Package ‘ggcyto’

November 21, 2016

Type Package

Title Visualize Cytometry data with ggplot

Version 1.2.0

Date 2015-11-02

Author Mike Jiang

Maintainer Mike Jiang <wjiang2@fhcrc.org>

Description With the dedicated fority method implemented for flowSet, ncdfflowSet and GatingSet classes, both raw and gated flow cytometry data can be plotted directly with ggplot. ggceto wrapper and some customized layers also make it easy to add gates and population statistics to the plot.

VignetteBuilder knitr

Depends methods, ggplot2(>= 2.0.0), flowCore, ncdfflow(>= 2.17.1), flowWorkspace(>= 3.17.24)

Imports plyr, scales, data.table, RColorBrewer, gridExtra

Suggests testthat, flowWorkspaceData, knitr, rmarkdown, flowStats, openCyto, flowViz

License Artistic-2.0

URL https://github.com/RGLab/ggcyto/issues

biocViews FlowCytometry, CellBasedAssays, Infrastructure, Visualization


RoxygenNote 5.0.1

NeedsCompilation no
R topics documented:

+ggcyto_flowSet ................................................. 2
+ggcyto_GatingLayout .......................................... 3
+ggcyto_GatingSet ............................................. 4
as.ggplot ......................................................... 5
autoplot.flowSet ................................................. 5
axis_x_inverse_trans .......................................... 7
compute_stats .................................................... 8
fortify.ellipsoidGate ........................................... 9
fortify.filterList ............................................... 9
fortify.flowFrame ............................................... 10
fortify.polygonGate ............................................ 11
fortify.rectangleGate .......................................... 12
fortify_fs ......................................................... 12
gem_gate ......................................................... 13
gem_hvline ....................................................... 15
gem_overlay ....................................................... 16
gem_stats ......................................................... 17
getFlowFrame ..................................................... 18
ggcyto.flowSet .................................................. 19
ggcyto.GatingSet ............................................... 20
ggcyto_arrange .................................................. 21
ggcyto_par_default ............................................. 21
ggcyto_par_set .................................................. 22
is.ggcyto ......................................................... 23
is.ggcyto_flowSet .............................................. 23
is.ggcyto_par ..................................................... 24
labs_cyto ......................................................... 24
marginalFilter .................................................... 25
print.ggcyto ....................................................... 26
print.ggcyto_GatingLayout .................................... 27
scale_x_flowJo_biexp .......................................... 27
scale_x_flowJo_fasinh .......................................... 28
scale_x_logicle .................................................. 29
stat_position ..................................................... 29

Index 31

+ggcyto_flowSet  overloaded `+` method for ggcyto

Description

It tries to copy pData from ggcyto object to the gate layers so that the gate layer does not need to have ‘pd’ to be supplied explicitly by users. It also calculates population statistics when geom_stats layer is added. It supports addition ggcyto layers such as ’ggcyto_par’ and ’labs_cyto’.

Usage

## S3 method for class 'ggcyto_flowSet'
e1 + e2

## S4 method for signature 'ggcyto_flowSet,ANY'
e1 + e2

Arguments

e1 An object of class ggcyto_flowSet
e2 A component to add to e1

Value

ggcyto_flowSet object

Examples

data(GvHD)
fs <- GvHD[subset(pData(GvHD), Patient %in% 5:7 & Visit %in% c(5:6))["name"]]
p <- ggcyto(fs, aes(x = "FSC-H", y = "SSC-H")) + geom_hex(bins = 128)
#add rectangleGate layer (2d)
rect.g <- rectangleGate(list("FSC-H" = c(300,500), "SSC-H" = c(50,200)))
rect.gates <- sapply(sampleNames(fs), function(sn)rect.g)
p + geom_gate(rect.gates) + geom_stats()

+ggcyto_GatingLayout overloaded '+' method for ggcyto_gate_layout

Description

It adds the layer specified by 'e2' to each individual ggplot object stored in ggcyto_gate_layout

Usage

## S3 method for class 'ggcyto_GatingLayout'
e1 + e2

## S4 method for signature 'ggcyto_GatingLayout,ANY'
e1 + e2

Arguments

e1 ggcyto_gate_layout
e2 any ggplot layer

Value

a modified ggcyto_gate_layout object
a GatingLayout object
Examples

#autplot for GatingSet
dataDir <- system.file("extdata",package="flowWorkspaceData")
gs <- load_gs(list.files(dataDir, pattern = "gs_manual",full = TRUE))
gh <- gs[[1]]
p <- autoplot(gh)
class(p)
# customize the font size of strip text for each ggcyo plots contained in GatingLayout object
p + theme(strip.text = element_text(size = 14))

Description

It takes care the speical format of some ggcyto layers. For example geom_gate or geom_stats layer
with just gate(population) name specified. It only supports some special axis transformations. (See
examples below)

Usage

## S3 method for class 'ggcyto_GatingSet'
e1 + e2

## S4 method for signature 'ggcyto_GatingSet,ANY'
e1 + e2

Arguments

e1 An object of class ggcyto
e2 A component to add to e1

Value

ggcyto_GatingSet object

Examples

dataDir <- system.file("extdata",package="flowWorkspaceData")
gs <- load_gs(list.files(dataDir, pattern = "gs_manual",full = TRUE))
p <- ggcyto(gs, aes(x = CD4, y = CD8), subset = "CD3+") + geom_hex(bins = 64)
p <- p + geom_gate("CD4") + geom_stats() #plot CD4 gate and it is stats
p
p + axis_x_reverse_trans() #inverse transform the x axis into raw scale
as.ggplot

It fortifies the data, fills some default settings and returns a regular ggplot object.

Description

The original data format is preserved during the ggcyto constructor because they still need to be used during the plot building process. This function is usually called automatically in the print/plot method of ggcyto. Sometimes it is useful to coerce it to ggplot explicitly by user so that it can be used as a regular ggplot object.

Usage

as.ggplot(x)

Arguments

x
ggcyto object with the data that has not yet been fortified to data.frame.

Value

ggplot object

Examples

data(GvHD)
fs <- GvHD[1:3]
# construct the `ggcyto` object (inherits from `ggplot` class)
p <- ggcyto(fs, aes(x = `FSC-H`)) + geom_histogram()
class(p) # a ggcyto object
p$data # data has not been fortified
p1 <- as.ggplot(p) # convert it to a ggplot object explicitly
class(p1)
p1$data # data is fortified

autoplot.flowSet

Plot fluorescence intensity in one or two dimension.

Description

Overloaded autoplot for the cytometry data structure: flowFrame or flowSet, Gatinghierarchy, GatingSet. It plots the cytometry data with geom_histogram, geom_density or geom_hex.
Usage

```r
## S3 method for class 'flowSet'
autoplot(object, x, y = NULL, bins = 30, ...)

## S3 method for class 'ncdfFlowList'
autoplot(object, ...)

## S3 method for class 'flowFrame'
autoplot(object, x, ...)

## S3 method for class 'GatingSetList'
autoplot(object, ...)

## S3 method for class 'GatingSet'
autoplot(object, gate, x = NULL, y = "SSC-A", bins = 30, ...)

## S3 method for class 'GatingHierarchy'
autoplot(object, gate, y = "SSC-A", bool = FALSE, arrange.main = sampleNames(object), arrange = TRUE, merge = TRUE, projections = list(), strip.text = c("parent", "gate"), ...)
```

Arguments

- `object` flowFrame, flowSet, GatingSet object
- `x` define the x dimension of the plot. When object is a flowFrame, it can be missing, which plots 1d density plot on all the channels.
- `y` define the y dimension of the plot. Default is NULL, which means 1d density plot.
- `bins` passed to geom_hex
- `...` other arguments passed to ggplot
- `gate` the gate to be plotted
- `bool` whether to plot boolean gates
- `arrange.main` the main title of the arranged plots
- `arrange` whether to use arrangeGrob to put multiple plots in the same page
- `merge` whether to merge multiple gates into the same panel when they share the same parent and projections
- `projections` a list of customized projections
- `strip.text` either "parent" (the parent population name) or "gate" (the gate name). The latter usually is used when merge is FALSE

Value

a ggcyto object

Examples

```r
library(flowCore)
data(GvHD)
```
fs <- GvHD[subset(pData(GvHD), Patient %in% 5:7 & Visit %in% c(5:6))]["name"]

# 1d- density plot
autoplot(fs, x = "SSC-H")

# 1d- density plot on all channels
autoplot(fs[[1]])

# 2d plot: default geom_hex plot
autoplot(fs, x = 'FSC-H', y = 'SSC-H')

# autoplot for GatingSet
dataDir <- system.file("extdata", package = "flowWorkspaceData")
gs <- load_gs(list.files(dataDir, pattern = "gs_manual", full = TRUE))
autoplot(gs, "CD3+")

# autoplot for GatingHierarchy
gh <- gs[[1]]
autoplot(gh) # by default the strip.text shows the parent population

# To display the gate name
# autoplot(gh, strip.text = "gate")

axis_x_inverse_trans --- Display axis labels in raw scales

Description
It is essentially a dummy continuous scale and will be instantiated by '+.ggcyto_GatingSet' with 'breaks' and 'labels' customized.

Usage
axis_x_inverse_trans(...)
axis_y_inverse_trans(...)

Arguments

... common continuous scale parameters passed to 'continuous_scale' (not used currently)

Value

a raw_scale object that inherits scale class.

Examples
dataDir <- system.file("extdata", package = "flowWorkspaceData")
gs <- load_gs(list.files(dataDir, pattern = "gs_manual", full = TRUE))
p <- ggcyto(gs, aes(x = CD4, y = CD8), subset = "CD3") + geom_hex(bins = 64)
p <- p + geom_gate("CD4") + geom_stats() # plot CD4 gate and it is stats
p
p + axis_x_inverse_trans() # inverse transform the x axis into raw scale
compute_stats  

**compute the statistics of the cell population defined by gates**

**Description**

It calls the underlining stats routine and merge it with the label position calculated by stat_position as well as the pData of flowSet.

**Usage**

```r
compute_stats(fs = NULL, gates, type = "percent", value = NULL,
data_range = NULL, ...)
```

**Arguments**

- `fs` flowSet. can be NULL when precaculated 'value' is provided
- `gates` a list of filters
- `type` can be "percent", "count" or "MFI".
- `value` the pre-calculated stats value. when supplied, the stats computing is skipped.
- `data_range` the data range for each channels
- `...` other arguments passed to stat_position function

**Details**

This function is usually not called directly by user but used by ggcyto when geom_stat layer is added.

**Value**

a data.table that contains percent and centroid locations as well as pData that used as data for geom_btext layer.

**Examples**

```r
data(GvHD)
fs <- GvHD[1:4]
rect.g <- rectangleGate(list("FSC-H" = c(300,500), "SSC-H" = c(50,200)))
rect.gates <- sapply(sampleNames(fs), function(sn)rect.g)
compute_stats(fs, rect.gates)
```
**fortify.ellipsoidGate**  
*Convert a ellipsoidGate to a data.table useful for ggplot*

**Description**

It interpolates the ellipsoidGate to polygongate before fortifying it.

**Usage**

```r
## S3 method for class 'ellipsoidGate'
fortify(model, data = NULL, ...)
```

**Arguments**

- `model` : ellipsoidGate
- `data` : data range used for polygon interpolation.
- `...` : not used.

**Value**

data.table

**Examples**

```r
## Defining the gate
cov <- matrix(c(6879, 3612, 3612, 5215), ncol=2,
dimnames=list(c("FSC-H", "SSC-H"), c("FSC-H", "SSC-H")))
mean <- c("FSC-H"=430, "SSC-H"=175)
eg <- ellipsoidGate(filterId= "myEllipsoidGate", .gate=cov, mean=mean)
fortify(eg)
```

---

**fortify.filterList**  
*Convert a filterList to a data.table useful for ggplot*

**Description**

It tries to merge with pData that is associated with filterList as attribute ‘pd’

**Usage**

```r
## S3 method for class 'filterList'
fortify(model, data = NULL, nPoints = NULL, ...)
```

**Arguments**

- `model` : filterList
- `data` : data range used for polygon interpolation
- `nPoints` : used for interpolating polygonGates to prevent it from losing shape when truncated by axis limits
- `...` : not used.

**Examples**

```r
## Defining the gate
cov <- matrix(c(6879, 3612, 3612, 5215), ncol=2,
dimnames=list(c("FSC-H", "SSC-H"), c("FSC-H", "SSC-H")))
mean <- c("FSC-H"=430, "SSC-H"=175)
fl <- filterList(filterId= "myFilterList", .gate=cov, mean=mean)
fortify(fl)
```
fortify.flowFrame

Convert a `flowFrame`/`flowSet`/`GatingSet` to a ggplot-compatible data.table

Description

It extracts events matrices and appends the pData to it so that ggplot can use the pData for facetting.

Usage

```r
## S3 method for class 'flowFrame'
fortify(model, data, ...)

## S3 method for class 'flowSet'
fortify(model, data, ...)

## S3 method for class 'ncdfFlowList'
fortify(model, ...)

## S3 method for class 'GatingSetList'
fortify(model, ...)

## S3 method for class 'GatingSet'
fortify(model, ...)```

Arguments

- `model`  
  flowFrame, flowSet or GatingSet
- `data`  
  not used.
- `...`  
  not used.

Value

- data.table
- data.table
- data.table
Examples

dataDir <- system.file("extdata", package = "flowWorkspaceData")
gs <- load_gs(list.files(dataDir, pattern = "gs_manual", full = TRUE))

attr(gs, "subset") <- "CD4"  # must attach subset information to GatingSet object before foritfying it
fortify(gs)

fs <- getData(gs, "CD8")
fortify(fs)  # fs is a flowSet/ncdfFlowSet

fr <- fs[[1]]
fortify(fr)  # fr is a flowFrame

---

**fortify.polygonGate**  
Convert a polygonGate to a data.table useful for ggplot

**Description**

It converts the boundaries slot into a data.table. When 'nPoints' is supplied, the method tries to
interpolate the polygon with more vertices.

**Usage**

```r
## S3 method for class 'polygonGate'
fortify(model, data = NULL, nPoints = NULL, ...)
```

**Arguments**

- `model`: polygonGate
- `data`: data range used to reset off-bound gate coordinates to prevent interpolating on
  the extremely large space unnecessarily.
- `nPoints`: total number of vertices of the polygon after interpolation. Default is NULL,
  which is no interpolation. The actual number may be more or less based on
  the lengths of edges due to the maximum and minimum limits on each edge.
  Interpolation is mainly for the purpose of plotting (so that it won’t lose its shape
  from subsetting through ‘limits’). But it is not necessary for other purposes like
  centroid calculation.
- `...`: not used.

**Value**

data.table

**Examples**

```r
sqrcut <- matrix(c(300,300,600,600,50,300,300,50), ncol=2, nrow=4)
colnames(sqrcut) <- c("FSC-H","SSC-H")
pg <- polygonGate(filterId = "nonDebris", .gate = sqrcut)
fortify(pg)  # no interpolation
fortify(pg, nPoints = 30)  # with interpolation
```
fortify.rectangleGate  Convert a rectangleGate to a data.table useful for ggplot

Description
For 2d rectangleGate, it is converted to a polygonGate first and then dispatch to the fortify method for polygonGate. For 1d, uses geom_vline/hline format.

Usage
```r
## S3 method for class 'rectangleGate'
fortify(model, data = NULL, ...)
```

Arguments
- `model`: rectangleGate
- `data`: data range used for polygon interpolation.
- `...`: not used.

Value
data.table

Examples
```r
#2d rectangleGate
rect.g <- rectangleGate(list("FSC-H" = c(300,500), "SSC-H" = c(50,200)))
fortify(rect.g)
#1d gate
rg <- rectangleGate(list("FSC-H" = c(300,500)))
fortify(rg)
```

fortify_fs  Fortify a model into flowSet object

Description
The method provides a universe interface to convert a generic R object into a flowSet useful for ggcyto

Usage
```r
fortify_fs(model, data, ...)
```

## S3 method for class 'flowSet'
fortify_fs(model, data, ...)

## Default S3 method:
fortify_fs(model, data, ...)

## S3 method for class 'flowFrame'
fortify_fs(model, data, ...)

## S3 method for class 'GatingSetList'
fortify_fs(model, data, ...)

## S3 method for class 'GatingSet'
fortify_fs(model, data, ...)

### Arguments

- **model**: flow object (flowFrame or GatingSet) to be converted to flowSet. when it is a GatingSet, it must contain the subset information stored as 'subset' attribute.
- **data**: original dataset, if needed
- **...**: other arguments passed to methods

### Value

a flowSet/ncdfFlowSet object

### Examples

```r
data(GvHD)
fr <- GvHD[[1]]
fortify_fs(fr)
dataDir <- system.file("extdata", package="flowWorkspaceData")
gs <- load_gs(list.files(dataDir, pattern = "gs_manual", full = TRUE))
attr(gs, "subset") <- "CD4"
fortify_fs(gs)
```

### geom_gate

add a flowCore gate layer

### Description

When 'data' is a gate (or flowCore filter) or a list of gates or a filterList object. When it is used directly with 'ggplot', pData of the flow data must be supplied through 'pd' argument explicitly in order for the gates to be dispatched to each panel. However it is not necessary when used with 'ggcyto' wrapper since the latter will attach pData automatically.

### Usage

```r
geom_gate(data, ...)
```

## Default S3 method:
geom_gate(data, ...)

## S3 method for class 'list'
geom_gate(data, ...)
## S3 method for class 'filterList'
geom_gate(data, ...)

## S3 method for class 'polygonGate'
geom_gate(data, ...)

## S3 method for class 'rectangleGate'
geom_gate(data, ...)

## S3 method for class 'ellipsoidGate'
geom_gate(data, ...)

## S3 method for class 'character'
geom_gate(data, ...)

## S3 method for class 'filters'
geom_gate(data, ...)

## S3 method for class 'filtersList'
geom_gate(data, ...)

## S3 method for class 'logicalFilterResult'
geom_gate(data, ...)

## S3 method for class 'logical'
geom_gate(data, ...)

### Arguments

- **data**: a filter (Currently only rectangleGate (1d or 2d), polygonGate, ellipsoidGate are supported.) or a list of these gates or filterList or character specifying a gated cell population in the GatingSet

- **...**: other arguments mapping. The mapping aesthetic mapping data a polygonGate fill polygonGate is not filled by default colour default is red pd pData (data.frame) that has rownames represents the sample names used as key to be merged with filterList

### Details

When 'data' is a character, it construct an abstract geom layer for a character that represents nodes in a Gating tree and will be instantiated later as a specific geom_gate layer or layers based on the gates extracted from the given GatingSet object.

### Value

- a geom_gate layer

### Examples

```r
data(GvHD)
fs <- GvHD[subset(pData(GvHD), Patient %in% 5:7 & Visit %in% c(5:6))[["name"]]]
p <- ggcyto(fs, aes(x = `FSC-H`, y = `SSC-H`))
p <- p + geom_hex(bins = 128)
```
rect.g <- rectangleGate(list("FSC-H" = c(300,500), "SSC-H" = c(50,200)))
# constructor for a list of filters
rect.gates <- sapply(sampleNames(fs), function(sn)rect.g)
p + geom_gate(rect.gates)

dataDir <- system.file("extdata", package="flowWorkspaceData")
gs <- load_gs(list.files(dataDir, pattern = "gs_manual", full = TRUE))
p <- ggcyto(gs, aes(x = CD4, y = CD8), subset = "CD3+") + geom_hex(bins = 64)
# add gate layer by gate name
p + geom_gate("CD4")

---

**Description**

This geom is based on the source code of `geom_hline` and `geom_vline`.

**Usage**

```
geom_hvline(mapping = NULL, data = NULL, position = "identity",
            show.legend = FALSE, ...)
```

**Arguments**

- **mapping**
  The aesthetic mapping, usually constructed with `aes` or `aes_string`. Only needs to be set at the layer level if you are overriding the plot defaults.
- **data**
  A layer specific dataset - only needed if you want to override the plot defaults.
- **position**
  The position adjustment to use for overlapping points on this layer
- **show.legend**
  should a legend be drawn? (defaults to FALSE)
- **...**
  other arguments passed on to `layer`. This can include aesthetics whose values you want to set, not map. See `layer` for more details.

**Details**

The goal is to determine the line to be either vertical or horizontal based on the 1-d data provided in this layer.

**Value**

a geom_hvline layer

**Aesthetics**

`geom_vline` understands the following aesthetics (required aesthetics are in bold):

- **xintercept**
- **alpha**
- **colour**
- **group**
- **linetype**
- **size**
Examples

```r
p <- ggplot(mtcars, aes(x = wt, y = mpg)) + geom_point()
# vline
p + geom_hvline(data = data.frame(wt= 3))
# hline
p + geom_hvline(data = data.frame(mpg= 20))
```

---

dotplot 

**Overlay other populations on existing plots.**

**Description**

It is useful for "backgating" plots.

**Usage**

```r
geom_overlay(data, ...)
```

## Default S3 method:
```r
gemm_overlay(data, ...)
```

## S3 method for class 'character'
```r
gemm_overlay(data, ...)
```

## S3 method for class 'flowSet'
```r
gemm_overlay(data, ...)
```

## S3 method for class 'flowFrame'
```r
gemm_overlay(data, ...)
```

**Arguments**

- `data`:
  a filter (Currently only rectangleGate (1d or 2d), polygonGate, ellipsoidGate are supported.) or a list of these gates or filterList or character specifying a gated cell population in the GatingSet

- `...`:
  other arguments mapping. The mapping aesthetic mapping data a polygonGate fill polygonGate is not filled by default colour default is red pd pData (data.frame) that has rownames represents the sample names used as key to be merged with filterList

**Value**

a geom_overlay layer
Examples

```r
library(ggcyto)
dataDir <- system.file("extdata",package="flowWorkspaceData")
gs <- load_gs(list.files(dataDir, pattern = "gs_manual",full = TRUE))
p <- autoplot(gs, "CD3+")

# add a flowSet as the overlay
fs <- getData(gs, "DPT")
p + geom_overlay(data = fs, size = 0.3, alpha = 0.7)

# add overlay layer by gate name
p + geom_overlay(data = "DNT", size = 0.3, alpha = 0.7)

# add overlay for 1d densityplot
p <- ggcyto(gs, aes(x = CD4), subset = "CD3+") + geom_density(aes(y = ..count..))
p + geom_overlay("DNT", aes(y = ..count..), fill = "red")
```

---

**geom_stats**  
*Population statistics layer*

**Description**

It is a virtual layer and will be instantiated as geom_label layer within ggcyto+ operator.

**Usage**

```r
geom_stats(gate = NULL, ..., value = NULL, type = "percent",
data_range = NULL, adjust = 0.5, label.padding = unit(0.05, "lines"),
label.size = 0)
```

**Arguments**

- `gate`  
  A `filterList` or character (represent as a population node in GatingSet) if not supplied, ggcyto then tries to parse the gate from the first geom_gate layer.

- `...`  
  Other arguments passed to geom_label layer.

- `value`  
  The pre-calculated stats value. When supplied, the stats computing is skipped.

- `type`  
  Can be "percent", "count" or "MFI".

- `data_range`  
  The data range for each channels.

- `adjust`  
  Adjust the position of the centroid. From 0 to 1.

- `label.padding`, `label.size`  
  Arguments passed to geom_label layer.

**Details**

So it is dedicated for ggcyto context and thus can’t not be added to ggplot object directly.

**Value**

A geom_popStats layer
getFlowFrame

Examples

dataDir <- system.file("extdata", package="flowWorkspaceData")
gs <- load_gs(list.files(dataDir, pattern = "gs_manual", full = TRUE))
p <- ggcyto(gs, aes(x = CD4, y = CD8), subset = "CD3+") + geom_hex(bins = 64)
p
# add gate and stats layer
p + geom_gate("CD4") + geom_stats()

getFlowFrame

extract flowFrame data structure from the given R object

Description

Mainly to get the channel and marker information.

Usage

getFlowFrame(x)

## S3 method for class 'flowSet'
getFlowFrame(x)

## S3 method for class 'ncdfFlowList'
getFlowFrame(x)

## S3 method for class 'GatingSetList'
getFlowFrame(x)

## S3 method for class 'GatingSet'
getFlowFrame(x)

## S3 method for class 'GatingHierarchy'
getFlowFrame(x)

Arguments

x

flowSet or GatingSet/GatingHierarchy

Value

a flowFrame. When x is a ncdfFlowSet or GatingSet that is associated with ncdfFlowSet, the raw event data is not read and an empty flowFrame is returned.

Examples

data(GvHD)
fs <- GvHD[1:2]
getFlowFrame(fs)# fs is a flowSet

dataDir <- system.file("extdata", package="flowWorkspaceData")
gs <- load_gs(list.files(dataDir, pattern = "gs_manual", full = TRUE))
getFlowFrame(gs)# gs is a GatingSet
Create a new ggcyto plot from a flowSet

Usage

```r
## S3 method for class 'FlowSet'
ggcyto(data, mapping, filter = NULL, ...)
```

Arguments

- `data`: default flowSet for plot
- `mapping`: default list of aesthetic mappings (these can be colour, size, shape, line type – see individual geom functions for more details)
- `filter`: a flowcore gate object or a function that takes flowSet and channels as input and returns a data-dependent flowcore gate. The gate is used to filter the flow data before it is plotted.
- `...`: ignored

Value

A `ggcyto_GatingSet` object which is a subclass of `ggcyto` class.

Examples

data(GvHD)
fs <- GvHD[, subset(pData(GvHD), Patient %in% 5:7 & Visit %in% c(5:6))]
# 1d histogram/density plot
p <- ggcyan(fs, aes(x = `FSC-H`))
#facet_wrap(~name) is used automatically
p1 <- p + geom_histogram()
p1
#overwriting the default faceeting
p1 + facet_grid(Patient~Visit)

#display density
p + geom_density()

# 2d scatter/dot plot
p <- ggcyan(fs, aes(x = `FSC-H`, y = `SSC-H`))
p <- p + geom_hex(bins = 128)
p
Create a new ggcyto plot from a GatingSet

Usage

## S3 method for class 'GatingSet'

```r
ggcyto(data, mapping, subset = "_parent_", ...)
```

## S3 method for class 'GatingSetList'

```r
ggcyto(data, ...)
```

## S3 method for class 'GatingHierarchy'

```r
ggcyto(data, ...)
```

Arguments

- **data**: GatingSet to plot
- **mapping**: default list of aesthetic mappings (these can be colour, size, shape, line type – see individual geom functions for more details)
- **subset**: character that specifies the node path or node name in the GatingSet. Default is "_parent_", which will be substitute with the actual node name based on the geom_gate layer to be added later.
- **...**: ignored

Value

- a ggcyto_GatingSet object which is a subclass of ggcyto_flowSet class.

Examples

```r
dataDir <- system.file("extdata", package="flowWorkspaceData")
gs <- load_gs(list.files(dataDir, pattern = "gs_manual", full = TRUE))
# 2d plot
ggcyto(gs, aes(x = CD4, y = CD8), subset = "CD3+") + geom_hex(bins = 64)

# 1d plot
ggcyto(gs, aes(x = CD4), subset = "CD3+") + geom_density()
```
`ggcyto_arrange`  

Arrange a list of ggplot objects into gtable

**Description**  
It is usually implicitly invoked by print and show method and can be called by user when the further manipulation is needed.

**Usage**  
`ggcyto_arrange(x, ...)`

**Arguments**  
- `x`: `ggcyto_gate_layout`, which is essentially a list of ggplot objects that were previously stored as `ggcyto_gate_layout` object by `autoplot` function.
- `...`: other arguments passed to `arrangeGrob`

**Value**  
gtable

**Examples**  
```r
## Not run:
# get `ggcyto_GatingLayout` object from first sample
res <- autoplot(gs[[1]], nodes, bins = 64)
class(res)
# arrange it as one-row gtable object
gt <- ggcyto_arrange(res, nrow = 1)
gt
# do the same to the second sample
gt2 <- ggcyto_arrange(autoplot(gs[[2]], nodes, bins = 64), nrow = 1)
# combine the two and print it on the same page
gt3 <- gridExtra::rbind.gtable(gt, gt2)
plot(gt3)
## End(Not run)
```

`ggcyto_par_default`  

Return The default `ggcyto` settings

**Description**  
Return The default `ggcyto` settings

**Usage**  
`ggcyto_par_default()`
Value

a list of default settings for ggcyto

Examples

ggcyto_par_default()

ggcyto_par_set(...)  

Arguments

dots

Value

a list of new settings for ggcyto

Elements

The individual elements are:

- Limits  
- Facet  
- Hex_fill  
- Lab

Examples

library(ggcyto)
dataDir <- system.file("extdata", package="flowWorkspaceData")
gs <- load_gs(list.files(dataDir, pattern = "gs_manual", full = TRUE))

p <- ggcyto(gs, aes(x = CD4, y = CD8), subset = "CD3+")

# 2d plot
p <- p + geom_hex(bins = 64)

# use instrument range by overwritting the default limits settings
p + ggcyto_par_set(limits = "instrument")

# manually set limits
myPars <- ggcyto_par_set(limits = list(x = c(0, 3.2e3), y = c(-10, 3.5e3)))
is.ggcyto

\[ p + myPars# \text{ or } \text{xlim}(0, 3.2e3) + \text{ylim}(-10, 3.5e3) \]

**Description**

Reports whether \( x \) is a ggcyto object

**Usage**

\[
\text{is.ggcyto}(x)
\]

**Arguments**

\( x \)

An object to test

**Value**

TRUE/FALSE

**Examples**

```
data(GvHD)
fs <- GvHD[1:2]
p <- ggcyto(fs, aes(x = 'FSC-H'))
is.ggcyto(p)
```

is.ggcyto_flowSet

**Description**

Reports whether \( x \) is a ggcyto_flowSet object

**Usage**

\[
\text{is.ggcyto}(_{\text{flowSet}})(x)
\]

**Arguments**

\( x \)

An object to test

**Value**

TRUE or FALSE

**Examples**

```
data(GvHD)
fs <- GvHD[1:2]
p <- ggcyto(fs, aes(x = 'FSC-H'))
is.ggcyto_flowSet(p)
```
is.ggcyto_par Reports whether x is a ggcyto_par object

Description
Reports whether x is a ggcyto_par object

Usage
is.ggcyto_par(x)

Arguments
x An object to test

Value
TRUE or FALSE

Examples
myPar <- ggcyto_par_set(limits = "instrument")
is.ggcyto_par(myPar)

labs_cyto Change axis labels and legend titles

Description
The actual labels text will be instantiated when it is added to ggcyto plot.

Usage
labs_cyto(labels = "both")

Arguments
labels default labels for x, y axis. Can be "channel", "marker", or "both" (default)

Value
a list
marginalFilter

Examples

dataDir <- system.file("extdata", package="flowWorkspaceData")
gs <- load_gs(list.files(dataDir, pattern = "gs_manual", full = TRUE))

# default is "both"
p <- ggcyto(gs, aes(x = CD4, y = CD8), subset = "CD3+") + geom_hex(bins = 64)
p

#use marker name as x,y labs
p + labs_cyto("marker")

#use channel name as x,y labs
p + labs_cyto("channel")

marginalFilter

Generate a marginal gate.

Description

It simply constructs an boundaryFilter that removes the marginal events. It can be passed directly
to ggcyto constructor. See the examples for details.

Usage

marginalFilter(fs, dims, ...)

Arguments

fs flowSet (not used.)
dims the channels involved
... arguments passed to boundaryFilter

Value

an boundaryFilter

Examples

data(GvHD)
fs <- GvHD[1]
chnls <- c("FSC-H", "SSC-H")
# before removing marginal events
summary(fs[, chnls])

# create marginal filter
g <- marginalFilter(fs, chnls)
g

# after remove marginal events
fs.clean <- Subset(fs, g)
summary(fs.clean[, chnls])
#pass the function directly to ggcyto
dataDir <- system.file("extdata", package = "flowWorkspaceData")
gs <- load_gs(list.files(dataDir, pattern = "gs_manual", full = TRUE))
# with marginal events
ggcyto(gs, aes(x = CD4, y = CD8), subset = "CD3+") + geom_hex(bins = 64)

# using marginalFilter to remove these events
ggcyto(gs, aes(x = CD4, y = CD8), subset = "CD3+", filter = marginalFilter) + geom_hex(bins = 64)

---

**print.ggcyto**

*Draw ggcyto on current graphics device.*

**Description**

A wrapper for print.ggplot. It converts the ggcyto to conventional ggplot object before printing it. This is usually invoked automatically when a ggcyto object is returned to R console.

**Usage**

```r
## S3 method for class 'ggcyto'
print(x, ...)

## S3 method for class 'ggcyto'
plot(x, ...)

## S4 method for signature 'ggcyto'
print(x, ...)

## S3 method for class 'ggcyto'
show(object)

## S4 method for signature 'ggcyto'
show(object)
```

**Arguments**

- `x`  
ggcyto object to display
- `...`  
other arguments not used by this method
- `object`  
ggcyto object

**Value**

nothing
print.ggcyto_GatingLayout

print method for ggcyto_gate_layout class

Description

print method for ggcyto_gate_layout class

Usage

## S3 method for class 'ggcyto_GatingLayout'
print(x, ...)

## S3 method for class 'ggcyto_GatingLayout'
show(object)

## S4 method for signature 'ggcyto_GatingLayout'
show(object)

Arguments

x ggcyto_gate_layout, which is essentially a list of ggplot objects that were previously stored as ggcyto_gate_layout object by autoplot function.

... other arguments passed to arrangeGrob

object ggcyto_GatingLayout

Value

nothing

scale_x_flowJo_biexp flowJo biexponential scale

Description

flowJo biexponential scale

Usage

scale_x_flowJo_biexp(..., maxValue = 262144, widthBasis = -10, pos = 4.5, neg = 0, equal.space = FALSE)

scale_y_flowJo_biexp(..., maxValue = 262144, widthBasis = -10, pos = 4.5, neg = 0, equal.space = FALSE)
**Arguments**

... common continuous scale parameters passed to `continuous_scale` (not used currently)

- `maxValue`, `widthBasis`, `pos`, `neg`
- see `help(flowJoTrans)`
- `equal.space` whether to display the breaks in equal.space format

**Value**

ScaleContinuous object

**Examples**

```r
data(GvHD)
fr <- GvHD[[1]]
p <- ggcyto(fr, aes(x = FL1-H)) + geom_density()
# display at raw scale
p
# display at transformed scale
p + scale_x_flowJo_fasinh(maxValue = 1e4, widthBasis = 0)
```

---

**scale_x_flowJo_fasinh**  
flowJo inverse hyperbolic sine scale

**Description**

flowJo inverse hyperbolic sine scale

**Usage**

```r
scale_x_flowJo_fasinh(..., m = 4, t = 1200)
scale_y_flowJo_fasinh(..., m = 4, t = 1200)
```

**Arguments**

... common continuous scale parameters passed to `continuous_scale` (not used currently)

- `m`, `t` see `help(flowJo.fasinh)`

**Value**

ScaleContinuous object

**Examples**

```r
data(GvHD)
fr <- GvHD[[1]]
p <- ggcyto(fr, aes(x = FL1-H)) + geom_density()
# display at raw scale
p
# display at transformed scale
p + scale_x_flowJo_fasinh(t = 1e4)
```
**scale_x_logicle**  
flowJo inverse hyperbolic sine scale

**Description**
flowJo inverse hyperbolic sine scale

**Usage**

```r
scale_x_logicle(..., w = 0.5, t = 262144, m = 4.5, a = 0)
```

```r
scale_y_logicle(..., w = 0.5, t = 262144, m = 4.5, a = 0)
```

**Arguments**

... common continuous scale parameters passed to `continuous_scale` (not used currently)

w, t, m, a see `help(logicleTransform)`

**Value**

ScaleContinuous object

**Examples**

data(GvHD)
fr <- GvHD[[1]]
p <- ggcyto(fr, aes(x = `/grave.Var FL1-H`)) + geom_density()
#display at raw scale
p
#display at transformed scale
p + scale_x_logicle(t = 1e4)

---

**stat_position**
compute the positions of the population statistics based on the geometric gate centroid

**Description**

It is usually not called directly by user but mainly used by compute_stats function (which is called by ggcyto add method when geom_states layer is added).

**Usage**

```r
stat_position(gate, ...)
```

```r
## S3 method for class 'filter'
stat_position(gate, ...)
```

```r
## S3 method for class 'filterList'
```
stat_position(gate, ...)

## S3 method for class 'list'
stat_position(gate, ...)

Arguments

gate a flowCore filter

... other arguments adjust adjust the position of the centroid

abs logical

data_range the actual data range

Value

a data.table

the gate centroid coordinates

Examples

data(GvHD)
fs <- GvHD[1:4]
rect.g <- rectangleGate(list("FSC-H" = c(300,500), "SSC-H" = c(50,200)))
rect.gates <- sapply(sampleNames(fs), function(sn)rect.g)
stat_position(rect.gates)
Index

+,ggcyto_GatingLayout,ANY-method
   (+.ggcyto_GatingLayout), 3
+,ggcyto_GatingSet,ANY-method
   (+.ggcyto_GatingSet), 4
+,ggcyto_flowSet,ANY-method
   (+.ggcyto_flowSet), 2
+.ggcyto_GatingLayout, 3
+.ggcyto_GatingSet, 4
+.ggcyto_flowSet, 2

aes, 15
aes_string, 15
as.ggplot, 5
autoplot.flowFrame (autoplot.flowSet), 5
autoplot.flowSet, 5
autoplot.GatingHierarchy (autoplot.flowSet), 5
autoplot.GatingSet (autoplot.flowSet), 5
autoplot.GatingSetList (autoplot.flowSet), 5
autoplot.ncdfFlowList (autoplot.flowSet), 5
axis_x_inverse_trans, 7
axis_y_inverse_trans
   (axis_x_inverse_trans), 7
boundaryFilter, 25
compute_stats, 8
fortify (fortify.flowFrame), 10
fortify.ellipsoidGate, 9
fortify.filterList, 9
fortify.flowFrame, 10
fortify.polygonGate, 11
fortify.rectangleGate, 12
fortify_fs, 12
geom_gate, 13
geom_hline, 15
geom_hvline, 15
geom_overlay, 16
geom_stats, 17
geom_vline, 15
getFlowFrame, 18
ggcyto.flowSet, 19
ggcyto.GatingHierarchy
   (ggcyto.GatingSet), 20
ggcyto.GatingSet, 20
ggcyto.GatingSetList
   (ggcyto.GatingSet), 20
ggcyto_arrange, 21
ggcyto_par_default, 21
ggcyto_par_set, 22
is.ggcyto, 23
is.ggcyto_flowSet, 23
is.ggcyto_par, 24
labs_cyto, 24
layer, 15
marginalFilter, 25
plot.ggcyto (print.ggcyto), 26
print, ggcyto-method (print.ggcyto), 26
print.ggcyto, 26
print.ggcyto_GatingLayout, 27
scale_x_flowJo_biexp, 27
scale_x_flowJo_fasinh, 28
scale_x_logicle, 29
scale_y_flowJo_biexp
   (scale_x_flowJo_biexp), 27
scale_y_flowJo_fasinh
   (scale_x_flowJo_fasinh), 28
scale_y_logicle (scale_x_logicle), 29
show, ggcyto-method (print.ggcyto), 26
show, ggcyto_GatingLayout-method
   (print.ggcyto_GatingLayout), 27
show.ggcyto (print.ggcyto), 26
show.ggcyto_GatingLayout
   (print.ggcyto_GatingLayout), 27
stat_position, 29