

ncdfFlow

March 24, 2012

addFrame-methods *add/replace the data in ncdfFlowSet*

Description

Add one flowFrame to the ncdfFlowSet.

Usage

```
addFrame(ncfs, data, sampleName)
```

Arguments

ncfs	a ncdfFlowSet object
data	a flowFrame to be added
sampleName	a character object with the name of the target sample

Details

The dimensions of the flowFrame to be added has to match the target sample data in ncdfFlowSet. It updates the target sample data if it already exists in ncdfFlowSet object.

Author(s)

Mike Jiang, Greg Finak, N. Gopalakrishnan
Maintainer: Mike Jiang <wjiang2@fhcrc.org>

See Also

[read.ncdfFlowSet](#)

Examples

```

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

##create empty ncdfFlowSet from fcs
nc1 <- read.ncdfFlowSet(files=files,ncdfFile="ncfsTest.nc",flowSetId="fs1",isWriteSlice=
fs1<-read.flowSet(files);
#add the actual data slices afterwards
addFrame(nc1,fs1[[1]],sampleNames(fs1)[1])
nc1[[1]]##show the added flowFrame
nc1[[2]]##show empty flowFrame

```

clone.ncdfFlowSet *Clone a ncdfFlowSet*

Description

Create a new ncdfFlowSet object from an existing one

Usage

```

clone.ncdfFlowSet(ncfs, fileName=NULL, isEmpty=TRUE,
isNew=TRUE, isSaveMeta=FALSE)

```

Arguments

ncfs	A ncdfFlowSet .
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.
fileName	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 original one.
isSaveMeta	A logical scalar indicating whether the meta data other than raw data should be saved in cdf. It should be set as TRUE if the entire ncdfFlowSet is going to be loaded by ncdfFlowSet_open, character-method .

Value

A ncdfFlowSet object

Author(s)

Mike Jiang, Greg Finak, N. Gopalakrishnan
Maintainer: Mike Jiang <wjiang2@fhcrc.org>

See Also[read.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
nc1 <- read.ncdfFlowSet(isSaveMeta=FALSE,files=files,ncdfFile="ncfsTest.nc",flowSetId="f")

##clone the ncdfFlowSet object,by default the actual raw data is not added
nc2<-clone.ncdfFlowSet(nc1,"clone.nc")
nc2[[1]]

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

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

ncfsUnlink(nc1)
rm(nc1)

```

ncdfFlow-package *ncdfFlow: A package that provides netCDF storage based flow cytometry data analysis.*

Description

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.

Details

Package: ncdfFlow
 Version: 0.99.4
 Date: 2011-08-02
 Depends: R (>= 2.8.1), flowCore
 License: Artistic-2.0

Author(s)

Mike Jiang, Greg Finak, N. Gopalakrishnan
Maintainer: Mike Jiang <wjiang2@fhcrc.org>

References

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

ncdfFlowList-class *'ncdfFlowList'*: a class that stores multiple *ncdfFlowSet* objects

Description

It is a list of *ncdfFlowSet* objects

Objects from the Class

Objects can be created using `new("ncdfFlowList", sampleNames = ..., #character of sample names
datalist = ... #a list of ncdfFlowSet objects)`
or via the constructor, which takes a list of *ncdfFlowSet* objects: `ncdfFlowList(ncdfList)`

Slots

`sampleNames`: character of sample names
`datalist`: a list of *ncdfFlowSet* objects

Methods

[return a list of *ncdfFlowSet*s
[[return a specific *ncdfFlowSet* object
show display object summary.
sampleNames Extract and replace sample names from the *phenoData* object.
colnames Extract or replace the *colnames* slot.
ncfsUnlink delete the *cdf* files associated with *ncdfFlowSet* objects contained in current *ncdfFlowList*

Author(s)

Mike Jiang, Greg Finak, N. Gopalakrishnan
Maintainer: Mike Jiang <wjiang2@fhcrc.org>

See Also

[ncdfFlowSet](#)

```
ncdfFlowSet-class  'ncdfFlowSet': a class for storing flow cytometry raw data in netCDF
                  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.

Objects from the Class

Objects can be created by using: [read.ncdfFlowSet](#), [clone.ncdfFlowSet](#)
 or `ncdfFlowSet(x, #x is a flowSet fileName #the output cdf file name)`

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 TRUE 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](#)

colnames: see [colnames](#). Here it serves as the current data view which does not reflect the actual number and order of columns stored in cdf file.

Extends

Class "[flowSet](#)", directly.

Methods

Most of the other flowSet methods are not listed here, but they all work as the same due to its inheritance from flowSet. Please see for more [flowSet](#) details for these methods.

addFrame add or replace the flowFrame in ncdfFlowSet. See [addFrame](#) for more details

Usage:

```
addFrame(ncfs, data, sampleName)
```

`[,[]` Subsetting. similar to `[,[]` for `flowSet`.

getIndices extract the event indices of one or multiple samples from `ncdfFlowSet`, return a logical vector.

Usage:

```
getIndices(ncfs, sampleName)
```

initIndices initialize the event indices for the entire `ncdfFlowSet` with either an NA or TRUE/FALSE logical value

Usage:

```
initIndices(ncfs, y)
```

updateIndices update the event indices of the target sample in `ncdfFlowSet`

Usage:

```
updateIndices(ncfs, sampleName, y)
```

ncdfFlowSet_open load the `ncdfFlowSet` object from the cdf file, return a `ncdfFlowSet` object

Usage:

```
ncdfFlowSet_open(filename)
```

Note that in order to successfully recover the entire `ncdfFlowSet` object, the `phenoData` has to be already saved in cdf either by explicitly calling `ncdfFlowSet_sync` or setting `isSaveMeta` as TRUE when `ncdfFlowSet` is created by `link{read.ncdfFlowSet}` or `link{clone.ncdfFlowSet}`

ncdfFlowSet create `ncdfFlowSet` from a `flowSet` object

Usage:

```
ncdfFlowSet(fsl)
```

Note that only `flowSet` can be coerced to `ncdfFlowSet`, attempt to apply this method to `flowFrame` will get an error message.

ncdfFlowSet_sync save `phenoData` to cdf file.

Usage:

```
ncdfFlowSet_sync(filename)
```

ncfsUnlink delete the netCDF file associated with the `ncdfFlowSet` object

Usage:

```
ncfsUnlink(ncfs)
```

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

ncdfExprs Extract or replace the raw data intensities, equivalent to `exprs`. It reads the data matrix from cdf file instead of memory.

Usage:

```
ncdfExprs(ncfs, sampleName, channel, subByIndice)
```

NcdfFlowSetToFlowSet convert a subset of `ncdfFlowSet` to `flowSet`.

Usage:

```
NcdfFlowSetToFlowSet(ncfs, top)
```

Argument `top` specifies the number of samples evenly selected from `ncdfFlowSet`.

ncfsApply equivalent to `fsApply`., which could cause memory issue due to the. So `ncfsApply` will return a `ncdfFlowSet` object.

Usage:

```
ncfsApply(x="ncdfFlowSet", FUN="ANY")
```

Note that the function given by argument "FUN" should return an entire `flowFrame` object with the same size of the original one (such as `compensate`, `transform`...) Otherwise, `fsApply` should be used instead.

rbind2 equivalent to `flowCore:rbind2`. It returns a new `ncdfFlowSet` with a new cdf file that combines two raw datasets.

split equivalent to `flowCore:split`. It returns a new `ncdfFlowSet` object without creating new cdf raw data file but assigning logical indices to subset the original raw data.

Subset equivalent to `flowCore:Subset`. It returns a new `ncdfFlowSet` object without creating new cdf raw data file but assigning logical indices to subset the original raw data.

densityplot,xyplot equivalent to `flowViz:densityplot` and `flowViz:xyplot`. User need to be careful about applying these plot methods to `ncdfFlowSet` because it could be slow for large number of flow data.

Author(s)

Mike Jiang,Greg Finak,N.Gopalakrishnan

Maintainer: Mike Jiang <wjiang2@fhcrc.org>

See Also

[flowSet](#),[read.ncdfFlowSet](#),[ncdfFlowList](#)

`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,
isWriteSlice=TRUE,isSaveMeta=FALSE,phenoData)
```

Arguments

<code>files</code>	A character vector giving the source FCS raw file paths.
<code>ncdfFile</code>	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 <code>.cdf</code> suffix unless a file extension is already present. It is only valid when <code>isNewNcFile=TRUE</code>
<code>flowSetId</code>	A character scalar giving the unique <code>ncdfFlowSet</code> ID.
<code>isWriteSlice</code>	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.
<code>isSaveMeta</code>	A logical scalar indicating whether the meta data other than raw data should be saved in cdf. It should be set as TRUE if the entire <code>ncdfFlowSet</code> is going to be loaded by <code>ncdfFlowSet_open,character-method</code> .
<code>phenoData</code>	An object of <code>AnnotatedDataFrame</code> providing a way to manually set the phenotypic data for the whole data set in <code>ncdfFlowSet</code> .

Value

A ncdfFlowSet object

Author(s)

Mike Jiang, Greg Finak, N. Gopalakrishnan

Maintainer: Mike Jiang <wjiang2@fhcrc.org>

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(isSaveMeta=FALSE,files=files,ncdfFile="ncfsTest.nc",flowSetId="fcs")
nc1
nc1[[1]]
ncfsUnlink(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)
addFrame(nc1,fs1[[1]],sampleNames(fs1)[1])
nc1[[1]]
nc1[[2]]
```


Index

***Topic IO**
 clone.ncdfFlowSet, 2
 read.ncdfFlowSet, 7
***Topic classes**
 ncdFlowList-class, 4
 ncdFlowSet-class, 5
***Topic methods**
 addFrame-methods, 1
***Topic package**
 ncdFlow-package, 3
 [, 6
 [,ncdfFlowList,ANY-method
 (ncdfFlowList-class), 4
 [,ncdfFlowSet,ANY-method
 (ncdfFlowSet-class), 5
 [[, 6
 [[,ncdfFlowList,ANY-method
 (ncdfFlowList-class), 4
 [[,ncdfFlowSet,ANY-method
 (ncdfFlowSet-class), 5
 [[<- ,ncdfFlowSet,ANY,ANY,flowFrame-method
 (ncdfFlowSet-class), 5
 addFrame, 5
 addFrame (addFrame-methods), 1
 addFrame,ncdfFlowSet,flowFrame-method
 (addFrame-methods), 1
 addFrame-methods, 1
 clone.ncdfFlowSet, 2, 5, 8
 colnames, 5
 colnames,ncdfFlowList-method
 (ncdfFlowList-class), 4
 colnames<- ,ncdfFlowSet-method
 (ncdfFlowSet-class), 5
 compensate,ncdfFlowSet,ANY-method
 (ncdfFlowSet-class), 5
 densityplot,formula,ncdfFlowSet-method
 (ncdfFlowSet-class), 5
 exprs, 5, 6
 flowCore:rbind2, 7
 flowCore:split, 7
 flowCore:Subset, 7
 flowSet, 3, 5, 7
 flowViz:densityplot, 7
 flowViz:xyplot, 7
 fsApply, 6
 getIndices (ncdfFlowSet-class), 5
 getIndices,ncdfFlowSet,character-method
 (ncdfFlowSet-class), 5
 initIndices (ncdfFlowSet-class), 5
 initIndices,ncdfFlowSet,logical-method
 (ncdfFlowSet-class), 5
 ncdFlowExprs (ncdfFlowSet-class), 5
 ncdFlow (ncdfFlow-package), 3
 ncdFlow-package, 3
 ncdFlowList, 3, 7
 ncdFlowList
 (ncdfFlowList-class), 4
 ncdFlowList-class, 4
 ncdFlowSet, 2-4
 ncdFlowSet (ncdfFlowSet-class), 5
 ncdFlowSet,flowFrame-method
 (ncdfFlowSet-class), 5
 ncdFlowSet,flowSet-method
 (ncdfFlowSet-class), 5
 ncdFlowSet-class, 5
 ncdFlowSet_open
 (ncdfFlowSet-class), 5
 ncdFlowSet_open,character-
 method, 2,
 7
 ncdFlowSet_open,character-method
 (ncdfFlowSet-class), 5
 ncdFlowSet_sync, 6
 ncdFlowSet_sync
 (ncdfFlowSet-class), 5
 ncdFlowSet_sync,ncdfFlowSet-method
 (ncdfFlowSet-class), 5
 NcdFlowSetToFlowSet
 (ncdfFlowSet-class), 5
 NcdFlowSetToFlowSet,ncdfFlowSet-method
 (ncdfFlowSet-class), 5

`ncfsApply(ncdfFlowSet-class)`, 5
`ncfsApply`, `ncdfFlowSet`-method
 (`ncdfFlowSet-class`), 5
`ncfsUnlink(ncdfFlowSet-class)`, 5
`ncfsUnlink`, `ncdfFlowList`-method
 (`ncdfFlowList-class`), 4
`ncfsUnlink`, `ncdfFlowSet`-method
 (`ncdfFlowSet-class`), 5

`phenoData`, 5

`rbind2(ncdfFlowSet-class)`, 5
`rbind2`, `ncdfFlowSet`, `flowFrame`-method
 (`ncdfFlowSet-class`), 5
`rbind2`, `ncdfFlowSet`, `ncdfFlowSet`-method
 (`ncdfFlowSet-class`), 5
`read.ncdfFlowSet`, 1, 3, 5, 7, 7

`show`, `ncdfFlowList`-method
 (`ncdfFlowList-class`), 4
`show`, `ncdfFlowSet`-method
 (`ncdfFlowSet-class`), 5
`split(ncdfFlowSet-class)`, 5
`split`, `ncdfFlowSet`, `factor`-method
 (`ncdfFlowSet-class`), 5
`split`, `ncdfFlowSet`, `filter`-method
 (`ncdfFlowSet-class`), 5
`split`, `ncdfFlowSet`, `filterResultList`-method
 (`ncdfFlowSet-class`), 5
`split`, `ncdfFlowSet`, `list`-method
 (`ncdfFlowSet-class`), 5
`Subset`, 5
`Subset(ncdfFlowSet-class)`, 5
`Subset`, `ncdfFlowSet`, `filter`-method
 (`ncdfFlowSet-class`), 5
`Subset`, `ncdfFlowSet`, `filterResultList`-method
 (`ncdfFlowSet-class`), 5
`Subset`, `ncdfFlowSet`, `list`-method
 (`ncdfFlowSet-class`), 5

`transform`, `ncdfFlowSet`-method
 (`ncdfFlowSet-class`), 5

`updateIndices`
 (`ncdfFlowSet-class`), 5
`updateIndices`, `ncdfFlowSet`, `character`, `logical`-method
 (`ncdfFlowSet-class`), 5

`xyplot`, `formula`, `ncdfFlowSet`-method
 (`ncdfFlowSet-class`), 5