Package 'flowPloidy'

January 18, 2021

Title Analyze flow cytometer data to determine sample ploidy

Version 1.16.0

Author Tyler Smith <tyler@plantarum.ca>

Maintainer Tyler Smith <tyler@plantarum.ca>

Author@R person(given = ``Tyler'', middle = ``William'', family = ``Smith'', email = ``tyler@plantarum.ca'', role = c(``cre'', ``aut''), comment = c(ORCID = ``0000-0001-7683-2653''))

Description Determine sample ploidy via flow cytometry histogram analysis. Reads Flow Cytometry Standard (FCS) files via the flowCore bioconductor package, and provides functions for determining the DNA ploidy of samples based on internal standards.

biocViews FlowCytometry, GUI, Regression, Visualization

URL https://github.com/plantarum/flowPloidy

BugReports https://github.com/plantarum/flowPloidy/issues

License GPL-3

LazyData true

Imports flowCore, car, caTools, knitr, rmarkdown, minpack.lm, shiny, methods, graphics, stats, utils

Suggests flowPloidyData, testthat

RoxygenNote 6.1.1

VignetteBuilder knitr

git_url https://git.bioconductor.org/packages/flowPloidy

git_branch RELEASE_3_12

git_last_commit ccf3b7c

git_last_commit_date 2020-10-27

Date/Publication 2021-01-17

R topics documented:

`browseFlowHist` *browseFlowHist*

Description

Visually assess and correct histogram fits

Usage

```
browseFlowHist(flowList, debug = FALSE)
```

Arguments

`flowList`	either a `FlowHist` object, or a list of `FlowHist` objects
`debug`	boolean, turns on debugging messages

Details

Visually assess histogram fits, correcting initial values, and selecting model components.

This function will open a browser tab displaying the first `FlowHist` object from the argument `flowList`. Using the interface, the user can modify the starting values for the histogram peaks, select different debris model components, toggle the linearity option, select which peak to treat as the standard, and, if multiple standard sizes are available, select which one to apply.

See the "Getting Started" vignette for a tutorial introduction.

Value

Returns the list of `FlowHist` objects, updated by any changes made in the GUI.

Author(s)

Tyler Smith

Examples

```
library(flowPloidyData)
batch1 <- batchFlowHist(flowPloidyFiles(), channel = "FL3.INT.LIN")
## Not run:
batch1 <- browseFlowHist(batch1)

## End(Not run)
```

`fhAccessors` *FlowHist Accessors*

Description

Functions to access slot values in `FlowHist` objects

Usage

```
fhGate(fh)

fhLimits(fh)

fhSamples(fh)

fhPeaks(fh)

fhInit(fh)

fhComps(fh)

fhModel(fh)

fhSpecialParams(fh)

fhArgs(fh)

fhNLS(fh)

fhCounts(fh)

fhCV(fh)

fhRCS(fh)

fhFile(fh)

fhChannel(fh)

fhBins(fh)

fhLinearity(fh)
```

```
fhDebris(fh)

fhHistData(fh)

fhRaw(fh)

fhStandards(fh)

fhStdPeak(fh)

fhStdSelected(fh)

fhStdSizes(fh)

fhOpts(fh)

fhG2(fh)

fhAnnotation(fh)

fhFail(fh)
```

Arguments

fh	a FlowHist

Details

For normal users, these functions aren't necessary. Overly curious users, or those wishing to hack on the code, may find these useful for inspecting the various bits and pieces inside a FlowHist object.

The versions of these functions that allow modification of the FlowHist object are not exported. Functions are provided for users to update FlowHist objects in a safe way.

Value

Used to access a slot, returns the value of the slot. Used to update the value of a slot, returns the updated FlowHist object.

Author(s)

Tyler Smith

Examples

```
library(flowPloidyData)
fh1 <- FlowHist(file = flowPloidyFiles()[1], channel = "FL3.INT.LIN")
fhModel(fh1) ## prints the model to screen
```

`fhAnalyze` *fhAnalyze*

Description

Complete non-linear regression analysis of FlowHist histogram data

Usage

```
fhAnalyze(fh)
```

Arguments

`fh`	a `FlowHist` object

Details

Completes the NLS analysis, and calculates the modelled events and CVs for the result.

Value

a `FlowHist` object with the analysis (nls, counts, cv, RCS) slots filled.

Author(s)

Tyler Smith

See Also

`FlowHist`

Examples

```
library(flowPloidyData)
fh1 <- FlowHist(file = flowPloidyFiles()[1], channel = "FL3.INT.LIN")
fh1 <- fhAnalyze(fh1)
```

`fhModels` *Building Flow Histogram Models*

Description

Functions for assembling non-linear regression models for `FlowHist` objects.

Usage

```
addComponents(fh)

dropComponents(fh, components)

setLimits(fh)

makeModel(fh, env = parent.frame())
```

Arguments

fh	a FlowHist object
components	character, a vector of ModelComponent names.
env	an R environment. Don't change this, it's R magic to keep the appropriate environment in scope when building our model.

Details

addComponents examines the model components in fhComponents and includes the ones that pass their includeTest.

dropComponents removes a component from the FlowHist model

setLimits collates the parameter limits for the model components included in a FlowHist object. (could be called automatically from addComponents, as it already is from dropComponents?)

makeModel creates a model out of all the included components.

Value

The updated FlowHist object.

Author(s)

Tyler Smith

FlowHist *FlowHist*

Description

Creates a FlowHist object from an FCS file, setting up the histogram data for analysis.

Usage

```
FlowHist(file, channel, bins = 256, analyze = TRUE,
  linearity = "variable", debris = "SC", samples = 2, pick = FALSE,
  standards = 0, g2 = TRUE, debrisLimit = 40, ...)

batchFlowHist(files, channel, verbose = TRUE, ...)
```

Arguments

file	character, the name of the single file to load
channel	character, the name of the data column to use
bins	integer, the number of bins to use to aggregate events into a histogram
analyze	logical, if TRUE the model will be analyzed immediately
linearity	character, either "variable", the default, or "fixed". If "fixed", linearity is fixed at 2; if "variable", linearity is fit as a model parameter.
debris	character, either "SC", the default, "MC", or "none", to set the debris model component to the Single-Cut or Multi-Cut models, or to not include a debris component (such as for gated data).

samples	integer; the number of samples in the data. Default is 2 (unknown and standard), but can be set to 3 if two standards are used, or up to 6 for endopolyploidy analysis.
pick	logical; if TRUE, the user will be prompted to select peaks to use for starting values. Otherwise (the default), starting values will be detected automatically.
standards	numeric; the size of the internal standard in pg. When loading a data set where different samples have different standards, a vector of all the standard sizes. If set to 0, calculation of pg for the unknown sample will not be done.
g2	a logical value, default is TRUE. Should G2 peaks be included in the model?
debrisLimit	an integer value, default is 40. Passed to `cleanPeaks`. Peaks with fluorescence values less than `debrisLimit` will be ignored by the automatic peak-finding algorithm. Used to ignore the debris often found at the left side of the histogram.
...	additional arguments passed from `batchFlowHist` to `FlowHist`, or to assorted helper functions. See `findPeaks` (arguments `window` and `smooth`)
files	character, a vector of file names to load, or a single character value giving the path to a directory; if the latter, all files in the directory will be loaded
verbose	logical; if TRUE, `batchFlowHist` will list files as it processes them.

Details

For most uses, simpling calling `FlowHist` with a `file`, `channel`, and `standards` argument will do what you need. The other arguments are provided for optional tuning of this process. In practice, it's easier to correct the model fit using `browseFlowHist` than to determine 'perfect' values to pass in as arguments to `FlowHist`.

Similarly, `batchFlowHist` is usually used with only the `files`, `channel`, and `standards` arguments.

In operation, `FlowHist` starts by reading an FCS file (using the function `read.FCS` internally). This produces a `flowFrame` object, which we extend to a `FlowHist` object as follows:

1. Extract the fluorescence data from `channel`.
2. Remove the top bin, which contains off-scale readings we ignore in the analysis.
3. Remove negative fluorescence values, which are artifacts of instrument compensation
4. Removes the first 5 bins, which often contain noisy values, probably further artifacts of compensation.
5. aggregates the raw data into the desired number of bins, as specified with the `bins` argument. The default is 256, but you may also try 128 or 512. Any integer is technically acceptable, but I wouldn't stray from the default without a good reason. (I've never had a good reason!)
6. identify model components to include. All `FlowHist` objects will have the single-cut debris model and the G1 peak for sample A, and the broadened rectangle for the S-phase of sample A. Depending on the data, additional components for the G2 peak and sample B (G1, G2, s-phase) may also be added. The `debris` argument can be used to select the Multi-Cut debris model instead, or this can be toggled in `browseFlowHist`
7. Build the NLS model. All the components are combined into a single model.
8. Identify starting values for Gaussian (G1 and G2 peaks) model components. For reasonably clean data, the built-in peak detection is ok. You can evaluate this by plotting the `FlowHist` object with the argument `init = TRUE`. The easiest way to fix bad peak detection is via the `browseFlowHist` interface. You can also play with the `window` and `smooth` arguments (which is tedious!), or pick the peaks visually yourself with `pick = TRUE`.
9. Finally, we fit the model and calculate the fitted parameters. Model fitting is suppressed if the `analyze` argument is set as `FALSE`

Value

FlowHist returns a FlowHist object.

batchFlowHist returns a list of FlowHist objects.

Slots

raw a flowFrame object containing the raw data from the FCS file

channel character, the name of the data column to use

bins integer, the number of bins to use to aggregate events into a histogram

linearity character, either "fixed" or "variable" to indicate if linearity is fixed at 2 or fit as a model parameter

debris character, either "SC" or "MC" to indicate if the model should include the single-cut or multi-cut model

gate logical, a vector indicating events to exclude from the analysis. In normal use, the gate will be modified via interactive functions, not set directly by users.

histdata data.frame, the columns are the histogram bin number (xx), florescence intensity (intensity), and the raw single-cut and multi-cut debris model values (SCvals and MCvals), and the raw doublet, triplet and quadruplet aggregate values (DBvals, TRvals, and QDvals). The debris and aggregate values are used in the NLS fitting procedures.

peaks matrix, containing the coordinates used for peaks when calculcating initial parameter values.

opts list, currently unused. A convenient place to store flags when trying out new options.

comps a list of ModelComponent objects included for these data.

model the function (built from comps) to fit to these data.

limits list, a list of lower and upper bounds for model parameters

init a list of initial parameter estimates to use in fitting the model.

nls the nls object produced by the model fitting

counts a list of cells counted in each peak of the fitted model

CV a list of the coefficients of variation for each peak in the fitted model.

RCS numeric, the residual chi-square for the fitted model.

samples numeric, the number of samples included in the data. The default is 2 (i.e., unknown and standard), but if two standards are used it should be set to 3. It can be up to 6 for endopolyploidy analysis, and can be interactively increased (or decreased) via browseFlowHist

standards a FlowStandards object.

g2 logical, if TRUE the model will include G2 peaks for each sample (as long as the G1 peak is less than half-way across the histogram). Set to FALSE to drop the G2 peaks for endopolyploidy analyses.

annotation character, user-added annotation for the sample.

fail logical, set by the user via the browseFlowHist interface to indicate the sample failed and no model fitting should be done.

Author(s)

Tyler Smith

Examples

```
library(flowPloidyData)
fh1 <- FlowHist(file = flowPloidyFiles()[1], channel = "FL3.INT.LIN")
fh1
batch1 <- batchFlowHist(flowPloidyFiles(), channel = "FL3.INT.LIN")
batch1
```

---

flowPloidy — *flowPloidy: A package for analyzing Flow Cytometry Histograms*

Description

The flowPloidy package provides functions for reading and analyzing flow cytometry histograms. Specifically, it builds and fits a non-linear regression model, from which peak parameters (mean, CV) can be estimated. In normal use, samples will include a co-chopped size standard. Comparing the unknown peak mean to the standard peak mean, we determine the genome content for the unknown sample.

Details

Please see the vignettes for an overview: `histogram-tour` and `flowPloidy-gettingStarted`. To follow along with the examples in the vignettes, and also in the documentation listed below, you'll need to install the `flowPloidyData` package from Bioconductor.

Primary Functions

Most users will need only the functions:

1. `viewFlowChannels`, to determine the name of the primary data channel to use.
2. `batchFlowHist`, to load a list of FCM files into R.
3. `browseFlowHist`, to review and correct the model-fitting for the files, using an interactive graphical browser.
4. `tabulateFlowHist`, to extract the results and save them to a file.

Additional User Tools

Additional functions for inspecting and manipulating `FlowHist` objects and analyses:

1. `FlowHist`, to load a single FCM file into R.
2. `plot.FlowHist`, for plotting the data and fitted model using base R graphics.
3. `pickInit`, to interactively select initial peak estimates, using base R graphics (this is more easily accomplished via `browseFlowHist`.
4. `setBins`, to reset the bins, selecting the number of bins to use.
5. `fhAnalyze`, to (re-)analyze the FCM data, presumably after updating the settings for a file. Most functions that make changes that would require reanalysis provide the option to do this automatically, and this option is usually the default.
6. `updateFlowHist`, to update the settings for an FCM file.

Internal Functions

These functions aren't necessary for regular use, and are not exported for direct access by users. They may be useful to those interested in modifying or extending the package, or just curious about details:

1. `fhAccessors`, for inspecting the slots of a `FlowHist` object
2. `findPeaks`, the functions which perform the initial peak detection
3. `ModelComponent`, the S4 class for the various model components used in constructing the non-linear regression model.
4. `GaussianComponents`, a description of the Gaussian model component that is fit to cell peaks.
5. `DebrisModels`, a description of the debris model components.
6. `FlowStandards`, the S4 class for the size standard data.
7. `plotFH`, a low-level plotting function for displaying raw histogram data.
8. `resetFlowHist`, a function for safely resetting various portions of a `FlowHist` object.
9. `flowModels`, functions for assembling `ModelComponent` into a complete model.
10. `fhDoNLS`, `fhDoCounts`, `fhDoCV`, `fhDoRCS`: the functions which actually complete the model fitting and extract the parameters of interest.
11. `setGate`, the function for applying a gate to a `FlowHist` object.

Author(s)

Tyler Smith

`FlowStandards` *An S4 class to represent internal standard details for `FlowHist` objects*

Description

The `sizes` slot is set in `FlowHist` or `batchFlowHist`. The other values are updates through interaction with the `browseFlowHist` GUI.

Usage

```
stdSizes(std)

stdSelected(std)

stdPeak(std)
```

Arguments

`std`	a `FlowStandards` object

Value

`stdSizes`, `stdSelected` and `stdPeak` return the corresponding slot values

Slots

`sizes` numeric, the size (in pg) of the internal size standard. Can be a vector of multiple values, if the sample is part of a set that included different standards for different samples.

`selected` numeric, the size (in pg) of the internal size standard actually used for this sample. Must be one of the values in the `sizes` slot.

`peak` character, "A" or "B", indicating which of the histogram peaks is the size standard.

Examples

```
library(flowPloidyData)
fh1 <- FlowHist(file = flowPloidyFiles()[1], channel = "FL3.INT.LIN",
                standards = c(1.96, 5.43))
fhStandards(fh1)  ## display standards included in this object
stdSizes(fhStandards(fh1))  ## list standard sizes
```

---

gauss — *Gaussian model components*

Description

Components for modeling Gaussian features in flow histograms

Arguments

`a1, a2, b1, b2, c1, c2`	area parameters
`Ma, Mb, Mc`	curve mean parameter
`Sa, Sb, Sc`	curve standard deviation parameter
`xx`	vector of histogram intensities
`linearity`	numeric, the ratio of G2/G1 peak means. When linearity is fixed, this is set to 2. Otherwise, it is fit as a model parameter bounded between flowPloidy:::linL and flowPloidy:::linH.

Details

Typically the complete models will contain fA1 and fB1, which model the G1 peaks of the sample and the standard. In many cases, they will also contain fA2 and fB2, which model the G2 peaks. The G2 peaks are linked to the G1 peaks, in that they require some of the parameters from the G1 peaks as well (mean and standard deviation).

If the linearity parameter is set to "fixed", the G2 peaks will be fit as exactly 2 times the mean of the G1 peaks. If linearity is set to "variable", the ratio of the G2 peaks to the G1 peaks will be fit as a model parameter with an initial value of 2, and constrained to the range 1.5 – 2.5. (The range is coded as linL and linH. If in doubt, check the values of those, i.e., flowPloidy:::linL, flowPloidy:::linH, to be sure Tyler hasn't changed the range without updating this documentation!!)

Additionally, for each set of peaks (sample and standard(s)), a broadened rectangle component is included to model the S-phase. At present, this is component has a single parameter, the height of the rectangle. The standard deviation is fixed at 1. Allowing the SD to vary in the model fitting doesn't make an appreciable difference in my tests so far, so I've left it simple.

Value

NA

Author(s)

Tyler Smith

`ModelComponent` *An S4 class to represent model components*

Description

`ModelComponent` objects bundle the actual mathematical function for a particular component with various associated data necesarry to incorporate them into a complete NLS model.

Details

To be included in the automatic processing of potential model components, a `ModelComponent` needs to be added to the variable `fhComponents`.

Slots

`name` character, a convenient name with which to refer to the component

`desc` character, a short description of the component, for human readers

`color` character, the color to use when plotting the component

`includeTest` function, a function which takes a single argument, a `FlowHist` object, and returns `TRUE` if the component should be included in the model for that object.

`function` function, a single-line function that returns the value of the component. The function can take multiple arguments, which usually will include `xx`, the bin number (i.e., `x` value) of the histogram. The other arguments are model parameters, and should be included in the `initParams` function.

`initParams` function, a function with a single argument, a `FlowHist` object, which returns named list of model parameters and their initial estimates.

`specialParams` list, a named list. The names are variables to exclude from the default argument list, as they aren't parameters to fit in the NLS procedure, but are actually fixed values. The body of the list element is the object to insert into the model formula to account for that variable. Note that this slot is not set directly, but should be provided by the value returned by `specialParamSetter` (which by default is `list(xx = substitute(xx))`).

`specialParamSetter` function, a function with one argument, the `FlowHist` object, used to set the value of `specialParams`. This allows parameters to be declared 'special' based on values in the `FlowHist` object. The default value for this slot is a function which returns `list(xx = substitute(xx))`

`paramLimits` list, a named list with the upper and lower limits of each parameter in the function.

`doCounts` logical, should cell counts be evaluated for this component? Used to exclude the debris models, which don't work with R's Integrate function.

Coding Concepts

See the source code file `models.R` for the actual code used in defining model components. Here are a few examples to illustrate different concepts.

We'll start with the G1 peaks. They are modelled by the components `fA1` and `fB1` (for the A and B samples). The `includeTest` for `fA1` is simply `function(fh) TRUE`, since there will always be at least one peak to fit. `fB1` is included if there is more than 1 detected peak, and the setting `samples` is more than 1, so the `includeTest` is

```
function(fh) nrow(fhPeaks(fh)) > 1 && fhSamples(fh) > 1
```

The G1 component is defined by the function

```
(a1 / (sqrt(2 * pi) * Sa) * exp(-((xx - Ma)^2)/(2 *
  Sa^2)))
```

with the arguments `a1,Ma,Sa,xx`. `xx` is treated specially, by default, and we don't need to deal with it here. The initial estimates for the other parameters are calculated in `initParams`:

```
function(fh){
  Ma <- as.numeric(fhPeaks(fh)[1, "mean"])
  Sa <- as.numeric(Ma / 20)
  a1 <- as.numeric(fhPeaks(fh)[1, "height"] * Sa / 0.45)
  list(Ma = Ma, Sa = Sa, a1 = a1)
}
```

`Ma` is the mean of the distribution, which should be very close to the peak. `Sa` is the standard distribution of the distribution. If we assume the CV is 5%, that means the `Sa` should be 5% of the distribution mean, which gives us a good first estimate. `a1` is a scaling parameter, and I came up with the initial estimate by trial-and-error. Given the other two values are going to be reasonably close, the starting value of `a1` doesn't seem to be that crucial.

The limits for these values are provided in `paramLimits`.

```
paramLimits = list(Ma = c(0, Inf), Sa = c(0, Inf), a1 =
  c(0, Inf))
```

They're all bound between 0 and Infinity. The upper bound for `Ma` and `Sa` could be lowered to the number of bins, but I haven't had time or need to explore this yet.

The G2 peaks include the `d` argument, which is the ratio of the G2 peak to the G1 peak. That is, the linearity parameter:

```
func = function(a2, Ma, Sa, d, xx){
  (a2 / (sqrt(2 * pi) * Sa * 2) *
    exp(-((xx - Ma * d)^2)/(2 * (Sa * 2)^2)))
}
```

`d` is the ratio between the G2 and G1 peaks. If `linearity = "fixed"`, it is set to 2. Otherwise, it is fit as a model parameter. This requires special handling. First, we check the `linearity` value in `initParams`, and provide a value for `d` if needed:

```
res <- list(a2 = a2)
if(fhLinearity(fh) == "variable")
    res <- c(res, d = 2)
```

Here, a2 is always treated as a parameter, and d is appended to the initial paramter list only if needed.

We also need to use the specialParamSetter function, in this case calling the helper function setLinearity(fh). This function checks the value of linearity, and returns the appropriate object depending on the result.

Note that we use the arguments Ma and Sa appear in the function slot for fA2, but we don't need to provide their initial values or limits. These values are already supplied in the definition of fA1, which is always present when fA2 is.

NB.: This isn't checked in the code! I know fA1 is always present, but there is no automated checking of this fact. If you create a ModelComponent that has parameters that are not defined in that component, and are not defined in other components (like Ma is in this case), you will cause problems. There is also nothing to stop you from defining a parameter multiple times. That is, you could define initial estimates and limits for Ma in fA1 and fA2. This may also cause problems. It would be nice to do some sanity-checking to protect against using parameters without defining initial estimates or limits, or providing multiple/conflicting definitions.

The Single-Cut Debris component is unusual in two ways. It doesn't include the argument xx, but it uses the pre-computed values SCvals. Consequently, we must provide a function for specialParamSetter to deal with this:

```
specialParamSetter = function(fh){ list(SCvals =
substitute(SCvals)) }
```

The Multi-Cut Debris component MC is similar, but it needs to include xx as a special parameter. The aggregate component AG also includes several special parameters.

For more discussion of the debris components, see DebrisModels.

The code responsible for this is in the file models.R. Accessor functions are provided (but not exported) for getting and setting ModelComponent slots. These functions are named mcSLOT, and include mcFunc, mcColor, mcName, mcDesc, mcSpecialParams, mcSpecialParamSetter, mcIncludeTest, mcInitParams.

Examples

```
## The 'master list' of components is stored in fhComponents:
flowPloidy:::fhComponents ## outputs a list of component summaries

## adding a new component to the list:
## Not run:
fhComponents$pois <-
  new("ModelComponent", name = "pois", color = "bisque",
      desc = "A poisson component, as a silly example",
      includeTest = function(fh){
          ## in this case, we check for a flag in the opt slot
          ## We could also base the test on some feature of the
          ## data, perhaps something in the peaks or histData slots
          "pois" %in% fh@opt
      },
      func = function(xx, plam){
          ## The function needs to be complete on a single line, as it
          ## will be 'stitched' together with other functions to make
          ## the complete model.
          exp(-plam)*plam^xx/factorial(xx)
      },
      initParams = function(fh){
```

```
            ## If we were to use this function for one of our peaks, we
            ## could use the peak position as our initial estimate of
            ## the Poisson rate parameter:
            plam <- as.numeric(fhPeaks(fh)[1, "mean"])
        },
        ## bound the search for plam between 0 and infinity. Tighter
        ## bounds might be useful, if possible, in speeding up model
        ## fitting and avoiding local minima in extremes.
        paramLimits = list(plam = c(0, Inf))
    )

    ## specialParamSetter is not needed here - it will default to a
    ## function that returns "xx = xx", indicating that all other
    ## parameters will be fit. That is what we need for this example. If
    ## the component doesn't include xx, or includes other fixed
    ## parameters, then specialParamSetter will need to be provided.

    ## Note that if our intention is to replace an existing component with
    ## a new one, we either need to explicitly change the includeTest for
    ## the existing component to account for situations when the new one
    ## is used instead. As a temporary hack, you could add both and then
    ## manually remove one with \code{dropComponents}.

## End(Not run)
```

`pickInit` *Interactively select model starting values*

Description

Prompts the user to select the peaks to use as initial values for non-linear regression on a plot of the histogram data.

Usage

```
pickInit(fh)
```

Arguments

fh	A FlowHist object

Details

The raw histogram data are plotted, and the user is prompted to select the peak positions to use as starting values in the NLS procedure. This is useful when the automated peak-finding algorithm fails to discriminate between overlapping peaks, or is confused by noise.

Note that the A peak must be lower (smaller mean, further left) than the B peak. If the user selects the A peak with a higher mean than the B peak, the peaks will be swapped to ensure A is lower.

Value

`pickInit` returns the `FlowHist` object with its initial value slot updated.

Author(s)

Tyler Smith

Examples

```
library(flowPloidyData)
fh2 <- FlowHist(file = flowPloidyFiles()[2], channel = "FL3.INT.LIN")
plot(fh2, init = TRUE) ## automatic peak estimates
## Not run:
fh2 <- pickInit(fh2)   ## hand-pick peak estimates

## End(Not run)
plot(fh2, init = TRUE) ## revised starting values
```

---

plot.FlowHist *Plot histograms for FlowHist objects*

Description

Plot histograms for FlowHist objects

Usage

```
## S3 method for class 'FlowHist'
plot(x, init = FALSE, nls = TRUE, comps = TRUE,
  main = fhFile(x), ...)
```

Arguments

x	a FlowHist object
init	boolean; if TRUE, plot the regression model using the initial parameter estimates over the raw data.
nls	boolean; if TRUE, plot the fitted regression model over the raw data (i.e., using the final parameter values)
comps	boolean; if TRUE, plot the individual model components over the raw data.
main	character; the plot title. Defaults to the filename of the FlowHist object.
...	additional arguments passed on to plot()

Value

Not applicable

Author(s)

Tyler Smith

plotFH — *Plot the raw data for a FlowHist object*

Description

Creates a simple plot of the raw histogram data. Used as a utility for other plotting functions, and perhaps useful for users who wish to create their own plotting routines.

Usage

```
plotFH(fh, main = fhFile(fh), ...)
```

Arguments

fh	a FlowHist object
main	character; the plot title. Defaults to the filename of the FlowHist object.
...	additional parameters passed to plot

Value

Not applicable, used for plotting

Author(s)

Tyler Smith

Examples

```
library(flowPloidyData)
fh1 <- FlowHist(file = flowPloidyFiles()[1], channel = "FL3.INT.LIN")
plotFH(fh1)
```

setBins — *setBins*

Description

(Re-)set the bins for a FlowHist object

Usage

```
setBins(fh, bins = 256)
```

Arguments

fh	a FlowHist object
bins	integer, the number of bins to use in aggregating FCS data

Details

This function sets (or resets) the number of bins to use in aggregating FCS data into a histogram, and generates the corresponding data matrix. Not exported for general use.

The `histData` matrix also contains the columns corresponding to the raw data used in calculating the single-cut and multiple-cut debris components, as well as the doublet, triplet, and quadruplet aggregate values. (i.e., `SCvals`, `MCvals`, `DBvals`, `TRvals`, and `QDvals`).

`setBins` includes a call to `resetFlowHist`, so all the model components that depend on the bins are updated in the process (as you want!).

Value

a `FlowHist` object, with the `bins` slot set to `bins`, and the corresonding binned data stored in a matrix in the `histData` slot. Any previous analysis slots are removed: `peaks,comps,model,init,nls,counts,CV,RCS`.

Author(s)

Tyler Smith

Examples

```
## defaults to 256 bins:
library(flowPloidyData)
fh1 <- FlowHist(file = flowPloidyFiles()[1], channel = "FL3.INT.LIN")
plot(fh1)
## reset them to 512 bins:
fh1 <- setBins(fh1, 512)
plot(fh1)
```

`tabulateFlowHist` *exportFlowHist*

Description

Extract analysis results from a FlowHist object

Usage

```
tabulateFlowHist(fh, file = NULL)
```

Arguments

`fh`	a `FlowHist` object, or a list of `FlowHist` objects.
`file`	character, the name of the file to save data to

Details

A convenience function for extracting the results of the NLS curve-fitting analysis on a FlowHist object.

If `fh` is a single FlowHist object, a data.frame with a single row is returned. If `fh` is a list of `FlowHist` objects, a row for each object will be added to the data.frame.

If a file name is provided, the data will be saved to that file.

The columns of the returned data.frame may include:

StdPeak: which peak (A, B etc) was identified by the user as the internal standard

ratio: the ratio of the sample peak size to the standard peak size, if the standard size was set and the standard peak identified

StdSize: the size of the standard in pg, if set

pg: genome size estimate, if the sample peak was identified and the size of the standard was set

RCS: the residual Chi-Square for the model fit

a_mean, b_mean etc: the peak position for the G1 peak of each sample

a_stddev, b_stddev etc: standard devation for each G1 peak position

a1_count, b1_count etc: the cell counts for the G1 peak of each sample

a2_count, b2_count etc: the cell counts for the G2 peak of each sample

a_s_count, b_s_count etc: the cell counts for the S-phase for each sample

a_CV, b_CV etc: the coefficient of variation for each sample

linearity: the linearity value, if not fixed at 2

Note that columns are only produced for parameters that exist in your data. That is, if none of your samples have a G2 peak for the A sample, you won't get a2_count column. Similarly, if you didn't set the standard size, or identify which peak was the standard, you won't get StdPeak, ratio, StdSize, or pg columns.

Value

a data frame

Author(s)

Tyler Smith

Examples

```
library(flowPloidyData)
fh1 <- FlowHist(file = flowPloidyFiles()[1], channel = "FL3.INT.LIN")
fh1 <- fhAnalyze(fh1)
tabulateFlowHist(fh1)
```

updateFlowHist *updateFlowHist*

Description

Update, and optionally re-analyze, a FlowHist object

Usage

```
updateFlowHist(fh, linearity = NULL, debris = NULL, samples = NULL,
  analyze = TRUE)
```

Arguments

fh	a FlowHist object
linearity	character, either "variable", the default, or "fixed". If "fixed", linearity is fixed at 2; if "variable", linearity is fit as a model parameter.
debris	character, either "SC", the default, or "MC", to set the debris model component to the Single-Cut or Multi-Cut models.
samples	integer, the number of samples in the data
analyze	logical, if TRUE the updated model will be analyzed immediately

Details

Allows users to switch the debris model from Single-Cut to Multi-Cut (or vice-versa), or to toggle linearity between fixed and variable.

Value

a FlowHist object with the modified values of linearity and/or debris, and, if analyze was TRUE, a new NLS fitting

Author(s)

Tyler Smith

Examples

```
## defaults to 256 bins:
library(flowPloidyData)
fh1 <- FlowHist(file = flowPloidyFiles()[1], channel = "FL3.INT.LIN")
## default is Single-Cut, change that to Multi-Cut:
fh1mc <- updateFlowHist(fh1, debris = "MC")
plot(fh1)
```

`viewFlowChannels` *viewFlowChannels*

Description

Displays the column names present in an FCS file

Usage

```
viewFlowChannels(file, emptyValue = TRUE)
```

Arguments

`file`	character, the name of an FCS data file; or the name of a FlowHist object.
`emptyValue`	boolean, passed to `read.FCS`, needed to deal with unusual FCS file formats. Default is TRUE - if your file loads without errors, then don't change this value

Details

A convenience function for viewing column names in a FCS data file, or a FlowHist object. Used to select one for the `channel` argument in `FlowHist`, or for viewing additional channels for use in gating.

Value

A vector of column names from the FCS file/FlowHist object.

Author(s)

Tyler Smith

See Also

`FlowHist`

Examples

```
library(flowPloidyData)
viewFlowChannels(flowPloidyFiles()[1])
```

Index