Package ‘ggcyto’

May 3, 2024

Type Package

Title Visualize Cytometry data with ggplot

Version 1.32.0

Date 2015-11-02

Author Mike Jiang

Maintainer Mike Jiang <mike@ozette.com>

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

VignetteBuilder knitr

Depends methods, ggplot2(>= 3.5.0), flowCore(>= 1.41.5), ncdfflow(>= 2.17.1), flowWorkspace(>= 4.3.1)

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

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

License file LICENSE

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

biocViews ImmunoOncology, FlowCytometry, CellBasedAssays, Infrastructure, Visualization

**RoxygenNote** 7.2.1

**Roxygen** list(markdown=TRUE)

**git_url** https://git.bioconductor.org/packages/ggcyto

**git_branch** RELEASE_3_19

**git_last_commit** 29bf0b0

**git_last_commit_date** 2024-04-30

**Repository** Bioconductor 3.19

**Date/Publication** 2024-05-03

### Contents

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>as.ggplot</td>
<td>3</td>
</tr>
<tr>
<td>autoplot.flowSet</td>
<td>4</td>
</tr>
<tr>
<td>axis_x_inverse_trans</td>
<td>6</td>
</tr>
<tr>
<td>compute_stats</td>
<td>7</td>
</tr>
<tr>
<td>faust_gating_plot</td>
<td>8</td>
</tr>
<tr>
<td>flowCore_asinh_trans</td>
<td>8</td>
</tr>
<tr>
<td>fortify.cytoframe</td>
<td>9</td>
</tr>
<tr>
<td>fortify.ellipsoidGate</td>
<td>10</td>
</tr>
<tr>
<td>fortify.filterList</td>
<td>11</td>
</tr>
<tr>
<td>fortify.multiRangeGate</td>
<td>12</td>
</tr>
<tr>
<td>fortify.polygonGate</td>
<td>12</td>
</tr>
<tr>
<td>fortify.rectangleGate</td>
<td>13</td>
</tr>
<tr>
<td>fortify_fs</td>
<td>14</td>
</tr>
<tr>
<td>gate_null</td>
<td>15</td>
</tr>
<tr>
<td>geom_gate</td>
<td>15</td>
</tr>
<tr>
<td>geom_hvline</td>
<td>17</td>
</tr>
<tr>
<td>geom_multi_range</td>
<td>18</td>
</tr>
<tr>
<td>geom_overlay</td>
<td>19</td>
</tr>
<tr>
<td>geom_stats</td>
<td>20</td>
</tr>
<tr>
<td>getFlowFrame</td>
<td>21</td>
</tr>
<tr>
<td>ggcyto-class</td>
<td>22</td>
</tr>
<tr>
<td>ggcyto_add</td>
<td>24</td>
</tr>
<tr>
<td>ggcyto_arrange</td>
<td>25</td>
</tr>
<tr>
<td>ggcyto_par_default</td>
<td>26</td>
</tr>
<tr>
<td>ggcyto_par_set</td>
<td>26</td>
</tr>
<tr>
<td>is.ggcyto</td>
<td>27</td>
</tr>
<tr>
<td>is.ggcyto_flowSet</td>
<td>28</td>
</tr>
<tr>
<td>is.ggcyto_par</td>
<td>28</td>
</tr>
<tr>
<td>labs_cyto</td>
<td>29</td>
</tr>
<tr>
<td>marginalFilter</td>
<td>30</td>
</tr>
<tr>
<td>merge_quad.gates</td>
<td>31</td>
</tr>
<tr>
<td>print.ggcyto</td>
<td>32</td>
</tr>
<tr>
<td>print.ggcyto_GatingLayout</td>
<td>32</td>
</tr>
<tr>
<td>replace_data</td>
<td>33</td>
</tr>
</tbody>
</table>
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, pre_binning = FALSE)
```

Arguments

- `x`: ggcyto object with the data that has not yet been fortified to data.frame.
- `pre_binning`: whether to pass the binned data to ggplot to avoid the overhead to scaling the original raw data for geom_hex layer

Value

`ggplot` object

Examples

```r
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$data # data has not been fortified
p1 <- as.ggplot(p) # convert it to a ggplot object explicitely
class(p1)
p1$data # data is fortified
```
Plot cytometry data in one or two dimension with the ggcyto package.

Description

Overloaded autoplot methods for the cytometry data structure: flowFrame or flowSet, Gatinghierarchy, GatingSet. It plots the cytometry data with geom_histogram, geom_density or geom_hex. When autoplot is called on a GatingSet/Gatinghierarchy, the second argument should be a gate or population node. And the dimensions(channels/markers) are deduced from the gate dimensions.

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 'cytoset'
autoplot(object, ...)

## S3 method for class 'cytoframe'
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,
  axis_inverse_trans = TRUE,
  ...
)

## S3 method for class 'GatingHierarchy'
autoplot(
  object,
  gate,
  y = "SSC-A",
  bool = FALSE,
  ...}
```
autoplot.flowSet

arrange.main = sampleNames(object),
arrange = TRUE,
merge = TRUE,
projections = list(),
strip.text = c("parent", "gate"),
path = "auto",
...
)

Arguments

object The data source. A core cytometry data structure. A flowFrame, flowSet, GatingSet or GatingHierarchy object
x define the x dimension of the plot (not used when object is a GatingSet). 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
axis_inverse_trans logical flag indicating whether to add axis_x_inverse_trans and axis_x_inverse_trans layers.
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
path the gating path format (passed to gs_get_pop_paths)

Value

a ggcyto object

Examples

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")
# Density plot on all channels
autoplot(fs[[1]])

# 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 ggcyto axis labels using their raw values (as stored in the data structure)

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

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

Arguments

fs flowSet. can be NULL when precaculated 'value' is provided
gates a list of filters
type a vector of strings to specify the stats types. can be any or multiple values of "percent", "count", "gate_name", or "MFI" (MFI is currently not supported yet).
value the pre-calculated stats value. when supplied, the stats computing is skipped.
... 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

data(GvHD)
fs <- GvHD[1:4]
rect.g <- rectangleGate(list("FSC-H" = c(300,500), "SSC-H" = c(50,200)), filterId = "P1")
rect.gates <- sapply(sampleNames(fs), function(sn)rect.g)
compute_stats(fs, rect.gates)
compute_stats(fs, rect.gates, type = c("gate_name", "percent"))
faust_gating_plot  plot faust gating schemes

Description

plot faust gating schemes

Usage

faust_gating_plot(gh, start_node, end_node, ...)

Arguments

start_node  faust start node
end_node  the terminal leaf node generated by faust

Examples

## Not run:
gs=load_gs("~/Downloads/ics")

end_node = "/S/LV/L/CD4+/CD8-/TNF+/CD107a-/IFNg+/IL2+/CD154-/IL17a-"
start_node = "/S/LV/L"
gh=gs[[1]]
p = faust_gating_plot(gh, start_node, end_node, bins=128)
plot(ggcyto_arrange(p, nrow=1))

## End(Not run)

flowCore_asinht_trans  Inverse hyperbolic sine transformation(flowCore version).

Description

Used to construct inverse hyperbolic sine transform object.

Usage

flowCore_asinht_trans(..., n = 6, equal.space = FALSE)

Arguments

...  parameters passed to arcsinhTransform
n  desired number of breaks (the actual number will be different depending on the
data range)
equal.space  whether breaks at equal-spaced intervals
Value

asinht transformation object

Examples

trans.obj <- flowCore_asinht_trans(equal.space = TRUE)
data <- 1:1e3brks.func <- trans.obj[['breaks']]brks <- brks.func(data)brks # fasinh space displayed at raw data scale

#transform it to verify it is equal-spaced at transformed scaletrans.func <- trans.obj[['transform']]brks.trans <- trans.func(brks)brks.trans

---

fortify.cytoframe 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 faceting.

Usage

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

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

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

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

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

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

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

Arguments

model ellipsoidGate
... not used.
data 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 fortifying it
fortify(gs)

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

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

fortify.ellipsoidGate Convert a ellipsoidGate to a data.table useful for ggplot

Description

It interpolates the ellipsoidGate to polygongate before fortifying it.

Usage

## 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`: not used
- `nPoints`: not used
- `...`: not used.

Value
---

data.table

Examples
---

```r
dataDir <- system.file("extdata",package="flowWorkspaceData")
gs <- load_gs(list.files(dataDir, pattern = "gs_manual",full = TRUE))
gates <- gs_pop_get_gate(gs, "CD4")
gates <- as(gates, "filterList") #must convert list to filterList in order for the method to dispatch properly
fortify(gates)
```
fortify.multiRangeGate

Convert a multiRangeGate to a data.table useful for ggplot

Description

It converts the boundaries slot into a data.table

Usage

## S3 method for class 'multiRangeGate'
fortify(model, data = NULL, ...)

Arguments

- model: multiRangeGate
- data: Not used
- ...: not used.
- nPoints: not used

Value

data.table

Examples

mrq = multiRangeGate(ranges = list(min=c(100, 350), max=c(250, 400)))
fortify(mrq)

fortify.polygonGate

Convert a polygonGate to a data.table useful for ggplot

Description

It converts the boundaries slot into a data.table

Usage

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

Examples

mrq = polygonGate(ranges = list(min=c(100, 350), max=c(250, 400)))
fortify(mrq)
fortify.rectangleGate

Arguments

model rectangleGate
data data range used for polygon interpolation.
... not used.

Value
data.table

Examples

sqrcut <- matrix(c(300,300,600,600,50,300,50,300),ncol=2,nrow=4)
colnames(sqrcut) <- c("FSC-H","SSC-H")
pg <- polygonGate(filterId="nonDebris", .gate= sqrcut)
fortify(pg)

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

## 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

```
# 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

```
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 'cytoframe'

```
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

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)

gate_null clear all the geom_gate() layer previously added

Description

clear all the geom_gate() layer previously added

Usage

gate_null()

Examples

dataDir <- system.file("extdata",package="flowWorkspaceData")
gs <- load_gs(list.files(dataDir, pattern = "gs_manual",full = TRUE))
#autoplot display pop stats by default
p <- autoplot(gs, "CD4")
#it is easy to remove the default gate
p <- p + gate_null()
#and add a new one
p <- p + geom_gate("CD8")
p

geom_gate Add a gate layer to a ggcyto plot.

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

geom_gate(data, ...)  
## S3 method for class 'filterList'
geom_gate(data, pd, nPoints = 100, ...)
## S3 method for class 'filter'
geom_gate(data, mapping = NULL, fill = NA, colour = "red", nPoints = 100, ...)

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

pd pData (data.frame) that has rownames represents the sample names used as key to be merged with filterList

nPoints used for interpolating polygonGates to prevent them from losing shape when truncated by axis limits

mapping The aesthetic mapping

fill fill color for the gate. Not filled by default.

colour default is red

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 instanatiated 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

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

```r
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

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

- `xintercept`
- `alpha`
- `colour`
- `group`
• **linetype**
• **linewidth**

Learn more about setting these aesthetics in vignette("ggplot2-specs").

**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))
```

---

**geom_multi_range**

*Draw multi-ranges as multiple rectangles on 1D or 2D plot*

**Description**

This geom is based on the source code of `geom_rect`

**Usage**

```r
geom_multi_range(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,  
  linejoin = "mitre",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

**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
- **...**  other arguments passed on to `layer`. This can include aesthetics whose values you want to set, not map. See `layer` for more details.
- **show.legend**  should a legend be drawn? (defaults to `FALSE`)
Details

The goal is to determine the line to be either vertical or horizontal based on the data provided in this layer. Also convert input 1D intervals to geom_rect acceptable shapes.

Value

a geom_rect layer

Aesthetics

@section Aesthetics: geom_vline() understands the following aesthetics (required aesthetics are in bold):

- xintercept
- alpha
- colour
- group
- linetype
- linewidth

Learn more about setting these aesthetics in vignette("ggplot2-specs").

---

**geom_overlay**

*Overlay a population on an existing ggcyto plot analogous to backgating.*

Description

It is useful for "backgating" plots.

Usage

`geom_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 <- gs_pop_get_data(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")
```

description

Add a population statistics layer to a ggcyto plot.

Usage

```r
geom_stats(
  gate = NULL,
  ..., 
  value = NULL,
  type = "percent",
  negated = FALSE,
  adjust = 0.5,
  location = "gate",
  label.padding = unit(0.05, "lines"),
  label.size = 0,
  digits = 3
)
```

Arguments

gate a 'filterList' or character (represent as a population node in GatingSet) if not supplied, ggcyno 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 a vector of strings to specify the stats types. can be any or multiple values of "percent", "count", "gate_name", or "MFI" (MFI is currently not supported yet).
getFlowFrame

negated whether the gate needs to be negated
adjust see details for stat_position
location see details for stat_position
label.padding, label.size arguments passed to geom_label layer
digits control the stats format

Details
So it is dedicated for ggcyto context and thus cannot be added to ggplot object directly.

Value
a geom_popStats layer

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()

# display gate name
p + geom_gate(c("CD4", "CD8")) + geom_stats(type = "gate_name")
# display gate name and percent
p + geom_gate(c("CD4", "CD8")) + geom_stats(type = c("gate_name", "percent"))

getFlowFrame extract flowFrame data structure from the given R object

Description
Mainly to get the channel and marker information.

Usage
getFlowFrame(x)

Arguments
x flowSet, ncdfFlowList, GatingSet, GatingHierarchy, or GatingSetList

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

```r
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
```

---

**ggcyto-class**

*Plot cytometry data using the ggcyto API*

---

**Description**

`ggcyto()` initializes a ggcyto object that inherits ggplot class. Similarly the `+` operator can be used to add layers to the existing ggcyto object.

**Usage**

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

```r
## S3 method for class 'GatingSet'
ggcyto(data, mapping, subset = "_parent_", ...)
```

```r
## S3 method for class 'GatingSetList'
ggcyto(data, ...)
```

```r
## S3 method for class 'GatingHierarchy'
ggcyto(data, ...)
```

```r
## S3 method for class 'flowSet'
ggcyto(data, mapping, filter = NULL, max_nrow_to_plot = 50000, ...)
```

**Arguments**

- **data**
  The data source. A core cytometry data structure. (flowSet, flowFrame, ncdfFlowSet, GatingSet or GatingHierarchy)

- **...**
  other arguments passed to specific methods

- **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 case of GatingSet. Default is "parent", which will be substituted with the actual node name based on the geom_gate layer to be added later.

- **filter**
  a flowcore gate object or a function that takes a 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.
max_nrow_to_plot

the maximum number of cells to be plotted. When the actual data exceeds it, The subsampling process will be triggered to speed up plotting. Default is 5e4. To turn off the subsampling, simply set it to a large enough number or Inf.

Details

To invoke ggcyto:

• ggcyto(fs, aes(x, y, <other aesthetics>))

Value

ggcyto object

Examples

data(GvHD)
fs <- GvHD[1:3]
#construct the `ggcyto` object (inherits from `ggplot` class)
p <- ggcyto(fs, aes(x = `FSC-H`))
p + geom_histogram()

# display density/area
p + geom_density()
p + geom_area(stat = "density")

# 2d scatter plot
p <- ggcyto(fs, aes(x = `FSC-H`, y = `SSC-H`))
p + geom_hex(bins = 128)
# do it programatically through aes_string and variables
col1 <- "FSC-H" #note that the dimension names with special characters needs to be quoted by backticks
col2 <- "SSC-H"
ggcyto(fs, aes_string(col1,col2)) + geom_hex()

## More flowSet examples
fs <- GvHD[subset(pData(GvHD), Patient %in% 5:7 & Visit %in% c(5:6))]["name"]
# 1d histogram/densityplot
p <- ggcyto(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()

#you can use ggridges package to display stacked density plot
require(ggridges)
#stack by fcs file ('name')
p + geom_density_ridges(aes(y = name)) + facet_null() #facet_null is used to remove the default facet_wrap (by 'name')
#or to stack by Visit and facet by patient
p + geom_density_ridges(aes(y = Visit)) + facet_grid(~Patient)

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

## GatingSet
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_add**

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**

e1 + e2

**Arguments**

e1 An object of class ggcyto or a class inheriting from ggcyto, such as ggcyto_flowSet, ggcyto_GatingSet, or ggcyto_GatingLayout. In the case of ggcyto_GatingLayout, the component of e2 will be added to each subsidiary plot.
e2 A component to add to e1

**Value**

ggcyto object

**Examples**

## flowSet
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)))
### ggcyo_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**

```r
ggcyo_arrange(x, ...)```

**Arguments**

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

- `...`  
  other arguments passed to arrangeGrob

**Value**

gtable

**Examples**

```r
## Not run:
# get ggcyo_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)

# 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::gtable_rbind(gt, gt2)
plot(gt3)

## End(Not run)

---

**ggcyto_par_default**  
*Return the default ggcyto settings*

**Description**

Return the default ggcyto settings.

**Usage**

```r
ggcyto_par_default()
```

**Value**

a list of default settings for ggcyto

**Examples**

```r
ggcyto_par_default()
```

---

**ggcyto_par_set**  
*Set some default parameters for ggcyto *

**Description**

Use this function to modify ggcyto parameters. These are the regular (or to be instantiated as) scales, labs, facet objects. They can be added as a single layer to the plot for the convenience.

**Usage**

```r
ggcyto_par_set(...)
```

**Arguments**

```r
...
```

a list of element name, element pairings that modify the existing parameter settings.
is.ggcyto

Value

a list of new settings for ggcyto

elements

The individual elements are:

- limits: can be "data" (default) or "instrument" or a list of numeric limits for x and y (e.g. list(x = c(0, 4000))
- facet: the regular facet object
- hex_fill: default scale_fill_gradientn for geom_hex layer
- lab: labs_cyto object

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)
p

# use instrument range by overwriting 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)))
p + myPars# or xlim(0, 3.2e3) + ylim(-10, 3.5e3)

is.ggcyto

Reports whether x is a ggcyto object

Description

Reports whether x is a ggcyto object

Usage

is.ggcyto(x)

Arguments

x An object to test

Value

TRUE/FALSE
Examples

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

---

is.ggcyto_flowSet Reports whether x is a ggcyto_flowSet object

Description

Reports whether x is a ggcyto_flowSet object

Usage

```r
is.ggcyto_flowSet(x)
```

Arguments

- `x` An object to test

Value

TRUE or FALSE

Examples

```r
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

```r
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

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 a 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 removign marginal events
summary(fs[, chnls])

# create merginal 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)
```
merge.quad.gates

extend the original flowWorkspace::mergeGates function to restore quadGate when applicable

Description

For internal usage.

Usage

```r
## S3 method for class 'quad.gates'
merge(gh, pops, bool = TRUE)
```

Arguments

- `gh`: a GatingHierarchy
- `pops`: a vector of population names
- `bool`: whether to deal with boolean gate

Value

A nested list of data structure that captures the information of parent, grouped populations (with the same projections) and the reconstructed quadGate object and the respective quadrant pattern.

Examples

```r
library(flowWorkspace)
dataDir <- system.file("extdata", package="flowWorkspaceData")
gs <- load_gs(file.path(dataDir, "gs_manual"))
# get the GatingHierarchy object
gh <- gs[[1]]
pops <- gs_pop_get_children(gh, "CD4")
grps <- ggcryo::merge.quad.gates(gh, pops)
length(grps) # pops are grouped into two
grps[[1]] # each group is annotated with quadGate information

ggcryo::merge.quad.gates(gh, gs_pop_get_children(gh, "CD3+")) # cd3 subsets are not coercible to quadgate thus return as they are
```
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, ...)

## S3 method for class '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**

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

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

Arguments

- \texttt{x} \quad \text{ggcyto\_gate\_layout}, which is essentially a list of ggplot objects that were previously stored as \texttt{ggcyto\_gate\_layout} object by autoplot function.
- \ldots\ldots \quad \text{other arguments passed to arrangeGrob object}
- \texttt{object} \quad \texttt{ggcyto\_GatingLayout}

Value

- \text{nothing}

**Description**

It essentially reconstructs the entire \texttt{ggcyto} plot object based on the new data and the original mapping and layers recorded in the plot object.

**Usage**

\texttt{e1 \%+\% e2}

**Arguments**

- \texttt{e1} \quad the \texttt{ggcyto} object
- \texttt{e2} \quad the new cytometry data. It can be \texttt{‘GatingSet’} or \texttt{‘flowSet’}.

**Value**

- the new \texttt{ggcyto} object

**Examples**

```r

dataDir <- system.file("extdata", package="flowWorkspaceData")
gs <- load_gs(list.files(dataDir, pattern = "gs\_bcell\_auto", full = TRUE))
gs1 <- gs[1]
gs2 <- gs[2]

# construct the \texttt{ggcyto} object for gs1
p <- ggcyto(gs1, aes(cd24, cd38)) + geom_hex(bins = 128)
p <- p + geom_gate("Transitional") # add gate
# customize the stats layer
p <- p + geom_stats(type = "count", size = 6, color = "white", fill = "black", adjust = 0.3)
# customize the layer
p <- p + labs_cyto("channel")
# customize the axis limits
p <- p + ggcyto_par_set(limits = "instrument")
# add another population as the overlay dots
```
```r
p <- p + geom_overlay("IgD-CD27-", col = "black", size = 1.2, alpha = 0.4)
p

# replace the data with gs2 and see the same visual effect
p %+% gs2
```

---

**scales_flowjo_biexp**  
Add a flowJo biexponential scale to the x or y axes of a ggcyto plot.

---

**Description**

Add a flowJo biexponential scale to the x or y axes of a ggcyto plot.

**Usage**

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

type = "scale",

class = "scale_continuous",

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

type = "scale",

class = "scale_continuous"
```

**Arguments**

- `...` common continuous scale parameters passed to 'continuous_scale' (not used currently)
- `maxValue`, `widthBasis`, `pos`, `neg`
  - see 'help(flowjo_biexp)'
- `equal.space` whether to display the breaks in equal.space format

**Value**

ScaleContinuous object
scales_flowjo_fasinh

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)
```

scales_flowjo_fasinh  
*Add a flowJo inverse hyperbolic sine scale to the x or y axes of a ggcyto plot.*

Description

Add a flowJo inverse hyperbolic sine scale to the x or y axes of a ggcyto plot.

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_flowCore_fasinh

Add a flowCore inverse hyperbolic sine scale to the x or y axes of a ggcyto plot.

Description
Add a flowCore inverse hyperbolic sine scale to the x or y axes of a ggcyto plot.

Usage
scale_x_flowCore_fasinh(..., a = 1, b = 1, c = 0)
scale_y_flowCore_fasinh(..., a = 1, b = 1, c = 0)

Arguments
... common continuous scale parameters passed to 'continuous_scale' (not used currently)
a, b, c see 'help(arcsinhTransform')

Value
ScaleContinuous object

Examples
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_flowCore_fasinh(a = 2)

scale_x_logicle

Add a logicle scale to the x or y axes of a ggcyto plot.

Description
Add a logicle scale to the x or y axes of a ggcyto plot.

Usage
scale_x_logicle(..., w = 0.5, t = 262144, m = 4.5, a = 0)
scale_y_logicle(..., w = 0.5, t = 262144, m = 4.5, a = 0)
**stats_null**

**Arguments**

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

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

**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_logicle(t = 1e4)
```

---

**stats_null**  
**clear all the geom_stats() layer previously added**

**Description**

clear all the geom_stats() layer previously added

**Usage**

```r
stats_null()
```

**Examples**

```r
dataDir <- system.file("extdata", package="flowWorkspaceData")
gs <- load_gs(list.files(dataDir, pattern = "gs_manual", full = TRUE))
#autoplot display pop stats by default
p <- autoplot(gs, "CD4")
#it is easy to remove the default stats
p <- p + stats_null()
#and add a new one
p <- p + geom_stats(type = "count")
```
Compute the positions of the population statistics based on the geometric gate centroid for a ggcyto plot.

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

stat_position(gate, ...)

## S3 method for class 'filter'
stat_position(
  gate,
  negated = FALSE,
  adjust = 0.5,
  location = "gate",
  data_range = NULL,
  limits = NULL,
  ...
)

Arguments

gate a flowCore filter
...
other arguments
negated logical indicating whether position needs to be moved to negative side of gate
adjust see details
location see details
data_range a two-row data.frame representing the actual data range. Each column is a a range for a specific channel. First row is min, Second row is max.
limits used to fix the gate range

Details

Specifying location for statistical annotation:
The adjust and location arguments allow for a few different ways to adjust the location of the statistical annotation for a gate on a ggcyto plot. The valid values for location are "gate" (default), "data", "plot", and "fixed".

Relative location:
If location is not "fixed", the starting position of the annotation will be determined with respect to a rectangular window whose bounds are determined in the following way:
  • For location = "gate", the window will be set by the range of the data in the gate
- For `location = "data"`, the window will be set by the range of values in all of the data on the plot (provided by `data_range`).
- For `location = "plot"`, the window will be set by the axis limits of the plot (adjusted by `ggcyto_par_set`).

This starting position can then be adjusted by passing values in a vector to the `adjust` parameter, where they will be interpreted as relative proportions of the window dimension, where 0.0 represents the lower bound of the dimension and 1.0 represents the upper bound. So, for a 2-D plot, `adjust=c(0,0)` places the annotation at the lower left corner of this window and `adjust=c(1,1)` places it at the upper right corner.

As another example, for a 2-D gate, if `location = "gate"` and `adjust=c(0.25, 0.75)`, the statistical annotation will be placed 1/4 of the way across the x-range of the gate and 3/4 of the way across the y-range of the gate.

The `adjust` argument will also accept values less than 0.0 or greater than 1.0. This can be an easy way to simply move the annotation outside of a gate so it does not obstruct the view of the data within. For example, `location == "gate"` and `adjust=c(-0.2, 1.2)` will move the annotation outside of the upper left corner of the gate range.

**Fixed location:**
If `location = "fixed"`, the numeric vector passed to `adjust` will be interpreted as values on the data scales of the plot to provide an explicit location for the annotation. For example, if the annotation should be at the location 3000, 5000 on the plot, that could be done with `location="fixed"` and `adjust = c(3000, 5000)`.

**Default:**
The default behavior if no values are provided to `location` or `adjust` will be to place the annotation at the center of the range of the data in the gate.

**Value**
a data.table of gate centroid coordinates

**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)
stat_position(rect.gates)
```

**Description**
rescale the gate coordinates with the transformation provided
Usage

transform('data', ...)

rescale_gate(gate, trans, param)

Arguments

_data the filter or filterList object. Currently support polygonGate, ellipsoidGate, rectangleGate and quadGate.
...
trans the transformation function or transformList object param the parameter/dimension to be transformed. When trans is transformList object, param is not needed since it is derived from transformList.
gate gate object
trans the transformation function
param the parameter/dimension to be transformed.

Value

the transformed filter/filterList object
Index

+ (ggcyto_add), 24
+ , ggcyto_GatingLayout, ANY-method
  (ggcyto_add), 24
+ , ggcyto_GatingLayout-method
  (ggcyto_add), 24
+ , ggcyto_GatingSet, ANY-method
  (ggcyto_add), 24
+ , ggcyto_GatingSet-method (ggcyto_add), 24
+ , ggcyto_flowSet, ANY-method
  (ggcyto_add), 24
+ , ggcyto_flowSet-method (ggcyto_add), 24
+ , ggcyto_GatingLayout
  (ggcyto_add), 24
+ , ggcyto_GatingSet
  (ggcyto_add), 24
+ , ggcyto_flowSet
  (ggcyto_add), 24
+ , ggcyto_ncdfFlowList
  (ggcyto_add), 24
%+%
  (replace_data), 33
%+%, ggcyto-method
  (replace_data), 33
%+%, ggcyto_GatingLayout, ANY-method
  (replace_data), 33
%+%, ggcyto_GatingLayout-method
  (replace_data), 33
aes, 17, 18
aes_string, 17, 18
alpha, 17, 19
as.ggplot, 3
autoplot (autoplot.flowSet), 4
autoplot.flowSet, 4
axis_x_inverse_trans, 5, 6
axis_y_inverse_trans
  (axis_x_inverse_trans), 6
boundaryFilter, 30
colour, 17, 19
compute_stats, 7
faust_gating_plot, 8
flowCore_asinht_trans, 8
fortify (fortify.cytoframe), 9
fortify.cytoframe, 9
fortify.ellipsoidGate, 10
fortify.filterList, 11
fortify.multiRangeGate, 12
fortifypolygonGate, 12
fortify.rectangleGate, 13
fortify_fs, 14
gate_null, 15
gem_gate, 15
gem_hline, 17
gem_hvline, 17
gem_multi_range, 18
gem_overlay, 19
gem_rect, 18
gem_stats, 20
gem_vline, 17
getFlowFrame, 21
ggcyto (ggcyto-class), 22
ggcyto-class, 22
ggcyto.default (ggcyto-class), 22
ggcyto.flowSet (ggcyto-class), 22
ggcyto.GatingHierarchy (ggcyto-class), 22
ggcyto.GatingSet (ggcyto-class), 22
ggcyto.GatingSetList (ggcyto-class), 22
ggcyto.GatingLayout-class
  (ggcyto-class), 22
ggcyto.GatingSet-class (ggcyto-class), 22
ggcyto_par_default, 26
ggcyto_par_set, 26, 39
group, 17, 19
gs_get_pop_paths, 5
is.ggcyto, 27
is.ggcyto_flowSet, 28
is.ggcyto_par, 28

labs_cyto, 29
layer, 17, 18
linetype, 18, 19
linewidth, 18, 19

marginalFilter, 30
merge.quad.gates, 31

plot.ggcyto (print.ggcyto), 32
print,ggcyto-method (print.ggcyto), 32
print.ggcyto, 32
print.ggcyto_GatingLayout, 32
replace_data, 33
rescale_gate (transform-gate), 39

scale_x_flowCore_fasinh, 36
scale_x_flowJo_biexp
  (scales_flowjo_biexp), 34
scale_x_flowJo_biexp
  (scales_flowjo_biexp), 34
scale_x_flowJo_fasinh
  (scales_flowjo_fasinh), 35
scale_x_flowJo_fasinh
  (scales_flowjo_fasinh), 35

scale_x_logicle, 36
scale_y_flowCore_fasinh
  (scale_x_flowCore_fasinh), 36
scale_y_flowJo_biexp
  (scales_flowjo_biexp), 34
scale_y_flowJo_biexp
  (scales_flowjo_biexp), 34
scale_y_flowJo_fasinh
  (scales_flowjo_fasinh), 35
scale_y_flowJo_fasinh
  (scales_flowjo_fasinh), 35

scale_y_logicle (scale_x_logicle), 36
scales_flowjo_biexp, 34
scales_flowjo_fasinh, 35
show,ggcyto-method (print.ggcyto), 32
show,ggcyto_GatingLayout-method
  (print.ggcyto_GatingLayout), 32
show.ggcyto (print.ggcyto), 32
show.ggcyto_GatingLayout
  (print.ggcyto_GatingLayout), 32
stat_position, 27, 38
stats_null, 37
transform(transform-gate), 39
transform,filter-method
  (transform-gate), 39
transform,filterList-method
  (transform-gate), 39
transform-gate, 39