Package 'flowWorkspace'

October 7, 2014

Type Package

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

Version 3.10.09

Date 2011-06-10

Author Greg Finak, Mike Jiang

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, BiocGenerics, graph, graphics, lattice, methods,stats, stats4, utils, RBGL, graph, XML, Biobase, IDPmisc,Cairo, tools,hexbin,gridExtra,Rgraphviz ,data.table ,plyr ,latticeExtra ,Rcpp

Collate 'AllGenerics.R' 'AllClasses.R' 'flowJoWorkspace-Methods.R' 'GatingHierarchy-Methods.R' 'GatingSet-Methods.R' 'GatingSetList-Methods.R' 'filterObject-Methods.R' 'add-Methods.R' 'setGate-Methods.R' 'zzz.R'

Depends R (>= 2.16.0),flowCore,flowViz,ncdfFlow,gridExtra

biocViews FlowCytometry, DataImport, Preprocessing, DataRepresentation

Suggests testthat ,flowWorkspaceData ,RSVGTipsDevice

LinkingTo Rcpp

R topics documented:

`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:	R (>= 2.16.0),Rcpp (>= 0.9.9)

Author(s)

Greg Finak, Mike Jiang

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. `setGate` method update the gate of one population node in 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.

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 `GatingHierrarchy` 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. • 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 • 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 • 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.
`symbol`	A `character` identifies the population node in a `GatingHierrarchy` 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 `GatingHierrarchy`.

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
    getNodes(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
    getNodes(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
    getNodes(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 to/from a tar file*

Description

Defunct by save_gs/load_gs

Usage

```
archive(G, file = tempfile())

unarchive(file, path = tempdir())
```

Arguments

`G`	a `GatingSet`
`file`	a `character` target/source archive file name
`path`	a `character` target folder that stores cdf file

`booleanFilter-class` *A class describing logical operation (& or |) of the reference populations*

Description

`booleanFilter` class inherits class `expressionFilter` and exists for the purpose of methods dispatching.

`booleanFilter` is a constructor from an expression

`char2booleanFilter` is a constructor from a character string

Usage

```
booleanFilter(expr, ...,
  filterId = "defaultBooleanFilter")

char2booleanFilter(expr, ...,
  filterId = "defaultBooleanFilter")
```

Arguments

`expr`	`expression` or `character`
`...`	further arguments to the expression
`filterId`	`character` identifier

See Also

`add` `GatingHierarchy`

`clone` *clone a GatingSet*

Description

clone a GatingSet

Arguments

`x`	A `GatingSet`
`...`	ncdfFile = NULL: see `clone.ncdfFlowSet`

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 `GatingSet`), 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`.

Examples

```
## Not run:
    #G is  a GatingSet
    G1<-clone(G)


## End(Not run)
```

`flowData` *Fetch or replace the flowData object associated with a GatingSet .*

Description

Accessor method that gets or replaces the flowset/ncdfFlowSet object in a GatingSet or GatingHierarchy

Arguments

`x`	A `GatingSet`
`value`	The replacement `flowSet` or `ncdfFlowSet` object

Details

Accessor method that sets or replaces the ncdfFlowSet object in the GatingSet or GatingHierarchy.

Value

the object with the new flowSet in place.

`flowJoWorkspace-class` *An R representation of a flowJo workspace.*

Description

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.

`options`: Object of class `"integer"`. The XML parsing options passed to `xmlTreeParse`.

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.par.init` *workspace version is parsed from xml node '/Workspace/version' in flowJo workspace and matched with this list to dispatch to the one of the three workspace parsers*

Description

workspace version is parsed from xml node '/Workspace/version' in flowJo workspace and matched with this list to dispatch to the one of the three workspace parsers

Usage

```
flowWorkspace.par.init()
```

`flowWorkspace.par.set` *flowWorkspace.par.set sets a set of parameters in the flowWorkspace package namespace.*

Description

flowWorkspace.par.set sets a set of parameters in the flowWorkspace package namespace.

flowWorkspace.par.get gets a set of parameters in the flowWorkspace package namespace.

Usage

```
flowWorkspace.par.set(name, value)

flowWorkspace.par.get(name = NULL)
```

Arguments

`value`	A named list of values to set for category name or a list of such lists if name is missing.
`name`	The name of a parameter category to get or set.

Details

It is currently used to add/remove the support for a specific flowJo versions (parsed from xml node '/Workspace/version' in flowJo workspace)

Examples

```
#get the flowJo versions currently supported
 old <- flowWorkspace.par.get("flowJo_versions")

 #add the new version
 old[["win"]] <- c(old[["win"]], "1.7")
 flowWorkspace.par.set("flowJo_versions", old)

 flowWorkspace.par.get("flowJo_versions")
```

`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.

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

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.

See Also

`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,path=d,name=1));
 gh <- G[[1]]
getPopStats(gh);
plotPopCV(gh)
 nodes <- getNodes(gh)
 thisNode <- nodes[4]
plotGate(gh,thisNode);
getGate(gh,thisNode);
getData(gh,thisNode)
```

`GatingSet` *constructors for GatingSet*

Description

construct object from xml workspace file and a list of sampleIDs

construct a gatingset with empty trees (just root node)

construct object from existing gating hierarchy(gating template) and flow data

---

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

Description

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

`[` subsets a `GatingSet` or `GatingSetList` using the familiar bracket notation

`[[` extract a `GatingHierarchy` object from a `GatingSet` or `GatingSetList`

Arguments

`x`	`GatingSet` or `GatingSetList`
`i`	`numeric` or `logical` or `character` used as sample index
`j`	not used
`drop`	not used
`...`	not used

Details

Objects stores a collection of GatingHierarchies and represent a group in a flowJo workspace. A GatingSet can have two “states”. After a call to parseWorkspace(...,execute=FALSE) , the workspace is imported but the data is not. Setting `execute` to `TRUE` is needed in order to load, transform, compensate, and gate the associated data. 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 executed. This mechanism is in place, largely for the purpose of speed when working with larger workspaces. It allows the use to load a workspace and subset desired samples before proceeding to load the data.

Slots

- `FCSPath`**:** Object of class `"character"`. A path to the fcs files associated with this GatingSet
- `data`**:** Object of class `"flowSet"`. flow data associated with this GatingSet
- `flag`**:** Object of class `"logical"`. A flag indicating whether the gates, transformations, and compensation matrices have been applied to data, or simply imported.
- `pointer`**:** Object of class `"externalptr"`. points to the gating hierarchy stored in C data structure.
- `guid`**:** Object of class `"character"`. the unique identifier for GatingSet object.

See Also

`GatingHierarchy` `flowJoWorkspace` `parseWorkspace`

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.

use `GatingSetList` constructor to create a GatingSetList from a list of GatingSet

Usage

```
GatingSetList(x, samples = NULL)
```

Arguments

`x`	`a list of GatingSet`
`samples`	`character` vector specifying the sample names. if NULL, the sample names are extracted from GatingSets

Details

Objects store a collection of GatingSets,which usually has the same gating trees and markers. Most GatingSets methods can be applied to GatingSetList.

See Also

`GatingSet` `GatingHierarchy`

Examples

```
## Not run:
    #load several GatingSets from disk
   gs_list<-lapply(list.files("../gs_toMerge",full=T) ,function(this_folder){
                    load_gs(this_folder)
                    })

   #gs_list is a list
    gs_groups <- merge(gs_list)
    #returns a list of GatingSetList objects
    gslist2 <- gs_groups[[2]]
   #gslist2 is a GatingSetList that contains multiple GatingSets and they share the same gating and data structure
    gslist2
    class(gslist2)
    sampleNames(gslist2)

    #reference a GatingSet by numeric index
    gslist2[[1]]
    #reference a GatingSet by character index
    gslist2[["30104.fcs"]]

    #loop through all GatingSets within GatingSetList
    lapply(gslist2,sampleNames)

    #subset a GatingSetList by [
    sampleNames(gslist2[c(4,1)])
    sampleNames(gslist2[c(1,4)])
    gslist2[c("30104.fcs")]

    #get flow data from it
    getData(gslist2)
    #get gated flow data from a particular popoulation (by numeric or character index)
    getData(gslist2,4)

    #extract the gates associated with one popoulation
    getGate(gslist2,"3+")
    getGate(gslist2,5)

    #extract the pheno data
    pData(gslist2[3:1])
    #modify the pheno data
    pd <- pData(gslist2)
    pd$id <- 1:nrow(pd)
    pData(gslist2) <- pd
    pData(gslist2[3:2])

    #plot the gate
    plotGate(gslist2[1:2],5,smooth=T)
    plotGate_labkey(gslist2[3:4],4,x="<APC Cy7-A>",y="<PE Tx RD-A>",smooth=T)

    #remove cerntain gates by loop through GatingSets
    getNodes(gslist2[[1]])
```

```
  lapply(gslist2,function(gs)Rm("Excl",gs))

  #extract the stats
  getPopStats(gslist2)
  #extract statistics by using getQAStats defined in QUALIFIER package
  res<-getQAStats(gslist2[c(4,2)],isMFI=F,isSpike=F,nslaves=1)

  #archive the GatingSetList
 save_gslist(gslist2, path ="~/rglab/workspace/flowIncubator/output/gslist",overwrite=T)
  gslist2 <- load_gslist(path ="~/rglab/workspace/flowIncubator/output/gslist")

  #convert GatingSetList into one GatingSet by rbind2
 gs_merged2 <- rbind2(gslist2,ncdfFile=path.expand(tempfile(tmpdir="~/rglab/workspace/flowIncubator/output/",
  gs_merged2

## End(Not run)
```

`getChannelMarker` *get channel and marker information from a `flowFrame` that matches to the given keyword*

Description

This function tries best to guess the flow parameter based on the keyword supplied by `name` It first does a complete word match(case insensitive) between `name` and flow channels and markers. If there are duplcated matches, throw the error. If no matches, it will try the partial match.

Usage

```
getChannelMarker(frm, name, ...)
```

Arguments

`frm`	`flowFrame` object
`name`	`character` the keyword to match
`...`	other arguments: not used.

Value

an one-row `data.frame` that contains "name"(i.e. channel) and "desc"(i.e. stained marker) columns.

`getCompensationMatrices` *Retrieve the compensation matrices from a GatingHierarchy*

Description

Retrieve the compensation matrices from a GatingHierarchy.

Arguments

`x`	A `GatingHierarchy` object.

Details

Return all the compensation matrices in a GatingHierarchy.

Value

A list of `matrix` representing the spillover matrix in `GatingHierarchy`

Examples

```
## Not run:
#Assume gh is a GatingHierarchy
  getCompensationMatrices(gh);

## End(Not run)
```

`getData` *get gated flow data from a GatingHierarchy/GatingSet/GatingSetList*

Description

get gated flow data from a GatingHierarchy/GatingSet/GatingSetList

Arguments

`obj`	A `GatingHierarchy`, `GatingSet` or `GatingSetList` object.
`y`	`character` the node name or full(/partial) gating path or `numeric` node index. If not specified, will return the complete flowFrame/flowSet at the root node.
`...`	arguments passed to ncdfFlow::[[

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. To update the data, use `flowData` method.

Value

A `flowFrame` object if `obj` is a GatingHierarchy. A `flowSet` or `ncdfFlowSet` if a `GatingSet`. A `ncdfFlowList` if a `GatingSetList`.

See Also

`flowData` `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.
    geData(G,"cd4")

    #gh is a GatingHierarchy
    getData(gh)

## End(Not run)
```

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

Description

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.

Usage

```
getFJWSubsetIndices(ws, key = NULL, value = NULL, group,
  requiregates = TRUE)
```

Arguments

`ws`	`flowJoWorkspace` object
`key`	`character` The name of the keyword.
`value`	`character` The value of the keyword.
`group`	`numeric` The group of samples to subset.
`requiregates`	`TRUE` or `FALSE`, specifying whether we include only samples that have gates attached or whether we include any sample in the workspace.

Details

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.

Value

A numeric vector of indices.

See Also

`parseWorkspace`

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

Description

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

return gate `y` for a given hierarchy (by index) Note that this index is ordered by regular sorting method

Arguments

`obj`	A `GatingHierrarchy` or `GatingSet`
`y`	A `character` the name or full(/partial) gating path of the node of interest. Or, a `numeric` index into the node list of nodes in the `GatingHierarchy` or `GatingSet`.

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`.

See Also

`getData` `getNodes`

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)
```

`getIndiceMat` *Return the single-cell matrix of 1/0 dichotomized expression*

Description

Return the single-cell matrix of 1/0 dichotomized expression

Usage

```
getIndiceMat(gh, y)
```

Arguments

gh	`GatingHierarchy` object
y	`character` node name

`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.

It adds the boolean gates and does the gating on the fly, and return the indices associated with that bool gate, and remove the bool gate the typical use case would be extracting any-cytokine-expressed cells

Arguments

`obj`	A `GatingHierarchy` representing a sample.
`y`	A `character` or `numeric` giving the name or full(/partial) gating path or index of the population / node of interest.
`y`	a quoted expression.

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.

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. 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.

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)
## Not run:

getIndices(gs,quote(4+/TNFa+|4+/IL2+))


## End(Not run)
```

`getKeywords` *Get Keywords*

Description

Retrieve keywords associated with a workspace

Arguments

`obj`	A `flowJoWorkspace`
`y`	`ccharacter` specifying the sample names

Details

Retrieve a list of keywords from a `flowJoWorkspace`

Value

A list of keyword - value pairs.

Examples

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

  getSamples(ws)
  getKeywords(ws,"CytoTrol_CytoTrol_1.fcs")
```

`getLoglevel` *get/set the log level*

Description

It is helpful sometime to get more detailed print out for the purpose of trouble shooting

Usage

```
getLoglevel()

setLoglevel(level = "none")
```

Arguments

`level`	a `character` that represents the log level , can be value of c("none", "GatingSet", "GatingHierarchy", "Population", "gate") default is "none" , which does not print any information from C parser.

Value

a character that represents the internal log level

Examples

```
getLoglevel()
setLoglevel("Population")
getLoglevel()
```

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

Description

`getNodes` returns a character vector of names of the nodes (populations) in the `GatingSet`.

Arguments

`x`	A `GatingSet` Assuming the gating hierarchy are identical within the `GatingSet`, the Gating tree of the first sample is used to query the node information.
`y`	A `character` the name or full(/partial) gating path of the population 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`	(Deprecated by 'path') A `logical` scalar to tell the method whether to return the full gating path or just terminal node name
`path`	A `character` or `numeric` scalar. when `numeric`, it specifies the fixed length of gating path (length 1 displays terminal name). When `character`, it can be either 'full' (full path, which is default) or 'auto' (display the shortest unique gating path from the bottom of gating tree).
`prefix`	A `character` scalar to tell the method whether to add internal node index as the prefix to the node name (only valid when 'path' is set to 1). the valid values are "auto", "none", "all".
`...`	Additional arguments.

Details

integer indices of nodes are based on regular order,so whenver need to map from character node name to integer node ID,make sure to use default order which is regular.

Value

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

Examples

```
## Not run:
    #G is a gating hierarchy
    getNodes(G, path = 1])#return node names (without prefix)
    getNodes(G, path = 1, prefix = "all"])#return node names with unqiue ID
    getNodes(G, path = 1, prefix = "auto"])#prepend unqiue ID as needed
    getNodes(G,path = "full")#return the full path
    getNodes(G,path = 2)#return the path as length of two
    getNodes(G,path = "auto)#automatically determine the length of path
    setNode(G,"L","lymph")

## 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`

Arguments

`obj`	A `GatingHierarchy`
`y`	a `character/numeric` the name or full(/partial) gating path or node indices of the node / population.
`...`	other arguments passed to getNodes methods

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.

See Also

`getNodes`

Examples

```
## Not run:
    #G is a gatinghierarchy
    #return the name of the parent of the fifth node in the hierarchy.
    getParent(G,getNodes(G[[1)[5])
    n<-getNodes(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` *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*

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

Arguments

`x`	A `GatingHierarchy` or `GatingSet`
`statistic`	`character` specifies the type of population statistics to extract. Either "freq" or "count" is currently supported.
`flowJo`	`logical` indicating whether the statistics come from FlowJo (if parsed from xml workspace) or from flowCore.
`...`	Additional arguments passed to getNodes

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 pouplation). getProp returns a population frequency `numeric`. getTotal returns a `numeric` value of the total number of elements in the population.

See Also

`getNodes`

Examples

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


## End(Not run)
```

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

Description

Return the sample name

Arguments

`x`	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.

Examples

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

## 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

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.

See Also

`flowJoWorkspace-class` `openWorkspace`

Examples

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

## End(Not run)
```

`getSamples` *Get a list of samples from a flowJo workspace*

Description

Return a data frame of samples contained in a flowJo workspace

Arguments

`x`	A `flowJoWorkspace`

Details

Returns a `data.frame` of samples in the `flowJoWorkspace`, including their `sampleID`, `name`, and `compID` (compensation matrix ID).

Value

A `data.frame` with columns `sampleID`, `name`, and `compID` if `x` is a `flowJoWorkspace`.

Examples

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

## End(Not run)
```

`getSingleCellExpression` *Return the cell events data that express in any of the single populations defined in* `y`

Description

Returns a list of matrix containing the events that expressed in any one of the populations efined in `y`

Arguments

`x`	A `GatingSet` or `GatingSetList` object .
`nodes`	`character` vector specifying different cell populations
`map`	mapping node names (as specified in the gating hierarchy of the gating set) to channel names (as specified in either the `desc` or `name` columns of the parameters of the associated `flowFrame`s in the `GatingSet`).

Value

A `list` of `numerci matrices`

Author(s)

Mike Jiang <`wjiang2@fhcrc.org`>

See Also

`getIndices` `getProp` `getPopStats`

Examples

```
## Not run:
  #G is a GatingSet
geData(G,3)
res <- getSingleCellExpression(gs[1], c("4+/TNFa+", "4+/IL2+"))
res[[1]]
res <- getSingleCellExpression(gs[1], c("4+/TNFa+", "4+/IL2+") , list("4+/TNFa+" = "TNFa", "4+/IL2+" = "IL2"))

## End(Not run)
```

`getTransformations` *Return a list of transformations or a transformation in a GatingHierarchy*

Description

Return a list of all the transformations or a transformation in a GatingHierarchy

Arguments

`x`	A `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. this method currently is used convert transformation funtion from c++ to R mainly for transforming range info

Value

lists of functions, with each element of the list representing a transformation applied to a specific channel/parameter of a sample.

Examples

```
## Not run:
#Assume gh is a GatingHierarchy
getTransformations(gh);

## End(Not run)
```

`isNcdf` *determine the flow data associated with a Gating Hiearchy is based on ‘ncdfFlowSet‘ or ‘flowSet‘*

Description

determine the flow data associated with a Gating Hiearchy is based on ‘ncdfFlowSet‘ or ‘flowSet‘

Usage

```
isNcdf(x)
```

Arguments

`x`	`GatingHiearchy` object

Value

`logical`

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

Description

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

Arguments

`object`	`GatingHierarchy` or `GatingSet` or `GatingSetList`
`keyword`	`character` specifying keyword name. When `missing`, extract all keywords.

Details

See `keyword` in Package ‘flowCore’

See Also

`keyword-methods`

Examples

```
## Not run:
      #get all the keywords from all samples
      keyword(G)
      #get all the keywords from one sample
      keyword(G[[1]])
      #get single keyword from all samples
      keyword(G, "FILENAME")
      #get single keyword from one sample
      keyword(G[[1, "FILENAME")

## End(Not run)
```

`lapply` *apply* `FUN` *to each sample (i.e.* `GatingHierarchy`*)*

Description

sample names are used for names of the returned list

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

Description

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

`mkformula` *make a formula from a character vector*

Description

construct a valid formula to be used by flowViz::xyplot

Usage

```
mkformula(dims, isChar = FALSE)
```

Arguments

`dims`	a `character` vector that contains `y` , x axis, if it is unnamed, then treated as the order of c(y,x)
`isChar`	`logical` flag indicating whehter to return a formula or a pasted string

Value

when `isChar` is TRUE, return a character, otherwise coerce it as a `formula`

Examples

```
all.equal(mkformula(c("SSC-A", "FSC-A")),SSC-A ~ FSC-A)#unamed vecotr
all.equal(mkformula(c(x = "SSC-A", y = "FSC-A")),FSC-A ~ SSC-A)#named vector
```

`ncFlowSet` *Fetch the flowData object associated with a GatingSet .*

Description

Deprecated by `flowData` method

Deprecated by `flowData` method

`openWorkspace` *Open/Close a flowJo workspace*

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.

Arguments

`file`	Full path to the XML flowJo workspace file.
`options`	xml parsing options passed to `xmlTreeParse`
`...`	other arguments passed to `xmlTreeParse`
`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 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

a `flowJoWorkspace` object.

Examples

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

## 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 `recompute()` is made on the `GatingHierarchy` objects in the `GatingSet`.

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.
`subset`	`numeric` vector specifying the subset of samples in a group to import. Or a `character` specifying the FCS filenames to be imported.
`requiregates`	`logical` Should samples that have no gates be included?
`includeGates`	`logical` Should gates be imported, or just the data with compensation and transformation?
`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.
`...`	• sampNloc="keyword": a `character` scalar indicating where to get sampleName(or FCS filename) within xml workspace. It is either from "keyword" or "sampleNode". • compensation=NULL: a `matrix` that allow the customized compensation matrix to be used instead of the one specified in flowJo workspace. • options=0: a `integer` option passed to `xmlTreeParse`

- ignore.case a `logical` flag indicates whether the colnames(channel names) matching needs to be case sensitive (e.g. compensation, gating..)
- extend_val `numeric` the threshold that determine wether the gates need to be extended. default is 0. It is triggered when gate coordinates are below this value.
- extend_to `numeric` the value that gate coordinates are extended to. Default is -4000. Usually this value will be automatically detected according to the real data range. But when the gates needs to be extended without loading the raw data (i.e. `execute` is set to FALSE), then this hard-coded value is used.
- ...: Additional arguments to be passed to read.ncdfFlowSet or read.flowSet.

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

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.

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 directo
#G is a GatingSet.
G1<-parseWorkspace(ws)
#G1 is a GatingSet.

## End(Not run)
```

`pData` *read/set pDatа of flow data associated with* `GatingSet` *or* `GatingSetList`

Description

Accessor method that gets or replaces the pData of the flowset/ncdfFlowSet object in a GatingSet or GatingSetList

Arguments

`object`	`GatingSet` or `GatingSetList`
`value`	`data.frame` The replacement of pData for `flowSet` or `ncdfFlowSet` object

Value

a `data.frame`

`plot` *plot a gating tree*

Description

Plot a tree/graph representing the GatingHierarchy

plot a subgraph

Arguments

`x`	`GatingHierarchy` or `GatingSet`. If `GatingSet`, the first sample will be used to extract gating tree.
`y`	`missing`.
`...`	other arguments: • dir: `character` Default is NULL, which render the gating tree in regular R plot device. Otherwise it specifies a folder where the gating tree is output to a svg image with some interactivity (e.g. when click on each node, the actual gates will be displayed) This interactivity currently only works when the svg is rendered within a HTML webpage (e.g. generated as knitr report) • svg.par: a `list` of parameters passed to `devSVGTips` • png.par: a `list` of parameters passed to `png` • 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.

Examples

```
## Not run: 
 #gs is a GatingSet
 plot(gs) # the same as plot(gs[[1]])
 #plot a substree rooted from CD4
 plot(gs, "CD4")

 # output as svg format with onclick event of each node displaying the gate(saved as png files) within pop-up window
 plot(gs[[1]], dir = "myFolder")

 # customize the size of svg and png files
 plot(gh, dir = "myFolder", svg.par =list(width = 7, height = 7), png.par = list(width = 200, height = 200)

 #each node link to multiple gates across samples
 #here is the example code showing how to embed
 #the svg output within knitr quick report

 #+ eval=T, results = "asis", message = F
 svgFile <- plot(gs, dir = ./svg, svg.par =list(width = 7, height = 7), png.par = list(width = 400, height = 400 ))
 cat("<embed src=", svgFile, " type=image/svg+xml />", sep = "")

## End(Not run)
```

---

plotGate

Plot gates and associated cell population contained in a `GatingHierarchy` *or* `GatingSet`

Description

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.

Arguments

`x`	`GatingSet` or `GatingHierarchy`object
`y`	`character` the node name or full(/partial) gating path or `numeric` representing the node index in the `GatingHierarchy`. or `missing` which will plot all gates and one gate per page. It is useful for generating plots in a multi-page pdf. Nodes can be accessed with `getNodes`.

`...`

- bool `logical` specifying whether to plot boolean gates.
- arrange.main `character` The title of the main page of the plot. Default is the sample name. Only valid when `x` is GatingHierarchy
- arrange `logical` indicating whether to arrange different populations/nodes on the same page via `grid.arrange` call.
- merge `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;
- projections `list` of character vectors used to customize x,y axis. By default, the x,y axis are determined by the respective gate parameters. The elements of the list are named by the population name or path (see `y`). Each element is a pair of named character specifying the channel name(or marker name) for x, y axis. Short form of channel or marker names (e.g. "APC" or "CD3") can be used as long as they can be uniquely matched to the dimentions of flow data. For example, projections = list("lymph" = c(x = "SSC-A", y = "FSC-A"), "CD3" = c(x = "CD3", y = "SSC-A"))
- par.settings `list` of graphical parameters passed to `lattice`;
- gpar `list` of grid parameters passed to `grid.layout`;
- lattice `logical` deprecated;
- 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.
- cond `character` the conditioning variable to be passed to lattice plot.
- overlay `numeric` scalar indicating the index of a gate/populationwithin 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.
- default.y `character` specifiying y channel for xyplot when plotting a 1d gate. Default is "SSC-A".
- type `character` either "xyplot" or "densityplot". Default is "xyplot"
- fitGate used to disable behavior of plotting the gate region in 1d densityplot
- strip `ligcal` specifies whether to show pop name in strip box,only valid when x is `GatingHierarchy`
- marker.only `ligcal` specifies whether to show both channel and marker names
- xlim, ylim `character` can be either "instrument" or "data" which determines the x, y axis scale either by instrument measurement range or the actual data range. or `numeric` which specifies customized range.
- ...
  path A `character` or `numeric` scalar passed to getNodes method (used to control how the gating/node path is displayed)
  ... The other additional arguments to be passed to xyplot.

Value

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

References

http://www.rglab.org/

Examples

```
## Not run:
projections <- list("cd3" = c(x = "cd3", y = "AViD")
                    , "cd4" = c(x = "cd8", y = "cd4")
                    , "cd4/IL2" = c(x = "IL2", y = "IFNg")
                    , "cd4/IFNg" = c(x = "IL2", y = "IFNg")
                )
plotGate(gh, c("cd3", "cd4", "cd4/IL2", "cd4/IFNg"), path = "auto", projections = projections, gpar = c(nrow = 2))


## End(Not run)
## 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.

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.

See Also

`getPopStats`

Examples

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

## End(Not run)
```

`sampleNames` *Get/update sample names in a GatingSet*

Description

Return a sample names contained in a GatingSet

Arguments

`object`	or a `GatingSet`
`value`	`character` new sample names

Details

The sample names comes from pdata of fs.

Value

A character vector of sample names

Examples

```
## Not run:
        #G is  a GatingSet
        sampleNames(G)

## End(Not run)
```

`save_gs` *save/load a GatingSet/GatingSetList to/from disk.*

Description

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

Usage

```
save_gs(G, path, overwrite = FALSE,
  cdf = c("copy", "move", "skip", "symlink", "link"),
  type = c("binary", "text", "xml"), ...)

load_gs(path)

save_gslist(gslist, path, ...)

load_gslist(path)
```

Arguments

`G`	A `GatingSet`
`gslist`	A `GatingSetList`
`path`	A character scalar giving the path to save/load the GatingSet to/from.
`overwrite`	A logical scalar specifying whether to overwrite the existing folder.
`cdf`	a character scalar. The valid options are :"copy","move","skip","symlink","link" specifying what to do with the cdf data file. Sometime it is more efficient to move or create a link of the existing cdf file to the archived folder. It is useful to "skip" archiving cdf file if raw data has not been changed.
`type`	a character scalar. The valid options are :"binary","text","xml" specifying format to store tree structure. default is "binary", which is smaller and faster but machine-dependent. use "text" or "xml" for cross-platform data interchange (experimental at this stage, may or maynot work).
`...`	other arguments: not used.

Value

`load_gs` returns a GatingSet object `load_gslist` returns a GatingSetList object

See Also

`GatingSet-class`,`GatingSetList-class`

Examples

```
## Not run:
#G is a GatingSet
save_gs(G,path="tempFolder")
G1<-load_gs(path="tempFolder")

#G is a GatingSet

save_gslist(gslist1,path="tempFolder")
gslist2<-load_gslist(path="tempFolder")

## End(Not run)
```

`setNode` *Update the name of one node in a gating hierarchy/GatingSet.*

Description

`setNode` update the name of one node in a gating hierarchy/GatingSet.

hide/unhide a node

Arguments

`value`	A `character` the name of the node.
`x`	`GatingHierarchy` object
`y`	`numeric` node index
`value`	`logical` whether to hide a node

Examples

```
## Not run:
    #G is a gating hierarchy
    getNodes(G[[1]])#return node names
    setNode(G,"L","lymph")

## End(Not run)
## Not run:
     setNode(gh, 4, FALSE) # hide a node
     setNode(gh, 4, TRUE) # unhide a node

## End(Not run)
```

Index