

# MANOR

April 20, 2011

---

arrayTrend                      *Spatial trend of microarray spots statistic*

---

## Description

The function `arrayTrend` computes the spatial trend.

## Usage

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

## Arguments

<code>Statistic</code>	Statistic to be smoothed.
<code>Col</code>	Vector of columns coordinates.
<code>Row</code>	Vector of rows coordinates.
<code>arrayCGH</code>	Object of class <code>arrayCGH</code> .
<code>variable</code>	Variable to be smooth.
<code>...</code>	Parameters to be passed to <code>loess</code> function.

## Details

Spatial trend of microarray spots statistic.

## Value

Either a data frame with elements :

<code>Trend</code>	Trend fitted by <code>loess</code> function.
<code>Col</code>	Vector of columns coordinates.
<code>Row</code>	Vector of row 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>.

**Author(s)**

Philippe Hupé, <Philippe.Hupe@curie.fr>.

**References**

P. Neuvial, P. Hupé, I. Brito, S. Liva, E. Manié, C. Brennetot, A. Aurias, F. Radvanyi, and E. Barillot. *Spatial normalization of array-CGH data*. BMC Bioinformatics, 7(1):264. May 2006.

**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")
arrayPlot(edgeTrend, "Trend", main="Spatial trend of array CGH", bar="v")
```

---

detectSB

*Spatial bias detection*

---

**Description**

This function detects spatial bias on array CGH.

**Usage**

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

**Arguments**

arrayCGH	Object of <a href="#">arrayCGH</a> .
variable	Variable used to compare the mean of zones detected by <a href="#">nem</a>
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).
thresholddup	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)

Philippe Hupé, <Philippe.Hupe@.curie.fr>.

### References

P. Neuvial, P. Hupé, I. Brito, S. Liva, E. Manié, C. Brennetot, A. Aurias, F. Radvanyi, and E. Barillot. *Spatial normalization of array-CGH data*. BMC Bioinformatics, 7(1):264. May 2006.

### See Also

`arrayTrend`, `nem`

### Examples

```

data(spatial)  ## arrays with local spatial effects

## Plot of LogRatio measured on the array CGH
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")
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")
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", thresholdu
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)

```

---

flag.arrayCGH      *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>.

### 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 spots excluded by Genepix pro
gradient <- flag.arrayCGH(spot.flag, gradient)

## flag local spatial bias zones
## Not run: gradient <- flag.arrayCGH(local.spatial.flag, gradient)

## correct global spatial bias
gradient <- flag.arrayCGH(global.spatial.flag, gradient)

## flag spots with low signal to noise
gradient <- flag.arrayCGH(SNR.flag, gradient)

## flag spots with extremely high log-ratios
gradient <- flag.arrayCGH(amplicon.flag, gradient)

## flag spots with poor within replicate consistency
gradient <- flag.arrayCGH(replicate.flag, gradient)

## flag spots corresponding to clones for which all other spot
## replicates have already been flagged
gradient <- flag.arrayCGH(unique.flag, gradient)

summary.factor(gradient$arrayValues$Flag)

```

---

 flags

*Examples of flag objects to apply to CGH arrays*


---

**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:

<code>amplicon.flag</code>	flags spots with high log-ratios (temp flag)
<code>chromosome.flag</code>	flags spots located on sexual chromosomes (named "X" and "Y")

<code>control.flag</code>	flag control spots
<code>global.spatial.flag</code>	corrects arrayCGH from global spatial trend on the array
<code>local.spatial.flag</code>	flags spots belonging to local spatial bias zones on the array
<code>num.chromosome.flag</code>	flags spots located on sexual chromosomes (named 23 and 24)
<code>position.flag</code>	flag spots with no available genome position
<code>replicate.flag</code>	flag spots with poor within-clone-replicate consistency
<code>ref.snr.flag</code>	flags spots with low signal to noise ratio for reference
<code>dapi.snr.flag</code>	flags spots with low signal to noise ratio for DAPI
<code>SNR.flag</code>	flags spots with low signal to noise ratio
<code>spot.corr.flag</code>	flags spots with low correlation coefficient after image analysis
<code>spot.flag</code>	flags spots excluded by the image analysis software
<code>unique.flag</code>	exclude last non-flagged spot of a clone
<code>val.mark.flag</code>	flags spots corresponding to bad quality clones
<code>intensity.flag</code>	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)

```

---

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", nflab="not flagged", ...)
## Default S3 method:
flag.summary(spot.flags, flag.list, nflab="not flagged", ...)

```

## Arguments

arrayCGH	an object of type arrayCGH, <i>after normalization</i> 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:

name	flag character
label	flag label
arg	first numeric argument of flag\$FUN
count	number of flagged spots

**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 <- "O"
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")

print("Flag and normalization parameters summary")
print(fs)
```

---

genome.plot

*Pan-genomic representation of a normalized arrayCGH*

---

**Description**

Displays a pan-genomic representation of a normalized arrayCGH.



**Usage**

```
## 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 <a href="#">par</a>
xlab	a title for the x axis: see <a href="#">title</a>
ylab	a title for the y axis: see <a href="#">title</a>
pch	either an integer specifying a symbol or a single character to be used as the default in plotting points: see <a href="#">par</a>
...	further arguments to be passed to <a href="#">plot</a>

**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>.

**Author(s)**

Pierre Neuvial, <[manor@curie.fr](mailto: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**

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

**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 an `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**

```
import(file, var.names=NULL, spot.names=NULL, clone.names=NULL, type=c("default"))
```

**Arguments**

<code>file</code>	a connection or character string giving the name of the file to import.
<code>var.names</code>	a vector of variables names used to compute the array design. If default is not overwritten, it is set to <code>c("Block", "Column", "Row", "X", "Y")</code> for gpr files, <code>c("Arr.colx", "Arr.rowy", "Spot.colx", "Spot.rowy")</code> for SPOT files, and <code>c("Col", "Row")</code> for other text files
<code>spot.names</code>	a list with spot-level variable names to be added to <code>arrayCGH\$arrayValues</code>
<code>clone.names</code>	a list with clone-level variable names to be added to <code>arrayCGH\$cloneValues</code> (only used in case of within-slide replicates)
<code>type</code>	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
<code>id.rep</code>	index of the replicate identifier (e.g. the name of the clone) in the vector( <code>clone.names</code> )
<code>design</code>	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
<code>add.lines</code>	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
<code>...</code>	additional import parameters (e.g. <code>'sep='</code> , or <code>'comment.char='</code> , to be passed to <code>read.delim</code> function. Note that argument <code>as.is=TRUE</code> is always passed to <code>read.delim</code> , in order to avoid unappropriate 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("data", package="MANOR")

## import from 'spot' files
spot.names <- c("LogRatio", "RefFore", "RefBack", "DapiFore", "DapiBack", "SpotFlag", "So
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)
```

---

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

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

## Value

Either a data frame with the following added elements:

<code>ZoneNem</code>	Vector of label zones.
----------------------	------------------------

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

<code>ZoneNem</code>	Vector of label zones.
----------------------	------------------------

## Note

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

## Author(s)

Philippe Hupé, <[manor@curie.fr](mailto:manor@curie.fr)>.

## References

- C. Ambroise, *Approche probabiliste en classification automatique et contraintes de voisinage*, Ph.D. thesis, Université de Technologie de Compiègne, 1996.
- C. Ambroise, M. Dang, and G. Govaert, *Clustering of spatial data by the em algorithm* in *Geostatistics for Environmental Applications*, A. Soares, J. Gomez-Hernandez, and R. Froidevaux, Eds., pp. 493-504. Kluwer Academic Publisher, 1997.
- P. Neuvial, P. Hupé, I. Brito, S. Liva, E. Manié, C. Brennetot, A. Aurias, F. Radvanyi, and E. Barillot. *Spatial normalization of array-CGH data*. *BMC Bioinformatics*, 7(1):264. May 2006.

## Examples

```
data(spatial) ## arrays with local spatial effects

## Plot of LogRatio measured on the array CGH
## Not run:
arrayPlot(edge, "LogRatio", main="Log2-Ratio measured on the array
CGH", zlim=c(-1,1), bar="v", mediancenter=TRUE)

## End(Not run)

## 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")

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

## End(Not run)

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

## End(Not run)
```

---

norm

*Normalize an object of type arrayCGH*


---

## 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

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

### Arguments

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

### 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](mailto:manor@curie.fr)>.

### References

P. Neuvial, P. Hupé, I. Brito, S. Liva, E. Manié, C. Brennetot, A. Aurias, F. Radvanyi, and E. Barillot. *Spatial normalization of array-CGH data*. BMC Bioinformatics, 7(1):264. May 2006.

### 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
## Not run: edge.norm <- norm(edge, flag.list=flag.list1,
var="LogRatio", FUN=median, na.rm=TRUE)
## End(Not run)
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(1,2,4,5,3,3,6,6), 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
## Not run: gradient.norm <- norm(gradient, flag.list=flag.list2, var="LogRatio", FUN=med
## 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(1,2,4,5,3,3,6,6), 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)", xlim=c(-2,2),
ylim=c(-3,2))
report.plot(gradient.norm, chrLim="LimitChr", layout=FALSE,
main="Pangenomic representation (after normalization)", xlim=c(-2,2),
ylim=c(-3,2))
```

---

qscore.arrayCGH      *arrayCGH quality score*

---

## Description

Computes a quality score for a given arrayCGH.

## Usage

```
qscore.arrayCGH(qscore, arrayCGH)
```

## Arguments

qscore	an object of type <code>qscore</code> .
arrayCGH	an object of type <code>arrayCGH</code> .

## 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

```
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 *Create an object of type qscore*

---

## Description

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

## Usage

```
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 <code>arrayCGH</code> , 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 <code>arrayCGH</code> 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>.

## Author(s)

Pierre Neuvial, <[manor@curie.fr](mailto:manor@curie.fr)>.

## See Also

[qscore.arrayCGH](#), [qscore.summary.arrayCGH](#)

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**

```
data(qscores)
```

**Format**

The following `qscore` objects are provided:

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

**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](#), [qscore.summary.arrayCGH](#), [qscore](#)

**Examples**

```
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])
```

---

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

```
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**

```
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**

```
## 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 genome.plot.
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 <code>arrayCGH\$arrayValues</code> and is plotted on the array; the second one (or the first one if only one name was provided) is taken from <code>arrayCGH\$cloneValues</code> and is plotted along the genome.
chrLim	an optional variable name from <code>arrayCGH\$cloneValues</code> 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 <code>arrayPlot</code> : minimum and maximum signal values for array image display.
...	further arguments to be passed to <code>genome.plot</code> .

### 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>.

### Author(s)

Pierre Neuvial, <[manor@curie.fr](mailto:manor@curie.fr)>.

### See Also

[genome.plot](#), [arrayPlot](#), [html.report](#)

### Examples

```
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
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.nonnorm, chrLim="LimitChr", layout=FALSE,
main="Pangenomic representation (before normalization)", xlim=c(-2,2),
ylim=c(-3,2))
report.plot(gradient.norm, chrLim="LimitChr", layout=FALSE,
main="Pangenomic representation (after normalization)", xlim=c(-2,2),
ylim=c(-3,2))
```

---

sort

*Sorting for normalized arrayCGH objects*

---

## Description

Sorts clone-level information of a normalized arrayCGH object.

## Usage

```
## 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>.

## Author(s)

Pierre Neuvial, <manor@curie.fr>.

## See Also

[norm.arrayCGH](#)

## Examples

```
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**

```
data(spacial)
```

**Format**

- `edge`, `gradientarrayCGH` objects before normalization:

<code>arrayValues</code>	spot-level information
<code>arrayDesign</code>	block design of the array
<code>cloneValues</code>	additional clone-level data (chromosome, position)

- `edge.norm`, `gradient.normarrayCGH` 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>.

**Author(s)**

Pierre Neuvial, <[manor@curie.fr](mailto:manor@curie.fr)>.

**Source**

Institut Curie, <[manor@curie.fr](mailto:manor@curie.fr)>.

**See Also**

[flags](#)

## Examples

```

data(spatial)

## edge: example of array with local spatial effects

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

```

---

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

<code>FUN</code>	a R function to be applied to an <a href="#">arrayCGH</a> , and optionally other arguments. If <code>char</code> is not <code>NULL</code> , must return a list of spots (lines of <code>arrayCGH\$arrayValues</code> ) to be flagged out; if <code>char==NULL</code> , must return an object of type <a href="#">arrayCGH</a>
<code>char</code>	a character value to identify flagged spots; defaults to <code>NULL</code>
<code>args</code>	a list of further arguments to be passed to <code>FUN</code> ; defaults to <code>NULL</code> (ie <a href="#">arrayCGH</a> is the only argument to <code>FUN</code> )
<code>type</code>	a character value defaulting to "perm.flag" which makes the distinction between permanent flags ( <code>type="perm.flag"</code> ) and temporary flags ( <code>type="temp.flag"</code> )
<code>label</code>	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
chromosome.FUN <- function(arrayCGH, var)
  which(!is.na(match(as.character(arrayCGH$arrayValues[[var]]), c("X", "Y"))))
chromosome.char <- "X"
chromosome.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

## \*Topic **IO**

html.report, 10  
import, 12

## \*Topic **datasets**

flags, 5  
qscores, 20  
spatial, 25

## \*Topic **file**

import, 12

## \*Topic **hplot**

genome.plot, 8  
report.plot, 22

## \*Topic **loess**

arrayTrend, 1

## \*Topic **misc**

flag.arrayCGH, 4  
flag.summary, 7  
qscore, 19  
qscore.arrayCGH, 18  
qscore.summary, 21  
to.flag, 26

## \*Topic **models**

detectSB, 2  
nem, 14  
norm, 15

## \*Topic **smooth**

arrayTrend, 1

## \*Topic **spatial**

arrayTrend, 1  
detectSB, 2  
nem, 14

## \*Topic **utilities**

sort, 24

amplicon.flag (*flags*), 5  
arrayCGH, 1, 2, 4, 12–14, 19, 25, 26  
arrayPlot, 23  
arrayTrend, 1, 3

chromosome.flag (*flags*), 5  
clone.qscore (*qscores*), 20  
control.flag (*flags*), 5

dapi.snr.flag (*flags*), 5

detectSB, 2

dyn.x.qscore (*qscores*), 20  
dyn.y.qscore (*qscores*), 20  
dynamics.qscore (*qscores*), 20

edge (*spatial*), 25

flag, 5, 6, 8, 9, 16, 25  
flag (*flag.arrayCGH*), 4  
flag.arrayCGH, 4, 27  
flag.summary, 6, 7, 11, 12  
flags, 5, 25

genome.plot, 8, 23  
global.spatial.flag (*flags*), 5  
gradient (*spatial*), 25

html.report, 8, 10, 22, 23

import, 12, 25  
intensity.flag (*flags*), 5

local.spatial.flag (*flags*), 5  
loess, 1, 2  
loess.control, 2

MANOR (*norm*), 15  
manor (*norm*), 15

nem, 2, 3, 14  
norm, 15  
norm.arrayCGH, 4, 6, 24, 25, 27

par, 9  
pct.clone.qscore (*qscores*), 20  
pct.replicate.qscore (*qscores*), 20  
pct.spot.before.qscore (*qscores*),  
20

pct.spot.qscore (*qscores*), 20  
position.flag (*flags*), 5

qscore, 18, 19, 20–22  
qscore.arrayCGH, 18, 19  
qscore.summary, 11, 18, 21, 22  
qscore.summary.arrayCGH, 19, 20

qscores, 20

ref.snr.flag(*flags*), 5

rep.flag(*flags*), 5

replicate.flag(*flags*), 5

report.plot, 9, 11, 12, 22

smoothness.qscore(*qscores*), 20

SNR.flag(*flags*), 5

sort, 24

spatial, 6, 20, 25

spatial.flag(*flags*), 5

spot.flag(*flags*), 5

title, 9

to.flag, 4, 26

to.qscore(*qscore*), 19

unique.flag(*flags*), 5

val.mark.flag(*flags*), 5

var.replicate.qscore(*qscores*), 20