flowQ

November 11, 2009

R topics documented:

aggregatorList-class *Class "aggregatorList"*

Description

A list of qaAggregators

Details

This class directly extends class "list" and is intended to exclusively hold objects of class qaAggregator, where each list item represents the outcome of a QA subprocess for a single flowFrame. It mainly exists to allow for method dispatch and should never be populated manualy; instead, use the constructor qaAggregatorList which checks for valid objects.

Objects from the Class

Objects should be created using the constructor:

aggregatorList(...), where \code{...} are objects inheriting form class qaAggregator or a list of such objects.

Slots

.Data: Object of class "list", the list data

Extends

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

Methods

initialize signature(.Object = "aggregatorList"): constructor

show signature(object = "aggregatorList"): print object details

Author(s)

Florian Hahne

See Also

qaGraph, writeQAReport, qaProcess, qaAggregator

Examples

```
showClass("aggregatorList")
```

`binaryAggregator-class` *Class "binaryAggregator"*

Description

Abstraction of a binary type of aggregator with possible states "passed" and "not passed"

Objects from the Class

Objects can be created by calls of the form `new("binaryAggregator", ...)`, or using the constructor `binaryAggregator(passed)`, where `passed` is a logical scalar.

Slots

`passed`: Object of class `"logical"` indicating whether the process has passed the QA requirements

Extends

Class `"qaAggregator"`, directly.

Methods

show `signature(object = "binaryAggregator")`: print object details

writeLines `signature(text = "binaryAggregator", con = "file", sep = "missing")`: write to HTML file connection

Author(s)

Florian Hahne

See Also

`qaProcess.marginevents`, `qaReport`, `qaProcess`, `qaProcess.timeline`, `discreteAggregator`, `factorAggregator`, `numericAggregator`, `stringAggregator`, `rangeAggregator`,

Examples

```
showClass("binaryAggregator")
```

`discreteAggregator-class` *Class "discreteAggregator"*

Description

Abstraction of a discrete type of aggregator with possible states "passed", "not passed" and "warning"

Objects from the Class

Objects can be created by calls of the form `new("discreteAggregator", ...)` or using the constructor `discreteAggregator(x)`, where `x` is an integer value in `[0,1,2]` or a factor with levels `0`, `1` and `2`.

Slots

x: Object of class `"factor"` One in `0` (not passes), `1` (passed) or `2` (warning)

passed: Object of class `"logical"` indicating whether the process has passed the QA requirements (not eveluated)

Extends

Class `"qaAggregator"`, directly.

Methods

show `signature(object = "discreteAggregator")`: print object details

writeLines `signature(text = "discreteAggregator", con = "file", sep = "missing")`: write to HTML file connection

Author(s)

Florian Hahne

See Also

`qaProcess.marginevents`, `qaReport`, `qaProcess`, `qaProcess.timeline`, `binaryAggregator`, `factorAggregator`, `numericAggregator`, `stringAggregator`, `rangeAggregator`,

Examples

```
showClass("discreteAggregator")
```

`evaluateProcess` *Evaluate QA processes*

Description

Re-evaluate an object of class `qaProcess`, e.g. for the case that a threshold value has changed.

Usage

```
evaluateProcess(process, thresh, ...)
```

Arguments

`process`	An object of class `qaProcess`.
`thresh`	The new treshold on which the process is to be evaluated.
`...`	Further arguments that are passed on to the individual functions for each QA process type.

Details

It is sometimes useful to update the state of aggregators in a `qaProcess` , for instances after changing the threshold value, without having to recompute all images, which can be very time consuming.

Value

An updated object of class `qaProcess`

Note

This function needs to be extended for new types of `qaProcess`.

Author(s)

Florian Hahne

See Also

`qaProcess`, `writeQAReport`

Examples

```
## Not run:
data(GvHD)
dest <- tempdir()
qp1 <- qaProcess.timeline(GvHD[1:3], channel="FL1-H", outdir=dest,
cutoff=1)
evaluateProcess(qp1, thresh=4)
## End(Not run)
```

`factorAggregator-class` *Class "factorAggregator"*

Description

Abstraction of a factor type of aggregator with possible states coded by the factor levels

Objects from the Class

Objects can be created by calls of the form `new("factorAggregator", ...)` or using the constructor `factorAggregator(x, passed)`, where `x` is a factor, or an object which can be coerced to a factor, and `passed` is a logical scalar.

Slots

`x`: Object of class `"factor"` coding the outcome state

`passed`: Object of class `"logical"` indicating whether the process has passed the QA requirements

Extends

Class `"qaAggregator"`, directly.

Methods

show `signature(object = "factorAggregator")`: print object details

writeLines `signature(text = "factorAggregator", con = "file", sep = "missing")`: write to HTML file connection

Author(s)

Florian Hahne

See Also

`qaProcess.marginevents`, `qaReport`, `qaProcess`, `qaProcess.timeline`, `discreteAggregator`, `binaryAggregator`, `numericAggregator`, `stringAggregator`, `rangeAggregator`,

Examples

```
showClass("factorAggregator")
```

`flowQ-package` *Quality control for flow cytometry*

Description

Functions and methods for quality control and QA of flow cytometry data. This package heavily depends on the flowCore package.

Details

Package:	flowQ
Type:	Package
Version:	0.2.1
Date:	2006-11-16
License:	Artistic

Quality control is an important aspect when dealing with large amounts of complex high-throughput data. This package comprises functionality for efficient QA of flow cytometry data.

Author(s)

Maintainer: Florian Hahne <f.hahne@dkfz.de> Authors: R. Gentleman, F. Hahne, J. Kettman, N. Le Meur, M. Tang

References

references go here

See Also

flowCore

Examples

```
## examples go here
```

---

locateDuplicatedParameters

Locate Duplicated Parameters

Description

Identifies dyes that are repeated arcross multiple flowSets using information provided in the description field of each flowFrame.

Usage

```
locateDuplicatedParameters(flowList)
```

Arguments

flowList	A list of flowSet

Details

This function is used internally by the qaProcess functions to identify dyes that are repeated across multiple aliquots. The function takes in a list of flowSet and identifies stains that are repeated across multiple flowSets (aliquots). These parameters are expected to have the same distribution across aliquots and can be used for Quality control. The method looks for the information provided in the description field of each flowFrame.

Value

An object of class qaProcess.

Author(s)

Nishant Gopalakrishnan

See Also

normalizeSets

Examples

```
## Not run:

## End(Not run)
```

---

normailzeSets *Locate Duplicated Parameters*

Description

Aligns up peaks for parameters across multiple flowSets using the warpSet method from the flowStats package.

Usage

```
normalizeSets(flowList,dupes,peaks=NULL)
```

Arguments

flowList	A list of flowSet
dupes	Stains that are to be aligned across multiple flowSet
peaks	Number of peaks expected in the distribution. If NULL is passed as input, The number of peaks to align are estimated from the distribution

Details

This function can be used to normalize dye information across multiple flowSets so that the peaks in the distributions align. The method makes use of information provided in the description field of each flowFrame to align up floursecense information from multiple flowSet.

Value

A list of objects of class flowSet.

Author(s)

Nishant Gopalakrishnan

See Also

locateDuplicatedParameters

Examples

```
## Not run:

## End(Not run)
```

---

numericAggregator-class

Class "numericAggregator"

Description

Abstraction of a numeric type of aggregator for which possible states are coded by a numeric value

Objects from the Class

Objects can be created by calls of the form `new("numericAggregator", ...)` or using the constructor `numericAggregator(x, passed)`, where `x` is a numeric scalar, and `passed` is a logical scalar.

Slots

x: Object of class `"numeric"` coding the outcome state

passed: Object of class `"logical"` indicating whether the process has passed the QA requirements

Extends

Class `"qaAggregator"`, directly.

Methods

show `signature(object = "numericAggregator")`: print object details

writeLines `signature(text = "numericAggregator", con = "file", sep = "missing")`: write to HTML file connection

Author(s)

Florian Hahne

See Also

qaProcess.marginevents, qaReport, qaProcess, qaProcess.timeline, discreteAggregator, factorAggregator, binaryAggregator, stringAggregator, rangeAggregator,

Examples

```
showClass("numericAggregator")
```

outlier-class *Virtual Class "outlier"*

Description

A class to represent outlier tests

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

test: Object of class "character" ~~

parameters: Object of class "ANY" ~~

Methods

No methods defined with class "outlier" in the signature.

Note

~~further notes~~

Author(s)

~~who you are~~

References

~put references to the literature/web site here ~

Examples

```
## showClass("outlier")
```

outlierResult-class *Class "outlierResult"*

Description

A class to hold the results of an outlier test

Objects from the Class

Objects can be created by calls of the form `new("outlierResult", ...)`. ~~ describe objects here ~~

Slots

frameId: Object of class "character" ~~

filterDetails: Object of class "list" ~~

test: Object of class "character" ~~

parameters: Object of class "ANY" ~~

Extends

Class "outlier", directly.

Methods

No methods defined with class "outlierResult" in the signature.

Note

~~further notes~~

Author(s)

~~who you are~~

References

~put references to the literature/web site here ~

Examples

```
#showClass("outlierResult")
```

---

outliers-methods *~~ Methods for Function outliers in Package '.GlobalEnv' ~~*

Description

~~ Methods for function outliers in Package '.GlobalEnv' ~~

Methods

x = "flowSet" ~~describe this method here

`qaAggregator-class` *Abstraction of the possible outcomes of a QA process*

Description

Virtual parent class for different types of QA aggregators

Details

In the context of this package, `qaAggregators` are objects that hold the outcome of a QA process. Each subclass implements its own `writeLine` method, which creates the appropriate HTML code for a graphical representation of the object.

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

`passed`: Object of class `"logical"` indicating whether the process has passed the QA requirements

Methods

No methods defined with class "qaAggregator" in the signature.

Author(s)

Florian Hahne

See Also

`qaProcess.marginevents`, `qaReport`, `qaProcess`, `qaProcess.timeline`, `binaryAggregator`, `discreteAggregator`, `factorAggregator`, `numericAggregator`, `stringAggregator`, `rangeAggregator`,

Examples

```
showClass("qaAggregator")
```

qaGraph-class *Class "qaGraph"*

Description

Abstraction of the graphical output created in the cause of a QA process

Objects from the Class

Objects should be created using the constructor:

`qaGraph(fileName, imageDir, width, empty=FALSE, pdf=TRUE)`, where `fileName` is a path to an image file, `imageDir` is the destination path for the images and the optional argument `width` is the final width to which the bitmap images are converted. The `pdf` argument controls wether vectorized versions of the image are to be produced. For the special case of an empty object without any images one can use option `empty=TRUE`, in which case the constructor ignores all other arguments.

During object instantiation the image file will be converted, resized and copied if necessary.

Slots

`fileNames`: Object of class `"character"` The paths to the image files, both the vectorized and unverctorized versions

`dimensions`: Object of class `"matrix"` The dimensions of the image files, both for the vectorized and unvectorized version

`types`: Object of class `"character"` The file extensions for both versions

`id`: Object of class `"character"` A unique identifier for the images

Methods

initialize `signature(.Object = "qaGraph")`: constructor

names `signature(x = "qaGraph")`: returns the file name of the bitmap version of the image

show `signature(object = "qaGraph")`: print object details

Author(s)

Florian Hahne

See Also

`qaGraphList`, `writeQAReport`, `qaProcess`

Examples

```
showClass("qaGraph")
```

qaGraphList-class *Class "qaGraphList"*

Description

A list of qaGraph objects

Details

This class directly extends class "list" and is intended to exclusively hold objects of class qaGraph, where each list item represents the grapical output of a QA subprocess for a single flowFrame. It mainly exists to allow for method dispatch and should never be populated manualy; instead, use the constructor qaGraphList which makes sure, that all image files are converted into the appropriate types and sizes and copied to the expected file location.

Objects from the Class

Objects should be created using the constructor:

qaGraphList(imageFiles, imageDir, width, pdf=TRUE), where imageFiles are paths to image files, imageDir is the destination path for the images and width is the final width to which the bitmap images are converted. pdf can be used to toggle the generation of vectorized versions of the images.

Slots

.Data: Object of class "list", the list data

Extends

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

Methods

initialize signature(.Object = "qaGraphList"): constructor

show signature(object = "qaGraphList"): print object details

Author(s)

Florian Hahne

See Also

qaGraph, writeQAReport, qaProcess

Examples

```
showClass("qaGraphList")
```

`qaProcess.2DStatsPlot` *Creates a QA process for 2D summary statistic comparisons across multiple samples*

Description

This function takes a list of `flowSet` as input and creates all necessary output for a '2DStatsPlot' type QA process. Objects created by this function can be laid out as HTML using `writeQAReport`.

Usage

```
qaProcess.2DStatsPlot(flowList,dyes=c("FSC-A","SSC-A"),
                      outdir="QAReport",outBound=0.25,func=mean,
                      det.dimensions=c(400,400),pdf=FALSE,...)
```

Arguments

`flowList`	A list of `flowSet`
`dyes`	Flow parameters to be compared across multiple samples
`outdir`	The directory to which the graphical output is to be saved. If multiple QA processes are to be combined, make sure to use the same directory every time.
`outBound`	A numeric value between 0 and 1 which is used as a threshold by the outlier detection algorithm
`func`	Summary statistic function to be applied to each parameter in a flow Frame (mean,median etc)
`det.dimensions`	the dimensions of each image generated by the QA process.
`pdf`	boolen value determinining if pdf files or jpeg images will be produced by the QA process
`...`	Further arguments.

Details

QA processes of type '2DStatsPlot' detects differences in the value of a summary statistic such as the mean, median etc across multiple samples.

For more details on how to layout `qaProcess` objects to HTML, see `writeQAReport` and `qaReport`.

Value

An object of class `qaProcess`.

Author(s)

Nishant Gopalakrishnan

See Also

writeQAReport, qaReport, qaProcess, qaProcess.marginevents, qaProcess.DensityPlot, qaProcess.ECDFPlot

Examples

```
data(qData)
dest <- tempdir()
par<-c("FSC-A","SSC-A","CD4","CD8")
resMean <- qaProcess.2DStatsPlot(qData,dyes=par,outdir=dest,func=mean,outBound=0.28,pdf=T
resMean
```

---

qaProcess.BoundaryPlot

Creates a QA process for comparison of the percentage of boundary events for the specified parameters across multiple samples

---

Description

This function takes a list of flowSet as input and creates all necessary output for a 'BoundaryPlot' type QA process. Objects created by this function can be laid out as HTML using writeQAReport.

Usage

```
qaProcess.BoundaryPlot(flowList,dyes=NULL,
                       outdir="QAReport",cutoff=3,
                       det.dimensions=c(400,400),pdf=FALSE,...)
```

Arguments

flowList	A list of flowSet
dyes	Flow parameters to be compared across multiple samples. If set to NULL, all parameters that are duplicated across the list of flowSet are identified and compared.
outdir	The directory to which the graphical output is to be saved. If multiple QA processes are to be combined, make sure to use the same directory every time.
cutoff	A numeric value for the percentage of boundary events that is used by the process to identify samples that failed the QA process
det.dimensions	the dimensions of each image generated by the QA process.
pdf	boolen value determinining if pdf files or jpeg images will be produced by the QA process
...	Further arguments.

Details

QA processes of type 'BoundaryPlot' helps identify samples that have a large number of boundary events that must be removed by further gating before detailed analysis of the data.

For more details on how to layout `qaProcess` objects to HTML, see `writeQAReport` and `qaReport`.

Value

An object of class `qaProcess`.

Author(s)

Nishant Gopalakrishnan

See Also

`writeQAReport`, `qaReport`, `qaProcess`, `qaProcess.DensityPlot`, `qaProcess.ECDFPlot`

Examples

```
data(qData)
dest <- tempdir()
resBoundary <- qaProcess.BoundaryPlot(qData, dyes=c("FSC-A","CD3"),
                                      outdir=dest, cutoff=3, pdf=TRUE)
resBoundary
```

---

qaProcess.cellnumber

Create QA process of type 'cellnumber'

Description

This function takes a `flowSet` as input and creates all necessary output for a 'cellnumber' type QA process. Objects created by this function can be laid out as HTML using `writeQAReport`.

Usage

```
qaProcess.cellnumber(set, grouping=NULL, outdir, cFactor=2,
                     absolute.value=NULL, two.sided = FALSE, name="cell number",
                     sum.dimensions=c(7,7), pdf=TRUE, ...)
```

Arguments

`set`	A `flowSet`.
`grouping`	A character vector defining one of the variables in the phenoData of `set` used as a grouping variable. If this argument is used and if `absolute.value` is `NULL`, outlier detection will be performed within groups rather than across all samples.

outdir	The directory to which the graphical output is to be saved. If multiple QA processes are to be combined, make sure to use the same directory for all of them.
cFactor	The outlier threshold at which the QA criterion is considered to have failed. This is essentially the factor of standard deviations away from the average number of cells per sample, either in both directions if two.sided=TRUE or only towards smaller event numbers if two.sided=FALSE.
absolute.value	An absolute event count below which the QA criterion is considered to be failed. If this argument is not NULL, cFactor and two.sided are ignored.
two.sided	Perform a two-sided outlier detection, i.e., report both unproportionally high and low event numbers.
name	The name of the process used for the headings in the HTML output.
sum.dimensions	The dimensions of the pdf deviced in inches used for the summary plot.
pdf	Logical indicating whether to create vectorized versions of images for this quality process. This should be set to FALSE if disk space is critical, since the pdf versions of the image consume much more space on the hard drive compared to the bitmap version.
...	Further arguments.

Details

QA processes of type 'cellnumber' detect aberations in the number of events per sample. These are either determined dynamically as outliers from the typical distribution of event counts for the whole set, or, if absolute.value is not NULL, by an absolute cutoff value. If there is a natural grouping among the samples, this can be specified using the grouping argument. In this case, the outlier detection will be performed within its respective group for a particular sample.

For more details on how to layout qaProcess objects to HTML, see writeQAReport and qaReport.

Value

An object of class qaProcess.

Author(s)

Florian Hahne

See Also

writeQAReport, qaReport, qaProcess, qaProcess.marginevents, qaProcess.timeflow, qaProcess.timeline

Examples

```
## Not run:
data(GvHD)
dest <- tempdir()
qp <- qaProcess.cellnumber(GvHD, outdir=dest, cFactor=2)
qp
## End(Not run)
```

qaProcess-class *Abstraction of the results of a QA process*

Description

QA processes create graphical output which can be bundled in a single HTML document. This class stores all information that is needed by writeQAReport to produce such HTML reports.

Objects from the Class

Objects should be created using the constructor functions. See qaProcess.timeline and qaProcess.marginevents for details. When writing new QA process functions, the constructors qaProcessFrame and qaProcess should be used. The latter expects the mandatory arguments id, type and frameProcesses and also accepts the optional arguments name and summaryGraph. See the vignette of this package for details.

Slots

id: Object of class "character", the objects unique identifier

name: Object of class "character", the name of the process

type: Object of class "character", the type of process

frameIDs: Object of class "character", the identifiers of the flowSets to which the subprocesses are linkes

summaryGraph: Object of class "qaGraph", a graphical summary of the processe's outcome

frameProcesses: Object of class "list", more detailed information for each flowFrame

Methods

initialize signature(.Object = "qaProcess"): constructor

Author(s)

Florian Hahne

See Also

qaGraphList, writeQAReport, qaProcessFrame

Examples

```
showClass("qaProcess")
```

qaProcess.DensityPlot *Creates a QA process for comparisons of density plots across multiple samples*

Description

This function takes a list of flowSet as input and creates all necessary outputs for a 'DensityPlot' type QA process. Objects created by this function can be laid out as HTML using writeQAReport.

Usage

```
qaProcess.DensityPlot(flowList,dyes=NULL,outdir="QAReport",
                      alpha=0.05,det.dimensions=c(400,400),
                    pdf=FALSE,...)
```

Arguments

flowList	A list of flowSet
dyes	Flow parameters to be compared across multiple samples. If set to NULL, all parameters that are duplicated across the list of flowSet are identified and compared.
outdir	The directory to which the graphical output is to be saved. If multiple QA processes are to be combined, make sure to use the same directory every time.
alpha	Outlier mislabeling rate
det.dimensions	the dimensions of each image generated by the QA process.
pdf	boolen value determinining if pdf files or jpeg images will be produced by the QA process
...	Further arguments.

Details

QA processes of type 'DensityPlot' detects differences in the density plots across multiple samples. A summary measure based on the sum of the pairwise KL Distances of density plots is used as a parameter to flag patient panels that are potential outliers.

For more details on how to layout qaProcess objects to HTML, see writeQAReport and qaReport.

Value

An object of class qaProcess.

Author(s)

Nishant Gopalakrishnan

See Also

writeQAReport, qaReport, qaProcess, qaProcess.marginevents, qaProcess.ECDFPlot, qaProcess.2DStatsPlot

Examples

```
data(qData)
dest <- tempdir()
resDensityFSC <- qaProcess.DensityPlot(qData,dyes=c("FSC-A","SSC-A"),outdir=dest,alpha=0.
```

`qaProcess.ECDFPlot` *Creates a QA process for comparisons of ECDF's across multiple samples*

Description

This function takes a list of `flowSet` as input and creates all necessary outputs for an 'ecdfOutliers' type of QA process. Objects created by this function can be laid out as HTML using `writeQAReport`.

Usage

```
qaProcess.ECDFPlot(flowList,dyes=NULL,outdir="QAReport",
                        alpha=0.05,det.dimensions=c(400,400),pdf=FALSE,...)
```

Arguments

`flowList`	A list of `flowSet`
`dyes`	Flow parameters to be compared across multiple samples. If set to NULL, all parameters that are duplicated across the list of `flowSet` are identified and compared.
`outdir`	The directory to which the graphical output is to be saved. If multiple QA processes are to be combined, make sure to use the same directory every time.
`alpha`	Outlier mislabeling rate
`det.dimensions`	the dimensions of each image generated by the QA process.
`pdf`	boolen value determinining if pdf files or jpeg images will be produced by the QA process
`...`	Further arguments.

Details

QA processes of type 'ECDFPLot' detects differences in the empirical cumulative distribution of flow parameters across multiple samples.A summary measure based on the sum of the pairwise KL Distances of ECDF plots across multiple samples is used as a parameter to flag patient panels that are potential outliers.

For more details on how to layout `qaProcess` objects to HTML, see `writeQAReport` and `qaReport`.

Value

An object of class `qaProcess`.

Author(s)

Nishant Gopalakrishnan

See Also

writeQAReport, qaReport, qaProcess, qaProcess.marginevents, qaProcess.DensityPlot, qaProcess.2DStatsPlot

Examples

```
data(qData)
dyes<- c("FSC-A","SSC-A")
dest <- tempdir()
resFSCECDF <- qaProcess.ECDFPlot(qData,dyes=dyes,outdir=dest,alpha=0.4,pdf=TRUE)
resFSCECDF
```

---

qaProcessFrame-class

Class "qaProcessFrame"

Description

Abstraction of subitems within a qQA process

Details

This class bundles graphs and aggregators for a single flowFrame. This allows to create processes with subcomponents, where each item in the frameAggregators and frameGraphs lists corresponds to one subprocess, which can be used, for instance, to create individual plots for each flow channel. For QA processes without subcomponents, these slots would simply not be populated.

Objects from the Class

Objects should be created using the constructor:

qaProcessFrame(frameID, summaryAggregator, summaryGraph, frameAggregators, frameGraphs, details) where frameID is the ID of the flowFrame the process is linked to, summaryAggregator is an object inheriting from class qaAggregator which summarizes the outcome, summaryGraph is an object of class qaGraph which is the overview graph of the process for the whole frame, details is a list containing any additional information regarding the QA process, frameAggregators is an object of class aggregatorList and frameGraphs is an object of class qaGraphList. The latter two are the collections of aggregators and graphs for each subprocess. Only frameID and summaryAggregator are mandatory arguments.

Slots

id: Object of class "character", a unique ID of the object

frameID: Object of class "character", ID of the flowFrame the process is linked to

summaryAggregator: Object of class "qaAggregator", an aggregator summarizing the output of the process

summaryGraph: Object of class "qaGraph", a graphical summary of the process

frameAggregators: Object of class "aggregatorList" a list of aggregators for the subprocesses

frameGraphs: Object of class "qaGraphList" a list of graphical summaries for the subprocesses

details: A list for any additional information

Methods

initialize signature(.Object = "qaProcessFrame"): constructor

Author(s)

Florian Hahne

See Also

qaGraphList, writeQAReport, qaProcess

Examples

```
showClass("qaProcessFrame")
```

---

qaProcess.KLDistPlot

Creates a QA process for comparisons of KL Distances across multiple samples

---

Description

This function takes a list of flowSet as input and creates all necessary outputs for an 'KLDistPlot' type of QA process. Objects created by this function can be laid out as HTML using writeQAReport.

Usage

```
qaProcess.KLDistPlot(flowList,dyes=NULL,outdir="QAReport",
                     alpha=0.05,det.dimensions=c(400,400),pdf=FALSE,...)
```

Arguments

flowList	A list of flowSet
dyes	Flow parameters to be compared across multiple samples. If set to NULL, all parameters that are duplicated across the list of flowSet are identified and compared.
outdir	The directory to which the graphical output is to be saved. If multiple QA processes are to be combined, make sure to use the same directory every time.
alpha	Outlier mislabeling rate

det.dimensions
: the dimensions of each image generated by the QA process.

pdf
: boolen value determinining if pdf files or jpeg images will be produced by the QA process

...
: Further arguments.

Details

QA processes of type 'KLDistPlot' detects differences in the pairwise KL Distances of the flow parameters across multiple samples.

For more details on how to layout qaProcess objects to HTML, see writeQAReport and qaReport.

Value

An object of class qaProcess.

Author(s)

Nishant Gopalakrishnan

See Also

writeQAReport, qaReport, qaProcess, qaProcess.BoundaryPlot, qaProcess.DensityPlot, qaProcess.2DStatsPlot

Examples

```
data(qData)
dest <- tempdir()
resKLdist <- qaProcess.KLDistPlot(qData,dyes=c("SSC-A","FSC-A"),outdir=dest,alpha=0.05,pc
resKLdist
```

---

qaProcess.marginevents

Create QA process of type 'marginevents'

Description

This function takes a flowSet as input and creates all necessary output for a 'marginevents' type QA process. Objects created by this function can be laid out as HTML using writeQAReport.

Usage

```
qaProcess.marginevents(set, channels=NULL, grouping=NULL, outdir,
cFactor=2, absolute.value=NULL, name="margin events",
sum.dimensions=NULL, det.dimensions=c(7,3), pdf=TRUE, ...)
```

Arguments

set	A flowSet.
channels	A character vector of channel names for which margin events are to be recorded. Will use all available channels except for the time channel if NULL.
grouping	A character vector defining one of the variables in the phenoData of set used as a grouping variable. If this argument is used and if absolute.value is NULL, outlier detection will be performed within groups rather than across all samples.
outdir	The directory to which the graphical output is to be saved. If multiple QA processes are to be combined, make sure to use the same directory for all of them.
cFactor	The outlier threshold at which the QA criterion is considered to have failed. This is essentially the factor of standard deviations away from the average number of cells per sample for the respective channel.
absolute.value	An absolute percentage of margin events below which the QA criterion is considered to be failed. If this argument is not NULL, cFactor is ignored.
name	The name of the process used for the headings in the HTML output.
sum.dimensions, det.dimensions	The dimensions of the pdf deviced in inches used for the summary plot and the sample-specific plots, respectively.
pdf	Logical indicating whether to create vectorized versions of images for this quality process. This should be set to FALSE if disk space is critical, since the pdf versions of the image consume much more space on the hard drive compared to the bitmap version. Setting the argument to FALSE will also speed up the process.
...	Further arguments.

Details

QA processes of type 'marginevents' record the number of events that fall on the margins of the measurement range for each channel. Unproportionally high numbers of such events can indicate problems with the instrument settings or with compensation. These are either determined dynamically as outliers from the typical distribution of margin event percentages for the whole set, or, if absolute.value is not NULL, by an absolute cutoff value. If there is a natural grouping among the samples, this can be specified using the grouping argument. In this case, the outlier detection will be performed within its respective group for a particular sample.

For more details on how to layout qaProcess objects to HTML, see writeQAReport and qaReport.

Value

An object of class qaProcess.

Author(s)

Florian Hahne

See Also

writeQAReport, qaReport, qaProcess, qaProcess.timeline, qaProcess.timeflow, qaProcess.cellnumber

Examples

```
## Not run:
data(GvHD)
dest <- tempdir()
qp <- qaProcess.marginevents(GvHD, channels=c("FSC-H", "SSC-H"),
  outdir=dest)
qp
## End(Not run)
```

---

qaProcessSummary *Class "qaProcessSummary"*

Description

An internal class to represent QA summaries

Objects from the Class

The class is internal and not ment for interactive use.

Slots

panels: Object of class `"list"`

ranges: Object of class `"matrix"`

summary: Object of class `"list"`

mapping: Object of class `"list"`

Methods

writeLines `signature(text = "qaGraphList", con = "file")`: HTML output

show `signature(object = "qaGraphList")`: print object details

Author(s)

Florian Hahne

`qaProcess.timeflow` *Create QA process of type 'timeflow'*

Description

This function takes a `flowSet` as input and creates all necessary output for a 'timeflow' type QA process. Objects created by this function can be laid out as HTML using `writeQAReport`.

Usage

```
qaProcess.timeflow(set, outdir, cutoff=2, name="time flow",
sum.dimensions=c(7,7), det.dimensions=c(7,7), pdf=TRUE, ...)
```

Arguments

`set`	A `flowSet`.
`outdir`	The directory to which the graphical output is to be saved. If multiple QA processes are to be combined, make sure to use the same directory for all of them.
`cutoff`	The threshold at which the QA criterion is considered to have failed. An absolute value in the timeline deviation score as computed by the `timeLinePlot` function.
`name`	The name of the process used for the headings in the HTML output.
`sum.dimensions, det.dimensions`	The dimensions of the pdf device in inches used for the summary and the detailed plots.
`pdf`	Logical indicating whether to create vectorized versions of images for this quality process. This should be set to `FALSE` if disk space is critical, since the pdf versions of the image consume much more space on the hard drive compared to the bitmap version.
`...`	Further arguments.

Details

QA processes of type 'timeflow' detect disturbances in the flow of cells over time. These indicate problems with the cell suspension, clogging of the needle or similar issues. If the flow rate is to high, the frequency of measuring cell douplettes increases.

For more details on how to layout `qaProcess` objects to HTML, see `writeQAReport` and `qaReport`.

Value

An object of class `qaProcess`.

Author(s)

Florian Hahne

See Also

writeQAReport, qaReport, qaProcess, qaProcess.marginevents, qaProcess.timeline, qaProcess.cellnumber

Examples

```
## Not run:
data(GvHD)
dest <- tempdir()
qp <- qaProcess.timeflow(GvHD, outdir=dest, cutoff=1)
qp
## End(Not run)
```

---

qaProcess.timeline *Create QA process of type 'timeline'*

Description

This function takes a flowSet as input and creates all necessary output for a 'timeline' type QA process. Objects created by this function can be laid out as HTML using writeQAReport.

Usage

```
qaProcess.timeline(set, channels=NULL, outdir, cutoff=1,
  name="time line", sum.dimensions=NULL, det.dimensions=c(7,7),
  pdf=TRUE, ...)
```

Arguments

set	A flowSet
channels	A character vector of channel names for which the qaReport is to be produced. Will use all available channels except for the time channel if NULL.
outdir	The directory to which the graphical output is to be saved. If multiple QA processes are to be combined, make sure to use the same directory for all of them.
cutoff	The threshold at which the QA criterion is considered to be failed. An absolute value in the timeline deviation score as computed by the timeLinePlot function.
name	The name of the process used for the headings in the HTML output.
sum.dimensions, det.dimensions	The dimensions of the pdf device in inches used for the summary and the detailed plots.
pdf	Logical indicating whether to create vectorized versions of images for this quality process. This should be set to FALSE if disk space is critical, since the pdf versions of the image consume much more space on the hard drive compared to the bitmap version.
...	Further arguments.

Details

QA processes of type 'timeline' detect unusal patterns in the acquisition of fluorescense and light scatter measurements over time. These are detected dynamically by identifying trends in the signal intensity over time or local changes in the measurement intensities. The underlying hypothesis is that measurement values are acquired randomly, hence there shouldn't be any correlation to time.

For more details on how to layout `qaProcess` objects to HTML, see `writeQAReport` and `qaReport`.

Value

An object of class `qaProcess`.

Author(s)

Florian Hahne

See Also

`writeQAReport`, `qaReport`, `qaProcess`, `qaProcess.marginevents`, `qaProcess.timeflow`, `qaProcess.cellnumber`

Examples

```
## Not run:
data(GvHD)
GvHD <- transform(GvHD, "FL1-H"=asinh(`FL1-H`), "FL2-H"=asinh(`FL2-H`))
dest <- tempdir()
qp <- qaProcess.timeline(GvHD, channel="FL1-H", outdir=dest, cutoff=1)
qp
## End(Not run)
```

---

qaReport *Create HTML report using one or several QA process function(s)*

Description

This function combines all graphical output of multiple QA process functions for one `flowSet` in a single hyperlinked HTML document.

Usage

```
qaReport(set, qaFunctions, outdir = "./qaReport", argLists, grouping =
NULL, ...)
```

Arguments

set	A flowSet
qaFunctions	A character vector of the names of QA process functions to be used
outdir	The directory to which the HTML report is to be saved.
argLists	lists of argument lists for each of the QA process functions specified via qaFunctions
grouping	A character scalar indicating a variable in the flowSet's phenoData that is used as a grouping factor in the output.
...	Further arguments that are passed on to writeQAReport.

Details

This is a simple convenience function to produce HTML QA reports for a single flowSet given a list of QA process functions. For more fine-grained control use function writeQAReport directly.

An entry point to the output of this function can be found at outdir/index.html.

Value

The function is called for it's side effects

Author(s)

Florian Hahne

See Also

qaProcess.marginevents, writeQAReport, qaProcess, qaProcess.timeline

Examples

```
## Not run:
data(GvHD)
GvHD <- transform(GvHD, "FL1-H"=asinh(`FL1-H`), "FL2-H"=asinh(`FL2-H`))
dest <- tempdir()
qaReport(GvHD, c("qaProcess.timeline", "qaProcess.marginevents"), dest,
 list(list(channel="FL1-H", cutoff=1), list(channels=c("FL1-H",
  "FL2-H"), cFactor=4)))
browseURL(file.path(dest, "index.html"))
## End(Not run)
```

---

qData *Data for demonstrating QA processes that compares dye parameters from each patient across multiple aliquots*

Description

A list of 8 flowSet. Each flowSet has data from four patients and each element in the list corresponds to an aliquot.

Usage

```
data(qData)
```

Format

The object contains a list of 8 objects of class `flowSet`, each composed of 4 `flowFrames`. Each `flowFrame` correponds to a sample from an aliquot.

Details

This qData dataset contains flouresence information for samples from four patients. The sample from each patient was separated into eight aliquots and each aliquot was stained using a different combination of dyes. For each patient, some dyes appear more than once in multiple alquots. These along with the Forward and Side scatter information can be used for Quality control.

---

rangeAggregator-class

Class "rangeAggregator"

Description

Abstraction of a range type of aggregator where possible states are within certain ranges (e.g. percentages)

Objects from the Class

Objects can be created by calls of the form `new("rangeAggregator", ...)` or using the constructor `rangeAggregator(x, min, max, passed)`, where `x`, `min` and `max` are numeric scalars, with `x` in the range of `[min, max]`, and `passed` is a logical scalar.

Slots

`min`: Object of class `"numeric"`, the range minimum

`max`: Object of class `"numeric"`, the range maximum

`x`: Object of class `"numeric"`, the value within the range

`passed`: Object of class `"logical"` indicating whether the process has passed the QA requirements

Extends

Class `"numericAggregator"`, directly. Class `"qaAggregator"`, by class "numericAggregator", distance 2.

Methods

show `signature(object = "rangeAggregator")`: print object details

writeLines `signature(text = "rangeAggregator", con = "file", sep = "missing")`: write to HTML file connection

Author(s)

Florian Hahne

See Also

qaProcess.marginevents, qaReport, qaProcess, qaProcess.timeline, discreteAggregator, factorAggregator, numericAggregator, stringAggregator, binaryAggregator,

Examples

```
showClass("rangeAggregator")
```

---

stringAggregator-class
Class "stringAggregator"

Description

Abstraction of a string type of aggregator for which possible states are indicated through a textual description

Objects from the Class

Objects can be created by calls of the form `new("stringAggregator", ...)` or using the constructor `stringAggregator(x, passed)`, where `x` is a character scalar, and `passed` is a logical scalar.

Slots

x: Object of class `"character"` which is a textual description of the outcome

passed: Object of class `"logical"` indicating whether the process has passed the QA requirements

Extends

Class `"qaAggregator"`, directly.

Methods

show `signature(object = "stringAggregator")`: print object details

writeLines `signature(text = "stringAggregator", con = "file", sep = "missing")`: write to HTML file connection

Author(s)

Florian Hahne

See Also

qaProcess.marginevents, qaReport, qaProcess, qaProcess.timeline, discreteAggregator, factorAggregator, numericAggregator, binaryAggregator, rangeAggregator,

Examples

```
showClass("stringAggregator")
```

`validProcess` *Validate a QAProcess object*

Description

Check for integrety and existence of files specified in a `qaProcess` object.

Usage

```
validProcess(object)
```

Arguments

`object`	A `qaProcess` object

Value

The function is called for it's side effects

Author(s)

Florian Hahne

See Also

`qaProcess.marginevents`, `qaReport`, `qaProcess`, `qaProcess.timeline`, `writeQAReport`

`writeQAReport` *Create HTML report for (lists of) qaProcess objects*

Description

This function combines all graphical output of multiple QA processes for one or several `flowSet`s in a single hyperlinked HTML document.

Usage

```
writeQAReport(set, processes, outdir = "./qaReport", grouping = NULL,
pagebreaks=TRUE, pdf=TRUE)
```

Arguments

set	A flowSet or a list of several flowSets.
processes	A list of A qaProcess objects or, in the case of multiple flowSets, a list of lists of qaProcess objects. See below for further details.
outdir	The directory to which the HTML report is to be saved. Each qaProcess object should have been created in the same directory.
grouping	A character scalar indicating a variable in the flowSet's phenoData that is used as a grouping factor in the output.
pagebreaks	A logical indicating whether the output should be on one long page, or split over several pages.
pdf	A logical indicating whether vectorized versions of the images should be included. Setting this to TRUE implies that the individual QA processes have also been created with setting pdf=TRUE.

Details

Both the information about graphical output generated as part of a QA process as well as the qunatitative or qualitative results are stored in objects of class qaProcess. The creation of such objects is abstracted in dedicated constructor function and the user should call these functions directly rather than creating qaProcess manually. writeQAReport takes lists of such objects and combines their information in a unified HTML document. A grouping factor can be specified to indicate subgroups of the data. In the case of multiple panels, a list of flowSets can be given to writeQAReport, and the function expects a list of lists of processes, where each process list is specific to one panel.

An entry point to the output of this function can be found at outdir/index.html.

Value

The function is mostly called for it's side effects, that is, the generation of a HTML quality report in outdir. A URL to an entry point for browsing of the report is returned.

Author(s)

Florian Hahne

See Also

qaProcess.marginevents, qaReport, qaProcess, qaProcess.timeline

Examples

```
## Not run:
data(GvHD)
GvHD <- transform(GvHD, "FL1-H"=asinh(`FL1-H`), "FL2-H"=asinh(`FL2-H`))
dest <- tempdir()
qp1 <- qaProcess.timeline(GvHD, channel="FL1-H", outdir=dest, cutoff=1)
qp2 <- qaProcess.marginevents(GvHD, channels=c("FL1-H", "FL2-H"),
  outdir=dest, cFactor=4)
url <- writeQAReport(GvHD, processes=list(qp1, qp2), outdir=dest,
  grouping="Patient")
browseURL(url)
## End(Not run)
```

Index