Package 'flowTime'

October 16, 2019

Title Annotation and analysis of biological dynamical systems using flow cytometry

Version 1.8.0

Description This package was developed for analysis of both dynamic and steady state experiments examining the function of gene regulatory networks in yeast (strain W303) expressing fluorescent reporter proteins using a BD Accuri C6 and SORP cytometers. However, the functions are for the most part general and may be adapted for analysis of other organisms using other flow cytometers. Functions in this package facilitate the annotation of flow cytometry data with experimental metadata, as is requisite for dissemination and general ease-of-use. Functions for creating, saving and loading gate sets are also included. In the past, we have typically generated summary statistics for each flowset for each timepoint and then annotated and analyzed these summary statistics. This method loses a great deal of the power that comes from the large amounts of individual cell data generated in flow cytometry, by essentially collapsing this data into a bulk measurement after subsetting. In addition to these summary functions, this package also contains functions to facilitate annotation and analysis of steady-state or time-lapse data utilizing all of the data collected from the thousands of individual cells in each sample.

Depends R (>= 3.4), flowCore, plyr

License Artistic-2.0

LazyData true

biocViews FlowCytometry, TimeCourse, Visualization, DataImport, CellBasedAssays, ImmunoOncology

Suggests knitr, rmarkdown, flowViz, ggplot2, BiocGenerics, moments, stats

Imports utils

VignetteBuilder knitr

RoxygenNote 6.0.1

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

git_branch RELEASE_3_9

git_last_commit dee638a

git_last_commit_date 2019-05-02

Date/Publication 2019-10-15

Author R. Clay Wright [aut, cre],
Nick Bolten [aut],
Edith Pierre-Jerome [aut]

Maintainer R. Clay Wright <wright.clay@gmail.com>

R topics documented:

---

addbs *Add background subtraction to a summary data frame*

Description

Subtracts the background fluorescence of a given control strain from the chosen column.

Usage

```
addbs(flowData, column = "FL3.Amean", baseline_column = "strain",
  baseline = "noYFP")
```

Arguments

flowData	the summary data frame of flowSet to be background subtracted
column	the column containing the fluorescent measurement to be background subtracted
baseline_column	the column containing the name of the strain representing background fluorescent values
baseline	character the name of the strain representing background fluorescent values

Value

A summary data frame with an additional column "column_bs" containing the background subtracted fluorescent values

Examples

```
dat<-read.flowSet(path=system.file("extdata", "tc_example",
package = "flowTime"),alter.names = TRUE)
annotation <- read.csv(system.file("extdata", "tc_example.csv",
package = "flowTime"))
annotation[which(annotation$treatment == 0), 'strain'] <- 'background'
adat <- annotateFlowSet(dat, annotation)
loadGates(gatesFile = 'C6Gates')
dat_sum <- summarizeFlow(adat, ploidy = 'diploid', only = 'singlets',
channel = 'FL1.A')
dat_sum <- addbs(dat_sum, column = "FL1.Amean", baseline = "background")
```

---

addnorm *Normalize fluorescence*

Description

Produces a normalized fluorescence column 'normed'. Expects the 'FL1.A_bs' column to exist or a column to be specified. Has two different methods, version 1 and version 2, described in the script

Usage

```
addnorm(frame, factor_in = c("strain", "treatment"), method = 1,
  column = "FL3.Amean_bs")
```

Arguments

frame	data frame of summary statistics to be normalized
factor_in	character vector containing the varibles to split the data frame by
method	which normalization method to use, 1, 2 or 3.
column	character the column to apply the normalization to

Details

Method 1, the default normalization method, takes the highest point in each dataset grouped by 'factor_in' and normalizes all values in the group by this point. This method is default because it works regardless of whether the data is a time series. Method 2 finds the mean value of all time points with time values less than 0 for each group and normalizes each group by this respective value. Requires a time series with negative time values to work. Version 3 fits a linear model to the pre-zero time points for each groups, infers the y-intercept, and normalizes using this intercept. Method 3 also requires a time series with negative time values to work.

Value

`data frame` containing the additional normalized variable

Examples

```
dat <- read.flowSet(path=system.file("extdata", "tc_example",
package = "flowTime"), alter.names = TRUE)
annotation <- read.csv(system.file("extdata", "tc_example.csv",
package = "flowTime"))
adat <- annotateFlowSet(dat, annotation)
loadGates(gatesFile = 'C6Gates')
dat_sum <- summarizeFlow(adat, ploidy = "diploid", only = "singlets",
channel = "FL1.A")
dat_sum <- addnorm(dat_sum, c("strain", "treatment"), method = 1,
column = "FL1.Amean")
```

`annotateFlowSet` *Annotate a flowSet with experimental metadata*

Description

Add annotations to a flowSets phenoData and plate numbers, strain names, and treatment also set T0

Usage

```
annotateFlowSet(yourFlowSet, annotation_df, mergeBy = "name")
```

Arguments

`yourFlowSet`	a flowSet with sampleNames of the format 'plate#_Well', we typically use the following code chunk to read data from individual plates as exported from BD Accuri C6 software.
`annotation_df`	A data frame with columns 'well', 'strain', 'treatment', containing all of the wells in the flowset labeled with the strain and treatment in that well.
`mergeBy`	the unique identifier column

Value

An annotated flowSet

Examples

```
dat <- read.flowSet(path = system.file("extdata", "ss_example",
package = "flowTime"), alter.names = TRUE)
annotation <- read.csv(system.file("extdata", "ss_example.csv", package =
"flowTime"))
annotateFlowSet(dat, annotation, mergeBy = "name")
```

---

createAnnotation *Create an annotation dataframe*

Description

Creates a data frame with rows containing the sample names of your flow set that can then be filled in with experimental metadata.

Usage

```
createAnnotation(yourFlowSet)
```

Arguments

yourFlowSet	the flowSet to create an annotation data frame for

Value

annotation_df a data frame containing the sample names of your flow set

Examples

```
dat <- read.flowSet(path = system.file("extdata", "ss_example",
package = "flowTime"), alter.names = TRUE)
annotation <- createAnnotation(yourFlowSet = dat)
head(annotation)
```

---

dipdoubletGate *A gate for the set of all diploid doublets*

Description

A gate for the set of all diploid doublets

Usage

```
data(dipdoubletGate)
```

Format

formal class polygonGate

dipsingletGate — *A gate for the set of all diploid singlet yeast cells*

Description

Typically set in FSC.A by FSC.H space Diploids are typically 5um x 6um ellipsoids while haploids are typically 4um x 4um spheroids. As a result, diploids are longer and you get a larger 'area/volume'.

Usage

```
data(dipsingletGate)
```

Format

formal class polygonGate

flsummary — *Get summary statistics for fluorescence or other data channels of a flowSet*

Description

Get summary statistics for fluorescence or other data channels of a flowSet

Usage

```
flsummary(flowset, channel = "FL3.A", moments = FALSE)
```

Arguments

flowset	the flowSet to create summary statistics for
channel	character the data channel to summarize
moments	boolean if TRUE then split each frame into early, middle, and late events

Value

A data frame containing summary statistics (mean, median, SD) for the specified fluorescent channel and time moments of the flowSet.

Examples

```
plate1 <- read.flowSet(path = system.file("extdata",
"ss_example", package = "flowTime"), alter.names = TRUE)
flsummary(plate1)
```

getTime *Get the time at which at flowFrame began collection*

Description

Get the time at which at flowFrame began collection

Usage

```
getTime(flowframe)
```

Arguments

flowframe	The flowFrame for which you would like the initial time

Value

numeric time value in minutes

Examples

```
plate1<-read.flowSet(path = system.file("extdata", "ss_example", package =
"flowTime"),alter.names = TRUE)
getTime(plate1$A01.fcs)
```

hapdoubletGate *A gate for the set of all haploid doublets*

Description

A gate for the set of all haploid doublets

Usage

```
data(hapdoubletGate)
```

Format

formal class polygonGate

hapsingletGate *A gate for the set of all haploid singlets*

Description

A gate for the set of all haploid singlets

Usage

```
data(hapsingletGate)
```

Format

formal class polygonGate

loadGates *Load a yeast gate file*

Description

Loads a set of yeast gates into active memory to be used in analysis functions

Usage

```
loadGates(gatesFile = NULL, path = NULL, envir = environment())
```

Arguments

gatesFile	the gates file to be loaded into memory, or path to the gates file
path	The path to the gates file. If 'NULL' this will look through lazy loaded data for the gatesFile
envir	The environment in which to load the gates

Value

gate objects created in the current environment

Examples

```
loadGates(system.file("extdata/SORPGates.RData", package = "flowTime"))
```

ploidy *Guess the ploidy of a given flowframe*

Description

Use the FSC.A/FSC.H ratio. Diploids are typically 5um x 6um ellipsoids while haploids are typically 4um x 4um spheroids. As a result, diploids are longer and you get a larger 'area/volume' FSC.A. 'Width' might also be useful on certain cytometers.

Usage

```
ploidy(flowframe)
```

Arguments

flowframe	the flowFrame you would like to identify the ploidy of

Value

"Diploid" or "Haploid" and the mean FSC.A/FSC.H quotient

Examples

```
dat <- read.flowSet(path = system.file("extdata", "ss_example",
package = "flowTime"), alter.names = TRUE)
ploidy(dat$A01.fcs)
```

polyGate *Create a polygon gate*

Description

Create a polygon gate

Usage

```
polyGate(x, y, filterID = "newGate", channels = c("FSC.A", "FSC.H"))
```

Arguments

x	a vector of x coordinates
y	a vector of y coordinates
filterID	name of the gate
channels	vector containing the channels matching the x and y coordinates above

Value

a polygon gate object

Examples

```
polyGate(x = c(1,1,10000,10000), y = c(1,10000, 10000, 1), )
```

---

qaGating *Quality assurance check*

Description

Check whether a flowSet (or a single flowFrame) contains empty values, in which case normalization may fail (divide by zero). This is particularly useful for removing wash wells from a flowSet.

Usage

```
qaGating(x, threshold = 100)
```

Arguments

x	flowSet or flowFrame to be checked
threshold	flowFrames with fewer events than this threshold will be identified.

Value

A vector containing the flowFrames with fewer events than the threshold.

Examples

```
plate1<-read.flowSet(path = system.file("extdata", "ss_example", package =
"flowTime"), alter.names = TRUE)
qaGating(plate1)
```

---

read.plateSet *Read FCS files from set of plates*

Description

Reads all folders within the specified path containing the specified pattern in the folder names. Each folder contains a set a plate of FCS files. These folders typically make up a whole experiment. Plates are numbered according to the standard lexicographical ordering of your operating system.

Usage

```
read.plateSet(path = getwd(), pattern = "", ...)
```

Arguments

path	The path to search for folders containing FCS files
pattern	The regex pattern used to identify the folders of FCS files to be read
...	Additional arguments passed to read.flowSet. Note that ‘alter.names‘ is forced to be TRUE in this implementation.

Value

A single flowSet containing all FCS files within the identified folders. The index of each folder in the list according to lexicographical ordering (1,2,...) is prepended to the sampleNames.

Examples

```
# Read in both of the example data sets as a single flowSet
plate1<-read.plateSet(path = system.file("extdata", package = "flowTime"),
pattern = "")
```

`saveGates` *Save a yeast gate set*

Description

Save a yeast gate set

Usage

```
saveGates(yeastGate = "yeastGate", dipsingletGate = "dipsingletGate",
  dipdoubletGate = "dipdoubletGate", hapsingletGate = "hapsingletGate",
  hapdoubletGate = "hapdoubletGate", path = getwd(),
  fileName = "defaultGates.RData")
```

Arguments

`yeastGate`	a gate object defining the population of yeast cells
`dipsingletGate`	a gate object defining the population of diploid singlet cells
`dipdoubletGate`	a gate object defining the population of diploid doublet cells
`hapsingletGate`	a gate object defining the population of haploid singlet cells
`hapdoubletGate`	a gate object defining the population of haploid doublet cells
`path`	path to the folder in which you would like to save the gates
`fileName`	name of the .Rdata file you would like to save these gates within

Value

a .RData file in the "extdata" folder of the package containing the specified gates

Examples

```
loadGates("SORPGates")
saveGates()
```

steadyState *Analysis of steady state fluorescence flow cytometry*

Description

Generates a data frame which can be used to visualize and analyze steady state flow cytometry data. Steady state in this case means that

Usage

```
steadyState(flowset, gated = FALSE, ploidy = "diploid", only = "singlets")
```

Arguments

flowset	your flowSet to be analyzed
gated	boolean is the data already gated?
ploidy	character gate to subset your flowset based on the ploidy of you strains
only	character which population of events to analyze, 'yeast', singlets', or 'doublets'?

Value

a data frame containing all of the selected subset of events from the original flowSet

Examples

```
dat <- read.flowSet(path = system.file("extdata", "ss_example",
package = "flowTime"), alter.names = TRUE)
annotation <- read.csv(system.file("extdata", "ss_example.csv",
package = "flowTime"))
dat <- annotateFlowSet(dat, annotation, mergeBy = "name")
loadGates(gatesFile = 'SORPGates')
steadyState(dat, gated = FALSE, ploidy = "diploid", only = "singlets")
```

summarizeFlow *Generate summary statistics for a flowSet*

Description

Gates a sample to all yeast, then singlet, then doublets. Also calculates singlet to doublet ratio. Returns a list of data frames, e.g. output$singlets, output$doublets, etc.

Usage

```
summarizeFlow(flowset, channel = "FL1.A", gated = FALSE, ploidy = FALSE,
  moments = FALSE, only = FALSE)
```

Arguments

flowset	the flowSet to be summarized
channel	character which data channel should be summarized
gated	boolean is the data already appropriately gated?
ploidy	character does the flowSet contain haploid or diploid cells?
moments	boolean split the data into early, middle, and late moments?
only	character summarize only "singlet", "doublet", or all "yeast" cells, FALSE will return all

Value

data frame containing the specified summary statistics of the specified cell populations for each frame

Examples

```
plate1 <- read.flowSet(path = system.file("extdata", "ss_example",
package = "flowTime"), alter.names = TRUE)
summarizeFlow(plate1, channel = "FL1.A", gated = TRUE,
ploidy = "diploid", moments = FALSE, only = "yeast")
```

---

yeastGate *A gate for the set of all yeast cells*

Description

Typically set in FSC.A by SSC.A space to exclued any debris

Usage

```
data(yeastGate)
```

Format

formal class polygonGate

Index