flowWorkspace

March 24, 2012

`GatingHierarchy-class` *Class '"GatingHierarchy"'*

Description

GatingHierarchy is a class for representing the gating hierarchy imported from a flowJo workspace.

Details

There is a one-to-one correspondence between GatingHierarchy objects and FCS files in the flowJo workspace. Each sample (FCS file) is associated with it's own GatingHierarchy. This is different from the workflow representation used in flowCore.

A GatingHierarchy can have two "states". After a call to parseWorkspace(...,execute=FALSE), the workspace is imported but the data is not. A call to execute() is needed in order to load, transform, compensate, and gate the associated data. Alternately, one may call parseWorkspace(...,execute=TRUE). Whether or not a GatingHierarchy has been applied to data is encoded in the `flag` slot. Some methods will warn the user, or may not function correctly if the GatingHierarchy has not been execute()d.This mechanism is in place, largely for the purpose of memory efficiency when working with larger workspaces. It allows the use to load a workspace and subset desired samples before proceeding to load the data. If one has netCDF 4 library installed, then memory is no longer an issue.

Given a GatingHierarchy, one can extract the data associated with any subpopulation, extract gates, plot gates, and extract population proportions. This facilitates the comparison of manual gating methods with automated gating algorithms.

GatingHierarchy objects can be converted to workflows.

Objects from the Class

GatingHierarchy objects are elements of a GatingSet, which is returned by a call to `parseWorkspace()`.

Slots

`tree`: Object of class `"graphNEL"` representing the tree-structured gating hierarchy.

`nodes`: Object of class `"character"`. A vector of node names representing the populations/gates in the tree.

`name`**:** Object of class `"character"`. The name of the sample. Usually the FCS filename, but it depends on how it was defined in the flowJo workspace.

`flag`**:** Object of class `"logical"`. A flag indicating whether the gates, transformations, and compensation matrices have been applied to data, or simply imported.

`transformations`**:** Object of class `"list"`. The list of transformations applied to each dimension of the data.

`compensation`**:** Object of class `"matrix"`. The compensation matrix applied to the data

`dataPath`**:** Object of class `"character"`. A path to the fcs file associated with this GatingHierarchy

`isNcdf`**:** Specifies if the netcdf is used to store the data for this object.

Methods

[[<- `signature(x = "GatingSet", i = "ANY", j = "ANY", value = "GatingHierarchy")`: replacement method for GatingHierarchy objects within a GatingSet

execute `signature(hierarchy = "GatingHierarchy")`: Apply the compensation, transformation, and gating of a GatingHierarchy to its associated fcs file.

flowWorkspace2flowCore `signature(obj = "GatingHierarchy")`: convert a GatingHierarchy to a flowCore workflow.

getBoundaries `signature(obj = "GatingHierarchy", y = "character")`: Get the vertices of gate y in GatingHierarchy obj.

getChildren `signature(obj = "GatingHierarchy", y = "character")`: Get the child nodes of population y in obj.

getData `signature(obj = "GatingHierarchy")`: Return a flowFrame for population y in GatingHierarchy obj.

getDimensions `signature(obj = "GatingHierarchy", y = "character")`: Get the dimensions for the gate of population y

getGate `signature(obj = "GatingHierarchy", y = "character")`: Return the gate for population y.

getGate `signature(obj = "GatingHierarchy", y = "numeric")`: REturn the gate for population y, by index rather than name.

getIndices `signature(obj = "GatingHierarchy", y = "character")`: Return the event membership indices for population y.

getKeywords `signature(obj = "GatingHierarchy", y = "missing")`: Return the keywords for the GatingHierarchy

getNodes `signature(x = "GatingHierarchy")`: Return the node list for the gating hierarchy.

getParent `signature(obj = "GatingHierarchy", y = "character")`: Get the parent node of a population.

getParent `signature(obj = "GatingHierarchy", y = "numeric")`: Get the parent node of a population, by index.

getPopStats `signature(x = "GatingHierarchy")`: Return a table of population statistics (proportions and counts) for a gating hierarchy

getProp `signature(x = "GatingHierarchy", y = "character")`: return the population proportion for population y.

getSample `signature(x = "GatingHierarchy")`: Return the sample name of the gating hierarchy

getTotal `signature(x = "GatingHierarchy", y = "character")`: get the total number of events in population y

keyword `signature(object = "GatingHierarchy", keyword = "character")`: get a specific keyword from the gating hierarchy

plot `signature(x = "GatingHierarchy", y = "missing")`: plot a gating hierarchy graph

plotGate `signature(x = "GatingHierarchy", y = "character")`: plot a manual gate for the population over the parent data.

plotGate `signature(x = "GatingHierarchy", y = "numeric")`: plot a manual gate for a population referenced by index, over the parent data

plotPopCV `signature(x = "GatingHierarchy")`: plot the coefficient of variation for all populations, between flowCore and flowJo counts.

show `signature(object = "GatingHierarchy")`: Summarize a Gating Hierarchy

Author(s)

Greg Finak <gfinak@fhcrc.org>

References

http://www.rglab.org/import-flowjo-workspaces-into-r-bioconductor/

See Also

`parseWorkspace` `GatingSet`

Examples

```
require(flowWorkspaceData)
d<-system.file("extdata",package="flowWorkspaceData")
wsfile<-list.files(d,pattern="A2004Analysis.xml",full=TRUE)
ws <- openWorkspace(wsfile);
G<-try(parseWorkspace(ws,execute=TRUE,path=d,name=1));
getPopStats(G[[1]]);
plotPopCV(G[[1]])
plotGate(G[[1]],getNodes(G[[1]])[4]);
getGate(G[[1]],getNodes(G[[1]])[4]);
getBoundaries(G[[1]],getNodes(G[[1]])[4])
getData(G[[1]],getNodes(G[[1]])[4])
```

`GatingSet-class` *Class '"GatingSet"'*

Description

Class that holds a set of `GatingHierarchy` objects, representing a set of samples and the gating scheme associated with each.

Objects from the Class

Objects can be created by a call to parseWorkspace(). The annotated data frame can be populated with the keywords from each sample.

Description

Objects store a collection of GatingHierarchies and represent a group in a flowJo workspace.

Slots

`set`: Object of class `"list"`. A list of GatingHierarchy objects

`metadata`: Object of class `"AnnotatedDataFrame"`. Stores the metadata associated with this set of FCS samples.

Methods

[`signature(x = "GatingSet", i = "ANY")`: Subset a GatingSet using the familiar bracket notation

[<- `signature(x = "GatingSet", i = "ANY", j = "ANY", value = "GatingSet")`: Replace elements of a GatingSet.

[[`signature(x = "GatingSet", i = "ANY")`: Extract a GatingHierarchy from a GatingSet

[[<- `signature(x = "GatingSet", i = "ANY", j = "ANY", value = "GatingHierarchy")`: Replace a GatingHierarchy in a GatingSet

flowWorkspace2flowCore `signature(obj = "GatingSet")`: Convert a GatingSet to a single workflow (if they have a common set of gates) or list of workflows if the GatingHierarchies differ.

getData `signature(obj = "GatingSet")`: Return a flowSet for the GatingSet

getGate `signature(obj = "GatingSet", y = "numeric")`: Return a flowSet for a subpopulation of each GatingHierarchy, numerically indexed.

getKeywords `signature(obj = "GatingSet", y = "character")`: Get the keywords associated with sample y

getKeywords `signature(obj = "GatingSet", y = "numeric")`: Get the keywords associated with sample y, numerical index.

getSamples `signature(x = "GatingSet")`: Get the sample names of the GatingHierarchies in this GatingSet

keyword `signature(object = "GatingSet", keyword = "character")`: Get the specific keyword for all samples in this GatingSet.

lapply `signature(X = "GatingSet")`: lapply method for GatingSet

length `signature(x = "GatingSet")`: Return the length of the GatingSet, number of GatingHierarhcy objects

plotPopCV `signature(x = "GatingSet")`: plot the population coefficients of variation between flowJo and flowCore for all populations and all samples

show `signature(object = "GatingSet")`: Print information about the GatingSet.

Author(s)

Greg Finak <gfinak@fhcrc.org>

References

http://www.rglab.org/import-flowjo-workspaces-into-r-bioconductor/

See Also

`AnnotatedDataFrame` `GatingHierarchy` `flowJoWorkspace`

Examples

```
require(flowWorkspaceData)
d<-system.file("extdata",package="flowWorkspaceData")
wsfile<-list.files(d,pattern="A2004Analysis.xml",full=TRUE)
ws <- openWorkspace(wsfile);
G<-try(parseWorkspace(ws,execute=TRUE,path=d,name=1));
plotPopCV(G[[1]]);
```

`closeWorkspace` *Close a flowJoWorkspace*

Description

Close a flowJoWorkspace, destroying the internal representation of the XML document, and freeing the associated memory.

Usage

```
## S4 method for signature 'flowJoWorkspace'
closeWorkspace(workspace)
```

Arguments

`workspace`	A `flowJoWorkspace`

Details

* Close a flowJoWorkpsace after finishing with it. This is necessary to explicitly clean up the C-based representation of the XML tree. (See the XML package).

Value

This function doesn't return anything.

Author(s)

Greg Finak `<gfinak@fhcrc.org>`

References

`http://www.rglab.org/`

See Also

`xmlTreeParse`

Examples

```
## Not run:
ws<-openWorkspace("myworkspace.xml");
G<-parseWorkspace(ws,path=".") #path to fcs files, will search recursively
closeWorkspace(ws);

## End(Not run)
```

`ellipsoidGate2FlowJoVertices`

Get the vertices of an ellipsoid gate needed to export it to flowJo

Description

Fetches the top, bottom, right, and leftmost points of an ellipsoid gate (represented by a covariance matrix), for export to flowJo.

Usage

```
## S4 method for signature 'ellipsoidGate'
ellipsoidGate2FlowJoVertices(gate, level = 0.95,...)
```

Arguments

`gate`	`ellipsoidGate` object.
`level`	`numeric`. The quantile of the ellipse to be retrieved. Defaults to 0.95 (95%)
`...`	Additional arguments. Currently not used.

Details

flowJo's XML representation of ellipsoid gates uses the top, bottom,right, and leftmost points of the ellipse. This function retrieves those vertices given an `ellipsoidGate` flowCore object.

Value

A `matrix` with two columns. Rows are the top, bottom, right, and leftmost points on the ellipse (rotated).

Note

The implementation of this function is based on the `ellipse` function in the `ellipse` package.

Author(s)

Greg Finak `<gfinak@fhcrc.org>`

References

`http://www.rglab.org/`

See Also

`ellipse`

Examples

```
require(flowCore)
e<-ellipsoidGate(.gate={d<-diag(2);colnames(d)<-c("A","B");d},mean=c(2,2))
ellipsoidGate2FlowJoVertices(e);
```

`execute` *Apply the GatingHierarchy to data, computing population statistics along the way.*

Description

A GatingHierarchy is associated with an fcs file. Calling `execute` on the GatingHierarchy will load the fcs file, perform compensation and transformation, and calculate the gates as described in the flowJoWorkspace. The method is not meant to be called by the user, but is used internally by flowWorkspace.

Usage

```
## S4 method for signature 'GatingHierarchy'
execute(hierarchy, cleanup=FALSE,keep.indices=TRUE,isNcdf=FALSE,ncfs=NULL,dataEn
```

Arguments

`hierarchy`	A GatingHierarchy object
`cleanup`	`cleanup=TRUE|FALSE` When not using netcdf, this logical flag indicates whether the data should be retained in memory after processing or whether it should be scrapped to save RAM, keeping only population statistics. If you are loading a large data set, you may want to consider using netcdf, or setting this to TRUE. However, you will not be able to visualize the results.
`keep.indices`	`keep.indices=TRUE|FALSE` Logical indicating whether the indices calculated from gating should be stored, or deleted, leaving just their counts.
`isNcdf`	`TRUE|FALSE` a logical flag indicating whether the data (FlowFrame) is saved on the disc in netCDF format.
`ncfs`	`ncdfFlowSet` a ncdfFlowSet object created by parseWorkspace(when isNcdf is set as TRUE) which contains information of fcs metaData and netCDF file that stores the real data .
`dataEnvironment`	`environment` for storing ncdfFlowSet common to all GatingHierarchies in a GatingSet.
`...`	Additional arguments. `path="character"` A file path to the fcs file or files.

Details

This method is not meant to be called by the user. Rather, passing `execute=TRUE` to `parseWorkspace` will execute the gating scheme after the flowJo workspace has been loaded. Cleanup is FALSE by default. This may lead to memory issues when you have lots of data, but it is necessary to visualize the analysis. Netcdf is strongly recommended.

Value

Returns a GatingHierarchy with calculated population statistics and gate indices.

Note

This function is not meant to be called by the user. Gating of samples in a flowJo workspace can be invoked by passing `execute=TRUE` to `parseWorkspace`.

Author(s)

Greg Finak `<gfinak@fhcrc.org>`

References

http://www.rglab.org/

See Also

`parseWorkspace`

Examples

```
## Not run:
ws<-openWorkspace("myworkspace.xml");
G<-parseWorkspace(ws,execute=TRUE,isNcdf=FALSE);

## End(Not run)
```

`exportAsFlowJoXML` *Export a workFlow to FlowJo XML*

Description

Converts a workFlow and associated metadata to a FlowJo 9.2 OSX compatible workspace.

Usage

```
## S4 method for signature 'workFlow'
exportAsFlowJoXML(obj, file=NULL,...)
## S4 method for signature 'list'
exportAsFlowJoXML(obj, file=NULL,...)
## S4 method for signature 'ellipsoidGate'
exportAsFlowJoXML(obj,transforms,...)
## S4 method for signature 'polygonGate'
exportAsFlowJoXML(obj,transforms,...)
```

```
## S4 method for signature 'rectangleGate'
exportAsFlowJoXML(obj,transforms,...)
## S4 method for signature 'intersectFilter'
exportAsFlowJoXML(obj,transforms,gate_view,workflow)
```

Arguments

`obj`	`workFlow` workFlow to be exported.
`list`	`list` list of workFlows to be exported.
`file`	`character` the name of the XML output file.
`transforms`	`function` the transform from raw scale to channel space
`gate_view`	`gate_view` the view from the flowCore workflow referencing the gates to be combined
`workflow`	`workflow` the flowCore workFlow this filter is used in
`...`	Additional arguments used by some S4 methods.

Details

Exports a `flowCore` workFlow object to an XML workspace readable by FlowJo.

Value

If `file` is `NULL` it will return the string for the XML output. Otherwise it returns the file name.

Author(s)

Mose Andre `<mandre@fhcrc.org>`

References

`http://www.rglab.org/`

See Also

`workFlow-class`

Examples

```
## Not run:
#Assume w is a workFlow object
exportAsFlowJoXML(w, "export.xml")

## End(Not run)
```

`flowJoWorkspace-class`

Class '"flowJoWorkspace"'

Description

An R representation of a flowJo workspace.

Objects from the Class

Objects can be created by calls of the form `new("flowJoWorkspace.xml", ...)`.

Slots

`version`**:** Object of class `"character"`. The version of the XML workspace.

`file`**:** Object of class `"character"`. The file name.

`.cache`**:** Object of class `"environment"`. An environment for internal use.

`path`**:** Object of class `"character"`. The path to the file.

`doc`**:** Object of class `"XMLInternalDocument"`. The XML document object.

Methods

closeWorkspace `signature(workspace = "flowJoWorkspace")`: Close the workspace file and delete the C representation of the XML document, freeing memory.

flowWorkspace2flowCore `signature(obj = "flowJoWorkspace")`: Convert a flowJo workspace to a flowCore workflow

getCompensationMatrices `signature(x = "flowJoWorkspace")`: Retrieve the compensation matrices in the flowJo workspace.

getKeywords `signature(obj = "flowJoWorkspace", y = "character")`: Get the keywords for sample y from the flowJo workspace

getSampleGroups `signature(x = "flowJoWorkspace")`: Get the sample groups defined in the flowJo workspace.

getSamples `signature(x = "flowJoWorkspace")`: Get the samples listed in the flowJo workspace.

getTransformations `signature(x = "flowJoWorkspace")`: Get the data transformations listed in the flowJo workspace

parseWorkspace `signature(obj = "flowJoWorkspace")`: Parse a workspace, creating a GatingSet.

show `signature(object = "flowJoWorkspace")`: Print information about a workspace

summary `signature(object = "flowJoWorkspace")`: Summarize

Author(s)

Greg Finak <gfinak@fhcrc.org>

References

http://www.rglab.org/

See Also

`GatingSet` `GatingHierarchy`

Examples

```
require(flowWorkspaceData)
d<-system.file("extdata",package="flowWorkspaceData")
wsfile<-list.files(d,pattern="A2004Analysis.xml",full=TRUE)
ws <- openWorkspace(wsfile);
summary(ws)
getSamples(ws)
```

`flowWorkspace-package` *Import and replicate flowJo workspaces and gating schemes using flowCore.*

Description

Import flowJo workspaces into R. Generate the flowJo gating hierarchy and gates using flowCore functionality. Transform and compensate data in accordance with flowJo settings. Plot gates, gating hierarchies, population statistics, and compare flowJo vs flowCore population summaries.

Details

Package:	flowWorkspace
Type:	Package
Version:	0.5.40
Date:	2011-03-04
License:	Artistic 2.0
LazyLoad:	yes
Depends:	methods, RBGL, graph, XML, flowCore, flowViz, Rgraphviz, Biobase

Author(s)

Greg Finak, Mike Jiang, Mose Andre

Maintainer: Greg Finak `<gfinak@fhcrc.org>`

References

`http://www.rglab.org/`

`flowWorkspace2flowCore` *Convert the GatingHierarchies in a GatingSet to a flowCore workflow.*

Description

Extract the compensation matrices,transformation functions and all the gates from GatingHierarchies in a GatingSet generated by the flowWorkspace package, and convert them to the respective views and actionItems of workFlows defined by flowCore package.

Usage

```
## S4 method for signature 'GatingSet'
flowWorkspace2flowCore(obj, ...)
## S4 method for signature 'GatingHierarchy'
flowWorkspace2flowCore(obj, ...)
## S4 method for signature 'flowJoWorkspace'
flowWorkspace2flowCore(obj, ...)
```

Arguments

`obj`	can be a `flowJoWorkspace`,`GatingSet` or a `GatingHierarchy`
`...`	Additional arugments. `path="character"` a file path to the fcs file or files. `groupId="integer"` a number indicating which group of the data (FlowFrame) should be processed when obj is a flowJoWorkspace. `isCompare` a logical flag indicating whether the gatingHierarchies should be compared and merged when they have the same structure if a flowJoWorkspace or GatingSet is provided as the input,default is TRUE

Details

When the function is applied to a flowJoWorkspace or GatingSet, it compares gating hierarchies and generate one workflow object for multiple samples if they have the same gating hierarchy structure. When `obj` is a flowJoWorkspace it first calls parseWorkspace function to parse the Workspace and generate `GatingSet` object and then convert the GatingSet to workflows.

Value

Returns a workflow if `obj` is a GatingHierarchy. Returns a list of workflows if `obj` is a flowJoWorkspace or a GatingSet.

Author(s)

Mike Jiang `<wjiang2@fhcrc.org>`

References

`http://www.rglab.org/`

See Also

`GatingSet-class` `GatingHierarchy-class` `flowJoWorkspace-class` `parseWorkspace`

Examples

```
##locate workspace xml file and fcs files
dataDir <- system.file("extdata", package = "flowWorkspace")
wsfile<-list.files(dataDir,pattern="xml",full=TRUE)[1]

##open workspace xml file
## Not run:
ws<-openWorkspace(wsfile)

##Convert a flowWorkspace to workFlows
wfs<-flowWorkspace2flowCore(ws,groupId=1,path=dataDir)
plotWf(wfs[[1]])

##parse workspace and convert a GatingSet to workFlows
G <- parseWorkspace(ws,execute=TRUE,name=1,path=dataDir)
wfs<-flowWorkspace2flowCore(G,isCompare=TRUE,path=dataDir)
plotWf(wfs[[1]])

##Convert a GatingHierarchy to workFlow
wf <- flowWorkspace2flowCore(G[[1]],path=dataDir)
plotWf(wf)

## End(Not run)
```

`getBoundaries` *Get the boundaries of a flowJo gate*

Description

Get the boundaries (vertices) of a flowJo gate, on the transformed scale.

Usage

```
## S4 method for signature 'GatingHierarchy,character'
getBoundaries(obj, y)
```

Arguments

`obj`	A `GatingHierarchy`
`y`	A `character`, the name of the node / gate / population of interest whose gate boundaries you wish to return.

Details

Each node in a GatingHierarchy represents a population. That population is defined by a gate. `getBoundaries` will return the vertices of the gate.

Value

A `matrix` with column names corresponding to channels / dimensions, and rows to `x,y` tuples of vertices for polygon gates in these dimensions.

Author(s)

Greg Finak `<gfinak@fhcrc.org>`

References

`http://www.rglab.org`

See Also

`getGate`

Examples

```
## Not run:
file<-"myworkspace.xml"
ws<-openWorkspace(file)
G<-parseWorkspace(ws,execute=TRUE,path=".")
n<-getNodes(G[[1]],tsort=TRUE)[3] #get the third node in the first gating hierarchy (topo
getGate(G[[1]],n); #return the gate for that node.

## End(Not run)
```

`getChildren` *Returns a list of child populations of the current node in the 'GatingHierarchy'*

Description

Returns a character vector of all the children of the current node in the `GatingHierarchy`

Usage

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

Arguments

`obj`	a `GatingHierarchy`
`y`	a `character` name of the node / population.

Details

Get the child nodes / populations of the given node / population, `y` in the GatingHierarchy `obj`

Value

A `character` vector of the names of the child nodes of the current node. An empty vector if the node has no children.

Author(s)

Greg Finak `<gfinak@fhcrc.org>`

References

http://www.rglab.org/

Examples

```
## Not run:
#G is a GatingHierarchy
n<-getNodes(G,tsort=T)[4];
getChildren(G,n);#Get the names of the child nodes of the 4th node in this gating hierarc

## End(Not run)
```

`getCompensationMatrices`

Retrieve the compensation matrices from a flowJo Workspace

Description

Retrieve all the compensation matrices from a flowJo workspace

Usage

```
## S4 method for signature 'flowJoWorkspace'
getCompensationMatrices(x)
```

Arguments

`x`	A `flowJoWorkspace` object.

Details

Return all the compensation matrices in a flowJoWorkspace object.

Value

A list of `matrix` representing the spillover matrices in the `flowJoWorkspace`

Author(s)

Greg Finak `<gfinak@fhcrc.org>`

References

http://www.rglab.org/

See Also

`openWorkspace`

Examples

```
## Not run:
#ws is a flowJoWorkspace
file<-"myworkspace.xml"
ws<-openWorkspace(file)
getCompensationMatrices(ws);

## End(Not run)
```

`getData` *Return the flowFrame associated with a GatingHierarchy*

Description

Return the `flowFrame` associated with a `GatingHierarchy`

Usage

```
## S4 method for signature 'GatingHierarchy'
getData(obj,y=NULL,tsort=FALSE)
## S4 method for signature 'GatingSet'
getData(obj,y=NULL,tsort=FALSE)
## S4 method for signature 'graphNEL'
getData(obj,y=NULL,tsort=FALSE)
```

Arguments

`obj`	A `GatingHierarchy`, `GatingSet`, or `graphNEL` object extracted from a `GatingHierarchy @tree` slot.
`y`	`character` node name or `numeric` node index. Default is `NULL`. If `obj` is a GatingHierarchy or graphNEL, `y` is the name of the node in `obj` for which you wish to extract the data or a numeric index into `getNodes(obj)`. If `obj` is a GatingSet, `y` is a numeric index into `getNodes(obj[[i]])`, where `i` is any GatingHierarchy in the GatingSet. The trees represented by the GatingHierarchies are ASSUMED to be the same. Defaults to `NULL`, will return the complete flowFrame at the root node.
`tsort`	`TRUE|FALSE` logical indicating whether to retrieve the node list in topological sort order. Defaults to FALSE for internal compatibility.

Details

Returns a flowFrame containing the events in the gate defined at node `y`. Subset membership can be obtained using `getIndices`. Population statistics can be obtained using `getPop` and `getPopStats`. When calling `getData` on a GatingSet, `y="numeric"` only. Furthermore, the trees representing the GatingHierarchy for each sample in the GaingSet are presumed to have the same structure, facilitating identical node ordering with tsort=TRUE, and numeric indexing of the nodes.

Value

A `flowFrame` object if `obj` is a GatingHierarchy or graphNEL. A `flowSet` if

Note

The argument `tsort` ensures that the nodes are ordered in topological sort order. This is useful if you are using numeric node indices to access data across two or more identical trees (GatingHierarchies) with different node names (population names).

WARNING The trees in a GatingSet are assumed to have the same structure, such that a topological sort of the nodes in any tree will return the populations in the same order.

Author(s)

Greg Finak `<gfinak@fhcrc.org>`

References

`http://www.rglab.org/`

See Also

`getIndices` `getProp` `getPopStats`

Examples

```
## Not run:
#G is a GatingSet
geData(G,3) #get a flowSet constructed from the third node / population in the tree.

#G is a GatingHierarchy
getData(G,)


## End(Not run)
```

`getDimensions` *Return the dimensions on which a gate is applied within a GatingHierarchy*

Description

Return the dimension names on which a gate in a `GatingHierarchy` is applied.

Usage

```
## S4 method for signature 'GatingHierarchy,character'
getDimensions(obj, y, index=FALSE)
```

Arguments

`obj`	A `GatingHierarchy`
`y`	The name of the population (node) in the `GatingHierarchy` for which you want the dimension names
`index`	`TRUE | FALSE` a logical indicating whether we should return the names of the dimensions (FALSE, default) or the indices of the dimensions (TRUE)

Value

A `character` vector of dimension names on which the gate is applied (when index=FALSE), or a `numeric` vector of the indices of the dimensions on which the gate is applied (when index=TRUE).

Author(s)

Greg Finak `<gfinak@fhcrc.org>`

References

`http://www.rglab.org`

See Also

`getNodes`

Examples

```
## Not run:
#G is a GatingHierarchy
#Fetch the dimensions for the fifth population in the hierarchy.
getDimensions(G,getNodes(G)[5],index=FALSE)

## End(Not run)
```

`getFJWSubsetIndices` *Fetch the indices for a subset of samples in a flowJo workspace, based on a keyword value pair*

Description

Returns an index vector into the samples in a flowJo workspace for use with parseWorkspace(subset=), based on a keyword/value filter in a specific group of samples.

Usage

```
getFJWSubsetIndices(ws, key, value, group)
```

Arguments

`ws`	The flowJoWorkspace object
`key`	The name of the keyword. Type "character"
`value`	The value of the keyword. Type "character"
`group`	The group of samples to subset. Type numeric.

Details

This function will calculate the indices of a subset of samples in a flowJoWorkspace, based on a keyword/value filter. It is applied to a specific group of samples in the workspace. It is meant to be passed to the subset= argument of parseWorkspace.

Value

A numeric vector of indices.

Author(s)

Greg Finak <gfinak@fhcrc.org>

See Also

`parseWorkspace`

`getGate` *Return the flowCore gate definition associated with a node in a GatingHierarchy.*

Description

Return the flowCore gate definition object associated with a node in a `GatingHierarchy` object.

Usage

```
## S4 method for signature 'GatingHierarchy,character'
getGate(obj, y)
## S4 method for signature 'GatingHierarchy,numeric'
getGate(obj, y,tsort=FALSE)
## S4 method for signature 'GatingSet,numeric'
getGate(obj, y,tsort=FALSE)
```

Arguments

`obj`	A `GatingHierrarchy` or `GatingSet`
`y`	A `character` when `obj` is a GatingHierarchy: the name of the node of interest. Or, a `numeric` when `obj` is either a GatingHierarchy or GatingSet. An index into the node list of nodes in the `GatingHierarchy` or `GatingSet`.
`tsort`	`TRUE|FALSE` if true, return the index `y` will access the nodes in topological sort order.

Value

A gate object from `flowCore`. Usually a `polygonGate`, but may be a `rectangleGate`. Boolean gates are represented by a `"BooleanGate"` S3 class. This is a list boolean gate definition that references populations in the GatingHierarchy and how they are to be combined logically. If `obj` is a `GatingSet`, assuming the trees associated with each `GatingHierarchy` are identical, then this method will return a list of gates, one for each sample in the `GatingSet` corresponding to the same population indexed by `y`.

Note

You should not have to deal with boolean gates. It is sufficient to retrieve the contents of a boolean gate node with `getData`.

Author(s)

Greg Finak `<gfinak@fhcrc.org>`

References

`http://www.rglab.org`

See Also

`getData` `getNodes`

Examples

```
## Not run:  #G is a GatingHierarchy
getGate(G,5,tsort=TRUE) #return the gate for the fifth node in the tree.
getGate(G,getNodes(G,tsort=TRUE)[5]) #return the gate for the fifth node in the tree, but
#G is a GatingSet
getGate(G,5,tsort=TRUE) #return a list of gates for the fifth node in each tree, assuming

## End(Not run)
```

`getIndices` *Get the membership indices for each event with respect to a particular gate in a GatingHierarchy*

Description

Returns a logical vector that describes whether each event in a sample is included or excluded by this gate.

Usage

```
## S4 method for signature 'GatingHierarchy,character'
getIndices(obj, y)
```

Arguments

`obj`	A `GatingHierarchy` representing a sample.
`y`	A `character` giving the name of the population / node of interest.

Details

Returns a logical vector that describes whether each event in the data file is included in the given gate of this `GatingHierarchy`. The indices are for all events in the file, and do not reflect the population counts relative to the parent but relative to the root. To get population frequencies relative to the parent one cross-tabulate the indices of `y` with the indices of its parent.

Value

A logical vector of length equal to the number of events in the FCS file that determines whether each event is or is not included in the current gate.

Warning

The indices returned reference all events in the file and are not directly suitable for computing population statistics, unless subsets are taken with respect to the parent populations.

Note

Generally you should not need to use `getIndices` but the more convenient methods `getProp` and `getPopStats` which return population frequencies relative to the parent node.

Author(s)

Greg Finak `<gfinak@fhcrc.org>`

References

`http://www.rglab.org`

See Also

See also `getProp`, `getPopStats`.

Examples

```
## Not run:
#G is a gating hierarchy
#Return the indices for population 5 (topological sort)
getIndices(G,getNodes(G,tsort=TRUE)[5]);

## End(Not run)
```

`getKeywords` *Get List of Keywords for a Flow Sample*

Description

Retrieve the list of keywords associated with a sample

Usage

```
## S4 method for signature 'GatingHierarchy,missing'
getKeywords(obj, y)
## S4 method for signature 'GatingSet,character'
getKeywords(obj, y)
## S4 method for signature 'GatingSet,numeric'
getKeywords(obj, y)
## S4 method for signature 'flowJoWorkspace,character'
getKeywords(obj, y)
```

Arguments

`obj`	A `flowJoWorkspace`, `GatingSet`, or `GatingHierarchy`
`y`	can be omitted if `obj` is a `GatingHierarchy`. A `character`, or `numeric` if `obj` is a `GatingSet`. A `character` if `obj` is a `flowJoWorkspace`

Details

Retrieve a list of keywords from a `flowJoWorkspace`, `GatingSet`, or `GatingHierarchy` for a particular sample. The sample is specified via `y`, either a numeric index into a `GatingSet`, or a sample name (`character`) for all other types of `obj`.

Value

A list of keyword - value pairs.

Author(s)

Greg Finak `<gfinak@fhcrc.org>`

References

`http://www.rglab.org/`

Examples

```
## Not run:
#G is a GatingHierarchy
getKeywords(G);
#G is a GatingSet
getKeywords(G[[1]])
getKeywords(G,1)

## End(Not run)
```

`getNodes` *Get the names of all nodes in a gating hierarchy*

Description

Returns a character vector of names of the nodes (populations) in the `GatingHierarchy`.

Usage

```
## S4 method for signature 'GatingHierarchy'
getNodes(x,tsort=FALSE,...)
```

Arguments

`x`	A `GatingHierarchy`
`tsort`	`tsort=TRUE|FALSE` returns the nodes in topological sort (TRUE) order.
`...`	Additional arguments.

Value

a `character` vector of node/population names, ordered appropriately.

Author(s)

Greg Finak <`gfinak@fhcrc.org`>

References

`http://www.rglab.org/`

See Also

`tsort`

Examples

```
## Not run:
#G is a gating hierarchy
getNodes(G,tsort=TRUE)#return nodes in topological sort order.

## End(Not run)
```

`getParent` *Return the name of the parent population of the current population in the GatingHierarchy*

Description

Returns the name of the parent population of the current population in the given `GatingHierarchy`

Usage

```
## S4 method for signature 'GatingHierarchy,character'
getParent(obj, y)
## S4 method for signature 'GatingHierarchy,numeric'
getParent(obj, y, tsort=FALSE)
```

Arguments

`obj`	A `GatingHierarchy`
`y`	The population whose parent you want to retrieve, either `character` or `numeric`
`tsort`	`logical` If TRUE, nodes are ordered by topological sort. Important when comparing across identical trees and using numeric indices. Default FALSE.

Value

Returns a `character` vector, the name of the parent population.

Author(s)

Greg Finak <`gfinak@fhcrc.org`>

References

`http://www.rglab.org/`

See Also

`getChildren`

Examples

```
## Not run:
#G is a gatinghierarchy
#return the name of the parent of the fifth node in the hierarchy.
getParent(G,getNodes(G,tsort=TRUE)[5])

## End(Not run)
```

`getPopStats` *Return a table of population statistics for all populations in a GatingHierarchy*

Description

More useful than getPop. Returns a table of population statistics for all populations in a `GatingHierarchy`. Includes the flowJo counts, flowCore counts and frequencies.

Usage

```
## S4 method for signature 'GatingHierarchy'
getPopStats(x,...)
```

Arguments

`x`	A `GatingHierarchy`
`...`	Additional arguments

Details

Returns a table population statistics for all populations in the gating hierarchy. The output is useful for verifying that the import was successful, if the flowJo and flowCore derived counts don't differ much (i.e. if they have a small coefficient of variation.)

Value

A `data.frame` with columns for the population name, flowJo derived counts, flowCore derived counts, and the population proportions (relative to their parent pouplation).

Author(s)

Greg Finak `<gfinak@fhcrc.org>`

References

`http://www.rglab.org/`

See Also

`getProp`

Examples

```
## Not run:
#If G is a GatingHierarchy
getPopStats(G);

## End(Not run)
```

`getProp` *Get the population proportions of a node (population) in a GatingHierarchy*

Description

Calculate the population proportion (events in the gate / events in the parent population) associated with a node in the `GatingHierarchy`.

Usage

```
## S4 method for signature 'GatingHierarchy,character'
getProp(x, y)
```

Arguments

`x`	A `GatingHierarchy` object.
`y`	`character` The name of the node. A list of nodes is accessible via `getNodes(x)`.

Details

Returns the proportion of cells in the gate, relative to its parent.

Value

Returns a population frequency `numeric`.

Author(s)

Greg Finak `<gfinak@fhcrc.org>`

References

`http://www.rglab.org/`

See Also

`getPopStats`

Examples

```
## Not run:
#G is a GatingHierarchy
#proportion for the fifth population
getProp(G,getNodes(G)[5])

## End(Not run)
```

`getSample` *Get the sample name associated with a GatingHierarchy*

Description

Return the sample name

Usage

```
## S4 method for signature 'GatingHierarchy'
getSample(x,isFullPath=FALSE)
## S4 method for signature 'graphNEL'
getSample(x)
```

Arguments

`x`	A `GatingHierarchy` or a `graphNEL` object from the `@tree` slot of a GatingHierarchy
`isFullPath`	`isFullPath` is a logical value indicating whether the full path of the sample FCS file is returned.Default is FALSE.

Details

Returns the name of the sample, or the path to the FCS file.

Value

A "character" vector of length 1. Either the sample name or the path to the FCS file.

Author(s)

Mike Jiang `<wjiang2@fhcrc.org>`

References

`http://www.rglab.org/`

Examples

```
## Not run:
#G is  a GatingHierarhcy
getSample(G)
getSample(G@tree);

## End(Not run)
```

`getSampleGroups` *Get a table of sample groups from a flowJo workspace*

Description

Return a data frame of sample group information from a flowJo workspace

Usage

```
## S4 method for signature 'flowJoWorkspace'
getSampleGroups(x)
```

Arguments

`x`	A `flowJoWorkspace` object.

Details

Returns a table of samples and groups defined in the flowJo workspace

Value

A `data.frame` containing the `groupName`, `groupID`, and `sampleID` for each sample in the workspace. Each sample may be associated with multiple groups.

Author(s)

Greg Finak `<gfinak@fhcrc.org>`

References

`http://www.rglab.org/`

See Also

`flowJoWorkspace-class` `openWorkspace`

Examples

```
## Not run:
#ws is a flowJoWorkspace
getSampleGroups(ws);

## End(Not run)
```

`getSamples` *Get a list of samples in a flowJo workspace or a GatingSet*

Description

Return a data frame of samples contained in a flowJo workspace or a GatingSet

Usage

```
## S4 method for signature 'GatingSet'
getSamples(x,isFullPath=FALSE)
## S4 method for signature 'flowJoWorkspace'
getSamples(x)
```

Arguments

`x`	A `flowJoWorkspace` or a `GatingSet`
`isFullPath`	`isFullPath` is a logical value indicating whether the full path of the sample file is returned.Default is FALSE.

Details

Returns a `data.frame` of samples in the `flowJoWorkspace`, including their `sampleID`, `name`, and `compID` (compensation matrix ID). If `x` is a `GatingSet`, returns a character vector of sample names.

Value

A `data.frame` with columns `sampleID`, `name`, and `compID` if `x` is a `flowJoWorkspace`. A character vector of sample names if `x` is a `GatingSet`.

Author(s)

Greg Finak `<gfinak@fhcrc.org>`

References

`http://www.rglab.org/`

Examples

```
## Not run:
#G is  a GatingSet
getSamples(G)
#f is a flowJoWorkspace
getSamples(f);

## End(Not run)
```

`getTotal` *Fetch the total number of events in a gate defined in a GatingHierarchy*

Description

Returns the total number of events in the gate defined in the GatingHierarchy object

Usage

```
## S4 method for signature 'GatingHierarchy,character'
getTotal(x, y)
```

Arguments

`x`	The `GatingHierarchy`
`y`	A `character` name of the gate / population

Details

Will return the total number of events included in this gate. The contents of "thisTot" variable in the "metadata" environment of the `nodeData` element associated with the gating tree and gate / population.

Value

A `numeric` value of the total number of elements in the population.

Author(s)

Greg Finak `<gfinak@fhcrc.org>`

References

`http://www.rglab.org/`

See Also

`getProp` `getPopStats`

Examples

```
## Not run:
#G is a gating hierarchy
#Fifth node total.
getTotal(G,getNodes(G,tsort=T)[5])

## End(Not run)
```

`getTransformations` *Return a list of transformations in a flowJo workspace*

Description

Return a list of all the transformations in a flowJo workspace

Usage

```
## S4 method for signature 'flowJoWorkspace'
getTransformations(x)
```

Arguments

`x`	A `flowJoWorkspace` object

Details

Returns a list of the transformations in the flowJo workspace. The list is of length `L`, where `L` is the number of distinct transformations applied to samples in the `flowJoWorkspace`. Each element of `L` is itself a `list` of length `M`, where `M` is the number of parameters that were transformed for a sample or group of samples in a `flowJoWorkspace`. For example, if a sample has 10 parameters, and 5 are transformed during analysis, using two different sets of transformations, then L will be of length 2, and each element of L will be of length 5. The elements of `L` represent channel- or parameter-specific transformation functions that map from raw intensity values to channel-space used by flowJo.

Value

`comp 1`	The first transformaton in the workspace.
`comp 2`	The second transformation in the workspace.

Comp 1 .. Comp `L` are themselves lists of functions, with each element of the list representing a transformation applied to a specific channel/parameter of a sample.

Note

This representation will likely be changed in the future to use the `flowCore` internal `transformation` classes.

Author(s)

Greg Finak `<gfinak@fhcrc.org>`

References

`http://www.rglab.org/`

See Also

`linknames`

Examples

```
## Not run:
#Assume f is a flowJoWorkspace
getTransformations(f);

## End(Not run)
```

`keyword-methods` *Methods to retrieve keywords associated with an FCS sample*

Description

See Methods for function `keyword` in Package ‘flowCore’

Methods

Retrieve a specific keyword for a specific sample in a `GatingHierarchy` or or set of samples in a `GatingSet`

Retrieve a keyword for the `GatingHierarchy`.

`signature(object = "GatingHierarchy", keyword = "character")` `signature(object="Gatir`

Retrieve a specific keyword for each sample in the `GatingSet`

See Also

`keyword-methods`

`lapply-methods` *Methods for iterating over a gating set*

Description

~~ Methods for function `lapply` ~~

Methods

`signature(X = "ANY")`

`signature(X = "GatingSet")`

`length-methods` *Methods to get the length of a GatingSet*

Description

Return the length of a `GatingSet` object (number of samples).

Methods

`signature(x = "GatingSet")`

`openWorkspace` *Open a flowJo workspace*

Description

Open a flowJo workspace and return a `flowJoWorkspace` object.

Usage

```
## S4 method for signature 'character'
openWorkspace(file)
```

Arguments

`file`	Full path to the XML flowJo workspace file.

Details

Open an XML flowJo workspace file and return a `flowJoWorkspace` object. The workspace is represented using a `XMLInternalDocument` object.

Value

Returns a `flowJoWorkspace` object.

Author(s)

Greg Finak `<gfinak@fhcrc.org>`

References

`http://www.rglab.org/`

See Also

See Also as `xmlTreeParse`

Examples

```
## Not run:
file<-"myworkspace.xml"
ws<-openWorkspace(file);
class(ws); #flowJoWorkspace

## End(Not run)
```

`parseWorkspace` *Parse a flowJo Workspace*

Description

Function to parse a flowJo Workspace, generate a `GatingHierarchy` or `GatingSet` object, and associated flowCore gates. The data are not loaded or acted upon until an explicit call to `execute()` is made on the `GatingHierarchy` objects in the `GatingSet`.

Usage

```
## S4 method for signature 'flowJoWorkspace'
parseWorkspace(obj,name=NULL,execute=FALSE,isNcdf=FALSE,subset=NULL,nslaves=4,..
```

Arguments

`obj`	A `flowJoWorkspace` to be parsed.
`name`	`numeric` or `character`. The name or index of the group of samples to be imported. If `NULL`, the groups are printed to the screen and one can be selected interactively. Usually, multiple groups are defined in the flowJo workspace file.
`execute`	`TRUE|FALSE` a logical specifying if the gates, transformations, and compensation should be immediately calculated after the flowJo workspace have been imported. You should probably set this to TRUE.
`isNcdf`	`TRUE|FALSE` logical specifying if you would like to use netcdf to store the data, or if you would like to keep all the flowFrames in memory. For a small data set, you can safely set this to FALSE, but for larger data, we suggest using netcdf. You will need the netcdf C library installed.
`subset`	`numeric` vector specifying the subset of samples in a group to import.
`nslaves`	`numeric` number of slave processes for executing the gating under Rmpi
`...`	Additional arguments. `path="character"` The path to the fcs files that are to be imported. The code will search recursively, so you can point it to a location above the files. This argument is mandatory.

Details

A flowJoWorkspace is generated with a call to `openWorkspace()`, passing the name of the xml workspace file. This returns a `flowJoWorkspace`, which can be parsed using the `parseWorkspace()` method. The function can be called non-interactively by passing the index or name of the group of samples to be imported via `parseWorkspace(obj,name=x)`, where `x` is either the numeric index, or the name.

Value

Returns a `GatingSet`, which is a wrapper around a list of `GatingHierarchy` objects, each representing a single sample in the workspace. The `GatingHierarchy` objects contain `graphNEL` trees that represent the gating hierarchy of each sample. Each node in the `GatingHierarchy` has associated data, including the population counts from flowJo, the parent population counts, the `flowCore` gates generated from the flowJo workspace gate definitions. Data are not yet loaded or acted upon at this stage. To execute the gating of each data file, a call to `execute()` must be made on each `GatingHierarchy` object in the `GatingSet`. These can be iterated over using `lapply()`. Options to `execute()` allow the user to specify whether compensated and transformed data should be kept in memory for each sample, whether indices specifying the population membership of each event at each node in the `GatingHierarchy` should be kept in memory, or whether these should be deleted after each node is processed and only summary statistics kept associated with each population. The latter option forgoes the ability to plot the data, and is more useful in a high–throughput, large scale setting where keeping the data around in memory can be inefficient.

Author(s)

Greg Finak `<gfinak@fhcrc.org>`

References

`http://www.rglab.org/`

See Also

`getSampleGroups`

Examples

```
## Not run:
#f is a xml file name of a flowJo workspace
ws<-openWorkspace(f)
G<-parseWorkspace(ws,execute=TRUE,isNcdf=FALSE,path="."); #assume that the fcs files are
#G is a GatingSet.

## End(Not run)
```

`plot` *Plot a GatingHierarchy*

Description

Plot a tree representing the GatingHierarchy

Usage

```
## S4 method for signature 'GatingHierarchy,missing'
plot(x,y,layout="dot",width=3,height=2,fontsize=14,labelfontsize=14,fixedsize=FA
```

Arguments

`x`	The `GatingHierarchy to be plotted`
`y`	missing.
`layout`	See `layoutGraph` in package Rgraphviz
`width`	See `layoutGraph` in package Rgraphviz
`height`	See `layoutGraph` in package Rgraphviz
`fontsize`	See `layoutGraph` in package Rgraphviz

`labelfontsize`	See `layoutGraph` in package Rgraphviz
`fixedsize`	See `layoutGraph` in package Rgraphviz
`boolean`	`TRUE|FALSE` logical specifying whether to plot boolean gate nodes. Defaults to FALSE.
`...`	Additional arguments passed to plot in Rgraphviz

Details

Plot a GatingHierarchy object using the Rgraphviz plot function.

Value

Nothing to return

Author(s)

Greg Finak `<gfinak@fhcrc.org>`

References

`http://www.rglab.org/`

See Also

`layoutGraph`

Examples

```
## Not run:
#G is a GatingHierarchy
plot(G);

## End(Not run)
```

`plotGate` *Plot a flowJo Gate and Cell Population*

Description

Plots a flowJo gate and associated cell population using it's `flowCore` definition contained in a `GatingHierarchy`

Usage

```
## S4 method for signature 'GatingHierarchy,character'
plotGate(x, y, add=FALSE,border="red",tsort=FALSE,smooth=FALSE,...)
```

Arguments

`x`	A `GatingHierarchy`
`y`	A `character` or `numeric` representing the node in the `GatingHierarchy`. Nodes can be accessed with `getNodes(GatingHierarchy)`.
`add`	`TRUE|FALSE` logical specifying whether to add the gate to the current plot.
`border`	`character`, The color to plot the border of the gate. Default is "red".
`tsort`	`TRUE|FALSE` logical indicating if nodes should be referenced in topological sort order when `y="numeric"`;
`smooth`	`TRUE|FALSE` logical indicating whether a smoothed 2D scatter plot should be generated
`...`	Additional arguments to the `plot` function.

Details

The function will plot the gate if the gating hierarchy represented by `x` has been `execute()`'d. That is to say, the associated data has been loaded, compensated, transformed, and had the gates applied to it. If the data has not been gated, `plotGate` will print a message, and return without plotting anything.

Value

If the data has been gated, the function will plot the gate. If it has not been gated, the function will print a message and return nothing.

Author(s)

Greg Finak `<gfinak@fhcrc.org>`

References

`http://www.rglab.org/`

Examples

```
## Not run:
#G is a GatingHierarchy
plotGate(G,getNodes(G)[5]);#plot the gate for the  fifth node

## End(Not run)
```

`plotPopCV` *Plot the coefficient of variation between flowJo and flowCore population statistics for each population in a gating hierarchy.*

Description

This function plots the coefficient of variation calculated between the flowJo population statistics and the flowCore population statistics for each population in a gating hierarchy extracted from a flowJoWorkspace.

Usage

```
## S4 method for signature 'GatingHierarchy'
plotPopCV(x,m=2,n=2,...)
## S4 method for signature 'GatingSet'
plotPopCV(x,...)
```

Arguments

`x`	A `GatingHierarchy` from a `flowJoWorkspace`, or a `GatingSet`.
`m`	`numeric` The number of rows in the panel plot. Now deprecated, uses lattice.
`n`	`numeric` The number of columns in the panel plot. Now deprecated, uses lattice.
`...`	Additional arguments to the `barplot` methods.

Details

The CVs are plotted as barplots across panels on a grid of size `m` by `n`.

Value

Nothing is returned.

Author(s)

Greg Finak `<gfinak@fhcrc.org>`

References

`http://www.rglab.org`

See Also

`getPopStats`

Examples

```
## Not run:
#G is a GatingHierarchy
plotPopCV(G,4,4);

## End(Not run)
```

`plotWf` *plot a workflow*

Description

Plots a workflow that contains one `GatingHierarchy`

Usage

```
plotWf(x,...)
```

Arguments

`x`	A `workflow`
`...`	Additional arguments to the `plot` function.

Author(s)

Mike Jiang

Examples

```
## Not run: dataDir <- system.file("extdata", package = "flowWorkspace")
wsfile<-list.files(dataDir,pattern="xml",full=TRUE)[1]

##open workspace xml file
ws<-openWorkspace(wsfile)

##Convert a flowWorkspace to workFlows
wfs<-flowWorkspace2flowCore(ws,groupId=1,path=dataDir)
plotWf(wfs[[1]])
## End(Not run)
```

`[<-,GatingSet,ANY,ANY,GatingSet-method` *Replace parts of a GatingSet*

Description

Replace part of a GatingSet

Arguments

`x`	A `GatingSet` object
`i`	The index to be replaced
`...`	Additional arguments
`value`	A `GatingSet` object of size `length(i)`.

`[[<-,GatingSet,ANY,ANY,GatingHierarchy-method` *Replace parts of a GatingSet*

Description

Replace parts of a GatingSet

Arguments

`x`	A `GatingSet` object
`i`	The index to be replaced
`...`	Additional arguments. Not used.
`value`	A `GatingHierarchy` object

`summary-methods` *Summarize a flowJoWorkspace object*

Description

Summarize a `flowJoWorkspace` object.

Index

∗Topic **IO**
  closeWorkspace, 5
∗Topic **\textasciitilde\textasciitilde other possible keyword(s) \textasciitilde\textasciitilde**
  lapply-methods, 31
  length-methods, 31
∗Topic **classes**
  flowJoWorkspace-class, 10
  GatingHierarchy-class, 1
  GatingSet-class, 3
∗Topic **hplot**
  plot, 34
  plotGate, 35
  plotPopCV, 36
∗Topic **manip**
  ellipsoidGate2FlowJoVertices, 6
  execute, 7
  exportAsFlowJoXML, 8
  flowWorkspace2flowCore, 12
  getBoundaries, 13
  getChildren, 14
  getCompensationMatrices, 15
  getData, 16
  getDimensions, 17
  getGate, 19
  getIndices, 20
  getNodes, 22
  getParent, 23
  getPopStats, 24
  getProp, 25
  getSample, 26
  getSampleGroups, 27
  getSamples, 28
  getTotal, 29
  getTransformations, 30
  openWorkspace, 32
  parseWorkspace, 33
∗Topic **methods**
  keyword-methods, 31
  lapply-methods, 31
  length-methods, 31
  summary-methods, 39
∗Topic **package**
  flowWorkspace-package, 11
∗Topic **univar**
  plotPopCV, 36
∗Topic **utilities**
  flowWorkspace-package, 11
[,GatingSet,ANY-method *(GatingSet-class)*, 3
[<-,GatingSet,ANY,ANY,GatingSet-method, 38
[[,GatingSet,ANY-method *(GatingSet-class)*, 3
[[<-,GatingSet,ANY,ANY,GatingHierarchy-method, 39

AnnotatedDataFrame, *5*

closeWorkspace, 5
closeWorkspace,flowJoWorkspace-method *(closeWorkspace)*, 5
closeWorkspace-methods *(closeWorkspace)*, 5

ellipse, *7*
ellipsoidGate2FlowJoVertices, 6
ellipsoidGate2FlowJoVertices,ellipsoidGate-met *(ellipsoidGate2FlowJoVertices)*, 6
ellipsoidGate2FlowJoVertices-methods *(ellipsoidGate2FlowJoVertices)*, 6
execute, 7
execute,GatingHierarchy-method *(execute)*, 7
execute-methods *(execute)*, 7
exportAsFlowJoXML, 8
exportAsFlowJoXML,ellipsoidGate-method *(exportAsFlowJoXML)*, 8
exportAsFlowJoXML,intersectFilter-method *(exportAsFlowJoXML)*, 8
exportAsFlowJoXML,list-method *(exportAsFlowJoXML)*, 8

exportAsFlowJoXML,polygonGate-method *(exportAsFlowJoXML)*, 8
exportAsFlowJoXML,rectangleGate-method *(exportAsFlowJoXML)*, 8
exportAsFlowJoXML,workFlow-method *(exportAsFlowJoXML)*, 8
exportAsFlowJoXML-methods *(exportAsFlowJoXML)*, 8

flowJoWorkspace, *5*
flowJoWorkspace-class, *12*, *27*
flowJoWorkspace-class, 10
flowWorkspace *(flowWorkspace-package)*, 11
flowWorkspace-package, 11
flowWorkspace2flowCore, 12
flowWorkspace2flowCore,flowJoWorkspace-method *(flowWorkspace2flowCore)*, 12
flowWorkspace2flowCore,GatingHierarchy-method *(flowWorkspace2flowCore)*, 12
flowWorkspace2flowCore,GatingSet-method *(flowWorkspace2flowCore)*, 12
flowWorkspace2flowCore-methods *(flowWorkspace2flowCore)*, 12

GatingHierarchy, *5*, *11*
GatingHierarchy-class, *12*
GatingHierarchy-class, 1
GatingSet, *3*, *11*
GatingSet-class, *12*
GatingSet-class, 3
getBoundaries, 13
getBoundaries,GatingHierarchy,character-method *(getBoundaries)*, 13
getBoundaries-methods *(getBoundaries)*, 13
getChildren, 14, *24*
getChildren,GatingHierarchy,character-method *(getChildren)*, 14
getChildren-methods *(getChildren)*, 14
getCompensationMatrices, 15
getCompensationMatrices,flowJoWorkspace-method *(getCompensationMatrices)*, 15
getCompensationMatrices-methods *(getCompensationMatrices)*, 15
getData, 16, *20*
getData,GatingHierarchy-method *(getData)*, 16
getData,GatingSet-method *(getData)*, 16
getData,graphNEL-method *(getData)*, 16
getData-methods *(getData)*, 16
getDimensions, 17
getDimensions,GatingHierarchy,character-metho *(getDimensions)*, 17
getDimensions-methods *(getDimensions)*, 17
getFJWSubsetIndices, 18
getGate, *14*, 19
getGate,GatingHierarchy,character-method *(getGate)*, 19
getGate,GatingHierarchy,numeric-method *(getGate)*, 19
getGate,GatingSet,numeric-method *(getGate)*, 19
getIndices, *17*, 20
getIndices,GatingHierarchy,character-method *(getIndices)*, 20
getIndices-methods *(getIndices)*, 20
getKeywords, 21
getKeywords,flowJoWorkspace,character-method *(getKeywords)*, 21
getKeywords,GatingHierarchy,missing-method *(getKeywords)*, 21
getKeywords,GatingSet,character-method *(getKeywords)*, 21
getKeywords,GatingSet,numeric-method *(getKeywords)*, 21
getKeywords-methods *(getKeywords)*, 21
getNodes, *18*, *20*, 22
getNodes,GatingHierarchy-method *(getNodes)*, 22
getNodes-methods *(getNodes)*, 22
getParent, 23
getParent,GatingHierarchy,character-method *(getParent)*, 23
getParent,GatingHierarchy,numeric-method *(getParent)*, 23
getParent-methods *(getParent)*, 23
getPopStats, *17*, *21*, 24, *25*, *29*, *37*
getPopStats,GatingHierarchy-method *(getPopStats)*, 24
getPopStats-methods *(getPopStats)*, 24
getProp, *17*, *21*, *24*, 25, *29*

getProp,GatingHierarchy,character-method *(getProp)*, 25
getProp-methods *(getProp)*, 25
getSample, 26
getSample,GatingHierarchy-method *(getSample)*, 26
getSample,graphNEL-method *(getSample)*, 26
getSample-method *(getSample)*, 26
getSampleGroups, 27, *34*
getSampleGroups,flowJoWorkspace-method *(getSampleGroups)*, 27
getSampleGroups-methods *(getSampleGroups)*, 27
getSamples, 28
getSamples,flowJoWorkspace-method *(getSamples)*, 28
getSamples,GatingSet-method *(getSamples)*, 28
getSamples-methods *(getSamples)*, 28
getTotal, 29
getTotal,GatingHierarchy,character-method *(getTotal)*, 29
getTotal-methods *(getTotal)*, 29
getTransformations, 30
getTransformations,flowJoWorkspace-method *(getTransformations)*, 30
getTransformations-methods *(getTransformations)*, 30

keyword *(keyword-methods)*, 31
keyword,GatingHierarchy,character *(keyword-methods)*, 31
keyword,GatingHierarchy,character-method *(keyword-methods)*, 31
keyword,GatingSet,character *(keyword-methods)*, 31
keyword,GatingSet,character-method *(keyword-methods)*, 31
keyword-methods, *31*
keyword-methods, 31

lapply,ANY-method *(lapply-methods)*, 31
lapply,GatingSet-method *(lapply-methods)*, 31
lapply-methods, 31
layoutGraph, *34, 35*
length,GatingSet-method *(length-methods)*, 31
length-methods, 31

openWorkspace, *15*, *27*, 32
openWorkspace,character-method *(openWorkspace)*, 32
openWorkspace-methods *(openWorkspace)*, 32

parseWorkspace, *3*, *8*, *12*, *19*, 33
parseWorkspace,flowJoWorkspace-method *(parseWorkspace)*, 33
parseWorkspace-methods *(parseWorkspace)*, 33
plot, 34
plot,GatingHierarchy,missing-method *(plot)*, 34
plot-methods *(plot)*, 34
plotGate, 35
plotGate,GatingHierarchy,character-method *(plotGate)*, 35
plotGate,GatingHierarchy,numeric-method *(plotGate)*, 35
plotGate-methods *(plotGate)*, 35
plotPopCV, 36
plotPopCV,GatingHierarchy-method *(plotPopCV)*, 36
plotPopCV,GatingSet-method *(plotPopCV)*, 36
plotPopCV-methods *(plotPopCV)*, 36
plotWf, 38

show,flowJoWorkspace-method *(summary-methods)*, 39
show,GatingHierarchy-method *(summary-methods)*, 39
show,GatingSet-method *(summary-methods)*, 39
show-methods *(summary-methods)*, 39
summary,ANY-method *(summary-methods)*, 39
summary,flowJoWorkspace-method *(summary-methods)*, 39
summary-methods, 39

tsort, *23*

workFlow-class, *9*

xmlTreeParse, *5*, *32*