Estimate proliferation in cell-tracking dye studies

Davide Rambaldi

May 3, 2016

Abstract

Background In cells proliferation tracking experiments, cells are stained with a tracking dye before culture. During cell division, the dye will be divided between daughter cells so that each daughter cell brings about a halving of fluorescence intensity of the mother. The fluorescence intensity of a cell, by comparison with the fluorescence intensity of resting cells, provides an indication of how many divisions the cell has undergone since stimulation. Analysis of the florescence of cells in the flow cytometric data file acquired after some days of culture (with stimulation of cell proliferation) provides an estimate of the percentage of cells for generation (**?**), (**?**).

Methods: This package uses an R implementation of the Levenberg-Marquardt algorithm (minpack.lm) to fit a set of peaks (corresponding to different generations of cells) over the proliferation-tracking dye distribution in a FACS experiment. The package in integrated with other Bioconductor softwares for analysis of flow cytometry datasets like flowCore and flowViz (**?**).

Results: We have tested the fitting algorithm with three samples of mouse lymphocytes stained with carboxyfluorescein diacetate succinimidyl ester (CFSE), Cell Trace Violet (CTV) and Cell Proliferation Dye eFluor 670 (CPD) (**?**)

keywords Flow Cytometry, high throughput, data fitting

1 Introduction

Fluorescent dyes are increasingly being exploited to track cells migration and proliferation. The typical experimental pipeline with tracking dyes involves the following steps:

1 Label cells with a Proliferation Tracking Dye

2 Culture in vitro +/- stimulus for 0-10 days

3 Counterstain with subset specific antibodie

4 Analyze by flow cytometry

In the Flow Cytometer:

1 Acquire with FACS a sample of unlabelled cells

2 Acquire with FACS a sample of labeled and unstimulated cells (the Parent Population)

3 Acquire with FACS a sample of labeled and stimulated cells (the Proliferative Population)

In the last 20 years, a set of fluorochromes has been used to track cell proliferation (**?**). We can divide those dyes in 4 major groups (with some examples):

- **DNA-binding fluorescent dyes:**
  - Hoechst 33342
  - Thiazole Orange
- **Cytoplasmic fluorescent dyes:**
  - Calcein
  - BCECF-AM
- **Covalent coupling fluorescent dyes:**
  - Carboxyfluorescein diacetate succinimidyl ester (CFSE)
  - Fluorescein isothiocyanate (FITC)
- **Membrane-inserting fluorescent dyes:**
  - PKH26
  - CellVue Lavender

If the characteristics of the logarithmic amplifier on the flow cytometer are known, it is possible to derive, from a histogram of fluorescence intensity, the proportion of cells that have undergone any particular number of divisions (**?**). Generally, a flow cytometer with 1024 channels (channels range) represents a nominal range of 4 log-decades. The calibration can be done using standardized beads (for example: Rainbow Beads from Spherotech) (**?**).
Here we introduce flowFit, an R bioconductor library that fit a set of peaks (corresponding to different generations of cells) over the histogram of fluorescence intensity acquired during a FACS experiment. The package is integrated with the Bioconductor libraries used for the analysis of flow cytometry datasets: flowCore and flowViz.

2 Methods

2.1 Single Peak Formula

The algorithm fit a set of N peaks on the acquired fcs files using the minpack.lm. The Levenberg-Marquardt algorithm provides a numerical solution to the problem of minimizing a function over a space of parameters. It is an iterative technique that locates the minimum of a multivariate function. The "minimum" is expressed as the sum of squares of non-linear real-valued functions.
We fit an initial single peak on the population of cells labeled and un-stimulated (the Parent Population) according to this formula:

$$a^2 \exp \frac{(x-\mu)^2}{2\sigma^2}$$

Where a^2 is proportional to the height of the peak, μ is the peak position on the FACS scale and σ is proportional to the peak size.
Once we have fitted a single peak on the parent population, we can use the parent peak position and size as parameters for the fitting on the proliferative population. Given that the tracking dye is partitioned equally between daughter cells, the noise in the data is mainly due to the variance in the initial staining. The formula for the next peak (corresponding to cells that have divided one time) will be:

$$b^2 \exp \frac{(x-(\mu-D))^2}{2s^2}$$

Where D is the estimated distance between 2 generations of cells.

2.2 Generations Distance, data range and log decades

The distance between two cell generations is defined as the distance between a cell and his child (that contains half of the dye of the mother). This distance is constant on a logarithmic scale and depends from the number of data points in the FACS instrument and from the log decades. We can convert the FACS Fluorescence Intensity (FFI) recorded by the instrument into a Relative Fluorescence Intensity (RFI) expressed in Molecules of Equivalent Fluorochrome (MEF):

$$RFI = 10^{\frac{FFI \cdot l}{c}}$$

The inverse formula can be used to convert from RFI to FACS fluorescence:

$$FFI = \frac{c \cdot \log(RFI)}{(l \cdot \log(10))}$$

Where:

- RFI is the Relative Fluorescence Intensity

- FFI is the fluorescence on the FACS scale
- l is the number of log decades in the FACS instrument
- c is the number of data points (channels) in the instrument.

Using this formulas it is possible to estimate the spacing between generations on the FACS scale. The spacing value is automatically computed, based on the number of decades and the assumption that each generation has one-half of the intensity of the previous generation. In the flowFit library this spacing is automatically computed with the function `generationsDistance`.
The number of log decades represented by the original data's linear scale can be calculated as $log(R)$ where R is the acquisition resolution (data range for the detector).
The `proliferationFitting` and `parentFitting` functions automatically compute the log decades from the keywords in the FCS file or using the logarithm of the data Range ($log(R)$). If the functions find a "log decades" keyword for the current detector in the FCS file, they use the value in the keyword, otherwise log decades are estimated from the detector acquisition resolution.

2.3 Fitting Formula

The algorithm automatically computes the number of peaks to be fitted on the proliferating population using the data range for the detector: we first estimate the distance between 2 generations of cells using the generationsDistance function, then we estimate the maximum number of peaks that can be fitted on the FACS instrument scale with the following formula:

$$N = \frac{D}{c}$$

Where:

- N is the number of peaks to fit on the dataset
- D is the distance between 2 generations of cells
- c is the number of data point (channels) in the instrument

The formula for a model with 10 peaks will be:

$$M = a^2 \exp\frac{(x-M)^2}{2s^2} + b^2 \exp\frac{(x-(M-D))^2}{2s^2} + c^2 \exp\frac{(x-(M-2D))^2}{2s^2} + d^2 \exp\frac{(x-(M-3D))^2}{2s^2} + e^2 \exp\frac{(x-($$

Where the parameters a,b,c,d,e,f,g,h,i and l are proportional to the height of the peak. In the "fixed model" (options fixedModel=TRUE) the minpack.lm algorithm estimates the height of each peak but allows the user to keep constant one ore more of these variables: parent peak position (m), parent peak size (s) and generation's distance (D). In the "dynamic model" (options fixedModel=FALSE) the minpack.lm algorithm uses as parameters all the variables in

the model: the height of each peak, the parent peak position (m), the parent peak size (s) and the generation's distance (D).
In the Levenberg-Marquadt algorithm implementation we use this formula to estimate the error between the model and the real data:

$$residFun = (Observed - Model)^2$$

2.4 % of cells for generation

With the `nls.lm` function we have estimated the "heights" parameter (the height of each peak), we can use it to get an estimate of the number of cells for generation. We compute the total number of cells analyzed in the model with a numerical integration on the complete model:

$$I_{all} = \int_v^w M$$

Where v and w, the lower and upper limits for numerical integration, are the `minRange` and `maxRange` values for the analyzed fcs column. For each generation, we compute the number of cells in the peak with a numerical integration. For example, the formula for the first generation of cells is:

$$I_1 = \int_v^w a^2 \exp \frac{(x-\mu)^2}{2\sigma^2}$$

Where v and w, the lower and upper limits for numerical integration, are the `minRange` and `maxRange` values for the analyzed fcs column.
To estimate the % of cells for generation we simply take the ratio between the complete model numerical integration and the integration of a single generation of cells:

$$Gen_1 = \frac{I_1}{I_{all}} * 100$$

2.5 Proliferation Index

To evaluate the proliferation in the sample we can use the `proliferationIndex` function: proliferation index is calculated as the sum of the cells in all generations including the parental divided by the computed number of original parent cells theoretically present at the start of the experiment.
The proliferation index it's a measure of the fold increase in cell number in the culture over the course of the experiment:

$$\frac{\sum_0^i N_i}{\sum_0^i N^i/2^i}$$

Where i is the generation number (parent generation = 0). In the absence of proliferation, that is, when all cells are in the parent generation, the formula gives:

$$\frac{N_0}{N_0/2^0} = 1$$

defining the lower limit of the PI.

3 Fitting Flow Cytometry Data

The primary task offlowFit is to estimate the number of cells for generation in a proliferation tracking dye study. This is accomplished with the functions: `parentFitting` and `proliferationFitting`.
In order to reduce the download size of flowFit, we have moved the example dataset to a Bioconductor data package (flowFitExampleData) that can be downloaded in the usual way.

```
> source("http://www.bioconductor.org/biocLite.R")
> biocLite("flowFitExampleData")
```

First of all we must load the package and the examples:

```
> library(flowFitExampleData)
> library(flowFit)
```

We must also load the `flowCore` package to load some functions used in this vignette:

```
> library(flowCore)
```

In this example we will use the package flowFit to calculate the percentage of cells for generation in a single CD4+ population of lymphocytes labeled with: CFSE, CPD and CTV. flowFit use the Bioconductor package flowCore to load .FCS files. For this example, we provide some example data in the flowFitExampleData that can be loaded in your environment with the command:

```
> data(QuahAndParish)
```

The QuahAndParish dataset contains 4 FCS files, those files are the raw data of the figure 2 (CD4+ subpopulation) in the paper: **New and improved methods for measuring lymphocyte proliferation in vitro and in vivo using CFSE-like fluorescent dyes** (**?**).

1. CD4+ labeled with CFSE, CPD and CTV (Non stimulated)
2. CD4+ labeled with CFSE (stimulated)
3. CD4+ labeled with CPD (stimulated)
4. CD4+ labeled with CTV (stimulated)

3.1 Parent Fitting

First of all we must estimate the parent population position and size with the function `parentFitting`. The first sample in the flowSet, correspond to the cells non stimulated and stained with the 3 dyes (parent population):

```
> QuahAndParish[[1]]

flowFrame object 'Fig 2a All CD4 T Nonstim.fcs'
with 4541 cells and 13 observables:
                    name desc  range     minRange     maxRange
$P1                FSC-A      262207 51906.250000 2.622060e+05
$P2                FSC-H      262207 47800.500000 2.622060e+05
$P3                FSC-W      262207 68451.750000 2.622060e+05
$P4                SSC-A      261588    90.194656 2.615870e+05
$P5                SSC-H       29999   100.010956 2.999800e+04
$P6                SSC-W      262207 47132.250000 2.622060e+05
$P7             <FITC-A>      261588     3.077193 5.417616e+00
$P8               <PE-A>      261588     3.450857 5.417616e+00
$P9            <PE-Cy5-A>      261588     0.000000 5.417616e+00
$P10 <Alexa Fluor 405-A>      261588     0.000000 5.417616e+00
$P11 <Alexa Fluor 430-A>      261588     0.000000 5.417616e+00
$P12              <APC-A>      261588     0.000000 5.417616e+00
$P13          <APC-Cy7-A>      261588     0.000000 5.417616e+00
154 keywords are stored in the 'description' slot
```

We fit a parent population peak for each dye (CFSE, CPD and CTV) (See figure **??**,figure **??**,figure **??**). To estimate position and size of the parent population peak, we must have the following informations on the FACS instrument:

1. The FCS detector name: the name of the FCS column in the *flowFrame*
2. The data range for the FCS detector (after transformations).
3. The log decades for the FCS detector: the number of log decades represented by the original data's linear scale

3.1.1 Fitting And Plots

CFSE Parent Population

```
> parent.fitting.cfse <- parentFitting(QuahAndParish[[1]], "<FITC-A>")
> plot(parent.fitting.cfse)
```

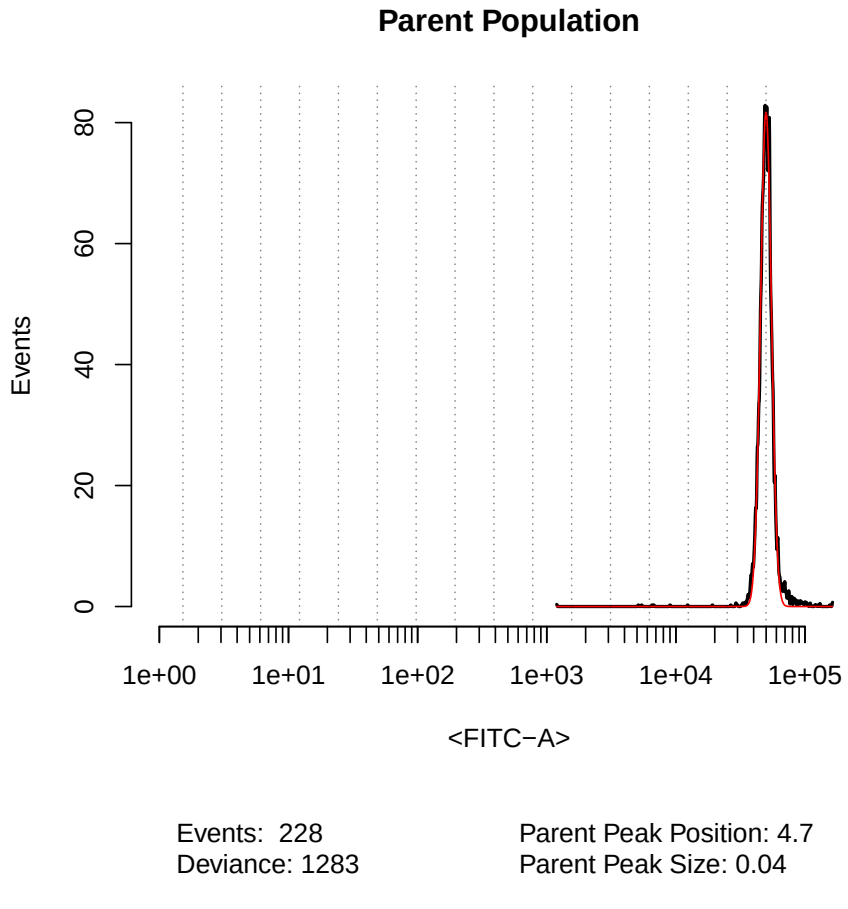


Figure 1: CD4+ CELLS PARENT FITTING CFSE

```
> parent.fitting.cpd <- parentFitting(QuahAndParish[[1]], "<APC-A>")
> plot(parent.fitting.cpd)
```

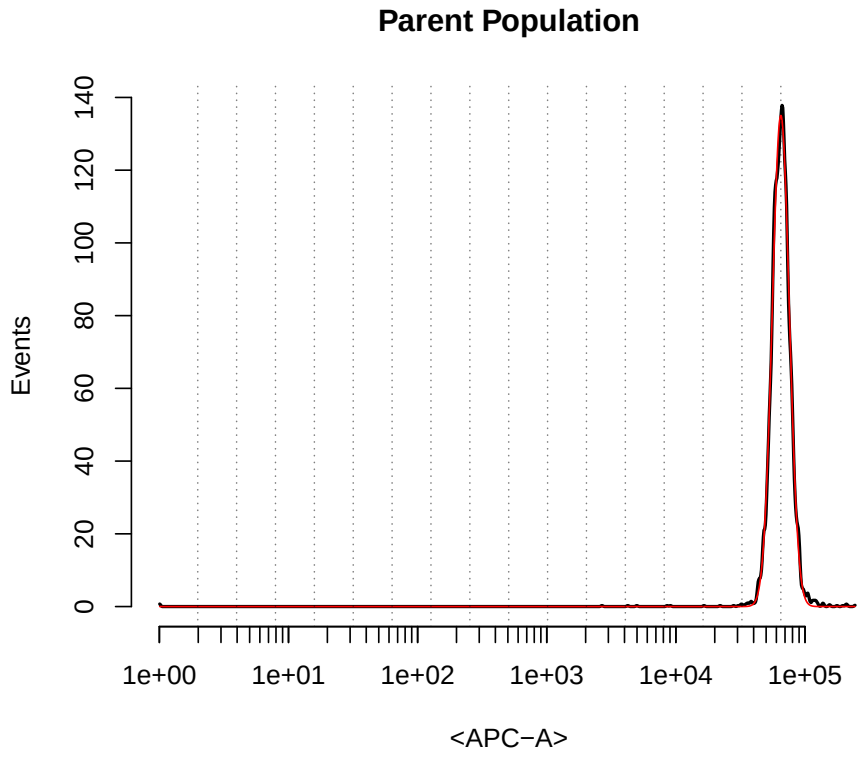


Figure 2: CD4+ CELLS PARENT FITTING CPD

```
> parent.fitting.ctv <- parentFitting(QuahAndParish[[1]], "<Alexa Fluor 405-A>")
> plot(parent.fitting.ctv)
```

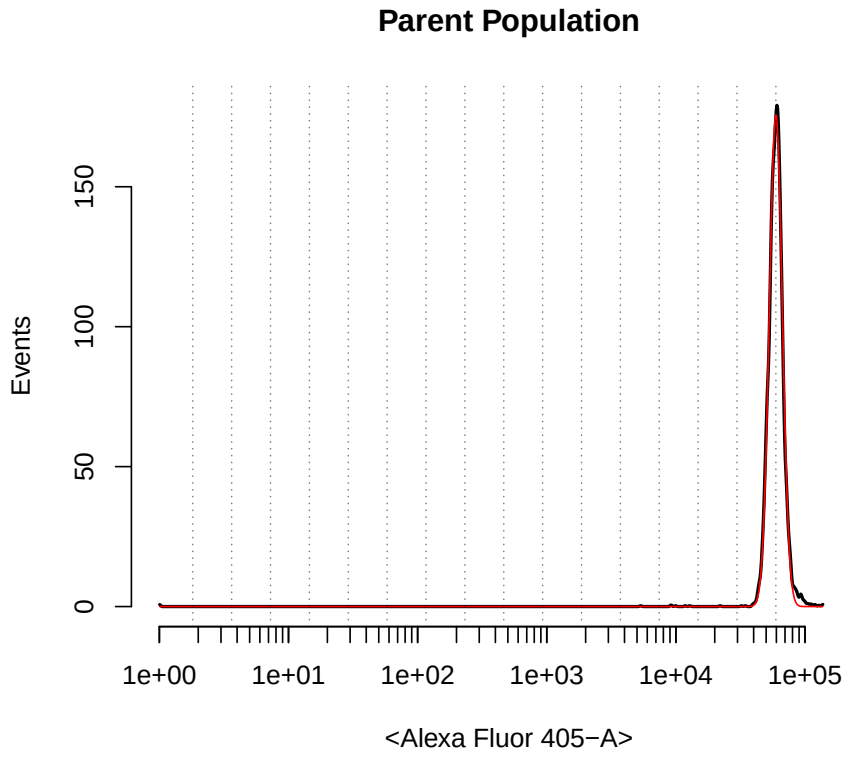


Figure 3: CD4+ CELLS PARENT FITTING CTV

3.1.2 summary and coef

The `summary` function produces some result summaries:

```
> summary(parent.fitting.cfse)

********* flowFit: Parent Population Data Object *********
* Data Range = 5.417616
* Log Decades = 5.417618
* Channel used for fitting = <FITC-A>
* Number of events in flowFrame = 4541
******************** Fitting **********************
* Unfixed model.
* Parent Peak Position = 4.697861
* Parent Peak Size = 0.04219839
* Fitting Deviance = 1283.347
* Termination message:
* `ptol' is too small. No further improvement in the approximate solution `par' is possible.
* Iterations: 14
* Residual degrees-of-freedom: 1066
***********************************************************
```

We can extract the confidence interval for our fitting with the function `confint`:

```
> confint(parent.fitting.cfse)

       2.5 %     97.5 %
a 9.01517880 9.06282697
M 4.69760377 4.69811748
S 0.04194153 0.04245524
```

3.1.3 dataRange

The data range for the FACS detector is automatically computed from the "maxRange" slot in the *flowFrame*:

```
> QuahAndParish[[1]]@parameters@data[7,]

        name desc  range minRange maxRange
$P7 <FITC-A>      261588 3.077193 5.417616
```

In this experiment we have transformed the <FITC-A> column with a log transformation:

```
> logTrans <- logTransform(transformationId="log10-transformation",
+  logbase=10, r=1, d=1)
```

The original range [0 - 261588] was converted to a LOG range [0-5.417616].

3.1.4 logDecades

The number of log decades represented by the original data's linear scale can be calculated as $log(R)$ where R is the acquisition resolution (data Range for the detector). The `proliferationFitting` and `parentFitting` functions automatically compute the logDecades from the keywords in the FCS file or using the $log(R)$ formula. If the functions find a log decades keyword for this detector, they use the value in they keyword. Otherwise log decades are estimated from the detector acquisition resolution. In the PKH26 example data, log decaedes are extracted using the function `keyword` from package flowCore:

```
> data(PKH26data)
> keyword(PKH26data[[1]])$`$P4M`

[1] "   4.00"
```

In the QuahAndParish example data there is no log decades keyword in the fcs file, log decades are estimated with the $log(R)$ formula.

```
> acquisiton.resolution <- QuahAndParish[[1]]@parameters@data$range[7]
> log10(acquisiton.resolution)

[1] 5.417618
```

In any case, you can provide your `dataRange` and `logDecades` as arguments of the `proliferationFitting` and `parentFitting` functions.

3.2 Proliferation Fitting

3.2.1 Input

With the `parentFitting` function we have estimated the position and size of the parent population. For example, in the CFSE sample:

```
> parent.fitting.cfse@parentPeakPosition

[1] 4.697861

> parent.fitting.cfse@parentPeakSize

[1] 0.04219839
```

We can use the estimates as a best guess for the `proliferationFitting` function:

```
> fitting.cfse <- proliferationFitting(QuahAndParish[[2]],
+   "<FITC-A>", parent.fitting.cfse@parentPeakPosition,
+   parent.fitting.cfse@parentPeakSize)
> fitting.cfse

********* flowFit: Final Population Data Object *********
* Data Range = 5.417616
* Log Decades = 5.417618
* Channel used for fitting = <FITC-A>
* Number of events in flowFrame = 16556
*********************************************************
* FITTING with:
* Unfixed model.
* Number of Peaks in the model = 10
* Parent Peak Position = 4.617102
* Parent Peak Size = 0.06339392
* Estimated generations distance:0.294378
* Fitting Deviance = 22103.37
********************** Generations **********************
* generation 1 = 11.42 %
* generation 2 = 20.9 %
* generation 3 = 21.95 %
* generation 4 = 22.6 %
* generation 5 = 19.32 %
* generation 6 = 11.53 %
* generation 7 = 4.07 %
* generation 8 = 0.93 %
* generation 9 = 0.07 %
* generation 10 = 0.02 %
*********************************************************
```

As for the `parentFitting-data` object, we have access to the functions `coef` and `summary`.

```
> coef(fitting.cfse)

         a          b          c          d          e          f
 7.15029353  9.67210238  9.91179799 10.05621495  9.29823950  7.18322117
         g          h          i          j          M          S
 4.26702380  2.04458814  0.55925520  0.29099380  4.61710241  0.06339392
         D
 0.29437801

> summary(fitting.cfse)

********* flowFit: Final Population Data Object *********
* Data Range = 5.417616
* Log Decades = 5.417618
* Channel used for fitting = <FITC-A>
* Number of events in flowFrame = 16556
******************** Fitting **********************
* Unfixed model.
* Number of Peaks in the model = 10
* Parent Peak Position = 4.617102
* Parent Peak Size = 0.06339392
* Estimated generations distance:0.294378
* Fitting Deviance = 22103.37
******************** Model **********************
* Termination message:
 * `ptol' is too small. No further improvement in the approximate solution `par' is possible.
* Iterations: 29
* Residual degrees-of-freedom: 658
******************** Parameters ********************
a = 7.15029353378665
b = 9.67210237733387
c = 9.91179799485009
d = 10.0562149515981
e = 9.29823950157696
f = 7.18322116936387
g = 4.26702379704017
h = 2.04458813658271
i = 0.559255195120004
j = 0.290993800883944
********************** Generations **********************
* generation 1 = 11.42 %
* generation 2 = 20.9 %
* generation 3 = 21.95 %
* generation 4 = 22.6 %
```

```
* generation 5 = 19.32 %
* generation 6 = 11.53 %
* generation 7 = 4.07 %
* generation 8 = 0.93 %
* generation 9 = 0.07 %
* generation 10 = 0.02 %
*********************************************************
```

3.2.2 Plots

The function `proliferationFitting` return a `proliferationFittingData` object, with the `plot` function we can generate some graphic output. You can use the parameter *which* to select the plots you want to draw. You can plot:

1. Input data
2. Model data and Fitting
3. Fitting and single generations peaks
4. Generations barplot
5. Fitting diagnostics

All `plot` functions in the flowFit package support these options:

1. `main`: an overall title for the plot.
2. `xlab`: a title for the x axis.
3. `ylab`: a title for the y axis.
4. `legend`: show/hide messages AND legend.
5. `logScale`: put a log scale on the x axis.
6. `drawGrid`: put some dashed lines at the `generationsDistance` expected positions (`proliferationFitting` object only).

For more info about plots in flowFit:

```
> library(flowFit)
> ?plot
```

Input Data Plot the observed data extracted from the FCS file and the data smoothed with the `kz` function (`kza` package).

```
> plot(fitting.cfse, which=1)
```

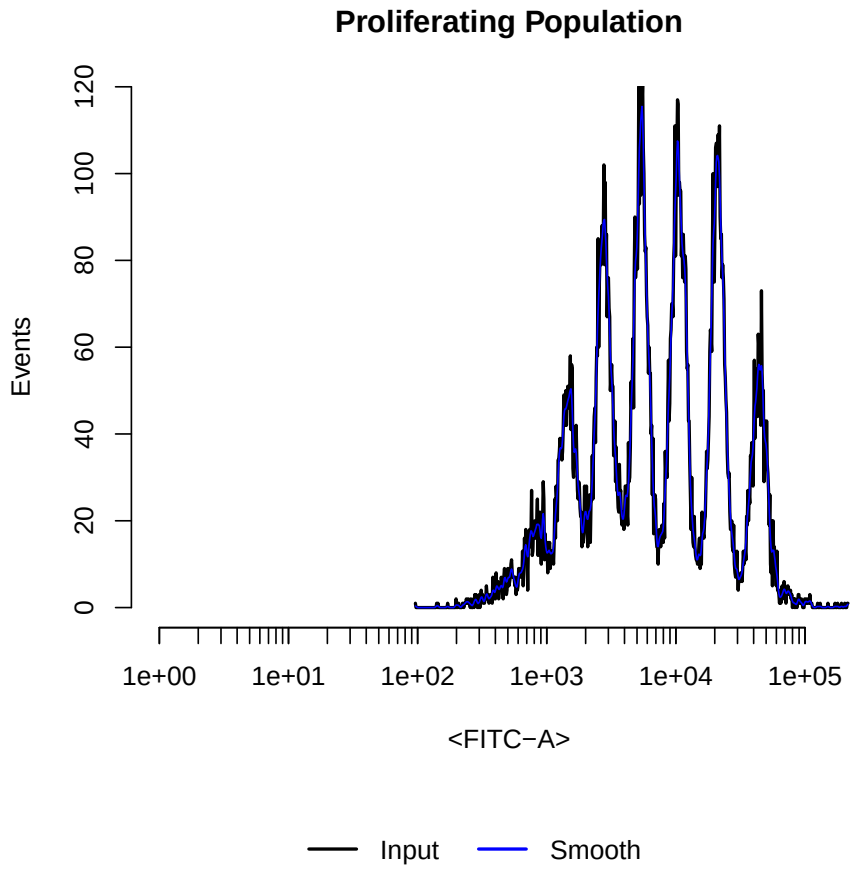


Figure 4: Input Data: `plot(fitting.cfse, which=1)`

Model data and Fitting Plot the smoothed data and the fitted model function.

```
> plot(fitting.cfse, which=2)
```

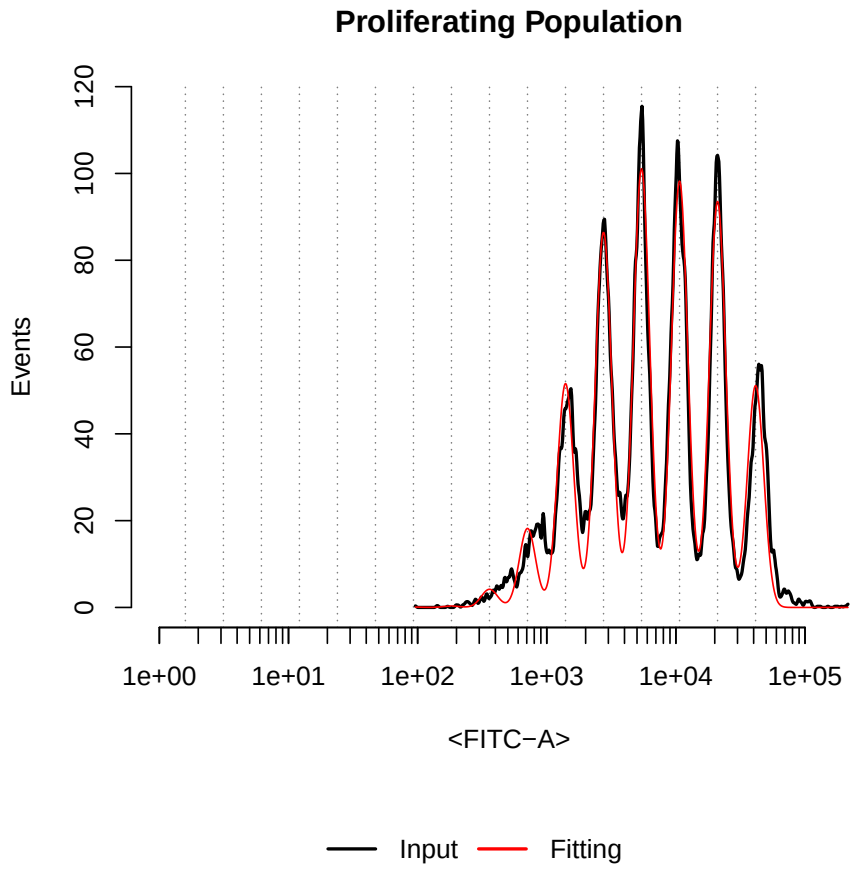


Figure 5: Model data and Fitting: `plot(fitting.cfse, which=2)`

Fitting and single generations peaks Plot the smoothed data, the fitted model function and a single peak for each generation.

```
> plot(fitting.cfse, which=3)
```

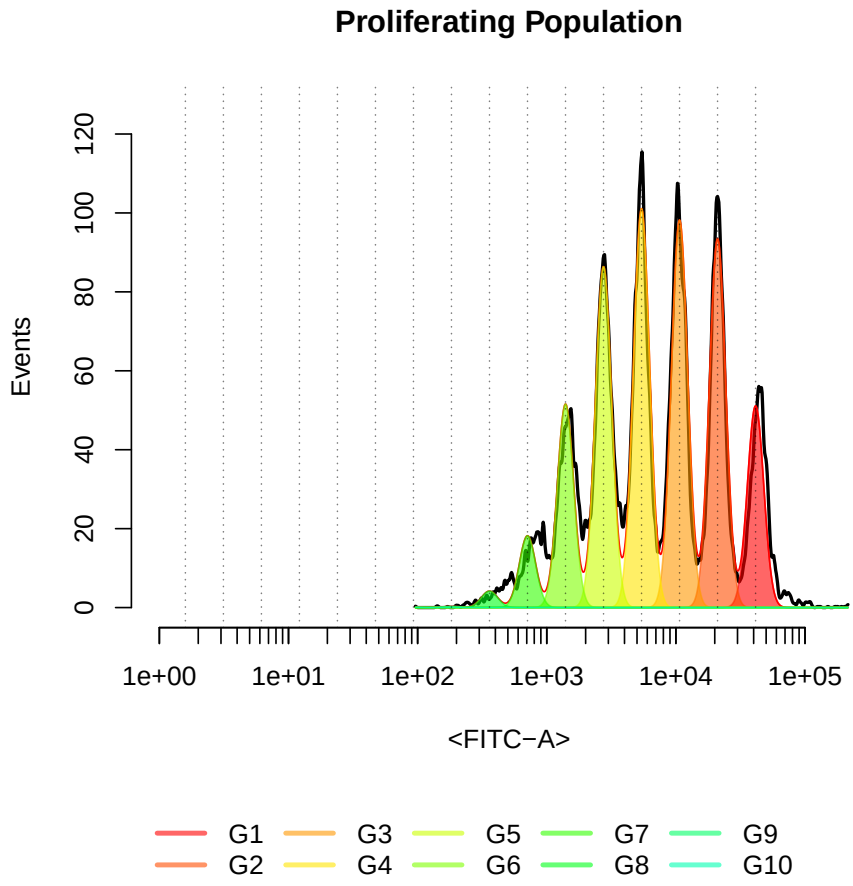


Figure 6: Fitting and single generations peaks: plot(fitting.cfse, which=3)

Generations barplot Plot the estimated numer of cells for generation (%) as a barplot.

```
> plot(fitting.cfse, which=4)
```

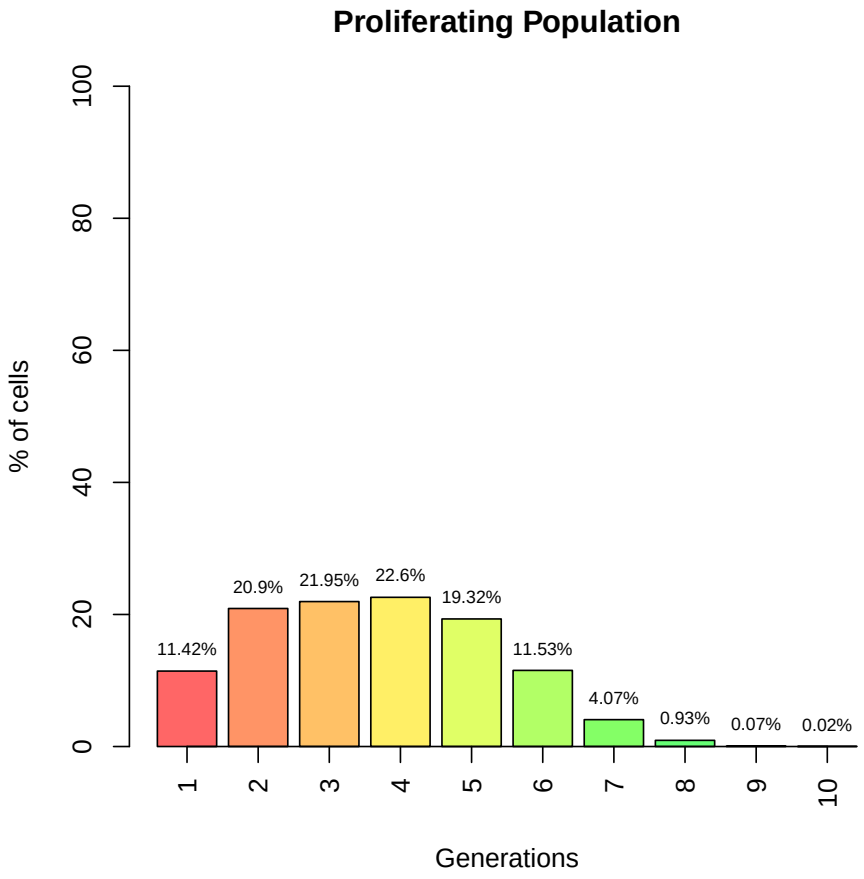
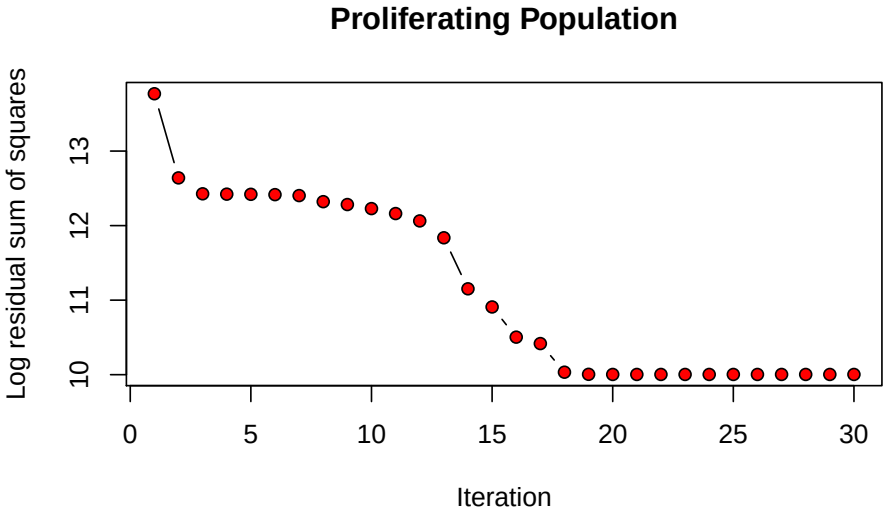


Figure 7: Generations barplot: plot(fitting.cfse, which=4)

Fitting diagnostics Plot the log residual sum of squares against the iteration number and some informations about the fitting:

1. Number of events
2. Fitting Deviance
3. Parent Population Peak position (before and after modeling)
4. Parent Population Peak size (before and after modeling)

```
> plot(fitting.cfse, which=5)
```



Events: 542
Deviance: 22103
[Distance] model: 0.29 , guess: 0.3
[Parent Peak Position] model: 4.62 , guess: 4.7
[Parent Peak Size] model: 0.06 , guess: 0.04

Figure 8: Fitting diagnostics: plot(fitting.cfse, which=5)

Multiple plots We can combine multiple plots removing the `legend` and using the `mfrow` option of `par`.

```
> par(mfrow=c(2,2))
> plot(fitting.cfse, which=1:4, legend=FALSE)
```

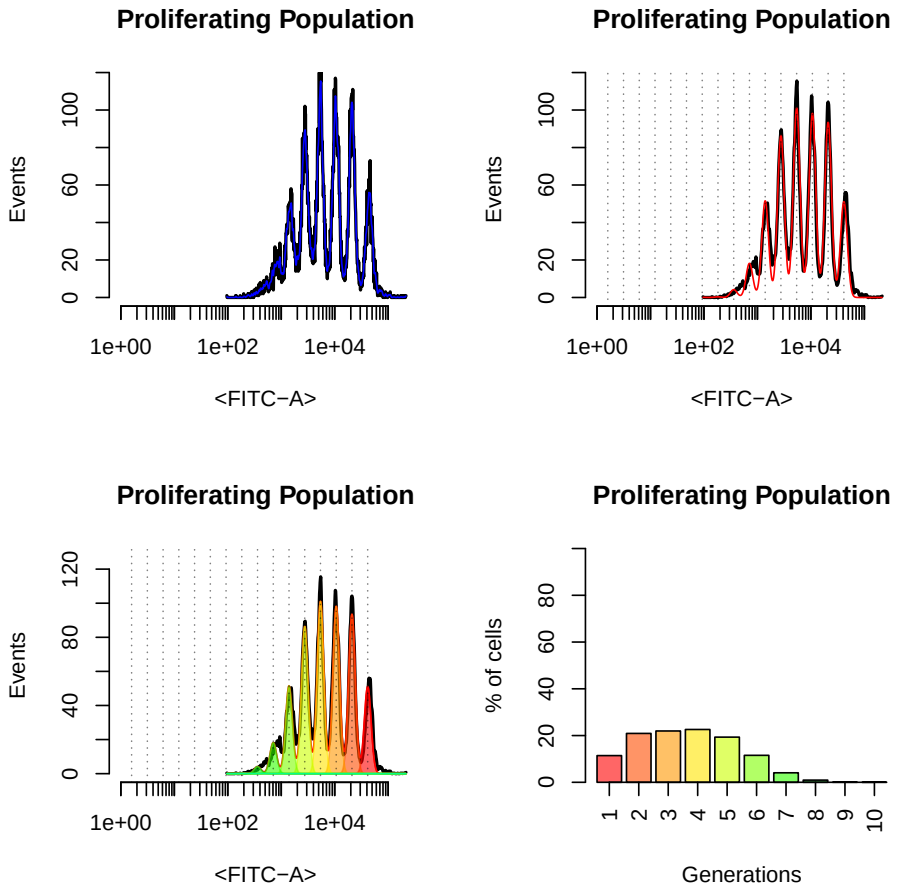


Figure 9: Multiple plots: plot(fitting.cfse, which=1:4, legend=FALSE)

```
> par(mfrow=c(2,1))
> plot(fitting.cfse, which=c(3,5), legend=FALSE)
```

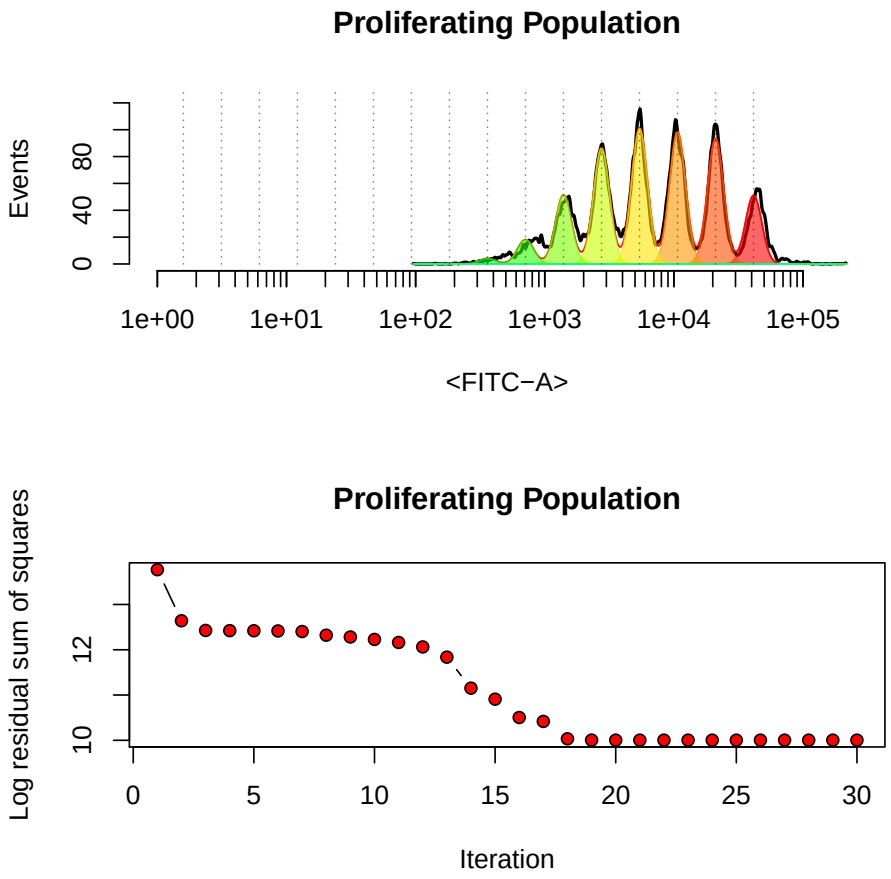


Figure 10: Multiple plots: plot(fitting.cfse, which=c(3,5), legend=FALSE)

3.2.3 Cells for generation

The percentage of cells for each generation can be extracted from the `proliferationFitting-Data` object with the function `getGenerations`, the data are in a `list`.

```
> gen <- getGenerations(fitting.cfse)
> class(gen)
```

```
[1] "list"
```

We can extract also the percentage of cells for generation as a vector instead of a list. To do this, you need to use the `generations` slot in the `proliferationFittingData` object:

```
> fitting.cfse@generations
```

```
 [1] 11.42454593 20.90417229 21.95311191 22.59749517 19.31935680 11.53000999
 [7]  4.06856573  0.93412045  0.06988939  0.01892164
```

We compute the total number of cells analyzed in the model with a numerical integration on the complete model:

$$I_{all} = \int_v^w M$$

Where v and w, the lower and upper limits for numerical integration, are the `minRange` and `maxRange` values for the analyzed fcs column. For each generation, we compute the number of cells in the peak with a numerical integration. For example, the formula for the first generation of cells is:

$$I_1 = \int_v^w a^2 \exp \frac{(x-\mu)^2}{2\sigma^2}$$

Where v and w, the lower and upper limits for numerical integration, are the `minRange` and `maxRange` values for the analyzed fcs column.
To estimate the % of cells for generation we simply take the ratio between the complete model numerical integration and the integration of a single generation of cells:

$$Gen_1 = \frac{I_1}{I_{all}} * 100$$

3.2.4 Proliferation Index

Finally we can calculate the Proliferation Index (**?**) for this sample. Proliferation index is calculated as the sum of the cells in all generations including the parental divided by the computed number of original parent cells theoretically present at the start of the experiment.

```
> proliferationIndex(fitting.cfse)
```

```
[1] 3.544918
```

The proliferation index it's a measure of the fold increase in cell number in the culture over the course of the experiment:

$$\frac{\sum_0^i N_i}{\sum_0^i N^i/2^i}$$

Where i is the generation number (parent generation = 0). In the absence of proliferation, that is, when all cells are in the parent generation, the formula gives:

$$\frac{N_0}{N_0/2^0} = 1$$

defining the lower limit of the PI.

3.3 Comparing Samples

We can estimate the % of cells for generation in the 3 samples.

1. lymphocytes CD4+ labeled with CFSE (stimulated)
2. lymphocytes CD4+ labeled with CPD (stimulated)
3. lymphocytes CD4+ labeled with CTV (stimulated)

The 3 samples analyzed came from the same cells population and have proliferated in the same way. The only difference in the experiement is the stain used for track the cells proliferation. We expect to observe the same cells/generation distribution across the 3 samples. To compare the generation distributions we can extract the % of cells for generation for the 3 samples and compare those distributions with a chi-squared test.

3.3.1 Proliferation Fitting CTV

A fitting for the CTV sample:

```
> fitting.ctv <- proliferationFitting(QuahAndParish[[4]],
+   "<Alexa Fluor 405-A>", parent.fitting.ctv@parentPeakPosition,
+   parent.fitting.ctv@parentPeakSize)

> plot(fitting.ctv, which=3)
```

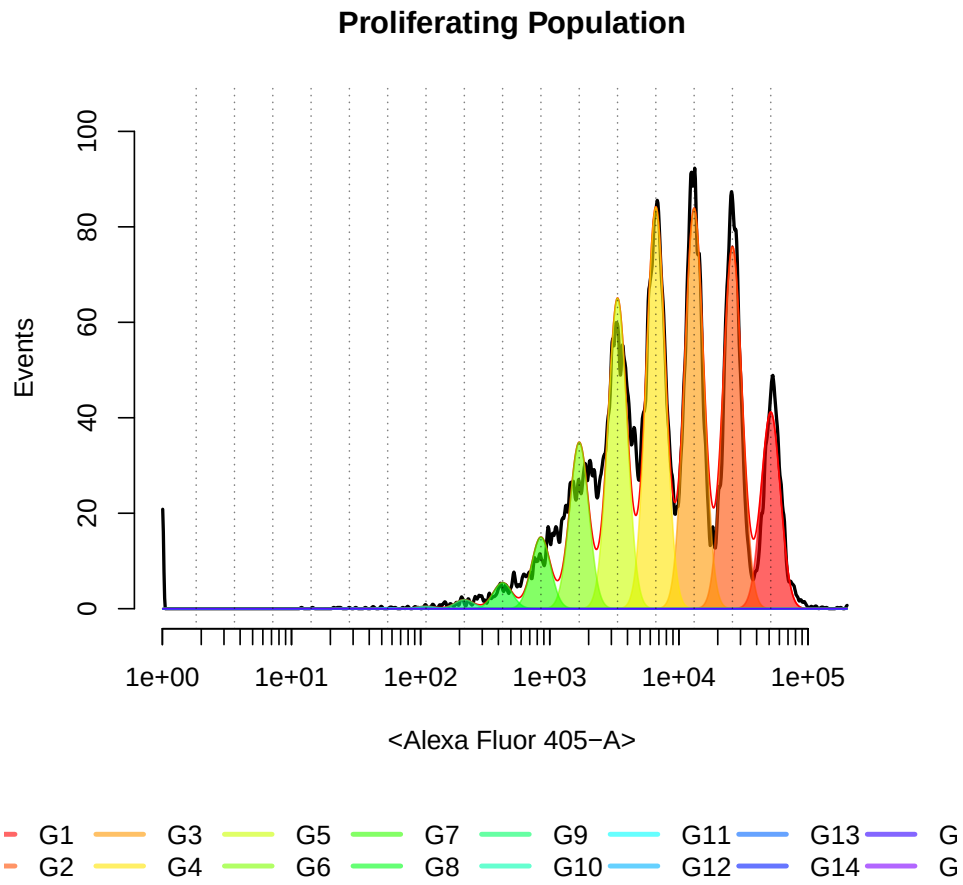


Figure 11: Proliferation Fitting CTV

3.3.2 Proliferation Fitting CPD

A fitting for the CPD sample. In this sample there are no visible peaks:

```
> plot(QuahAndParish[[3]],"<APC-A>", breaks=1024, main="CPD sample")
```

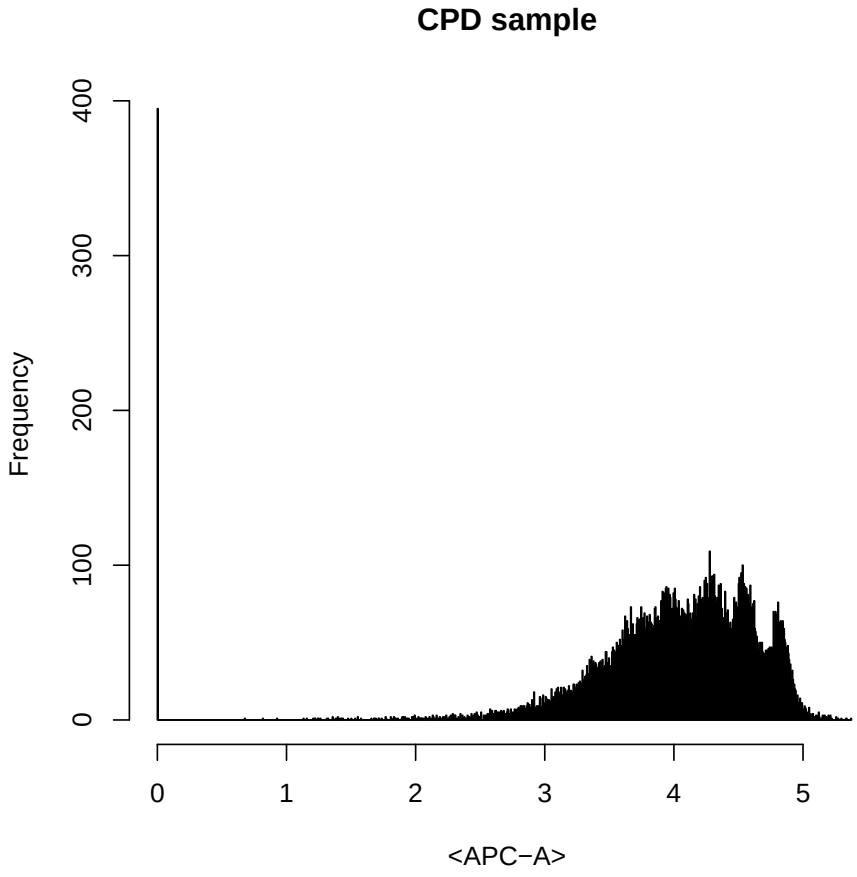


Figure 12: The CPD Sample

To improve the fitting we can keep fixed some parameters from the `parentFitting` function. Here we launch the fitting function (`proliferationFitting`) keeping fixed the `parentPeakPosition`:

```
> fitting.cpd <- proliferationFitting(QuahAndParish[[3]],
+   "<APC-A>",
+   parent.fitting.cpd@parentPeakPosition,
+   parent.fitting.cpd@parentPeakSize,
+   fixedModel=TRUE,
+   fixedPars=list(M=parent.fitting.cpd@parentPeakPosition))

> plot(fitting.cpd, which=3)
```

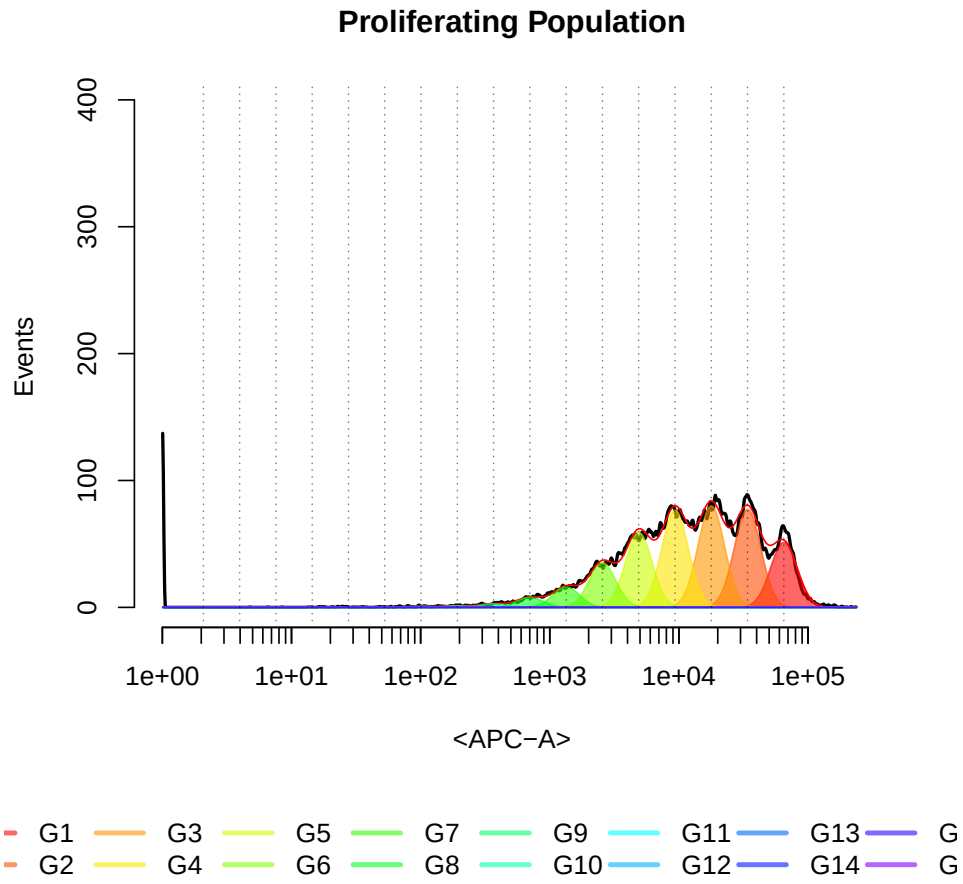


Figure 13: Proliferation Fitting CPD

3.3.3 Comparing Samples

```
> perc.cfse <- fitting.cfse@generations
> perc.cpd <- fitting.cpd@generations
> perc.ctv <- fitting.ctv@generations
```

In the CFSE sample we have estimated 10 generations, while in hte other 2 samples we have estimated 16 generations. We trim the length of the shorter vector with NA:

```
> perc.cfse <- c(perc.cfse, rep(0,6))
```

Chi-squared Test:

```
> M <- rbind(perc.cfse, perc.cpd, perc.ctv)
> colnames(M) <- 1:16
> (Xsq <- chisq.test(M, B=100000, simulate.p.value=TRUE))

        Pearson's Chi-squared test with simulated p-value (based on 1e+05
        replicates)

data:  M
X-squared = 3.3721, df = NA, p-value = 1

> plot(perc.cfse, type="b", axes=F, ylim=c(0,50), xlab="generations", ylab="Percentage of cel
> lines(perc.cpd, type="b", col="red")
> lines(perc.ctv, type="b", col="blue")
> legend("topleft", c("CFSE","CPD","CTV"), pch=1, col=c("black","red","blue"), bg = 'gray90',
> axis(2, at=seq(0,50,10), labels=paste(seq(0,50,10),"%"))
> axis(1, at=1:16,labels=1:16)
> text(8,40,paste("Chi-squared Test p=", round(Xsq$p.value, digits=4), sep=""))
```

According to the Chi-Squared Test, we didn't observe any effect for the different staining technique in the 3 samples.

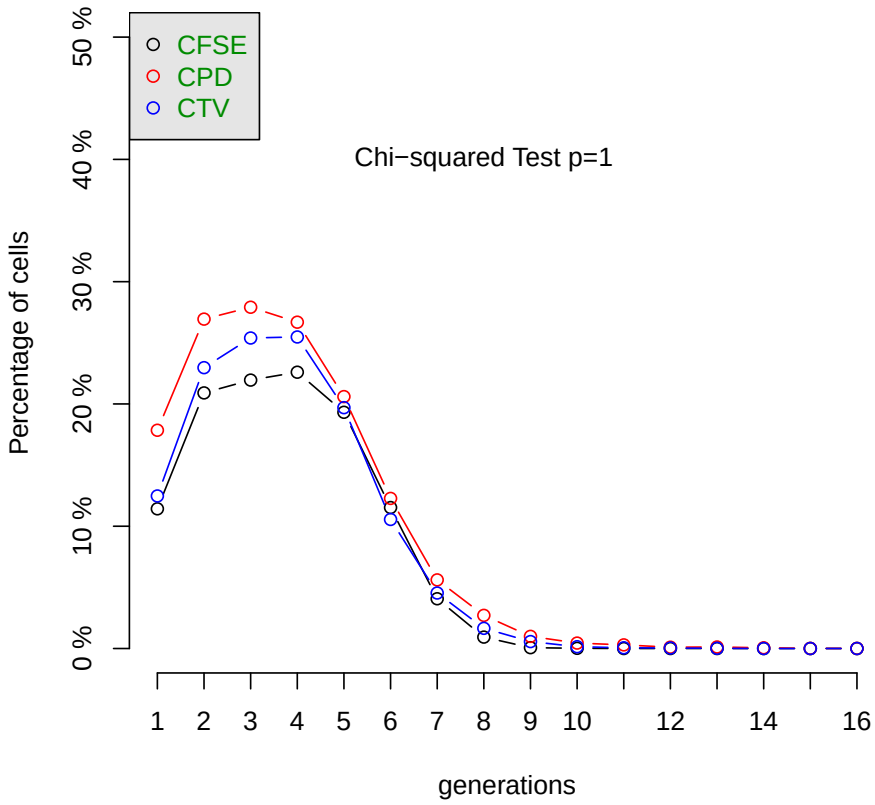


Figure 14: Comparing the % of cells/generation in the 3 samples

4 References