Package ‘flowFit’

November 20, 2016

Type Package
Title Estimate proliferation in cell-tracking dye studies
Version 1.12.0
Date 2012-11-29
Author Davide Rambaldi
Maintainer Davide Rambaldi <davide.rambaldi@gmail.com>
Description This package estimate the proliferation of a cell population in cell-tracking dye studies. The 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.
License Artistic-2.0
LazyLoad yes
Imports flowCore, flowViz, graphics, kza, methods, minpack.lm, gplots
Depends R (>= 2.12.2)
Suggests flowFitExampleData
URL
BugReports Davide Rambaldi <davide.rambaldi@gmail.com>
biocViews FlowCytometry, CellBasedAssays
NeedsCompilation no

R topics documented:

flowFit-package ......................................................... 2
generationsDistance .................................................. 5
getGenerations .......................................................... 6
logTicks ................................................................ 7
parentFitting .............................................................. 8
Description

This package estimate the proliferation of a cell population in cell-tracking dye studies.

In cells proliferation tracking experiments, cells are stained with a tracking dye before culture. During cell division, the tracking dye is partitioned between daughter cells, so that each division brings about a halving of fluorescence intensity; the intensity of a cell, by comparison with the intensity of resting cells, provides an indication of how many divisions the cell has undergone since stimulation.

This package uses an R implementation of the Levenberg-Marquardt algorithm (\texttt{nls.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 define two data structure (S4 classes): \texttt{proliferationFittingData}, \texttt{parentFittingData} and their methods and accessors.

The package is integrated with other \texttt{www.bioconductor.org} libraries for analysis of flow cytometry data: \texttt{flowCore} and \texttt{flowViz}.

Details

- Package: flowFit
- Type: Package
- Version: 0.2
- Date: 2012-11-29
- License: Artistic-2.0

Author(s)

Maintainer: Davide Rambaldi <davide.rambaldi@gmail.com> Author: Davide Rambaldi

References

1. Timur V. Elzhov, Katharine M. Mullen and Ben Bolker (2012). \texttt{minpack.lm}: R interface to the Levenberg-Marquardt nonlinear least-squares algorithm found in MINPACK.


**See Also**

1. `proliferationFitting` generations fitting function.
2. `parentFitting` parent population fitting function.
3. `proliferationIndex` proliferation index calculator.
4. `getGenerations` get percentage of cells for generation.
5. `logTicks` draw a log scale on your FACS plots.
6. `generationsDistance` calculate the distance between 2 generations of cells on the FACS scale.

**Examples**

```r
if(require(flowFitExampleData)){
  data(QuahAndParish)
  parent.fitting.cfse <- parentFitting(QuahAndParish[[1]], "<FITC-A>")
  fitting.cfse <- proliferationFitting(QuahAndParish[[2]], "<FITC-A>",
                                        parent.fitting.cfse@parentPeakPosition,
                                        parent.fitting.cfse@parentPeakSize)
  summary(fitting.cfse)
  confint(fitting.cfse)
  coef(fitting.cfse)
  Data(fitting.cfse)
  plot(parent.fitting.cfse)
  plot(fitting.cfse)

  # for this sample we use a Fixed Model: we keep fixed in the model the Parent Peak Position
  parent.fitting.cpd <- parentFitting(QuahAndParish[[1]], "<APC-A>")
  fitting.cpd <- proliferationFitting(QuahAndParish[[3]], "<APC-A>",
                                       parent.fitting.cpd@parentPeakPosition,
                                       parent.fitting.cpd@parentPeakSize,
                                       fixedModel=TRUE,
                                       fixedPars=list(M=parent.fitting.cpd@parentPeakPosition))
  parent.fitting.ctv <- parentFitting(QuahAndParish[[1]], "<Alexa Fluor 405-A>")
  fitting.ctv <- proliferationFitting(QuahAndParish[[4]], "<Alexa Fluor 405-A>",
                                        parent.fitting.ctv@parentPeakPosition,
                                        parent.fitting.ctv@parentPeakSize)

  # let's compare the generations across the 3 samples:
  par(mfrow=c(3,4))
  plot(parent.fitting.cfse, main="CFSE Non Stimulated")
  plot(fitting.cfse, which=3, main="CFSE")
  plot(fitting.cfse, which=4, main="CFSE")
  plot(fitting.cfse, which=5, main="CFSE")
}
```
plot(parent.fitting.cpd, main="CPD Non Stimulated")
plot(fitting.cpd, which=3, main="CPD")
plot(fitting.cpd, which=4, main="CPD")
plot(fitting.cpd, which=5, main="CPD")
plot(parent.fitting.ctv, main="CTV Non Stimulated")
plot(fitting.ctv, which=3, main="CTV")
plot(fitting.ctv, which=4, main="CTV")
plot(fitting.ctv, which=5, main="CTV")

# ESTIMATE GOODNESS of FITTING with KS TEST
perc.cfse <- fitting.cfse@generations
perc.cpd <- fitting.cpd@generations
perc.ctv <- fitting.ctv@generations
perc.cfse <- c(perc.cfse, rep(0,6))

# EXPLORATIVE PLOT
par(mfrow=c(1,1), ask=FALSE)
plot(perc.cfse, type="b", axes=FALSE, ylim=c(0,50),
     xlab="generations", ylab="Percentage of cells", main=""
)
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", text.col = "green4")
axis(2, at=seq(0,50,10), labels=paste(seq(0,50,10),"%"))
axis(1, at=1:16,labels=1:16)

# Pearson's Chi-squared Test for Count Data
M <- rbind(perc.cfse, perc.cpd, perc.ctv)
colnames(M) <- 1:16
(Xsq <- chisq.test(M, B=100000, simulate.p.value=TRUE))
text(8,40,paste("Chi-squared Test p=", round(Xsq$p.value, digits=4), sep=""))

# PKH26
# load data
data(PKH26data)
parent.fitting <- parentFitting(PKH26data[[1]], "FL2-Height LOG")
my.fit <- proliferationFitting(PKH26data[[2]], "FL2-Height LOG",
                               parent.fitting@parentPeakPosition,
                               parent.fitting@parentPeakSize)

my.fit
summary(my.fit)
confint(my.fit)
coef(my.fit)
Data(my.fit)

# plot results
plot(my.fit)

# modeling with locked Peak Size
my.fitb <- proliferationFitting(PKH26data[[2]], "FL2-Height LOG",
                                parent.fitting@parentPeakPosition,
                                parent.fitting@parentPeakSize,
                                fixedModel=TRUE,
                                fixedPars=list(S=16))
plot(my.fitb)

# modeling with locked Peak Size and Position
my.fitc <- proliferationFitting(PKH26data[[2]], "FL2-Height LOG",
                                parent.fitting@parentPeakPosition,
                                parent.fitting@parentPeakSize,
                                fixedModel=TRUE,
                                fixedPars=list(S=16))
generationsDistance

    parent.fitting@parentPeakPosition,
    parent.fitting@parentPeakSize,
    fixedModel=TRUE,
    fixedPars=list(S=16, M=810))

plot(my.fitc)
# modeling with locked Peak Size, Position and Distance
my.fitd <- proliferationFitting(PKH26data[[2]], "FL2-Height LOG",
    parent.fitting@parentPeakPosition,
    parent.fitting@parentPeakSize,
    fixedModel=TRUE,
    fixedPars=list(S=16, M=810, D=76))

plot(my.fitd)
}

---

### generationsDistance

**Calculate the distance between 2 generations of cells on the FACS scale**

**Description**

This function calculate the distance between 2 generations of cells on the FACS scale.

**Usage**

```r
generationsDistance(dataRange, logDecades)
```

**Arguments**

- `dataRange` Digital Data range on the FACS instrument
- `logDecades` Number of log decades on the FACS instrument (dynamic range)

**Details**

We can use this formula to convert FFI (FACS fluorescence Intensity) to RFI (Relative Fluorescence Intensity): 

\[
RFI = 10^{\frac{FFI}{c+l}}
\]

The inverse formula is used to convert from RFI to FACS fluorescence:

\[
FFI = c \cdot \log(RFI) \left(\frac{l}{l+\log(10)}\right)
\]

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.

**Value**

Return the spacing between generations on the FACS scale.
Author(s)
Davide Rambaldi

References

Examples

```r
distance <- generationsDistance(1024, 4)
```

---

getGenerations
*Get percentage of cells for generation in a flowFit model*

Description

getGenerations: get percentage of cells for generation in a flowFit model from an object of class proliferationFittingData generated by the proliferationFitting function.

Usage

```r
generateGenerations(object)
```

Arguments

- **object**: An object of class proliferationFittingData

Details

This function return a list. In order to get the percentage of cells for generation as vector you can use th slot generations of the proliferationFittingData (see also examples).

Value

return a list object

Author(s)

Davide Rambaldi

See Also

proliferationFitting and proliferationFittingData
logTicks

Examples

if(require(flowFitExampleData)){
  data(QuahAndParish)
  parent.fitting.cfse <- parentFitting(QuahAndParish[[1]], "<FITC-A>")
  fitting.cfse <- proliferationFitting(QuahAndParish[[2]], "<FITC-A>",
    parent.fitting.cfse@parentPeakPosition,
    parent.fitting.cfse@parentPeakSize)
  generationList <- getGenerations(fitting.cfse)
  # to extract a vector of percentage of cells for generation you can use:
  fitting.cfse@generations
}

logTicks(dataRange, logDecades, doScale = TRUE)

Arguments

dataRange  Range of your data (number of data points in the FACS)
logDecades Number of log decades in the FACS
doScaled   Scale according to dataRange and logDecades: Scale.factor = dataRange / logDecades

Value

Return a list with 3 elements:

major Position of the Major ticks
all   Position of the All ticks
label Labels for Major Ticks

Author(s)

Davide Rambaldi

Examples

if(require(flowFitExampleData)){
  # using flowViz
  # load data
data(PKH26data)
  # plot data
# parentFitting

Fitting a parent population

## Description

Estimate the proliferation of a cell population in cell-tracking dye studies. parentFitting: fit the parent population

## Usage

```r
parentFitting(flowframe, channel,  
  estimatedPeakPosition = NA,  
  estimatedPeakSize = NA,  
  dataRange = NA,  
  logDecades = NA,  
  binning = TRUE,  
  breaks = 1024,  
  dataSmooth = TRUE,  
  smoothWindow = 2,  
  fixedModel = FALSE,  
  fixedPars = NA,  
  verbose = FALSE )
```

## Arguments

- **flowframe**: An object of class `flowFrame` from `flowCore`
- **channel**: FACS column/channel (`flowFrame` column)
- **estimatedPeakPosition**: Estimated peak position. If not provided will be used the `exprs` mean
- **estimatedPeakSize**: Estimated peak size. If not provided will be used the `exprs` standard deviation
- **dataRange**: Number of digital data points on the machine. If not provided will be extracted from `flowFrame` using `keyword`
- **logDecades**: FACS dynamic range (log decades). If not provided will be extracted from `flowFrame` using `keyword`
- **binning**: Should I bin data? Some FACS have a large data range (Es: FACSCanto have 65536 data points, may be is convenient in this case to group data in bins to avoid acquiring too many cells). If you have you data log tranformed in range 0-5 it is mandatory to bin data
parentFitting

breaks How many breaks if I bin data?
dataSmooth Should I smooth data with a Kolmogorov-Zurbenko low-pass linear filter?
smoothWindow Window used to smooth data with the Kolmogorov-Zurbenko low-pass linear filter.
fixedModel Should I use a model with fixed parameters? (Peak Position or Size).
fixedPars A list of fixed parameters. If you give me a value, I use that value, otherwise I use estimates (check examples)
verbose Verbose mode.

Details
The formula used to fit the parent population:

\[ a^2 \exp \left( \frac{(x - \mu)^2}{2\sigma^2} \right) \]

The algorithm estimate the position (\(\mu\)) and size (\(\sigma\)) of the Parent Population.

Value
return a parentFittingData object

Author(s)
Davide Rambaldi

References
1. Timur V. Elzhov, Katharine M. Mullen and Ben Bolker (2012). minpack.lm: R interface to the Levenberg-Marquardt nonlinear least-squares algorithm found in MINPACK.

See Also
proliferationFitting

Examples

```r
if(require(flowFitExampleData)){
  # CFSE
  data(QuahAndParish)
parent.fitting.cfse <- parentFitting(QuahAndParish[[1]], "<FITC-A>")
parent.fitting.cfse
summary(parent.fitting.cfse)
confint(parent.fitting.cfse)
coef(parent.fitting.cfse)
plot(parent.fitting.cfse)
Data(parent.fitting.cfse)

  # PKH26
  data(PKH26data)
parent.fitting <- parentFitting(PKH26data[[1]], "FL2-Height LOG")
parent.fitting
```
```
summary(parent.fitting)
confint(parent.fitting)
coef(parent.fitting)
plot(parent.fitting)
Data(parent.fitting)

# fixedModel with estimates
parent.fitting <- parentFitting(PKH26data[[1]], "FL2-Height LOG",
                               fixedModel=TRUE, fixedPars=list(M=NA, S=NA))

# fixedModel with user values
parent.fitting <- parentFitting(PKH26data[[1]], "FL2-Height LOG",
                               fixedModel=TRUE, fixedPars=list(M=810, S=16))

# fixedModel with locked Peak Size (one fixed parameter)
parent.fitting <- parentFitting(PKH26data[[1]], "FL2-Height LOG",
                               fixedModel=TRUE, fixedPars=list(S=17))

```
dataSmooth: Object of class "logical" ~
smoothWindow: Object of class "numeric" ~
parStart: Object of class "list" ~
dataMatrix: Object of class "matrix" ~
dataPoints: Object of class "data.frame" ~
modelPoints: Object of class "data.frame" ~
residFun: Object of class "function" ~

Methods

**plot**  Basic plots for parentFittingData objects. *Usage: plot(parentFittingData, main="Original data and Parent fitting", xlab="FACS data range", ylab="# of Events", showLegend=TRUE, logScale = TRUE, drawGrid = TRUE, ...*)

**show**  Display details about the parentFittingData object.

**summary**  Return a descriptive summary about the parentFittingData object.

**Data**  Return the flowFrame object.

**coef**  Return coefficients of the model.

**confint**  Return confidence intervals of the model.

Author(s)

Davide Rambaldi

See Also

parentFitting

Examples

```
showClass("parentFittingData")
if(require(flowFitExampleData)){
  data(PKH26data)
  parent.fitting <- parentFitting(PKH26data[[1]], "FL2-Height LOG")
  parent.fitting
  summary(parent.fitting)
  confint(parent.fitting)
  coef(parent.fitting)
  plot(parent.fitting)
  Data(parent.fitting)
}
```
Very basic plotting of flowFit objects: proliferationFittingData and parentFittingData

Description
A basic method to plot proliferationFittingData and parentFittingData objects. See below for details.

Details
Basic plots for flowFit objects.

Supported arguments for parentFittingData:
1. main: an overall title for the plot, see title.
2. xlab: a title for the x axis, see title.
3. ylab: a title for the y axis, see title.
4. legend: show/hide messages.
5. logScale: put a log scale on the x axis.

Supported arguments for proliferationFittingData:
1. which: which plots I should show? ["all" or 1:5]
2. main: an overall title prefix for the plots.
3. xlab: a title for the x axis, see title.
4. ylab: a title for the y axis, see title.
5. legend: show/hide messages.
6. logScale: put a log scale on the x axis.
7. drawGrid: put some dashed lines at the generationsDistance expected positions.

Methods
x = "proliferationFittingData", y = "missing"  Multiple plots of generations fitting data (data generated by the function proliferationFitting)

x = "parentFittingData", y = "missing" ) Single plot of a parent population fitting (data generated by the function parentFitting)

Author(s)
Davide Rambaldi

See Also
proliferationFitting, parentFitting
Estimate proliferation in cell-tracking dye studies

Description

The algorithm fit a set of N peaks on the flowframe data using the nls.lm function. The number of peaks to be fitted is automatically estimated using generationsDistance.

The algorithm take the position ($\mu$) and size ($\sigma$) of the Parent Population as estimates and fit a set of peaks on a flowFrame data.

The first peak correspond to the parent population:

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

The next peak (corresponding to the next generation of cells) will be:

$$b^2 \exp \left( -\frac{(x - (\mu - D))^2}{2\sigma^2} \right)$$

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

The complete formula for the fitting of the 15 peaks is the following:

$$a^2 \exp \left( -\frac{(x - M)^2}{2s^2} \right) + b^2 \exp \left( -\frac{(x - (M - D))^2}{2s^2} \right) + ... + p^2 \exp \left( -\frac{(x - (M - 14 \cdot D))^2}{2s^2} \right)$$

Where the parameters [a-q] represent an estimate of the number of cells for a given generation.

In the Levenberg-Marquadt algorithm implementation we use this formula to estimate the error between the model and the real data:

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

The ration between the intergral of a single peak and the integral of all model formula is an estimate of the percentage of cells in a given generation.

Usage

proliferationFitting(flowframe, channel, estimatedParentPosition, estimatedParentSize, dataRange = NA, logDecades = NA, estimatedDistance = NA, binning = TRUE, breaks = 1024, dataSmooth = TRUE, smoothWindow = 2, fixedModel = FALSE, fixedPars = NA, verbose = FALSE)
Arguments

- **flowframe**: An object of class `flowFrame` from `flowCore`
- **channel**: FACS column/channel (`flowFrame` column)
- **estimatedParentPosition**: Estimated parent peak position.
- **estimatedParentSize**: Estimated parent peak size.
- **dataRange**: Number of digital data points on the machine. If not provided will be extracted from `flowFrame` using `keyword`.
- **logDecades**: FACS dynamic range (log decades). If not provided will be extracted from `flowFrame` using `keyword`.
- **estimatedDistance**: Estimated distance between generations. If not provided will be estimated with `generationsDistance`.
- **binning**: Should I bin data? Some FACS have a large data range (Es: FACSCanto have 65536 data points, may be is convenient in this case to group data in bins to avoid acquiring too many cells). If you have you data log tranformed in range 0-5 it is mandatory to bin data.
- **breaks**: How many breaks if I bin data?
- **dataSmooth**: Should I smooth data with a Kolmogorov-Zurbenko low-pass linear filter (`kz`)?
- **smoothWindow**: Window used to smooth data with the Kolmogorov-Zurbenko low-pass linear filter (`kz`).
- **fixedModel**: Should I use a model with fixed parameters? (Peak Position or Size).
- **fixedPars**: A list of fixed parameters. If you give me a value, I use that value, otherwise I use estimates (check examples).
- **verbose**: Verbose mode.

Details

See the vignette for more details on this function.

Value

- return a `proliferationFittingData` object

Author(s)

Davide Rambaldi

References

1. Timur V. Elzhov, Katharine M. Mullen and Ben Bolker (2012). *minpack.lm: R interface to the Levenberg-Marquardt nonlinear least-squares algorithm found in MINPACK.*
Examples

if(require(flowFitExampleData)){
  # PKH26
  data(PKH26data)
  parent.fitting <- parentFitting(PKH26data[[1]], "FL2-Height LOG")
  my.fit <- proliferationFitting(PKH26data[[2]], "FL2-Height LOG",
                                  parent.fitting@parentPeakPosition,
                                  parent.fitting@parentPeakSize)

  my.fit
  summary(my.fit)
  confint(my.fit)
  coef(my.fit)
  Data(my.fit)
  # plot results
  plot(my.fit)

  # modeling with locked Peak Size
  my.fit <- proliferationFitting(PKH26data[[2]], "FL2-Height LOG",
                                  parent.fitting@parentPeakPosition,
                                  parent.fitting@parentPeakSize,
                                  fixedModel=TRUE,
                                  fixedPars=list(S=16))

  # modeling with locked Peak Size and Position
  my.fit <- proliferationFitting(PKH26data[[2]], "FL2-Height LOG",
                                  parent.fitting@parentPeakPosition,
                                  parent.fitting@parentPeakSize,
                                  fixedModel=TRUE,
                                  fixedPars=list(S=16, M=810))

  # modeling with locked Peak Size, Position and Distance
  my.fit <- proliferationFitting(PKH26data[[2]], "FL2-Height LOG",
                                  parent.fitting@parentPeakPosition,
                                  parent.fitting@parentPeakSize,
                                  fixedModel=TRUE, fixedPars=list(S=16, M=810, D=76))

  # generations as vector
  my.fit@generations
  # generations as list
  getGenerations(my.fit)

  # CFSE, CPD and CTV data
  data(QuahAndParish)
  parent.fitting.cfse <- parentFitting(QuahAndParish[[1]], "<FITC-A>")
  fitting.cfse <- proliferationFitting(QuahAndParish[[2]], "<FITC-A>",
                                      parent.fitting.cfse@parentPeakPosition,
                                      parent.fitting.cfse@parentPeakSize)

  summary(fitting.cfse)
  confint(fitting.cfse)
  coef(fitting.cfse)
  Data(fitting.cfse)

  plot(parent.fitting.cfse)
  plot(fitting.cfse)
# for CPD samples we use a Fixed Model: we keep fixed in the model the Parent Peak Position
parent.fitting.cpd <- parentFitting(QuahAndParish[[1]], "<APC-A>")
fitting.cpd <- proliferationFitting(QuahAndParish[[3]], "<APC-A>",
parent.fitting.cpd@parentPeakPosition,
parent.fitting.cpd@parentPeakSize,
fixedModel=TRUE,
fixedPars=list(M=parent.fitting.cpd@parentPeakPosition))

parent.fitting.ctv <- parentFitting(QuahAndParish[[1]], "<Alexa Fluor 405-A>")
fitting.ctv <- proliferationFitting(QuahAndParish[[4]], "<Alexa Fluor 405-A>",
parent.fitting.ctv@parentPeakPosition,
parent.fitting.ctv@parentPeakSize)

# let's compare the generations across the 3 samples:
plot(parent.fitting.cfse, main="CFSE Non Stimulated")
plot(fitting.cfse, which=3, main="CFSE")
plot(fitting.cfse, which=4, main="CFSE")
plot(fitting.cfse, which=5, main="CFSE")
plot(parent.fitting.cpd, main="CPD Non Stimulated")
plot(fitting.cpd, which=3, main="CPD")
plot(fitting.cpd, which=4, main="CPD")
plot(fitting.cpd, which=5, main="CPD")
plot(parent.fitting.ctv, main="CTV Non Stimulated")
plot(fitting.ctv, which=3, main="CTV")
plot(fitting.ctv, which=4, main="CTV")
plot(fitting.ctv, which=5, main="CTV")

proliferationFittingData-class

Class "proliferationFittingData"

Description

Provides S4 data structure and basic infrastructure and functions to store proliferation tracking data of the Parent Population.

Objects from the Class

Objects can be created by calls of the form new("proliferationFittingData", flowframe, channel, ...). This class is for internal use.

Slots

data: Object of class "FlowFrame" ~~
channel: Object of class "character" ~~
estimatedPeakPosition: Object of class "numeric" ~~
estimatedPeakSize: Object of class "numeric" ~~
dataRange: Object of class "numeric" ~~
logDecades: Object of class "numeric" ~~
estimatedDistance: Object of class "numeric" ~~
Methods

plot Basic plots for proliferationFittingData objects. Usage: plot(proliferationFittingData, main="Original data and Parent fitting", xlab="FACS data range", ylab="# of Events", showLegend=TRUE, logScale = TRUE, drawGrid = TRUE, ...)

show Display details about the proliferationFittingData object.

summary Return a descriptive summary about the proliferationFittingData object.

Data Return the flowFrame object.

coef Return coefficients of the model.

confint Return confidence intervals of the model.

Author(s)

Davide Rambaldi

See Also

proliferationFitting

Examples

showClass("proliferationFittingData")
if(require(flowFitExampleData)){
data(PKH26data)
parent.fitting <- parentFitting(PKH26data[[1]], "FL2-Height LOG")
my.fit <- proliferationFitting(PKH26data[[2]], "FL2-Height LOG",
parent.fitting@parentPeakPosition,
proliferationGrid

my.fit
summary(my.fit)
confint(my.fit)
coef(my.fit)
plot(my.fit)
Data(my.fit)
}

proliferationGrid  
proliferationGrid function for plotting

Description
This function draws a proliferation grid. The grid marks the distance between cell generations calculated with the function `generationsDistance`.

Usage
```r
proliferationGrid(parentPosition, 
  fittedDistance = NA, dataRange = 1024, logDecades = 4, 
  lwd=1, lty=3, col=rgb(0,0,0.5))
```

Arguments
- `parentPosition`: Position of the parent Peak from `parentFitting`
- `fittedDistance`: You can provide the distance estimated from the `proliferationFitting` function
- `dataRange`: Range of your data (number of data points in the FACS)
- `logDecades`: Number of log decades in the FACS
- `lwd`: Grid line size. See `par` 
- `lty`: Grid line type. See `par` 
- `col`: Grid color. See `par` and `rgb`

Author(s)
Davide Rambaldi

Examples
```r
plot(c(0,1023),c(0,1000),
  xlim=c(0,1023),
  ylim=c(0,1000),
  xlab="FACS CHANNEL",
  ylab="# OF EVENTS",
  main="A flowFit Empty Plot")

# create a grid with parent at 800
proliferationGrid(1000, dataRange=1024, logDecades=4)
```
proliferationIndex

proliferation index calculator

Description

Proliferation index calculator. 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. It is a measure of the fold increase in cell number in the culture over the course of the experiment.

Usage

proliferationIndex(object)

Arguments

- object: An object of class proliferationFittingData

Details

The formula is: \( \sum_{i=0}^{\infty} \frac{N_i}{N_0^{i/2}} \). 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: \( \frac{N_0}{N_0^{0/2}} = 1 \) defining the lower limit of the PI.

Value

return a numeric

Author(s)

Davide Rambaldi

References


See Also

proliferationFitting proliferationFittingData-class

Examples

```r
# load data
if(require(flowFitExampleData)){
  data(PKH26data)
  parent.fitting <- parentFitting(PKH26data[[1]], "FL2-Height LOG")
  my.fit <- proliferationFitting(PKH26data[[2]], "FL2-Height LOG",
                                   parent.fitting@parentPeakPosition,
                                   parent.fitting@parentPeakSize)
  my.index <- proliferationIndex(my.fit)
}
```
Index

*Topic classes
  - parentFittingData-class, 10
  - proliferationFittingData-class, 16

*Topic methods
  - plot-methods, 12

*Topic package
  - flowFit-package, 2

coef, 11, 17
  - coef, parentFittingData-method (parentFittingData-class), 10
  - coef, proliferationFittingData-method (proliferationFittingData-class), 16

confint, 11, 17
  - confint, parentFittingData-method (parentFittingData-class), 10
  - confint, proliferationFittingData-method (proliferationFittingData-class), 16

Data, 11, 17
  - Data, parentFittingData-method (parentFittingData-class), 10
  - Data, proliferationFittingData-method (proliferationFittingData-class), 16

exprs, 8

flowCore, 2, 8, 14
  - flowFit (flowFit-package), 2
  - flowFit-package, 2
  - flowFrame, 8, 11, 14, 17
  - flowViz, 2

generationsDistance, 3, 5, 12–14, 18
  - generationsDistance, ANY-method (generationsDistance), 5
  - generationsDistance-methods (generationsDistance), 5
  - getGenerations, 3, 6
  - getGenerations, proliferationFittingData-method (getGenerations), 6

getGenerations-methods (getGenerations), 6
  - initialize, parentFittingData-method (parentFittingData-class), 10
  - initialize, proliferationFittingData-method (proliferationFittingData-class), 16

keyword, 8, 14
  - kz, 14

logTicks, 3, 7
  - logTicks, ANY-method (logTicks), 7
  - logTicks-methods (logTicks), 7

nls.lm, 2, 13

par, 18

parentFitting, 3, 8, 11, 12, 18
  - parentFitting, flowFrame-method (parentFitting), 8
  - parentFitting-methods (parentFitting), 8
  - parentFittingData, 2, 9, 12
  - parentFittingData (parentFittingData-class), 10
  - parentFittingData-class, 10

plot, 11, 17
  - plot (plot-methods), 12
  - plot, parentFittingData, ANY-method (plot-methods), 12
  - plot, parentFittingData, character-method (plot-methods), 12
  - plot, parentFittingData, missing-method (plot-methods), 12
  - plot, proliferationFittingData, ANY-method (plot-methods), 12
  - plot, proliferationFittingData, character-method (plot-methods), 12
  - plot, proliferationFittingData, missing-method (plot-methods), 12

plot-methods, 12
  - proliferationFitting, 3, 6, 9, 12, 13, 17–19
    - proliferationFitting, flowFrame-method (proliferationFitting), 13
proliferationFitting-methods
  (proliferationFitting), 13
proliferationFittingData, 2, 6, 12, 14, 19
proliferationFittingData
  (proliferationFittingData-class), 16
proliferationFittingData-class, 16
proliferationGrid, 18
proliferationGrid, ANY-method
  (proliferationGrid), 18
proliferationGrid-methods
  (proliferationGrid), 18
proliferationIndex, 3, 19
proliferationIndex, proliferationFittingData-method
  (proliferationIndex), 19
proliferationIndex-methods
  (proliferationIndex), 19
rgb, 18

show, 11, 17
show, parentFittingData-method
  (parentFittingData-class), 10
show, proliferationFittingData-method
  (proliferationFittingData-class), 16
summary, 11, 17
summary, parentFittingData-method
  (parentFittingData-class), 10
summary, proliferationFittingData-method
  (proliferationFittingData-class), 16

title, 12