

ncdfFlow: Provides netCDF storage based methods and functions for manipulation of flow cytometry data

Mike Jiang, Greg Finak, N. Gopalakrishnan

April 6, 2012

Abstract

Background The Bioconductor package `flowCore` is the object model and a collection of standard tools designed for flow cytometry data analysis. The related R packages including data analysis (`flowClust`, `flowMerge`, `flowMeans`, `flowTrans`, `flowStats`), visualization (`flowViz`) and quality control (`flowQ`) use the `flowCore` infrastructure to deal with flow cytometry data. However the `flowFrame` or `flowSet` which represent a single or a set of FCS files are memory-resident data structures and require the entire data elements to remain in memory in order to perform all kinds of the data manipulations. Hundreds or thousands of datasets generated by high throughput instruments can easily hit the memory limit if they are imported as the `flowSet` or `flowFrames` in R. It presents a challenge to scientists and bioinformaticians who use the R tools described above to perform statistical data analysis on a regular computer. We propose a new R object model and related functions to address this problem. The new model `ncdfFlowSet` inherit most of data structures from `flowSet`. It stores the large volume of event-level data on disk and only keeps the file handler and meta data in memory. Thus the memory consumption is significantly reduced. NetCDF is used as the data formats because it is self-describing, machine-independent and specifically optimized for storing and accessing array-oriented scientific data. With the compression and chunking features introduced by the new release of netCDF4, the new model is able to maintain high performance of data processing.

Most of the functions and methods including transformation, compensation, gating and subsetting methods for `flowSet` are extended to `ncdfFlowSet` (`spillover`, `normalize` and `workflow` methods of `flowCore` are currently not supported yet.). Thus the change of data structure is almost transparent to the users of `flowCore`-based R packages.

keywords Flow cytometry, high throughput, netCDF, `flowSet`, `ncdfFlowSet`

1 Representing Flow Cytometry Data with `ncdfFlowSet`

`ncdfFlow` represents a flow cytometry data model that is very similar to the `flowSet` structure. The only difference is that the event-based 2-D data matrices from multiple samples of the same experiment are stored as one single 3D data matrix on disk in netcdf format. Each sample can be accessed efficiently from the 3-D matrix as a data chunk or slice and further manipulated in memory.

The basic unit of manipulation in `ncdfFlow` is the `ncdfFlowSet`. It inherits all the slots from `flowSet`. However, the `flowFrame` objects stored in the "frames" slot of a `ncdfFlowSet` object do not host the data matrix. Instead, their "exprs" slots are kept empty and the

actual data are stored in the 3-D data matrix on disk and only the file name is stored in "file" slot of `ncdfFlowSet`. Thus `ncdfFlowSet` reduces the memory requirements and meanwhile ensures the consistent data structure with *flowSet*.

2 Creating a *ncdfFlowSet*

We provide a function to read FCS files into a *ncdfFlowSet* object:

```
> path<-system.file("extdata","compdata","data",package="flowCore")
> files<-list.files(path,full.names=TRUE)[1:3]
> nc1 <- read.ncdfFlowSet(files=files,ncdfFile="ncfsTest.nc")
> nc1
```

An *ncdfFlowSet* with 3 samples.

flowSetId :

NCDF file : ncfsTest.nc

An object of class "AnnotatedDataFrame"

rowNames: 060909.001 060909.002 060909.003

varLabels: name

varMetadata: labelDescription

column names:

FSC-H, SSC-H, FL1-H, FL2-H, FL3-H, FL1-A, FL4-H

As we see, the constructor function is very similar to the *flowSet* except that it requires a filename for the ncdf file.

```
> fs1 <- read.flowSet(files=files)
```

Note that an ncdf file that stores the actual data is generated and saved on disk once a *ncdfFlowSet* is created. Users need to explicitly call the `ncfsUnlink` method to remove the file before delete the object from memory by `rm`.

```
> ncfsUnlink(nc1)
```

```
> rm(nc1)
```

Sometime it is necessary to save both data matrix and meta data on disk and reload the *ncdfFlowSet* object later. In order to do this, `isSaveMeta` argument has to be set to `TRUE` when the *ncdfFlowSet* is created by `read.ncdfFlowSet`. `ncdfFlowSet_sync` can be also called to save the meta data explicitly in cdf file.

```
> nc1 <- read.ncdfFlowSet(files=files,ncdfFile="ncfsTest.nc",isSaveMeta=TRUE)
> rm(nc1)
> nc1<-ncdfFlowSet_open("ncfsTest.nc")
> nc1[[1]]
> ncfsUnlink(nc1)
> rm(nc1)
```

Users can also create an empty `ncdfFlowSet` first and add data slices later by assigning argument `isWriteSlice` as `FALSE`.

```
> nc1 <- read.ncdfFlowSet(files=files,ncdfFile="ncfsTest.nc",isWriteSlice= FALSE)
> nc1[[1]]
```

```
flowFrame object '060909.001'
with 0 cells and 7 observables:
      name      desc range minRange maxRange
$P1 FSC-H FSC-Height 1024      0      1023
$P2 SSC-H SSC-Height 1024      0      1023
$P3 FL1-H      <NA> 1024      1     10000
$P4 FL2-H      <NA> 1024      1     10000
$P5 FL3-H      <NA> 1024      1     10000
$P6 FL1-A      <NA> 1024      0      1023
$P7 FL4-H      <NA> 1024      1     10000
139 keywords are stored in the 'description' slot
```

As we see here,before add the actual flowFrame by `addFrame`, the flowFrame object returned by `[[` has 0 events.

```
> targetSampleName<-sampleNames(fs1)[1]
> addFrame(nc1,fs1[[1]],targetSampleName)
> nc1[[1]]
```

```
flowFrame object '060909.001'
with 10000 cells and 7 observables:
      name      desc range minRange maxRange
$P1 FSC-H FSC-Height 1024      0      1023
$P2 SSC-H SSC-Height 1024      0      1023
$P3 FL1-H      <NA> 1024      1     10000
$P4 FL2-H      <NA> 1024      1     10000
$P5 FL3-H      <NA> 1024      1     10000
$P6 FL1-A      <NA> 1024      0      1023
$P7 FL4-H      <NA> 1024      1     10000
141 keywords are stored in the 'description' slot
```

```
> nc1[[2]]
```

```
flowFrame object '060909.002'
with 0 cells and 7 observables:
      name      desc range minRange maxRange
$P1 FSC-H Forward Scatter 1024      0      1023
$P2 SSC-H      Side Scatter 1024      0      1023
$P3 FL1-H      FITC 1024      1     10000
$P4 FL2-H      <NA> 1024      1     10000
```

```
$P5 FL3-H          <NA> 1024      1    10000
$P6 FL1-A          <NA> 1024      0     1023
$P7 FL4-H          <NA> 1024      1    10000
144 keywords are stored in the 'description' slot
```

Note that it is important to always use sample name to specify the target position in the data matrix where the actual is added. Because the sample name is the identifier used to index the data matrix.

Sometime it is helpful to copy the structure from an existing `ncdfFlow` object and then add the data slice to the empty `ncdfFlow` cloned by `clone.ncdfFlowSet`.

```
> nc2<-clone.ncdfFlowSet(nc1,"clone.nc")
> nc2[[1]]
> addFrame(nc2,fs1[[1]],sampleNames(fs1)[1])
> nc2[[1]]
```

We also provide the `coerce` function to convert the `flowSet` to a `ncdfFlowSet`.

```
> data(GvHD)
> GvHD <- GvHD[pData(GvHD)$Patient %in% 6:7][1:4]
> nc1<-ncdfFlowSet(GvHD)

[1] "ncdfFlowSet saved in /tmp/RtmpiK0bkn/ncfs16d1c50f51d6c.nc"
```

Or coerce a `ncdfFlowSet` to `flowSet`

```
> fs1<-NcdfFlowSetToFlowSet(nc1,top=2)
```

Note that `ncdfFlowSet` is designed to store large datasets and it is not recommended to coerce the entire `ncdfFlowSet` to `flowSet`. Usually users want to select a subset from `ncdfFlowSet` by `[` and convert the subtyped data. Sometimes it is helpful to randomly select a certain number of `flowFrames` from the entire datasets represented by `ncdfFlowSet` to have a preview of the data. The argument `"top"` can be used here for this purpose.

3 Working with metadata

Like `flowSet`, `ncdfFlowSet` has an associated `AnnotatedDataFrame` that provides metadata of experiments. This data frame is accessed and modified via the same methods of `flowCore`. :

```
> phenoData(nc1)
> pData(nc1)
> varLabels(nc1)
> varMetadata(nc1)
> sampleNames(nc1)
> keyword(nc1,"FILENAME")
> identifier(nc1)
> colnames(nc1)
> colnames(nc1,prefix="s6a01")
> length(nc1)
> getIndices(nc1,"s6a01")
```

4 Manipulating a *ncdfFlowSet*

You can extract a *flowFrame* from a *ncdfFlowSet* object in the same way as *flowSet* by using the `[[` or `$` extraction operators. Note that using the `[` extraction operator returns a new *ncdfFlowSet* that points to the same *ncdf* file. SO the original *ncdf* file serves as a data repository and the *ncdfFlowSet* works as view of the data in this sense.

```
> nm<-sampleNames(nc1)[1]
> expr1<-paste("nc1$",nm,"'",sep="")
> eval(parse(text=expr1))
```

flowFrame object 's6a01'

with 2205 cells and 8 observables:

	name	desc	range	minRange	maxRange
\$P1	FSC-H	FSC-Height	1024	0	1023
\$P2	SSC-H	SSC-Height	1024	0	1023
\$P3	FL1-H	CD15 FITC	1024	1	10000
\$P4	FL2-H	CD45 PE	1024	1	10000
\$P5	FL3-H	CD14 PerCP	1024	1	10000
\$P6	FL2-A	<NA>	1024	0	1023
\$P7	FL4-H	CD33 APC	1024	1	10000
\$P8	Time	Time (102.40 sec.)	1024	0	1023

150 keywords are stored in the 'description' slot

```
> nc1[[nm]]
```

flowFrame object 's6a01'

with 2205 cells and 8 observables:

	name	desc	range	minRange	maxRange
\$P1	FSC-H	FSC-Height	1024	0	1023
\$P2	SSC-H	SSC-Height	1024	0	1023
\$P3	FL1-H	CD15 FITC	1024	1	10000
\$P4	FL2-H	CD45 PE	1024	1	10000
\$P5	FL3-H	CD14 PerCP	1024	1	10000
\$P6	FL2-A	<NA>	1024	0	1023
\$P7	FL4-H	CD33 APC	1024	1	10000
\$P8	Time	Time (102.40 sec.)	1024	0	1023

150 keywords are stored in the 'description' slot

```
> nm<-sampleNames(nc1)[c(1,3)]
> nc2<-nc1[nm]
> summary(nc2)
```

\$s6a01

	FSC-H	SSC-H	FL1-H	FL2-H	FL3-H	FL2-A	FL4-H	Time
Min.	60.0	0.0	1.000	1.00	1.000	0.0	1.000	11.0

1st Qu.	159.0	48.0	1.046	35.35	1.000	6.0	1.000	40.0
Median	196.0	65.0	2.644	160.40	1.383	36.0	5.289	57.0
Mean	220.8	108.9	57.540	210.10	7.367	48.7	16.240	51.9
3rd Qu.	264.0	97.0	7.055	320.90	2.460	75.0	20.780	66.0
Max.	1023.0	1023.0	3782.000	1637.00	326.700	516.0	503.300	80.0

\$s6a03

	FSC-H	SSC-H	FL1-H	FL2-H	FL3-H	FL2-A	FL4-H	Time
Min.	59.0	0	1.000	1.0	1.000	0.0	1.000	0.0
1st Qu.	147.0	49	1.000	341.8	1.000	79.0	1.165	105.0
Median	192.0	71	1.145	526.5	1.070	124.0	2.228	215.5
Mean	188.5	116	62.170	543.7	5.400	127.9	8.352	233.9
3rd Qu.	226.0	119	10.200	702.3	2.208	164.0	4.834	353.0
Max.	1023.0	1023	10000.000	8504.0	7565.000	1023.0	665.400	567.0

flowSet-specific iterator `fsApply` can also be applied to `RobobjectncdfFlowSet`:

```
> fsApply(nc1, range)
> fsApply(nc1, each_col, median)
```

However, we recommend to use another iterator `ncfsApply` designed for the function that returns a `flowFrame` (such as `compensate`, `transform`...). `ncfsApply` works the same as `fsApply` except that it returns a `ncdfFlowSet` object with the actual data stored in `cdf` to avoid the huge memory consumption. Note that the return value of the function applied in `ncfsApply` must be a `flowFrame` object and it should have the same dimensions (channels and events) as the original data.

5 Compensation, Transformation and Gating

`transform` and `compensate` for `ncdfFlowSet` work the same as `flowSet`.

```
> cfile <- system.file("extdata", "compdata", "compmatrix", package="flowCore")
> comp.mat <- read.table(cfile, header=TRUE, skip=2, check.names = FALSE)
> comp <- compensation(comp.mat)
> #compensation
> summary(nc1)[[1]]
> nc2 <- compensate(nc1, comp)
> summary(nc2)[[1]]
> ncfsUnlink(nc2)
> rm(nc2)
> #transformation
> asinhTrans <- arcsinhTransform(transformationId="ln-transformation", a=1, b=1, c=1)
> nc2 <- transform(nc1, `FSC-H` = asinhTrans(`FSC-H`))
> summary(nc1)[[1]]
> summary(nc2)[[1]]
> ncfsUnlink(nc2)
> rm(nc2)
```

Note that compensation/transformation return the `ncdfFlowSet` objects that point to the new cdf file containing the compensated/transformed data.

`filter` for `flowSet` also works for `ncdfFlowSet`:

```
> rectGate <- rectangleGate(filterId="nonDebris", "FSC-H"=c(200, Inf))
> fr <- filter(nc1, rectGate)
> summary(fr)
> rg2 <- rectangleGate(filterId="nonDebris", "FSC-H"=c(300, Inf))
> rg3 <- rectangleGate(filterId="nonDebris", "FSC-H"=c(400, Inf))
> flist <- list(rectGate, rg2, rg3)
> names(flist) <- sampleNames(nc1[1:3])
> fr3 <- filter(nc1[1:3], flist)
> summary(fr3[[3]])
```

6 Subsetting

The `Subset` and `split` methods for `ncdfFlowSet`:

```
> nc2 <- Subset(nc1, rectGate)
> summary(nc2[[1]])
```

	FSC-H	SSC-H	FL1-H	FL2-H	FL3-H	FL2-A	FL4-H	Time
Min.	200	0	1.000	1.00	1.000	0.00	1.000	11.00
1st Qu.	230	69	1.334	22.33	1.000	3.00	2.692	40.00
Median	266	87	3.372	77.37	1.500	16.00	12.900	57.00
Mean	297	141	91.900	165.70	10.830	38.24	22.020	51.82
3rd Qu.	316	122	18.990	223.80	2.551	53.00	29.790	66.00
Max.	1023	1023	1943.000	1637.00	326.700	516.00	464.200	80.00

```
> nc2 <- Subset(nc1, fr)
> summary(nc2[[1]])
```

	FSC-H	SSC-H	FL1-H	FL2-H	FL3-H	FL2-A	FL4-H	Time
Min.	200	0	1.000	1.00	1.000	0.00	1.000	11.00
1st Qu.	230	69	1.334	22.33	1.000	3.00	2.692	40.00
Median	266	87	3.372	77.37	1.500	16.00	12.900	57.00
Mean	297	141	91.900	165.70	10.830	38.24	22.020	51.82
3rd Qu.	316	122	18.990	223.80	2.551	53.00	29.790	66.00
Max.	1023	1023	1943.000	1637.00	326.700	516.00	464.200	80.00

```
> rm(nc2)
> morphGate <- norm2Filter("FSC-H", "SSC-H", filterId = "MorphologyGate", scale = 2)
> smaller <- Subset(nc1[c(1,3)], morphGate, c("FSC-H", "SSC-H"))
> smaller[[1]]
```

```

flowFrame object 's6a01'
with 1647 cells and 2 observables:
      name      desc range minRange maxRange
$P1 FSC-H FSC-Height 1024      0    1023
$P2 SSC-H SSC-Height 1024      0    1023
150 keywords are stored in the 'description' slot

```

```
> nc1[[1]]
```

```

flowFrame object 's6a01'
with 2205 cells and 8 observables:
      name      desc range minRange maxRange
$P1 FSC-H      FSC-Height 1024      0    1023
$P2 SSC-H      SSC-Height 1024      0    1023
$P3 FL1-H      CD15 FITC 1024      1   10000
$P4 FL2-H      CD45 PE 1024      1   10000
$P5 FL3-H      CD14 PerCP 1024      1   10000
$P6 FL2-A      <NA> 1024      0    1023
$P7 FL4-H      CD33 APC 1024      1   10000
$P8 Time Time (102.40 sec.) 1024      0    1023
150 keywords are stored in the 'description' slot

```

```
> rm(smaller)
```

Note that `Subset` does not create the new ncdf file (the same as extraction operator `[]`). So we need to be careful about using `ncfsUnlink` to delete the subsetted data since it points to the same ncdf file that is also used by the original `ncdfFlowSet` object.

`split` returns a `ncdfFlowList` object which is a container of multiple `ncdfFlowSet` objects.

```

> ##splitting by a gate
> qGate <- quadGate(filterId="qg", "FSC-H"=200, "SSC-H"=400)
> fr<-filter(nc1,qGate)
> ncList<-split(nc1,fr)
> ncList

```

An `ncdfFlowList` with 4 `ncdfFlowSets` containing 4 unique samples.

```
> nc1[[1]]
```

```

flowFrame object 's6a01'
with 2205 cells and 8 observables:
      name      desc range minRange maxRange
$P1 FSC-H      FSC-Height 1024      0    1023
$P2 SSC-H      SSC-Height 1024      0    1023
$P3 FL1-H      CD15 FITC 1024      1   10000
$P4 FL2-H      CD45 PE 1024      1   10000

```

```

$P5 FL3-H          CD14 PerCP  1024      1    10000
$P6 FL2-A          <NA>      1024      0     1023
$P7 FL4-H          CD33 APC   1024      1    10000
$P8 Time Time (102.40 sec.) 1024      0     1023
150 keywords are stored in the 'description' slot

```

```
> ncList[[2]][[1]]
```

```
flowFrame object 's6a01'
```

```
with 36 cells and 8 observables:
```

	name	desc	range	minRange	maxRange
\$P1	FSC-H	FSC-Height	1024	0	1023
\$P2	SSC-H	SSC-Height	1024	0	1023
\$P3	FL1-H	CD15 FITC	1024	1	10000
\$P4	FL2-H	CD45 PE	1024	1	10000
\$P5	FL3-H	CD14 PerCP	1024	1	10000
\$P6	FL2-A	<NA>	1024	0	1023
\$P7	FL4-H	CD33 APC	1024	1	10000
\$P8	Time	Time (102.40 sec.)	1024	0	1023

150 keywords are stored in the 'description' slot

```
> ncList[[1]][[1]]
```

```
flowFrame object 's6a01'
```

```
with 74 cells and 8 observables:
```

	name	desc	range	minRange	maxRange
\$P1	FSC-H	FSC-Height	1024	0	1023
\$P2	SSC-H	SSC-Height	1024	0	1023
\$P3	FL1-H	CD15 FITC	1024	1	10000
\$P4	FL2-H	CD45 PE	1024	1	10000
\$P5	FL3-H	CD14 PerCP	1024	1	10000
\$P6	FL2-A	<NA>	1024	0	1023
\$P7	FL4-H	CD33 APC	1024	1	10000
\$P8	Time	Time (102.40 sec.)	1024	0	1023

150 keywords are stored in the 'description' slot

Keep in mind that the `ncdfFlowSet` objects contained in `ncdfFlowList` share the same `ncdf` file as the original `ncdfFlowSet`. So deleting `ncdfFlowList` by `ncfsUnlink` is also going to make the data not accessible to the original `ncdfFlowSet`.