Package 'cyanoFilter'

April 5, 2026

Type Package

Title Phytoplankton Population Identification using Cell Pigmentation and/or Complexity

Version 1.18.0

Description An approach to filter out and/or identify phytoplankton cells from all particles measured via flow cytometry pigment and cell complexity information. It does this using a sequence of one-dimensional gates on pre-defined channels measuring certain pigmentation and complexity. The package is especially tuned for cyanobacteria, but will work fine for phytoplankton communities where there is at least one cell characteristic that differentiates every phytoplankton in the community.

URL https://github.com/fomotis/cyanoFilter

BugReports https://github.com/fomotis/cyanoFilter/issues

Depends R(>= 4.1.0)

Imports Biobase, flowCore, flowDensity, flowClust, cytometree, ggplot2, GGally, graphics, grDevices, methods, mrfDepth, stats, utils

License MIT + file LICENSE

biocViews FlowCytometry, Clustering, OneChannel

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Suggests magrittr, dplyr, purrr, knitr, stringr, rmarkdown, tidyr

VignetteBuilder knitr

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

git_branch RELEASE_3_22

git_last_commit e3c99a3

git_last_commit_date 2025-10-29

Repository Bioconductor 3.22

Date/Publication 2026-04-05

Author Oluwafemi Olusoji [cre, aut],
Aerts Marc [ctb],
Delaender Frederik [ctb],
Neyens Thomas [ctb],
Spaak jurg [aut]

Maintainer Oluwafemi Olusoji <oluwafemi.olusoji@uhasselt.be>

Contents

accTest *tests the accuracy of several automated gating functions on monoculture flow cytometry experiments.*

Description

This function gates all flowFrames in the supplied flowSet to attach cluster labels. Then it mixes up the flowSet into one giant flowFrame and re-gates this to attach another label. These labels are used to examine if the gating algorithms can reproduce the earlier clusters before the mixing.

Usage

```
accTest(
  fs,
  sfts = c("phytoFilter", "flowClust", "cytometree"),
  channels,
  nrun = 10000,
  ...
)
```

Arguments

fs	flowSet with each flowFrame being a phytoplankton monoculture FCM experiment
sfts	character vector of gating function to test.
channels	channels to be used for gating
nrun	number of times the resampling should be done
...	extra options to be parsed to the gating function

Value

a named list containing the following objects;

- **depth -** the multivariate-depth (median) of each flowFrame in the flowset supplied
- **accuracy -** computed accuracy based on resampling after joining the flowFrames together.

Examples

```
flowfile_path <- system.file("extdata", "B4_18_1.fcs",
package = "cyanoFilter",
              mustWork = TRUE)
flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE,
                               transformation = FALSE,
                               emptyValue = FALSE,
                               dataset = 1)
flowfile_nona <- cyanoFilter::noNA(x = flowfile)
flowfile_noneg <- cyanoFilter::noNeg(x = flowfile_nona)
flowfile_logtrans <- cyanoFilter::lnTrans(x = flowfile_noneg,
                      c('SSC.W', 'TIME'))
cells_nonmargin <- cellMargin(flowframe = flowfile_logtrans,
                              Channel = 'SSC.W',
```

```
           type = 'estimate', y_toplot = "FSC.HLin")
cells_nodebris <- debrisNc(flowframe = reducedFlowframe(cells_nonmargin),
                           ch_chlorophyll = "RED.B.HLin",
                           ch_p2 = "YEL.B.HLin",
                           ph = 0.05)
#phytoFilter specification
gateFunc(flowfile = reducedFlowframe(cells_nodebris),
              channels = c("RED.B.HLin", "YEL.B.HLin",
              "RED.R.HLin", "FSC.HLin", "SSC.HLin"),
              sfts = "phytoFilter",
              list(ph = 0.1, proportion = 0.90)
              )
```

`accuracy` *samples two rows in a matrix and check if the samples are similar or different based on their cluster labels*

Description

This function

Usage

```
accuracy(mat, mono_clust, bi_clust, nrun = 10000)
```

Arguments

`mat`	matrix to be sampled from
`mono_clust`	monoculture cluster label
`bi_clust`	biculture cluster label
`nrun`	number of times the resampling should be carried out. Defaults to 10000

Value

a vector of integer values

Examples

```
x <- matrix(NA, nrow = 100, ncol = 3)
xx <- apply(x, 2, rnorm, 100)
xx <- cbind(xx, Mono = rep(1:2, each = 50),
            Bi = rep(1:2, times = 50))
accuracy(xx, "Mono", "Bi", nrun = 5000)
```

`cellMargin` *Removes or assign indicators to margin events.*

Description

The function identifies margin events, i.e. cells that are too large for the flow cytometer to measure.

Usage

```
cellMargin(
  flowframe,
  Channel = "SSC.W",
  type = c("manual", "estimate"),
  cut = NULL,
  y_toplot = "FSC,HLin"
)
```

Arguments

`flowframe`	Flowframe containing margin events to be filtered out
`Channel`	The channel on which margin events are. Defaults to SSC.W (side scatter width)
`type`	The method to be used in gating out the margin cells. Can either be 'manual' where user supplies a cut off point on the channel, 1 = not margin 0 = margin
`cut`	sould not be NULL if type = 'manual'
`y_toplot`	channel on y-axis of plot with *Channel* used to gate out margin events

Details

Users can either supply a cut-off point along the channel describing particle width or allow the function to estimate the cut-off point using the `deGate` function from the *flowDensity* package. A plot of channel against "FSC.HLin" is provided with a vertical line showing the cut-off point separating margin events from other cells.

Value

an object of class MarginEvents class containing slots;

- **reducedflowframe -** flowframe without margin events
- **fullflowframe -** flowframe with an Margin.Indicator added as an extra column added to the expression matrix to indicate which particles are margin events. 1 = not margin event, 0 = margin event
- **N_margin -** number of margin events recorded
- **N_cell -** numner of non-margin events
- **N_particle -** is the number of particles in total, i.e. N_cell + N_margin

Examples

```
flowfile_path <- system.file("extdata", "B4_18_1.fcs",
                 package = "cyanoFilter",
                 mustWork = TRUE)
flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE,
                               transformation = FALSE, emptyValue = FALSE,
                               dataset = 1)
flowfile_nona <- cyanoFilter::noNA(x = flowfile)
flowfile_noneg <- cyanoFilter::noNeg(x = flowfile_nona)
flowfile_logtrans <- lnTrans(x = flowfile_noneg, c('SSC.W', 'TIME'))
cellMargin(flowframe = flowfile_logtrans, Channel = 'SSC.W',
           type = 'estimate', y_toplot = "FSC.HLin")
```

`clusterExtract` *extract clusters based on supplied cluster indicator*

Description

extract clusters based on supplied cluster indicator

Usage

```
clusterExtract(flowfile, cluster_var = "Clusters", cluster_val = NULL)
```

Arguments

`flowfile`	flowframe containing cluster indicators as well
`cluster_var`	column name in expression matrix containing the cluter indicators, cannot be NULL.
`cluster_val`	cluster number, cannot be NULL.

Value

flowFrame containing the clusters

Examples

```
flowfile_path <- system.file("extdata", "B4_18_1.fcs",
                             package = "cyanoFilter",
                             mustWork = TRUE)
flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE,
                               transformation = FALSE, emptyValue = FALSE,
                               dataset = 1)
flowfile_nona <- cyanoFilter::noNA(x = flowfile)
flowfile_noneg <- cyanoFilter::noNeg(x = flowfile_nona)
flowfile_logtrans <- cyanoFilter::lnTrans(x = flowfile_noneg,
                                          c('SSC.W', 'TIME'))
cells_nonmargin <- cellMargin(flowframe = flowfile_logtrans,
                              Channel = 'SSC.W',
           type = 'estimate', y_toplot = "FSC.HLin")
fin <- phytoFilter(flowfile = reducedFlowframe(cells_nonmargin),
              pig_channels = c("RED.B.HLin", "YEL.B.HLin", "RED.R.HLin"),
```

```
              com_channels = c("FSC.HLin", "SSC.HLin"))

clusterExtract(flowfile = reducedFlowframe(fin),
    cluster_var = "Clusters",
    cluster_val = 1)
```

---

clusterExtractp *takes a flowframe, name of cluster column and extracts part of flowframe that makes up proportion.*

Description

takes a flowframe, name of cluster column and extracts part of flowframe that makes up proportion.

Usage

```
clusterExtractp(flowfile, cluster_var = "Clusters", proportion = 1)
```

Arguments

flowfile	flowframe after debris are removed.
cluster_var	column name in expression matrix containing the cluter indicators
proportion	value between 0 and 1 indicating percentage of the total particles wanted

Value

a list containing

- **particles_per_cluster**
- **clusters_proportion**
- **flowfile_proportion**

Examples

```
flowfile_path <- system.file("extdata", "B4_18_1.fcs",
                             package = "cyanoFilter",
                             mustWork = TRUE)
flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE,
                               transformation = FALSE, emptyValue = FALSE,
                               dataset = 1)
flowfile_nona <- cyanoFilter::noNA(x = flowfile)
flowfile_noneg <- cyanoFilter::noNeg(x = flowfile_nona)
flowfile_logtrans <- cyanoFilter::lnTrans(x = flowfile_noneg,
                                          c('SSC.W', 'TIME'))
cells_nonmargin <- cyanoFilter::cellMargin(flowframe = flowfile_logtrans,
                              Channel = 'SSC.W',
           type = 'estimate', y_toplot = "FSC.HLin")
fin <- phytoFilter(flowfile = reducedFlowframe(cells_nonmargin),
              pig_channels = c("RED.B.HLin", "YEL.B.HLin", "RED.R.HLin"),
              com_channels = c("FSC.HLin", "SSC.HLin"))

clusterExtractp(flowfile = reducedFlowframe(fin),
```

```
    cluster_var = "Clusters",
    proportion = 0.80)
```

`cyanoFilter` *cyanoFilter: A package to identify and cluster phytoplankton cells contained in flow cytometry data.*

Description

The package provides two categories of functions: *metafile* preprocessing functions and *fcsfile* processing functions.

metafile preprocessing functions

This set of functions (`goodFcs` and `retain`) helps to identify the appropriate fcs file to read.

fcsfile processing functions

These functions (`noNA` and `noNeg`, `phytoFilter`) works on the fcs file to identify the phytoplankton populations contained in the fcs file.

`DebrisFilter` *the Debris class*

Description

the Debris class

constructor for the DebrisFilter class

Usage

```
DebrisFilter(
  fullflowframe,
  reducedflowframe,
  deb_pos,
  syn_all_pos,
  deb_cut,
  ch_chlorophyll,
  ch_p2
)

DebrisFilter(
  fullflowframe,
  reducedflowframe,
  deb_pos,
  syn_all_pos,
  deb_cut,
  ch_chlorophyll,
  ch_p2
)
```

Arguments

fullflowframe	same as the input flowFrame
reducedflowframe	a partial flowframe containing non-margin events
deb_pos	number of margin particles measured
syn_all_pos	number of non-margine particles
deb_cut	estimated inflection point between debris and good cells
ch_chlorophyll	channel estimating chlorophyll level
ch_p2	plotting channel

Value

object of class DebrisFilter

Slots

fullflowframe object of class "flowFrame" same as the input flowFrame

reducedflowframe object of class "flowFrame" a partial flowframe containing a proportion of the measured particles

deb_pos object of class "numeric" representing the proportion of particles in each cluster

syn_all_pos object of class "numeric" representing the number of particles in each cluster

deb_cut object of class "numeric" representing the inflection point between debris and good cells.

ch_chlorophyll objet of class "character" representing the chlorophyll channel.

ch_p2 object of class character to plot

---

debrisNc *gates out or assign indicators to debris particle based on their chlorophyll expression.*

Description

The function takes in a flowframe and identifies debris contained in the provided flowframe.

Usage

```
debrisNc(flowframe, ch_chlorophyll, ch_p2, ph = 0.09, n_sd = 2)
```

Arguments

flowframe	flowframe with debris and other cells.
ch_chlorophyll	first flowcytometer channel that can be used to separate debris from the rest, e.g. "RED.B.HLin".
ch_p2	second flowcytometer channel use for plotting from the rest, e.g. "YEL.B.HLin"
ph	the minimum peak height that should be considered. This aids the removal of tiny peaks. Defaults to 0.1
n_sd	number of standard deviations away from peak should be considered to filter out debris

Details

The function uses the getPeaks and deGate functions in the flowDensity package to identify peaks in ch_chlorophyll, and identify cut-off points #between these peaks. A plot of both channels supplied with horizontal line separating debris from other cell populations is also returned.

Value

list containing;

- **syn - flowframe containing non-debris particles**
- **deb_pos - position of particles that are debris**
- **syn_pos - position of particles that are not debris**

Examples

```
flowfile_path <- system.file("extdata", "B4_18_1.fcs",
                  package = "cyanoFilter",
              mustWork = TRUE)
flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE,
                               transformation = FALSE, emptyValue = FALSE,
                               dataset = 1)
flowfile_nona <- cyanoFilter::noNA(x = flowfile)
flowfile_noneg <- cyanoFilter::noNeg(x = flowfile_nona)
flowfile_logtrans <- cyanoFilter::lnTrans(x = flowfile_noneg,
c('SSC.W', 'TIME'))
cells_nonmargin <- cellMargin(flowframe = flowfile_logtrans,
Channel = 'SSC.W',
           type = 'estimate', y_toplot = "FSC.HLin")
debrisNc(flowframe = reducedFlowframe(cells_nonmargin),
          ch_chlorophyll = "RED.B.HLin",
          ch_p2 = "YEL.B.HLin",
          ph = 0.05)
```

`fullFlowframe` *generic function for extracting the full flowframe*

Description

generic function for extracting the full flowframe

Usage

```
fullFlowframe(x)
```

Arguments

x	an object of either class PhytoFilter, MarginEvents or DebrisFilter

Value

generic to extract fullFlowframe

Examples

```
flowfile_path <- system.file("extdata", "B4_18_1.fcs",
package = "cyanoFilter",
              mustWork = TRUE)
flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE,
                               transformation = FALSE,
                               emptyValue = FALSE,
                               dataset = 1)
flowfile_nona <- cyanoFilter::noNA(x = flowfile)
flowfile_noneg <- cyanoFilter::noNeg(x = flowfile_nona)
flowfile_logtrans <- cyanoFilter::lnTrans(x = flowfile_noneg,
                      c('SSC.W', 'TIME'))
cells_nonmargin <- cellMargin(flowframe = flowfile_logtrans,
                              Channel = 'SSC.W',
           type = 'estimate', y_toplot = "FSC.HLin")
 reducedFlowframe(cells_nonmargin)
```

`fullFlowframe,DebrisFilter-method`
accesor method for reduced flowframe (DebrisFilter class)

Description

accesor method for reduced flowframe (DebrisFilter class)

Usage

```
## S4 method for signature 'DebrisFilter'
fullFlowframe(x)
```

Arguments

x	an object of class DebrisFilter

Value

full flowFrame method for DebrisFilter

`fullFlowframe,MarginEvents-method`
accesor method for the fullflowframe (MarginEvent class)

Description

accesor method for the fullflowframe (MarginEvent class)

Usage

```
## S4 method for signature 'MarginEvents'
fullFlowframe(x)
```

Arguments

x	an object of class MarginEvents

Value

full Flowframe method for MarginEvents

Examples

```
flowfile_path <- system.file("extdata", "B4_18_1.fcs",
                  package = "cyanoFilter",
              mustWork = TRUE)
flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE,
                               transformation = FALSE, emptyValue = FALSE,
                               dataset = 1)
flowfile_nona <- cyanoFilter::noNA(x = flowfile)
flowfile_noneg <- cyanoFilter::noNeg(x = flowfile_nona)
flowfile_logtrans <- cyanoFilter::lnTrans(x = flowfile_noneg,
c('SSC.W', 'TIME'))
cells_nonmargin <- cellMargin(flowframe = flowfile_logtrans,
Channel = 'SSC.W',
           type = 'estimate', y_toplot = "FSC.HLin")
fullFlowframe(cells_nonmargin)
```

---

fullFlowframe,PhytopFilter-method

accesor method for full flowframe(PhytoFilter class)

Description

accesor method for full flowframe(PhytoFilter class)

Usage

```
## S4 method for signature 'PhytopFilter'
fullFlowframe(x)
```

Arguments

x	an object of class PhytoFilter

Value

fullFlowframe method for PhytoFilter

Examples

```
flowfile_path <- system.file("extdata", "B4_18_1.fcs",
package = "cyanoFilter",
              mustWork = TRUE)
flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE,
                               transformation = FALSE,
                               emptyValue = FALSE,
```

```
                                dataset = 1)
flowfile_nona <- cyanoFilter::noNA(x = flowfile)
flowfile_noneg <- cyanoFilter::noNeg(x = flowfile_nona)
flowfile_logtrans <- cyanoFilter::lnTrans(x = flowfile_noneg,
                      c('SSC.W', 'TIME'))
cells_nonmargin <- cellMargin(flowframe = flowfile_logtrans,
                              Channel = 'SSC.W',
           type = 'estimate', y_toplot = "FSC.HLin")
cells_nodebris <- debrisNc(flowframe = reducedFlowframe(cells_nonmargin),
                           ch_chlorophyll = "RED.B.HLin",
                           ch_p2 = "YEL.B.HLin",
                           ph = 0.05)
phy1 <- phytoFilter(flowfile = reducedFlowframe(cells_nodebris),
              pig_channels = c("RED.B.HLin", "YEL.B.HLin", "RED.R.HLin"),
              com_channels = c("FSC.HLin", "SSC.HLin"))
fullFlowframe(phy1)
```

`gateFunc` *tests the accuracy of several automated gating functions on monoculture flow cytometry experiments.*

Description

This function gates all flowFrames in the supplied flowSet to attach cluster labels. Then it mixes up the flowSet into one giant flowFrame and re-gates this to attach another label. These labels are used to examine if the gating algorithms can reproduce the earlier clusters before the mixing.

Usage

```
gateFunc(
  flowfile,
  sfts = c("phytoFilter", "flowClust", "cytometree"),
  channels,
  funargs_list
)
```

Arguments

`flowfile`	flowSet with each flowFrame being a phytoplankton monoculture FCM experiment
`sfts`	character vector of gating function to test.
`channels`	channels to be used for gating
`funargs_list`	additional options for the chosen gating function

Value

a flowFrame with cluster indicator generated by the software used added to the expression matrix.

Examples

```
 flowfile_path <- system.file("extdata", "B4_18_1.fcs",
package = "cyanoFilter",
              mustWork = TRUE)
flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE,
                               transformation = FALSE,
                               emptyValue = FALSE,
                               dataset = 1)
flowfile_nona <- cyanoFilter::noNA(x = flowfile)
flowfile_noneg <- cyanoFilter::noNeg(x = flowfile_nona)
flowfile_logtrans <- cyanoFilter::lnTrans(x = flowfile_noneg,
                      c('SSC.W', 'TIME'))
cells_nonmargin <- cellMargin(flowframe = flowfile_logtrans,
                              Channel = 'SSC.W',
           type = 'estimate', y_toplot = "FSC.HLin")
cells_nodebris <- debrisNc(flowframe = reducedFlowframe(cells_nonmargin),
                           ch_chlorophyll = "RED.B.HLin",
                           ch_p2 = "YEL.B.HLin",
                           ph = 0.05)
#phytoFilter specification
gateFunc(flowfile = reducedFlowframe(cells_nodebris),
              channels = c("RED.B.HLin", "YEL.B.HLin",
              "RED.R.HLin", "FSC.HLin", "SSC.HLin"),
              sfts = "phytoFilter",
              list(ph = 0.1, proportion = 0.90)
              )
#flowClust specification
gateFunc(flowfile = reducedFlowframe(cells_nodebris),
              channels = c("RED.B.HLin", "YEL.B.HLin",
              "RED.R.HLin", "FSC.HLin", "SSC.HLin"),
              sfts = "flowClust",
              list(K = 1:4, B = 100)
              )
#cytometree specification
gateFunc(flowfile = reducedFlowframe(cells_nodebris),
              channels = c("RED.B.HLin", "YEL.B.HLin",
              "RED.R.HLin", "FSC.HLin", "SSC.HLin"),
              sfts = "cytometree",
              list(minleaf = 1, t = 0.10)
              )
```

`getChannel` *returns the channel with more than one peak present. It returns NA if there is only one peak present.*

Description

returns the channel with more than one peak present. It returns NA if there is only one peak present.

Usage

```
getChannel(flowfile, ch, ph)
```

Arguments

flowfile	flowframe after debris are removed.
ch	channel to be checked for multiple peaks.
ph	maximum peak height to be ignored. This allows ignoring of tiny peaks that could affect the gating process.

Value

name of channel with more than one peak

@examples flowfile_path <- system.file("extdata", "B4_18_1.fcs", package = "cyanoFilter", mustWork = TRUE) flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE, transformation = FALSE, emptyValue = FALSE, dataset = 1) flowfile_nona <- cyanoFilter::noNA(x = flowfile) flowfile_noneg <- cyanoFilter::noNeg(x = flowfile_nona) flowfile_logtrans <- cyanoFilter::lnTrans(x = flowfile_noneg, c('SSC.W', 'TIME')) getChannel(flowfile_logtrans, 'RED.B.HLin', 0.05)

---

ggpairsDens *produces a scatter plot of the expression matrix of the flowframe. If a cluster variable is given, it assigns different colors to the clusters.*

Description

produces a scatter plot of the expression matrix of the flowframe. If a cluster variable is given, it assigns different colors to the clusters.

Usage

```
ggpairsDens(flowfile, channels = NULL, group = NULL, notToPlot = NULL, ...)
```

Arguments

flowfile	flowframe to be plotted
channels	a character vector of length 2 or more. It must contain channel names in the flowfile.
group	cluster groups. It must be equal to the number of particles in the flowfile. If group is null cluster boundaries are not drawn.
notToPlot	columns not to plot. This is especially useful for for plotting all columns in a
...	not used at the moment @return a ggplot object

Value

a ggplot object

Examples

```
 # example without clustering
flowfile_path <- system.file("extdata", "B4_18_1.fcs",
                 package = "cyanoFilter",
                 mustWork = TRUE)
flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE,
                               transformation = FALSE, emptyValue = FALSE,
                               dataset = 1)
flowfile_nona <- cyanoFilter::noNA(x = flowfile)
flowfile_noneg <- cyanoFilter::noNeg(x = flowfile_nona)
flowfile_logtrans <- lnTrans(x = flowfile_noneg, c('SSC.W', 'TIME'))
ggpairsDens(flowfile = flowfile_logtrans,
            channels = c("FSC.HLin", "RED.R.HLin", "RED.B.HLin",
            "NIR.R.HLin"))
```

`ggplotDens` *plots two channels of a flowframe.*

Description

plots two channels of a flowframe.

Usage

```
ggplotDens(flowfile, channels, ...)
```

Arguments

`flowfile`	flowframe to be plotted
`channels`	a character vector of length 2, must contain channel names in the flowfile.
`...`	not used at the moment

Value

a ggplot object

Examples

```
flowfile_path <- system.file("extdata", "B4_18_1.fcs",
              package = "cyanoFilter",
              mustWork = TRUE)
flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE,
                               transformation = FALSE, emptyValue = FALSE,
                               dataset = 1)
flowfile_nona <- cyanoFilter::noNA(x = flowfile)
flowfile_noneg <- cyanoFilter::noNeg(x = flowfile_nona)
flowfile_logtrans <- cyanoFilter::lnTrans(x = flowfile_noneg,
                     c('SSC.W', 'TIME'))
ggplotDens(flowfile_logtrans,
           channels = c("FSC.HLin", "RED.R.HLin"))
```

ggplotDens2 *plots two channels of a flowframe with different colors for clusters identified.*

Description

plots two channels of a flowframe with different colors for clusters identified.

Usage

```
ggplotDens2(flowfile, channels, group, ...)
```

Arguments

flowfile	flowframe to be plotted
channels	a character vector of length 2, must contain channel names in the flowfile.
group	cluster groups. must be equal to the number of particles in the flow cytometer.
...	not used at the moment

Value

a ggplot object

Examples

```
flowfile_path <- system.file("extdata", "B4_18_1.fcs",
package = "cyanoFilter",
              mustWork = TRUE)
flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE,
                               transformation = FALSE,
                               emptyValue = FALSE,
                               dataset = 1)
flowfile_nona <- cyanoFilter::noNA(x = flowfile)
flowfile_noneg <- cyanoFilter::noNeg(x = flowfile_nona)
flowfile_logtrans <- cyanoFilter::lnTrans(x = flowfile_noneg,
                      c('SSC.W', 'TIME'))
cells_nonmargin <- cellMargin(flowframe = flowfile_logtrans,
                              Channel = 'SSC.W',
           type = 'estimate', y_toplot = "FSC.HLin")
cells_nodebris <- debrisNc(flowframe = reducedFlowframe(cells_nonmargin),
                           ch_chlorophyll = "RED.B.HLin",
                           ch_p2 = "YEL.B.HLin",
                           ph = 0.05)
cct <- phytoFilter(flowfile = reducedFlowframe(cells_nodebris),
              pig_channels = c("RED.B.HLin", "YEL.B.HLin", "RED.R.HLin"),
              com_channels = c("FSC.HLin", "SSC.HLin"))
ggplotDens2(reducedFlowframe(cct),
c("RED.B.HLin", "YEL.B.HLin"),
group = "Clusters")
```

goodFcs *indicates if measurement from a flowfile is good or bad.*

Description

This function examines the column containig $cells/\mu L$ and determins if the measurement can be used for further analysis or not based on a supplied range.

Usage

```
goodFcs(metafile, col_cpml = "CellspML", mxd_cellpML = 1000, mnd_cellpML = 50)
```

Arguments

metafile	associated metafile to the supplied fcsfile. This is a csv file containig computed stats from the flow cytometer.
col_cpml	column name or column number in metafile containing cell per microlitre measurements.
mxd_cellpML	maximal accepted cell per microlitre. Flowfiles with larger cell per microlitre are termed bad. Defaults to 1000.
mnd_cellpML	minimum accepted cell per microlitre. Flowfiles with lesser cell per microlitre are termed bad. Defaults to 50.

Details

Most flow cytometer makers will always inform clients within which range can measurements from the machine be trusted. The machines normally stores the amount of $cells/\mu L$ it counted in a sample. Too large value could mean possible doublets and too low value could mean too little cells.

Value

character vector with length same as the number of rows in the metafile whose entries are **good** for good files and **bad** for bad files.

Examples

```
require("stringr")
 metadata <- system.file("extdata", "2019-03-25_Rstarted.csv",
 package = "cyanoFilter",
              mustWork = TRUE)
 metafile <- read.csv(metadata, skip = 7, stringsAsFactors = FALSE,
                      check.names = TRUE, encoding = "UTF-8")
 metafile <- metafile[, seq_len(65)] #first 65 columns contains useful information
 #extract the part of the Sample.ID that corresponds to BS4 or BS5
 metafile$Sample.ID2 <- stringr::str_extract(metafile$Sample.ID, "BS*[4-5]")
 #clean up the Cells.muL column
 names(metafile)[which(stringr::str_detect(names(metafile),
 "Cells."))] <- "CellspML"
 goodFcs(metafile = metafile, col_cpml = "CellspML", mxd_cellpML = 1000,
 mnd_cellpML = 50)
```

is.DebrisFilter *function to check if object is of class cyanoFilter(DebrisFilter)*

Description

function to check if object is of class cyanoFilter(DebrisFilter)

Usage

```
is.DebrisFilter(x)
```

Arguments

x	any R object

Value

TRUE if object is of class DebrisFilter. FALSE otherwise

Examples

```
x <- c(1, 5, 4)
is.DebrisFilter(x)
```

is.flowFrame *function to check if object is a flowFrame*

Description

function to check if object is a flowFrame

Usage

```
is.flowFrame(x)
```

Arguments

x	any R object

Value

TRUE if object is a flowFrame. FALSE otherwise

Examples

```
x <- c(1, 5, 4)
is.flowFrame(x)
```

is.flowSet *function to check if object is a flowSet*

Description

function to check if object is a flowSet

Usage

```
is.flowSet(x)
```

Arguments

x	any R object

Value

TRUE if object is a flowSet. FALSE otherwise

Examples

```
x <- c(1, 5, 4)
is.flowSet(x)
```

is.MarginEvents *function to check if object is of class cyanoFilter(MarginEvents)*

Description

function to check if object is of class cyanoFilter(MarginEvents)

Usage

```
is.MarginEvents(x)
```

Arguments

x	any R object

Value

TRUE if object is of class MarginEvents. FALSE otherwise

Examples

```
x <- c(1, 5, 4)
is.MarginEvents(x)
```

is.PhytopFilter *function to check if object is of class cyanoFilter(PhytoFilter)*

Description

function to check if object is of class cyanoFilter(PhytoFilter)

Usage

```
is.PhytopFilter(x)
```

Arguments

x	any R object

Value

TRUE if object is of class PhytoFilter. FALSE otherwise

Examples

```
x <- c(1, 5, 4)
is.PhytopFilter(x)
```

lnTrans *log transforms the expression matrix of a flowframe*

Description

log transforms the expression matrix of a flowframe

Usage

```
lnTrans(x, notToTransform = c("SSC.W", "TIME"))
```

Arguments

x	flowframe to be transformed
notToTransform	columns not to be transformed

Value

flowframe with log transformed expression matrix

Examples

```
flowfile_path <- system.file("extdata", "B4_18_1.fcs",
                  package = "cyanoFilter",
              mustWork = TRUE)
flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE,
                               transformation = FALSE, emptyValue = FALSE,
                               dataset = 1)
flowfile_nona <- cyanoFilter::noNA(x = flowfile)
flowfile_noneg <- cyanoFilter::noNeg(x = flowfile_nona)
lnTrans(x = flowfile_noneg, c('SSC.W', 'TIME'))
```

`MarginEvents` *the marginEvent class*

Description

the marginEvent class

constructor for the MarginEvents class

Usage

```
MarginEvents(
  fullflowframe,
  reducedflowframe,
  N_margin,
  N_nonmargin,
  N_particle,
  Channel,
  y_toplot,
  cut
)

MarginEvents(
  fullflowframe,
  reducedflowframe,
  N_margin,
  N_nonmargin,
  N_particle,
  Channel,
  y_toplot,
  cut
)
```

Arguments

`fullflowframe`	same as the input flowFrame
`reducedflowframe`	a partial flowframe containing non-margin events
`N_margin`	number of margin particles measured

N_nonmargin	number of non-margine particles
N_particle	total number of particles measured
Channel	channel measuring the width of the particles
y_toplot	another channel to use in a bivariate plot
cut	the cut-off point estimated or supplied.

Value

object of class MarginEvents

Slots

fullflowframe object of class "flowFrame" same as the input flowFrame

reducedflowframe object of class "flowFrame" a partial flowframe containing a proportion of the measured particles

N_margin object of class "numeric" representing the proportion of particles in each cluster

N_nonmargin object of class "integer" representing the number of particles in each cluster

N_particle object of class "integer" representing the labels for each cluster

Channel object of class character representing channel measuring cell width

y_toplot object of class character representing plot variable

cut object of class numberic representing estimated inflection point or supplied cut-off point

Examples

```
flowfile_path <- system.file("extdata", "B4_18_1.fcs",
                 package = "cyanoFilter",
                 mustWork = TRUE)
flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE,
                               transformation = FALSE, emptyValue = FALSE,
                               dataset = 1)
flowfile_nona <- cyanoFilter::noNA(x = flowfile)
flowfile_noneg <- cyanoFilter::noNeg(x = flowfile_nona)
flowfile_logtrans <- lnTrans(x = flowfile_noneg, c('SSC.W', 'TIME'))
cellMargin(flowframe = flowfile_logtrans, Channel = 'SSC.W',
           type = 'estimate', y_toplot = "FSC.HLin")
```

---

newFlowframe *takes a flowframe, a group indicator and formulates another flowframe with group indicator as part of the expression matrix of the new flowframe.*

Description

takes a flowframe, a group indicator and formulates another flowframe with group indicator as part of the expression matrix of the new flowframe.

Usage

```
newFlowframe(flowfile, group = NULL, togate = NULL)
```

Arguments

flowfile	flowframe after debris are removed.
group	cluster group to be added to the expression matrix
togate	channel detected to have more than one peak

Value

flowframe with indicators for particle cluster

Examples

```
flowfile_path <- system.file("extdata", "B4_18_1.fcs",
                  package = "cyanoFilter",
              mustWork = TRUE)
flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE,
                               transformation = FALSE, emptyValue = FALSE,
                               dataset = 1)
flowfile_nona <- cyanoFilter::noNA(x = flowfile)
flowfile_noneg <- cyanoFilter::noNeg(x = flowfile_nona)
flowfile_logtrans <- cyanoFilter::lnTrans(x = flowfile_noneg,
c('SSC.W', 'TIME'))
oneDgate(flowfile, 'RED.B.HLin')
```

---

noNA *Removes NA values from the expression matrix of a flow cytometer file.*

Description

Removes NA values from the expression matrix of a flow cytometer file.

Usage

```
noNA(x)
```

Arguments

x	flowframe with expression matrix containing NAs.

Value

flowframe with expression matrix rid of NAs.

Examples

```
flowfile_path <- system.file("extdata", "B4_18_1.fcs",
package = "cyanoFilter",
mustWork = TRUE)
flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE,
                               transformation = FALSE, emptyValue = FALSE,
                               dataset = 1)
noNA(x = flowfile)
```

noNeg *Removes negative values from the expression matrix*

Description

Removes negative values from the expression matrix

Usage

```
noNeg(x)
```

Arguments

x	is the flowframe whose expression matrix contains negative values

Value

flowframe with non-negative values in its expression matrix

Examples

```
flowfile_path <- system.file("extdata", "B4_18_1.fcs",
package = "cyanoFilter",
mustWork = TRUE)
flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE,
                               transformation = FALSE, emptyValue = FALSE,
                               dataset = 1)
flowfile_nona <- cyanoFilter::noNA(x = flowfile)
noNeg(x = flowfile_nona)
```

oneDgate *returns the labels stating the cluster of each row in a flowfile.*

Description

returns the labels stating the cluster of each row in a flowfile.

Usage

```
oneDgate(flowfile, togate)
```

Arguments

flowfile	flowframe after debris are removed.
togate	channels detected to have more than one peak present. Provide by the getChannel function.

Value

list of indicators for cells above and below an estimated threshold

Examples

```
flowfile_path <- system.file("extdata", "B4_18_1.fcs",
                 package = "cyanoFilter",
             mustWork = TRUE)
flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE,
                               transformation = FALSE, emptyValue = FALSE,
                               dataset = 1)
flowfile_nona <- cyanoFilter::noNA(x = flowfile)
flowfile_noneg <- cyanoFilter::noNeg(x = flowfile_nona)
flowfile_logtrans <- cyanoFilter::lnTrans(x = flowfile_noneg,
c('SSC.W', 'TIME'))
oneDgate(flowfile, 'RED.B.HLin')
```

---

pairsPlot *produces a scatter plot of the expression matrix of a flowframe. Note that, it takes some time to display the plot.*

Description

produces a scatter plot of the expression matrix of a flowframe. Note that, it takes some time to display the plot.

Usage

```
pairsPlot(x, notToPlot = c("TIME"), ...)
```

Arguments

x	flowframe to be plotted
notToPlot	column in expression matrix not to be plotted
...	other arguments. Not used at the moment

Value

a plot object

Examples

```
flowfile_path <- system.file("extdata", "B4_18_1.fcs",
             package = "cyanoFilter",
             mustWork = TRUE)
flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE,
                               transformation = FALSE, emptyValue = FALSE,
                               dataset = 1)
flowfile_nona <- cyanoFilter::noNA(x = flowfile)
flowfile_noneg <- cyanoFilter::noNeg(x = flowfile_nona)
flowfile_logtrans <- cyanoFilter::lnTrans(x = flowfile_noneg,
```

```
                        c('SSC.W', 'TIME'))
pairsPlot(flowfile_logtrans,
          notToPlot = c("TIME", "SSC.W",
          "SSC.HLin", "NIR.R.HLin",
          "FSC.HLin"))
```

`phytoFilter` *gates out and assign indicators to phytoplankton cells based on the expression of measured cell complexity channels.*

Description

This function takes in a flowframe with debris removed and identifies the different phytoplankton cell population based on cell pigmentation and/or complexity.

Usage

```
phytoFilter(
  flowfile,
  pig_channels = NULL,
  com_channels = NULL,
  ph = 0.05,
  proportion = 0.8
)
```

Arguments

`flowfile`	flowframe after debris are removed.
`pig_channels`	flowcytometer channels measuring cell pigments.
`com_channels`	flowcytometer channels measuring cell complexity.
`ph`	maximum peak height to be ignored. This allows ignoring of tiny peaks that could affect the gating process.
`proportion`	proportion of cell count to be returned.

Details

The function uses the `getPeaks` and `deGate` functions in the *flowDensity* package to identify peaks and identify cut-off points between these peaks.

Value

object of class PhytopFilter containing;

- **fullflowframe -** flowframe containing all phytoplankton cells with added columns indicating cluster
- **flowframe_proportion -** a part of fullflowframe containing proportion of cell count.
- **clusters_proportion -** proportion of cells in each cluster
- **particles_per_cluster -** number of particles per cluster
- **Cluster_ind -** indicator for each cluster
- **gated_channels -** channels with multiple peaks

Examples

```
 flowfile_path <- system.file("extdata", "B4_18_1.fcs",
package = "cyanoFilter",
              mustWork = TRUE)
flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE,
                               transformation = FALSE,
                               emptyValue = FALSE,
                               dataset = 1)
flowfile_nona <- cyanoFilter::noNA(x = flowfile)
flowfile_noneg <- cyanoFilter::noNeg(x = flowfile_nona)
flowfile_logtrans <- cyanoFilter::lnTrans(x = flowfile_noneg,
                      c('SSC.W', 'TIME'))
cells_nonmargin <- cellMargin(flowframe = flowfile_logtrans,
                              Channel = 'SSC.W',
           type = 'estimate', y_toplot = "FSC.HLin")
cells_nodebris <- debrisNc(flowframe = reducedFlowframe(cells_nonmargin),
                           ch_chlorophyll = "RED.B.HLin",
                           ch_p2 = "YEL.B.HLin",
                           ph = 0.05)
phytoFilter(flowfile = reducedFlowframe(cells_nodebris),
              pig_channels = c("RED.B.HLin", "YEL.B.HLin", "RED.R.HLin"),
              com_channels = c("FSC.HLin", "SSC.HLin"))
```

`PhytopFilter` *the phytofilter class*

Description

the phytofilter class

constructor for the PhytoFilter class

Usage

```
PhytopFilter(
  fullflowframe,
  flowframe_proportion,
  clusters_proportion,
  particles_per_cluster,
  Cluster_ind,
  gated_channels,
  channels
)

PhytopFilter(
  fullflowframe,
  flowframe_proportion,
  clusters_proportion,
  particles_per_cluster,
  Cluster_ind,
```

```
  gated_channels,
  channels
)
```

Arguments

fullflowframe	same as the input flowFrame
flowframe_proportion	a partial flowframe containing containing a proportion of the measured particles
clusters_proportion	number of margin particles measured
particles_per_cluster	number of particles in each cluster
Cluster_ind	labels for each cluster
gated_channels	channels used for gating
channels	all channels supplied

Value

object of class PhytoFilter

Slots

fullflowframe object of class "flowFrame" same as the input flowFrame

flowframe_proportion object of class "flowFrame" a partial flowframe containing a proportion of the measured particles

clusters_proportion object of class "numeric" representing the proportion of particles in each cluster

particles_per_cluster object of class "data.frame" representing the number of particles in each cluster

Cluster_ind object of class "integer" representing the labels for each cluster

gated_channels object of class "character" representing the names of channels with multiple peaks

channels object of class "character" representing the names of the channels

Examples

```
flowfile_path <- system.file("extdata", "B4_18_1.fcs",
                 package = "cyanoFilter",
                 mustWork = TRUE)
flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE,
                               transformation = FALSE, emptyValue = FALSE,
                               dataset = 1)
flowfile_nona <- cyanoFilter::noNA(x = flowfile)
flowfile_noneg <- cyanoFilter::noNeg(x = flowfile_nona)
flowfile_logtrans <- lnTrans(x = flowfile_noneg, c('SSC.W', 'TIME'))
cellMargin(flowframe = flowfile_logtrans, Channel = 'SSC.W',
           type = 'estimate', y_toplot = "FSC.HLin")

flowfile_path <- system.file("extdata", "B4_18_1.fcs",
                 package = "cyanoFilter",
```

```
                mustWork = TRUE)
flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE,
                               transformation = FALSE, emptyValue = FALSE,
                               dataset = 1)
flowfile_nona <- cyanoFilter::noNA(x = flowfile)
flowfile_noneg <- cyanoFilter::noNeg(x = flowfile_nona)
flowfile_logtrans <- lnTrans(x = flowfile_noneg, c('SSC.W', 'TIME'))
cellMargin(flowframe = flowfile_logtrans, Channel = 'SSC.W',
           type = 'estimate', y_toplot = "FSC.HLin")
```

`pigmentGate` *gates out or assign indicators to phytoplankton cells based on the expression of the measured pigments.*

Description

This function takes in a flowframe with debris removed and identifies phytoplankton cell population in the provided frame.

Usage

```
pigmentGate(flowfile, pig_channels, ph = 0.05)
```

Arguments

`flowfile`	flowframe after debris are removed.
`pig_channels`	flowcytometer channels measuring phytoplankton pigmentations.
`ph`	maximum peak height to be ignored. This allows ignoring of tiny peaks that could affect the gating process.

Details

The function uses the `getPeaks` and `deGate` functions in the *flowDensity* package to identify peaks and identify cut-off points between these peaks.

Value

list containing;

- **full_flowframe -** flowframe containing only phytoplankton cells
- **phy_ind -** indicator for phytoplankton clusters found
- **gated_channels -** pigment channels with more than one peak

Examples

```
flowfile_path <- system.file("extdata", "B4_18_1.fcs",
                  package = "cyanoFilter",
              mustWork = TRUE)
flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE,
                               transformation = FALSE, emptyValue = FALSE,
                               dataset = 1)
flowfile_nona <- cyanoFilter::noNA(x = flowfile)
```

```
flowfile_noneg <- cyanoFilter::noNeg(x = flowfile_nona)
flowfile_logtrans <- cyanoFilter::lnTrans(x = flowfile_noneg,
c('SSC.W', 'TIME'))
cyanoFilter::pigmentGate(flowfile = flowfile_logtrans,
pig_channels = c("RED.B.HLin", "YEL.B.HLin",
                    "FSC.HLin", "RED.R.HLin"),
ph = 0.06)
```

---

plot,DebrisFilter,ANY-method

plot method for DebrisFilter objects

Description

plot method for DebrisFilter objects

Usage

```
## S4 method for signature 'DebrisFilter,ANY'
plot(x)
```

Arguments

x	an object of class DebrisFilter

Value

object of class ggplot

---

plot,MarginEvents,ANY-method

plot method for MarginEvents objects

Description

plot method for MarginEvents objects

Usage

```
## S4 method for signature 'MarginEvents,ANY'
plot(x)
```

Arguments

x	an object of class MarginEvents

Value

object of class ggplot

plot,PhytopFilter,ANY-method *plot method for PhytoFilter objects*

Description

plot method for PhytoFilter objects

Usage

```
## S4 method for signature 'PhytopFilter,ANY'
plot(x)
```

Arguments

x	an object of class PhytoFilter

Value

object of class ggplot

reducedFlowframe *generic function for extracting the full flowframe*

Description

generic function for extracting the full flowframe

Usage

```
reducedFlowframe(x)
```

Arguments

x	an object of either class PhytoFilter, MarginEvents or DebrisFilter

Value

generic to extract fullFlowframe

Examples

```
flowfile_path <- system.file("extdata", "B4_18_1.fcs",
package = "cyanoFilter",
              mustWork = TRUE)
flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE,
                               transformation = FALSE,
                               emptyValue = FALSE,
                               dataset = 1)
flowfile_nona <- cyanoFilter::noNA(x = flowfile)
flowfile_noneg <- cyanoFilter::noNeg(x = flowfile_nona)
```

```
flowfile_logtrans <- cyanoFilter::lnTrans(x = flowfile_noneg,
                      c('SSC.W', 'TIME'))
cells_nonmargin <- cellMargin(flowframe = flowfile_logtrans,
                              Channel = 'SSC.W',
           type = 'estimate', y_toplot = "FSC.HLin")
 reducedFlowframe(cells_nonmargin)
```

`reducedFlowframe,DebrisFilter-method`
accesor method for reduced flowframe (DebrisFilter class)

Description

accesor method for reduced flowframe (DebrisFilter class)

Usage

```
## S4 method for signature 'DebrisFilter'
reducedFlowframe(x)
```

Arguments

x	an object of class DebrisFilter

Value

reduced flowFrame method for DebrisFilter

`reducedFlowframe,MarginEvents-method`
accesor method for reduced flowframe (MarginEvent class)

Description

accesor method for reduced flowframe (MarginEvent class)

Usage

```
## S4 method for signature 'MarginEvents'
reducedFlowframe(x)
```

Arguments

x	an object of class MarginEvents

Value

reduced Flowframe method for MarginEvents

Examples

```
flowfile_path <- system.file("extdata", "B4_18_1.fcs",
                  package = "cyanoFilter",
              mustWork = TRUE)
flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE,
                               transformation = FALSE, emptyValue = FALSE,
                               dataset = 1)
flowfile_nona <- cyanoFilter::noNA(x = flowfile)
flowfile_noneg <- cyanoFilter::noNeg(x = flowfile_nona)
flowfile_logtrans <- cyanoFilter::lnTrans(x = flowfile_noneg,
c('SSC.W', 'TIME'))
cells_nonmargin <- cellMargin(flowframe = flowfile_logtrans,
Channel = 'SSC.W',
           type = 'estimate', y_toplot = "FSC.HLin")
reducedFlowframe(cells_nonmargin)
```

---

reducedFlowframe,PhytopFilter-method

accesor method for reduced flowframe(PhytoFilter class)

Description

accesor method for reduced flowframe(PhytoFilter class)

Usage

```
## S4 method for signature 'PhytopFilter'
reducedFlowframe(x)
```

Arguments

x	an object of class PhytoFilter

Value

reduced flowFrame method for PhytoFilter #' @examples flowfile_path <- system.file("extdata", "B4_18_1.fcs", package = "cyanoFilter", mustWork = TRUE) flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE, transformation = FALSE, emptyValue = FALSE, dataset = 1) flowfile_nona <- cyanoFilter::noNA(x = flowfile) flowfile_noneg <- cyanoFilter::noNeg(x = flowfile_nona) flowfile_logtrans <- cyanoFilter::lnTrans(x = flowfile_noneg, c('SSC.W', 'TIME')) cells_nonmargin <- cellMargin(flowframe = flowfile_logtrans, Channel = 'SSC.W', type = 'estimate', y_toplot = "FSC.HLin") cells_nodebris <- debrisNc(flowframe = reducedFlowframe(cells_nonmargin), ch_chlorophyll = "RED.B.HLin", ch_p2 = "YEL.B.HLin", ph = 0.05) phy1 <- phytoFilter(flowfile = reducedFlowframe(cells_nodebris), pig_channels = c("RED.B.HLin", "YEL.B.HLin", "RED.R.HLin"), com_channels = c("FSC.HLin", "SSC.HLin")) reducedFlowframe(phy1)

`retain` *Decides if a file should be retiained or removed based on its status.*

Description

Function to determine what files to retain and finally read from the flow cytometer FCS file.

Usage

```
retain(
  meta_files,
  make_decision = c("maxi", "mini", "unique"),
  Status = "Status",
  CellspML = "CellspML"
)
```

Arguments

`meta_files`	dataframe from meta file that has been preprocessed by the `goodFcs` function.
`make_decision`	decision to be made should more than one $cells/\mu L$ be good.
`Status`	column name in meta_files containing status obtained from the `goodFcs` function.
`CellspML`	column name in meta_files containing $cells/\mu L$ measurements.

Details

It is typically not known in advance which dilution level would result in the desired $cells/\mu L$, therefore the samples are ran through the flow cytometer at two or more dilution levels. Out of these, one has to decide which to retain and finally use for further analysis. This function and `goodFcs` are to help you decide that. If more than one of the dilution levels are judged good, the option *make_decision = "maxi"* will give "Retain" to the row with the maximum $cells/\mu L$ while the opposite occurs for *make_decision = "mini"*. *make_decision = "unique"* i there is only one measurement for that particular sample, while *make_decision = "maxi"* and *make_decision = "mini"* should be used for files with more than one measurement for the sample in question.

Value

a character vector with entries "Retain" for a file to be retained or "No!" for a file to be discarded.

See Also

`goodFcs`

Examples

```
require("stringr")
metadata <- system.file("extdata", "2019-03-25_Rstarted.csv",
package = "cyanoFilter",
mustWork = TRUE)
metafile <- read.csv(metadata, skip = 7, stringsAsFactors = FALSE,
                     check.names = TRUE, encoding = "UTF-8")
```

```
metafile <- metafile[, seq_len(65)] #first 65 columns contain useful information
#extract the part of the Sample.ID that corresponds to BS4 or BS5
metafile$Sample.ID2 <- stringr::str_extract(metafile$Sample.ID, "BS*[4-5]")
#clean up the Cells.muL column
names(metafile)[which(stringr::str_detect(names(metafile), "Cells."))] <-
"CellspML"
metafile$Status <- cyanoFilter::goodFcs(metafile = metafile, col_cpml =
"CellspML",
mxd_cellpML = 1000, mnd_cellpML = 50)
metafile$Retained <- NULL
# first 3 rows contain BS4 measurements at 3 dilution levels
metafile$Retained[seq_len(3)] <-
 cyanoFilter::retain(meta_files = metafile[seq_len(3),],
make_decision = "maxi",
Status = "Status", CellspML = "CellspML")
# last 3 rows contain BS5 measurements at 3 dilution levels as well
metafile$Retained[seq(4, 6, by = 1)] <-
  cyanoFilter::retain(meta_files = metafile[seq(4, 6, by = 1),],
make_decision = "maxi",
Status = "Status", CellspML = "CellspML")
```

---

rowNumbers *returns the position of the cells below, above or between estimated gates*

Description

returns the position of the cells below, above or between estimated gates

Usage

```
rowNumbers(flowframe, gates, ch)
```

Arguments

flowframe	after debris are removed.
gates	cut point between the identified clusters
ch	gated channel

Value

a numeric vector

Examples

```
flowfile_path <- system.file("extdata", "B4_18_1.fcs",
                  package = "cyanoFilter",
              mustWork = TRUE)
flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE,
                               transformation = FALSE, emptyValue = FALSE,
                               dataset = 1)
```

```
flowfile_nona <- cyanoFilter::noNA(x = flowfile)
flowfile_noneg <- cyanoFilter::noNeg(x = flowfile_nona)
flowfile_logtrans <- cyanoFilter::lnTrans(x = flowfile_noneg,
c('SSC.W', 'TIME'))
oneDgate(flowfile, 'RED.B.HLin')
```

`summaries` *takes a flowframes, a vector of channels, cluster indicator and return desired summaries per cluster*

Description

takes a flowframes, a vector of channels, cluster indicator and return desired summaries per cluster

Usage

```
summaries(object, channels, cluster_var, summary)
```

Arguments

`object`	An object of class cyanoFilter to be summarised.
`channels`	channels whose summaries are to be computed
`cluster_var`	column name in expression matrix containing the cluter indicators
`summary`	summary statistic of interest. Only mean and variance-covariance matrix supported at the moment.

Value

list containing computed summaires

Examples

```
flowfile_path <- system.file("extdata", "B4_18_1.fcs",
              package = "cyanoFilter",
              mustWork = TRUE)
flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE,
                               transformation = FALSE, emptyValue = FALSE,
                               dataset = 1)
flowfile_nona <- cyanoFilter::noNA(x = flowfile)
flowfile_noneg <- cyanoFilter::noNeg(x = flowfile_nona)
flowfile_logtrans <- cyanoFilter::lnTrans(x = flowfile_noneg,
c('SSC.W', 'TIME'))
cells_nonmargin <- cellMargin(flowframe = flowfile_logtrans,
Channel = 'SSC.W',
           type = 'estimate', y_toplot = "FSC.HLin")
cells_nodebris <- debrisNc(flowframe = reducedFlowframe(cells_nonmargin),
                           ch_chlorophyll = "RED.B.HLin",
                            ch_p2 = "YEL.B.HLin",
                            ph = 0.05)
```

summaries,DebrisFilter-method
takes a flowframes, a vector of channels, cluster indicator and return desired summaries per cluster

Description

takes a flowframes, a vector of channels, cluster indicator and return desired summaries per cluster

Usage

```
## S4 method for signature 'DebrisFilter'
summaries(object, channels = NULL)
```

Arguments

object	An object of class MarginEvents to be summarised.
channels	channels whose summaries are to be computed

Value

list containing the required summaries

Examples

```
flowfile_path <- system.file("extdata", "B4_18_1.fcs",
              package = "cyanoFilter",
              mustWork = TRUE)
flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE,
                               transformation = FALSE, emptyValue = FALSE,
                               dataset = 1)
flowfile_nona <- cyanoFilter::noNA(x = flowfile)
flowfile_noneg <- cyanoFilter::noNeg(x = flowfile_nona)
flowfile_logtrans <- cyanoFilter::lnTrans(x = flowfile_noneg,
c('SSC.W', 'TIME'))
cells_nonmargin <- cellMargin(flowframe = flowfile_logtrans,
Channel = 'SSC.W',
           type = 'estimate', y_toplot = "FSC.HLin")
summaries(cells_nonmargin,
c("RED.B.HLin", "YEL.B.HLin", "RED.R.HLin"))
```

summaries,MarginEvents-method
takes a flowframes, a vector of channels, cluster indicator and return desired summaries per cluster

Description

takes a flowframes, a vector of channels, cluster indicator and return desired summaries per cluster

Usage

```
## S4 method for signature 'MarginEvents'
summaries(object, channels = NULL)
```

Arguments

object	An object of class MarginEvents to be summarised.
channels	channels whose summaries are to be computed

Value

list containing the required summaries

Examples

```
flowfile_path <- system.file("extdata", "B4_18_1.fcs",
              package = "cyanoFilter",
              mustWork = TRUE)
flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE,
                               transformation = FALSE, emptyValue = FALSE,
                               dataset = 1)
flowfile_nona <- cyanoFilter::noNA(x = flowfile)
flowfile_noneg <- cyanoFilter::noNeg(x = flowfile_nona)
flowfile_logtrans <- cyanoFilter::lnTrans(x = flowfile_noneg,
c('SSC.W', 'TIME'))
cells_nonmargin <- cellMargin(flowframe = flowfile_logtrans,
Channel = 'SSC.W',
           type = 'estimate', y_toplot = "FSC.HLin")
summaries(cells_nonmargin,
c("RED.B.HLin", "YEL.B.HLin", "RED.R.HLin"))
```

---

summaries,PhytopFilter-method

takes a flowframes, a vector of channels, cluster indicator and return desired summaries per cluster

---

Description

takes a flowframes, a vector of channels, cluster indicator and return desired summaries per cluster

Usage

```
## S4 method for signature 'PhytopFilter'
summaries(
  object,
  channels = NULL,
  cluster_var = "Clusters",
  summary = c("mean", "median", "cov", "n")
)
```

Arguments

object	An object of class cyanoFilter to be summarised.
channels	channels whose summaries are to be computed
cluster_var	column name in expression matrix containing the cluter indicators
summary	summary statistic of interest. Only mean and variance-covariance matrix supported at the moment.

Value

list containing computed summaires

Examples

```
flowfile_path <- system.file("extdata", "B4_18_1.fcs",
              package = "cyanoFilter",
              mustWork = TRUE)
flowfile <- flowCore::read.FCS(flowfile_path, alter.names = TRUE,
                               transformation = FALSE, emptyValue = FALSE,
                               dataset = 1)
flowfile_nona <- cyanoFilter::noNA(x = flowfile)
flowfile_noneg <- cyanoFilter::noNeg(x = flowfile_nona)
flowfile_logtrans <- cyanoFilter::lnTrans(x = flowfile_noneg,
c('SSC.W', 'TIME'))
cells_nonmargin <- cellMargin(flowframe = flowfile_logtrans,
Channel = 'SSC.W',
           type = 'estimate', y_toplot = "FSC.HLin")
cells_nodebris <- debrisNc(flowframe = reducedFlowframe(cells_nonmargin),
                           ch_chlorophyll = "RED.B.HLin",
                            ch_p2 = "YEL.B.HLin",
                            ph = 0.05)
fin <- phytoFilter(flowfile = reducedFlowframe(cells_nodebris),
              pig_channels = c("RED.B.HLin", "YEL.B.HLin", "RED.R.HLin"),
              com_channels = c("FSC.HLin", "SSC.HLin"))

summaries(object = fin,
        channels = c("RED.B.HLin", "YEL.B.HLin", "RED.R.HLin"),
        cluster_var = "Clusters",
        summary = 'mean')
```

Index