Package ‘MANOR’

March 5, 2024

Version  1.74.0
Date    2021-04-15
Title   CGH Micro-Array NORmalization
Description Importation, normalization, visualization, and quality control functions to correct identified sources of variability in array-CGH experiments.
Author  Pierre Neuvi\text{al} <pierre.neuvial@math.cnrs.fr>, Philippe Hupé <philippe.hupe@curie.fr>
Maintainer Pierre Neuvi\text{al} <pierre.neuvial@math.cnrs.fr>
License  GPL-2
Depends R (\text{\geq} 2.10)
Imports GLAD, graphics, grDevices, stats, utils
Suggests knitr, rmarkdown, bookdown
VignetteBuilder knitr
BugReports https://github.com/pneuvial/MANOR/issues
Encoding UTF-8
biocViews Microarray, TwoChannel, DataImport, QualityControl, Preprocessing, CopyNumberVariation, Normalization
git\_url https://git.bioconductor.org/packages/MANOR
git\_branch RELEASE_3_18
git\_last\_commit bbfd064
git\_last\_commit\_date 2023-10-24
Repository Bioconductor 3.18
Date/Publication 2024-03-05
Description

The function `arrayTrend` computes the spatial trend.

Usage

```r
## Default S3 method:
arrayTrend(Statistic, Col, Row, ...)
## S3 method for class 'arrayCGH'
arrayTrend(arrayCGH, variable, ...)
```

Arguments

- **Statistic**: Statistic to be smoothed.
- **Col**: Vector of columns coordinates.
- **Row**: Vector of rows coordinates.
- **arrayCGH**: Object of class `arrayCGH`.
- **variable**: Variable to be smooth.
- **...**: Parameters to be passed to `loess` function.
arrayTrend

Details

Spatial trend of microarray spots statistic.

Value

Either a data frame with elements:

- `Trend`: Trend fitted by `loess` function.
- `Col`: Vector of columns coordinates.
- `Row`: Vector of rows coordinates.

or the element `Trend` is added to the data.frame `arrayValues` of the `arrayCGH` object.

Note

People interested in tools for array-CGH analysis can visit our web-page: [http://bioinfo.curie.fr](http://bioinfo.curie.fr).

Author(s)

Philippe Hup?, <Philippe.Hupe@curie.fr>.

References


See Also

`loess`, `loess.control`.

Examples

data(spatial)  ## arrays with local spatial effects

edgeTrend <- arrayTrend(edge, "LogRatio", span=0.03, degree=1, iterations=3, family="symmetric")
GLAD::arrayPlot(edgeTrend, "Trend", main="Spatial trend of array CGH", bar="v")
**description**

This function detects spatial bias on array CGH.

**Usage**

```r
## S3 method for class 'arrayCGH'
detectSB(arrayCGH, variable, proportionup=0.25,
         proportiondown,type="up", thresholdup=0.2, thresholddown=0.2, ... )
```

**Arguments**

- `arrayCGH`: Object of `arrayCGH`.
- `variable`: Variable used to compare the mean of zones detected by `nem`.
- `proportionup`: Maximal proportion of the array which may be affected by spatial bias with high values.
- `proportiondown`: Maximal proportion of the array which may be affected by spatial bias with low values.
- `type`: Type of spatial bias detected. Specify either "up" (to detect spatial bias with high values), or "down" (to detect spatial bias with low values) or "upanddown" (to detect both type of spatial bias).
- `thresholdup`: Threshold used to detect spatial bias with high values.
- `thresholddown`: Threshold used to detect spatial bias with low values.
- `...`: ...

**Details**

You must run the `arrayTrend` and `nem` function before detecting spatial bias: the `arrayTrend` computes a spatial trend and the `nem` function performs a classification with spatial constraints defining different zones on the array. Based on those results, spatial bias is detected.

**Value**

An object of class `arrayCGH` with the following added information in the data.frame attribute `arrayValues`:

- `SB`: Spots located in zone of spatial bias are coded either by 1 (if they correspond to a spatial bias with high values) or by -1 (if they correspond to a spatial bias with low values). Otherwise they are coded by 0.

**Note**

People interested in tools for array-CGH analysis can visit our web-page: `http://bioinfo.curie.fr`.
Author(s)


References


See Also

arrayTrend, nem

Examples

data(spatial) ## arrays with local spatial effects

## Plot of LogRatio measured on the array CGH
GLAD::arrayPlot(edge,"LogRatio", main="Log2-Ratio measured on the array CGH", zlim=c(-1,1), bar="v", mediancenter=TRUE)

## Spatial trend of the scaled log-ratios (the variable "ScaledLogRatio" ## equals to the log-ratio minus the median value of the corresponding ## chromosome arm)
edgeTrend <- arrayTrend(edge, variable="ScaledLogRatio", span=0.03, degree=1, iterations=3, family="symmetric")
GLAD::arrayPlot(edgeTrend, variable="Trend", main="Spatial trend of the array CGH", bar="v")

## Not run: ## Classification with spatial constraint of the spatial trend
edgeNem <- nem(edgeTrend, variable="Trend")
GLAD::arrayPlot(edgeNem, variable="ZoneNem", main="Spatial zones identified by nem", bar="v")

# Detection of spatial bias
edgeDet <- detectSB(edgeNem, variable="LogRatio", proportionup=0.25,type="up", thresholdup=0.15)
GLAD::arrayPlot(edgeDet, variable="SB", main="Zone of spatial bias in red", bar="v")

# CGH profile
plot(LogRatio ~ PosOrder, data=edgeDet$arrayValues,
col=c("black","red")[as.factor(SB)], pch=20, main="CGH profile: spots located in spatial bias are in red")

## End(Not run)
Apply a flag to an arrayCGH

Description

Function flag$FUN is applied to a flag object for normalization.

Usage

flag.arrayCGH(flag, arrayCGH)

Arguments

- flag: an object of type 'flag'
- arrayCGH: an object of type arrayCGH

Details

Optional arguments in flag$args are passed to flag$FUN.

Value

An object of class arrayCGH, which corresponds to the return value of flag$FUN if flag$char is null, and to the input arrayCGH object with spots given by flag$FUN flagged with flag$char.

Note

People interested in tools for array-CGH analysis can visit our web-page: [http://bioinfo.curie.fr](http://bioinfo.curie.fr).

Author(s)

Pierre Neuvial, <manor@curie.fr>.

See Also

to.flag, norm.arrayCGH

Examples

data(spatial)
data(flags)

gradient$arrayValues$LogRatioNorm <- gradient$arrayValues$LogRatio
## flag spots with no available position on the genome
gradient <- flag.arrayCGH(position.flag, gradient)

## flag spots corresponding to low poor quality clones
gradient <- flag.arrayCGH(val.mark.flag, gradient)
flag.summary

Summarize information about flags after array normalization

Description
Compute spot-level information (number of flagged spots, normalization parameters), and display it in a convenient way

Usage
## S3 method for class 'arrayCGH'
flag.summary(arrayCGH, flag.list, flag.var="Flag", nflag="not flagged", ...)
## Default S3 method:
flag.summary(spot.flags, flag.list, nflag="not flagged", ...)

Arguments
arrayCGH an object of type arrayCGH, after normalization by MANOR
flag.list a list of flags with flag\$char corresponding to the values of spot.flags
flag.var the name of a variable of arrayCGH\$arrayValues containing information about flags (defaults to Flag)
var the name of a variable of arrayCGH\$cloneValues containing signal values (defaults to LogRatio)
spot.flags a character vector containing information about flags
nflab a character vector providing a legend for "not flagged" spots

Details

This function is used by the function html.report for the generation of an HTML report of the normalization step. It can also be used by itself.

Value

A data.frame data.frame with 4 columns:

<table>
<thead>
<tr>
<th>name</th>
<th>flag character</th>
</tr>
</thead>
<tbody>
<tr>
<td>label</td>
<td>flag label</td>
</tr>
<tr>
<td>arg</td>
<td>first numeric argument of flag$FUN</td>
</tr>
<tr>
<td>count</td>
<td>number of flagged spots</td>
</tr>
</tbody>
</table>

Note

People interested in tools for array-CGH analysis can visit our web-page: http://bioinfo.curie.fr.

Author(s)

Pierre Neuvial, <manor@curie.fr>.

See Also

html.report, flag

Examples

data(spatial)
data(flags)
flag.list <- list(spatial=local.spatial.flag, spot=spot.corr.flag, ref.snr=ref.snr.flag, dapi.snr=dapi.snr.flag, rep=rep.flag, unique=unique.flag)
flag.list$spatial$args <- alist(var="ScaledLogRatio", by.var=NULL, nk=5, prop=0.25, thr=0.15, beta=1, family="symmetric")
flag.list$spot$args <- alist(var="SpotFlag")
flag.list$spot$char <- "0"
flag.list$spot$label <- "Image analysis"

## normalize arrayCGH
## Not run: edge.norm <- norm(edge, flag.list=flag.list, var="LogRatio", FUN=median, na.rm=TRUE)
## End(Not run)
fs <- flag.summary(edge.norm, flag.list=flag.list, flag.var="Flag")
Description

This data set provides flag objects that can be applied to arrayCGH objects in order to normalize them.

Usage

data(flags)

Format

These flag objects typically take part to a normalization process:

- `amplicon.flag`: flags spots with high log-ratios (temp flag)
- `chromosome.flag`: flags spots located on sexual chromosomes (named "X" and "Y")
- `control.flag`: flag control spots
- `global.spatial.flag`: corrects arrayCGH from global spatial trend on the array
- `local.spatial.flag`: flags spots belonging to local spatial bias zones on the array
- `num.chromosome.flag`: flags spots located on sexual chromosomes (named 23 and 24)
- `position.flag`: flag spots with no available genome position
- `replicate.flag`: flag spots with poor within-clone-replicate consitency
- `ref.snr.flag`: flags spots with low signal to noise ratio for reference
- `dapi.snr.flag`: flags spots with low signal to noise ratio for DAPI
- `SNR.flag`: flags spots with low signal to noise ratio
- `spot.corr.flag`: flags spots with low correlation coefficient after image analysis
- `spot.flag`: flags spots excluded by the image analysis software
- `unique.flag`: exclude last non-flagged spot of a clone
- `val.mark.flag`: flags spots corresponding to bad quality clones
- `intensity.flag`: corrects for an intensity effect (using loess regression)

Note

People interested in tools for array-CGH analysis can visit our web-page: http://bioinfo.curie.fr.

Author(s)

Pierre Neuvial, <manor@curie.fr>.

Source

Institut Curie, <manor@curie.fr>.
See Also

`spatial`, `norm.arrayCGH`, `flag`, `flag.summary`

Examples

data(flags)

### complete normalization of an arrayCGH object (with spatial gradient):
## Initialize $flag$args

flag.list1 <- list(local.spatial=local.spatial.flag,
  global.spatial=global.spatial.flag, spot=spot.flag, SNR=SNR.flag,
  val.mark=val.mark.flag, unique=unique.flag,
  amplicon=amplicon.flag, chromosome=chromosome.flag,
  replicate=replicate.flag)
data(spatial)
## Not run: gradient.norm <- norm(gradient, flag.list=flag.list1,
  var="LogRatio", FUN=median, na.rm=TRUE)
## End(Not run)
print(gradient.norm$flags) ## spot-level flag summary (computed by flag.summary)

### complete normalization of an arrayCGH object (with local spatial bias):
## Initialize $flag$args

flag.list2 <- list(spatial=local.spatial.flag, spot=spot.corr.flag,
  ref.snr=ref.snr.flag, dapi.snr=dapi.snr.flag, rep=rep.flag,
  unique=unique.flag)
flag.list2$spatial$args <- alist(var="ScaledLogRatio", by.var=NULL,
  nk=5, prop=0.25, thr=0.15, beta=1, family="symmetric")
flag.list2$spot$args <- alist(var="SpotFlag")
flag.list2$spot$char <- "O"
flag.list2$spot$label <- "Image analysis"
## Not run: edge.norm <- norm(edge, flag.list=flag.list2,
  var="LogRatio", FUN=median, na.rm=TRUE)
## End(Not run)
print(edge.norm$flags) ## spot-level flag summary (computed by flag.summary)

---

**genome.plot**

Pan-genomic representation of a normalized arrayCGH

Description

Displays a pan-genomic representation of a normalized arrayCGH.
Usage

```r
## S3 method for class 'arrayCGH'
genome.plot(arrayCGH, x="PosOrder", y="LogRatio",
            chrLim=NULL, col.var=NULL, clim=NULL, cex=NULL, pch=NULL, ...)
## Default S3 method:
genome.plot(data, pch=NULL, cex=NULL, xlab="", ylab="", ...)
```

Arguments

- `arrayCGH`  
an object of type `arrayCGH`  
- `data`  
a data frame with two columns: 'x' and 'y', and optionally a column `data$chrLim` giving the limits of each chromosome
- `x`  
a variable name from `arrayCGH$cloneValues` giving the order position of the clones along the genome (defaults to 'PosOrder')
- `y`  
a variable name from `arrayCGH$cloneValues` to be plotted along the genome (defaults to 'LogRatio')
- `chrLim`  
an optional variable name from `arrayCGH$cloneValues` giving the limits of each chromosome
- `col.var`  
a variable name from `arrayCGH$cloneValues` defining the color legend
- `clim`  
a numeric vector of length 2: color range limits (used if `col.var` is numeric)
- `cex`  
a numerical value giving the amount by which plotting text and symbols should be scaled relative to the default: see `par`
- `xlab`  
a title for the x axis: see `title`
- `ylab`  
a title for the y axis: see `title`
- `pch`  
either an integer specifying a symbol or a single character to be used as the default in plotting points: see `par`
- `...`  
further arguments to be passed to `plot`

Details

If `col.var` is a numeric variable, y colors are proportionnal to `col.var` values; if it is a character variable or a factor, one color is assigned to each different value of `col.var`. If `col.var` is NULL, colors are proportionnal to `y` values.

Note

People interested in tools for array-CGH analysis can visit our web-page: [http://bioinfo.curie.fr](http://bioinfo.curie.fr).

Author(s)

Pierre Neuvial, <manor@curie.fr>.

See Also

`flag`, `report.plot`
Examples

data(spatial)

## default color code: log-ratios
## Not run:
genome.plot(edge.norm, chrLim="LimitChr")

## End(Not run)

## color code determined by a qualitative variable: ZoneGNL (DNA copy number code)
edge.norm$cloneValues$ZoneGNL <- as.factor(edge.norm$cloneValues$ZoneGNL)
## Not run:
genome.plot(edge.norm, col.var="ZoneGNL")

## End(Not run)
## comparing profiles with and without normalization
## aggregate data without normalization (flags)
gradient.nonorm <- norm(gradient, flag.list=NULL, var="LogRatio",
FUN=median, na.rm=TRUE)
gradient.nonorm <- sort(gradient.nonorm)
## Not run:
genome.plot(gradient.nonorm, pch=20, main="Genomic profile without normalization",
chrLim="LimitChr")
x11()
genome.plot(gradient.norm, pch=20, main="Genomic profile with normalization",
chrLim="LimitChr")
## End(Not run)

html.report

Generate an HTML report of array normalization

Description

Create an HTML file with various illustrations of array normalization, including plots before and after normalization, and statistics about flagged spots and clones

Usage

## S3 method for class 'arrayCGH'
html.report(array.norm, array.nonorm=NULL, dir.out=".",
array.name=NULL, x="PosOrder", y=c("LogRatioNorm", "LogRatio"), chrLim=NULL,
ylim=NULL, zlim=NULL, clim=NULL, intensity=NULL, light=FALSE,
file.name="report", width=10, height=5, ...)

## Default S3 method:
html.report(spot.data, clone.data=NULL,
flag.data=NULL, quality.data=NULL, ...)
Arguments

array.norm: an object of type arrayCGH after normalization step
array.nonorm: an object of type arrayCGH after a normalization step with no flags
spot.data: a data.frame containing spot-level informations (e.g. arrayCGH\$arrayValues)
clone.data: a data.frame containing clone-level informations (e.g. arrayCGH\$cloneValues)
flag.data: a data.frame containing information about flags, with fields char, label, arg, count as generated by function flag.summary
quality.data: a data.frame containing information about quality scores with fields name, label, score as generated by function qscore.summary
dir.out: absolute path of a directory where the file is generated (defaults to the current directory)
array.name: name or identifier of the array
x: a variable name from arrayCGH\$cloneValues giving the order position of the clones along the genome (defaults to 'PosOrder')
y: a vector of one or two variable names to be passed to report.plot
chrLim: an optional variable name from arrayCGH\$cloneValues giving the limits of each chromosome
ylim: a numeric vector of length 2 to be passed to report.plot: y axis range of the genomic profile display
clim: a numeric vector of length 2 to be passed to report.plot: color range of the genomic profile
zlim: a numeric vector of length 2 to be passed to report.plot: color range for array image display
intensity: an optional list with names c("M.var", "A.var", "pred.var", "span"). The first 3 items specify existing variable names from arrayCGH\$arrayValues that will be used to draw a MA-plot. The last item is the value of the loess 'span'
light: boolean value: if (light), only the core of the html file is generated; if (!light), a complete html file is generated
file.name: file name of the generated report (defaults to "report")
width: plot width, in inches
height: plot height, in inches
...: further arguments to be passed to report.plot

details

This function creates an HTML report file showing - the array image and the genome representation before normalization (if array.nonorm is provided) and after normalization, and optionally a MA-plot - a table with information about the number of flagged spots for each flag, and the number of remaining spots after normalization - a table with information about various quality criteria for the array

Value

none
Note

People interested in tools for array-CGH analysis can visit our web-page: http://bioinfo.curie.fr.

Author(s)

Pierre Neuvial, <manor@curie.fr>.

See Also

flag.summary, report.plot

---

**import**

*Import raw file to an arrayCGH object*

**Description**

Load raw data from a text file coming from image analysis and convert it to a `arrayCGH` object, using additional information about the array design.

Supported file types are Genepix Results file (.gpr), outputs from SPOT, or any text file with appropriate fields "Row" and "Column" and specified array design.

**Usage**

```r
import(file, var.names=NULL, spot.names=NULL, clone.names=NULL, 
      type=c("default", "gpr", "spot"), id.rep=1, design=NULL, add.lines=FALSE, ...)
```

**Arguments**

- `file` a connection or character string giving the name of the file to import.
- `var.names` a vector of variables names used to compute the array design. If default is not overwritten, it is set to c("Block", "Column", "Row", "X", "Y") for gpr files, c("Arr.colx", "Arr.rowy", "Spot.colx", "Spot.rowy") for SPOT files, and c("Col", "Row") for other text files
- `spot.names` a list with spot-level variable names to be added to `arrayCGH$arrayValues` (only used in case of within-slide replicates)
- `clone.names` a list with clone-level variable names to be added to `arrayCGH$cloneValues` (only used in case of within-slide replicates)
- `type` a character value specifying the type of input file: currently .gpr files ("gpr"), spot files ("spot") and other text files with fields "Col" and "Row" ("default") are supported
- `id.rep` index of the replicate identifier (e.g. the name of the clone) in the vector(clone.names)
- `design` a numeric vector of length 4 specifying array design as number of blocks per column, number of blocks per row, number of columns by block, number of rows per block. This field is mandatory for "default" text files, optional for "gpr" files, and not used for "SPOT" files
add.lines  boolean value to handle the case when array design does not match number of lines. If TRUE, empty lines are added; if FALSE, execution is stopped

...  additional import parameters (e.g. 'sep=', or 'comment.char=') to be passed to read.delim function. Note that argument as.is=TRUE is always passed to read.delim, in order to avoid inappropriate conversion of character vectors to factors

Details

Mandatory elements of arrayCGH objects are the array design and the x and y absolute coordinates of each spot on the array. Output files from SPOT contain x and y relative coordinates of each spot within a block, as well as block coordinates on the array; one can therefore easily construct the corresponding arrayCGH object.

.gpr files currently only contain x and y relative coordinates of each spot within a block, and block index with no specification of the spatial block design: if block design is not specified by user, we compute it using the real pixel locations of each spot (X and Y variables in usual .gpr files)

If clone.names is provided, an additional data frame is created with clone-level information (e.g. clone names, positions, chromosomes, quality marks), aggregated from array-level information using the identifier specified by id.rep. This identifier is also added to the arrayCGH object created, with name 'id.rep'.

Due to space limitations, only the first 100 lines of sample 'gpr' and 'spot' files are given in the standard distribution of MANOR. Complete files are available at http://bioinfo.curie.fr/projects/manor/index.html

Value

an object of class arrayCGH

Note

People interested in tools for array-CGH analysis can visit our web-page: http://bioinfo.curie.fr.

Author(s)

Pierre Neuvial, <manor@curie.fr>.

See Also

arrayCGH

Examples

dir.in <- system.file("extdata", package="MANOR")

## import from 'spot' files
clone.names <- c("PosOrder", "Chromosome")
edge <- import(paste(dir.in, "/edge.txt", sep=""), type="spot", spot.names=spot.names, clone.names=clone.names, add.lines=TRUE)

# import from 'gpr' files
spot.names <- c("Clone", "FLAG", "TEST_B_MEAN", "REF_B_MEAN", "TEST_F_MEAN", "REF_F_MEAN", "ChromosomeArm")
clone.names <- c("Clone", "Chromosome", "Position", "Validation")

ac <- import(paste(dir.in, "/gradient.gpr", sep=""), type="gpr", spot.names=spot.names, clone.names=clone.names, sep="\t", comment.char="@", add.lines=TRUE)

MANOR-internal     Internal Functions for MANOR Package

Description

Internal functions not intended for direct calls by user.

Note

People interested in tools for array-CGH analysis can visit our web-page: http://bioinfo.curie.fr.

Author(s)

Pierre Neuvial, <manor@curie.fr>.

nem     Spatial Classification by EM algorithm

Description

The function nem computes spatial classification by EM algorithm.

Usage

## Default S3 method:
nem(LogRatio, Col, Row, nk=nk, beta=1, iters=2000, ...)
## S3 method for class 'arrayCGH'
nem(arrayCGH, variable, nk=5, beta=1, iters=2000, ...)
Arguments

- **LogRatio**: Vector that corresponds to the values to be classified.
- **Col**: Vector of columns coordinates.
- **Row**: Vector of rows coordinates.
- **nk**: Integer value corresponding to the number classes.
- **beta**: Scale parameter for importance of spatial information.
- **iters**: Maximum number of iterations allowed.
- **arrayCGH**: Object of class `arrayCGH`.
- **variable**: Variable that corresponds to the values to be classified.
- ... ...

Value

Either a data frame with the following added elements:

- **ZoneNem**: Vector of label zones.

or an object of class `arrayCGH` with the following elements added to the data.frame attribute `arrayValues`:

- **ZoneNem**: Vector of label zones.

Note

People interested in tools for array-CGH analysis can visit our web-page: [http://bioinfo.curie.fr](http://bioinfo.curie.fr).

Author(s)

Philippe Hup?, <manor@curie.fr>.

References


Examples

data(spatial) ## arrays with local spatial effects

## Plot of LogRatio measured on the array CGH
## Not run:
GLAD::arrayPlot(edge,"LogRatio", main="Log2-Ratio measured on the array..."
## Spatial trend of the scaled log-ratios (the variable "ScaledLogRatio"
## equals to the log-ratio minus the median value of the corresponding chromosome arm)
edgeTrend <- arrayTrend(edge, variable="ScaledLogRatio",
span=0.03, degree=1, iterations=3, family="symmetric")

## Classification with spatial constraint of the spatial trend
dgeNem <- nem(edgeTrend, variable="Trend")

### Examples

## Not run:
GLAD::arrayPlot(edgeTrend, variable="Trend", main="Spatial trend of the array CGH", bar="v")
## End(Not run)

### Arguments

- **arrayCGH**: an object of type arrayCGH
- **flag.list**: a list of objects of type flag
- **var**: a variable name (from arrayCGH$arrayValues) from which normalization coefficient has to be computed; default is "LogRatio"
- **printTime**: boolean value; if TRUE, the time taken by each step of the normalization process is displayed
- **FUN**: an aggregation function (e.g. mean, median) to compute a normalization coefficient; default is median
- **...**: further arguments to be passed to FUN

### Description

Normalize an object of type arrayCGH using a list of criteria specified as (temporary or permanent) flags. If a replicate identifier (clone name) is provided, a single target value is computed for all the replicates.

The normalization coefficient is computed as a function, and is applied to all good quality spots of the array.

### Usage

```r
## S3 method for class 'arrayCGH'
norm(arrayCGH, flag.list=NULL, var="LogRatio", printTime=FALSE, FUN=median, ...)
```

### Examples

```r
edge <- arrayCGH(zlim=c(-1,1), bar="v", mediancenter=TRUE)
## End(Not run)

## Not run:
GLAD::arrayPlot(edge, variable="LogRatio", main="Spatial trend of the array CGH", bar="v")
## End(Not run)
```
Details

The two flag types are treated differently: - permanent flags detect poor quality spots, which are removed from further analysis - temporary flags detect good quality spots that would bias the normalization coefficient if they were not excluded from its computation, e.g. amplicons or sexual chromosomes. Thus they are not taken into account for the computation of the coefficient, but at the end of the analysis they are normalized as any good quality spots of the array.

The normalization coefficient is computed as a function (e.g. mean or median) of the target value (e.g. log-ratios); it is then applied to all good quality spots (including temporary flags), i.e. subtracted from each target value.

If clone level information is available (i.e. if arrayCGH$cloneValues is not null), a normalized mean target value and basic statistics (such as the number of replicates per clone) are calculated for each clone.

Value

an object of type arrayCGH

Note

People interested in tools for array-CGH analysis can visit our web-page: http://bioinfo.curie.fr.

Author(s)
Pierre Neuvial, <manor@curie.fr>.

References


See Also

flag

Examples

data(spatial)
data(flags)

### 'edge': local spatial bias
## define a list of flags to be applied
flag.list1 <- list(spatial=local.spatial.flag, spot=spot.corr.flag,
ref.snr=ref.snr.flag, dapi.snr=dapi.snr.flag, rep=rep.flag,
unique=unique.flag)
flag.list1$spatial$args <- alist(var="ScaledLogRatio", by.var=NULL,
nk=5, prop=0.25, thr=0.15, beta=1, family="symmetric")
flag.list1$spot$args <- alist(var="SpotFlag")
flag.list1$spot$char <- "O"
flag.list1$spot$label <- "Image analysis"
## normalize arrayCGH
edge.norm <- norm(edge, flag.list=flag.list1, var="LogRatio", FUN=median, na.rm=TRUE)
print(edge.norm$flags) ## spot-level flag summary (computed by flag.summary)

## aggregate arrayCGH without normalization
edge.nonorm <- norm(edge, flag.list=NULL, var="LogRatio", FUN=median, na.rm=TRUE)

## sort genomic informations
edge.norm <- sort(edge.norm, position.var="PosOrder")
edge.nonorm <- sort(edge.nonorm, position.var="PosOrder")

## plot genomic profiles
layout(matrix(c(1,2,4,5,3,6,6), 4,2),width=c(1, 4), height=c(6,1,6,1))
report.plot(edge.nonorm, chrLim="LimitChr", layout=FALSE, main="Pangenomic representation (before normalization)", zlim=c(-1,1), ylim=c(-3,1))
report.plot(edge.norm, chrLim="LimitChr", layout=FALSE, main="Pangenomic representation (after normalization)", zlim=c(-1,1), ylim=c(-3,1))

### 'gradient': global array Trend
## define a list of flags to be applied
flag.list2 <- list(  
    spot=spot.flag, global.spatial=global.spatial.flag, SNR=SNR.flag,  
    val.mark=val.mark.flag, position=position.flag, unique=unique.flag,  
    amplicon=amplicon.flag, replicate=replicate.flag,  
    chromosome=chromosome.flag)

## normalize arrayCGH
gradient.norm <- norm(gradient, flag.list=flag.list2, var="LogRatio", FUN=median, na.rm=TRUE)

## aggregate arrayCGH without normalization
gradient.nonorm <- norm(gradient, flag.list=NULL, var="LogRatio", FUN=median, na.rm=TRUE)

## sort genomic informations
gradient.norm <- sort(gradient.norm)
gradient.nonorm <- sort(gradient.nonorm)

## plot genomic profiles
layout(matrix(c(1,2,4,5,3,6,6), 4,2),width=c(1, 4), height=c(6,1,6,1))
report.plot(gradient.nonorm, chrLim="LimitChr", layout=FALSE, main="Pangenomic representation (before normalization)", zlim=c(-2,2), ylim=c(-3,2))
report.plot(gradient.norm, chrLim="LimitChr", layout=FALSE, main="Pangenomic representation (after normalization)", zlim=c(-2,2), ylim=c(-3,2))
**qscore.arrayCGH**

**Description**

qscore object is a list which contains a function, a name, and optionally a label and arguments to be passed to the function.

**Usage**

```r
to.qscore(FUN, name=NULL, args=NULL, label=NULL, dec=3)
```

**Arguments**

- **FUN**: a R function returning a numeric value, with first argument of type `arrayCGH`, and optionally other arguments.
- **name**: a short character value for qscore object identification
- **args**: a list of arguments to be passed to FUN; defaults to NULL (ie `arrayCGH` is the only argument to FUN)
- **label**: a character value for qscore object labelling
- **dec**: an integer value giving the number of significant digits to keep (defaults to 3)

**Value**

An object of class qscore.

**Note**

People interested in tools for array-CGH analysis can visit our web-page: [http://bioinfo.curie.fr](http://bioinfo.curie.fr).

**Author(s)**

Pierre Neuvial, `<manor@curie.fr>`.

**See Also**

- `qscore.arrayCGH`, `qscore.summary.arrayCGH`
Arguments

- `qscore` an object of type `qscore`.
- `arrayCGH` an object of type `arrayCGH`.

Value

A numeric value.

Note

People interested in tools for array-CGH analysis can visit our web-page: http://bioinfo.curie.fr.

Author(s)

Pierre Neuvial, <manor@curie.fr>.

See Also

`qscore`, `qscore.summary`

Examples

```r
data(qscores)
data(spatial)

## compute a quality score for a couple of arrays: signal smoothness
qscore.arrayCGH(smoothness.qscore, edge.norm)
qscore.arrayCGH(smoothness.qscore, gradient.norm)
```

---

**qscore.summary**

*Compute quality scores for a given arrayCGH object*

Description

Compute useful quality scores for the `arrayCGH` and display them in a convenient way.

Usage

```r
qscore.summary.arrayCGH(arrayCGH, qscore.list)
```

Arguments

- `arrayCGH` an object of type `arrayCGH`
- `qscore.list` a list of objects of type `qscore`
Details

This function is used by the function html.report for the generation of an HTML report of the normalization step. It can also be used by itself.

Value

A data.frame with 3 columns:

- name  qscore name
- label  qscore label
- qscore  quality qscore

Note

People interested in tools for array-CGH analysis can visit our web-page: http://bioinfo.curie.fr.

Author(s)

Pierre Neuvial, <manor@curie.fr>.

See Also

qscore, qscore.summary, html.report

Examples

```r
data(qscores)
data(spatial)

## define a list of qscores
qscore.list <- list(clone=clone.qscore, pct.clone=pct.clone.qscore, pct.spot=pct.spot.qscore, pct.replicate=pct.replicate.qscore, smoothness=smoothness.qscore, dyn.x=dyn.x.qscore, dyn.y=dyn.y.qscore, var.replicate=var.replicate.qscore)

## compute quality scores for a couple of normalized arrays
gradient.norm$quality <- qscore.summary.arrayCGH(gradient.norm, qscore.list)
print(gradient.norm$quality[, 2:3])

qscore.list$dyn.x$args$test <- 23
qscore.list$dyn.y$args$test <- 24
edge.norm$quality <- qscore.summary.arrayCGH(edge.norm, qscore.list)
print(edge.norm$quality[, 2:3])
```
**qscores**

Examples of qscore objects (quality scores) to apply to CGH arrays

**Description**

This data set provides qscore objects that can be applied to normalized arrayCGH objects in order to evaluate data quality after normalization.

**Usage**

```r
data(qscores)
```

**Format**

The following qscore objects are provided:

- `clone.qscore`  number of clones
- `pct.clone.qscore`  percentage of clones
- `pct.spot.qscore`  percentage of spots
- `pct.spot.before.qscore`  percentage of spots before normalization
- `pct.replicate.qscore`  average percentage of replicates
- `smoothness.qscore`  signal smoothness
- `var.replicate.qscore`  signal smoothness
- `dyn.x.qscore`  signal dynamics on X chromosome
- `dyn.y.qscore`  signal dynamics on Y chromosome

**Note**

People interested in tools for array-CGH analysis can visit our web-page: [http://bioinfo.curie.fr](http://bioinfo.curie.fr).

**Author(s)**

Pierre Neuvial, <manor@curie.fr>.

**Source**

Institut Curie, <manor@curie.fr>.

**See Also**

`spatial.qscore.summary.arrayCGH`, `qscore`
**Examples**

```r
data(qscores)
data(spatial)

## define a list of qscores
qscore.list <- list(clone=clone.qscore, pct.clone=pct.clone.qscore,
pct.spot=pct.spot.qscore, pct.replicate=pct.replicate.qscore,
smoothness=smoothness.qscore, dyn.x=dyn.x.qscore, dyn.y=dyn.y.qscore,
var.replicate=var.replicate.qscore)

## compute quality scores for a couple of normalized arrays
gradient.norm$quality <- qscore.summary.arrayCGH(gradient.norm,
qscore.list)
print(gradient.norm$quality[, 2:3])

qscore.list$dyn.x$args$test <- 23
qscore.list$dyn.y$args$test <- 24
edge.norm$quality <- qscore.summary.arrayCGH(edge.norm, qscore.list)
print(edge.norm$quality[, 2:3])
```

---

**report.plot**

*Array image and a genomic representation of a normalized arrayCGH*

**Description**

Displays an array image and a genomic representation of a normalized arrayCGH.

**Usage**

```r
## S3 method for class 'arrayCGH'
report.plot(arrayCGH, x="PosOrder", y=c("LogRatioNorm", "LogRatio"), chrLim=NULL, layout=TRUE, main=NULL, zlim=NULL, ...)

## Default S3 method:
report.plot(spot.data, clone.data, design, x="PosOrder",
y=c("LogRatioNorm", "LogRatio"), chrLim=NULL, layout=TRUE, main=NULL, zlim=NULL, ...)  
```

**Arguments**

- `arrayCGH` an object of type `arrayCGH`.
- `spot.data` data.frame with spot-level information to be passed to `arrayPlot`.
- `clone.data` data.frame with clone-level information to be passed to `genomePlot`.
- `design` vector of length 4 with array design: number of blocks per column and per row, number of columns and rows per block.
- `x` a variable name from `arrayCGH(cloneValues)` giving the order position of the clones along the genome.
y a vector of one or two variable names to be plotted on the array and along the genome. The first one is taken from `arrayCGH\$arrayValues` and is plotted on the array; the second one (or the first one if only one name was provided) is taken from `arrayCGH\$cloneValues` and is plotted along the genome.

chrLim an optional variable name from `arrayCGH\$cloneValues` giving the limits of each chromosome.

layout if TRUE, plot layout is set to a 1*2 matrix with relative column widths 1 and 4.

main title for the genomic profile.

zlim numeric vector of length 2 to be passed to `arrayPlot`: minimum and maximum signal values for array image display.

... further arguments to be passed to `genome.plot`.

Details
This function successively calls `arrayPlot` and `genome.plot`.

Note
People interested in tools for array-CGH analysis can visit our web-page: [http://bioinfo.curie.fr](http://bioinfo.curie.fr).

Author(s)
Pierre Neuvial, <manor@curie.fr>.

See Also
`genome.plot`, `arrayPlot`, `html.report`

Examples
```r
data(spatial)
### edge: local spatial bias
## aggregate arrayCGH without normalization for comparison with
## normalized array
edge.nonorm <- norm(edge, flag.list=,NULL, FUN=median, na.rm=TRUE)
edge.nonorm <- sort(edge.nonorm, position.var="PosOrder")

layout(matrix(c(1,2,4,5,3,3,6,6), 4,2),width=c(1, 4), height=c(6,1,6,1))
report.plot(edge.nonorm, chrLim="LimitChr", layout=FALSE,
main="Pangenomic representation (before normalization)", zlim=c(-1,1),
ylim=c(-3,1))
report.plot(edge.norm, chrLim="LimitChr", layout=FALSE,
main="Pangenomic representation (after normalization)", zlim=c(-1,1),
ylim=c(-3,1))

### gradient: global array Trend
## aggregate arrayCGH without normalization for comparison with
## normalized array
```
sort

```r
gradient.nonorm <- norm(gradient, flag.list=NULL, FUN=median, na.rm=TRUE)
gradient.nonorm <- sort(gradient.nonorm)

layout(matrix(c(1,2,4,5,3,3,6,6), 4,2),width=c(1, 4), height=c(6,1,6,1))
report.plot(gradient.nonorm, chrLim="LimitChr", layout=FALSE,
            main="Pangenomic representation (before normalization)", zlim=c(-2,2),
            ylim=c(-3,2))
report.plot(gradient.norm, chrLim="LimitChr", layout=FALSE,
            main="Pangenomic representation (after normalization)", zlim=c(-2,2),
            ylim=c(-3,2))
```

---

**sort**

Sorting for normalized arrayCGH objects

### Description

Sorts clone-level information of a normalized arrayCGH object.

### Usage

```r
## S3 method for class 'arrayCGH'
sort(x, decreasing = FALSE, position.var="Position",
     chromosome.var="Chromosome", ...)
```

### Arguments

- **x**: an object of type arrayCGH.
- **decreasing**: (for compatibility with `sort` class) currently unused.
- **position.var**: name of position variable.
- **chromosome.var**: name of chromosome variable.
- **...**: further arguments to be passed to `sort`.

### Note

People interested in tools for array-CGH analysis can visit our web-page: [http://bioinfo.curie.fr](http://bioinfo.curie.fr).

### Author(s)

Pierre Neuvial, <manor@curie.fr>.

### See Also

- `norm.arrayCGH`
Examples

```r
data(spatial)

## sort a normalized array by clone position
gradient.norm <- sort(gradient.norm)

report.plot(gradient.norm, main="Genomic profile after normalization")
```

---

spatial Examples of array-CGH data with spatial artifacts

Description

This data set provides an example of array-CGH data with spatial artifacts, consisting of including arrayCGH objects before and after normalization.

Usage

```r
data(spatial)
```

Format

- `edge, gradient` arrayCGH objects before normalization:
  - `arrayValues` spot-level information
  - `arrayDesign` block design of the array
  - `cloneValues` additional clone-level data (chromosome, position)

- `edge.norm, gradient.norm` arrayCGH objects after normalization

Details

'edge' presents local spatial bias in the top-right edge corner, and 'gradient' presents global spatial trend. 'edge' and 'gradient' are arrayCGH objects before normalization. They have been created respectively from spot and gpr files using `import`. 'edge.norm' and 'gradient.norm' are the corresponding arrayCGH objects after normalization using `norm.arrayCGH`. flag objects used for data normalization come from flags dataset.

Note

People interested in tools for array-CGH analysis can visit our web-page: [http://bioinfo.curie.fr](http://bioinfo.curie.fr).

Author(s)

Pierre Neuvial, <manor@curie.fr>.
to.flag

Create an object of type flag

description

A flag object is a list which contains essentially a function (flag action) and a character, optionally arguments to be passed to the function. We make the distinction between two different flag types, corresponding to two different purposes: - permanent flags identify poor quality spots or clones and remove them from further analysis (eg spots with low signal to noise ratio) - temporary flags identify spots or clones that have not to be taken into account for the computation of a (scaling) normalization coefficient (eg X chromosome in case of sex mismatch)

usage

to.flag(FUN, char=NULL, args=NULL, type="perm.flag", label=NULL)

arguments

FUN a R function to be applied to an arrayCGH, and optionally other arguments. If char is not NULL, must return a list of spots (lines of arrayCGH$arrayValues) to be flagged out; if char==NULL, must return an object of type arrayCGH

Examples
data(spatial)

## edge: example of array with local spatial effects

layout(matrix(1:4, 2, 2), height=c(9,1))
GLAD::arrayPlot(edge, "LogRatio", main="log-ratios before normalization", zlim=c(-1,1), bar="h", layout=FALSE, mediancenter=TRUE)
GLAD::arrayPlot(edge.norm, "LogRatioNorm", main="log-ratios after spatial normalization", zlim=c(-1,1), bar="h", layout=FALSE, mediancenter=TRUE)

## gradient: example of array with spatial gradient

layout(matrix(1:4, 2, 2), height=c(9,1))
GLAD::arrayPlot(gradient, "LogRatio", main="Log-ratios before normalization", zlim=c(-2,2), bar="h", layout=FALSE)
GLAD::arrayPlot(gradient.norm, "LogRatioNorm", main="log-ratios after spatial normalization", zlim=c(-2,2), bar="h", layout=FALSE)
to.flag

char a character value to identify flagged spots; defaults to NULL.
args a list of further arguments to be passed to FUN; defaults to NULL (ie arrayCGH is the only argument to FUN)
type a character value defaulting to "perm.flag" which makes the distinction between permanent flags (type="perm.flag") and temporary flags (type="temp.flag")
label a character value for flag labelling

Details
If flag$char is null, flag$FUN is supposed to return a arrayCGH object; if it is not null, flag$FUN is supposed to return a list of spots to be flagged with flag$char.

Value
An object of class flag.

Note
People interested in tools for array-CGH analysis can visit our web-page: http://bioinfo.curie.fr.

Author(s)
Pierre Neuvial, <manor@curie.fr>.

See Also
flag.arrayCGH, norm.arrayCGH

Examples
### creation of a permanent flag:
## flag spots with low signal to noise ratios
SNR.FUN <- function(arrayCGH, snr.thr)
  which(arrayCGH$arrayValues$F2 < arrayCGH$arrayValues$B2+log(snr.thr, 2))
SNR.char <- "B"
SNR.flag <- to.flag(SNR.FUN, SNR.char, args=alist(snr.thr=3))

### creation of a permanent flag returning an arrayCGH object:
## correct log-ratios for spatial trend
global.spatial.FUN <- function(arrayCGH, var)
{
  Trend <- arrayTrend(arrayCGH, var, span=0.03, degree=1,
                     iterations=3, family="symmetric")
  arrayCGH$arrayValues[[var]] <- Trend$arrayValues[[var]]-Trend$arrayValues$Trend
  arrayCGH
}
global.spatial.flag <- to.flag(global.spatial.FUN, args=alist(var="LogRatio"))

### creation of a temporary flag:
## exclude sexual chromosomes from signal scaling

```r
cromosome.FUN <- function(arrayCGH, var)
    which(!is.na(match(as.character(arrayCGH$arrayValues[[var]]), c("X", "Y"))))
cromosome.char <- "X"
cromosome.flag <- to.flag(chromosome.FUN, chromosome.char, type="temp.flag",
    args=alist(var="Chromosome"))

data(spatial)

SNR.flag$args$snr.thr <- 3  # set SNR threshold
gradient <- flag.arrayCGH(SNR.flag, gradient)  # apply SNR.flag to array CGH

gradient <- flag.arrayCGH(global.spatial.flag, gradient)
gradient <- flag.arrayCGH(chromosome.flag, gradient)

summary.factor(gradient$arrayValues$Flag)  # permanent flags
summary.factor(gradient$arrayValues$FlagT)  # temporary flags
```
## Index

### * IO
- html.report, 12
- import, 14

### * datasets
- flags, 9
- qscores, 24
- spatial, 28

### * file
- import, 14

### * hplot
- genome.plot, 10
- report.plot, 25

### * internal
- MANOR-internal, 16

### * loess
- arrayTrend, 2

### * misc
- flag.arrayCGH, 6
- flag.summary, 7
- qscore, 21
- qscore.arrayCGH, 21
- qscore.summary, 22
- to.flag, 29

### * models
- detectSB, 4
- nem, 16
- norm, 18

### * smooth
- arrayTrend, 2

### * spatial
- arrayTrend, 2
- detectSB, 4
- nem, 16

### * utilities
- sort, 27

add.lines (MANOR-internal), 16
amplicon.flag (flags), 9
arrayCGH, 2, 4, 6, 14, 15, 17, 21, 28–30
arrayCGH2txt (MANOR-internal), 16
arrayPlot, 26
arrayTrend, 2, 4, 5
center (MANOR-internal), 16
check.names (MANOR-internal), 16
chrom (MANOR-internal), 16
chromosome.flag (flags), 9
clone.qscore (qscores), 24
control.flag (flags), 9
cv (MANOR-internal), 16
dapi.snr.flag (flags), 9
detectSB, 4
dyn.x.qscore (qscores), 24
dyn.y.qscore (qscores), 24
dynamics.qscore (qscores), 24
detectSB, 4
dyn.x.qscore (qscores), 24
dyn.y.qscore (qscores), 24
dynamics.qscore (qscores), 24
decided (spatial), 28
flag, 8–11, 19, 28
flag (flag.arrayCGH), 6
flag.aggregate.arrayCGH (MANOR-internal), 16
flag.arrayCGH, 6, 30
flag.summary, 7, 10, 13, 14
flags, 9, 28, 29
genome.plot, 10, 26
getChromosomeArm (MANOR-internal), 16
global.spatial.flag (flags), 9
gradient (spatial), 28
html.report, 8, 12, 23, 26
import, 14, 28
intensity.flag (flags), 9
intensity.plot (MANOR-internal), 16
local.spatial.flag (flags), 9
loess, 2, 3
loess.control, 3
INDEX

MANOR (norm), 18
manor (norm), 18
MANOR-internal, 16
my.aggregate.data.frame
   (MANOR-internal), 16

nem, 4, 5, 16
norm, 18
norm.arrayCGH, 6, 10, 27, 28, 30
num.chromosome.flag (MANOR-internal), 16

par, 11
pct.clone.qscore (qscores), 24
pct.replicate.qscore (qscores), 24
pct.spot.before.qscore (qscores), 24
pct.spot.qscore (qscores), 24
position.flag (flags), 9
print.flag (MANOR-internal), 16

qscore, 20, 22–24
qscore.arrayCGH, 21
qscore.summary, 13, 22, 22, 23
qscore.summary.arrayCGH, 21, 24
qscores, 24

ref.snr.flag (flags), 9
rename (MANOR-internal), 16
rep.flag (flags), 9
replicate.aux (MANOR-internal), 16
replicate.flag (flags), 9
report.plot, 11, 13, 14, 25

smoothness.qscore (qscores), 24
SNR.flag (flags), 9
sort, 27
spatial, 10, 24, 28
spatial.flag (flags), 9
spot.corr.flag (MANOR-internal), 16
spot.flag (flags), 9

title, 11
to.flag, 6, 29
to.qscore (qscores), 21

unique.flag (flags), 9

val.mark.flag (flags), 9
var.replicate.qscore (qscores), 24

w (MANOR-internal), 16