

Package ‘flowWorkspace’

April 23, 2016

Type Package

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

Version 3.16.16

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, XML, tools,gridExtra,Rgraphviz ,data.table ,dplyr ,latticeExtra ,Rcpp ,RColorBrewer ,stringr ,flowUtils ,jsonlite

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' 'RcppExports.R' 'merge-GatingSet.R' 'utils.R' 'read.gatingML.cytobank.R' 'graphGML-methods.R' 'zzz.R'

Depends R (>= 2.16.0),flowCore(>= 1.36.4),flowViz(>= 1.29.27),ncdfFlow(>= 2.16.1),gridExtra

biocViews FlowCytometry, DataImport, Preprocessing, DataRepresentation

Suggests testthat ,flowWorkspaceData ,RSVGTipsDevice ,knitr

LinkingTo Rcpp, BH(>= 1.60.0-1)

VignetteBuilder knitr

SystemRequirements xml2, GNU make

NeedsCompilation yes

R topics documented:

flowWorkspace-package	3
add,GatingSet,list-method	4
archive	7
booleanFilter-class	8
clone	8
compare.counts	9
compensate,flowSet,graphGML-method	10
constructTree	11
extend	11
filterObject,rectangleGate-method	12
flowData,GatingSet-method	13
flowJo.fasinh	14
flowJoTrans	14
flowJoWorkspace-class	15
flowWorkspace.par.init	16
flowWorkspace.par.set	16
gating	17
GatingHierarchy-class	17
GatingSet,character,character-method	18
GatingSet-class	19
GatingSetList-class	21
getChildren,graphGML,character-method	23
getCompensationMatrices,GatingHierarchy-method	23
getCompensationMatrices,graphGML-method	24
getData,GatingHierarchy,missing-method	25
getFJWSubsetIndices	26
getGate,GatingHierarchy,character-method	27
getGate,graphGML,character-method	28
getIndiceMat	28
getIndices,GatingHierarchy,character-method	29
getIndices,GatingSet,name-method	30
getKeywords,flowJoWorkspace,character-method	31
getLogLevel	32
getNodes,GatingSet-method	32
getNodes,graphGML-method	33
getParent,GatingSet,character-method	34
getParent,graphGML,character-method	35
getProp,GatingHierarchy,character-method	36
getSampleGroups,flowJoWorkspace-method	37
getSamples,flowJoWorkspace-method	38
getSingleCellExpression,GatingSet,character-method	39
getTransformations,GatingHierarchy-method	40

getTransformations,graphGML-method	41
graphGML-class	42
insertGate	42
isNcdf	43
keyword,GatingHierarchy,character-method	43
lapply,GatingSet-method	45
length,GatingSet-method	45
matchPath	46
mkformula	46
ncFlowSet	47
openWorkspace,character-method	47
parse.gateInfo	48
parse.gatingML	48
parseWorkspace,flowJoWorkspace-method	49
pData,GatingHierarchy-method	52
plot,GatingSet,missing-method	52
plot,graphGML,missing-method	54
plotGate	55
plotPopCV,GatingHierarchy-method	57
prettyAxis	58
read.gatingML.cytobank	59
recompute,GatingSet-method	59
sampleNames,GatingHierarchy-method	60
save_gs	61
setGate,GatingHierarchy,character,filter-method	62
setNode,GatingHierarchy,character,character-method	63
show,graphGML-method	64
subset.GatingSet	64
transform,GatingSet-method	65
updateChannels	66
Index	67

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,GatingSet,list-method

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.

Usage

```
## S4 method for signature 'GatingSet,list'  
add(wf, action, ...)
```

```
## S4 method for signature 'GatingSetList,list'  
add(wf, action, ...)
```

```
## S4 method for signature 'GatingSet,filtersList'  
add(wf, action, ...)
```

```
## S4 method for signature 'GatingSet,filterList'
```

```

add(wf, action, validityCheck = TRUE, ...)

## S4 method for signature 'GatingSetList,filterList'
add(wf, action, ...)

## S4 method for signature 'GatingSetList,filtersList'
add(wf, action, ...)

## S4 method for signature 'GatingSet,filter'
add(wf, action, ...)

## S4 method for signature 'GatingSet,filters'
add(wf, action, ...)

## S4 method for signature 'GatingSetList,filter'
add(wf, action, ...)

## S4 method for signature 'GatingSetList,filters'
add(wf, action, ...)

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

## S4 method for signature 'GatingHierarchy,filters'
add(wf, action, names = NULL, ...)

## S4 method for signature 'GatingHierarchy,quadGate'
add(wf, action, names = NULL, ...)

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

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

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

```

Arguments

wf	A GatingHierrarchy or GatingSet
action	A filter or a list of filters to be added to the GatingHierarchy or GatingSet.
...	some other arguments to specify how the gates are added to the gating tree. <ul style="list-style-type: none"> 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
	<ul style="list-style-type: none"> • 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.
validityCheck	logical whether to check the consistency of tree structure across samples. default is TRUE. Can be turned off when speed is preferred to the robustness.
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.
symbol	A character identifies the population node in a GatingHierarchy or GatingSet to remove
envir	A GatingHierarchy or GatingSet
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.

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
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
getNodeIDs(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)
getNodeIDs(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")
```

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

Arguments

expr	expression or character
...	further arguments to the expression
filterId	character identifier
object	booleanFilter

See Also

[add GatingHierarchy](#)

clone *clone a GatingSet*

Description

clone a GatingSet

Usage

```
clone(x, ...)
```


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)
```

compare.counts	<i>compare the counts to cytobank's exported csv so that the parsing result can be verified.</i>
----------------	--

Description

compare the counts to cytobank's exported csv so that the parsing result can be verified.

Usage

```
compare.counts(gs, file, id.vars = c("FCS Filename", "population"))
```

Arguments

gs parsed GatingSet
 file the stats file (contains the populatio counts) exported from cytobank.
 id.vars either "population" or "FCS filename" that tells whether the stats file format is one population per row or FCS file per row.

Value

a data.table (in long format) that contains the counts from openCyto and Cytobank side by side.

Examples

```
## Not run:
xmlfile <- system.file("extdata/cytotrol_tcell_cytobank.xml", package = "flowWorkspace")
fcsFiles <- list.files(pattern = "CytoTrol", system.file("extdata", package = "flowWorkspaceData"), full = T)
gs <- parse.gatingML(xmlfile, fcsFiles)
## verify the stats are correct
statsfile <- system.file("extdata/cytotrol_tcell_cytobank_counts.csv", package = "flowWorkspace")
dt_merged <- compare.counts(gs, statsfile, id.vars = "population")
all.equal(dt_merged[, count.x], dt_merged[, count.y], tol = 5e-4)

## End(Not run)
```

compensate,flowSet,graphGML-method

*compensate a flowSet based on the compensation information stored
in graphGML object*

Description

compensate a flowSet based on the compensation information stored in graphGML object

Usage

```
## S4 method for signature 'flowSet,graphGML'
compensate(x, spillover, ...)
```

Arguments

x	flowSet
spillover	graphGML
...	unused.

Value

compensated flowSet

constructTree	<i>Reconstruct the population tree from the GateSets</i>
---------------	--

Description

Reconstruct the population tree from the GateSets

Usage

```
constructTree(flowEnv, gateInfo)
```

Arguments

flowEnv	the environment contains the elements parsed by read.gatingML function
gateInfo	the data.frame contains the gate name, fcs filename parsed by parse.gateInfo function

Value

a graphNEL represent the population tree. The gate and population name are stored as nodeData in each node.

extend	<i>extend the gate to the minimum and maximum limit of both dimensions based on the bounding information. It is equivalent to the behavior of shifting the off-scale boundary events into the gate boundary that is described in bounding transformation section of gatingML standard. The advantage of extending gates instead of shifting data are two folds: 1. Avoid the extra computation each time applying or plotting the gates 2. Avoid changing the data distribution caused by adding the gates</i>
--------	--

Description

extend the gate to the minimum and maximum limit of both dimensions based on the bounding information. It is equivalent to the behavior of shifting the off-scale boundary events into the gate boundary that is described in bounding transformation section of gatingML standard. The advantage of extending gates instead of shifting data are two folds: 1. Avoid the extra computation each time applying or plotting the gates 2. Avoid changing the data distribution caused by adding the gates

Usage

```

extend(gate, bound, data.range = NULL, plot = FALSE)

## S3 method for class 'polygonGate'
extend(gate, bound, data.range = NULL, plot = FALSE)

## S3 method for class 'rectangleGate'
extend(gate, ...)

## S3 method for class 'ellipsoidGate'
extend(gate, ...)

```

Arguments

gate	a flowCore filter/gate
bound	numeric matrix representing the bounding information parsed from gatingML. Each row corresponds to a channel. rownames should be the channel names. colnames should be c("min", "max")
data.range	numeric matrix specifying the data limits of each channel. It is used to set the extended value of vertices and must have the same structure as 'bound'. when it is not supplied, c(-.Machine\$integer.max, -.Machine\$integer.max) is used.
plot	whether to plot the extended polygon.
...	other arguments

```

filterObject,rectangleGate-method
      convert flowCore filter to a list

```

Description

It convert the flowCore gate to a list whose structure can be understood by underlying c++ data structure.

Usage

```

## S4 method for signature 'rectangleGate'
filterObject(x)

## S4 method for signature 'polygonGate'
filterObject(x)

## S4 method for signature 'booleanFilter'
filterObject(x)

## S4 method for signature 'ellipsoidGate'
filterObject(x)

```

Arguments

x filter a flowCore gate. Currently supported gates are: "rectangleGate", "polygonGate", "ellipsoidGate" and "booleanFilter"

Value

a list

flowData,GatingSet-method

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

Usage

```
## S4 method for signature 'GatingSet'
flowData(x)

## S4 replacement method for signature 'GatingSet'
flowData(x) <- value
```

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.

flowJo.fasinh	<i>inverse hyperbolic sine transform function</i>
---------------	---

Description

hyperbolic sine/inverse hyperbolic sine (flowJo-version) transform function constructor

Usage

```
flowJo.fasinh(m = 4, t = 12000, a = 0.7, length = 256)
```

```
flowJo.fsinh(m = 4, t = 12000, a = 0.7, length = 256)
```

Arguments

m	numeric the full width of the transformed display in asymptotic decades
t	numeric the maximum value of input data
a	numeric Additional negative range to be included in the display in asymptotic decades
length	numeric the maximum value of transformed data

Value

fasinh/fsinh transform function

Examples

```
trans <- flowJo.fasinh()
data.raw <- c(1,1e2,1e3)
data.trans <- trans(data.raw)
data.trans
```

```
inverse.trans <- flowJo.fsinh()
inverse.trans(data.trans)
```

flowJoTrans	<i>construct the flowJo-type biexponential transformation function</i>
-------------	--

Description

Normally it was parsed from flowJo xml workspace. This function provides the alternate way to construct the flowJo version of logicle transformation function within R.

Usage

```
flowJoTrans(channelRange = 4096, maxValue = 262144, pos = 4.5, neg = 0,
  widthBasis = -10, inverse = FALSE)
```

Arguments

channelRange	numeric	the maximum value of transformed data
maxValue	numeric	the maximum value of input data
pos	numeric	the full width of the transformed display in asymptotic decades
neg	numeric	Additional negative range to be included in the display in asymptotic decades
widthBasis	numeric	unkown.
inverse	logical	whether to return the inverse transformation function.

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);
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

name The name of a parameter category to get or set.

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

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")
```

gating

*Apply the gatingML graph to a GatingSet***Description**

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

Usage

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

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

Arguments

x	graphGML
y	GatingSet
...	other arguments

Value

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

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,character,character-method
constructors for GatingSet

Description

construct object from xml workspace file and a list of sampleIDs (not intended to be called by user.)

construct a gatingset with empty trees (just root node)

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

Usage

```
## S4 method for signature 'character,character'
GatingSet(x, y, guids, includeGates = FALSE,
  sampNloc = "keyword", xmlParserOption, wsType)

## S4 method for signature 'flowSet,ANY'
GatingSet(x)
```

```
## S4 method for signature 'GatingHierarchy,character'
GatingSet(x, y, path = ".", ...)
```

Arguments

x	character or flowSet or GatingHierarchy
y	character or missing
guids	character vectors to uniquely identify each sample (Sometime FCS file names alone may not be unique)
includeGates	logical whether to parse the gates or just simply extract the flowJo stats
sampNloc	character scalar indicating where to get sampleName(or FCS filename) within xml workspace. It is either from "keyword" or "sampleNode".
xmlParserOption	integer option passed to xmlTreeParse
wsType	character workspace type, can be value of "win", "macII", "vX", "macIII".
path	character specifies the path to the flow data (FCS files)
...	other arguments. see parseWorkspace

Examples

```
## Not run:
#fdata could be a flowSet or ncdfFlowSet
gs <- GatingSet(fdata)

## End(Not run)
```

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

Usage

```
## S4 method for signature 'GatingSet,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'GatingSet,numeric'
x[[i, j, ...]]

## S4 method for signature 'GatingSetList,ANY'
x[i, j, ..., drop = TRUE]
```

Arguments

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

`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.

`axis`: Object of class "list". stores the axis information used for `plotGate`.

`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)
```

```
## S4 method for signature 'GatingSetList,missing'
rbind2(x, y = "missing", ...)
```

Arguments

x	a list of GatingSet
samples	character vector specifying the sample names. if NULL, the sample names are extracted from GatingSets
y	missing not used.
...	other arguments passed to rbind2 method for ncdFlowList

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(gslis2)

#reference a GatingSet by numeric index
gslis2[[1]]
#reference a GatingSet by character index
gslis2[["30104.fcs"]]

#loop through all GatingSets within GatingSetList
lapply(gslis2,sampleNames)

#subset a GatingSetList by [
sampleNames(gslis2[c(4,1)])
sampleNames(gslis2[c(1,4)])
gslis2[c("30104.fcs")]

#get flow data from it
getData(gslis2)
#get gated flow data from a particular popoulation
getData(gslis2, "3+")

#extract the gates associated with one popoulation
getGate(gslis2,"3+")
getGate(gslis2,5)

#extract the pheno data
pData(gslis2[3:1])
#modify the pheno data
pd <- pData(gslis2)
pd$id <- 1:nrow(pd)
pData(gslis2) <- pd
pData(gslis2[3:2])

#plot the gate
plotGate(gslis2[1:2],5,smooth=T)
plotGate_labkey(gslis2[3:4],4,x="<APC Cy7-A>",y="<PE Tx RD-A>",smooth=T)

#remove cerntain gates by loop through GatingSets
getNodes(gslis2[[1]])
lapply(gslis2,function(gs)Rm("Excl",gs))

#extract the stats
getPopStats(gslis2)
#extract statistics by using getQASStats defined in QUALIFIER package
res<-getQASStats(gslis2[c(4,2)],isMFI=F,isSpike=F,nslaves=1)

#archive the GatingSetList
save_gslis2(gslis2, path = "~/rglab/workspace/flowIncubator/output/gslis2",overwrite=T)
gslis2 <- load_gslis2(path = "~/rglab/workspace/flowIncubator/output/gslis2")

#convert GatingSetList into one GatingSet by rbind2
gs_merged2 <- rbind2(gslis2,ncdfFile=path.expand(tempfile(tmpdir="~/rglab/workspace/flowIncubator/output/"
gs_merged2

```

```
## End(Not run)
```

getChildren,graphGML,character-method
get children nodes

Description

get children nodes

Usage

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

Arguments

obj	graphGML
y	character parent node path

Examples

```
## Not run:  
xmlfile <- system.file("extdata/cytotrol_tcell_cytobank.xml", package = "flowWorkspace")  
g <- read.gatingML.cytobank(xmlfile)  
getChildren(g, "GateSet_694130")  
getParent(g, "GateSet_694130")  
  
## End(Not run)
```

getCompensationMatrices,GatingHierarchy-method
Retrieve the compensation matrices from a GatingHierarchy

Description

Retrieve the compensation matrices from a GatingHierarchy.

Usage

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

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)
```

getCompensationMatrices,graphGML-method
Extract compensation from graphGML object.

Description

Extract compensation from graphGML object.

Usage

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

Arguments

x graphGML

Value

compensation object or "FCS" when compensation comes from FCS keywords

getData,GatingHierarchy,missing-method
get gated flow data from a GatingHierarchy/GatingSet/GatingSetList

Description

get gated flow data from a GatingHierarchy/GatingSet/GatingSetList

Usage

```
## S4 method for signature 'GatingHierarchy,missing'
getData(obj, y, ...)

## S4 method for signature 'GatingHierarchy,character'
getData(obj, y, ...)

## S4 method for signature 'GatingSet,missing'
getData(obj, y, ...)

## S4 method for signature 'GatingSet,character'
getData(obj, y, ...)

## S4 method for signature 'GatingSetList,ANY'
getData(obj, y, ...)
```

Arguments

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

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](#) [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	<i>Fetch the indices for a subset of samples in a flowJo workspace, based on a keyword value pair</i>
---------------------	---

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,GatingHierarchy,character-method`*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.

Usage

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

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

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

Arguments

<code>obj</code>	A GatingHierrarchy or GatingSet
<code>y</code>	A character the name or full(/partial) gating path of the node of interest.

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.

See Also[getData](#) [getNode](#)s

Examples

```
## Not run: #gh is a GatingHierarchy
  getGate(gh, "CD3") #return the gate for the fifth node in the tree, but fetch it by name.
  #G is a GatingSet
  getGate(G, "CD3") #return a list of gates for the fifth node in each tree

## End(Not run)
```

```
getGate,graphGML,character-method
      get gate from the node
```

Description

get gate from the node

Usage

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

Arguments

obj	graphGML
y	character node path

```
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,GatingHierarchy,character-method

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

Description

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

Usage

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

Arguments

obj	A GatingHierarchy representing a sample.
y	A character giving the name or full(/partial) gating path 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.

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

[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)
```

```
getIndices,GatingSet,name-method
```

routine to return the indices by specify boolean combination of reference nodes:

Description

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

Usage

```
## S4 method for signature 'GatingSet,name'
getIndices(obj, y)
```

Arguments

obj	GatingSet
y	a quoted expression.

Examples

```
## Not run:

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

## End(Not run)
```

getKeywords,flowJoWorkspace,character-method
Get Keywords

Description

Retrieve keywords associated with a workspace

Usage

```
## S4 method for signature 'flowJoWorkspace,character'  
getKeywords(obj, y, ...)
```

```
## S4 method for signature 'flowJoWorkspace,numeric'  
getKeywords(obj, y, ...)
```

Arguments

obj	A flowJoWorkspace
y	ccharacter or numeric specifying the sample name or sample ID
...	other arguments sampNloc a character the location where the sample name is specified. See parseWorkspace for more details.

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)  
res <- try(getKeywords(ws,"CytoTrol_CytoTrol_1.fcs"), silent = TRUE)  
print(res[[1]])  
getKeywords(ws, 1)
```

getLogLevel	<i>get/set the log level</i>
-------------	------------------------------

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,GatingSet-method

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.

Usage

```
## S4 method for signature 'GatingSet'
getNodes(x, y = NULL, order = "regular",
  path = "full", showHidden = FALSE, ...)
```


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 not used.
order	order=c("regular", "tsort", "bfs") returns the nodes in regular, topological or breadth-first sort order. "regular" is default.
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).
showHidden	logical whether to include the hidden nodes
...	Additional arguments.

Details

integer indices of nodes are based on regular order,so whenever 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 = "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)
```

getNodes,graphGML-method

get nodes from graphGML object

Description

get nodes from graphGML object

Usage

```
## S4 method for signature 'graphGML'
getNodes(x, y, order = c("default", "bfs", "dfs",
  "tsort"), only.names = TRUE)
```

Arguments

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

Value

It returns the node names and population names by default. Or return the entire nodeData associated with each node.

Examples

```
## Not run:
xmlfile <- system.file("extdata/cytotrol_tcell_cytobank.xml", package = "flowWorkspace")
g <- read.gatingML.cytobank(xmlfile)
getNode(g)
getNode(gt, only.names = FALSE)

## End(Not run)
```

getParent,GatingSet,character-method

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 'GatingSet,character'
getParent(obj, y, ...)

## S4 method for signature 'GatingSet,character'
getChildren(obj, y, showHidden = TRUE, ...)
```

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 getNode methods
showHidden	logical whether to include the hidden children nodes.

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

[getNode](#)s

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)
```

getParent,graphGML,character-method
get parent nodes

Description

get parent nodes

Usage

```
## S4 method for signature 'graphGML,character'
getParent(obj, y)
```

Arguments

obj	graphGML
y	character child node path

getProp,GatingHierarchy,character-method

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

Usage

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

## S4 method for signature 'GatingHierarchy,character'
getTotal(x, y, flowJo = FALSE)

## S4 method for signature 'GatingHierarchy'
getPopStats(x, path = "auto", ...)

## S4 method for signature 'GatingSet'
getPopStats(x, statistic = c("freq", "count"),
  flowJo = FALSE, subpopulations = NULL, format = c("long", "wide"),
  path = "auto", ...)

## S4 method for signature 'GatingSetList'
getPopStats(x, format = c("long", "wide"), ...)
```

Arguments

x	A GatingHierarchy or GatingSet
y	character node name or path
flowJo	logical indicating whether the statistics come from FlowJo (if parsed from xml workspace) or from flowCore.
path	character see getNode s
...	Additional arguments passed to getNode s
statistic	character specifies the type of population statistics to extract.(only valid when format is "wide"). Either "freq" or "count" is currently supported.
subpopulations	character vector to specify a subset of populations to return. (only valid when format is "long")

format character value of c("wide", "long") specifying whether to organize the output in long or wide format

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.

See Also

[getNode](#)s

Examples

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

  #gs is a GatingSet
  getPopStats(gs)
  #optionally output in long format as a data.table
  getPopStats(gs, format = "long", path = "auto")
  #only get stats for a subset of populations
  getPopStats(gs, format = "long", subpopulations = getNode(s)[4:6])

## End(Not run)
```

getSampleGroups, flowJoWorkspace-method

Get a table of sample groups from a flowJo workspace

Description

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

Usage

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

Arguments

x A flowJoWorkspace object.

Details

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

Value

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

See Also

[flowJoWorkspace-class](#) [openWorkspace](#)

Examples

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

## End(Not run)
```

getSamples, flowJoWorkspace-method

Get a list of samples from a flowJo workspace

Description

Return a data frame of samples contained in a flowJo workspace

Usage

```
## S4 method for signature 'flowJoWorkspace'
getSamples(x, sampNloc = "keyword")
```

Arguments

x A flowJoWorkspace
 sampNloc character either "keyword" or "sampleNode". see [parseWorkspace](#)

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, GatingSet, character-method

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

Usage

```
## S4 method for signature 'GatingSet,character'
getSingleCellExpression(x, nodes,
  other.markers = NULL, swap = FALSE, threshold = TRUE, ...)

## S4 method for signature 'GatingSetList,character'
getSingleCellExpression(x, nodes, ...)
```

Arguments

x	A GatingSet or GatingSetList object .
nodes	character vector specifying different cell populations
other.markers	character vector specifying the extra markers/channels to be returned besides the ones derived from "nodes" and "map" argument.It is only valid when threshold is set to FALSE.
swap	logical indicates whether channels and markers of flow data are swapped.
threshold	logical indicates whether to threshold the flow data by setting intensity value to zero when it is below the gate threshold.

... other arguments

map a named list providing the mapping between node names (as specified in the gating hierarchy of the gating set) and channel names (as specified in either the desc or name columns of the parameters of the associated flowFrames in the GatingSet). see examples.

ignore.case whether to ignore case when match the marker names. Default is FALSE.

Value

A list of numeric matrices

Author(s)

Mike Jiang <wjiang2@fhcrc.org>

See Also

[getIndices](#) [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,GatingHierarchy-method

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

Usage

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


Arguments

x	A GatingHierarchy object
...	other arguments
	inverse logical whether to return the inverse transformation function. channel character channel name

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); # return a list transformation functions
  getTransformations(gh, inverse = TRUE); # return a list inverse transformation functions
  getTransformations(gh, channel = "<B710-A") # only return the transfomation associated with given channel

## End(Not run)
```

getTransformations,graphGML-method

Extract transformations from graphGML object.

Description

Extract transformations from graphGML object.

Usage

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

Arguments

x graphGML

Value

transformList object

graphGML-class	<i>A graph object returned by 'read.gatingML.cytobank' function.</i>
----------------	--

Description

Each node corresponds to a population(or GateSet) defined in gatingML file. The actual gate object (both global and tailored gates) is associated with each node as nodeData. Compensation and transformations are stored in graphData slot.

Details

The class simply extends the graphNEL class and exists for the purpose of method dispatching.

insertGate	<i>insert a dummy gate to the GatingSet</i>
------------	---

Description

Is is useful trick to make the tree structure of GatingSet same with other so that they can be combined into a 'GatingSetList' object.

Usage

```
insertGate(gs, gate, parent, children)
```

Arguments

gs	GatingSet to work with
gate	filter a dummy gate to be inserted, its 'filterId' will be used as the population name
parent	character full path of parent node where the new dummy gate to be added to
children	character full path of children nodes that the new dummy gate to be parent of

Value

a new GatingSet object with the new gate added but share the same flow data with the input 'GatingSet'

Examples

```
## Not run:
#construct a dummy singlet gate
dummyGate <- rectangleGate("FSC-A" = c(-Inf, Inf), "FSC-H" = c(-Inf, Inf), filterId = "singlets")
#insert it between the 'not debris' node and "lymph" node
gs_clone <- insertGate(gs, dummyGate, "not debris", "lymph")

## End(Not run)
```

isNcdf	<i>determine the flow data associated with a Gating Hierarchy is based on 'ncdfFlowSet' or 'flowSet'</i>
--------	--

Description

determine the flow data associated with a Gating Hierarchy is based on 'ncdfFlowSet' or 'flowSet'

Usage

```
isNcdf(x)
```

Arguments

x GatingHierarchy object

Value

logical

keyword, GatingHierarchy, character-method	<i>Retrieve a specific keyword for a specific sample in a GatingHierarchy or or set of samples in a GatingSet or GatingSetList</i>
--	--

Description

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

Usage

```
## S4 method for signature 'GatingHierarchy,character'  
keyword(object, keyword)  
  
## S4 method for signature 'GatingHierarchy,missing'  
keyword(object, keyword = "missing", ...)  
  
## S4 method for signature 'GatingSet,missing'  
keyword(object, keyword = "missing", ...)  
  
## S4 method for signature 'GatingSet,character'  
keyword(object, keyword)  
  
## S4 method for signature 'GatingSetList,missing'  
keyword(object, keyword = "missing", ...)  
  
## S4 method for signature 'GatingSetList,character'  
keyword(object, keyword)
```

Arguments

object	GatingHierarchy or GatingSet or GatingSetList
keyword	character specifying keyword name. When missing, extract all keywords.
...	other arguments passed to keyword-methods

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]])  
# filter the instrument setting  
keyword(G[[1]], compact = TRUE)  
#get single keyword from all samples  
keyword(G, "FILENAME")  
#get single keyword from one sample  
keyword(G[[1], "FILENAME")  
  
## End(Not run)
```

lapply,GatingSet-method
apply FUN to each sample (i.e. GatingHierarchy)

Description

sample names are used for names of the returned list

Usage

```
## S4 method for signature 'GatingSet'
lapply(X, FUN, ...)
```

Arguments

X	GatingSet
FUN	function to be applied to each sample in 'GatingSet'
...	other arguments to be passed to 'FUN'

length,GatingSet-method
Methods to get the length of a GatingSet

Description

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

Usage

```
## S4 method for signature 'GatingSet'
length(x)

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

Arguments

x	GatingSet
object	object

matchPath	<i>Given the leaf node, try to find out if a collection of nodes can be matched to a path in a graph(tree) by the bottom-up searching</i>
-----------	---

Description

Given the leaf node, try to find out if a collection of nodes can be matched to a path in a graph(tree) by the bottom-up searching

Usage

```
matchPath(g, leaf, nodeSet)
```

Arguments

g	graphNEL
leaf	the name of leaf(terminal) node
nodeSet	a set of node names

Value

TRUE if path is found, FALSE if not path is matched.

mkformula	<i>make a formula from a character vector</i>
-----------	---

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, character-method

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.

Usage

```
## S4 method for signature 'character'
openWorkspace(file, options = 0, ...)
```

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

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 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

a flowJoWorkspace object.

Examples

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

## End(Not run)
```

parse.gateInfo	<i>Parse the cytobank custom_info for each gate</i>
----------------	---

Description

Fcs filename and gate name stored in 'custom_info' element are beyond the scope of the gatingML standard and thus not covered by the default 'read.gatingML'.

Usage

```
parse.gateInfo(file, ...)
```

Arguments

file	xml file path
...	additional arguments passed to the handlers of 'xmlTreeParse'

Value

a data.frame that contains three columns: id (gateId), name (gate name), fcs (fcs_file_filename).

parse.gatingML	<i>A wrapper that parse the gatingML and FCS files into GatingSet</i>
----------------	---

Description

A wrapper that parse the gatingML and FCS files into GatingSet

Usage

```
parse.gatingML(xml, FCS)
```

Arguments

xml	the full path of gatingML file
FCS	FCS files to be loaded

Value

a GatingSet

Examples

```
## Not run:
xmlfile <- system.file("extdata/cytotrol_tcell_cytobank.xml", package = "flowWorkspace")
fcsFiles <- list.files(pattern = "CytoTrol", system.file("extdata", package = "flowWorkspaceData"), full = T)
gs <- parse.gatingML(xmlfile, fcsFiles)
plotGate(gs[[1]])

## End(Not run)
```

parseWorkspace, flowJoWorkspace-method

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.

Usage

```
## S4 method for signature 'flowJoWorkspace'
parseWorkspace(obj, ...)
```

Arguments

obj	A flowJoWorkspace to be parsed.
...	<ul style="list-style-type: none"> 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. Or an expression to be passed to 'subset' function to filter samples by 'pData' (Note that the columns referred by the expression must also be explicitly specified in 'keywords' argument)

- `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` either a character scalar or `data.frame`. When character, it is a 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. When it is a `data.frame`, it is expected to contain two columns: 'sampleID' and 'file', which is used as the mapping between 'sampleID' and FCS file (absolute) path. When such mapping is provided, the file system searching is avoided.
- `sampNloc` 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`
- `channel.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.
- `leaf.bool` a logical whether to compute the leaf boolean gates. Default is TRUE. It helps to speed up parsing by turning it off when the statistics of these leaf boolean gates are not important for analysis. (e.g. COMPASS package will calculate them by itself.) If needed, they can be calculated by calling `recompute` method at later stage.
- `additional.keys` character vector: The keywords (parsed from FCS header) to be combined(concatenated with "_") with FCS filename to uniquely identify samples. Default is '\$TOT' (total number of cells) and more keywords can be added to make this GUID.
- `keywords` character vector specifying the keywords to be extracted as `pData` of `GatingSet`
- `keywords.source` character the place where the keywords are extracted from, can be either "XML" or "FCS"
- `keyword.ignore.case` a logical flag indicates whether the keywords matching needs to be case sensitive.
- ...: 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)
#parse the second group
gs <- parseWorkspace(ws, name = 2); #assume that the fcs files are under the same folder as workspace

gs <- parseWorkspace(ws, name = 4
                    , path = dataDir      #specify the FCS path
                    , subset = "CytoTrol_CytoTrol_1.fcs" #subset the parsing by FCS filename
                    , isNcdf = FALSE)#turn off cdf storage mode (normally you don't want to do this for parsing la

gs <- parseWorkspace(ws, path = dataDir, name = 4
                    , keywords = c("PATIENT ID", "SAMPLE ID", "$TOT", "EXPERIMENT NAME") #tell the parser to extr
                    , keywords.source = "XML" # keywords are extracted from xml workspace (alternatively can be s
                    , additional.keys = c("PATIENT ID") #use additional keywords together with FCS filename to un
                    , execute = F) # parse workspace without the actual gating (can save time if just want to get

#subset by pData (extracted from keywords)
gs <- parseWorkspace(ws, path = dataDir, name = 4
                    , subset = `TUBE NAME` %in% c("CytoTrol_1", "CytoTrol_2")
                    , keywords = "TUBE NAME")

## End(Not run)
```

pData,GatingHierarchy-method
read/set pData 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

Usage

```
## S4 method for signature 'GatingHierarchy'
pData(object)

## S4 method for signature 'GatingSet'
pData(object)

## S4 replacement method for signature 'GatingSet,data.frame'
pData(object) <- value

## S4 replacement method for signature 'GatingSetList,data.frame'
pData(object) <- value
```

Arguments

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

Value

a data.frame

plot,GatingSet,missing-method
plot a gating tree

Description

Plot a tree/graph representing the GatingHierarchy

Usage

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

## S4 method for signature 'GatingSet,character'
plot(x, y, ...)
```

Arguments

x	GatingHierarchy or GatingSet. If GatingSet, the first sample will be used to extract gating tree.
y	missing or character specifies.
...	other arguments: <ul style="list-style-type: none"> • boolean: TRUE FALSE logical specifying whether to plot boolean gate nodes. Defaults to FALSE. • showHidden: TRUE FALSE logical whether to show hidden nodes • 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

Examples

```
## Not run:
#gs is a GatingSet
plot(gs) # the same as plot(gs[[1]])
#plot a subtree 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)

```

```
plot,graphGML,missing-method
```

plot the population tree stored in graphGML.

Description

The node with dotted order represents the population that has tailored gates (sample-specific gates) defined.

Usage

```

## S4 method for signature 'graphGML,missing'
plot(x, y = "missing", label = c("popName",
  "gateName"))

```

Arguments

x	a graphNEL generated by constructTree function
y	not used
label	specifies what to be displayed as node label. Can be either 'popName' (population name parsed from GateSets) or 'gateName' (the name of the actual gate associated with each node)

Examples

```

## Not run:
xmlfile <- system.file("extdata/cytotrol_tcell_cytobank.xml", package = "flowWorkspace")
g <- read.gatingML.cytobank(xmlfile)
plot(g)

## End(Not run)

```

plotGate	<i>Plot gates and associated cell population contained in a GatingHierarchy or GatingSet</i>
----------	--

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.

Usage

```
plotGate(x, y, ...)
```

```
## S4 method for signature 'GatingHierarchy,numeric'
```

```
plotGate(x, y, ...)
```

```
## S4 method for signature 'GatingSet,missing'
```

```
plotGate(x, y, ...)
```

```
## S4 method for signature 'GatingSetList,character'
```

```
plotGate(x, y, ...)
```

Arguments

x	<code>GatingSet</code> or <code>GatingHierarchy</code> object
y	character the node name or full/(partial) gating path or numeric representing the node index in the <code>GatingHierarchy</code> . or <code>missing</code> 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 getNode .
...	<ul style="list-style-type: none"> • <code>bool</code> logical specifying whether to plot boolean gates. • <code>arrange.main</code> character The title of the main page of the plot. Default is the sample name. Only valid when <code>x</code> is <code>GatingHierarchy</code> • <code>arrange</code> logical indicating whether to arrange different populations/nodes on the same page via <code>arrangeGrob</code> call. • <code>merge</code> logical indicating whether to draw multiple gates on the same plot if these gates share the same parent population and same <code>x,y</code> dimensions/parameters; • <code>projections</code> list of character vectors used to customize <code>x,y</code> axis. By default, the <code>x,y</code> axis are determined by the respective gate parameters. The elements of the list are named by the population name or path (see <code>y</code>). Each element is a pair of named character specifying the channel name(or marker name) for <code>x, y</code> 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 dimensions 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.
- `overlayNode` names. These populations are plotted on top of the existing gates(defined by `y` argument) as the overlaid dots.
- `overlay.symbolA` named (lattice graphic parameter) list that defines the symbol color and size for each overlaid population. If not given, we automatically assign the colors.
- `keyLattice` legend parameter for overlay symbols.
- `default.y` character specifying y channel for `xyplot` when plotting a 1d gate. Default is "SSC-A" and session-wise setting can be stored by `'flowWorkspace.par.set("plotGate", list(default.y = "FSC-A"))'`
- `type` character either "xyplot" or "densityplot". Default is "xyplot" and session-wise setting can be stored by `'flowWorkspace.par.set("plotGate", list(type = "xyplot"))'`
- `fitGate` used to disable behavior of plotting the gate region in 1d densityplot. Default is `FALSE` and session-wise setting can be stored by `'flowWorkspace.par.set("plotGate", list(fitGate = FALSE))'`
- `strip.ligcal` specifies whether to show pop name in strip box, only valid when `x` is `GatingHierarchy`
- `raw.scale` logical whether to show the axis in raw(untransformed) scale. Default is `TRUE` and can be stored as session-wise setting by `'flowWorkspace.par.set("plotGate", list(raw.scale = TRUE))'`
- `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. They can be stored as session-wise setting by `'flowWorkspace.par.set("plotGate", list(xlim = "instrument"))'`
- ...
- `pathA` character or numeric scalar passed to `getNodePaths` 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,getNode(G)[5]);#plot the gate for the fifth node

## End(Not run)
```

```
plotPopCV,GatingHierarchy-method
```

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, path = "auto", ...)

## S4 method for signature 'GatingSet'
plotPopCV(x, scales = list(x = list(rot = 90)),
  path = "auto", ...)
```

Arguments

x	A GatingHierarchy from 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.
path	character see getNode
scales	list see barchart
...	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)
```

prettyAxis

Determine tick mark locations and labels for a given channel axis

Description

Determine tick mark locations and labels for a given channel axis

Usage

```
prettyAxis(gh, channel)
```

Arguments

gh	GatingHierarchy
channel	character channel name

Value

when there is transformation function associated with the given channel, it returns a list of that contains positions and labels to draw on the axis other wise returns NULL

Examples

```
## Not run:  
prettyAxis(gh, "<B710-A>")  
  
## End(Not run)
```

 read.gatingML.cytobank

Parser for gatingML exported by Cytobank

Description

The Default parser (flowUtils::read.gatingML) does not parse the population tree as well as the custom information from cytobank. (e.g. gate name, fcs filename).

Usage

```
read.gatingML.cytobank(file, ...)
```

Arguments

file	Gating-ML XML file
...	additional arguments passed to the handlers of 'xmlTreeParse'

Value

a graphGML that represents the population tree. The gate and population name are stored in node-Data of each node. Compensation and transformations are stored in graphData.

Examples

```
## Not run:
xml <- system.file("extdata/cytotrol_tcell_cytobank.xml", package = "flowWorkspace")
g <- read.gatingML.cytobank(xml) #parse the population tree
plot(g) #visualize it

## End(Not run)
```

 recompute,GatingSet-method

Compute the cell events by the gates stored within the gating tree

Description

Compute each cell event to see if it falls into the gate stored within the gating tree and store the result as cell count.

Usage

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

## S4 method for signature 'GatingSetList'
recompute(x, ...)
```

Arguments

x GatingSet

... other arguments y character node name or node path alwaysLoadData logical specifies whether to load the flow raw data for gating for boolean gates, sometime it is more efficient to skip loading the raw data if all the reference nodes and parent are already gates Default 'FALSE' will check the parent node and reference to determine whether to load the data but this check may not be sufficient since the further upstream ancestor nodes may not be gated yet In that case, we allow the gating to be failed and prompt user to recompute those nodes explicitly When TRUE, then it forces data to be loaded to guarantee the gating process to be uninterrupted , yet may at the cost of unnecessary data IO

Details

It is usually used immediately after [add](#) or [setGate](#) calls.

sampleNames,GatingHierarchy-method

Get/update sample names in a GatingSet

Description

Return a sample names contained in a GatingSet

Usage

```
## S4 method for signature 'GatingHierarchy'
sampleNames(object)
```

```
## S4 method for signature 'GatingSet'
sampleNames(object)
```

```
## S4 replacement method for signature 'GatingSet'
sampleNames(object) <- value
```

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	<i>save/load a GatingSet/GatingSetList to/from disk.</i>
---------	--

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"), lib = c("PB",
      "BS"), ...)

load_gs(path)

save_gslist(gslist, path, ...)

load_gslist(path)
```

Arguments

G	A GatingSet
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.(only meaningful when lib == 'BS') default is "binary", which is smaller and faster but machine-dependent. use "text" or "xml" for cross-platform data interchange (deprecated by 'PB' serialization).
lib	a character scalar specifying which serialization library to use. Can be either 'PB' (google protocol buffer) or 'BS'(boost serialization).
...	other arguments: not used.
gslist	A GatingSetList

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)
```

setGate,GatingHierarchy,character,filter-method
update the gate

Description

update the population node with a flowCore-compatible gate object

Usage

```
## S4 method for signature 'GatingHierarchy,character,filter'
setGate(obj, y, value,
        negated = FALSE, ...)

## S4 method for signature 'GatingSet,character,list'
setGate(obj, y, value, ...)

## S4 method for signature 'GatingSet,character,filterList'
setGate(obj, y, value, ...)
```

Arguments

obj	GatingHierarchy or GatingSet
y	character node name or path
value	filter or filterList or list of filter objects
negated	logical see add
...	other arguments

Details

Usually `recompute` is followed by this call since updating a gate doesn't re-calculating the cell events within the gate automatically. see `filterObject` for the gate types that are currently supported.

Examples

```
## Not run:
rg1 <- rectangleGate("FSC-H"=c(200,400), "SSC-H"=c(250, 400), filterId="rectangle")
rg2 <- rectangleGate("FSC-H"=c(200,400), "SSC-H"=c(250, 400), filterId="rectangle")
flist <- list(rg1,rg2)
names(flist) <- sampleNames(gs[1:2])
setGate(gs[1:2], "lymph", flist)
recompute(gs[1:2], "lymph")

## End(Not run)
```

setNode,GatingHierarchy,character,character-method

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

Usage

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

## S4 method for signature 'GatingHierarchy,character,logical'
setNode(x, y, value)

## S4 method for signature 'GatingSet,character,ANY'
setNode(x, y, value)
```

Arguments

x	GatingHierarchy object
y	character node name or path
value	A character the name of the node. or logical to indicate whether to hide a node

Examples

```
## Not run:
#G is a gating hierarchy
getNode(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)
```

show, graphGML-method *show method for graphGML*

Description

show method for graphGML

Usage

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

Arguments

object graphGML

subset.GatingSet *subset the GatingSet/GatingSetList based on 'pData'*

Description

subset the GatingSet/GatingSetList based on 'pData'

Usage

```
subset.GatingSet(x, subset, ...)
```

Arguments

x GatingSet or GatingSetList

subset logical expression(within the context of pData) indicating samples to keep. see [subset](#)

... other arguments. (not used)

Value

a codeGatingSet or GatingSetList object

transform,GatingSet-method

transform the flow data associated with the GatingSet

Description

The transformation functions are saved in the GatingSet and can be retrieved by [getTransformations](#). Currently only flowJo-type biexponential transformation(either returned by [getTransformations](#) or constructed by [flowJoTrans](#)) is supported.

Usage

```
## S4 method for signature 'GatingSet'
transform(`_data`, ...)

## S4 method for signature 'GatingSetList'
transform(`_data`, ...)
```

Arguments

<code>_data</code>	GatingSet or GatingSetList
<code>...</code>	expect a transformList object

Value

a GatingSet or GatingSetList object with the underlying flow data transformed.

Examples

```
## Not run:
#construct biexponential transformation function
biexpTrans <- flowJoTrans(channelRange=4096, maxValue=262144, pos=4.5,neg=0, widthBasis=-10)
#make a transformList object
transFuns <- transformList(chnlS, biexpTrans)
#add it to GatingSet
gs_trans <- transform(gs, transFuns)

## End(Not run)
```

updateChannels *Update the channel information of a GatingSet (c++ part)*

Description

It updates the channels stored in gates, compensations and transformations based on given mapping between the old and new channel names.

Usage

```
updateChannels(gs, map, all = TRUE)
```

Arguments

gs	a GatingSet object
map	data.frame contains the mapping from old (case insensitive) to new channel names Note: Make sure to remove the '<' or '>' characters from 'old' name because the API tries to only look at the raw channel name so that the gates with both prefixed and non-prefixed names could be updated.
all	logical whether to update the flow data as well

Value

when 'all' is set to TRUE, it returns a new GatingSet but it still shares the same underlying c++ tree structure with the original GatingSet otherwise it returns nothing (less overhead.)

Examples

```
## Not run:
  ##this will update both "Qdot 655-A" and "<Qdot 655-A>"
  gs <- updateGateParameter(gs, map = data.frame(old = c("Qdot 655-A")
                                                , new = c("QDot 655-A")
                                                )
                           )

## End(Not run)
```

Index

- [,GatingSet,ANY-method
(GatingSet-class), 19
- [,GatingSetList,ANY-method
(GatingSet-class), 19
- [[,GatingSet,character-method
(GatingSet-class), 19
- [[,GatingSet,logical-method
(GatingSet-class), 19
- [[,GatingSet,numeric-method
(GatingSet-class), 19

- add, 8, 60, 62
- add(add,GatingSet,list-method), 4
- add,GatingHierarchy,filter-method
(add,GatingSet,list-method), 4
- add,GatingHierarchy,filters-method
(add,GatingSet,list-method), 4
- add,GatingHierarchy,quadGate-method
(add,GatingSet,list-method), 4
- add,GatingSet,filter-method
(add,GatingSet,list-method), 4
- add,GatingSet,filterList-method
(add,GatingSet,list-method), 4
- add,GatingSet,filters-method
(add,GatingSet,list-method), 4
- add,GatingSet,filtersList-method
(add,GatingSet,list-method), 4
- add,GatingSet,list-method, 4
- add,GatingSetList,filter-method
(add,GatingSet,list-method), 4
- add,GatingSetList,filterList-method
(add,GatingSet,list-method), 4
- add,GatingSetList,filters-method
(add,GatingSet,list-method), 4
- add,GatingSetList,filtersList-method
(add,GatingSet,list-method), 4
- add,GatingSetList,list-method
(add,GatingSet,list-method), 4
- archive, 7

- barchart, 57
- booleanFilter(booleanFilter-class), 8
- booleanFilter-class, 8

- char2booleanFilter
(booleanFilter-class), 8
- clone, 8
- clone,GatingSet-method(clone), 8
- clone-methods(clone), 8
- clone.ncdfFlowSet, 9
- closeWorkspace
(openWorkspace,character-method),
47
- closeWorkspace,flowJoWorkspace-method
(openWorkspace,character-method),
47
- compare.counts, 9
- compensate,flowSet,graphGML-method, 10
- constructTree, 11

- expressionFilter, 8
- extend, 11

- filterObject, 63
- filterObject
(filterObject,rectangleGate-method),
12
- filterObject,booleanFilter-method
(filterObject,rectangleGate-method),
12
- filterObject,ellipsoidGate-method
(filterObject,rectangleGate-method),
12
- filterObject,polygonGate-method
(filterObject,rectangleGate-method),
12
- filterObject,rectangleGate-method, 12
- flowData, 26
- flowData(flowData,GatingSet-method), 13
- flowData,GatingSet-method, 13

- flowData<- (flowData,GatingSet-method),
13
- flowData<- ,GatingSet-method
(flowData,GatingSet-method), 13
- flowJo.fasinh, 14
- flowJo.fsinh (flowJo.fasinh), 14
- flowJoTrans, 14, 65
- flowJoWorkspace, 20
- flowJoWorkspace-class, 15
- flowWorkspace (flowWorkspace-package), 3
- flowWorkspace-package, 3
- flowWorkspace.par.get
(flowWorkspace.par.set), 16
- flowWorkspace.par.init, 16
- flowWorkspace.par.set, 16
- gating, 17
- gating,graphGML,GatingSet-method
(gating), 17
- GatingHierarchy, 8, 15, 20, 21, 55
- GatingHierarchy-class, 17
- GatingSet, 15, 18, 21, 51, 55
- GatingSet
(GatingSet,character,character-method),
18
- GatingSet,character,character-method,
18
- GatingSet,flowSet,ANY-method
(GatingSet,character,character-method),
18
- GatingSet,GatingHierarchy,character-method
(GatingSet,character,character-method),
18
- GatingSet-class, 19
- GatingSetList (GatingSetList-class), 21
- GatingSetList-class, 21
- getChildren
(getParent,GatingSet,character-method),
34
- getChildren,GatingSet,character-method
(getParent,GatingSet,character-method),
34
- getChildren,graphGML,character-method,
23
- getCompensationMatrices
(getCompensationMatrices,GatingHierarchy-method),
23
- getCompensationMatrices,GatingHierarchy-method,
23
- getCompensationMatrices,graphGML-method,
24
- getData, 27
- getData
(getData,GatingHierarchy,missing-method),
25
- getData,GatingHierarchy,character-method
(getData,GatingHierarchy,missing-method),
25
- getData,GatingHierarchy,missing-method,
25
- getData,GatingSet,character-method
(getData,GatingHierarchy,missing-method),
25
- getData,GatingSet,missing-method
(getData,GatingHierarchy,missing-method),
25
- getData,GatingSetList,ANY-method
(getData,GatingHierarchy,missing-method),
25
- getFJWSubsetIndices, 26
- getGate
(getGate,GatingHierarchy,character-method),
27
- getGate,GatingHierarchy,character-method,
27
- getGate,GatingSet,character-method
(getGate,GatingHierarchy,character-method),
27
- getGate,GatingSetList,character-method
(getGate,GatingHierarchy,character-method),
27
- getGate,graphGML,character-method, 28
- getIndiceMat, 28
- getIndices, 26, 40
- getIndices
(getIndices,GatingHierarchy,character-method),
29
- getIndices,GatingHierarchy,character-method,
29
- getIndices,GatingSet,name-method, 30
- getKeywords
(getKeywords,flowJoWorkspace,character-method),
31
- getKeywords,flowJoWorkspace,character-method,
31
- getKeywords,flowJoWorkspace,numeric-method
(getKeywords,flowJoWorkspace,character-method),

- 31
- getLogLevel, 32
- getNodes, 27, 34–37, 55–57
- getNodes (getNodes, GatingSet-method), 32
- getNodes, GatingSet-method, 32
- getNodes, graphGML-method, 33
- getParent
 - (getParent, GatingSet, character-method), 34
 - getParent, GatingSet, character-method, 34
 - getParent, graphGML, character-method, 35
- getPopStats, 26, 29, 40, 58
- getPopStats
 - (getProp, GatingHierarchy, character-method), 36
 - getPopStats, GatingHierarchy-method (getProp, GatingHierarchy, character-method), 36
 - getPopStats, GatingSet-method (getProp, GatingHierarchy, character-method), 36
 - getPopStats, GatingSetList-method (getProp, GatingHierarchy, character-method), 36
- getProp
 - (getProp, GatingHierarchy, character-method), 36
 - getProp, GatingHierarchy, character-method, 36
- getSampleGroups, 51
- getSampleGroups
 - (getSampleGroups, flowJoWorkspace-method), 37
 - getSampleGroups, flowJoWorkspace-method, 37
- getSamples
 - (getSamples, flowJoWorkspace-method), 38
 - getSamples, flowJoWorkspace-method, 38
- getSingleCellExpression
 - (getSingleCellExpression, GatingSet, character-method), 39
 - getSingleCellExpression, GatingSet, character-method, 39
 - getSingleCellExpression, GatingSetList, character-method (getSingleCellExpression, GatingSet, character-method), 39
- 39
- getTotal
 - (getProp, GatingHierarchy, character-method), 36
 - getTotal, GatingHierarchy, character-method (getProp, GatingHierarchy, character-method), 36
- getTransformations, 65
- getTransformations
 - (getTransformations, GatingHierarchy-method), 40
 - getTransformations, GatingHierarchy-method, 40
 - getTransformations, graphGML-method, 41
- graphGML-class, 42
- grid, layout, 56
- insertGate, 42
- isNcdf, 43
- keyword
 - (keyword, GatingHierarchy, character-method), 43
 - keyword, GatingHierarchy, character-method, 43
 - keyword, GatingHierarchy, missing-method (keyword, GatingHierarchy, character-method), 43
 - keyword, GatingSet, character-method (keyword, GatingHierarchy, character-method), 43
 - keyword, GatingSet, missing-method (keyword, GatingHierarchy, character-method), 43
 - keyword, GatingSetList, character-method (keyword, GatingHierarchy, character-method), 43
 - keyword, GatingSetList, missing-method (keyword, GatingHierarchy, character-method), 43
- lapply, GatingSet-method, 45
- lattice, 56
- layoutGraph, 53
- length (length, GatingSet-method), 45
- length, GatingSet-method, 45
- load_gs (save_gs), 61
- load_gslist (save_gs), 61
- matchPath, 46

- mkformula, 46
- ncFlowSet, 47
- ncFlowSet, GatingSet-method (ncFlowSet), 47
- ncFlowSet<- (ncFlowSet), 47
- ncFlowSet<-, GatingSet-method (ncFlowSet), 47
- openWorkspace, 38
- openWorkspace (openWorkspace, character-method), 47
- openWorkspace, character-method, 47
- parse.gateInfo, 48
- parse.gatingML, 48
- parseWorkspace, 19, 20, 27, 38
- parseWorkspace (parseWorkspace, flowJoWorkspace-method), 49
- parseWorkspace, flowJoWorkspace-method, 49
- pData (pData, GatingHierarchy-method), 52
- pData, GatingHierarchy-method, 52
- pData, GatingSet-method (pData, GatingHierarchy-method), 52
- pData<- (pData, GatingHierarchy-method), 52
- pData<-, GatingSet, data.frame-method (pData, GatingHierarchy-method), 52
- pData<-, GatingSetList, data.frame-method (pData, GatingHierarchy-method), 52
- plot (plot, GatingSet, missing-method), 52
- plot, GatingSet, character-method (plot, GatingSet, missing-method), 52
- plot, GatingSet, missing-method, 52
- plot, graphGML, missing-method, 54
- plotGate, 55
- plotGate, GatingHierarchy, character-method (plotGate), 55
- plotGate, GatingHierarchy, missing-method (plotGate), 55
- plotGate, GatingHierarchy, numeric-method (plotGate), 55
- plotGate, GatingSet, character-method (plotGate), 55
- plotGate, GatingSet, missing-method (plotGate), 55
- plotGate, GatingSet, numeric-method (plotGate), 55
- plotGate, GatingSetList, character-method (plotGate), 55
- plotGate-methods (plotGate), 55
- plotPopCV (plotPopCV, GatingHierarchy-method), 57
- plotPopCV, GatingHierarchy-method, 57
- plotPopCV, GatingSet-method (plotPopCV, GatingHierarchy-method), 57
- prettyAxis, 58
- rbind2, GatingSetList, missing-method (GatingSetList-class), 21
- read.flowSet, 50
- read.gatingML.cytobank, 59
- read.ncdfFlowSet, 50
- recompute, 63
- recompute (recompute, GatingSet-method), 59
- recompute, GatingSet-method, 59
- recompute, GatingSetList-method (recompute, GatingSet-method), 59
- Rm (add, GatingSet, list-method), 4
- Rm, character, GatingHierarchy, character-method (add, GatingSet, list-method), 4
- Rm, character, GatingSet, character-method (add, GatingSet, list-method), 4
- Rm, character, GatingSetList, character-method (add, GatingSet, list-method), 4
- sampleNames (sampleNames, GatingHierarchy-method), 60
- sampleNames, GatingHierarchy-method, 60
- sampleNames, GatingSet-method (sampleNames, GatingHierarchy-method), 60
- sampleNames<- (sampleNames, GatingHierarchy-method), 60

sampleNames<- ,GatingSet,ANY-method
 (unarchive (archive), 7
 (sampleNames,GatingHierarchy-method), updateChannels, 66
 60

sampleNames<- ,GatingSet-method
 (xmlTreeParse, 15, 19, 47, 50
 (sampleNames,GatingHierarchy-method), xyplot, 56
 60

save_gs, 61

save_gslist (save_gs), 61

setGate, 60

setGate
 (setGate,GatingHierarchy,character,filter-method),
 62

setGate,GatingHierarchy,character,filter-method,
 62

setGate,GatingSet,character,filterList-method
 (setGate,GatingHierarchy,character,filter-method),
 62

setGate,GatingSet,character,list-method
 (setGate,GatingHierarchy,character,filter-method),
 62

setLogLevel (getLogLevel), 32

setNode
 (setNode,GatingHierarchy,character,character-method),
 63

setNode,GatingHierarchy,character,character-method,
 63

setNode,GatingHierarchy,character,logical-method
 (setNode,GatingHierarchy,character,character-method),
 63

setNode,GatingSet,character,ANY-method
 (setNode,GatingHierarchy,character,character-method),
 63

show,booleanFilter-method
 (booleanFilter-class), 8

show,flowJoWorkspace-method
 (flowJoWorkspace-class), 15

show,GatingHierarchy-method
 (GatingHierarchy-class), 17

show,GatingSet-method
 (length,GatingSet-method), 45

show,graphGML-method, 64

subset, 64

subset.GatingSet, 64

transform,GatingSet-method, 65

transform,GatingSetList-method
 (transform,GatingSet-method),
 65