Package 'ncdfFlow'

April 16, 2024

Title ncdfFlow: A package that provides HDF5 based storage for flow cytometry data.

Version 2.48.0

Author Mike Jiang,Greg Finak,N. Gopalakrishnan

Description Provides HDF5 storage based methods and functions for manipulation of flow cytometry data.

Maintainer Mike Jiang <mike@ozette.com>

Depends R (>= 2.14.0), flowCore(>= 1.51.7), methods, BH

Imports Biobase,BiocGenerics,flowCore,zlibbioc

Suggests testthat,parallel,flowStats,knitr

License AGPL-3.0-only

biocViews ImmunoOncology, FlowCytometry

LinkingTo cpp11,BH, Rhdf5lib

VignetteBuilder knitr

RoxygenNote 7.1.1

git_url https://git.bioconductor.org/packages/ncdfFlow

git_branch RELEASE_3_18

git_last_commit 8367dea

git_last_commit_date 2023-10-24

Repository Bioconductor 3.18

Date/Publication 2024-04-15

R topics documented:

as.flowSet 2
clone.ncdfFlowSet 3
filter,ncdfFlowList,filter-method 4
getFileName 6
getIndices,ncdfFlowSet,character-method 7
lapply,ncdfFlowList-method 8

ncdfFlow 8
ncdfFlowList-class 9
ncdfFlowSet,flowSet-method 10
ncdfFlowSet-class 11
ncfsApply,ncdfFlowSet-method 12
rbind2,ncdfFlowList,ANY-method 13
read.ncdfFlowSet 14
replacement method for ncdfFlowSet 15
save_ncfs 17
split,ncdfFlowList,factor-method 18
Subset,ncdfFlowSet,filterResultList-method 19
subset.ncdfFlowSet 19
unlink,ncdfFlowSet-method 20
[,ncdfFlowSet,ANY-method 21
[[,ncdfFlowSet,ANY-method 21

Index **23**

`as.flowSet` *convert from a* `ncdfFlowSet` *to a* `flowSet`

Description

The main purpose of this API is to convert the archived data (stored in `ncdfFlowSet`) to `flowSet` when the speed is more concerned than memory effieciency. Although `ncdfFlowSet` is designed to minimize the disk-IO cost, so usually it is not necessary to do such coersion.

Usage

```
as.flowSet(from, top)
```

Arguments

`from`	a `ncdfFlowSet`
`top`	`integer` specifies a certain number of samples are evenly selected for the coersion. If this argument is missing, then coerce all the samples within the `ncdfFlowSet`. It is to be used with caution because it can incur the huge memory consumption given the `flowSet` is all-in-memory data structure.

Examples

```
data(GvHD)
nc1 <- ncdfFlowSet(GvHD[1:4])
fs <- as.flowSet(nc1)
```

clone.ncdfFlowSet *Clone a ncdfFlowSet*

Description

Create a new ncdfFlowSet object from an existing one

Usage

```
clone.ncdfFlowSet(
  ncfs,
  ncdfFile = NULL,
  isEmpty = FALSE,
  isNew = TRUE,
  dim = 2,
  compress = 0
)
```

Arguments

ncfs	A ncdfFlowSet.
ncdfFile	A character scalar giving the output file name. By default, It is NULL and the function will generate a random file name, potentially adding the .cdf suffix unless a file extension is already present. It is only valid when isNewNcFile=TRUE
isEmpty	A logical scalar indicating whether the raw data should also be copied.if FALSE, an empty cdf file is created with the same dimensions (sample*events*channels) as the orignial one.
isNew	A logical scalar indicating whether the new cdf file should be created. If FALSE, the original cdf file is associated with the new ncdfFlowSet object.
dim	integer see details in read.ncdfFlowset.
compress	integer see details in read.ncdfFlowset.

Value

A ncdfFlowSet object

See Also

read.ncdfFlowSet

Examples

```
path<-system.file("extdata","compdata","data",package="flowCore")
files<-list.files(path,full.names=TRUE)[1:3]

#create ncdfFlowSet from fcs
```

```
nc1 <- read.ncdfFlowSet(files=files,ncdfFile="ncfsTest.nc",flowSetId="fs1",isWriteSlice= TRUE)

##clone the ncdfFlowSet object
nc2<-clone.ncdfFlowSet(nc1,"clone.nc")
nc2[[1]]

#optionally create the empty hdf file without writting the acutal flow data into it
nc2 <- clone.ncdfFlowSet(nc1,"clone.nc", isEmpty = TRUE)

#add the actual raw data
fs1  <- read.flowSet(files=files)
nc2[[sampleNames(fs1)[1]]] <- fs1[[1]]
nc2[[1]]

#delete the cdf file associated with ncdfFlowSet before removing it from memory
unlink(nc2)
rm(nc2)

unlink(nc1)
rm(nc1)
```

---

filter,ncdfFlowList,filter-method

Accessors compatible with those for flowSet

Description

Accessors compatible with those for flowSet

Usage

```
## S4 method for signature 'ncdfFlowList,filter'
filter(
  x,
  filter,
  method = "missing",
  sides = "missing",
  circular = "missing",
  init = "missing"
)

## S4 method for signature 'ncdfFlowList,filterList'
filter(
  x,
  filter,
  method = "missing",
  sides = "missing",
  circular = "missing",
```

```
  init = "missing"
)

## S4 method for signature 'ncdfFlowList,list'
filter(
  x,
  filter,
  method = "missing",
  sides = "missing",
  circular = "missing",
  init = "missing"
)

## S4 method for signature 'ncdfFlowList'
length(x)

## S4 method for signature 'ncdfFlowList'
sampleNames(object)

## S4 method for signature 'ncdfFlowList'
phenoData(object)

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

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

## S4 method for signature 'ncdfFlowList'
colnames(x)

## S4 replacement method for signature 'ncdfFlowList'
colnames(x) <- value

## S4 method for signature 'ncdfFlowList'
markernames(object)

## S4 replacement method for signature 'ncdfFlowList'
markernames(object) <- value

## S4 method for signature 'ncdfFlowSet,ANY'
compensate(x, spillover)

## S4 method for signature 'flowSet,data.frame'
compensate(x, spillover)

## S4 method for signature 'ncdfFlowSet,list'
compensate(x, spillover)
```

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

## S4 replacement method for signature 'ncdfFlowSet,ANY'
sampleNames(object) <- value

## S4 replacement method for signature 'ncdfFlowSet'
colnames(x) <- value

## S4 method for signature 'ncdfFlowSet,list'
keyword(object, keyword)

## S4 replacement method for signature 'ncdfFlowSet,list'
keyword(object) <- value
```

Arguments

x	ncdfFlowSet
filter	filter to be applied
method	missing not used
sides	missing not used
circular	missing not used
init	missing not used
object	ncdfFlowList
value	character vector
spillover	spillover matrix
_data	ncdfFlowSet
translist	a 'transformList' object or a list of 'transformList' objects
...	other arguments
keyword	list

---

getFileName *get the cdf file name associated with ncdfFlowSet object*

Description

get the cdf file name associated with ncdfFlowSet object

Usage

```
getFileName(ncfs)
```

Arguments

ncfs	ncdfFlowSet

Value

character

---

getIndices,ncdfFlowSet,character-method

getIndices *extracts the event indices of one or multiple samples from ncdfFlowSet*

Description

These functions are mainly for internal usage and normally not to be used by users.

Usage

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

## S4 method for signature 'ncdfFlowSet'
initIndices(obj)

## S4 method for signature 'ncdfFlowSet,character,logical'
updateIndices(obj, y, z)
```

Arguments

obj	ncdfFlowSet object
y	character sample name
z	logical vector to be assigned.

Value

a logical vector.

Examples

```
data(GvHD)
nc <- ncdfFlowSet(GvHD[1:2])
sn <- sampleNames(nc)[1]
nrow(nc[[sn]])
getIndices(nc, sn) #initial index is NA
#subset with filter
library(flowStats)
morphGate <- norm2Filter("FSC-H", "SSC-H", filterId = "MorphologyGate",scale = 2)
```

```
nc1 <- Subset(nc, morphGate)
ind <- getIndices(nc1, sn)
# all.equal(sum(ind), nrow(nc1[[sn]]))
initIndices(nc1)
getIndices(nc1, sn) #reset indices
```

`lapply,ncdfFlowList-method` *lapply method for ncdfFlowList*

Description

Depending on `level` parameter, loop either iterates through the list of ncdfFlowSet objects or everyflowFrame objects.

Usage

```
## S4 method for signature 'ncdfFlowList'
lapply(X, FUN, level = 2, ...)
```

Arguments

`X`	`ncdfFlowList` object
`FUN`	`function` to apply
`level`	`numeric`. It controls whether loop at 'ncdfFlowSet' level or 'sample' level. when level = 2 (default value),`FUN` is applied to each sample. When level = 1, `FUN` is applied to each object stored in `data` slot.
`...`	other arguments passed to `FUN`

`ncdfFlow` *ncdfFlow: A package that provides CDF storage based flow cytometry data analysis.*

Description

ncdfFlow: A package that provides CDF storage based flow cytometry data analysis.

Details

Define important flow cytometry data classes: `ncdfFlowSet`(a subclass of `flowSet`) and `ncdfFlowList`(a list of ncdfFlowSet object) and their accessors.

Provide important compensation,transformation,filter,gating,subsetting,splitting functions for data analysis of large volumns of flow cytometry data that is too big to be held in memory.

Package:	ncdfFlow
Version:	2.9.24
Date:	2014-04-16
Depends:	R (>= 2.8.1), flowCore
License:	Artistic-2.0

Author(s)

Mike Jiang <mike@ozette.ai>, Greg Finak <greg@ozette.ai>

Maintainer: Mike Jiang <mike@ozette.ai>

`ncdfFlowList-class` *a class that stores multiple ncdfFlowSet objects*

Description

It is a list of ncdfFlowSet objects

Usage

```
ncdfFlowList(x, samples = NULL)

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

Arguments

`x`	`list` of `ncdfFlowSet` objects
`samples`	`integer` see `samples` slot of `ncdfFlowList` class. or `character` that specifiy the order to samples. If not given then reconstruct the index.
`object`	`ncdfFlowList`

Value

`ncdfFlowList-class`

Objects from the Class

Objects can be created by coercing a list of ncdfFlowSet objects as("ncdfFlowList",nclist = ... #a list of ncdfFlowSet objects)

Slots

`data`**:** A list containing the ncdfFlowSet objects.

`samples`**:** A `integer` vector containing the index of the ncdfFlowSet object to which each sample belongs. The name of the vector is the sample names that determine the order of samples exposed to the user, which can be different from the physical storing order.

See Also

ncdfFlowSet

Examples

```
data(GvHD)
nc1 <- ncdfFlowSet(GvHD[1])
nc2 <- ncdfFlowSet(GvHD[2])
nc3 <- ncdfFlowSet(GvHD[3])
list1 <- list(nc1, nc2, nc3)
#coerce from list to ncdfFlowList
nclist <- ncdfFlowList(list1)
nclist
#coerce(collapse) from ncdfFlowList to a single flowFrame
collapsedData <- as(nclist, "flowFrame")
collapsedData
```

---

ncdfFlowSet,flowSet-method

create ncdfFlowSet from flowSet

Description

Normally the ncdfFlowSet is constructed by loading raw FCS files using read.ncdfFlowSet. In case there is a legacy flowSet object, we can convert it to ncdfFlowSet with this constructor.

Usage

```
## S4 method for signature 'flowSet'
ncdfFlowSet(x, ncdfFile, dim = 2, compress = 0)
```

Arguments

x	flowSet
ncdfFile	character specifies the file name of cdf file
dim	integer see details in read.ncdfFlowset.
compress	integer see details in read.ncdfFlowset.

Examples

```
data(GvHD)
fs <- GvHD[1:2]
ncfs <- ncdfFlowSet(fs)
```

ncdfFlowSet-class *a class for storing flow cytometry raw data in HDF5 format*

Description

This class is a subclass of flowSet. It stores the raw data in cdf file instead of memory so that the analysis tools provided by flowCore based packages can be used in the study that produces hundreds or thousands FCS files.

Usage

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

Arguments

object	ncdfFlowSet show,ncdfFlowSet-method

Slots

file: A character containing the ncdf file name.

maxEvents: An integer containing the maximum number of events of all samples stored in this ncdfFlowSet object

flowSetId: A character for the id of ncdfFlowSet object

indices: Object of class "environment" containing events indices of each sample stored as "raw" vector. Each index value is either TURE or FALSE and the entire indices vector is used to subset the raw data. the indices vector of each sample is NA by default when the ncdfFlowSet first created.It is assigned with actual value when ncdfFlowSet object is subsetted by Subset or other subsetting methods.

origSampleVector: A character vector containing the sample names, which indicates the original order of samples physically stored in cdf format

origColnames: A character vector containing the flow channel names, which indicates the original order of columns physically stored in cdf format

frames: Object of class "environment", which replicates the "frame" slot in flowSet, except that exprs matrix is empty and the actual data is stored in cdf file.

phenoData: see phenoData

Extends

Class "flowSet", directly.

ncfsApply,ncdfFlowSet-method

apply method for ncdfFlowSet (for internal use)

Description

It is equivalent to fsApply. But the latter could cause memory issue when FUN returns a flowFrame. ncfsApply writes to a new cdf file instead of memory. Thus it will return a ncdfFlowSet object.

Usage

```
## S4 method for signature 'ncdfFlowSet'
ncfsApply(x, FUN, ..., use.exprs = FALSE, ncdfFile = NULL)
```

Arguments

x	ncdfFlowSet
FUN	function to apply
...	other arguments to pass to FUN
use.exprs	logical see fsApply
ncdfFile	A character scalar giving the output file name. By default, It is NULL and the function will generate a random file name, potentially adding the .cdf suffix unless a file extension is already present.

Details

When the function given by argument "FUN" does not return the entire flowFrame object with the same size of the original one (such as compensate,transform...), fsApply should be used instead.

Examples

```
data(GvHD)
nc <- ncdfFlowSet(GvHD[1:2])

#use fsApply when FUN does not return a flowFrame
fsApply(nc, nrow)
fsApply(nc, range)

#use ncfsApply when FUN returns a flowFrame
lgcl <- logicleTransform( w = 0.5, t= 10000, m =4.5)
translist <- transformList(c("FL1-H", "FL2-H"), lgcl)
nc1 <- ncfsApply(nc, transform, translist)
```

rbind2,ncdfFlowList,ANY-method

combine multiple ncdfFlowSet objects into one

Description

Similar to flowCore:rbind2. But one needs to first construct a ncdfFlowList and then apply rbind2 to it instead of merging them pairwise

Usage

```
## S4 method for signature 'ncdfFlowList,ANY'
rbind2(
  x,
  ncdfFile = tempfile(pattern = "ncfs"),
  dim = 2,
  compress = 0,
  samples = NULL
)
```

Arguments

x	ncdfFlowList
ncdfFile	character see details in read.ncdfFlowset when all the ncdfFlowSets shared the same cdf file, by supplying this argument, it will use the existing cdf and avoid writing to it unneccessarily.
dim	integer see details in read.ncdfFlowset.
compress	integer see details in read.ncdfFlowset.
samples	character the vector of sample names which determine the physical sample storage order in original cdf file. Default is NULL, which derives from the given ncdfFlowSet objects.

Value

a new ncdfFlowSet with a new cdf file that combines multiple raw datasets.

Examples

```
library(ncdfFlow)
data(GvHD)

nc1 <- ncdfFlowSet(GvHD[1:2])
nc2 <- ncdfFlowSet(GvHD[3:4])
nc3 <- ncdfFlowSet(GvHD[5:6])
ncfslist <- ncdfFlowList(list(nc1,nc2,nc3))
nc4 <- rbind2(ncfslist)
nc4
```

`read.ncdfFlowSet` *create ncdfFlowSet from FCS files*

Description

read FCS files from the disk and load them into a ncdfFlowSet object

Usage

```
read.ncdfFlowSet(
  files = NULL,
  ncdfFile,
  flowSetId = flowCore:::guid(),
  isWriteSlice = TRUE,
  phenoData,
  channels = NULL,
  channel_alias = NULL,
  alter.names = FALSE,
  dim = 2,
  compress = 0,
  mc.cores = NULL,
  ...
)
```

Arguments

`files`	A character vector giving the source FCS raw file paths.
`ncdfFile`	A character scalar giving the output file name. Default is NULL and the function will generate a random file in the temporary folder, potentially adding the `.cdf` suffix unless a file extension is already present. It is sometimes useful to specify this file path to avoid the failure of writing large flow data set to cdf file due to the the shortage of disk space in system temporary folder. It is only valid when isNewNcFile=TRUE
`flowSetId`	A character scalar giving the unique ncdfFlowSet ID.
`isWriteSlice`	A logical scalar indicating whether the raw data should also be copied.if FALSE, an empty cdf file is created with the dimensions (sample*events*channels) supplied by raw FCS files.
`phenoData`	An object of `AnnotatedDataFrame` providing a way to manually set the phenotyoic data for the whole data set in ncdfFlowSet.
`channels`	A character vector specifying which channels to extract from FCS files. It can be useful when FCS files do not share exactly the same channel names. Thus this argument is used to select those common channels that are of interests. Default value is NULL and the function will try to scan the FCS headers of all files and determine the common channels.
`channel_alias, alter.names`	see read.FCS

dim	integer the number of dimensions that specifies the physical storage format of hdf5 dataset. Default is 2, which stores each FCS data as a seperate 2d dataset. Normally, user shouldn't need to change this but dim can also be set to 3, which stores all FCS data as one single 3d dataset.
compress	integer the HDF5 compression ratio (from 0 to 9). Default is 0, which does not compress the data and is recommeneded (especially for 2d format) because the speed loss usually outweights the disk saving.
mc.cores	numeric passed to parallel::mclapply. Default is NULL, which read FCS files in serial mode.
...	extra arguments to be passed to read.FCS.

Value

A ncdfFlowSet object

See Also

clone.ncdfFlowSet

Examples

```
library(ncdfFlow)

path<-system.file("extdata","compdata","data",package="flowCore")
files<-list.files(path,full.names=TRUE)[1:3]

#create ncdfFlowSet from fcs with the actual raw data written in cdf
nc1 <- read.ncdfFlowSet(files=files,ncdfFile="ncfsTest.nc",flowSetId="fs1",isWriteSlice= TRUE)
nc1
nc1[[1]]
unlink(nc1)
rm(nc1)

#create empty ncdfFlowSet from fcs and add data slices afterwards
nc1 <- read.ncdfFlowSet(files=files,ncdfFile="ncfsTest.nc",flowSetId="fs1",isWriteSlice= FALSE)
fs1<-read.flowSet(files)
nc1[[1]] <- fs1[[1]]
nc1[[1]]
nc1[[2]]
```

`replacement method for ncdfFlowSet`

write the flow data from a `flowFrame` *to* `ncdfFlowSet` *flowFrame can have less channels than ncdfFlowSet,which is used for partial updating(useful for* `normalization`*)*

Description

write the flow data from a flowFrame to ncdfFlowSet

flowFrame can have less channels than ncdfFlowSet,which is used for partial updating(useful for normalization)

Usage

```
## S4 replacement method for signature 'ncdfFlowSet,ANY,ANY,flowFrame'
x[[i, j = "missing", compress = 0, ...]] <- value
```

Arguments

x	a ncdfFlowSet
i	a numeric or character used as sample index of ncdfFlowSet
j	not used
compress	integer It is only relevant to writing slice to '2d' format because the compression is set during the creation of hdf5 file for '3d' format. see details in read.ncdfFlowset.
...	not used
value	flowFrame

Examples

```
data(GvHD)
nc <- ncdfFlowSet(GvHD[1:2])
samples <- sampleNames(nc)
sn <- samples[1]
#return the entire flowFrame
fr <- nc[[sn]]

apply(exprs(nc[[sn]]), 2, range)

#transform the data
lgcl <- logicleTransform( w = 0.5, t= 10000, m =4.5)
fr_trans <- transform(fr, `FL1-H` = lgcl(`FL1-H`), `FL2-H` = lgcl(`FL2-H`))

#update the data
nc[[sn]] <- fr_trans
apply(exprs(nc[[sn]]), 2, range)

#subset on channels
nc1 <- nc[,2:3]
#only write the channels of interest (reduce disk IO)
nc1[[sn]] <- fr_trans[,2:3]

#chanel colnames
colnames(fr_trans)[3:4] <- c("<FL1-H>", "<FL2-H>")

#write data without matching up the colnames will fail
#nc[[sn]] <- fr_trans
```

save_ncfs *save/load a ncdfFlowSet object to/from disk.*

Description

The ncdfFlowSet object contains two parts: R object and cdf file. Save/load a ncdfFlowSet mainly involves the R part using saveRDS/readRDS.

Usage

```
save_ncfs(
  ncfs,
  path,
  overwrite = FALSE,
  cdf = c("copy", "move", "link", "skip", "symlink")
)

load_ncfs(path)
```

Arguments

ncfs	A ncdfFlowSet
path	A character scalar giving the path to save/load the ncdfFlowSet 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.

Value

load_ncfs returns a ncdfFlowSet object

See Also

ncdfFlowSet-class

Examples

```
## Not run:
#ncfs is a ncdfFlowSet
save_ncfs(fs, path = "tempFolder")
fs1 <- load_ncfs(path = "tempFolder")


## End(Not run)
```

split,ncdfFlowList,factor-method
split a ncdfFlowSet *object.*

Description

Equivalent to split method for flowSet object.

Usage

```
## S4 method for signature 'ncdfFlowList,factor'
split(x, f, drop = FALSE, ...)

## S4 method for signature 'ncdfFlowList,character'
split(x, f, drop = FALSE, ...)

## S4 method for signature 'ncdfFlowSet,filter'
split(x, f, drop = FALSE, population = NULL, prefix = NULL, ...)

## S4 method for signature 'ncdfFlowSet,filterResultList'
split(x, f, drop = FALSE, population = NULL, prefix = NULL, ...)

## S4 method for signature 'ncdfFlowSet,list'
split(x, f, isNew = FALSE, drop = FALSE, population = NULL, prefix = NULL, ...)

## S4 method for signature 'ncdfFlowSet,factor'
split(x, f, isNew = FALSE, drop = FALSE, ...)

## S4 method for signature 'ncdfFlowSet,character'
split(x, f, drop = FALSE, ...)
```

Arguments

x	ncdfFlowSet
f, drop, population, prefix, ...	see split-methods
isNew	logical wehther to create a new hdf file or using existing hdf file.

Value

a list of ncdfFlowSet objects that may not may not share the same hdf file depending on isNew argument.

Subset,ncdfFlowSet,filterResultList-method *subset a ncdfFlowSet by filter*

Description

Equivalent to Subset method for flowSet.

Usage

```
## S4 method for signature 'ncdfFlowSet,filterResultList'
Subset(x, subset, select, ...)

## S4 method for signature 'ncdfFlowList,filterResultList'
Subset(x, subset, select, ...)

## S4 method for signature 'ncdfFlowSet,filter'
Subset(x, subset, ...)

## S4 method for signature 'ncdfFlowList,filter'
Subset(x, subset, ...)

## S4 method for signature 'ncdfFlowSet,list'
Subset(x, subset, select, validityCheck = TRUE, ...)
```

Arguments

x	ncdfFlowSet or ncdfFlowList
subset, select, ...	see Subset-methods
validityCheck	logical whether to skip validity check for speed.

Value

one or more ncdfFlowSet objects which share the same hdf5 file with the original one.

subset.ncdfFlowSet *subset the ncdfFlowSet/ncdfFlowList based on 'pData'*

Description

subset the ncdfFlowSet/ncdfFlowList based on 'pData'

Usage

```
## S3 method for class 'ncdfFlowSet'
subset(x, subset, ...)

## S3 method for class 'ncdfFlowList'
subset(x, subset, ...)
```

Arguments

x	ncdfFlowSet or ncdfFlowList
subset	logical expression(within the context of pData) indicating samples to keep. see subset
...	other arguments. (not used)

Value

a subset of codencdfFlowSet or ncdfFlowList object

---

unlink,ncdfFlowSet-method

delete the cdf file associated with the ncdfFlowSet object ncdfFlowSet object is unrecoverable after cdf is deleted. So this method is usually called when ncdfFlowSet object is no longer in need.

Description

delete the cdf file associated with the ncdfFlowSet object

ncdfFlowSet object is unrecoverable after cdf is deleted. So this method is usually called when ncdfFlowSet object is no longer in need.

Usage

```
## S4 method for signature 'ncdfFlowSet'
unlink(x, recursive = FALSE, force = FALSE)
```

Arguments

x	ncdfFlowSet
recursive	see unlink
force	see unlink

Examples

```
data(GvHD)
nc <- ncdfFlowSet(GvHD[1:2])
nc[[1]] # data is loaded from cdf file
unlink(nc)
```

[,ncdfFlowSet,ANY-method *subsetting by sampleNames,channels(not for events) methods*

Description

similar to [.

Usage

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

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

Arguments

x	ncdfFlowSet
i	sample index(or name)
j	column(or channel) index (or name)
...	other arguments not used
drop	logical not used.

Examples

```
data(GvHD)
nc <- ncdfFlowSet(GvHD[1:2])
samples <- sampleNames(nc)
nc[1]
nc1 <- nc[samples[1]]
#nc1 and nc share the cdf file
all.equal(getFileName(nc1), getFileName(nc))
```

[[,ncdfFlowSet,ANY-method *extract a* flowFrame *object from* ncdfFlowSet

Description

Simliar to [[, and there are cerntain ways to reduce the disk IO and optimize the speed.

Usage

```
## S4 method for signature 'ncdfFlowSet,ANY'
x[[i, j, use.exprs = TRUE, ...]]

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

## S4 method for signature 'ncdfFlowList,logical'
x[[i, j, ...]]

## S4 method for signature 'ncdfFlowList,character'
x[[i, j, ...]]
```

Arguments

x	a ncdfFlowSet or ncdfFlowList
i	a numeric or character used as sample index
j	a numeric or character used as channel index
use.exprs	a logical scalar indicating whether to read the actual data from cdf
...	other arguments. not used.

Examples

```
data(GvHD)
nc <- ncdfFlowSet(GvHD[1:2])
samples <- sampleNames(nc)
sn <- samples[1]
#return the entire flowFrame
fr <- nc[[sn]]

#access the flowFrame meta data without loading the raw event data from disk
nc[[sn, use.exprs = FALSE]]

#only read a subset of channels (more efficient than reading entire data set)
nc[[sn, 1:2]]
```

Index

∗ **package**
    ncdfFlow, 8
[, *21*
[,ncdfFlowList,ANY-method
        ([,ncdfFlowSet,ANY-method), 21
[,ncdfFlowSet,ANY-method, 21
[[, *21*
[[,ncdfFlowList,character,missing-method
        ([[,ncdfFlowSet,ANY-method), 21
[[,ncdfFlowList,character-method
        ([[,ncdfFlowSet,ANY-method), 21
[[,ncdfFlowList,logical-method
        ([[,ncdfFlowSet,ANY-method), 21
[[,ncdfFlowList,numeric-method
        ([[,ncdfFlowSet,ANY-method), 21
[[,ncdfFlowSet,ANY-method, 21
[[<-,ncdfFlowSet,ANY,ANY,flowFrame-method
        *(*replacement method for
        ncdfFlowSet*)*, 15
[[<-,ncdfFlowSet,flowFrame-method
        *(*replacement method for
        ncdfFlowSet*)*, 15

as.flowSet, 2

clone.ncdfFlowSet, 3, *15*
colnames,ncdfFlowList-method
        *(*filter,ncdfFlowList,filter-method*)*,
        4
colnames<-
        *(*filter,ncdfFlowList,filter-method*)*,
        4
colnames<-,ncdfFlowList-method
        *(*filter,ncdfFlowList,filter-method*)*,
        4
colnames<-,ncdfFlowSet,ANY-method
        *(*filter,ncdfFlowList,filter-method*)*,
        4
colnames<-,ncdfFlowSet-method
        *(*filter,ncdfFlowList,filter-method*)*,
        4
compensate,flowSet,data.frame-method
        *(*filter,ncdfFlowList,filter-method*)*,
        4
compensate,ncdfFlowSet,ANY-method
        *(*filter,ncdfFlowList,filter-method*)*,
        4
compensate,ncdfFlowSet,list-method
        *(*filter,ncdfFlowList,filter-method*)*,
        4

exprs, *11*

filter,ncdfFlowList,filter-method, 4
filter,ncdfFlowList,filterList-method
        *(*filter,ncdfFlowList,filter-method*)*,
        4
filter,ncdfFlowList,list-method
        *(*filter,ncdfFlowList,filter-method*)*,
        4
flowCore:rbind2, *13*
flowSet, *8*, *11*
fsApply, *12*

getFileName, 6
getIndices
        *(*getIndices,ncdfFlowSet,character-method*)*,
        7
getIndices,ncdfFlowSet,character-method,
        7

initIndices
        *(*getIndices,ncdfFlowSet,character-method*)*,
        7
initIndices,ncdfFlowSet-method
        *(*getIndices,ncdfFlowSet,character-method*)*,
        7

keyword,ncdfFlowSet,list-method
        *(*filter,ncdfFlowList,filter-method*)*,
        4

keyword<-,ncdfFlowSet,list-method
(filter,ncdfFlowList,filter-method), 4

lapply,ncdfFlowList-method, 8
length,ncdfFlowList-method
(filter,ncdfFlowList,filter-method), 4
load_ncfs (save_ncfs), 17

markernames,ncdfFlowList-method
(filter,ncdfFlowList,filter-method), 4
markernames<-,ncdfFlowList-method
(filter,ncdfFlowList,filter-method), 4

ncdfFlow, 8
ncdfFlowList, *8*
ncdfFlowList (ncdfFlowList-class), 9
ncdfFlowList-class, 9
ncdfFlowSet, *3*, *8–10*
ncdfFlowSet
(ncdfFlowSet,flowSet-method), 10
ncdfFlowSet,flowSet-method, 10
ncdfFlowSet-class, 11
ncfsApply
(ncfsApply,ncdfFlowSet-method), 12
ncfsApply,ncdfFlowSet-method, 12

pData,ncdfFlowList-method
(filter,ncdfFlowList,filter-method), 4
pData<-,ncdfFlowList,data.frame-method
(filter,ncdfFlowList,filter-method), 4
phenoData, *11*
phenoData,ncdfFlowList-method
(filter,ncdfFlowList,filter-method), 4
phenoData<-,ncdfFlowList,AnnotatedDataFrame-method
(filter,ncdfFlowList,filter-method), 4

rbind2,ncdfFlowList,ANY-method, 13
read.FCS, *14*, *15*
read.ncdfFlowSet, *3*, 14
read.ncdfFlowset, *3*, *10*, *13*, *16*
read.ncdfFlowset (read.ncdfFlowSet), 14
replacement method for ncdfFlowSet, 15

sampleNames,ncdfFlowList-method
(filter,ncdfFlowList,filter-method), 4
sampleNames<-
(filter,ncdfFlowList,filter-method), 4
sampleNames<-,ncdfFlowSet,ANY-method
(filter,ncdfFlowList,filter-method), 4
save_ncfs, 17
show (ncdfFlowSet-class), 11
show,ncdfFlowList-method
(ncdfFlowList-class), 9
show,ncdfFlowSet-method
(ncdfFlowSet-class), 11
split,ncdfFlowList,character-method
(split,ncdfFlowList,factor-method), 18
split,ncdfFlowList,factor-method, 18
split,ncdfFlowSet,character-method
(split,ncdfFlowList,factor-method), 18
split,ncdfFlowSet,factor-method
(split,ncdfFlowList,factor-method), 18
split,ncdfFlowSet,filter-method
(split,ncdfFlowList,factor-method), 18
split,ncdfFlowSet,filterResultList-method
(split,ncdfFlowList,factor-method), 18
split,ncdfFlowSet,list-method
(split,ncdfFlowList,factor-method), 18
Subset, *11*
subset, *20*
Subset,ncdfFlowList,filter-method
(Subset,ncdfFlowSet,filterResultList-method), 19
Subset,ncdfFlowList,filterResultList-method
(Subset,ncdfFlowSet,filterResultList-method), 19
Subset,ncdfFlowSet,filter-method
(Subset,ncdfFlowSet,filterResultList-method), 19

Subset,ncdfFlowSet,filterResultList-method,
        19
Subset,ncdfFlowSet,list-method
        (Subset,ncdfFlowSet,filterResultList-method),
        19
subset.ncdfFlowList
        (subset.ncdfFlowSet), 19
subset.ncdfFlowSet, 19

transform,ncdfFlowSet-method
        (filter,ncdfFlowList,filter-method),
        4

unlink, *20*
unlink,ncdfFlowSet-method, 20
updateIndices
        (getIndices,ncdfFlowSet,character-method),
        7
updateIndices,ncdfFlowSet,character,logical-method
        (getIndices,ncdfFlowSet,character-method),
        7