

# Package ‘CytoML’

April 15, 2020

**Type** Package

**Title** A GatingML Interface for Cross Platform Cytometry Data Sharing

**Version** 1.12.1

**Date** 2016-04-15

**Author** Mike Jiang, J. Spidlen., N. Gopalakrishnan, F. Hahne, B. Ellis, R. Gentleman, M. Dalphin, N. Le Meur, B. Purcell

**Maintainer** Mike Jiang <wjiang2@fhcrc.org>

**Description** Uses platform-specific implemenations of the GatingML2.0 standard to exchange gated cytometry data with other software platforms.

**License** Artistic-2.0

**LazyData** TRUE

**Imports** flowCore (>= 1.43.10), flowWorkspace (>= 3.33.10), openCyto (>= 1.11.3), XML, data.table, jsonlite, RBGL, ncdfFlow, Rgraphviz, Biobase, methods, graph, graphics, utils, base64enc, plyr, dplyr, grDevices, methods, ggcyto (>= 1.11.4), yaml, lattice, stats, corpcor, RUnit

**biocViews** ImmunoOncology, FlowCytometry, DataImport, DataRepresentation

**LinkingTo** Rcpp, BH(>= 1.62.0-1), RProtoBufLib(>= 1.3.7), cytolib(>= 1.3.3), RcppParallel

**Suggests** testthat, flowWorkspaceData (>= 2.11.1), knitr, parallel

**VignetteBuilder** knitr

**RoxygenNote** 6.1.1

**BugReports** <https://github.com/RGLab/CytoML/issues>

**URL** <https://github.com/RGLab/CytoML>

**Collate** 'AllClasses.R' 'GatingSet2cytobank.R' 'GatingSet2flowJo.R' 'RcppExports.R' 'gate-methods.R' 'compensation.R' 'cytobank2GatingSet.R' 'cytobankExperiment.R' 'flowJoWorkspace\_Methods.R' 'diva2GatingSet.R' 'flowUtils\_functions.R' 'gatingML.R' 'read.gatingML.cytobank.R' 'graphGML\_methods.R' 'helperFunctions.R' 'parameter-methods.R' 'parseDivaWorkspace\_old.R' 'transforms.R' 'utils.R' 'writeGatingML.R' 'zzz.R'

**SystemRequirements** xml2, GNU make, C++11

**Encoding** UTF-8

**git\_url** <https://git.bioconductor.org/packages/CytoML>

**git\_branch** RELEASE\_3\_10

**git\_last\_commit** 2167017

**git\_last\_commit\_date** 2020-03-26

**Date/Publication** 2020-04-14

## R topics documented:

addCustomInfo . . . . .	3
compensate,GatingSet,graphGML-method . . . . .	3
constructTree . . . . .	4
cytobankExperiment . . . . .	4
cytobank_to_gatingset.default . . . . .	5
CytoML.par.init . . . . .	6
CytoML.par.set . . . . .	6
diva_workspace-class . . . . .	7
extend . . . . .	7
flowjo_workspace-class . . . . .	9
gating,graphGML,GatingSet-method . . . . .	9
GatingSet,character,character-method . . . . .	10
GatingSet2cytobank . . . . .	10
GatingSet2flowJo . . . . .	11
getChildren,graphGML,character-method . . . . .	12
getCompensationMatrices.graphGML . . . . .	13
getFJWSubsetIndices . . . . .	13
getGate,graphGML,character-method . . . . .	14
getKeywords . . . . .	14
getNodes,graphGML-method . . . . .	15
getParent,graphGML,character-method . . . . .	16
getSampleGroups . . . . .	16
getSamples . . . . .	17
getTransformations.graphGML . . . . .	18
graphGML-class . . . . .	18
gs_compare_cytobank_counts . . . . .	18
matchPath . . . . .	19
openDiva . . . . .	20
open_flowjo_xml . . . . .	20
parse.gateInfo . . . . .	21
parseWorkspace,flowjo_workspace-method . . . . .	22
plot,graphGML,missing-method . . . . .	24
range.GatingHierarchy . . . . .	25
read.gatingML.cytobank . . . . .	26
show,graphGML-method . . . . .	27

**Index**

**28**

---

addCustomInfo                    *add customInfo nodes to each gate node and add BooleanAndGates*

---

**Description**

add customInfo nodes to each gate node and add BooleanAndGates

**Usage**

```
addCustomInfo(root, gs, flowEnv, cytobank.default.scale = TRUE,
              showHidden)
```

**Arguments**

root	the root node of the XML
gs	a GatingSet object
flowEnv	the environment that stores the information parsed by 'read.GatingML'.
cytobank.default.scale	logical flag indicating whether to use the default Cytobank asinhtGml2 settings. Currently it should be set to TRUE in order for gates to be displayed properly in Cytobank because cytobank currently does not parse the global scale settings from GatingML.
showHidden	whether to include the hidden population nodes in the output

**Value**

XML root node

---

compensate,GatingSet,graphGML-method  
*compensate a GatingSet based on the compensation information stored in graphGML object*

---

**Description**

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

**Usage**

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

**Arguments**

x	GatingSet
spillover	graphGML
...	unused.

**Value**

compensated GatingSet

---

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.

---

cytobankExperiment	<i>Construct cytobank_experiment object from ACS file</i>
--------------------	---

---

**Description**

Construct cytobank\_experiment object from ACS file

**Usage**

```
cytobankExperiment(...)  
open_cytobank_experiment(acs, exdir = tempfile())
```

**Arguments**

acs	ACS file exported from Cytobank
exdir	the directory to extract files to

**Value**

cytobank\_experiment object

---

```
cytobank_to_gatingset.default
```

*A wrapper that parse the gatingML and FCS files (or cytobank\_experiment object) into GatingSet*

---

## Description

A wrapper that parse the gatingML and FCS files (or cytobank\_experiment object) into GatingSet

## Usage

```
## Default S3 method:
cytobank_to_gatingset(x, FCS, ...)

cytobank2GatingSet(...)

cytobank_to_gatingset(x, ...)

## S3 method for class 'cytobank_experiment'
cytobank_to_gatingset(x, ...)
```

## Arguments

x	the cytobank_experiment object or the full path of gatingML file
FCS	FCS files to be loaded
...	other arguments

## Value

a GatingSet

## Examples

```
## Not run:
acsfile <- system.file("extdata/cytobank_experiment.acs", package = "CytoML")
ce <- open_cytobank_experiment(acsfile)
xmlfile <- ce$gatingML
fcsFiles <- list.files(ce$fcsdir, full.names = TRUE)
gs <-< cytobank_to_gatingset(xmlfile, fcsFiles)
library(ggcyto)
autoplot(gs[[1]])

## End(Not run)
```

---

CytoML.par.init	<i>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</i>
-----------------	--

---

**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**

```
CytoML.par.init()
```

---

CytoML.par.set	<i>CytoML.par.set sets a set of parameters in the CytoML package namespace.</i>
----------------	---

---

**Description**

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

**Usage**

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

```
CytoML.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 <- CytoML.par.get("flowJo_versions")

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

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

---

diva\_workspace-class
*diva\_workspace class Inherited from flowjo\_workspace-class*

---

### Description

diva\_workspace class Inherited from [flowjo\\_workspace-class](#)

### Usage

```
diva_get_sample_groups(x)
```

```
diva_get_samples(x)
```

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

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

```
diva_to_gatingset(obj, name = NULL, subset = NULL, path = obj@path,
  fast = TRUE, worksheet = c("normal", "global"),
  swap_cols = list(`FSC-H` = "FSC-W", `SSC-H` = "SSC-W"),
  verbose = FALSE, ...)
```

### Arguments

x	diva_workspace
object	diva_workspace
obj	diva_workspace
...	other arguments

---

extend	<i>extend the gate to the minimum and maximum limit of both dimensions based on the bounding information.</i>
--------	---

---

### Description

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.

### Usage

```
extend(gate, bound, data.range = NULL, plot = FALSE,
  limits = c("original", "extended"))
```

```
## S3 method for class 'polygonGate'
extend(gate, bound, data.range = NULL,
  plot = FALSE, limits = c("original", "extended"))
```

```
## 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.
limits	character whether to plot in "extended" or "original" gate limits. Default is "original".
...	other arguments

### Details

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

Normally this function is not used directly by user but invoked when parsing GatingML file exported from Cytobank.

### Value

a flowCore filter/gate

### Examples

```
library(flowCore)
sqrcut <- matrix(c(300,300,600,600,50,300,300,50),ncol=2,nrow=4)
colnames(sqrcut) <- c("FSC-H", "SSC-H")
pg <- polygonGate(filterId="nonDebris", sqrcut)
pg
bound <- matrix(c(100,3e3,100,3e3),
  byrow = TRUE, nrow = 2,
  dimnames = list(c("FSC-H", "SSC-H"),
    c("min", "max")))
bound
pg.extened <- extend(pg, bound, plot = TRUE)
```



---

flowjo\_workspace-class

*An R representation of a flowJo workspace.*

---

### Description

Objects can be created by calls of the form `new("flowjo_workspace.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 <- open_flowjo_xml(wsfile);
ws
fj_ws_get_samples(ws)
```

---

`gating,graphGML,GatingSet-method`

*Apply the gatingML graph to a GatingSet*

---

### Description

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

### Usage

```
## 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.

---

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.)

**Usage**

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

**Arguments**

x	character or flowSet or GatingHierarchy
y	character or missing
guides	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 <a href="#">xmlTreeParse</a>
wsType	character workspace type, can be value of "win", "macII", "vX", "macIII".

---

GatingSet2cytobank      *Convert a GatingSet to a Cytobank-compatible gatingML*

---

**Description**

this function retrieves the gates from GatingSet and writes a customized GatingML-2.0 file that can be imported into cytobank.

**Usage**

```
GatingSet2cytobank(...)

gatingset_to_cytobank(gs, outFile, showHidden = FALSE,
  cytobank.default.scale = TRUE, ...)
```

**Arguments**

... rescale.gate default is TRUE. which means the gate is rescaled to the new scale that is understandable by cytobank. It is recommended not to change this behavior unless user wants to export to a gatingML file used for other purpose other than being imported into cytobank.

gs a GatingSet object

outFile a file name

showHidden whether to include the hidden population nodes in the output

cytobank.default.scale logical flag indicating whether to use the default Cytobank asinhtGml2 settings. Currently it should be set to TRUE in order for gates to be displayed properly in Cytobank because cytobank currently does not parse the global scale settings from GatingML.

**Details**

The process can be divided into four steps: 1. Read in gate geometry, compensation and transformation from gatingSet 2. Rescale gate boundaries with flowjo\_biexp() so gates can be displayed properly in Cytobank 3. Save gates and hierarchy structure to R environment 4. Write environment out to gatingML using write.GatingML()

**Value**

nothing

**Examples**

```
library(flowWorkspace)

dataDir <- system.file("extdata",package="flowWorkspaceData")
gs <- load_gs(list.files(dataDir, pattern = "gs_manual",full = TRUE))

gs_pop_remove(gs, "CD8")

#output to cytobank
outFile <- tempfile(fileext = ".xml")
gatingset_to_cytobank(gs, outFile) #type by default is 'cytobank'
```

---

GatingSet2flowJo

*Convert a GatingSet to flowJo workspace*


---

**Description**

Convert a GatingSet to flowJo workspace

**Usage**

```
GatingSet2flowJo(...)

gatingset_to_flowjo(gs, outFile, ...)
```

**Arguments**

... other arguments showHidden whether to include the hidden population nodes in the output  
 gs a GatingSet object  
 outFile the workspace file path to write

**Value**

nothing

**Examples**

```
library(flowWorkspace)

dataDir <- system.file("extdata",package="flowWorkspaceData")
gs <- load_gs(list.files(dataDir, pattern = "gs_manual",full = TRUE))

#output to flowJo
outFile <- tempfile(fileext = ".wsp")
gatingset_to_flowjo(gs, outFile)
```

---

*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

**Value**

a graphNEL node

**Examples**

```
## Not run:
xmlfile <- system.file("extdata/cytotrol_tcell_cytobank.xml", package = "CytoML")
g <- read.gatingML.cytobank(xmlfile)
getChildren(g, "GateSet_722326")
getParent(g, "GateSet_722326")

## End(Not run)
```

---

```
getCompensationMatrices.graphGML
```

*Extract compensation from graphGML object.*

---

**Description**

Extract compensation from graphGML object.

**Usage**

```
## S3 method for class 'graphGML'
getCompensationMatrices(x)
```

**Arguments**

x                    graphGML

**Value**

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

---

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

---

**Description**

This function will calculate the indices of a subset of samples in a flowjo\_workspace, 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 flowjo\_to\_gatingset.

**Usage**

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

**Arguments**

ws                    flowjo\_workspace 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 flowjo\_to\_gatingset(subset=), based on a keyword/value filter in a specific group of samples.

**Value**

A numeric vector of indices.

**See Also**

[flowjo\\_to\\_gatingset](#)

---

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

**Value**

the gate information associated with the node

---

getKeywords	<i>Get Keywords</i>
-------------	---------------------

---

**Description**

Retrieve keywords associated with a workspace

**Usage**

```
getKeywords(...)  

fj_ws_get_keywords(obj, y, ...)
```

**Arguments**

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

**Details**

Retrieve a list of keywords from a flowjo\_workspace

**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 <- open_flowjo_xml(wsfile);

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

---

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 = "CytoML")
g <- read.gatingML.cytobank(xmlfile)
getNode(g)
getNode(g, only.names = FALSE)

## 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
```

**Value**

a graphNEL node

---

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

---

**Description**

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

**Usage**

```
getSampleGroups(...)

fj_ws_get_sample_groups(x)
```

**Arguments**

```
x          A flowjo_workspace 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**

[flowjo\\_workspace-class open\\_flowjo\\_xml](#)

**Examples**

```
## Not run:
#ws is a flowjo_workspace
fj_ws_get_sample_groups(ws);

## End(Not run)
```

---

getSamples

*Get a list of samples from a flowJo workspace*

---

**Description**

Return a data frame of samples contained in a flowJo workspace

**Usage**

```
getSamples(...)  
fj_ws_get_samples(x, sampNloc = "keyword")
```

**Arguments**

x	A flowjo_workspace
sampNloc	character either "keyword" or "sampleNode". see <a href="#">flowjo_to_gatingset</a>

**Details**

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

**Value**

A data.frame with columns sampleID, name, and compID if x is a flowjo\_workspace.

**Examples**

```
## Not run:
#ws is a flowjo_workspace
fj_ws_get_samples(ws);

## End(Not run)
```

---

```
getTransformations.graphGML
```

*Extract transformations from graphGML object.*

---

### Description

Extract transformations from graphGML object.

### Usage

```
## S3 method for class 'graphGML'
getTransformations(x, ...)
```

### Arguments

x	graphGML
...	not used

### Value

transformerList object

---

```
graphGML-class
```

*A graph object returned by 'read.gatingML.cytobank' function.*

---

### 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.

---

```
gs_compare_cytobank_counts
```

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

---

### Description

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

### Usage

```
gs_compare_cytobank_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.
...	arguments passed to data.table::fread function

**Value**

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

**Examples**

```
acsfile <- system.file("extdata/cytobank_experiment.acs", package = "CytoML")
ce <- open_cytobank_experiment(acsfile)
gs <- cytobank_to_gatingset(ce)
## verify the stats are correct
statsfile <- ce$attachments[1]
dt_merged <- gs_compare_cytobank_counts(gs, statsfile, id.vars = "population", skip = "FCS Filename")
all.equal(dt_merged[, count.x], dt_merged[, count.y], tol = 5e-4)
```

---

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.

---

openDiva	<i>open Diva xml workspace</i>
----------	--------------------------------

---

### Description

open Diva xml workspace

### Usage

```
openDiva(...)
open_diva_xml(file, options = 0, ...)
```

### Arguments

...	arguments passed to <a href="#">xmlTreeParse</a>
file	xml file
options	argument passed to <a href="#">xmlTreeParse</a>

### Value

a diva\_workspace object

### Examples

```
## Not run:
library(flowWorkspace)
library(CytoML)
ws <- open_diva_xml(system.file('extdata/diva/PE_2.xml', package = "flowWorkspaceData"))
ws
diva_get_sample_groups(ws)
gs <- diva_to_gatingset(ws, name = 2, subset = 1)
sampleNames(gs)
gs_get_pop_paths(gs)
plotGate(gs[[1]])

## End(Not run)
```

---

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

---

### Description

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

**Usage**

```
open_flowjo_xml(file, options = 0, ...)

closeWorkspace(workspace)

flowjo_ws_close(workspace)
```

**Arguments**

file	Full path to the XML flowJo workspace file.
options	xml parsing options passed to <a href="#">xmlTreeParse</a>
...	other arguments passed to <a href="#">xmlTreeParse</a>
workspace	A flowjo_workspace

**Details**

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

**Value**

a flowjo\_workspace object.

**Examples**

```
## Not run:
file<-"myworkspace.xml"
ws<-open_flowjo_xml(file);
class(ws); #flowjo_workspace
flowjo_ws_close(ws);

## End(Not run)
```

---

parse.gateInfo

*Parse the cytobank custom\_info for each gate*

---

**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).

---

parseWorkspace, flowjo\_workspace-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 'flowjo_workspace'
parseWorkspace(obj, ...)

flowjo_to_gatingset(obj, name = NULL, subset = NULL,
  requiregates = TRUE, sampNloc = "keyword",
  additional.keys = "$TOT", additional.sampleID = FALSE,
  keywords = NULL, keywords.source = "XML", execute = TRUE,
  path = obj@path, keyword.ignore.case = FALSE, ...)
```

**Arguments**

- |     |   |
|-----|---|
| obj | A flowjo_workspace to be parsed.  |
| ... | <ul style="list-style-type: none"> <li>• 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.</li> <li>• 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.</li> <li>• 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.</li> <li>• 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)</li> <li>• requiregates logical Should samples that have no gates be included?</li> <li>• includeGates logical Should gates be imported, or just the data with compensation and transformation?</li> <li>• 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</li> </ul> |

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 compensation or a list of compensations 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.
- additional.sampleID boolean: A boolean specifying whether to include the flowJo sample ID in a GUID to uniquely identify samples. This can be helpful when the filename or other keywords are not enough to differentiate between samples. Default is FALSE.
- 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 flowjo\_workspace is generated with a call to `open_flowjo_xml()`, passing the name of the xml workspace file. This returns a flowjo\_workspace, which can be parsed using the `flowjo_to_gatingset()` method. The function can be called non-interactively by passing the index or name of the group of samples to be imported via `flowjo_to_gatingset(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**

[fj\\_ws\\_get\\_sample\\_groups,GatingSet](#)

**Examples**

```
## Not run:
#f is a xml file name of a flowJo workspace
ws <- open_flowjo_xml(f)
#parse the second group
gs <- flowjo_to_gatingset(ws, name = 2); #assume that the fcs files are under the same folder as workspace

gs <- flowjo_to_gatingset(ws, name = 4
                        , path = dataDir #specify the FCS path
                        , subset = "CytoTrol_CytoTrol1.fcs" #subset the parsing by FCS filename
                        , isNcdf = FALSE)#turn off cdf storage mode (normally you don't want to do this for parsing large files)

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

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

#override the default compensation defined in xml with the customized compensations
gs <- flowjo_to_gatingset(ws, name = 2, compensation = comps); #comp is either a compensation object or a list of compensation objects

## 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)

**Value**

nothing

**Examples**

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

## End(Not run)
```

---

range.GatingHierarchy *the parameter range from the flow data associated with GatingHierarchy*

---

**Description**

the parameter range from the flow data associated with GatingHierarchy

**Usage**

```
## S3 method for class 'GatingHierarchy'
range(..., na.rm = FALSE,
      type = c("instrument", "data"), raw.scale = FALSE)
```

**Arguments**

...	GatingHierarchy object
na.rm	not used
type	character of "instrument" or "data" indicating whether to retrieve the instrument or the actual data range
raw.scale	logical whether convert the range from transformed scale to raw scale

**Value**

matrix

**Examples**

```
## Not run:
range(gh, type = "data")#return data range
range(gh) #return instrument range
range(gh, raw.scale = TRUE) #inverse transform the range to the raw scale

## End(Not run)
```

---

read.gatingML.cytobank

*Parser for gatingML exported by Cytobank*

---

**Description**

The Default parser (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 = "CytoML")
g <- read.gatingML.cytobank(xml) #parse the population tree
#plot(g) #visualize it

## 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

**Value**

nothing

# Index

- addCustomInfo, 3
- closeWorkspace (open\_flowjo\_xml), 20
- compensate, GatingSet, graphGML-method, 3
- constructTree, 4
- cytobank2GatingSet (cytobank\_to\_gatingset.default), 5
- cytobank\_to\_gatingset (cytobank\_to\_gatingset.default), 5
- cytobank\_to\_gatingset.default, 5
- cytobankExperiment, 4
- CytoML.par.get (CytoML.par.set), 6
- CytoML.par.init, 6
- CytoML.par.set, 6
  
- diva\_get\_sample\_groups (diva\_workspace-class), 7
- diva\_get\_samples (diva\_workspace-class), 7
- diva\_to\_gatingset (diva\_workspace-class), 7
- diva\_workspace-class, 7
  
- extend, 7
  
- fj\_ws\_get\_keywords (getKeywords), 14
- fj\_ws\_get\_sample\_groups, 24
- fj\_ws\_get\_sample\_groups (getSampleGroups), 16
- fj\_ws\_get\_samples (getSamples), 17
- flowjo\_to\_gatingset, 14, 17
- flowjo\_to\_gatingset (parseWorkspace, flowjo\_workspace-method), 22
- flowjo\_workspace-class, 7, 9
- flowjo\_ws\_close (open\_flowjo\_xml), 20
  
- gating, graphGML, GatingSet-method, 9
- GatingHierarchy, 9
- GatingSet, 9, 24
  
- GatingSet (GatingSet, character, character-method), 10
- GatingSet, character, character-method, 10
- GatingSet2cytobank, 10
- GatingSet2flowJo, 11
- gatingset\_to\_cytobank (GatingSet2cytobank), 10
- gatingset\_to\_flowjo (GatingSet2flowJo), 11
- getChildren, graphGML, character-method, 12
- getCompensationMatrices.graphGML, 13
- getFJWSubsetIndices, 13
- getGate, graphGML, character-method, 14
- getKeywords, 14
- getNodes, graphGML-method, 15
- getParent, graphGML, character-method, 16
- getSampleGroups, 16
- getSamples, 17
- getTransformations.graphGML, 18
- graphGML-class, 18
- gs\_compare\_cytobank\_counts, 18
  
- matchPath, 19
  
- open\_cytobank\_experiment (cytobankExperiment), 4
- open\_diva\_xml (openDiva), 20
- open\_flowjo\_xml, 17, 20
- openDiva, 20
  
- parse.gateInfo, 21
- parseWorkspace, diva\_workspace-method (diva\_workspace-class), 7
- parseWorkspace, flowjo\_workspace-method, 22
- plot, graphGML, missing-method, 24
  
- range.GatingHierarchy, 25
- read.flowSet, 23
- read.gatingML.cytobank, 26

`read.ncdfFlowSet`, [23](#)

`show, diva_workspace-method`  
    (`diva_workspace-class`), [7](#)

`show, flowjo_workspace-method`  
    (`flowjo_workspace-class`), [9](#)

`show, graphGML-method`, [27](#)

`xmlTreeParse`, [9](#), [10](#), [20](#), [21](#), [23](#)