flowPloidy: Overview

Tyler Smith

2016-07-07

Installation

`flowPloidy` depends on the Bioconductor package `flowCore`. You will have to install this dependency before we get started (this requires an internet connection):

```
source("https://bioconductor.org/biocLite.R")
biocLite()
biocLite("flowCore")
```

You only need to do this once, not each time you use `flowPloidy`. Once `flowCore` is installed, you can install `flowPloidy`, and the remaining dependencies will automatically be installed for you. `flowPloidy` is currently available from BitBucket, and you can install it from there using the `devtools` package (which must already be installed):

```
library(devtools)
install_bitbucket("tws/flowPloidy")
```

Once the package is installed, load it as usual:

```
library(flowPloidy)
```

Preliminaries

Sample Data Files

To get started, we need an FCS file (i.e., an LMD file as generated by the flow cytometer). For the purposes of this tutorial, we're using sample files provided with the `flowPloidy` package. These are accessed via the `system.file()` function. For regular use, you would just use the name of the files in your local directory.

Data Channel

You will need to know which channel to use for your analysis. This will most likely remain the same for all analyses from the same flow cytometer. We can use the `read.FCS` function from the `flowCore` package to get the information we need here.

```
library(flowCore)
file1 <- system.file("extdata", "188-15.LMD", package = "flowPloidy")
fcs <- read.FCS(file1, alter.names = TRUE, dataset = 1)
fcs
```

```
## flowFrame object '188-15.LMD'
## with 11369 cells and 9 observables:
##             name         desc range minRange maxRange
## $P1   FS.INT.LIN   FS INT LIN  1024        0     1023
## $P2   SS.INT.LIN   SS INT LIN  1024        0     1023
## $P3         TIME         TIME  1024        0     1023
## $P4  FL3.INT.LIN  FL3 INT LIN  1024        0     1023
## $P5 FL3.PEAK.LIN FL3 PEAK LIN  1024        0     1023
## $P6  FS.PEAK.LIN  FS PEAK LIN  1024        0     1023
## $P7  SS.PEAK.LIN  SS PEAK LIN  1024        0     1023
## $P8   FS.TOF.LIN   FS TOF LIN  1024        0     1023
## $P9   SS.TOF.LIN   SS TOF LIN  1024        0     1023
## 241 keywords are stored in the 'description' slot
```

The channel names are in the `name` column, which you can extract directly with:

```
colnames(exprs(fcs))
```

```
##           $P1N           $P2N           $P3N           $P4N           $P5N
##   "FS.INT.LIN"   "SS.INT.LIN"         "TIME"  "FL3.INT.LIN" "FL3.PEAK.LIN"
##           $P6N           $P7N           $P8N           $P9N
##  "FS.PEAK.LIN"  "SS.PEAK.LIN"   "FS.TOF.LIN"   "SS.TOF.LIN"
```

Hopefully you see something there that corresponds to the channel you're interested in. In our case, it's "`FL3.INT.LIN`".

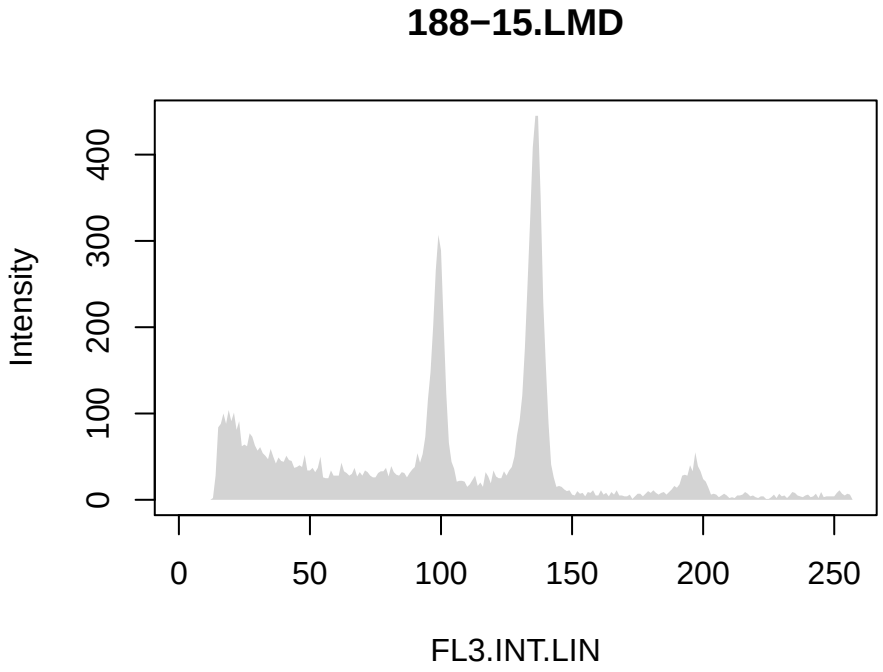
Simple Analysis

With all the necessary packages installed, and the name of the channel of interest identified, we are ready to proceed with a `flowPloidy` analysis.

```
fh1 <- flowHist(FILE = file1, CHANNEL = "FL3.INT.LIN")
```

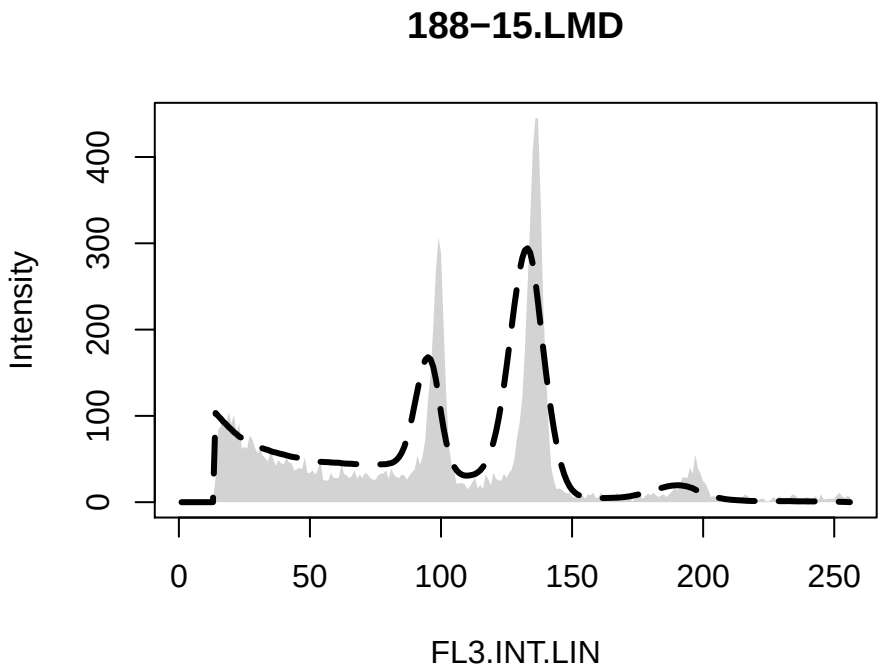
Now that we have our data loaded, `plot` will display it for us:

```
plot(fh1)
```



During the loading process, `flowPloidy` attempts to locate cell populations and identify initial values for model fitting. You can see this by setting the `init = TRUE` argument in the `plot` function:

```
plot(fh1, init = TRUE)
```



In this case, the initial values are close enough for the model-fitting routines to find a solution, so we can immediately proceed to the complete analysis. This will take a few moments:

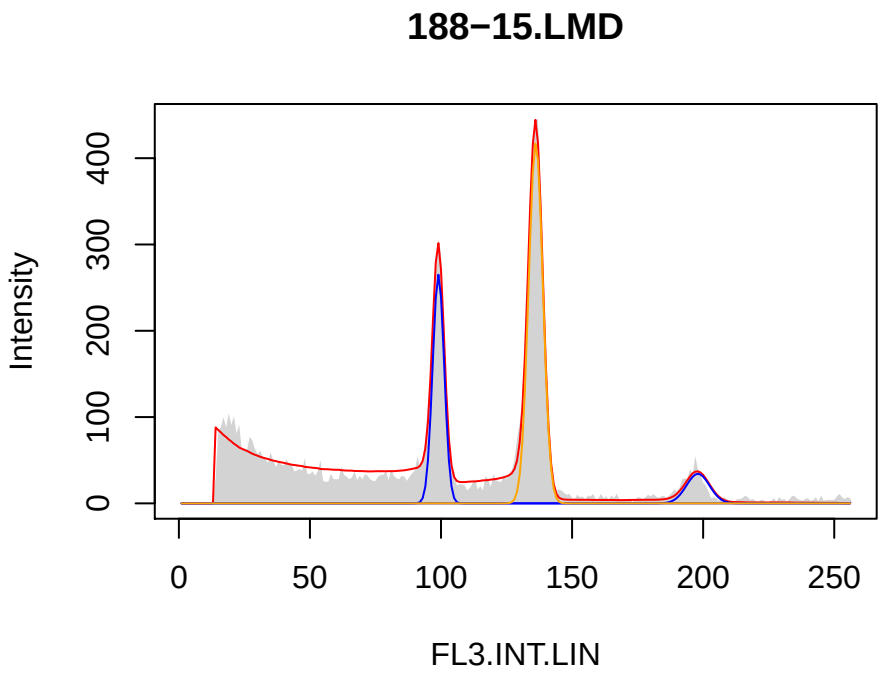
```
fh1 <- fhAnalyze(fh1)
```

Note that I have assigned the output of `fhAnalyze()` to the original `flowHist` object `fh1`. The output of

the analysis is stored in a new slot in the `fh1` object – all the original data remains unaltered in that object.

Now that `fh1` contains a fitted model, the results will be displayed when we call `plot` again:

```
plot(fh1)
```



We can see that the coarse initial estimates were indeed close enough to give us an acceptable model fitting. The numerical results are now available by printing the `flowHist` object:

```
fh1
```

```
## flowHist object
## ===============
## Source file: 188-15.LMD
## Channel: FL3.INT.LIN
## Values: 256
## Total events: 10351
## Model components: singleCut, fA1, fA2, fB1
## Standard not specified
## Fit!
##
## Analysis
## ========
## Modelled events:  10117.9
## Ratio Peak A / Peak B: 0.727, SE: 0.00043
##
## |       |   counts|    size|   cvs|
## |:------|--------:|-------:|-----:|
## |Peak A | 1452.540|  99.007| 0.022|
## |Peak B | 2847.929| 136.136| 0.020|
##
## RCS: 9.299
```

Here we see the number of nuclei modelled in total, the ratio between the peaks with the standard error of the ratio, and a table of peak data: nuclei counts, peak size, and the coefficient of variation for the peak.

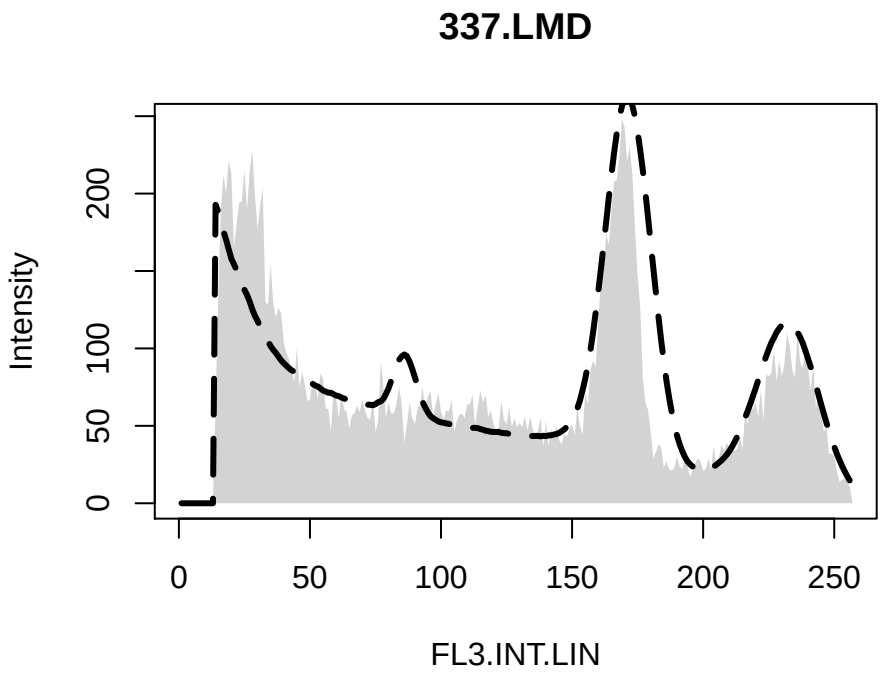
A Comment on Reduced Chi-Square Values

I have also included the Reduced Chi-Square value (RCS), recommended by Bagwell (1993) as an objective assessment of model fit. These values *should* be between 0.7 and 4.0. However, they are very sensitive to various other factors, not all strictly related to goodness of fit (Rabinovitch 1994). In particular, I have noticed that small numbers of events detected in the higher bins (beyond the limits of the modelled peaks) can exert a disproportionate influence on RCS. Compare the previous model fit (which has a long tail of low values) to the next to see an example of this. This could be *fixed* by tweaking the way RCS is calculated, excluding regions where the fitted values are very low. At this point, I'm not sure if this is legitimate or worthwhile.

Manually Selecting Starting Values

Sometimes, particularly with noisy histograms, `flowPloidy` will not produce useful starting values:

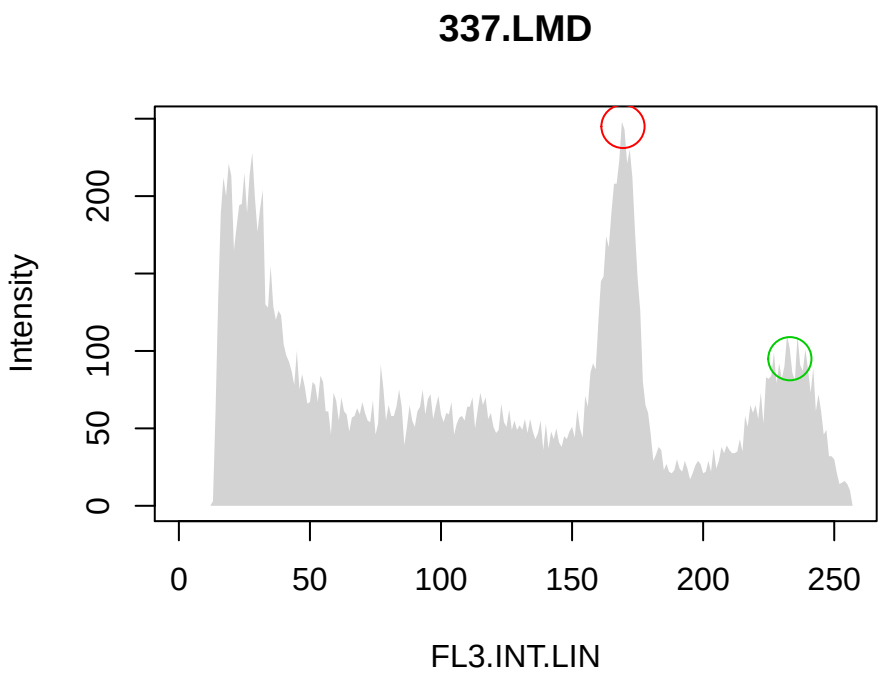
```
file2 <- system.file("extdata", "337.LMD", package = "flowPloidy")
fh2 <- flowHist(FILE = file2, CHANNEL = "FL3.INT.LIN")
plot(fh2, init = TRUE)
```



Here we can see that a high spike in the debris field has confused the peak-detection algorithm. We can manually set the starting values with `pickInit`:

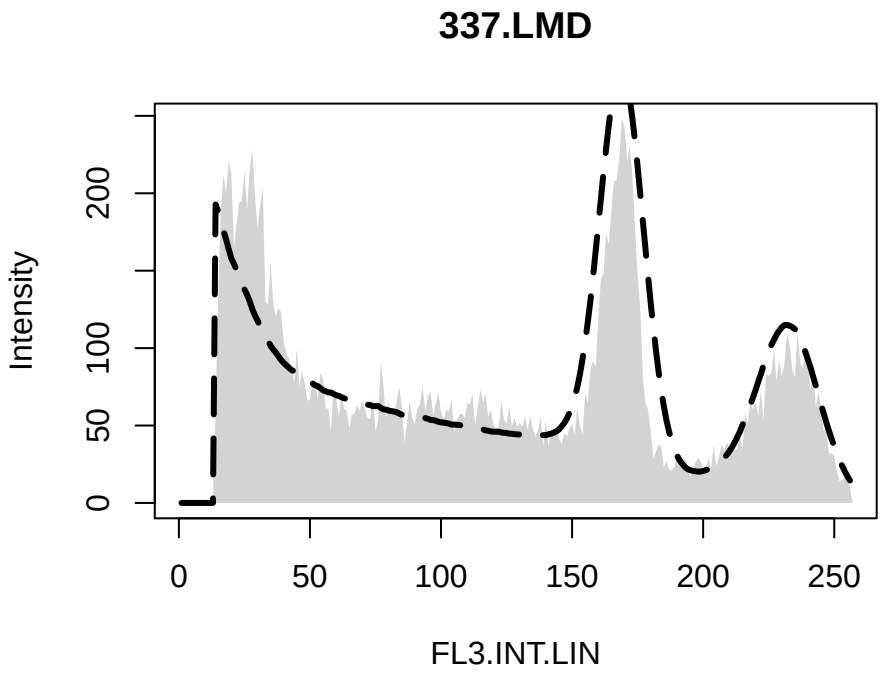
```
fh2 <- pickInit(fh2)
```

`flowHist` will prompt you to click on the peaks of the first and second peak manually, adding a circle to the plot at each point.



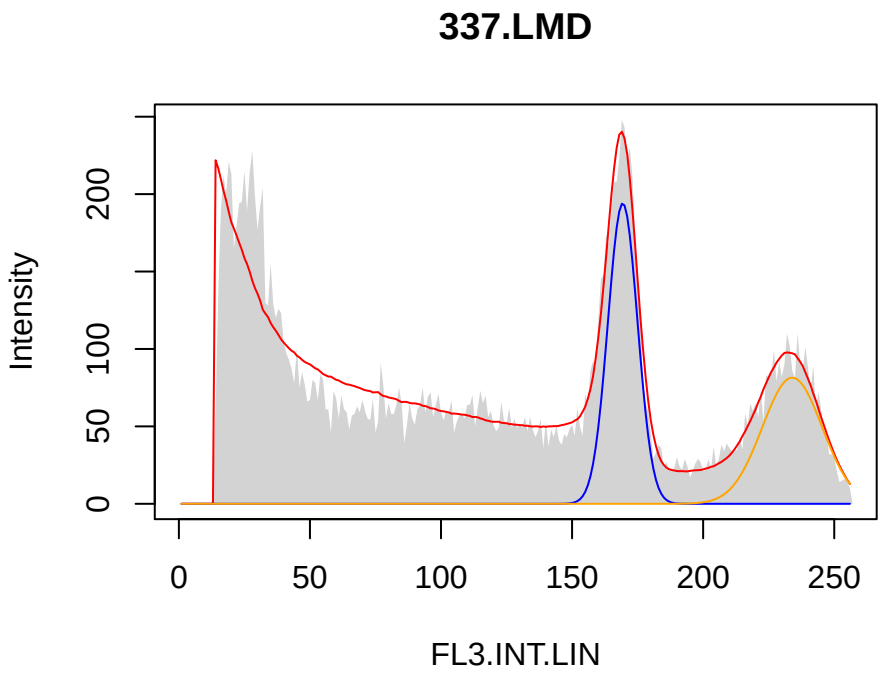
The new peak values are used to recalculate the initial values:

```
plot(fh2, init = TRUE)
```



Now we are ready for the analysis:

```
fh2 <- fhAnalyze(fh2)
plot(fh2)
```




```
fh2
```

```
## flowHist object
## ================
## Source file: 337.LMD
## Channel: FL3.INT.LIN
## Values: 256
## Total events: 18948
## Model components: singleCut, fA1, fB1
## Standard not specified
## Fit!
##
## Analysis
## ========
## Modelled events:  18960.7
## Ratio Peak A / Peak B: 0.724, SE: 0.00292
##
## |       |   counts|    size|   cvs|
## |:------|--------:|-------:|-----:|
## |Peak A | 2742.841| 169.322| 0.033|
## |Peak B | 2271.791| 233.990| 0.049|
##
## RCS: 3.093
```

Exporting Results

To summarize the results in a data table, use the `exportFlowHist` command:

```
exportFlowHist(fh1)
```

```
##         file     channel            components totalEvents modelledEvents
## 1 188-15.LMD FL3.INT.LIN singleCut;fA1;fA2;fB1       10351       10117.85
##   countsA  countsB    sizeA   sizeB        cvA        cvB   ratioAB
## 1 1452.54 2847.929 99.00657 136.136 0.02208578 0.01998339 0.7272621
##        ratioSE      rcs
## 1 0.0004339909 9.298841
```

To combine multiple objects in a single summary, combine them as a list:

```
exportFlowHist(list(fh1, fh2))
```

```
##         file     channel            components totalEvents modelledEvents
## 1 188-15.LMD FL3.INT.LIN singleCut;fA1;fA2;fB1       10351       10117.85
## 2    337.LMD FL3.INT.LIN     singleCut;fA1;fB1       18948       18960.67
##    countsA  countsB     sizeA    sizeB        cvA        cvB   ratioAB
## 1 1452.540 2847.929  99.00657 136.1360 0.02208578 0.01998339 0.7272621
## 2 2742.841 2271.791 169.32221 233.9903 0.03329260 0.04886879 0.7236292
##        ratioSE      rcs
## 1 0.0004339909 9.298841
## 2 0.0029154376 3.093300
```

As a convenience, if you pass a file name to the `file` argument of `exportFlowHist`, the table will be saved to that file:

```
exportFlowHist(list(fh1, fh2), file = "flow-results.csv")
```

TODO

- [ ] add linearity parameter
- [ ] convert to S4 classes for integration with flowCore and inclusion in BioConductor
- [ ] gating
- [ ] workflow for processing large numbers of FCS files, saving detailed analysis (not just summary table).

References

Bagwell, C. Bruce. 1993. “Theoretical Aspects of Flow Cytometry Data Analysis.” In *Clinical Flow Cytometry: Principles and Applications*, edited by Kennth D. Bauer, Ricardo E. Duque, and T. Vincent Shankey, 41–61. Williams & Wilkins.

Rabinovitch, Peter S. 1994. “DNA Content Histogram and Cell-Cycle Analysis.” In *Methods in Cell Biology*, edited by Zbigniew Darzynkiewicz, J. Paul Robinson, and Harry A. Crissman, 41:263–96. San Diego, California: Academic Press.