

Package ‘openCyto’

October 12, 2016

Type Package

Title Hierarchical Gating Pipeline for flow cytometry data

Version 1.10.3

Date 2012-06-11

Author Mike Jiang, John Ramey, Greg Finak, Raphael Gottardo

Maintainer Mike Jiang <wjiang2@fhcrc.org>

Description This package is designed to facilitate the automated gating methods in sequential way to mimic the manual gating strategy.

License Artistic-2.0

LazyLoad yes

Depends flowWorkspace(>= 3.17.43)

Imports methods,Biobase,gtools,flowCore(>= 1.31.17),flowViz,ncdfFlow(>= 2.11.34),flowWorkspace,flowStats(>= 3.29.1),flowClust,MASS,clue,plyr,RBGL,graph,data.table,ks,RColorBrewer,lattice,rrcov,R.utils

Suggests flowWorkspaceData, knitr, testthat, utils, tools, parallel

biocViews FlowCytometry, DataImport, Preprocessing, DataRepresentation

VignetteBuilder knitr

LinkingTo Rcpp

NeedsCompilation yes

R topics documented:

.isPolyfunctional	3
.prior_flowClust1d	3
.prior_kmeans	5
add,GatingHierarchy,multipleFilterResult-method	6
add_pop	7
as.data.table.gatingTemplate	8
boolMethod-class	9
dims,gtMethod-method	9

dummyMethod-class	9
fcEllipsoidGate	10
fcEllipsoidGate-class	10
fcFilter-class	10
fcFilterList	10
fcFilterList-class	11
fcPolygonGate	11
fcPolygonGate-class	11
fcRectangleGate	12
fcRectangleGate-class	12
fcTree	12
fcTree-class	13
filterObject,logicalFilterResult-method	13
flowClust.1d	13
flowClust.2d	15
gating	17
gating,boolMethod,GatingSet-method	18
gating,gtMethod,GatingSet-method	19
gating,polyFunctions,GatingSet-method	19
gating,refGate,GatingSet-method	20
gatingTemplate-class	20
getChildren,gatingTemplate,character-method	22
getGate,fcTree,character-method	22
getGate,gatingTemplate,character-method	23
getNodes,fcTree-method	23
getNodes,gatingTemplate-method	24
getParent,gatingTemplate,character-method	24
groupBy,gtMethod-method	25
gtMethod-class	25
gtPopulation-class	26
gtSubsets-class	26
isCollapse,gtMethod-method	27
listgtMethods	27
mindensity	28
mindensity2	29
names,gtMethod-method	30
names,gtPopulation-method	31
ocRectangleGate-class	31
ocRectRefGate	31
ocRectRefGate-class	32
openCyto	32
parameters,gtMethod-method	33
plot,fcFilterList,ANY-method	33
plot,fcTree,character-method	34
plot,gatingTemplate,missing-method	34
polyFunctions-class	35
posteriors,fcFilter,ANY-method	35
ppMethod,gatingTemplate,character-method	36

.isPolyfunctional 3

ppMethod-class	36
preprocessing,ppMethod,GatingSet-method	37
priors,fcFilter,ANY-method	37
prior_flowClust	38
quadGate.seq	39
quadGate.tmix	39
quantileGate	40
refGate-class	41
registerPlugins	41
show,boolMethod-method	42
show,fcFilter-method	43
show,gatingTemplate-method	43
show,gtMethod-method	44
tailgate	44
templateGen	45

Index 46

.isPolyfunctional *A function to tell wether a gating method is polyFunctions*

Description

A function to tell wether a gating method is polyFunctions

Usage

`.isPolyfunctional(gm)`

Arguments

`gm` an object that extends `gtMethod`

.prior_flowClust1d *Elicits data-driven priors from a flowSet object for a specified channel*

Description

We elicit data-driven prior parameters from a `flowSet` object for a specified channel. For each sample in the `flowSet` object, we apply a kernel-density estimator (KDE) and identify its local maxima (peaks). We then aggregate these peaks to elicit a prior parameters for each of `K` mixture components.

Usage

```
.prior_flowClust1d(flow_set, channel, K = NULL, hclust_height = NULL,
  clust_method = c("kmeans", "hclust"), hclust_method = "complete",
  artificial = NULL, nu0 = 4, w0 = 10, adjust = 2, min = -200,
  max = NULL, vague = TRUE)
```

Arguments

<code>flow_set</code>	a <code>flowSet</code> object
<code>channel</code>	the channel in the <code>flowSet</code> from which we elicit the prior parameters for the Student's <i>t</i> mixture
<code>K</code>	the number of mixture components to identify. By default, this value is <code>NULL</code> and determined automatically
<code>hclust_height</code>	the height of the <code>hclust</code> tree of peaks, where the should be cut By default, we use the median of the distances between adjacent peaks. If a value is specified, we pass it directly to <code>cutree</code> .
<code>clust_method</code>	the method used to cluster peaks together when for prior elicitation. By default, <code>kmeans</code> is used. However, if <code>K</code> is not specified, <code>hclust</code> will be used instead.
<code>hclust_method</code>	the agglomeration method used in the hierarchical clustering. This value is passed directly to <code>hclust</code> . Default is complete linkage.
<code>artificial</code>	a numeric vector containing prior means for artificial mixture components. The remaining prior parameters for the artificial components are copied directly from the most informative prior component elicited. If <code>NULL</code> (default), no artificial prior components are added.
<code>nu0</code>	prior degrees of freedom of the Student's <i>t</i> mixture components.
<code>w0</code>	the number of prior pseudocounts of the Student's <i>t</i> mixture components.
<code>adjust</code>	the bandwidth to use in the kernel density estimation. See <code>density</code> for more information.
<code>min</code>	a numeric value that sets the lower bound for data filtering. If <code>NULL</code> (default), no truncation is applied.
<code>max</code>	a numeric value that sets the upper bound for data filtering. If <code>NULL</code> (default), no truncation is applied.
<code>vague</code>	logical Whether to elicit a vague prior. If <code>TRUE</code> , we first calculate the median of standard deviations from all <code>flowFrames</code> . Then, we divide the overall standard deviation by the number of groups to the scale the standard deviation.

Details

Here, we outline the approach used for prior elicitation. First, we apply a KDE to each sample and extract all of its peaks (local maxima). It is important to note that different samples may have a different number of peaks. Our goal then is to align the peaks before aggregating the information across all samples. To do this, we utilize a technique similar to the peak probability contrasts (PPC) method from Tibshirani et al (2004). Effectively, we apply hierarchical clustering to the peaks from all samples to find clusters of peaks. We compute the sample mean and variance of the peaks within each cluster to elicit the prior means and its hyperprior variance, respectively, for a `flowClust`

mixture component. We elicit the prior variance for each mixture component by first assigning the observations within each sample to the nearest prior mean. Then, we compute the variance of the observations within each cluster. Finally, we average the variances corresponding to each mixture component across all samples in the `flowSet` object.

Following Tibshirani et al. (2004), we cluster the peaks from each sample using complete-linkage hierarchical clustering. The linkage type can be changed via the `hclust_method` argument. This argument is passed directly to `hclust`.

To cluster the peaks, we must cut the hierarchical tree by selecting either a value for `K` or by providing a height of the tree to cut. By default, we cut the tree using as the height the median of the distances between adjacent peaks within each sample. This value can be changed via the `hclust_height` argument and, if provided, will be passed to `cutree`. Also, by default, the number of mixture components `K` is `NULL` and is ignored. However, if `K` is provided, then it has priority over `hclust_height` and is passed instead directly to `cutree`.

To ensure that the KDEs are smooth, we recommend that the bandwidth set in the `adjust` argument be sufficiently large. We have defaulted this value to 2. If the bandwidth is not large enough, the KDE may contain numerous bumps, resulting in erroneous peaks.

Value

list of prior parameters

References

Tibshirani, R et al. (2004), "Sample classification from protein mass spectrometry, by 'peak probability contrasts'," *Bioinformatics*, 20, 17, 3034-3044. <http://bioinformatics.oxfordjournals.org/content/20/17/3034>.

<code>.prior_kmeans</code>	<i>Elicits data-driven priors from a flowSet object for specified channels using the K-Means clustering algorithm</i>
----------------------------	-----------------------------------------------------------------------------------------------------------------------

Description

We elicit data-driven prior parameters from a `flowSet` object for specified channels. For each sample in the `flowSet` object, we apply `kmeans` to obtain `K` clusters. From each cluster, we determine its centroid and the sample covariance matrix. We then aggregate these two sample moments across all samples for each cluster.

Usage

```
.prior_kmeans(flow_set, channels, K, nu0 = 4, w0 = 10, nstart = 10,  
pct = 0.1, min = NULL, max = NULL, ...)
```

Arguments

flow_set	a flowSet object
channels	a character vector containing the channels in the flowSet from which we elicit the prior parameters for the Student's t mixture
K	the number of mixture components to identify
nu0	prior degrees of freedom of the Student's t mixture components.
w0	the number of prior pseudocounts of the Student's t mixture components.
nstart	number of random starts used by kmeans algorithm
pct	percentage of randomly selected cells in each flowFrame that is used to elicit the prior parameters. The value should must be greater than 0 and less than or equal to 1.
min	a numeric vector that sets the lower bounds for data filtering. If NULL (default), no truncation is applied.
max	a numeric vector that sets the upper bounds for data filtering. If NULL (default), no truncation is applied.
...	Additional arguments passed to kmeans

Details

Because the cluster labels returned from kmeans are arbitrary, we align the clusters based on the centroids that are closest to a randomly selected reference sample. We apply the Hungarian algorithm implemented using the solve_LSAP function from the clue package to assist with the alignment.

If each frame within flow_set has a large number of cells, the computational costs of kmeans can be a burden. We provide the option to randomly select pct, a percentage of the cells from each flow frame to which kmeans is applied.

Value

list of flowClust prior parameters

add,GatingHierarchy,multipleFilterResult-method
bypass the default flowWorkspace:::addGate

Description

to support adding gate along with indices without loading flow data and computing
to support adding rectangleGate yet gating through boolean operations without loading flow data

Usage

```
## S4 method for signature 'GatingHierarchy,multipleFilterResult'
add(wf, action, name = NULL,
    ...)

## S4 method for signature 'GatingHierarchy,logicalFilterResult'
add(wf, action, parent, name,
    recompute, ...)

## S4 method for signature 'GatingHierarchy,ocRectangleGate'
add(wf, action, recompute, ...)

## S4 method for signature 'GatingHierarchy,ocRectRefGate'
add(wf, action, recompute, ...)
```

Arguments

wf	GatingHierarchy see add in flowWorkspace package
action	ocRectangleGate or logicalFilterResult
name	character vector that specifies the names of populations generated by multipleFilterResult
...	see add in flowWorkspace package
parent	character parent node path (full or partial)
recompute	logical see add in flowWorkspace package

Details

however it is proven that logical indices are too big to be efficiently passed around

add_pop	<i>apply a gating method to the GatingSet</i>
---------	-----------------------------------------------

Description

When interacting with the existing gated data, this function provides the alternative way to interact with the GatingSet by supplying the gating description directly through arguments without the need to write the complete csv gating template.

Usage

```
add_pop(gs, alias = "*", pop = "A+", parent, dims = NA, gating_method,
    gating_args = NA, collapseDataForGating = NA, groupBy = NA,
    preprocessing_method = NA, preprocessing_args = NA, ...)
```

Arguments

gs GatingSet or GatingSetList
 alias, pop, parent, dims, gating_method, gating_args, collapseDataForGating, groupBy, preprocessing_method,
 see details in [gatingTemplate](#)
 ... other arguments

- mc.cores passed to multicore package for parallel computing
- parallel_type character specifying the parallel type. The valid options are "none", "multicore", "cluster".
- cl cluster object passed to parallel package (when parallel_type is "cluster")

Examples

```
## Not run:
# add quad gates
add_pop(gs, gating_method = "mindensity", dims = "CCR7,CD45RA", parent = "cd4-cd8+", pop = "CCR7+/-CD45RA+/-")

# polyfunctional gates (boolean combinations of exsiting marginal gates)
add_pop(gs, gating_method = "polyFunctions", parent = "cd8", gating_args = "cd8/IFNg:cd8/IL2:cd8/TNFa")

#boolGate method
add_pop(gs, alias = "IL2orIFNg", gating_method = "boolGate", parent = "cd4", gating_args = "cd4/IL2|cd4/IFNg")

## End(Not run)
```

```
as.data.table.gatingTemplate
      convert a gatingTemplate object to a data.table
```

Description

It is the inverse function of gatingTemplate constructor.

Usage

```
## S3 method for class 'gatingTemplate'
as.data.table(x, keep.rownames = FALSE)
```

Arguments

x gatingTemplate object
 keep.rownames not used

Value

a data.table

boolMethod-class *A class to represent a boolean gating method.*

Description

It extends refGate class.

dims,gtMethod-method *get gating method dimensions*

Description

get gating method dimensions

Usage

```
## S4 method for signature 'gtMethod'
dims(object)
```

Arguments

object gtMethod

dummyMethod-class *A class to represent a dummy gating method that does nothing but serves as reference to be refered by other population*

Description

It is generated automatically by the csv template preprocessing to handle the gating function that returns multiple gates.

fcEllipsoidGate *constuctor for fcEllipsoidGate*

Description

constuctor for fcEllipsoidGate

Usage

fcEllipsoidGate(x, priors, posts)

Arguments

x	a ellipsoidGate object
priors	a list storing priors
posts	a list storing posteriors

fcEllipsoidGate-class *a concrete class that reprints the ellipsoidGate generated by flowClust*

Description

It stores priors and posteriors as well as the actual ellipsoidGate.

fcFilter-class *a virtual class that represents the gating result generated by flowClust gating function*

Description

Bascially it extends flowCore 'filter classes to have extra slot to store priors and posteriors

fcFilterList *constuctor for fcFilterList*

Description

constuctor for fcFilterList

Usage

fcFilterList(x)

Arguments

x	list of fcFilter (i.e. fcPolygonGate or fcRectangleGate)
---	----------------------------------------------------------

fcFilterList-class *a class that extends filterList class.*

Description

Each filter in the filterList must extends the fcFilter class

fcPolygonGate *constuctor for fcPolygonGate*

Description

constuctor for fcPolygonGate

Usage

fcPolygonGate(x, priors, posts)

Arguments

x	a polygonGate object
priors	a list storing priors
posts	a list storing posteriors

fcPolygonGate-class *a concrete class that reprints the polygonGate generated by flowClust*

Description

It stores priors and posteriors as well as the actual polygonGate.

fcRectangleGate *constructor for fcRectangleGate*

Description

constructor for fcRectangleGate

Usage

fcRectangleGate(x, priors, posts)

Arguments

x	a rectangleGate object
priors	a list storing priors
posts	a list storing posteriors

fcRectangleGate-class *a concrete class that represents the rectangleGate generated by flow-Clust*

Description

It stores priors and posteriors as well as the actual rectangleGate.

fcTree *constructor of fcTree*

Description

It adds an extra node data slot "fList"(which is a filterList object) to the gatingTemplate

Usage

fcTree(gt)

Arguments

gt	a gatingTemplate object
----	-------------------------

fcTree-class	<i>A class to represent a flowClust tree.</i>
--------------	-----------------------------------------------

Description

It is a graphNEL used as a container to store priors and posteriors for each flowClust gate that can be visualized for the purpose of fine-tuning parameters for flowClust algorithm

filterObject,logicalFilterResult-method	<i>construct logicalGate obj</i>
-----------------------------------------	----------------------------------

Description

construct logicalGate obj

Usage

```
## S4 method for signature 'logicalFilterResult'
filterObject(x)
```

Arguments

x	logicalFilterResult
---	---------------------

flowClust.1d	<i>Applies flowClust to 1 feature to determine a cutpoint between the minimum cluster and all other clusters.</i>
--------------	-------------------------------------------------------------------------------------------------------------------

Description

We cluster the observations in fr into K clusters.

Usage

```
flowClust.1d(fr, params, filterId = "", K = NULL, trans = 0,
  positive = TRUE, prior = NULL, criterion = c("BIC", "ICL"),
  cutpoint_method = c("boundary", "min_density", "quantile", "posterior_mean",
  "prior_density"), neg_cluster = 1, cutpoint_min = NULL,
  cutpoint_max = NULL, min = NULL, max = NULL, quantile = 0.99,
  quantile_interval = c(0, 10), plot = FALSE, debug = FALSE, ...)
```

Arguments

fr	a flowFrame object
params	character channel to be gated on
filterId	A character string that identifies the filter created.
K	the number of clusters to find
trans	numeric indicating whether the Box-Cox transformation parameter is estimated from the data. May take 0 (no estimation), 1 (estimation) or 2 (cluster-specific estimation). NOTE: for the Bayesian version of <code>flowClust</code> , this value cannot be 2.
positive	If TRUE, then the gate consists of the entire real line to the right of the cutpoint. Otherwise, the gate is the entire real line to the left of the cutpoint. (Default: TRUE)
prior	list of prior parameters for the Bayesian <code>flowClust</code> . If NULL, no prior is used.
criterion	a character string stating the criterion used to choose the best model. May take either "BIC" or "ICL". This argument is only relevant when K is NULL or $\text{length}(K) > 1$. The value selected is passed to <code>flowClust</code> .
cutpoint_method	How should the cutpoint be chosen from the fitted <code>flowClust</code> model? See Details.
neg_cluster	integer. The index of the negative cluster. The cutpoint is computed between clusters <code>neg_cluster</code> and <code>neg_cluster + 1</code> .
cutpoint_min	numeric value that sets a minimum threshold for the cutpoint. If a value is provided, any cutpoint below this value will be set to the given minimum value. If NULL (default), there is no minimum cutpoint value.
cutpoint_max	numeric value that sets a maximum threshold for the cutpoint. If a value is provided, any cutpoint above this value will be set to the given maximum value. If NULL (default), there is no maximum cutpoint value.
min	a numeric value that sets the lower bound for data filtering. If NULL (default), no truncation is applied.
max	a numeric value that sets the upper bound for data filtering. If NULL (default), no truncation is applied.
quantile	the quantile for which we will find the cutpoint using the quantile <code>cutpoint_method</code> . If the <code>cutpoint_method</code> is not set to <code>quantile</code> , this argument is ignored.
quantile_interval	a vector of length 2 containing the end-points of the interval of values to find the quantile cutpoint. If the <code>cutpoint_method</code> is not set to <code>quantile</code> , this argument is ignored.
plot	logical value indicating that the fitted <code>flowClust</code> model should be plotted along with the cutpoint
debug	logical indicating whether to carry the prior and posteriors with the gate for debugging purpose. Default is FALSE.
...	additional arguments that are passed to <code>flowClust</code>

Details

By default, the cutpoint is chosen to be the boundary of the first two clusters. That is, between the first two cluster centroids, we find the midpoint between the largest observation from the first cluster and the smallest observations from the second cluster. Alternatively, if the `cutpoint_method` is `min_density`, then the cutpoint is the point at which the density between the first and second smallest cluster centroids is minimum.

Value

a `rectangleGate` object consisting of all values beyond the cutpoint calculated

Examples

```
## Not run:
gate <- flowClust.1d(fr, params = "APC-A", K = 2) # fr is a flowFrame

## End(Not run)
```

flowClust.2d	<i>Automatic identification of a population of interest via flowClust based on two markers</i>
--------------	------------------------------------------------------------------------------------------------

Description

We cluster the observations in `fr` into `K` clusters. We set the cutpoint to be the point at which the density between the first and second smallest cluster centroids is minimum.

Usage

```
flowClust.2d(fr, xChannel, yChannel, filterId = "", K = 2,
  usePrior = "no", prior = list(NA), trans = 0, plot = FALSE,
  target = NULL, transitional = FALSE, quantile = 0.9,
  translation = 0.25, transitional_angle = NULL, min = NULL, max = NULL,
  ...)
```

Arguments

<code>fr</code>	a <code>flowFrame</code> object
<code>xChannel,yChannel</code>	character specifying channels to be gated on
<code>filterId</code>	A character string that identifies the filter created.
<code>K</code>	the number of clusters to find
<code>usePrior</code>	Should we use the Bayesian version of <code>flowClust</code> ? Answers are "yes", "no", or "vague". The answer is passed along to <code>flowClust</code> .
<code>prior</code>	list of prior parameters for the Bayesian version of <code>flowClust</code> . If <code>usePrior</code> is set to no, then the list is unused.

trans	numeric indicating whether the Box-Cox transformation parameter is estimated from the data. May take 0 (no estimation), 1 (estimation) or 2 (cluster-specific estimation). NOTE: for the Bayesian version of <code>flowClust</code> , this value cannot be 2.
plot	a logical value indicating if the fitted mixture model should be plotted. By default, no.
target	a numeric vector of length 2 (number of dimensions) containing the location of the cluster of interest. See details.
transitional	logical value indicating if a transitional gate should be constructed from the target <code>flowClust</code> cluster. By default, no.
quantile	the contour level of the target cluster from the <code>flowClust</code> fit to construct the gate
translation	a numeric value between 0 and 1 used to position a transitional gate if <code>transitional = TRUE</code> . This argument is ignored if <code>transitional = FALSE</code> . See details
transitional_angle	the angle (in radians) of the transitional gate. See details. Ignored if <code>transitional = FALSE</code> .
min	A vector of length 2. Truncate observations less than this minimum value. The first value truncates the <code>xChannel</code> , and the second value truncates the <code>yChannel</code> . By default, this vector is NULL and is ignored.
max	A vector of length 2. Truncate observations greater than this maximum value. The first value truncates the <code>xChannel</code> , and the second value truncates the <code>yChannel</code> . By default, this vector is NULL and is ignored.
...	additional arguments that are passed to <code>flowClust</code>

Details

The cluster for the population of interest is selected as the one with cluster centroid nearest the target in Euclidean distance. By default, the largest cluster (i.e., the cluster with the largest proportion of observations) is selected as the population of interest.

We also provide the option of constructing a transitional gate from the selected population of interest. The location of the gate can be controlled with the `translation` argument, which translates the gate along the major axis of the target cluster as a function of the appropriate chi-squared coefficient. The larger `translation` is, the more gate is shifted in a positive direction. Furthermore, the width of the transitional gate can be controlled with the `quantile` argument.

The direction of the transitional gate can be controlled with the `transitional_angle` argument. By default, it is NULL, and we use the eigenvector of the target cluster that points towards the first quadrant (has positive slope). If `transitional_angle` is specified, we rotate the eigenvectors so that the angle between the x-axis (with the cluster centroid as the origin) and the major eigenvector (i.e., the eigenvector with the larger eigenvalue) is `transitional_angle`.

Value

a `polyGate` object containing the contour (ellipse) for 2D gating.

Examples

```
## Not run:
gate <- flowClust.2d(fr, xChannel = "FSC-A", yChannel = "SSC-A", K = 3) # fr is a flowFrame

## End(Not run)
```

gating *Apply the gates to a GatingSet*

Description

It applies the gates to the GatingSet based on the population tree described in graphGML.

It loads the gating methods by topological order and applies them to GatingSet.

Usage

```
gating(x, y, ...)

## S4 method for signature 'gatingTemplate,GatingSet'
gating(x, y, env_fct = NULL, ...)

## S4 method for signature 'gatingTemplate,GatingSetList'
gating(x, y, env_fct = NULL, ...)
```

Arguments

x	a gatingTemplate object
y	a GatingSet object
...	<ul style="list-style-type: none"> start a character that specifies the population (corresponding to 'alias' column in csv template) where the gating process will start from. It is useful to quickly skip some gates and go directly to the target population in the testing run. Default is "root". stop.at a character that specifies the population (corresponding to 'alias' column in csv template) where the gating process will stop at. Default is NULL, indicating the end of gating tree. mc.cores passed to multicore package for parallel computing parallel_type character specifying the parallel type. The valid options are "none", "multicore", "cluster". cl cluster object passed to parallel package (when parallel_type is "cluster")
env_fct	a environment that contains fcTree object named as 'fct'. If NULL (by default), no fcTree will be constructed. It is currently reserved for the internal debugging.

Value

Nothing. As the side effect, gates generated by gating methods are saved in GatingSet.

Examples

```
## Not run:
gt <- gatingTemplate(file.path(path, "data/ICStemplate.csv"), "ICS")
gs <- GatingSet(fs) #fs is a flowSet/ncdfFlowSet
gating(gt, gs)
gating(gt, gs, stop.at = "v") #proceed the gating until population 'v'
gating(gt, gs, start = "v") # start from 'v'
gating(gt, gs, parallel_type = "multicore", mc.cores = 8) #parallel gating using multicore
#parallel gating by using cluster
cl1 <- makeCluster (8, type = "MPI")
gating(gt, gs, parallel_type = "cluster", cl = cl1)
stopCluster ( cl1 )

## End(Not run)
```

gating, boolMethod, GatingSet-method
apply a boolMethod to the GatingSet

Description

apply a boolMethod to the GatingSet

Usage

```
## S4 method for signature 'boolMethod,GatingSet'
gating(x, y, ...)
```

Arguments

x	boolMethod
y	GatingSet
...	other arguments

gating,gtMethod,GatingSet-method

apply a [gtMethod](#) to the GatingSet

Description

apply a [gtMethod](#) to the GatingSet

Usage

```
## S4 method for signature 'gtMethod,GatingSet'  
gating(x, y, ...)
```

Arguments

x	gtMethod
y	GatingSet
...	other arguments

gating,polyFunctions,GatingSet-method

apply a polyFunctions gating method to the GatingSet

Description

It generates a batch of boolMethods based on the expression defined in polyFunctions objects. It is a convenience way to generate different boolean combinations of cytokine gates.

Usage

```
## S4 method for signature 'polyFunctions,GatingSet'  
gating(x, y, ...)
```

Arguments

x	polyFunctions
y	GatingSet
...	other arguments

gating, refGate, GatingSet-method
apply a refGate to the GatingSet

Description

apply a refGate to the GatingSet

Usage

```
## S4 method for signature 'refGate,GatingSet'
gating(x, y, ...)
```

Arguments

x	refGate
y	GatingSet
...	other arguments

gatingTemplate-class *a class storing the gating method and population information in a graphNEL object*

Description

Each cell population is stored in graph node and is connected with its parent population or its reference node for boolGate or refGate.

It parses the csv file that specifies the gating scheme for a particular staining pannel.

Usage

```
gatingTemplate(x, ...)
```

```
## S4 method for signature 'character'
gatingTemplate(x, name = "default", strict = TRUE,
  ...)
```

Arguments

x	character csv file name
...	other arguments passed to data.table::fread
name	character the label of the gating template

strict logical whether to perform validity check(special characters) on the alias column. By default it is(and should be) turned on for the regular template parsing. But sometime it is useful to turned it off to bypass the check for the dummy nodes(e.g. the csv template generated by 'templateGen' with some existing boolean gates that has '!' or ':' symbol).

Details

This csv must have the following columns:

'alias': a name used label the cell population, the path composed by the alias and its precedent nodes (e.g. /root/A/B/alias) has to be uniquely identifiable. So alias can not contain '/' character, which is reserved as path delimiter.

'pop': population patterns of 'A+/-' or 'A+/-B+/-', which tells the algorithm which side (postive or negative) of 1d gate or which quadrant of 2d gate to be kept. when it is in the form of 'A+/-B+/-', 'A' and 'B' should be the full name (or a substring as long as it is unquely matched) of either channel or marker of the flow data.

'parent': the parent population alias, its path has to be uniquely identifiable.

'dims': characters seperated by comma specifying the dimensions(1d or 2d) used for gating. It can be either channel name or stained marker name.

'gating_method': the name of the gating function (e.g. 'flowClust'). It is invoked by a wrapper function that has the identical function name prefixed with a dot.(e.g. '.flowClust')

'gating_args': the named arguments passed to gating function (Note that double quotes are often used as text delimiter by some csv editors. So try to use single quote instead if needed.)

'collapseDataForGating': When TRUE, data is collapsed (within groups if 'groupBy' specified) before gating and the gate is replicated across collapsed samples. When set FALSE (or blank),then 'groupBy' argument is only used by 'preprocessing' and ignored by gating.

'groupBy': If given, samples are split into groups by the unique combinations of study variable (i.e. column names of pData,e.g."PTID:VISITNO"). when split is numeric, then samples are grouped by every N samples

'preprocessing_method': the name of the preprocessing function(e.g. 'prior_flowClust'). It is invoked by a wrapper function that has the identical function name prefixed with a dot.(e.g. '.prior_flowClust') the preprocessing results are then passed to gating wrapper function through 'pps_res' argument.

'preprocessing_args': the named arguments passed to preprocessing function.

Examples

```
## Not run:
gt <- gatingTemplate(system.file("extdata/gating_template/tcell.csv",package = "openCyto"))
plot(gt)

## End(Not run)
```

`getChildren,gatingTemplate,character-method`
get children nodes

Description

get children nodes

Usage

```
## S4 method for signature 'gatingTemplate,character'  
getChildren(obj, y)
```

Arguments

<code>obj</code>	<code>gatingTemplate</code>
<code>y</code>	character parent node path

Examples

```
## Not run:  
gt <- gatingTemplate(system.file("extdata/gating_template/tcell.csv",package = "openCyto"))  
  
getNode(gt, "/nonDebris")  
getChildren(gt, "/nonDebris")  
  
## End(Not run)
```

`getGate,fcTree,character-method`
get gates saved in fcTree

Description

get gates saved in fcTree

Usage

```
## S4 method for signature 'fcTree,character'  
getGate(obj, y, ...)
```

Arguments

<code>obj</code>	<code>fcTree</code>
<code>y</code>	character node name
<code>...</code>	other arguments (not used)

getGate,gatingTemplate,character-method
get gating method from the node

Description

get gating method from the node

Usage

```
## S4 method for signature 'gatingTemplate,character'  
getGate(obj, y, z)
```

Arguments

obj	gatingTemplate
y	character parent node path
z	character child node path

Examples

```
## Not run:  
gt <- gatingTemplate(system.file("extdata/gating_template/tcell.csv",package = "openCyto"))  
getNodes(gt, only.names = TRUE)  
getNodes(gt, "/nonDebris")  
getChildren(gt, "/nonDebris")  
getGate(gt, "/nonDebris", "/nonDebris/singlets")  
  
## End(Not run)
```

getNodes,fcTree-method
get nodes from fcTree

Description

get nodes from fcTree

Usage

```
## S4 method for signature 'fcTree'  
getNodes(x, y)
```

Arguments

x	fcTree
y	character node name

getNode, gatingTemplate-method
get nodes from [gatingTemplate](#) object

Description

get nodes from [gatingTemplate](#) object

Usage

```
## S4 method for signature 'gatingTemplate'
getNode(x, y, order = c("default", "bfs", "dfs",
  "tsort"), only.names = FALSE)
```

Arguments

x	gatingTemplate
y	character node index. When missing, return all the nodes
order	character specifying the order of nodes. options are "default", "bfs", "dfs", "tsort"
only.names	logical specifying whether user wants to get the entire gtPopulation object or just the name of the population node

Examples

```
## Not run:
gt <- gatingTemplate(system.file("extdata/gating_template/tcell.csv", package = "openCyto"))
getNode(gt)[1:2]
getNode(gt, only.names = TRUE)
getNode(gt, "/nonDebris")

## End(Not run)
```

getParent,gatingTemplate,character-method
get parent nodes

Description

get parent nodes

Usage

```
## S4 method for signature 'gatingTemplate,character'
getParent(obj, y, isRef = FALSE)
```


Arguments

obj	gatingTemplate
y	character child node path
isRef	logical whether show the reference node besides the parent node

Examples

```
## Not run:
gt <- gatingTemplate(system.file("extdata/gating_template/tcell.csv",package = "openCyto"))

getNode(gt, "/nonDebris")
getParent(gt, "/nonDebris/singlets")

## End(Not run)
```

```
groupBy,gtMethod-method
```

get the grouping variable for the gating method

Description

When specified, the flow data is grouped by the grouping variable (column names in pData). Within each group, when isCollapse is set to TRUE, the gating method is applied to the collapsed data. Otherwise, it is done independently for each individual sample(flowFrame). Grouping variable is also used by preprocessing method.

Usage

```
## S4 method for signature 'gtMethod'
groupBy(object)
```

Arguments

object	gtMethod
--------	----------

```
gtMethod-class
```

A class to represent a gating method.

Description

A gating method object contains the specifics for generating the gates.

Slots

- name** a character specifying the name of the gating method
- dims** a character vector specifying the dimensions (channels or markers) of the gate
- args** a list specifying the arguments passed to gating function
- groupBy** a character or integer specifying how to group the data if character, group the data by the study variables (columns in pData) if integer, group the data by every N samples.
- collapse** a logical specifying whether to collapse the data within group before gating. it is only valid when groupBy is specified

Examples

```
## Not run:
gt <- gatingTemplate(system.file("extdata/gating_template/tcell.csv", package = "openCyto"))
getGate(gt, '2', '3')

## End(Not run)
```

gtPopulation-class	<i>A class to represent a cell population that will be generated by a gating method.</i>
--------------------	------------------------------------------------------------------------------------------

Description

A class to represent a cell population that will be generated by a gating method.

Slots

- id** numeric unique ID that is consistent with node label of graphNEL in gating template
- name** character the name of population
- alias** character the more user friendly name of population

Examples

```
## Not run:
gt <- gatingTemplate(system.file("extdata/gating_template/tcell.csv", package = "openCyto"))

getNodes(gt, '2')

## End(Not run)
```

gtSubsets-class	<i>A class representing a group of cell populations.</i>
-----------------	----------------------------------------------------------

Description

It extends gtPopulation class.

`isCollapse,gtMethod-method`

get the flag that determines whether gating method is applied on collapsed data

Description

When TRUE, the flow data(multiple flowFrames) is collapsed into one and the gating method is applied on the collapsed data. Once the gate is generated, it is then replicated and applied to the each single flowFrame.

Usage

```
## S4 method for signature 'gtMethod'  
isCollapse(object)
```

Arguments

object gtMethod

Value

logical

`listgtMethods`

Print a list of the registered gating methods

Description

Print a list of the registered gating methods

Usage

```
listgtMethods()
```

Value

Does not return anything. Prints a list of the available gating methods.

mindensity	<i>Determines a cutpoint as the minimum point of a kernel density estimate between two peaks</i>
------------	--------------------------------------------------------------------------------------------------

Description

We fit a kernel density estimator to the cells in the `flowFrame` and identify the two largest peaks. We then select as the cutpoint the value at which the minimum density is attained between the two peaks of interest.

Usage

```
mindensity(fr, channel, filterId = "", positive = TRUE, pivot = FALSE,
  gate_range = NULL, min = NULL, max = NULL, peaks = NULL, ...)
```

Arguments

<code>fr</code>	a <code>flowFrame</code> object
<code>channel</code>	TODO
<code>filterId</code>	TODO
<code>positive</code>	If TRUE, then the gate consists of the entire real line to the right of the cutpoint. Otherwise, the gate is the entire real line to the left of the cutpoint. (Default: TRUE)
<code>pivot</code>	logical value. If TRUE, we choose as the two peaks the largest peak and its neighboring peak. See details.
<code>gate_range</code>	numeric vector of length 2. If given, this sets the bounds on the gate applied. If no gate is found within this range, we set the gate to the minimum value within this range if <code>positive</code> is TRUE and the maximum value of the range otherwise.
<code>min</code>	a numeric value that sets the lower boundary for data filtering
<code>max</code>	a numeric value that sets the upper boundary for data filtering
<code>peaks</code>	numeric vector. If not given, then perform peak detection first by <code>.find_peaks</code>
<code>...</code>	Additional arguments for peak detection.

Details

In the default case, the two peaks of interest are the two largest peaks obtained from the `link{density}` function. However, if `pivot` is TRUE, we choose the largest peak and its neighboring peak as the two peaks of interest. In this case, the neighboring peak is the peak immediately to the left of the largest peak if `positive` is TRUE. Otherwise, the neighboring peak is selected as the peak to the right.

In the special case that there is only one peak, we are conservative and set the cutpoint as the $\min(x)$ if `positive` is TRUE, and the $\max(x)$ otherwise.

Value

a `rectangleGate` object based on the minimum density cutpoint

Examples

```
## Not run:
gate <- mindensity(fr, channel = "APC-A") # fr is a flowFrame

## End(Not run)
```

mindensity2	<i>An improved version of mindensity used to determines a cutpoint as the minimum point of a kernel density estimate between two peaks.</i>
-------------	---------------------------------------------------------------------------------------------------------------------------------------------

Description

Analogous to the original openCyto::mindensity(), mindensity2 operates on a standard flowFrame. Its behavior is closely modeled on the original mindensity() whenever possible. However, the underlying peak-finding algorithm (improvedMindensity) behaves significantly differently.

Usage

```
mindensity2(fr, channel, filterId = "", positive = TRUE, pivot = FALSE,
  gate_range = NULL, min = NULL, max = NULL, peaks = NULL, ...)
```

Arguments

fr	a flowFrame object
channel	the channel to operate on
filterId	a name to refer to this filter
positive	If TRUE, then the gate consists of the entire real line to the right of the cutpoint. Otherwise, the gate is the entire real line to the left of the cutpoint. (Default: TRUE)
pivot	logical value. If TRUE, we choose as the two peaks the largest peak and its neighboring peak. See details.
gate_range	numeric vector of length 2. If given, this sets the bounds on the gate applied. If no gate is found within this range, we set the gate to the minimum value within this range if positive is TRUE and the maximum value of the range otherwise.
min	a numeric value that sets the lower boundary for data filtering
max	a numeric value that sets the upper boundary for data filtering
peaks	numeric vector. If not given, then perform peak detection first by .find_peaks
...	Additional arguments for peak detection.

Value

a rectangleGate object based on the minimum density cutpoint

Author(s)

Greg Finak, Phu T. Van

Examples

```
## Not run:  
gate <- mindensity2(fr, channel = "APC-A") # fr is a flowFrame  
  
## End(Not run)
```

names.gtMethod-method *get gating method name*

Description

get gating method name

Usage

```
## S4 method for signature 'gtMethod'  
names(x)
```

Arguments

x gtMethod

Examples

```
## Not run:  
gt <- gatingTemplate(system.file("extdata/gating_template/tcell.csv", package = "openCyto"))  
  
gtMthd <- getGate(gt, "/nonDebris/singlets", "/nonDebris/singlets/lymph")  
names(gtMthd)  
dims(gtMthd)  
parameters(gtMthd)  
isCollapse(gtMthd)  
groupBy(gtMthd)  
  
gtPop <- getNodes(gt, "/nonDebris/singlets/lymph/cd3/cd4+cd8-/CD38+")  
names(gtPop)  
alias(gtPop)  
  
## End(Not run)
```

names,gtPopulation-method
get population name

Description

get population name

Usage

```
## S4 method for signature 'gtPopulation'  
names(x)
```

Arguments

x gtPopulation object

ocRectangleGate-class *the class that carries event indices as well*

Description

the class that carries event indices as well

ocRectRefGate *constructor for ocRectRefGate*

Description

constructor for ocRectRefGate

Usage

```
ocRectRefGate(rectGate, boolExprs)
```

Arguments

rectGate rectangleGate
boolExprs character boolean expression of reference nodes

ocRectRefGate-class *special gate type that mix the rectangleGate with boolean gate*

Description

special gate type that mix the rectangleGate with boolean gate

openCyto *Hierarchical Gating Pipeline for flow cytometry data*

Description

Hierarchical Gating Pipeline for flow cytometry data.

Details

openCyto is a package designed to facilitate the automated gating methods in sequential way to mimic the manual gating strategy.

Package:	openCyto
Type:	Package
Version:	1.2.8
Date:	2014-04-10
License:	GPL (>= 2)
LazyLoad:	yes

Author(s)

Mike Jiang <wjiang2@fhcrc.org>, John Ramey <jramey@fhcrc.org>, Greg Finak <gfinak@fhcrc.org>

Maintainer: Mike Jiang <wjiang2@fhcrc.org>

See Also

See [gating](#), [flowClust.1d](#), for an overview of gating functions.

Examples

```
## Not run: gatingTemplate('test.csv')
```

```
parameters.gtMethod-method
    get parameters of the gating method/function
```

Description

get parameters of the gating method/function

Usage

```
## S4 method for signature 'gtMethod'
parameters(object)
```

Arguments

```
object          gtMethod
```

```
plot,fcFilterList,ANY-method
    plot a fcFilterList
```

Description

It is usually called by plot method for fcTree instead of directly by users.

Usage

```
## S4 method for signature 'fcFilterList,ANY'
plot(x, y, samples = NULL, posteriors = FALSE,
     xlim = NULL, ylim = NULL, node = NULL, data = NULL, breaks = 20,
     lwd = 1, ...)
```

Arguments

x	fcFilterList
y	character channel name
samples	character a vector of sample names to be plotted
posteriors	logical indicating whether posteriors should be plotted
xlim,ylim	scale settings for x,y axes
node	character population name associated with the fcFilterList
data	GatingSet object
breaks	passed to hist
lwd	line width
...	other arguments passed to base plot

Examples

```
## Not run:
env1<-new.env(parent=emptyenv())
#gt is a gatingTemplate, gs is a GatingSet
gating(gt,gs,env1) #the flowClust gating results are stored in env1
plot(env1$fct,"nonDebris",post=T) #plot the priors as well as posteriors for the "nonDebris" gate

## End(Not run)
```

```
plot,fcTree,character-method
      plot the flowClust gating results
```

Description

This provides the priors and posteriors as well as the gates for the purpose of debugging flowClust gating algorithm

Usage

```
## S4 method for signature 'fcTree,character'
plot(x, y, channel = NULL, data = NULL, ...)
```

Arguments

x	fcTree
y	character node name in the fcTree
channel	character specifying the channel.
data	GatingSet that the fcTree is associated with
...	other arguments

```
plot,gatingTemplate,missing-method
      plot the gating scheme
```

Description

plot the gating scheme using Rgraphviz

Usage

```
## S4 method for signature 'gatingTemplate,missing'
plot(x, y, ...)
```

Arguments

x	gatingTemplate object
y	either character specifying the root node which can be used to visualize only the subgraph or missing which display the entire gating scheme
...	other arguments

graphAttr, nodeAttr: graph rendering attributes passed to [renderGraph](#) showRef logical: whether to display the reference gates. Sometime it maybe helpful to hide all those reference gates which are not the cell population of interest and used primarily for generating other population nodes.

Examples

```
## Not run:
gt <- gatingTemplate(system.file("extdata/gating_template/tcell.csv",package = "openCyto"))
plot(gt) #plot entire tree
plot(gt, "lymph") #only plot the subtree rooted from "lymph"

## End(Not run)
```

polyFunctions-class *A class to represent a polyFunctions gating method.*

Description

It extends boolMethod class and will be expanded to multiple boolMethod object.

posteriors, fcFilter, ANY-method
get posteriors from a fcFilter object

Description

get posteriors from a fcFilter object

Usage

```
## S4 method for signature 'fcFilter,ANY'
posteriors(x, y = "missing")
```

Arguments

x	fcFilter
y	character or missing that specify which channel to look for

```
ppMethod,gatingTemplate,character-method
  get preprocessing method from the node
```

Description

get preprocessing method from the node

Usage

```
## S4 method for signature 'gatingTemplate,character'
ppMethod(obj, y, z)
```

Arguments

obj	gatingTemplate
y	character parent node path
z	character child node path

Examples

```
## Not run:
gt <- gatingTemplate(system.file("extdata/gating_template/tcell.csv",package = "openCyto"))
ppMethod(gt, "/nonDebris/singlets", "/nonDebris/singlets/lymph")

## End(Not run)
```

```
ppMethod-class      A class to represent a preprocessing method.
```

Description

It extends gtMethod class.

Examples

```
## Not run:
gt <- gatingTemplate(system.file("extdata/gating_template/tcell.csv",package = "openCyto"))
ppMethod(gt, '3', '4')

## End(Not run)
```

preprocessing,ppMethod,GatingSet-method
apply a [ppMethod](#) to the GatingSet

Description

apply a [ppMethod](#) to the GatingSet

Usage

```
## S4 method for signature 'ppMethod,GatingSet'  
preprocessing(x, y, ...)
```

Arguments

x	ppMethod
y	GatingSet or GatingSetList
...	other arguments

priors,fcFilter,ANY-method
get priors from a fcFilter object

Description

get priors from a fcFilter object

Usage

```
## S4 method for signature 'fcFilter,ANY'  
priors(x, y = "missing")
```

Arguments

x	fcFilter object
y	character specifying channel name. if missing then extract priors for all the channels

prior_flowClust *Elicits data-driven priors from a flowSet object for specified channels*

Description

We elicit data-driven prior parameters from a `flowSet` object for specified channels. For each sample in the `flowSet` object, we apply the given `prior_method` to elicit the priors parameters.

Usage

```
prior_flowClust(flow_set, channels, prior_method = c("kmeans"), K = 2,
  nu0 = 4, w0 = c(10, 10), shrink = 1e-06, ...)
```

Arguments

<code>flow_set</code>	a <code>flowSet</code> object
<code>channels</code>	a character vector containing the channels in the <code>flowSet</code> from which we elicit the prior parameters for the Student's t mixture
<code>prior_method</code>	the method to elicit the prior parameters
<code>K</code>	the number of mixture components to identify
<code>nu0</code>	prior degrees of freedom of the Student's t mixture components.
<code>w0</code>	the number of prior pseudocounts of the Student's t mixture components. (only the first element is used and the rest is ignored at the moment)
<code>shrink</code>	the amount of eigenvalue shrinkage to add in the case the prior covariance matrices are singular. See details.
<code>...</code>	Additional arguments passed to the prior elicitation method selected

Details

Currently, we have implemented only two methods. In the case that one channel is given, we use the kernel-density estimator (KDE) approach for each sample to obtain `K` peaks from which we elicit prior parameters. Otherwise, if more than one channel is specified, we apply K-Means to each of the samples in the `flowSet` and aggregate the clusters to elicit the prior parameters.

In the rare case that a prior covariance matrix is singular, we shrink the eigenvalues of the matrix slightly to ensure that it is positive definite. For instance, if the `flow_set` has two samples, this case can occur. The amount of shrinkage is controlled in `shrink`.

Value

list of the necessary prior parameters

quadGate.seq	<i>sequential quadrant gating function</i>
--------------	--------------------------------------------

Description

The order of 1d-gating is determined so that the gates better capture the distributions of flow data.

Usage

```
quadGate.seq(fr, channels, gFunc, min = NULL, max = NULL, ...)
```

Arguments

fr	flowFrame
channels	character two channels used for gating
gFunc	the name of the 1d-gating function to be used for either dimension
min	a numeric vector that sets the lower bounds for data filtering
max	a numeric vector that sets the upper bounds for data filtering
...	other arguments passed to <code>.find_peak</code> (e.g. <code>'num_peaks'</code> and <code>'adjust'</code>). see tailgate

Value

a filters that contains four rectangleGates

quadGate.tmix	<i>quadGate based on flowClust::tmixFiler</i>
---------------	-----------------------------------------------

Description

This gating method identifies two quadrants (first, and third quadrants) by fitting the data with tmixture model. It is particularly useful when the two markers are not well resolved thus the regular quadGate method based on 1d gating will not find the perfect cut points on both dimensions.

Usage

```
quadGate.tmix(fr, channels, K, usePrior = "yes", prior = list(NA),
  quantile1 = 0.8, quantile3 = 0.8, trans = 0, plot = FALSE, ...)
```

Arguments

fr	flowFrame
channels	character vector specifies two channels
K	see flowClust.2d
usePrior	see flowClust.2d
prior	see flowClust.2d
quantile1	numeric specifies the quantile level(see 'level' in flowClust) for the first quadrant (x-y+)
quantile3	numeric specifies the quantile level see 'level' in flowClust for third quadrant (x+y-)
trans	see flowClust.2d
plot	logical whether to plot flowClust clustering results
...	other arguments passed to flowClust

Value

a filters object that contains four polygonGates following the order of (-+,++,+,-)

quantileGate	<i>Determine the cutpoint by the events quantile.</i>
--------------	-------------------------------------------------------

Description

It is possible that the cutpoint calculated by quantile function may not produce the exact the probability set by 'probs' argument if there are not enough cell events to reach that precision. Sometime the difference could be significant.

Usage

```
quantileGate(fr, channel, probs = 0.999, plot = FALSE, positive = TRUE,
  filterId = "", min = NULL, max = NULL, ...)
```

Arguments

fr	a flowFrame object
channel	the channel from which the cytokine gate is constructed
probs	probabilities passed to 'stats::quantile' function.
plot	whether to plot the gate result
positive	If TRUE, then the gate consists of the entire real line to the right of the cutpoint. Otherwise, the gate is the entire real line to the left of the cutpoint. (Default: TRUE)
filterId	the name of the filter
min	a numeric value that sets the lower boundary for data filtering
max	a numeric value that sets the upper boundary for data filtering
...	additional arguments passed to 'stats::quantile' function.

Value

a rectangleGate

Examples

```
## Not run:
gate <- quantileGate(fr, Channel = "APC-A", probs = 0.995) # fr is a flowFrame

## End(Not run)
```

refGate-class	<i>A class to represent a reference gating method.</i>
---------------	--------------------------------------------------------

Description

It extends gtMethod class.

Slots

refNodes character specifying the reference nodes

registerPlugins	<i>Register a gating or preprocessing function with OpenCyto</i>
-----------------	------------------------------------------------------------------

Description

Function registers a new gating or preprocessing method with openCyto so that it may be used in the csv template.

Usage

```
registerPlugins(fun = NA, methodName, dep = NA, ...)
```

Arguments

fun	function to be registered
methodName	character name of the gating or preprocessing method
dep	character name of the library dependency required for the plugin method to work.
...	other arguments type character specifying the type of registering method. Should be either "gating" or "preprocessing".

Details

The fun argument should be a wrapper function definition for the gating or preprocessing method. Gating method must have formal arguments:

fr a flowFrame

pp_res a pre-processing result

xChannel character (optional)

yChannel character (required)

filterId character

... ellipses for the additional parameters.

Preprocessing method must have formal arguments:

fs a flowSet that stores the flow data (could be subgrouped data if groupBy column is defined in the csv template)

gs a GatingSet

gm a gtMethod object that stores the information from gating method

xChannel character (required)

yChannel character (required)

... ellipses for the additional parameters.

The gating function must return a filter (i.e. polygonGate or other instance) from flowCore. The preprocessing can return anything and it will be passed on to the gating function. So it is up to gating function to use and interpret the results of preprocessing. Not all formal parameters need to be used. Additional arguments are passed via the ... and can be processed in the wrapper

Value

logical TRUE if successful and prints a message. FALSE otherwise.

show,boolMethod-method

show method for boolMethod

Description

show method for boolMethod

Usage

```
## S4 method for signature 'boolMethod'
show(object)
```

Arguments

object boolMethod

show,fcFilter-method *show method for fcFilter*

Description

show method for fcFilter

Usage

```
## S4 method for signature 'fcFilter'  
show(object)
```

Arguments

object fcFilter show method for fcFilter

show,gatingTemplate-method
show method for gatingTemplate

Description

show method for gatingTemplate

Usage

```
## S4 method for signature 'gatingTemplate'  
show(object)
```

Arguments

object gatingTemplate

```
show,gtMethod-method  show method for gtMethod
```

Description

show method for gtMethod

Usage

```
## S4 method for signature 'gtMethod'
show(object)
```

Arguments

object gtMethod show method for gtMethod

tailgate *Gates the tail of a density using the derivative of a kernel density estimate*

Description

Gates the tail of a density using the derivative of a kernel density estimate

Usage

```
tailgate(fr, channel, filterId = "", num_peaks = 1, ref_peak = 1,
  strict = TRUE, tol = 0.01, positive = TRUE, side = "right",
  min = NULL, max = NULL, ...)
```

```
cytokine(fr, channel, filterId = "", num_peaks = 1, ref_peak = 1,
  tol = 0.01, positive = TRUE, side = "right", ...)
```

Arguments

fr a flowFrame object

channel the channel from which the cytokine gate is constructed

filterId the name of the filter

num_peaks the number of peaks expected to see. This effectively removes any peaks that are artifacts of smoothing

ref_peak After num_peaks are found, this argument provides the index of the reference population from which a gate will be obtained.

strict logical when the actual number of peaks detected is less than ref_peak. an error is reported by default. But if strict is set to FALSE, then the reference peak will be reset to the peak of the far right.

tol	the tolerance value used to construct the cytokine gate from the derivative of the kernel density estimate
positive	If TRUE, then the gate consists of the entire real line to the right of the cutpoint. Otherwise, the gate is the entire real line to the left of the cutpoint. (Default: TRUE)
side	On which side of the density do we want to gate the tail, the 'right' (default) or 'left'?
min	a numeric value that sets the lower boundary for data filtering
max	a numeric value that sets the upper boundary for data filtering
...	additional arguments.

Value

a filterList containing the gates (cutpoints) for each sample

Examples

```
## Not run:
gate <- tailgate(fr, Channel = "APC-A") # fr is a flowFrame

## End(Not run)
```

templateGen	<i>generate a partially complete csv template from the existing gating hierarchy</i>
-------------	--------------------------------------------------------------------------------------

Description

To ease the process of replicating the existing (usually a manual one) gating schemes, this function populate an empty gating template with the 'alias', 'pop', 'parent' and 'dims' columns that exacted from an GatingHierarchy, and leave the other columns (e.g. 'gating_method') blank. So users can make changes to that template instead of writing from scratch.

Usage

```
templateGen(gh)
```

Arguments

gh a GatingHierarchy likely parsed from a xml workspace

Value

a gating template in data.frame format that requires further edition after output to csv

Index

*Topic **package**
 openCyto, 32
 .isPolyfunctional, 3
 .prior_flowClust1d, 3
 .prior_kmeans, 5

 add, 7
 add,GatingHierarchy,logicalFilterResult-method
 (add,GatingHierarchy,multipleFilterResult-method), 6
 add,GatingHierarchy,multipleFilterResult-method, 6
 add,GatingHierarchy,ocRectangleGate-method
 (add,GatingHierarchy,multipleFilterResult-method), 6
 add,GatingHierarchy,ocRectRefGate-method
 (add,GatingHierarchy,multipleFilterResult-method), 6
 add_pop, 7
 as.data.table.gatingTemplate, 8

 boolMethod-class, 9

 cutree, 4, 5
 cytokine (tailgate), 44

 density, 4
 dims,gtMethod-method, 9
 dummyMethod-class, 9

 fcEllipsoidGate, 10
 fcEllipsoidGate-class, 10
 fcFilter-class, 10
 fcFilterList, 10
 fcFilterList-class, 11
 fcPolygonGate, 11
 fcPolygonGate-class, 11
 fcRectangleGate, 12
 fcRectangleGate-class, 12
 fcTree, 12
 fcTree-class, 13

 filterObject,logicalFilterResult-method, 13
 flowClust, 4, 14–16, 40
 flowClust.1d, 13, 32
 flowClust.2d, 15, 40

 gating, 17, 32
 gating,boolMethod,GatingSet-method, 18
 gating,boolMethod,GatingSetList-method
 (gating,boolMethod,GatingSet-method), 18
 gating,dummyMethod,GatingSet-method
 (gating,refGate,GatingSet-method), 18
 gating,dummyMethod,GatingSetList-method
 (gating,refGate,GatingSet-method), 18
 gating,gatingTemplate,GatingSet-method
 (gating), 17
 gating,gatingTemplate,GatingSetList-method
 (gating), 17
 gating,gtMethod,GatingSet-method, 19
 gating,gtMethod,GatingSetList-method
 (gating,gtMethod,GatingSet-method), 19
 gating,polyFunctions,GatingSet-method, 19
 gating,polyFunctions,GatingSetList-method
 (gating,polyFunctions,GatingSet-method), 19
 gating,refGate,GatingSet-method, 20
 gating,refGate,GatingSetList-method
 (gating,refGate,GatingSet-method), 20
 gatingTemplate, 8, 24
 gatingTemplate (gatingTemplate-class), 20
 gatingTemplate,character-method
 (gatingTemplate-class), 20
 gatingTemplate-class, 20

- getChildren, gatingTemplate, character-method, 22
- getGate, fcTree, character-method, 22
- getGate, gatingTemplate, character-method, 23
- getNodes, fcTree-method, 23
- getNodes, gatingTemplate-method, 24
- getParent, gatingTemplate, character-method, 24
- groupBy, gtMethod-method, 25
- gtMethod, 19
- gtMethod-class, 25
- gtPopulation-class, 26
- gtSubsets-class, 26

- hclust, 4, 5
- hist, 33

- isCollapse, gtMethod-method, 27

- listgtMethods, 27

- mindensity, 28
- mindensity2, 29

- names, gtMethod-method, 30
- names, gtPopulation-method, 31

- ocRectangleGate-class, 31
- ocRectRefGate, 31
- ocRectRefGate-class, 32
- openCyto, 32
- openCyto-package (openCyto), 32

- parameters, gtMethod-method, 33
- plot, fcFilterList, ANY-method, 33
- plot, fcTree, character-method, 34
- plot, gatingTemplate, ANY-method
 - (plot, gatingTemplate, missing-method), 34
- plot, gatingTemplate, character-method
 - (plot, gatingTemplate, missing-method), 34
- plot, gatingTemplate, missing-method, 34
- plot, gatingTemplate-method
 - (plot, gatingTemplate, missing-method), 34
- polyFunctions-class, 35
- posteriors, fcFilter, ANY-method, 35
 - (posteriors, fcFilter, ANY-method), 35
- ppMethod, 37
- ppMethod, gatingTemplate, character-method, 36
- ppMethod-class, 36
- preprocessing, ppMethod, GatingSet-method, 37
- prior_flowClust, 38
- priors, fcFilter, ANY-method, 37
- priors, fcFilter, character-method
 - (priors, fcFilter, ANY-method), 37

- quadGate.seq, 39
- quadGate.tmix, 39
- quantileGate, 40

- refGate-class, 41
- registerGatingFunction
 - (registerPlugins), 41
- registerPlugins, 41
- renderGraph, 35

- show, boolMethod-method, 42
- show, fcFilter-method, 43
- show, gatingTemplate-method, 43
- show, gtMethod-method, 44

- tailgate, 39, 44
- templateGen, 45