

# Package ‘flowWorkspace’

October 9, 2013

**Type** Package

**Title** Import flowJo Workspaces into BioConductor and replicate flowJo gating with flowCore

**Version** 1.6.1

**Date** 2011-06-10

**Author** Greg Finak, Mike Jiang, Mose Andre

**Maintainer** Greg Finak <gfinak@fhcrc.org>,Mike Jiang <wjiang2@fhcrc.org>

**Description** This package is designed to facilitate comparison of automated gating methods against manual gating done in flowJo. This package allows you to import basic flowJo workspaces into BioConductor and replicate the gating from flowJo using the flowCore functionality. Gating hierarchies, groups of samples, compensation, and transformation are performed so that the output matches the flowJo analysis.

**License** Artistic-2.0

**LazyLoad** yes

**Imports** Biobase, XML,flowCore, graph, graphics, lattice, methods,stats, stats4, utils

**Collate** AllClasses.R AllGenerics.R AllMethods.R InternalClasses.R  
GatingHierarchyInternal-Methods.R GatingSetInternal-Methods.R  
GatingSetList-Methods.R AllExportMethods.R  
flowWorkspace2flowCore-methods.R flowWorkspaceEdit-methods.R bitVector.R add-methods.R

**Depends** R (>= 2.16.0),Rcpp (>= 0.9.9),Cairo, BiocGenerics, methods,RBGL, graph, XML, flowCore (>= 1.25.9), flowViz (>= 1.23.1),Biobase, IDP-misc, tools,hexbin,gridExtra,Rgraphviz,ncdfFlow(>= 1.5.19)

**biocViews** FlowCytometry, DataImport, Preprocessing, DataRepresentation

**Enhances** multicore,Rmpi

**Suggests** testthat, flowWorkspaceData

**LinkingTo** Rcpp,ncdfFlow

**R topics documented:**

flowWorkspace-package	3
add	3
archive	6
booleanFilter-class	7
clone	8
ellipsoidGate2FlowJoVertices	9
execute	10
exportAsFlowJoXML	12
ExportTSVAnalysis	13
flowJoWorkspace-class	14
flowWorkspace2flowCore	15
GatingHierarchy-class	17
GatingSet-class	19
GatingSetList-class	21
getBoundaries	22
getCompensationMatrices	24
getData	25
getFJWSummaryIndices	26
getGate	27
getIndexFile	29
getIndices	29
getKeywords	31
getNodes	32
getParent	33
getPopStats	34
getSample	36
getSampleGroups	37
getSamples	38
getTransformations	39
haveSameGatingHierarchy	40
includedChannel2ExcludedChannel	41
includedGate2ExcludedGate	42
keyword-methods	42
lapply-methods	43
length-methods	43
openWorkspace	44
parseWorkspace	45
plot	47
plotGate	48
plotPopCV	50
plotWf	51
recomputeGatingSet	52
saveNcdf	52
summary-methods	53

---

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/>

---

add *Create a GatingSet and add/remove the flowCore gate(or population) to/from a GatingHierarchy/GatingSet.*

---

### Description

GatingSet method creates a gatingset from a flowSet with the ungated data as the root node. add method add the flowCore gate to a GatingHierarchy/GatingSet. Rm method Remove the population node from a GatingHierarchy/GatingSet. They are equivalent to the workFlow,add and Rm methods in flowCore package. recompute method does the actual gating after the gate is added,i.e. calculating the event indices according to the gate definition.

**Usage**

```
## S4 method for signature 'GatingHierarchyInternal,rectangleGate'
add(wf, action,...)
## S4 method for signature 'GatingHierarchyInternal,polygonGate'
add(wf, action,...)
## S4 method for signature 'GatingHierarchyInternal,booleanFilter'
add(wf, action,...)
## S4 method for signature 'GatingHierarchyInternal,quadGate'
add(wf, action, names = NULL,...)
## S4 method for signature 'GatingSetInternal,filter'
add(wf, action,...)
## S4 method for signature 'GatingSetInternal,filterList'
add(wf, action,...)
## S4 method for signature 'GatingSet,list'
add(wf, action,...)

## S4 method for signature 'character,GatingHierarchyInternal,character'
Rm(symbol,envir,subSymbol,...)
## S4 method for signature 'character,GatingSetInternal,character'
Rm(symbol,envir,subSymbol,...)

## S4 method for signature 'GatingSetInternal'
recompute(x,y)
```

**Arguments**

wf	A GatingHierrarchy or GatingSet,
envir	A GatingHierrarchy or GatingSet,
x	A GatingSet,
y	a node index within a GatingSet. It is the root node if unspecified.
action	A filter or a list of filters to be added to the GatingHierarchy or GatingSet.
names	a character vector of length four,which specifies the population names resulted by adding a quadGate. The order of the names is clock-wise starting from the top left quadrant population.
...	some other arguments to specify how the gates are added to the gating tree. <ul style="list-style-type: none"> <li>• parent: a character scalar to specify the parent node name where the new gate to be added to, by default it is NULL,which indicates the root node</li> <li>• name: a character scalar to specify the node name of population that is generated by the gate to be added. The default value is NULL,then the name will be extracted from the filterId of the gate to be added</li> <li>• negated: a logical scalar to specify whether the gate is negated,which means the the population outside of the gate will be kept as the result population. It is FALSE by default.</li> </ul>

symbol	A character identifies the population node in a GatingHierarchy or GatingSet to remove
subSymbol	Not used.

### Value

GatingSet method returns a GatingSet object with just root node. add method returns a population node ID (or four population node IDs when adding a quadGate) that uniquely identify the population node within a GatingHierarchy.

### Author(s)

Greg Finak <gfinak@fhcrc.org> Mike Jiang <wjiang2@fhcrc.org>

### References

<http://www.rglab.org>

### See Also

[GatingSet-class](#)

### Examples

```
## Not run:
data(GvHD)
#select raw flow data
fs<-GvHD[1:3]

#transform the raw data
tf <- transformList(colnames(fs[[1]])[3:6], asinh, transformationId="asinh")
fs_trans<-transform(fs,tf)

#add transformed data to a gatingset
gs <- GatingSet(fs_trans)
gs
getNodeIDs(gs[[1]]) #only contains root node

#add one gate
rg <- rectangleGate("FSC-H"=c(200,400), "SSC-H"=c(250, 400),
                    filterId="rectangle")

nodeID<-add(gs, rg)#it is added to root node by default if parent is not specified
nodeID
getNodeIDs(gs[[1]]) #the second population is named after filterId of the gate

#add a quadGate
qg <- quadGate("FL1-H"=2, "FL2-H"=4)
nodeIDs<-add(gs,qg,parent="rectangle")
nodeIDs #quadGate produces four population nodes
getNodeIDs(gs[[1]]) #population names are named after dimensions of gate if not specified
```

```

#add a boolean Gate
bg<-booleanFilter('CD15 FITC-CD45 PE+'|'CD15 FITC+CD45 PE-')
bg
nodeID2<-add(gs,bg,parent="rectangle")
nodeID2
getNodes(gs[[1]])
#do the actual gating
recompute(gs)

#plot one gate for one sample
plotGate(gs[[1]],"rectangle")
plotGate(gs[[1]],nodeIDs) #may be smoothed automatically if there are not enough events after gating

#plot gates across samples using lattice plot
plotGate(gs,nodeID)
#plot all gates for one sample
plotGate(gs[[1]])#boolean gate is skipped by default
plotGate(gs[[1]],bool=TRUE)

#plot the gating hierarchy
require(Rgraphviz)
plot(gs[[1]])
#remove one node causing the removal of all the descendants
Rm('rectangle', gs)
getNodes(gs[[1]])

## End(Not run)

```

---

archive

---

*Archive/unarchive a GatingSetInternal to/from tar files*


---

## Description

Save/load a GatingSetInternal which is the gated flow data including gates and populations to/from the disk. The GatingSetInternal object The internal C data structure (gating tree),ncdfFlowSet object(if applicable)

## Usage

```

archive(G,file=tempfile())
unarchive(file,path=tempdir())

```

## Arguments

G	A GatingSetInternal
file	A character scalar giving the output file name.
path	A character scalar giving the folder where to store cdf file(if the raw data within GatingSet is stored as ncdfFlowSet object) extracted from the archive.

**Value**

unarchive returns a GatingSetInternal object

**Author(s)**

Mike Jiang, Greg Finak, N. Gopalakrishnan

Maintainer: Mike Jiang <wjiang2@fhcrc.org>

**See Also**

[GatingSetInternal-class](#)

**Examples**

```
## Not run:
#G is a GatingSet
archive(G, file="gs.tar")
G1<-unarchive(file="gs.tar")

## End(Not run)
```

---

booleanFilter-class    *Class* "booleanFilter"

---

**Description**

A class describing logical operation (& or |) of the reference populations

**Objects from the Class**

Objects can be created by the constructors: booleanFilter(expr, ..., filterId="defaultBooleanFilter")  
char2booleanFilter(expr, ..., filterId="defaultBooleanFilter") expr is an expression or a character string.

**Description**

It inherits class [expressionFilter](#) and exists for the purpose of methods dispatching.

**Methods**

**add** signature(wf = "GatingHierarchyInternal", action="booleanFilter"): add a boolean gate to a GatingHierarchyInternal object.

**show** signature(object = "booleanFilter"): Print information about the booleanFilter.

**Author(s)**

Mike jiang <wjiang2@fhcrc.org>

## References

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

## See Also

[add GatingHierarchy](#)

---

clone	<i>clone a GatingSet</i>
-------	--------------------------

---

## Description

clone a GatingSet

## Usage

```
## S4 method for signature 'GatingSet'  
clone(x, ...)  
## S4 method for signature 'GatingSetInternal'  
clone(x, ...)
```

## Arguments

x	A GatingSet
...	additional arguments to be passed to clone method for GatingSet (not used by GatingSetInternal). <ul style="list-style-type: none"><li>• clone.data=FALSE:logical scalar indicating whether to clone the flow data</li><li>• clone.gating=FALSE:logical scalar indicating whether to clone the gates</li></ul>

## Details

Note that the regular R assignment operation on a GatingSet object does not return the copy as one would normally expect because the GatingSet contains environment slots (and external pointer for GatingSetInternal), which require deep-copying. So make sure to use this clone method in order to make a copy of existing object.

## Value

A copy of a given GatingSet.

## Author(s)

Greg Finak <gfinak@fhcrc.org>



## References

<http://www.rglab.org/>

## Examples

```
## Not run:  
#G is a GatingSet  
G1<-clone(G)  
  
## 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@fhrcr.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	<i>Apply the GatingHierarchy to data, computing population statistics along the way.</i>
---------	--

---

**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,dataEnvironment=NULL,...)
```

**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](mailto: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)
```

---

ExportTSVAnalysis	<i>Export a GatingSet as a set of TSV (tab separated value) files for statistics, gates, and plots.</i>
-------------------	---

---

## Description

This function exports a GatingSet as a set of TSV files for import by other tools such as LabKey.

## Usage

```
ExportTSVAnalysis(x = NULL, Keywords = NULL, EXPORT = "export")
```

## Arguments

x	A GatingSet to be exported.
Keywords	A character vector with a set of keywords used to annotate the GatingSet. (currently not used)
EXPORT	A character vector. The directory into which the exported files will be placed.

## Details

The function generates two tsv files. The first statistics.tsv has the sample names, population names, sample counts and fraction of parent population for all samples and populations in the GatingSet. The second gates.tsv has the population name, sample name, and path to the plots for each gate/population of each sample in the GatingSet. Inside the EXPORT directory it creates a subdirectory for each sample, named after that sample. The subdirectories hold png plots of the gates / populations defined for those samples. The names of the png files are based on the md5 hash of the png file itself. File names are in the graphs.tsv file.

## Value

The function doesn't return anything.

**Author(s)**

Greg Finak <gfinak@fhcrc.org>

**See Also**

[GatingSet-class](#)

---

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)
```

---

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 arguments. 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)
```



---

GatingHierarchy-class *Class "GatingHierarchy"*

---

### Description

GatingHierarchy is a class for representing the gating hierarchy, which can be either imported from a flowJo workspace or constructed in R. GatingHierarchyInternal is a derived class which represents the tree structure in C data structure.

### 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 its own GatingHierarchy. This is different from the workflow representation used in flowCore.

It is also more space efficient by storing gating results as logical/bit vector instead of copying the raw data.

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:** Flag indicating whether ncdf is used to store data and gating indices

## 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> Mike Jiang <wjiang2@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**

GatingSet holds a set of GatingHierarchy objects, representing a set of samples and the gating scheme associated with each. GatingSetInternal is the derived class that holds GatingHierarchyInternal objects.

## 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. Or by: `GatingSet(x,dMode=0,...)`, `x` is a `flowSet/ncdfFlowSet`. It creates an empty gating tree with only root node associated with flow data where `add` methods can be used to add gates. or by `GatingSet(x,y,path=".",isNcdf=FALSE,dMode=1,...)`, `x` is an existing gating hierarchy (used as a gating template) and `y` is the flow data to be gated (`flowSet/ncdfFlowSet`). It creates the `GatingSet` from the FCS files and copies the gates defined in the existing gating hierarchy to the new `GatingSet`. `dMode` specifies the level of print-out that is used for the debug purpose. ... include `compensation` argument that can provide the customized compensation object if specified and other arguments passed to `read.flowSet` or `read.ncdfFlowSet` functions.

## 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 GatingHierarchy 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.

**rbind2** signature(x = "GatingSetInternal", y = "GatingSetInternal"):combine two GatingSets into one.

### 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);
```

---

GatingSetList-class    *Class "GatingSetList"*

---

### Description

A list of of GatingSet objects. This class exists for method dispatching.

### Objects from the Class

Objects can be created by the constructor GatingSetList(x). x is a list of GatingSets.

### Description

Objects store a collection of GatingSets,which usually has the same gating trees and markers.

**Methods**

**getSamples** signature(x = "GatingSetList"): Get the sample names of all the GatingSets in this list

**rbind2** signature(x = "GatingSetList"):combine all the GatingSets from the list into one GatingSet object.

**show** signature(object = "GatingSetList"): Print information about the GatingSetList.

**Author(s)**

Greg Finak <gfinak@fhcrc.org>

**References**

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

**See Also**

[GatingSet](#) [GatingHierarchy](#)

**Examples**

```
## Not run:
#G1,G2,G3 is a GatingSet
GL<-GatingSetList(list(G1,G3,G3))
show(GL)
getSamples(GL)
G<-rbind2(GL)

## End(Not run)
```

---

getBoundaries	<i>Get the boundaries of a gate or dimensions on which a gate is applied within a GatingHierarchy</i>
---------------	---

---

**Description**

Get the boundaries (vertices) of a flowJo gate on the transformed scale or the dimension names on which a gate in a GatingHierarchy is applied.

**Usage**

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

**Arguments**

obj	A GatingHierarchy
y	A character, the name of the node / gate / population of interest .
index	TRUE   FALSE a logical indicating whether we should return the names of the dimensions (FALSE, default) or the indices of the dimensions (TRUE)

**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**

getBoundaries returns a matrix with column names corresponding to channels / dimensions, and rows to x,y tuples of vertices for polygon gates in these dimensions. getDimensions returns 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**

[getGate](#), [getNode](#)

**Examples**

```
## Not run:
file<-"myworkspace.xml"
ws<-openWorkspace(file)
G<-parseWorkspace(ws,execute=TRUE,path=".")
n<-getNode(G[[1]],tsort=TRUE)[3] #get the third node in the first gating hierarchy (topological sort order)
getGate(G[[1]],n); #return the gate for that node.
#Fetch the dimensions for the fifth population in the hierarchy.
getDimensions(G,getNode(G)[5],index=FALSE)
getBoundaries(G,getNode(G)[5])

## End(Not run)
```

getCompensationMatrices

*Retrieve the compensation matrices from a flowJo Workspace or GatingHierarchy*

---

### **Description**

Retrieve the compensation matrices from a flowJo workspace or GatingHierarchy.

### **Usage**

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

### **Arguments**

x                    A flowJoWorkspace or GatingHierarchy object.

### **Details**

Return all the compensation matrices in a flowJoWorkspace object or a compensation matrix in a GatingHierarchy.

### **Value**

A list of matrix representing the spillover matrices in the flowJoWorkspace or a spillover matrix in GatingHierarchy

### **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	<i>Return the flowFrame/flowSet associated with a GatingHierarchy/GatingSet</i>
---------	---

---

### Description

Return the flowFrame/flowSet associated with a GatingHierarchy/GatingSet

### 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	deprecated.

### Details

Returns a flowFrame/flowSet 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, the trees representing the GatingHierarchy for each sample in the GaingSet are presumed to have the same structure.

**Value**

A flowFrame object if obj is a GatingHierarchy or graphNEL. A flowSet if a GatingSet. A ncdfFlowSet if the node name/index (y) is NULL and the flow data is stored as cdf.

**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
getData(G,3) #get a flowSet constructed from the third node / population in the tree.

#G is a GatingHierarchy
getData(G,)

## End(Not run)
```

---

getFJWSubsetIndices	<i>Fetch the indices for a subset of samples in a flowJo workspace, based on a keyword value pair</i>
---------------------	---

---

**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=NULL, value=NULL, group, requiregates=TRUE)
```

**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.
requiregates	TRUE or FALSE, specifying whether we include only samples that have gates attached or whether we include any sample in the workspace.

**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. The output 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	<i>Return the flowCore gate definition associated with a node in a GatingHierarchy/GatingSet.</i>
---------	---

---

**Description**

Return the flowCore gate definition object associated with a node in a GatingHierarchy or GatingSet 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 GatingHierarchy or GatingSet
y	A character the name of the node of interest. Or, a numeric index into the node list of nodes in the GatingHierarchy or GatingSet.
tsort	deprecated.

**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](#) [getNode](#)

**Examples**

```
## Not run: #gh is a GatingHierarchy
getGate(gh,5) #return the gate for the fifth node in the tree.
getGate(gh,getNodes(gh)[5]) #return the gate for the fifth node in the tree, but fetch it by name.
#G is a GatingSet
getGate(G,5) #return a list of gates for the fifth node in each tree

## End(Not run)
```

---

getIndiceFile	<i>Returns the .nc file used to store the gating indices for the GatingHierarchy.</i>
---------------	---

---

**Description**

If netCDF is being used to store the data, then this function returns the .nc file used to store the gating indices for the GatingHierarchy.

**Usage**

```
getIndiceFile(obj)
```

**Arguments**

obj	A GatingHierarchy
-----	-------------------

**Details**

Returns the .nc file used to store the gate indices for this GatingHierarchy. Used internally, not meant for user consumption at this time.

**Value**

A character vector, the name of the .nc file.

**Author(s)**

Greg Finak <gfinak@fhcrc.org>

---

getIndices	<i>Get the membership indices for each event with respect to a particular gate in a GatingHierarchy</i>
------------	---

---

**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 or numeric giving the name or index 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	<i>Get List of Keywords for a Flow Sample</i>
-------------	---

---

## 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	<i>Get the names of all nodes in a gating hierarchy</i>
----------	---

---

**Description**

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

**Usage**

```
## S4 method for signature 'GatingHierarchy'
getNodes(x,y=NULL,tsort=FALSE,isPath=FALSE,...)
## S4 method for signature 'GatingHierarchyInternal'
getNodes(x,y=NULL,order="regular",isPath=FALSE,prefix=FALSE,...)
```

**Arguments**

x	A GatingHierarchy
y	A character the name of the node of interest. Or, a numeric index into the node list of nodes in the GatingHierarchy.
order	order=c("regular", "tsort", "bfs") returns the nodes in regular, topological or breadth-first sort order. "regular" is default.
isPath	A logical scalar to tell the method whether to return the full gating path or just terminal node name
prefix	A logical scalar to tell the method whether to add internal node index as the prefix to the node name
tsort	A logical scalar to tell the method whether to return the nodes in topological sort order. It is replaced by order argument for GatingHierarchyInternal class.
...	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
getNodeNames(G[[1]])#return node names
getNodeNames(G[[1]],isPath=TRUE)#return the full path

## End(Not run)
```

---

getParent

*Return the name of the parent population or a list of child populations of the current population in the GatingHierarchy*

---

**Description**

Returns the name of the parent population or a character/numeric vector of all the children 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)
## S4 method for signature 'GatingHierarchy,character'
getChildren(obj, y)
## S4 method for signature 'GatingHierarchyInternal,character'
getChildren(obj, y, tsort = FALSE)
## S4 method for signature 'GatingHierarchyInternal,numeric'
getChildren(obj, y)
```

**Arguments**

obj            A GatingHierarchy  
y                a character/numeric name or node indices of the node / population.  
tsort           deprecated.

**Value**

getParent returns a character vector, the name of the parent population. getChildren returns a character or numeric vector of the node names or node indices 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/>

**See Also**

[getNode](#)s

**Examples**

```
## Not run:
#G is a gatinghierarchy
#return the name of the parent of the fifth node in the hierarchy.
getParent(G,getNode(G[[1])[5])
n<-getNode(G,tsort=T)[4];
getChildren(G,n);#Get the names of the child nodes of the 4th node in this gating hierarchy.
getChildren(G,4);#Get the ids of the child nodes

## End(Not run)
```

---

getPopStats	<i>Return a table of population statistics for all populations in a GatingHierarchy/GatingSet or the population proportions or the total number of events of a node (population) in a GatingHierarchy</i>
-------------	---

---

**Description**

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

getPopStats is more useful than getPop. Returns a table of population statistics for all populations in a GatingHierarchy/GatingSet. Includes the flowJo counts, flowCore counts and frequencies.

getTotal returns the total number of events in the gate defined in the GatingHierarchy object

**Usage**

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

**Arguments**

x	A GatingHierarchy or GatingSet
...	Additional arguments
y	character The name of the node. A list of nodes is accessible via <code>getNode(x)</code> .

**Details**

`getPopStats` 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.) for a `GatingSet`, returns a matrix of proportions for all populations and all samples `getProp` returns the proportion of cells in the gate, relative to its parent. `getTotal` returns 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**

`getPopStats` returns a `data.frame` with columns for the population name, `flowJo` derived counts, `flowCore` derived counts, and the population proportions (relative to their parent population). `getProp` returns a population frequency numeric. `getTotal` returns a numeric value of the total number of elements in the population.

**Author(s)**

Greg Finak <[gfinak@fhcrc.org](mailto:gfinak@fhcrc.org)>

**References**

<http://www.rglab.org/>

**See Also**

[getNode](#)

**Examples**

```
## Not run:
#If gh is a GatingHierarchy
getPopStats(gh);
#proportion for the fifth population
getProp(G,getNode(gh)[5])
getTotal(G,getNode(G,tsort=T)[5])

## End(Not run)
```

---

getSample	<i>Get the sample name associated with a GatingHierarchy</i>
-----------	--

---

**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	<i>Get a table of sample groups from a flowJo workspace</i>
-----------------	---

---

**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)
```

---

getTransformations	<i>Return a list of transformations or a transformation in a flowJo workspace/GatingHierarchy</i>
--------------------	---

---

**Description**

Return a list of all the transformations or a transformation in a flowJo workspace/GatingHierarchy

**Usage**

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

**Arguments**

x                    A flowJoWorkspace or GatingHierarchy object

**Details**

Returns a list of the transformations or a transformation 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)
```

---

haveSameGatingHierarchy

*Tests whether two GatingHierarchy or GatingSet objects have the same gating structure.*

---

**Description**

Tests whether GatingHierarchy, or GatingSet objects have the same gating structure.

**Usage**

```
## S4 method for signature 'GatingHierarchy,GatingHierarchy'
haveSameGatingHierarchy(object1, object2)
## S4 method for signature 'GatingHierarchy,GatingSet'
haveSameGatingHierarchy(object1, object2)
## S4 method for signature 'GatingSet,GatingHierarchy'
haveSameGatingHierarchy(object1, object2)
## S4 method for signature 'GatingSet,GatingSet'
haveSameGatingHierarchy(object1, object2)
## S4 method for signature 'GatingSet,missing'
haveSameGatingHierarchy(object1, object2)
## S4 method for signature 'list,missing'
haveSameGatingHierarchy(object1, object2)
```



**Arguments**

object1            A GatingHierarchy, GatingSet, or list of GatingHierarchy objects.  
 object2            A GatingHierarchy, GatingSet, or missing.

**Details**

Compares the gating tree of the two objects, or of the GatingHierarchies in a single GatingSet.

**Value**

A logical, TRUE if the gating structures are the same, or FALSE if they differ.

**Author(s)**

Greg Finak <gfinak@fhcrc.org>

---

includedChannel2ExcludedChannel

*Convert a list of parameter names in a GatingSet to its complement.*

---

**Description**

Convert a list of parameter names in a GatingSet to its complement.

**Usage**

```
includedChannel2ExcludedChannel(gs, includedims = NULL)
```

**Arguments**

gs                    A GatingSet.  
 includedims        A character vector of channel / parameter names in the gating set.

**Value**

A character vector of channel names. The complement of includedims.

**Author(s)**

Greg Finak <gfinak@fhcrc.org>

**Examples**

```
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.
```

---

includedGate2ExcludedGate

*Convert a set of gates given by their flowJo gate paths to its complement (and excluding Boolean gates), referenced by gate indices recognized by [normalize](#).*

---

### Description

Convert a set of gates given by their flowJo gate paths to its complement (and excluding Boolean gates), referenced by gate indices recognized by [normalize](#).

### Usage

```
includedGate2ExcludedGate(gs, includegates)
```

### Arguments

gs	A GatingSet
includegates	A character vector of flowJo gating paths for the gates in gs

### Value

A numeric vector of gate indices to be used with [normalize](#).

### Author(s)

Greg Finak <gfinak@fhcrc.org>

### Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.

## The function is currently defined as
```

---

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="GatingSet",keyword="character")

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	<i>Open/Close a flowJo workspace</i>
---------------	--------------------------------------

---

### Description

Open a flowJo workspace and return a flowJoWorkspace object. Close a flowJoWorkspace, destroying the internal representation of the XML document, and freeing the associated memory.

### Usage

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

### Arguments

file	Full path to the XML flowJo workspace file.
workspace	A flowJoWorkspace

### Details

Open an XML flowJo workspace file and return a flowJoWorkspace object. The workspace is represented using a XMLInternalDocument object. Close a flowJoWorkspace after finishing with it. This is necessary to explicitly clean up the C-based representation of the XML tree. (See the XML package).

### Value

openWorkspace 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
closeWorkspace(ws);

## End(Not run)
```

---

parseWorkspace	<i>Parse a flowJo Workspace</i>
----------------	---------------------------------

---

**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,useInternal = TRUE,name=NULL,execute=TRUE,isNcdf=FALSE,subset=NULL,nslaves=4,requ
```

**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. TRUE by default.
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.
useInternal	TRUE FALSE a logical scalar specifying whether to parse the workspace as internal C structure or regular R object. internal structure will create GatingSetInternal object which is of the derived class from GatingSet
subset	numeric vector specifying the subset of samples in a group to import. Or a character specifying the FCS filenames to be imported.
nslaves	numeric number of slave processes for executing the gating under Rmpi
requiregates	logical Should samples that have no gates be included?

includeGates	logical Should gates be imported, or just the data with compensation and transformation?
dMode	An integer scalar specifying the level of print-out that is used for the debug purpose.
path	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.
...	<ul style="list-style-type: none"> <li>• sampNloc="keyword": a character scalar indicating where to get sample-Name(or FCS filename) within xml workspace. It is either from "keyword" or "sampleNode".</li> <li>• compensation=NULL: a matrix that allow the customized compensation matrix to be used instead of the one specified in flowJo workspace.</li> <li>• ...: Additional arguments to be passed to <a href="#">read.ncdfFlowSet</a> or <a href="#">read.flowSet</a>.</li> </ul>

### 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.

The `subset` argument allows one to select a set of files from the chosen sample group. The routine will take the intersection of the files in the sample group, the files specified in `subset` and the files available on disk, and import them.

### 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`. This is done automatically by default, and there is no more reason to set this argument to `FALSE`.

### Author(s)

Greg Finak <[gfinak@fhcrc.org](mailto:gfinak@fhcrc.org)>

### References

<http://www.rglab.org/>

### See Also

[getSampleGroups](#), [GatingSet](#)

**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 below the current direct
#G is a GatingSet.
G1<-parseWorkspace(ws,userInternal=TRUE)
#G1 is a GatingSetInternal.

## 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,fixedsized=FALSE,boolean=FALSE,...
```

**Arguments**

x	The GatingHierarchy to be plotted
y	missing.
layout	See <a href="#">layoutGraph</a> in package Rgraphviz
width	See <a href="#">layoutGraph</a> in package Rgraphviz
height	See <a href="#">layoutGraph</a> in package Rgraphviz
fontsize	See <a href="#">layoutGraph</a> in package Rgraphviz
labelfontsize	See <a href="#">layoutGraph</a> in package Rgraphviz
fixedsized	See <a href="#">layoutGraph</a> 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 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,...)
## S4 method for signature 'GatingHierarchy,numeric'
plotGate(x, y, bool=FALSE,main=getSample(x),arrange=TRUE,merge=TRUE,...)
## S4 method for signature 'GatingSet,character'
plotGate(x, y, lattice=TRUE,bool=FALSE,merge=TRUE,...)
```

**Arguments**

x	A GatingHierarchy
y	A character or numeric representing the node in the GatingHierarchy. Nodes can be accessed with <code>getNodes(GatingHierarchy)</code> .
bool	TRUE FALSE logical specifying whether to plot boolean gates.
main	character, The main title of the plot. Default is the sample name.
arrange	TRUE FALSE logical indicating whether to arrange different populations/nodes on the same page via <code>grid.arrange</code> call.



merge	TRUE FALSE logical indicating whether to draw multiple gates on the same plot if these gates share the same parent population and same x,y dimensions/parameters;
lattice	TRUE FALSE logical indicating whether to draw one node/gate on multiple samples on the same page through lattice plot;
...	<ul style="list-style-type: none"> <li>• formula: formula a formula passed to xyplot function of flowViz, by default it is NULL, which means the formula is generated according to the x,y parameters associated with gate.</li> <li>• cond: character the conditioning variable to be passed to lattice plot.</li> <li>• overlay: either a numeric scalar indicating the index of a gate/population within the GatingHierarchy or a logical vector that indicates the cell event indices representing a sub-cell population. This cell population is going to be plotted on top of the existing gates(defined by y argument) as an overlay.</li> <li>• ...: The other additional arguments to be passed to <a href="#">xyplot</a>.</li> </ul>

### 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. When applied to a `GatingHierarchy`, `arrange` is set as `TRUE`, then all the gates associated with it are plotted as different panel on the same page. If `arrange` is `FALSE`, then it plots one gate at a time. By default `merge` is set as `TRUE`, plot multiple gates on the same plot when they share common parent population and axis. When applied to a `GatingSet`, if `lattice` is `TRUE`, it plots one gate (multiple samples) per page, otherwise, one sample (with multiple gates) per page.

### Value

return a trellis object if `arrange` is `FALSE`,

### Author(s)

Greg Finak <[gfinak@fhcrc.org](mailto: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**

<code>x</code>	A GatingHierarchy from a flowJoWorkspace, or a GatingSet.
<code>m</code>	numeric The number of rows in the panel plot. Now deprecated, uses lattice.
<code>n</code>	numeric The number of columns in the panel plot. Now deprecated, uses lattice.
<code>...</code>	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	<i>plot a workflow</i>
--------	------------------------

---

**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)
```

---

recomputeGatingSet	<i>Recalculates all the gating in the GatingSet if netCDF storage is used.</i>
--------------------	--

---

**Description**

Re-runs all the gating in a GatingSet object if netCDF storage is used.

**Usage**

```
recomputeGatingSet(x)
```

**Arguments**

x	A GatingSet object.
---	---------------------

**Details**

This function will recalculate all the gates in a gating set object. Useful if you've run some normalization and need to recalculate the gates based on the adjusted data. Only if netCDF storage is used.

**Value**

Returns NULL. The gates are written directly to the netCDF file and reflected in the parent frame.

**Author(s)**

Greg Finak <gfinak@fhcrc.org>

---

saveNcdf	<i>Save the netcdf file of a GatingSet or GatingHierarchy to a permanent location.</i>
----------	--

---

**Description**

When creating a new GatingSet or GatingHierarchy using netcdf backed storage, the nc files are written to the TEMP directory. This will move those files to a more permanent location, allowing the object to be saved and persist after restarting R and reloading the workspace.

**Usage**

```
saveNcdf(objName, path = getwd())
```

**Arguments**

objName	The name of the GatingSet or GatingHierarchy. A character vector.
path	A character vector: the path where the netcdf files should be saved.

**Details**

The function will modify the GatingSet or GatingHierarchy to reflect the new location of the ncdf files. Do not pass the object directly via objName, but rather pass the name of the variable holding the GatingSet or GatingHierarchy as a character vector.

**Value**

Does not return anything. Modifies the object in the parent environment.

**Author(s)**

Greg Finak <gfinak@fhcrc.org>

---

summary-methods

*Summarize a flowJoWorkspace object*

---

**Description**

Summarize a flowJoWorkspace object.

# Index

- \*Topic **IO**
  - archive, 6
- \*Topic **add**
  - add, 3
- \*Topic **classes**
  - booleanFilter-class, 7
  - flowJoWorkspace-class, 14
  - GatingHierarchy-class, 17
  - GatingSet-class, 19
  - GatingSetList-class, 21
- \*Topic **hplot**
  - plot, 47
  - plotGate, 48
  - plotPopCV, 50
  - plotWf, 51
- \*Topic **manip**
  - clone, 8
  - ellipsoidGate2FlowJoVertices, 9
  - execute, 10
  - exportAsFlowJoXML, 12
  - ExportTSVAnalysis, 13
  - flowWorkspace2flowCore, 15
  - getBoundaries, 22
  - getCompensationMatrices, 24
  - getData, 25
  - getFJWSubsetIndices, 26
  - getGate, 27
  - getIndiceFile, 29
  - getIndices, 29
  - getNodes, 32
  - getParent, 33
  - getPopStats, 34
  - getSample, 36
  - getSampleGroups, 37
  - getSamples, 38
  - getTransformations, 39
  - haveSameGatingHierarchy, 40
  - includedChannel2ExcludedChannel, 41
- includedGate2ExcludedGate, 42
- keyword-methods, 42
- openWorkspace, 44
- parseWorkspace, 45
- saveNcdf, 52
- \*Topic **methods**
  - keyword-methods, 42
  - lapply-methods, 43
  - length-methods, 43
  - summary-methods, 53
- \*Topic **package**
  - flowWorkspace-package, 3
- \*Topic **univar**
  - plotPopCV, 50
- \*Topic **utilities**
  - flowWorkspace-package, 3
- .includedChannel2ExcludedChannel
  - (includedChannel2ExcludedChannel), 41
- .includedGate2ExcludedGate
  - (includedGate2ExcludedGate), 42
- [,GatingSet,ANY-method
  - (GatingSet-class), 19
- [,GatingSetInternal,ANY-method
  - (GatingSet-class), 19
- [[,GatingSet,ANY-method
  - (GatingSet-class), 19
- [[,GatingSetInternal,ANY-method
  - (GatingSet-class), 19
- add, 3, 8, 20
- add,GatingHierarchyInternal,booleanFilter-method
  - (add), 3
- add,GatingHierarchyInternal,polygonGate-method
  - (add), 3
- add,GatingHierarchyInternal,quadGate-method
  - (add), 3
- add,GatingHierarchyInternal,rectangleGate-method
  - (add), 3
- add,GatingSet,list-method (add), 3

- add, GatingSetInternal, filter-method (add), 3
- add, GatingSetInternal, filterList-method (add), 3
- AnnotatedDataFrame, 21
- archive, 6
  
- booleanFilter (booleanFilter-class), 7
- booleanFilter-class, 7
  
- char2booleanFilter (booleanFilter-class), 7
- clone, 8
- clone, GatingSet-method (clone), 8
- clone, GatingSetInternal-method (clone), 8
- clone-methods (clone), 8
- closeWorkspace (openWorkspace), 44
- closeWorkspace, flowJoWorkspace-method (openWorkspace), 44
- closeWorkspace-methods (openWorkspace), 44
  
- ellipse, 10
- ellipsoidGate2FlowJoVertices, 9
- ellipsoidGate2FlowJoVertices, ellipsoidGate-method (ellipsoidGate2FlowJoVertices), 9
- ellipsoidGate2FlowJoVertices-methods (ellipsoidGate2FlowJoVertices), 9
- execute, 10
- execute, GatingHierarchy-method (execute), 10
- execute-methods (execute), 10
- exportAsFlowJoXML, 12
- exportAsFlowJoXML, ellipsoidGate-method (exportAsFlowJoXML), 12
- exportAsFlowJoXML, intersectFilter-method (exportAsFlowJoXML), 12
- exportAsFlowJoXML, list-method (exportAsFlowJoXML), 12
- exportAsFlowJoXML, polygonGate-method (exportAsFlowJoXML), 12
- exportAsFlowJoXML, rectangleGate-method (exportAsFlowJoXML), 12
- exportAsFlowJoXML, workFlow-method (exportAsFlowJoXML), 12
  
- exportAsFlowJoXML-methods (exportAsFlowJoXML), 12
- ExportTSVAnalysis, 13
- expressionFilter, 7
  
- flowJoWorkspace, 21
- flowJoWorkspace-class, 14
- flowWorkspace (flowWorkspace-package), 3
- flowWorkspace-package, 3
- flowWorkspace2flowCore, 15
- flowWorkspace2flowCore, flowJoWorkspace-method (flowWorkspace2flowCore), 15
- flowWorkspace2flowCore, GatingHierarchy-method (flowWorkspace2flowCore), 15
- flowWorkspace2flowCore, GatingHierarchyInternal-method (flowWorkspace2flowCore), 15
- flowWorkspace2flowCore, GatingSet-method (flowWorkspace2flowCore), 15
- flowWorkspace2flowCore, GatingSetInternal-method (flowWorkspace2flowCore), 15
- flowWorkspace2flowCore-methods (flowWorkspace2flowCore), 15
  
- GatingHierarchy, 8, 15, 21, 22
- GatingHierarchy-class, 17
- GatingHierarchyInternal-class (GatingHierarchy-class), 17
- GatingSet, 15, 19, 22, 46
- GatingSet (GatingSet-class), 19
- GatingSet, character, character-method (GatingSet-class), 19
- GatingSet, flowSet, ANY-method (GatingSet-class), 19
- GatingSet, flowSet-method (GatingSet-class), 19
- GatingSet, GatingHierarchyInternal, character-method (GatingSet-class), 19
- GatingSet-class, 19
- GatingSetInternal-class (GatingSet-class), 19
- GatingSetList (GatingSetList-class), 21
- GatingSetList-class, 21
- getBoundaries, 22
- getBoundaries, GatingHierarchy, character-method (getBoundaries), 22
- getBoundaries-methods (getBoundaries), 22
- getChildren (getParent), 33

- getChildren,GatingHierarchy,character-method (getGate), 27  
(getParent), 33
- getChildren,GatingHierarchyInternal,character-method (getIndiceFile), 29  
(getParent), 33
- getChildren,GatingHierarchyInternal,numeric-method (getIndiceFile-methods (getIndiceFile),  
(getParent), 33 29
- getChildren-methods (getParent), 33
- getCompensationMatrices, 24
- getCompensationMatrices,flowJoWorkspace-method (getIndices), 29  
(getCompensationMatrices), 24
- getCompensationMatrices,GatingHierarchy-method (getIndices), 29  
(getCompensationMatrices), 24
- getCompensationMatrices,GatingHierarchyInternal-method (getIndices), 29  
(getCompensationMatrices), 24
- getCompensationMatrices-methods (getIndices-methods (getIndices), 29  
(getCompensationMatrices), 24
- getData, 25, 28
- getData,GatingHierarchy-method (getKeywords), 31  
(getData), 25
- getData,GatingHierarchyInternal-method (getKeywords,flowJoWorkspace,character-method  
(getData), 25 (getKeywords), 31
- getData,GatingSet-method (getData), 25 (getKeywords,GatingHierarchy,missing-method  
(getData), 25 (getKeywords), 31
- getData,GatingSetInternal-method (getKeywords,GatingHierarchyInternal,missing-method  
(getData), 25 (getKeywords), 31
- getData,graphNEL-method (getData), 25 (getKeywords,GatingSet,character-method  
(getData), 25 (getKeywords), 31
- getData-methods (getData), 25 (getKeywords-methods (getKeywords), 31
- getDimensions (getBoundaries), 22
- getDimensions,GatingHierarchy,character-method (getNode), 23, 28, 32, 34, 35  
(getBoundaries), 22 (getNode), 32
- getDimensions-methods (getBoundaries), 22 (getNode),GatingHierarchy-method  
(getBoundaries), 22 (getNode), 32
- getFJWSubsetIndices, 26 (getNode),GatingHierarchyInternal-method  
(getNode), 32
- getGate, 23, 27 (getNode)-methods (getNode), 32
- getGate,GatingHierarchy,character-method (getParent), 33  
(getGate), 27 (getParent,GatingHierarchy,character-method  
(getParent), 33
- getGate,GatingHierarchy,numeric-method (getParent,GatingHierarchy,numeric-method  
(getGate), 27 (getParent), 33
- getGate,GatingHierarchyInternal,character-method (getParent,GatingHierarchyInternal,character-method  
(getGate), 27 (getParent), 33
- getGate,GatingHierarchyInternal,numeric-method (getParent,GatingHierarchyInternal,numeric-method  
(getGate), 27 (getParent), 33
- getGate,GatingSet,character-method (getParent-methods (getParent), 33  
(getGate), 27 (getParent,GatingSet,character-method  
(getParent), 33
- getGate,GatingSet,numeric-method (getParent,GatingSet,numeric-method  
(getGate), 27 (getParent), 33
- getGate,GatingSetInternal,character-method (getParent,GatingSetInternal,character-method  
(getGate), 27 (getParent), 33
- getGate,GatingSetInternal,numeric-method (getParent,GatingSetInternal,numeric-method  
(getGate), 27 (getParent), 33
- getIndiceFile, 29
- getIndices, 26, 29
- getIndices,GatingHierarchy,character-method (getPopStats), 26, 30, 34, 50  
(getIndices), 29
- getIndices,GatingHierarchyInternal,character-method (getPopStats,GatingHierarchy-method  
(getIndices), 29 (getPopStats), 34
- getIndices,GatingHierarchyInternal,numeric-method (getPopStats,GatingHierarchyInternal-method  
(getIndices), 29 (getPopStats), 34
- getIndices-methods (getIndices), 29
- getKeywords, 31
- getKeywords,flowJoWorkspace,character-method (getPopStats,GatingSet-method
- getKeywords,GatingHierarchy,missing-method (getPopStats), 34
- getKeywords,GatingHierarchyInternal,missing-method (getPopStats), 34
- getKeywords,GatingSet,character-method (getPopStats), 34
- getKeywords,GatingSet,numeric-method (getPopStats), 34
- getKeywords-methods (getKeywords), 31
- getNode, 23, 28, 32, 34, 35
- getNode,GatingHierarchy-method (getPopStats), 34
- getNode,GatingHierarchyInternal-method (getPopStats), 34
- getNode-methods (getNode), 32
- getParent, 33
- getParent,GatingHierarchy,character-method (getPopStats), 34
- getParent,GatingHierarchy,numeric-method (getPopStats), 34
- getParent,GatingHierarchyInternal,character-method (getPopStats), 34
- getParent,GatingHierarchyInternal,numeric-method (getPopStats), 34
- getParent-methods (getParent), 33
- getPopStats, 26, 30, 34, 50
- getPopStats,GatingHierarchy-method (getPopStats), 34
- getPopStats,GatingHierarchyInternal-method (getPopStats), 34
- getPopStats,GatingSet-method



- (getPopStats), 34
- getPopStats,GatingSetInternal-method
  - (getPopStats), 34
- getPopStats-methods (getPopStats), 34
- getProp, 26, 30
- getProp (getPopStats), 34
- getProp,GatingHierarchy,character-method
  - (getPopStats), 34
- getProp,GatingHierarchyInternal,character-method
  - (getPopStats), 34
- getProp-methods (getPopStats), 34
- getSample, 36
- getSample,GatingHierarchy-method
  - (getSample), 36
- getSample,GatingHierarchyInternal-method
  - (getSample), 36
- getSample,graphNEL-method (getSample), 36
- getSample-method (getSample), 36
- getSampleGroups, 37, 46
- getSampleGroups,flowJoWorkspace-method
  - (getSampleGroups), 37
- getSampleGroups-methods
  - (getSampleGroups), 37
- getSamples, 38
- getSamples,flowJoWorkspace-method
  - (getSamples), 38
- getSamples,GatingSet-method
  - (getSamples), 38
- getSamples,GatingSetInternal-method
  - (getSamples), 38
- getSamples,GatingSetList-method
  - (GatingSetList-class), 21
- getSamples-methods (getSamples), 38
- getTotal (getPopStats), 34
- getTotal,GatingHierarchy,character-method
  - (getPopStats), 34
- getTotal,GatingHierarchyInternal,character-method
  - (getPopStats), 34
- getTotal-methods (getPopStats), 34
- getTransformations, 39
- getTransformations,flowJoWorkspace-method
  - (getTransformations), 39
- getTransformations,GatingHierarchy-method
  - (getTransformations), 39
- getTransformations,GatingHierarchyInternal-method
  - (getTransformations), 39
- getTransformations-methods
  - (getTransformations), 39
- haveSameGatingHierarchy, 40
- haveSameGatingHierarchy,GatingHierarchy,GatingHierarchy-method
  - (haveSameGatingHierarchy), 40
- haveSameGatingHierarchy,GatingHierarchy,GatingSet-method
  - (haveSameGatingHierarchy), 40
- haveSameGatingHierarchy,GatingSet,GatingHierarchy-method
  - (haveSameGatingHierarchy), 40
- haveSameGatingHierarchy,GatingSet,GatingSet-method
  - (haveSameGatingHierarchy), 40
- haveSameGatingHierarchy,GatingSet,missing-method
  - (haveSameGatingHierarchy), 40
- haveSameGatingHierarchy,GatingSetInternal,missing-method
  - (haveSameGatingHierarchy), 40
- haveSameGatingHierarchy,list,missing-method
  - (haveSameGatingHierarchy), 40
- haveSameGatingHierarchy-methods
  - (haveSameGatingHierarchy), 40
- includedChannel2ExcludedChannel, 41
- includedGate2ExcludedGate, 42
- keyword (keyword-methods), 42
- keyword,GatingHierarchy,character
  - (keyword-methods), 42
- keyword,GatingHierarchy,character-method
  - (keyword-methods), 42
- keyword,GatingHierarchyInternal,character
  - (keyword-methods), 42
- keyword,GatingHierarchyInternal,character-method
  - (keyword-methods), 42
- keyword,GatingSet,character
  - (keyword-methods), 42
- keyword,GatingSet,character-method
  - (keyword-methods), 42
- keyword-methods, 42
- lapply,ANY-method (lapply-methods), 43
- lapply,GatingSet-method
  - (lapply-methods), 43
- lapply-methods, 43
- layoutGraph, 47, 48
- length,GatingSet-method
  - (length-methods), 43
- length-methods, 43
- normalize, 42
- openWorkspace, 24, 37, 44

- openWorkspace, character-method  
(openWorkspace), 44
- openWorkspace-methods (openWorkspace),  
44
- parseWorkspace, 11, 16, 19, 27, 45
- parseWorkspace, flowJoWorkspace-method  
(parseWorkspace), 45
- parseWorkspace-methods  
(parseWorkspace), 45
- plot, 47
- plot, GatingHierarchy, missing-method  
(plot), 47
- plot, GatingHierarchyInternal, missing-method  
(plot), 47
- plot-methods (plot), 47
- plotGate, 48
- plotGate, GatingHierarchy, character-method  
(plotGate), 48
- plotGate, GatingHierarchy, missing-method  
(plotGate), 48
- plotGate, GatingHierarchy, numeric-method  
(plotGate), 48
- plotGate, GatingSet, character-method  
(plotGate), 48
- plotGate, GatingSet, missing-method  
(plotGate), 48
- plotGate, GatingSet, numeric-method  
(plotGate), 48
- plotGate-methods (plotGate), 48
- plotGate\_labkey (plotGate), 48
- plotPopCV, 50
- plotPopCV, GatingHierarchy-method  
(plotPopCV), 50
- plotPopCV, GatingSet-method (plotPopCV),  
50
- plotPopCV-methods (plotPopCV), 50
- plotWf, 51
- rbind2, GatingSet, GatingSet-method  
(GatingSet-class), 19
- rbind2, GatingSetList, missing-method  
(GatingSetList-class), 21
- read.flowSet, 20, 46
- read.ncdfFlowSet, 20, 46
- recompute (add), 3
- recompute, GatingSet-method (add), 3
- recompute, GatingSetInternal-method  
(add), 3
- recomputeGatingSet, 52
- Rm (add), 3
- Rm, character, GatingHierarchyInternal, character-method  
(add), 3
- Rm, character, GatingSetInternal, character-method  
(add), 3
- saveNcdf, 52
- show, booleanFilter-method  
(booleanFilter-class), 7
- show, flowJoWorkspace-method  
(summary-methods), 53
- show, GatingHierarchy-method  
(summary-methods), 53
- show, GatingHierarchyInternal-method  
(GatingSet-class), 19
- show, GatingSet-method  
(summary-methods), 53
- show, GatingSetInternal-method  
(GatingSet-class), 19
- show, GatingSetList-method  
(GatingSetList-class), 21
- show-methods (summary-methods), 53
- summary, ANY-method (summary-methods), 53
- summary, flowJoWorkspace-method  
(summary-methods), 53
- summary-methods, 53
- tsort, 33
- unarchive (archive), 6
- xmlTreeParse, 44
- xyplot, 49