

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

Mike Jiang,Greg Finak,N. Gopalakrishnan

April 3, 2020

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 (`ggcyto`) and quality control (`QUALIFIER`) 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`

```
## Loading required package: flowCore
## Loading required package: RcppArmadillo
## Loading required package: BH
```

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 `ncdf` 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)

## All FCS files have the same following channels:
## FSC-H
## SSC-H
## FL1-H
## FL2-H
## FL3-H
## FL1-A
## FL4-H
## write 060909.001 to empty cdf slot...
## write 060909.002 to empty cdf slot...
## write 060909.003 to empty cdf slot...
## done!

nc1

## An ncdfFlowSet with 3 samples.
## NCDF file : /tmp/Rtmp57qxjn/ncfs72a69408893.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 `unlink` method to remove the file before delete the object from memory by `rm`.

```
unlink(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, isWriteSlice= FALSE)

## All FCS files have the same following channels:
## FSC-H
## SSC-H
## FL1-H
## FL2-H
## FL3-H
## FL1-A
## FL4-H
## done!

nc1[[1]]

## flowFrame object 'anonymous'
## with 0 cells and 7 observables:
##      name      desc range minRange maxRange
## $P1 FSC-H FSC-Height 1024      -111      1023
## $P2 SSC-H SSC-Height 1024      -111      1023
## $P3 FL1-H      <NA> 1024      -111      1023
## $P4 FL2-H      <NA> 1024      -111      1023
## $P5 FL3-H      <NA> 1024      -111      1023
## $P6 FL1-A      <NA> 1024      -111      1023
## $P7 FL4-H      <NA> 1024      -111      1023
## 1 keywords are stored in the 'description' slot
```

As we see here, before writing the actual `flowFrame` by `[[<-`, the `flowFrame` object returned by `[[` has 0 events.

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

## write 060909.001 to empty cdf slot...

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 'anonymous'
## with 0 cells and 7 observables:
##      name      desc range minRange maxRange
## $P1 FSC-H FSC-Height 1024     -111     1023
## $P2 SSC-H SSC-Height 1024     -111     1023
## $P3 FL1-H      <NA> 1024     -111     1023
## $P4 FL2-H      <NA> 1024     -111     1023
## $P5 FL3-H      <NA> 1024     -111     1023
## $P6 FL1-A      <NA> 1024     -111     1023
## $P7 FL4-H      <NA> 1024     -111     1023
## 1 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, isEmpty = TRUE)
nc2[[1]]
nc2[[sampleNames(fs1)[1]]] <- fs1[[1]]

## write 060909.001 to empty cdf slot...

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)

## write s6a01 to empty cdf slot...
```

```
## write s6a02 to empty cdf slot...
## write s6a03 to empty cdf slot...
## write s6a04 to empty cdf slot...
```

Or coerce a `ncdfFlowSet` to `flowSet`

```
fs1<-as.flowSet(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 subsetted 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, 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
## 166 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
## 166 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
## Min.      60.0000      0.0000      1.000000      1.00000      1.000000      0.00000
## 1st Qu.    159.0000      48.0000      1.046045      35.34981      1.000000      6.00000
## Median    196.0000      65.0000      2.644158     160.42741      1.382810      36.00000
## Mean      220.7642     108.8853      57.543711     210.07988      7.366665      48.69569
## 3rd Qu.    264.0000      97.0000      7.054802     320.88828      2.460406      75.00000
## Max.      1023.0000    1023.0000    3781.922363    1637.10388    326.718719    516.00000
##      FL4-H      Time
## Min.      1.000000    11.00000
## 1st Qu.    1.000000    40.00000
## Median     5.288867    57.00000
## Mean      16.243151    51.90476
## 3rd Qu.    20.782274    66.00000
```

```
## Max.      503.335175 80.00000
##
## $s6a03
##           FSC-H      SSC-H           FL1-H      FL2-H           FL3-H      FL2-A
## Min.      59.0000      0.000      1.000000      1.0000      1.000000      0.0000
## 1st Qu.   147.0000     49.000      1.000000     341.7625      1.000000      79.0000
## Median    192.0000     71.000      1.144593     526.5112      1.069867     124.0000
## Mean      188.4942    116.008      62.174581     543.7355      5.400462     127.8592
## 3rd Qu.   226.0000    119.000      10.204639     702.3116      2.208442     164.0000
## Max.     1023.0000  1023.000  10000.000000  8503.9121  7564.633301  1023.0000
##           FL4-H      Time
## Min.      1.000000      0.0000
## 1st Qu.    1.165390    105.0000
## Median     2.228415    215.5000
## Mean       8.351631    233.9134
## 3rd Qu.    4.833503    353.0000
## Max.     665.379456    567.0000
```

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

```
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)

## write s6a01 to empty cdf slot...
## write s6a02 to empty cdf slot...
## write s6a03 to empty cdf slot...
## write s6a04 to empty cdf slot...
```

```

summary(nc2)[[1]]
unlink(nc2)
rm(nc2)

#transformation
asinhTrans <- arcsinhTransform(transformationId="ln-transformation", a=1, b=1, c=1)
nc2 <- transform(nc1, `FL1-H`=asinhTrans(`FL1-H`))

## write s6a01 to empty cdf slot...
## write s6a02 to empty cdf slot...
## write s6a03 to empty cdf slot...
## write s6a04 to empty cdf slot...

summary(nc1)[[1]]
summary(nc2)[[1]]
unlink(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
## Min.	200.0000	0.0000	1.000000	1.00000	1.000000	0.00000
## 1st Qu.	230.0000	69.0000	1.333897	22.33436	1.000000	3.00000
## Median	266.0000	87.0000	3.371780	77.36830	1.499523	16.00000


```
## Mean      296.9887  141.0131   91.895427  165.68587  10.830258  38.24038
## 3rd Qu.   316.0000  122.0000   18.992949  223.84692   2.550628  53.00000
## Max.     1023.0000 1023.0000 1942.529175 1637.10388 326.718719 516.00000
##           FL4-H      Time
## Min.       1.000000 11.00000
## 1st Qu.    2.692201 40.00000
## Median    12.896131 57.00000
## Mean      22.021327 51.81972
## 3rd Qu.   29.791735 66.00000
## Max.     464.158875 80.00000
```

```
library(flowStats)
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
## 166 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
## 166 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 `unlink` 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

## $`FSC-Height+SSC-Height+`
## An ncdfFlowSet with 4 samples.
## NCDF file : /tmp/Rtmp57qxjn/ncfs72a64daf94b7.nc
## An object of class 'AnnotatedDataFrame'
##   rowNames: s6a01 s6a02 s6a03 s6a04
##   varLabels: Patient Visit ... population (6 total)
##   varMetadata: labelDescription
##
##   column names:
##     FSC-H, SSC-H, FL1-H, FL2-H, FL3-H, FL2-A, FL4-H, Time
##
## $`FSC-Height-SSC-Height+`
## An ncdfFlowSet with 4 samples.
## NCDF file : /tmp/Rtmp57qxjn/ncfs72a64daf94b7.nc
## An object of class 'AnnotatedDataFrame'
##   rowNames: s6a01 s6a02 s6a03 s6a04
##   varLabels: Patient Visit ... population (6 total)
##   varMetadata: labelDescription
##
##   column names:
##     FSC-H, SSC-H, FL1-H, FL2-H, FL3-H, FL2-A, FL4-H, Time
##
## $`FSC-Height+SSC-Height-`
## An ncdfFlowSet with 4 samples.
## NCDF file : /tmp/Rtmp57qxjn/ncfs72a64daf94b7.nc
## An object of class 'AnnotatedDataFrame'
##   rowNames: s6a01 s6a02 s6a03 s6a04
##   varLabels: Patient Visit ... population (6 total)
##   varMetadata: labelDescription
##
##   column names:
##     FSC-H, SSC-H, FL1-H, FL2-H, FL3-H, FL2-A, FL4-H, Time
##
## $`FSC-Height-SSC-Height-`
## An ncdfFlowSet with 4 samples.
## NCDF file : /tmp/Rtmp57qxjn/ncfs72a64daf94b7.nc

```

```
## An object of class 'AnnotatedDataFrame'
##   rowNames: s6a01 s6a02 s6a03 s6a04
##   varLabels: Patient Visit ... population (6 total)
##   varMetadata: labelDescription
##
##   column names:
##     FSC-H, SSC-H, FL1-H, FL2-H, FL3-H, FL2-A, FL4-H, Time

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
## 166 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
## 166 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
## 166 keywords are stored in the 'description' slot
```

Note that the `ncdfFlowSet` objects contained in `ncdfFlowList` by default share the same `ncdf` file as the original `ncdfFlowSet`. In order to keep its own data copy, try to set argument `isNew` to `"TRUE"` in `split` method.