFit</td>
</tr>
<tr>
<td>runtype</td>
<td>Mode how the results are saved. Possible values are ff or bm. If ff is chosen the data will not be saved automatically. With bm the results will be saved permanently.</td>
</tr>
<tr>
<td>cores</td>
<td>cores</td>
</tr>
<tr>
<td>saveFile</td>
<td>Name of the file to save.</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
**fragLengCorr**

**Value**

data frame

**Author(s)**

Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>

---

**fragLengCorr** *Does a fragment length correction*

---

**Description**

Does a fragment length correction

**Usage**

`fragLengCorr(object, runtype = "ff", saveFile = "slDataFlc", ...)`

**Arguments**

- `object` An instance of `ExpressionSet`
- `runtype` Mode how the results are saved. Possible values are `ff` or `bm`.
- `...` Further parameters passed to the correction method.
- `saveFile` Name of the file to save.

**Value**

An instance of `ExpressionSet`.

**Author(s)**

Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>

**Examples**

```r
load(system.file("exampleData/slData.RData", package = "cn.farms"))
slDataFlc <- fragLengCorr(slData)
```
**getFragmentSet**

*Finds SNPs which belong to one fragment*

**Description**

Finds SNPs which belong to one fragment

**Usage**

```plaintext
getFragmentSet(fragLength)
```

**Arguments**

- `fragLength`

**Value**

Windows for fragments

**Author(s)**

Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>

---

**getSingleProbesetSize**

*Combines data for probeset summarization*

**Description**

Combines data for probeset summarization

**Usage**

```plaintext
getSingleProbesetSize(fsetid)
```

**Arguments**

- `fsetid`

**Value**

A indices which are used for probeset summarization

**Author(s)**

Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>
mlSummarization

Method for computation of the multi-loci summarization

Description

Method for computation of the multi-loci summarization

Usage

mlSummarization(object, windowMethod, windowParam, summaryMethod, summaryParam, callParam = list(runtype = "ff"), returnValues, saveFile = "mlData")

Arguments

object an instance of ExpressionSet
windowMethod Method for combination of neighbouring SNPs. Possible values are Std and Bps.
windowParam further parameters as the window size
summaryMethod allowed versions for the summarization step are: Gaussian, Variational, Exact. Default is Variational.
summaryParam The parameters for the summaryMethod. Further information can be obtained via the according functions: cn.farms, cn.farms or cn.farms
callParam Additional parameters for runtype (ff or bm) as well as cores for parallelization.
returnValues List with return values.
saveFile Name of the file to save. For possible values see summaryMethod.

Value

Multi-loci summarized data of an instance of ExpressionSet

Author(s)

Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>

Examples

load(system.file("exampleData/slData.RData", package = "cn.farms"))
windowMethod <- "std"
windowParam <- list()
windowParam$windowSize <- 5
windowParam$overlap <- TRUE
summaryMethod <- "Variational"
summaryParam <- list()
summaryParam$cyc <- c(20)
mlData <- mlSummarization(slData, windowMethod, windowParam, summaryMethod, summaryParam)
assayData(mlData)
**normAdd**

Extracts info from the package name

**Description**

Extracts info from the package name

**Usage**

```r
normAdd(pkgname)
```

**Arguments**

- `pkgname`: The package name according to the bioconductor annotation names.

**Value**

Additional info for save files.

**Author(s)**

Andreas Mitterecker

---

**normalizeAverage**

Scales the range of the non-polymorphic data to the range of a given array.

**Description**

Scales the range of the non-polymorphic data to the range of a given array.

**Usage**

```r
normalizeAverage(x, baselineArray, avg = median, targetAvg = 2200, ...)
```

**Arguments**

- `x`: Data matrix
- `baselineArray`: Choose the baseline channel array.
- `avg`: The function for averaging.
- `targetAvg`: Value to which the array should be averaged.
- `...`: Further optional parameters.

**Value**

Normalized non-polymorphic data.
normalizeCels

Author(s)
Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>

Examples

```r
x <- matrix(rnorm(100, 11), 20, 5)
normalizeAverage(x, x[, 1])
```

---

**normalizeCels**  
Wrapper for the normalization functions

**Description**

This function provides different normalization methods for microarray data. At the moment only SOR and quantile normalization are implemented.

**Usage**

```r
normalizeCels(filenames, method = c("SOR", "quantiles", "none"), cores = 1,
alleles = FALSE, runtype = "bm", annotDir = NULL,
saveFile = "normData", ...)
```

**Arguments**

- `filenames`: The absolute path of the CEL files as a list.
- `method`: The normalization method. Possible methods so far: SOR, quantiles
- `cores`: Number of cores for used for parallelization.
- `alleles`: States if information for allele A and B should be given back.
- `runtype`: Mode how the results are saved. Possible values are ff or bm. If ff is chosen the data will not be saved automatically. With bm the results will be saved permanently.
- `annotDir`: An optional annotation directory.
- `saveFile`: Name of the file to save.
- `...`: Further parameters for the normalization method.

**Value**

An ExpressionSet object with the normalized data.

**Author(s)**

Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>
normalizeNone

Examples

```r
## Not run:
library("hapmapsnp6")
celDir <- system.file("celFiles", package = "hapmapsnp6")
filenames <- dir(path = celDir, full.names = TRUE)
createAnnotation(filenames = filenames)
normData <- normalizeCels(filenames, method = "SOR")

## End(Not run)
```

normalizeNone

*Runs the SOR normalization on microarray data*

Description

Runs the SOR normalization on microarray data

Usage

```r
normalizeNone(filenames, cores = 1, annotDir = NULL, alleles = FALSE, 
runtype = "ff", cyc = 5, pkgname = NULL, saveFile = "Sor")
```

Arguments

- `filenames`: an absolute path of the CEL files
- `cores`: cores
- `annotDir`: annotDir
- `alleles`: alleles
- `cyc`: states the number of cycles for the EM algorithm.
- `runtype`: Mode how the results are saved. Possible values are ff or bm. If ff is chosen the data will not be saved automatically. With bm the results will be saved permanently.
- `pkgname`: Optional parameter for the CEL mapping.
- `saveFile`: Name of the file to save.

Value

An instance of `ExpressionSet`

Author(s)

Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>
normalizeNpData

*Processes the non-polymorphic data*

**Description**

Normalization for non-polymorphic data for Affymetrix SNP5 and SNP6

**Usage**

```r
normalizeNpData(filenames, cores = 1, annotDir = NULL, runtype = "ff", saveFile = "npData", method = c("baseline", "quantiles", "none"))
```

**Arguments**

- **filenames**
  - the absolute path of the CEL files as a list
- **cores**
  - number of cores for used for parallelization
- **annotDir**
  - Optional annotation directory.
- **runtype**
  - Mode how the results are saved. Possible values are ff or bm. If ff is chosen the data will not be saved automatically. With bm the results will be saved permanently.
- **saveFile**
  - Name of the file to save.
- **method**
  - The method for the normalization.

**Value**

An instance of `ExpressionSet` containing the non-polymorphic data of the microarray.

**Author(s)**

Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>

**Examples**

```r
## Not run:
library("hapmapsnp6")
celDir <- system.file("celFiles", package = "hapmapsnp6")
filenames <- dir(path = celDir, full.names = TRUE)
createAnnotation(filenames = filenames)
npData <- normalizeNpData(filenames)

## End(Not run)
```
normalizeQuantiles  

Normalization Quantiles

Description

Normalization Quantiles

Usage

normalizeQuantiles(filenames, cores = 1, batch = NULL, annotDir = NULL, runtype = "ff", pkgname = NULL, saveFile = "normDataQuant")

Arguments

filenames  filenames
cores  cores
batch  batch
annotDir  annotDir
runtype  Mode how the results are saved. Possible values are ff or bm. If ff is chosen the data will not be saved automatically. With bm the results will be saved permanently.
pkgname  Optional parameter for the CEL mapping.
saveFile  Name of the file to save.

Value

The normalized data.

Author(s)

Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>

normalizeSequenceEffect

Correction for probe sequence effects

Description

Correction for probe sequence effects

Usage

normalizeSequenceEffect(object, annotDir = NULL, runtype = "ff", saveFile = "seqNorm")
normalizeSor

Arguments

<table>
<thead>
<tr>
<th>object</th>
<th>an instance of ExpressionSet</th>
</tr>
</thead>
<tbody>
<tr>
<td>annotDir</td>
<td>the directory where the annotation can be found</td>
</tr>
<tr>
<td>runtype</td>
<td>mode how the results are saved. Possible values are ff or bm. If ff is chosen the data will not be saved automatically.</td>
</tr>
<tr>
<td>saveFile</td>
<td>name of the file to save.</td>
</tr>
</tbody>
</table>

Value

Some data

Author(s)

Andreas Mitterecker

---

normalizeSor  Runs the SOR normalization on microarray data

Description

Runs the SOR normalization on microarray data

Usage

normalizeSor(filenames, cores = 1, annotDir = NULL, alleles = FALSE, runtype = "ff", cyc = 5, pkgname = NULL, saveFile = "Sor")

Arguments

<table>
<thead>
<tr>
<th>filenames</th>
<th>an absolute path of the CEL files</th>
</tr>
</thead>
<tbody>
<tr>
<td>cores</td>
<td>cores</td>
</tr>
<tr>
<td>annotDir</td>
<td>annotDir</td>
</tr>
<tr>
<td>alleles</td>
<td>alleles</td>
</tr>
<tr>
<td>cyc</td>
<td>states the number of cycles for the EM algorithm.</td>
</tr>
<tr>
<td>runtype</td>
<td>Mode how the results are saved. Possible values are ff or bm. If ff is chosen the data will not be saved automatically. With bm the results will be saved permanently.</td>
</tr>
<tr>
<td>pkgname</td>
<td>Optional parameter for the CEL mapping.</td>
</tr>
<tr>
<td>saveFile</td>
<td>Name of the file to save.</td>
</tr>
</tbody>
</table>

Value

An instance of ExpressionSet

Author(s)

Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>
plotDendrogram

Plots a dendrogram

Description

Plots a dendrogram

Usage

plotDendrogram(DivMetric, colorLabels)

Arguments

DivMetric The input data (see example).
colorLabels A color label with the dimension of the columns.

Value

A dendrogram.

Author(s)

Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>

Examples

load(system.file("exampleData/normData.RData", package = "cn.farms"))
x <- assayData(normData)$intensity[, 1:3]
y <- distributionDistance(x)
attr(y, "Labels") <- substr(sampleNames(normData), 1, 7)
plotDendrogram(y)

plotDensity

Function to create a density plot

Description

Simple density plot. Adapted from the aroma.affymetrix package (www.aroma-project.org)

Usage

plotDensity(x, xlim = c(0, 16), ylim, col, lty, lwd, add = FALSE, xlab, ylab, log = TRUE, ...)
plotEvalIc

Arguments

- **x**: Matrix with numeric values.
- **xlim**: The limits for the x axis.
- **ylim**: The limits for the y axis.
- **col**: Vector with colors corresponding to the columns of the matrix.
- **lty**: The line type (see `graphics`).
- **lwd**: The line width, a positive number, defaulting to 1 (see `graphics`).
- **add**: If FALSE (the default) then a new plot is produced. If TRUE, density lines are added to the open graphics device.
- **xlab**: The labeling of the x axis.
- **ylab**: The labeling of the y axis.
- **log**: Logical values which states if the log2 should be taken from the data.
- **...**: Further arguments which states from the data.

Value

A plot written to the graphics device.

Author(s)

Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>

Examples

```r
load(system.file("exampleData/slData.RData", package = "cn.farms"))
plotDensity(assayData(slData)$intensity)
```

```r
plotEvalIc

Creates a plot with known regions and a numeric vector

Description

Creates a plot with known regions and a numeric vector

Usage

```r
plotEvalIc(object, segments, chrom, variable, ylim, ylab = "CN indicator", stripCol = "lightgray", regionCol = rgb(130, 0, 139, maxColorValue = 255), pointSize = 0.75, pointType = 4, bandwidth = c(0.01, 1000), nbin = 100)
```
plotRegions

Arguments

- **object**: an instance of `ExpressionSet`
- **segments**: A data.frame with known regions.
- **chrom**: the chromosome.
- **variable**: The numeric vector which should be plotted.
- **ylim**: the limits of the y axis.
- **ylab**: the ylab from function par.
- **stripCol**: color of points.
- **regionCol**: color of regions.
- **pointSize**: size of the points.
- **pointType**: type of the points.
- **bandwidth**: for the color of the plot.
- **nbin**: number of bins for the coloring.

Value

Some data

Author(s)

Andreas Mitterecker

Examples

```r
load(system.file("exampleData/slData.RData", package = "cn.farms"))
load(system.file("exampleData/testSegments.RData", package = "cn.farms"))
plotEvalIc(slData, fData(testSegments),
            variable = assayData(slData)$L_z[, 1], 23)
```

Description

A pdf in the working directory is produced.

Usage

```r
plotRegions(object, segments, addInd = NULL, ylim, variable,
            colorVersion = 0, plotLegend = TRUE, pdfname)
```
plotSmoothScatter

Arguments

- **object**: An instance of `ExpressionSet`.
- **segments**: An instance of `ExpressionSet` with the segments to plot.
- **addInd**: States how many indices should be plotted besides the region.
- **ylim**: The limits for the y axis.
- **variable**: States which variable of the assayData should be plotted.
- **colorVersion**: States different color versions.
- **plotLegend**: If a legend should be plotted or not.
- **pdfname**: The name of the pdf file.

Value

A graph. Normally a pdf in the current work directory.

Author(s)

Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>

Examples

```r
load(system.file("exampleData/slData.RData", package = "cn.farms"))
load(system.file("exampleData/testSegments.RData", package = "cn.farms"))
plotRegions(slData, testSegments, addInd = 10, ylim = c(-2, 2),
            variable = "L_z", colorVersion = 1, plotLegend = TRUE,
            pdfname = "slData.pdf")
```

Description

Creates a smooth scatter plot

Usage

```r
plotSmoothScatter(object, variable, chrom, start, end, ylim, pdfname, ...)```

Arguments

- **object**: An instance of `ExpressionSet`.
- **variable**: States which variable of the assayData should be plotted.
- **chrom**: The chromosome you want to plot.
- **start**: The physical start position.
- **end**: The physical end position.
- **ylim**: The limits for the y axis.
- **pdfname**: The name of the pdf file.
- **...**: Further arguments passed to `smoothScatter` function.
plotViolines

Value
A graph.

Author(s)
Andreas Mitterecker

Examples
load(system.file("exampleData/slData.RData", package = "cn.farms"))
plotSmoothScatter(slData[, 1:3], chrom = "23")

plotViolines
Create a violine plot

Description
This function creates a violine plot on intensity values

Usage
plotViolines(object, variable = "intensity", groups, ...)

Arguments
object An instance of ExpressionSet
variable states which variable of assayData should be plotted.
groups Vector with the dimension of the samples for coloring.
... Further arguments passed to the lattice graph.

Value
Creates a violine plot.

Author(s)
Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>

Examples
load(system.file("exampleData/normData.RData", package = "cn.farms"))
normData <- normData[, 1:10]
groups <- seq(sampleNames(normData))
plotViolines(normData, variable = "intensity", groups, xlab = "Intensity values")
slSummarization  Method for computation of the single-locus summarization

Description
The different probes of the SNPs of the array are summarized to a probeset.

Usage
slSummarization(object, summaryMethod = "Exact", summaryParam = list(),
callParam = list(runtype = "ff", cores = 1), summaryWindow = c("std",
"fragment"), returnValues, saveFile = "slData")

Arguments
- object: An instance of ExpressionSet
- summaryMethod: allowed versions for the summarization step are: Gaussian, Variational, Exact. Default is Variational.
- summaryParam: The parameters for the summaryMethod. Further information can be obtained via the according functions: cn.farms, cn.farms or cn.farms
- callParam: Additional parameters for runtype (ff or bm) as well as cores for parallelization.
- summaryWindow: Method for combination of the SNPs. Possible values are sl and fragment.
- returnValues: List with return values.
- saveFile: Name of the file to save.

Value
Single-locus summarized data of an instance of ExpressionSet

Author(s)
Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>

See Also
summarizeFarmsExact

Examples
load(system.file("exampleData/normData.RData", package = "cn.farms"))
notes(experimentData(normData))$annotDir <-
  system.file("exampleData/annotation/pd.genomewidesnp.6/1.1.0",
  package = "cn.farms")
summaryMethod <- "Variational"
summaryParam <- list()
summaryParam$cyc <- c(10)
slData <- slSummarization(normData,
sparseFarmsC

Normalizes the data with SOR

Description
Normalizes the data with SOR

Usage
sparseFarmsC(probes, cyc = 5)

Arguments

  - **probes**: The intensity matrix.
  - **cyc**: Number of cycles.

Value
Normalized Data.

Author(s)
Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>

Examples
x <- matrix(rnorm(100, 11), 20, 5)
sparseFarmsC(x, 50)
summarizeFarmsExact

Summarization Laplacian approach with exact computation

Description

This function implements an exact Laplace FARMS algorithm.

Usage

```r
summarizeFarmsExact(probes, mu = 1, weight = 0.001, weightSignal = 1, 
weightZ = 1, weightProbes = TRUE, cyc = c(10, 10), tol = 1e-05, 
weightType = "mean", centering = "median", rescale = FALSE, 
backscaleComputation = FALSE, maxIntensity = TRUE, refIdx, ...)
```

Arguments

- `probes`: A matrix with numeric values.
- `mu`: Hyperparameter value which allows to quantify different aspects of potential prior knowledge. Values near zero assumes that most positions do not contain a signal, and introduces a bias for loading matrix elements near zero. Default value is 0 and it’s recommended not to change it.
- `weight`: Hyperparameter value which determines the influence of the Gaussian prior of the loadings.
- `weightSignal`: Hyperparameter value on the signal.
- `weightZ`: Hyperparameter value which determines how strong the Laplace prior of the factor should be at 0. Users should be aware, that a change of weightZ in comparison to the default parameter might also entail a need to change other parameters. Unexperienced users should not change weightZ.
- `weightProbes`: Parameter (TRUE/FALSE), that determines, if the number of probes should additionally be considered in weight. If TRUE, weight will be modified.
- `cyc`: Number of cycles. If the length is two, it is assumed, that a minimum and a maximum number of cycles is given. If the length is one, the value is interpreted as the exact number of cycles to be executed (minimum == maximum).
- `tol`: States the termination tolerance if cyc[1]!=cyc[2]. Default is 0.00001.
- `weightType`: Flag, that is used to summarize the probes of a sample.
- `centering`: States how the data should be centered ("mean", "median"). Default is median.
- `rescale`: Parameter (TRUE/FALSE), that determines, if moments in exact Laplace FARMS are rescaled in each iteration. Default is FALSE.
- `backscaleComputation`: Parameter (TRUE/FALSE), that determines if the moments of hidden variables should be reestimated after rescaling the parameters.
- `maxIntensity`: Parameter (TRUE/FALSE), that determines if the expectation value (=FALSE) or the maximum value (=TRUE) of p(z|x_i) should be used for an estimation of the hidden varaible.
refIdx         index or indices which are used for computation of the centering
...            Further parameters for expert users.

Value
A list including: the found parameters: lambda0, lambda1, Psi
the estimated factors: z (expectation), maxZ (maximum)
p: log-likelihood of the data given the found lambda0, lambda1, Psi (not the posterior likelihood
that is optimized)
varz$x$: variances of the hidden variables given the data
KL: Kullback Leibler divergences between between posterior and prior distribution of the hidden
variables
IC: Information Content considering the hidden variables and data
ICtransform: transformed Information Content
Case: Case for computation of a sample point (non-exception, special exception)
L1median: Median of the lambda vector components
intensity: back-computed summarized probeset values with mean correction
L_z: back-computed summarized probeset values without mean correction
rawCN: transformed values of L_z
SNR: some additional signal to noise ratio value

Author(s)
Andreas Mayr <mayr@bioinf.jku.at> and Djork-Arne Clevert <okko@clevert.de> and Andreas
Mitterecker <mitterecker@bioinf.jku.at>

Examples
x <- matrix(rnorm(100, 11), 20, 5)
summarizeFarmsExact(x)

summarizeFarmsExact2 Summarization Laplacian approach with exact computation

Description
This function implements an exact Laplace FARMS algorithm.

Usage
summarizeFarmsExact2(probes, mu = 1, weight = 0.5, weightSignal = 1,
weightZ = 1, weightProbes = TRUE, cyc = c(10, 10), tol = 1e-05,
weightType = "mean", centering = "median", rescale = FALSE,
backscurveComputation = FALSE, maxIntensity = TRUE, refIdx, ...)

---

summarizeFarmsExact2

---
**Arguments**

- **probes**
  A matrix with numeric values.

- **mu**
  Hyperparameter value which allows to quantify different aspects of potential prior knowledge. Values near zero assumes that most positions do not contain a signal, and introduces a bias for loading matrix elements near zero. Default value is 0 and it’s recommended not to change it.

- **weight**
  Hyperparameter value which determines the influence of the Gaussian prior of the loadings

- **weightSignal**
  Hyperparameter value on the signal.

- **weightZ**
  Hyperparameter value which determines how strong the Laplace prior of the factor should be at 0. Users should be aware, that a change of weightZ in comparison to the default parameter might also entail a need to change other parameters. Unexperienced users should not change weightZ.

- **weightProbes**
  Parameter (TRUE/FALSE), that determines, if the number of probes should additionally be considered in weight. If TRUE, weight will be modified.

- **cyc**
  Number of cycles. If the length is two, it is assumed, that a minimum and a maximum number of cycles is given. If the length is one, the value is interpreted as the exact number of cycles to be executed (minimum == maximum).

- **tol**
  States the termination tolerance if cyc[1]!=cyc[2]. Default is 0.00001.

- **weightType**
  Flag, that is used to summarize the probes of a sample.

- **centering**
  States how the data should be centered ("mean", "median"). Default is median.

- **rescale**
  Parameter (TRUE/FALSE), that determines, if moments in exact Laplace FARMS are rescaled in each iteration. Default is FALSE.

- **backscaleComputation**
  Parameter (TRUE/FALSE), that determines if the moments of hidden variables should be reestimated after rescaling the parameters.

- **maxIntensity**
  Parameter (TRUE/FALSE), that determines if the expectation value (=FALSE) or the maximum value (=TRUE) of p(z|x_i) should be used for an estimation of the hidden varaible.

- **refIdx**
  index or indices which are used for computation of the centering

... Further parameters for expert users.

**Value**

A list including: the found parameters: lambda0, lambda1, Psi
the estimated factors: z (expectation), maxZ (maximum)
p: log-likelihood of the data given the found lambda0, lambda1, Psi (not the posterior likelihood that is optimized)
varzx: variances of the hidden variables given the data
KL: Kullback Leibler divergences between between posterior and prior distribution of the hidden variables
IC: Information Content considering the hidden variables and data
ICtransform: transformed Information Content
Case: Case for computation of a sample point (non-exception, special exception)
L1median: Median of the lambda vector components
intensity: back-computed summarized probeset values with mean correction
L_z: back-computed summarized probeset values without mean correction
rawCN: transformed values of L_z
SNR: some additional signal to noise ratio value

Author(s)
Andreas Mayr <mayr@bioinf.jku.at> and Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>

Examples

```r
x <- matrix(rnorm(100, 11), 20, 5)
summarizeFarmsExact3(x)
```

---

**summarizeFarmsExact3**  
*Summarization Laplacian approach with exact computation*

**Description**

This function implements an exact Laplace FARMS algorithm.

**Usage**

```r
summarizeFarmsExact3(probes, mu = 1, weight = 100, weightSignal = 1,
weightZ = 30, weightProbes = TRUE, updateSignal = FALSE, cyc = c(10,
10), tol = 1e-05, weightType = "mean", centering = "median",
rescale = FALSE, backscaleComputation = FALSE, maxIntensity = TRUE,
refIdx, ...)
```

**Arguments**

- **probes**: A matrix with numeric values.
- **mu**: Hyperparameter value which allows to quantify different aspects of potential prior knowledge. Values near zero assumes that most positions do not contain a signal, and introduces a bias for loading matrix elements near zero. Default value is 0 and it’s recommended not to change it.
- **weight**: Hyperparameter value which determines the influence of the Gaussian prior of the loadings
- **weightSignal**: Hyperparameter value on the signal.
weightZ Hyperparameter value which determines how strong the Laplace prior of the factor should be at 0. Users should be aware, that a change of weightZ in comparison to the default parameter might also entail a need to change other parameters. Unexperienced users should not change weightZ.

weightProbes Parameter (TRUE/FALSE), that determines, if the number of probes should additionally be considered in weight. If TRUE, weight will be modified.

updateSignal updateSignal.

cyc Number of cycles. If the length is two, it is assumed, that a minimum and a maximum number of cycles is given. If the length is one, the value is interpreted as the exact number of cycles to be executed (minimum == maximum).

tol States the termination tolerance if cyc[1]!=cyc[2]. Default is 0.00001.

weightType Flag, that is used to summarize the probes of a sample.

centering States how the data should be centered ("mean", "median"). Default is median.

rescale Parameter (TRUE/FALSE), that determines, if moments in exact Laplace FARMS are rescaled in each iteration. Default is FALSE.

backscaleComputation Parameter (TRUE/FALSE), that determines if the moments of hidden variables should be reestimated after rescaling the parameters.

maxIntensity Parameter (TRUE/FALSE), that determines if the expectation value (=FALSE) or the maximum value (=TRUE) of p(z\_i) should be used for an estimation of the hidden variable.

refIdx index or indices which are used for computation of the centering

... Further parameters for expert users.

Value

A list including: the found parameters: lambda0, lambda1, Psi
the estimated factors: z (expectation), maxZ (maximum)
p: log-likelihood of the data given the found lambda0, lambda1, Psi (not the posterior likelihood that is optimized)
varzx: variances of the hidden variables given the data
KL: Kullback Leibler divergences between between posterior and prior distribution of the hidden variables
IC: Information Content considering the hidden variables and data
ICtransform: transformed Information Content
Case: Case for computation of a sample point (non-exception, special exception)
L1median: Median of the lambda vector components
intensity: back-computed summarized probeset values with mean correction
L_z: back-computed summarized probeset values without mean correction
rawCN: transformed values of L_z
SNR: some additional signal to noise ratio value
summarizeFarmsGaussian

Author(s)

Andreas Mayr <mayr@bioinf.jku.at> and Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>

Examples

```r
x <- matrix(rnorm(100, 11), 20, 5)
summarizeFarmsExact(x)
```

---

summarizeFarmsGaussian

*Summarization Gaussian approach*

Description

This function runs the FARMS algorithm.

Usage

```r
summarizeFarmsGaussian(probes, weight = 0.15, mu = 0, cyc = 10,
tol = 1e-04, weightType = "mean", init = 0.6, correction = 0,
minNoise = 0.35, centering = "median", refIdx)
```

Arguments

- `probes` A matrix with numeric values.
- `weight` Hyperparameter value in the range of [0,1] which determines the influence of the prior.
- `mu` Hyperparameter value which allows to quantify different aspects of potential prior knowledge. Values near zero assumes that most genes do not contain a signal, and introduces a bias for loading matrix elements near zero. Default value is 0.
- `cyc` Number of cycles for the EM algorithm.
- `tol` States the termination tolerance. Default is 0.00001.
- `weightType` Flag, that is used to summarize the loading matrix. The default value is set to mean.
- `init` Parameter for estimation.
- `correction` Value that indicates whether the covariance matrix should be corrected for negative eigenvalues which might emerge from the non-negative correlation constraints or not. Default = 0 (means that no correction is done), 1 (minimal noise (0.0001) is added to the diagonal elements of the covariance matrix to force positive definiteness), 2 (Maximum Likelihood solution to compute the nearest positive definite matrix under the given non-negative correlation constraints of the covariance matrix)
- `minNoise` States the minimal noise. Default is 0.35.
- `centering` States how the data is centered. Default is median.
- `refIdx` index or indices which are used for computation of the centering
Value

A list containing the results of the run.

Author(s)

Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>

Examples

```r
x <- matrix(rnorm(100, 11), 20, 5)
summarizeFarmsGaussian(x)
```

---

### summarizeFarmsMethods

Lists methods for possible FARMS summarization

Description

Possible FARMS summarization

Usage

```r
summarizeFarmsMethods()
```

Value

Returns a data frame with all possible FARMS calls.

Author(s)

Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>

Examples

```r
summarizeFarmsMethods()
```
summarizeFarmsStatistics

Mean or median instead of the FARMS model

Description

Mean or median instead of the FARMS model

Usage

summarizeFarmsStatistics(probes, type = "median", ...)

Arguments

probes A matrix with numeric values.
type The statistic which you want to apply.
... Further parameters

Value

Some data

Author(s)

Andreas Mitterecker

summarizeFarmsVariational

Summarization variational Laplacian approach

Description

This function runs the FARMS algorithm.

Usage

summarizeFarmsVariational(probes, weight = 0.15, mu = 0, cyc = 10,
weightType = "median", init = 0.6, correction = 0, minNoise = 0.35,
spuriousCorrelation = 0.3, centering = "median")
summarizeWindowBps

Arguments

- **probes**: A matrix with numeric values.
- **weight**: Hyperparameter value in the range of [0,1] which determines the influence of the prior.
- **mu**: Hyperparameter value which allows to quantify different aspects of potential prior knowledge. Values near zero assumes that most genes do not contain a signal, and introduces a bias for loading matrix elements near zero. Default value is 0.
- **cyc**: Number of cycles for the EM algorithm.
- **weightType**: Flag, that is used to summarize the loading matrix. The default value is set to mean.
- **init**: Parameter for estimation.
- **correction**: Value that indicates whether the covariance matrix should be corrected for negative eigenvalues which might emerge from the non-negative correlation constraints or not. Default = 0 (means that no correction is done), 1 (minimal noise (0.0001) is added to the diagonal elements of the covariance matrix to force positive definiteness), 2 (Maximum Likelihood solution to compute the nearest positive definite matrix under the given non-negative correlation constraints of the covariance matrix)
- **spuriousCorrelation**: Numeric value for suppression of spurious correlation.
- **minNoise**: States the minimal noise. Default is 0.35.
- **centering**: States how the data is centered. Default is median.

Value

A list containing the results of the run.

Author(s)

Djork-Arne Clevert &lt;okko@clevert.de&gt; and Andreas Mitterecker &lt;mitterecker@bioinf.jku.at&gt;

Examples

```r
x <- matrix(rnorm(100, 11), 20, 5)
summarizeFarmsVariational(x)
```

---

**summarizeWindowBps**  
Combines neighbouring locations to windows

Description

Combines neighbouring locations to windows
Usage

summarizeWindowBps(phInf, fixedBps = 10000, upperLimit = 6)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>phInf</td>
<td>The locations on the chromosomes.</td>
</tr>
<tr>
<td>fixedBps</td>
<td>Size of the window in basepairs.</td>
</tr>
<tr>
<td>upperLimit</td>
<td>Maximal number of neighbouring locations to combine.</td>
</tr>
</tbody>
</table>

Value

Indices for summarization

Author(s)

Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>

Examples

```r
## create toy physical data
sizeTmp <- 30
phInf <- data.frame(chrom = rep("15", sizeTmp),
                     start = seq(from = 1, by = 300, length.out = sizeTmp),
                     end = seq(from = 3600, by = 300, length.out = sizeTmp),
                     man_fsetid = paste("SNP_A-", seq(sizeTmp)+1000, sep = ""))
summarizeWindowBps(phInf)
```

summarizeWindowMethods

*Lists methods for possible window methods*

Description

Function to list how neighbouring positions can be combined.

Usage

summarizeWindowMethods()

Value

Returns a data frame with all possible methods.

Author(s)

Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>
Examples

summarizeWindowMethods()

summarizeWindowStd  Combines neighbouring locations to windows

Description

Combines neighbouring locations to windows

Usage

summarizeWindowStd(phInf, windowSize = 3, overlap = TRUE)

Arguments

phInf  The locations on the chromosomes.
windowSize  Size of how many Locations should be combined.
overlap  States if the windows should overlap.

Value

Indices for summarization

Author(s)

Djork-Arne Clevert <okko@clevert.de> and Andreas Mitterecker <mitterecker@bioinf.jku.at>

Examples

## create toy physical data
sizeTmp <- 30
phInf <- data.frame(
  chrom = rep("15", sizeTmp),
  start = seq(from = 1, by = 300, length.out = sizeTmp),
  end = seq(from = 3600, by = 300, length.out = sizeTmp),
  man_fsetid = paste("SNP_A-", seq(sizeTmp)+1000, sep = ""))
summarizeWindowStd(phInf)
Index

callSummarize, 3
cn.farms, 4, 14, 26
cnLibrary, 4
combineData, 5
createAnnotation, 6
createMatrix, 7
distributionDistance, 8
dnaCopySf, 9
doCnFarmsSingle, 10

ExpressionSet, 4, 6, 9, 12, 14, 17, 18, 20, 23–26

flcSnp6Std, 10
flcStd, 11
fragLengCorr, 12

getFragmentSet, 13
getSingleProbeSetSize, 13
graphics, 22

mlSummarization, 14

normAdd, 15
normalizeAverage, 15
normalizeCels, 16
normalizeNone, 17
normalizeNpData, 18
normalizeQuantiles, 19
normalizeSequenceEffect, 19
normalizeSor, 20

plotDendrogram, 21
plotDensity, 21
plotEvalIc, 22
plotRegions, 23
plotSmoothScatter, 24
plotViolines, 25

slSummarization, 26

sparseFarmsC, 27
summarizeFarmsExact, 26, 28
summarizeFarmsExact2, 29
summarizeFarmsExact3, 31
summarizeFarmsGaussian, 33
summarizeFarmsMethods, 34
summarizeFarmsStatistics, 35
summarizeFarmsVariational, 35
summarizeWindowBps, 36
summarizeWindowMethods, 37
summarizeWindowStd, 38