flowCore Access to FlowJo Workspaces

John Gosink and Hugh Rand

Amgen, Inc.

Vignette last rewritten: 12th December 2008
Vignette last recompiled: October 1, 2012

Abstract

The flowFlowJo package provides a way to parse FlowJo workspaces into R data structures with flowCore compliant objects representing the various gates, compensation matrices, and other related information. The objective of this package it to make it easy (in R) to use compensation and gating information that has been produced using FlowJo.

1. Introduction

FlowJo (www.flowjo.com) is a commercially available software package used for the gating, visualization, and analysis of data from flow cytometry experiments. FlowJo saves its session information in an XML formatted file called a "workspace" that contains all the information necessary to describe the gating structures, compensation, transformation, location of the FCS files, and opened graphs and figures last created by the user. FlowJo workspace files do not contain raw data. This package (flowFlowJo) is a set of methods designed to extract the file locations, gates, compensation matrices, and other data contained in FlowJo workspace files and return the information in a manner consistent with the BioConductor flowCore packages. The typical internal steps taken by flowFlowJo are:

(1) read and parse a FlowJo workspace,
(2) extract the structure of the gates for the FCS files,
(3) extract compensation matrices for the FCS files,
(4a) produce a series of lists of R/flowCore information necessary to load, compensate, and gate all of the data files in R as was done in FlowJo , –or–
(4b) directly compile a table of summary statistics for each of the signal channels (parameters) on all the cell populations for all of the FCS files referenced in the workspace.

The flowFlowJo package also contains a method for combining extracted summary statistics information with any arbitrary meta data describing the experimental design or layout and/or meta data embedded within the header section of each FCS file. We describe all of these processes in detail below. Finally, this package is known to work with FlowJo version 7.2.2. As a commercial package FlowJo workspace formats change from time-to-time and we cannot guarantee compatibility with future versions.

2. Gates/Filters and Compensation Matrices

Prior to using this package, it is assumed that FlowJo will have been used to process (compensate and gate) one or more FCS files to produce one or more FlowJo workspaces. This is a routine process for those analyzing flow cytometry data. The only caution we mention is that the location of the FCS files is held in the FlowJo workspace, typically as absolute paths. Moving the FCS files to another location will cause the location of these files extracted from the workspace to be in error. For this reason the examples provided below have to go the extra step of specifying an alternate location for the FCS files other than as they are given within the provided demonstration FlowJo workspace.

The first step when using flowFlowJo is to read in all of the information associated with a set of FlowJo workspaces. This is accomplished with the "readFlowJoList" method. This method takes in a list, vector, or character string with the full path of one or more FlowJo workspaces and reads in the gating structures, compensation matrices, transformations and other information about the FCS files referenced in the workspace(s) and returns a flowJoList object.

```
> library(flowCore);
> library(XML);
> library(flowFlowJo);
> demoLocation <- system.file("extdata", "DemoWorkspace.wsp", package="flowFlowJo");
> actualFCSLoc <- system.file("extdata/fcsFiles", package="flowFlowJo");
> testList <- readFlowJoList(demoLocation, altFileLocation=actualFCSLoc);

Getting a workspace for: DemoWorkspace.wsp... got it!
Having a bit of trouble finding your FCS files,
I'll look in  D:/biocbld/bbs-2.11-bioc/tmpdir/RtmpSqLUcS/Rinst2e1042d97af4/flowFlowJo/extdat
Working on:  B02  A-Phase  G-Phase  M-Phase  M-Phase:M1 Pop  M-Phase:M2 Pop  S-Phase
Working on:  A02  A-Phase  G-Phase  M-Phase  M-Phase:M1 Pop  M-Phase:M2 Pop  S-Phase
Working on:  C02  A-Phase  G-Phase  M-Phase  M-Phase:M1 Pop  M-Phase:M2 Pop  S-Phase
Working on:  C03  A-Phase  G-Phase  M-Phase  M-Phase:M1 Pop  M-Phase:M2 Pop  S-Phase
Working on:  A01  A-Phase  G-Phase  M-Phase  M-Phase:M1 Pop  M-Phase:M2 Pop  S-Phase
Working on:  A03  A-Phase  G-Phase  M-Phase  M-Phase:M1 Pop  M-Phase:M2 Pop  S-Phase
Working on:  B03  A-Phase  G-Phase  M-Phase  M-Phase:M1 Pop  M-Phase:M2 Pop  S-Phase
Working on:  B01  A-Phase  G-Phase  M-Phase  M-Phase:M1 Pop  M-Phase:M2 Pop  S-Phase
Working on:  C01  A-Phase  G-Phase  M-Phase  M-Phase:M1 Pop  M-Phase:M2 Pop  S-Phase
 done!
```

To be clear, the readFlowJoList method does not look for, or use, any of the referenced FCS files. It only looks at the information contained in the workspace(s). It is important to note that at this stage the XML gating structures embedded within the workspace(s) are converted into flowCore style filter objects.

The flowJoList object is a list of flowJoObj objects. Each flowJoObj encapsulates the contents of one FlowJo workspace. Since FlowJo allows for the possibility of having a different compensation matrix for each FCS file referenced within a single workspace and because a given FCS file may be dealt with separately in several different FlowJo workspaces, the readFlowJoList method allows data from more than one workspace to be combined into one analysis. This latter approach is often used at our institution such that an assay may be run over many weeks or months, with the data from each day's run being accumulated into a single FlowJo workspace.

After generating a flowJoList object, a user may extract a subset of the gates. The "getFlowJoGates" method extracts the file name, file name with full path, filter objects, simplified filter names, and compensation matrices associated with all of the FCS files that match one or more file name patterns supplied by the user.

```
> z <- getFlowJoGates(testList, fileNamePatterns=c("CO2"));
> print(summary(z));

            Length Class  Mode
fcsName     6      -none- character
FCSFilename 6      -none- character
filter      6      -none- list
filterName  6      -none- character
compMats    6      -none- list

> print(summary(z$filter));

        Length Class           Mode
filter1 1      polygonGate     S4
filter2 1      polygonGate     S4
filter3 1      polygonGate     S4
filter4 1      intersectFilter S4
filter5 1      intersectFilter S4
filter6 1      polygonGate     S4
```

The return item is a list of lists, with each of the sub lists corresponding to the items described above. These items may be accessed and used arbitrarily. Note that a single FCS file may be partitioned with several different gates (filters) and that each of these gates may dependend on other gates as is appropriate to subdivide the cell populations. The getFlowJoGates method accounts for this by simply expanding the number of returned items in each of the sub lists to correspond to each of the intermediate gates. By default, the getFlowJoGates method concatenates each child gate to its parent gates as an intersect filter. Correspondingly, by default, the code concatenates the gate names for each child gate to its parents gate names with a colon. Finally, by default each gate name

is preappended with the name of the FCS file it references. Thus, typical gate names might be:

"SampleA3.fcs:Lymphocytes",
"SampleA3.fcs:Lymphocytes:CD3+",
"SampleA3.fcs:Lymphocytes:CD3+:CD8+"
etc...

Consider the case in which a researcher uses FlowJo to gate a set of 30 FCS files each for 6 different cell populations and sub-populations. A getFlowJoGates call on the resulting workspace file will return a list of length 5 with elements: "fcsName", "FCSFilename", "filter", "filterName", and "compMats". Each element (e.g. fcsName) will in turn be a list of length 180 because there are 6 gates for each of the 30 FCS files.

Importantly, each of the 5 lists will be in the same order. Thus, for example, the 37th filter in the "filter" list will be for the 37th FCS file in the "FCSFilename" list, and that FCS file should be compensated with the 37th compensation matrix in the "compMats" list. Although other, more compact representations could be found, these data structures as currently implemented were effective and general enough for the problems at hand.

Finally, FlowJo currently implements its compensation/spillover matrices differently than they are implemented in the general flow cytometry community. Currently, in order to obtain similar results (e.g. MFIs and cell counts) between FlowJo and flowCore, it is necessary to apply the compensation matrix to the data in the usual way (ie. via "compensate"), and then to divide all of the observed data by the maximum of the values in the compensation matrix. The flowFlowJo package implements a method, flowJoCompensate, to automatically take care of this issue.

In some cases a data analyst may wish to proceed with the list of lists to extract and analyze the data to their own design. In many cases however, the analyst may be satisfied with the gating choices derived during the FlowJo session and wish to simply proceed with a complete set of summary statistics on all of the cell populations. This is described in section 4 below.

3. Transformations

The information contained within the "DivaSettings" and "TransformSettings" sections of the FlowJo workspace is currently parsed by the readFlowJoList method. These components are returned in the data structure produced by the readFlowJoList method, but there are no other methods in the flowFlowJo package that utilize these data. All of the non-scatter gates (fluorescence channel gates) are encoded by their non-transformed gate coordinates. Furthermore, the scatter gates (FSC, and SSC) are currently (FlowJo 7.2.2) encoded as 1/64 of their actual (untransformed) gating coordinates. The readFlowJoList method automatically (internally) multiplies all of the scatter gating coordinates by 64 to adjust for this prior to generating its flowCore filter objects.

4. FlowJo Summary Objects

The flowFlowJo package contains a few methods for automatically extracting the major types of information that are often needed from flow experiments such as median fluorescent intensity, and cell counts. The first step, however, in automating the analysis of manually gated data is to ensure uniformity of the naming convention across all of the samples and to confirm that all of the expected data is present. It has been our experience that bench researchers often (accidentally) supply slightly different names for the same cell populations, neglect to collect certain populations, lose samples, "unexpectedly" add unplanned samples etc. during the course of a study. Such uncommunicated variances from the experimental plan often provide hours of entertainment for the downstream data analyst. A simple summary of the observed data sets often helps identify these anomalies. Toward this end, the "getFlowJoSummary" method returns a table showing the number and counts of different gate names associated with all of the FCS files.

```
> getFlowJoSummary(testList, gatesByFile=FALSE, removeParentalNames=TRUE);

A-Phase G-Phase M-Phase  M1 Pop  M2 Pop S-Phase
      9       9       9       9       9       9

> getFlowJoSummary(testList, removeParentalNames=TRUE);

     A-Phase G-Phase M-Phase M1 Pop M2 Pop S-Phase

A01        1       1       1      1      1       1
A02        1       1       1      1      1       1
A03        1       1       1      1      1       1
B01        1       1       1      1      1       1
B02        1       1       1      1      1       1
B03        1       1       1      1      1       1
C01        1       1       1      1      1       1
C02        1       1       1      1      1       1
C03        1       1       1      1      1       1
```

Passing this check, the "collectSummaryFlowInfo" method returns a data structure with median fluorescent intensities and cell counts for each of the channels (parameters) for each of the gates, and any arbitrary header information from each of the FCS files. It is only at this point that the flowFlowJo methods actually accesses the FCS files.

```
> summaryStatsObj <- collectSummaryFlowInfo(testList);

  Working on file  1 :  B02
  Working on file  2 :  A02
  Working on file  3 :  C02
  Working on file  4 :  C03
```

```
Working on file  5 :  A01
Working on file  6 :  A03
Working on file  7 :  B03
Working on file  8 :  B01
Working on file  9 :  C01
```

Depending on the scale of the experiment and the size of the FCS files, this method may take some time to execute. Specifically, as each FCS file may be many Mb in size, the code only reads one FCS file into memory at a time, extracts the appropriate information, and then moves on to the next file. The actual steps executed by the code include:

1. Read in the FCS file
2. Apply compensation (accounting for the FlowJo/flowCore compensation issues discussed above)
3. Gate out each population (and intermediate sub-population)
4. Collect summary statistics on each population and sub-population
5. Collect any header/keyword information embedded in the FCS file as requested
6. Advance to the next FCS file

A note about step 5. Each FCS file is composed of several parts in addition to the raw list-mode data. The header section of each FCS file contains 100+ pieces of information about each flow run including such things as laser settings, photomultiplier gain settings, run times, and other information. The collectSummaryFlowInfo method can be configured to collect one or more of these items from each FCS file using the "keywords" option. Through the "createFlowReport" method, the summary object can be converted directly into a simple tabular report (data frame). Alternatively the createFlowReport method can combine the summary object with a data frame containing additional meta data about the experiment. In that case, the data frame of additional meta data (e.g. experimental design factors and sample information) must contain at least the columns "FCSFilename" and "FlowJoWorkspace". Each of these columns should give the full path to the relevant FCS file and FlowJo workspace. Each row of the data frame should contain information relevant to one sample (FCS file) of data. The other columns of the data frame can be any arbitrary meta information such as drug name, treatment time, sample ID, etc. The resulting flow report will contain 1 line for each parameter of each cell population of each FCS file along with any associated meta data and keywords from the header section of the FCS file. For example if we had a standard R data frame that had experimental design information in it as shown:

```
> expDescFrame <- data.frame(Drug=c(rep("Amospho", 3),
+                                   rep("Gleevec", 3),
+                                   rep("Chloro", 3)),
+    Conc=rep(c(0.001, 0.0001, 0.00001), 3),
```

```
+    FCSFilename=dir(actualFCSLoc, full.names=TRUE),
+    FlowJoWorkspace=rep(demoLocation, length(dir(actualFCSLoc))));
```

We can combine this data frame with the summary statistics via the createFlowReport method to create a data frame as follows:

```
> flowReport <- createFlowReport(summaryStatsObj,  factorsFrame=expDescFrame);

Working through the summaryList information.  FCS files processed so far:
 Done!

> print(head(flowReport));

  Row Column    CellType       Measure  Value FCSname    Drug  Conc
1   B      2 B02:A-Phase <FSC-H> FSC-H  191.7     B02 Gleevec 1e-04
2   B      2 B02:A-Phase <SSC-H> SSC-H  317.8     B02 Gleevec 1e-04
3   B      2 B02:A-Phase <FL1-H> FL1-H  16.33     B02 Gleevec 1e-04
4   B      2 B02:A-Phase <FL2-H> FL2-H  5.288     B02 Gleevec 1e-04
5   B      2 B02:A-Phase <FL3-H> FL3-H  52.91     B02 Gleevec 1e-04
6   B      2 B02:A-Phase <FL1-A> FL1-A 0.4076     B02 Gleevec 1e-04
```

At this point we are free to create any arbitrary report or visualization of the data using any of the R packages or export some or all of the data to text files as appropriate. The lattice package can be particularly helpful in this endeavour. Finally, note that the demo data, gating, and experimental meta information shown here is completely fabricated for the purposes of this tutorial and don't reflect an actual experiment.

5. Summary

In summary, the flowFlowJo package provides a set of methods for extracting and organizing information from FlowJo workspaces and the FCS files referenced within. In its most basic application it allows the user to retrieve all of the gates and compensation matrices for all of the FCS files described within one or more FlowJo workspaces. The gates are returned as flowCore style filter objects, and the compensation matrices are returned as numeric matrices. Additional functionality is gained by the ability for the user to effectively run all of the compensation and gating functions described by the workspace(s) and automatically retrieve all of the relevant summary statistics into a concise data structure. These data may also be easily combined with any meta data describing the nature or source of each sample and any experimental conditions to which they were subjected.

6. References

Many thanks to Mark Dalphin, Cheng Su, Adam Triester, and Florian Hahne without whose gentle guidance none of this would be possible.