

# Package ‘flowGraph’

April 26, 2024

**Type** Package

**Title** Identifying differential cell populations in flow cytometry data  
accounting for marker frequency

**Version** 1.11.0

**Description** Identifies maximal differential cell populations in flow cytometry data taking into account dependencies between cell populations; flowGraph calculates and plots SpecEnr abundance scores given cell population cell counts.

**Date** 2019-11-30

**License** Artistic-2.0

**VignetteBuilder** knitr

**Depends** R (>= 4.1)

**Imports** effsize, furrr, future, purrr, ggiraph, ggrepel, ggplot2,  
igraph, Matrix, matrixStats, stats, utils, visNetwork,  
htmlwidgets, grDevices, methods, stringr, stringi, Rdpack,  
data.table (>= 1.9.5), gridExtra,

**Suggests** BiocStyle, dplyr, knitr, rmarkdown, testthat (>= 2.1.0)

**biocViews** FlowCytometry, StatisticalMethod, ImmunoOncology, Software,  
CellBasedAssays, Visualization

**BugReports** <https://github.com/aya49/flowGraph/issues>

**URL** <https://github.com/aya49/flowGraph>

**RoxygenNote** 7.1.1

**RdMacros** Rdpack

**LazyData** no

**git\_url** <https://git.bioconductor.org/packages/flowGraph>

**git\_branch** devel

**git\_last\_commit** 8b9001e

**git\_last\_commit\_date** 2023-10-24

**Repository** Bioconductor 3.19

**Date/Publication** 2024-04-26

**Author** Alice Yue [aut, cre]

**Maintainer** Alice Yue <aya43@sfu.ca>

## Contents

cell_type_layers . . . . .	3
extract_markers . . . . .	4
fg_add_feature . . . . .	5
fg_add_summary . . . . .	6
fg_clean_phen . . . . .	8
fg_clear_features . . . . .	9
fg_clear_summary . . . . .	9
fg_data_fca . . . . .	10
fg_data_pos2 . . . . .	11
fg_data_pos30 . . . . .	11
fg_extract_phenotypes . . . . .	12
fg_extract_raw . . . . .	13
fg_extract_samples . . . . .	14
fg_feat_cumsum . . . . .	15
fg_feat_edge_prop . . . . .	16
fg_feat_edge_specenr . . . . .	17
fg_feat_mean_class . . . . .	18
fg_feat_node_prop . . . . .	19
fg_feat_node_specenr . . . . .	20
fg_get_feature . . . . .	21
fg_get_feature_desc . . . . .	22
fg_get_feature_means . . . . .	23
fg_get_graph . . . . .	24
fg_get_markers . . . . .	25
fg_get_meta . . . . .	25
fg_get_summary . . . . .	26
fg_get_summary_desc . . . . .	29
fg_get_summary_index . . . . .	30
fg_get_summary_tables . . . . .	31
fg_gsub_ids . . . . .	32
fg_gsub_markers . . . . .	33
fg_load . . . . .	34
fg_merge . . . . .	35
fg_merge_samples . . . . .	36
fg_plot . . . . .	37
fg_plot_box . . . . .	40
fg_plot_pVSdiff . . . . .	42
fg_plot_qq . . . . .	44
fg_replace_meta . . . . .	46
fg_rm_feature . . . . .	47
fg_rm_summary . . . . .	48
fg_save . . . . .	49

<i>cell_type_layers</i>	3
fg_save_plots . . . . .	50
fg_set_layout . . . . .	52
fg_summary . . . . .	53
flowGraph . . . . .	55
flowGraph-class . . . . .	58
flowGraphSubset . . . . .	61
flowGraphSubset_summary_adjust . . . . .	64
flowGraphSubset_summary_pars . . . . .	64
fpurrr_map . . . . .	65
get_child . . . . .	65
get_eprop . . . . .	66
get_paren . . . . .	67
get_phen_list . . . . .	67
get_phen_meta . . . . .	68
ggdf . . . . .	69
loop_ind_f . . . . .	70
mean_diff . . . . .	71
ms_create . . . . .	71
ms_psig . . . . .	72
plot_gr . . . . .	73
set_layout_graph . . . . .	74
summary_table . . . . .	75
test_c . . . . .	75
time_output . . . . .	76
tstr . . . . .	77
<b>Index</b>	<b>78</b>

---

<i>cell_type_layers</i>	<i>Determines the layer on which a phenotype resides.</i>
-------------------------	---

---

### Description

Determines the layer on which the given phenotypes reside.

### Usage

```
cell_type_layers(phen)
```

### Arguments

phen	A string vector of phenotype or cell population name labels.
------	--

### Details

Given a vector of phenotypes, returns an equal length vector of the number of markers in each phenotype.

**Value**

A numeric vector with the same length as phen indicating which layer each phenotype resides on.

**See Also**

[get\\_phen\\_list](#) [get\\_phen\\_meta](#)

**Examples**

```
phen <- c('A+B+C-D++', 'A+B-', '', 'B++D-E+')
cell_type_layers(phen)
```

---

extract\_markers

*Extracts markers from cell population phenotypes*

---

**Description**

Extracts all unique markers from cell population phenotypes

**Usage**

```
extract_markers(phen)
```

**Arguments**

phen                    A vector of cell population phenotypes.

**Value**

A vector of unique markers

**See Also**

[str\\_split](#)

---

fg_add_feature	<i>Adds a feature.</i>
----------------	------------------------

---

## Description

Adds a feature created using `feat_fun` from `fg` OR `m` into a given `flowGraph` object. Only use this function if you cannot generate the desired features using the existing `flowGraph` functions starting with `fg_feat_<feature name>`.

## Usage

```
fg_add_feature(
  fg,
  type = "node",
  feature,
  m = NULL,
  feat_fun = NULL,
  overwrite = FALSE,
  ...
)
```

## Arguments

<code>fg</code>	flowGraph object.
<code>type</code>	A string specifying the type of the feature being added i.e. 'node' or 'edge'.
<code>feature</code>	A string indicating the unique name of the feature added.
<code>m</code>	A numeric matrix with feature values; it should contain the same sample id's on row names as in <code>fg_get_meta(fg)\$id</code> and node or edge names as column names (i.e. if <code>m</code> is a node feature, it would have the same column names as those in <code>fg_get_graph(fg)\$v\$phenotype</code> ; if it is an edge feature, its column names should be the same as <code>paste0(fg_get_graph(fg)\$e\$from, '_', fg_get_graph(fg)\$e\$to)</code> ).
<code>feat_fun</code>	A function that outputs a feature matrix as in <code>m</code> given <code>fg</code> and other optional parameters.
<code>overwrite</code>	A logical variable indicating whether or not the function should replace the existing feature with the same name if one is already in <code>fg</code> .
<code>...</code>	Other parameters that would be used as input into <code>feat_fun</code> .

## Details

`fg_add_feature` adds the given new feature matrix to the given `flowGraph` object `fg` updating slots `feat` and `feat_desc`. See [flowGraph-class](#) slot `feat` and `feat_desc` for what should be in these slots. We do not recommend users to directly use this method unless there is a clear understanding on how the row and column names should be specified. Instead, we recommend users to use the functions listed in the "See also" sections prefixed with `"fg_feat_"`.

**Value**

flowGraph object.

**See Also**

[flowGraph-class](#) [fg\\_feat\\_node\\_prop](#) [fg\\_feat\\_node\\_specenr](#) [fg\\_get\\_feature](#) [fg\\_rm\\_feature](#)  
[fg\\_get\\_feature\\_desc](#)

**Examples**

```
no_cores <- 1
data(fg_data_pos30)
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
               prop=FALSE, specenr=FALSE,
               no_cores=no_cores)
fg_get_feature_desc(fg)

fg <- fg_add_feature(fg, type="node", feature="count_copy",
                   m=fg_data_pos30$count)
fg_get_feature_desc(fg)
```

---

fg_add_summary	<i>Adds a feature summary.</i>
----------------	--------------------------------

---

**Description**

Adds a feature summary into a given flowGraph object. Only use this function if your summary statistic cannot be calculated using the [fg\\_summary](#) function.

**Usage**

```
fg_add_summary(
  fg,
  type = "node",
  summary_meta = NULL,
  p = NULL,
  summ_fun = NULL,
  overwrite = FALSE,
  ...
)
```

**Arguments**

fg	flowGraph object.
type	A string indicating feature type the summary was created for; 'node' or 'edge'.

summary_meta	The user must provide type and summary_meta. summary_meta is a list containing feature (feature name), test_name (summary statistic name), class (class), label1, and label2 (class labels compared). See <a href="#">fg_get_summary_desc</a> for details.
p	A list containing summary values; this list contains elements: values (a vector containing summary statistics e.g. p-values; this vector should be named by their associated phenotype or edge name), test_custom (a function of the statistical test used), and adjust_custom (a function of the p-value correction method used). This list must contain the values element.
summ_fun	A function that outputs a feature summary matrix as in p given fg and other optional parameters.
overwrite	A logical variable indicating whether or not the function should replace the existing feature summary with the same name if one is already in fg.
...	Other parameters that would be used as input into summ_fun.

### Details

fg\_add\_summary adds the given feature summary list p or the output of the given function summ\_fun to the given flowGraph object fg updating slots summary and summary\_desc. See [flowGraph-class](#) slot summary and summary\_desc for what should be in these slots. We do not recommend users directly use this function unless what is required is duly in the above slots is well understood — note these slots are used in plotting functions e.g. [fg\\_plot](#). We instead recommend users to use the [fg\\_summary](#) function.

### Value

flowGraph object.

### See Also

[flowGraph-class](#) [fg\\_summary](#) [fg\\_get\\_summary](#) [fg\\_rm\\_summary](#) [fg\\_get\\_summary\\_desc](#) [fg\\_add\\_feature](#)

### Examples

```
no_cores <- 1
data(fg_data_pos30)
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
               no_cores=no_cores)

# get samples that we are going to compare
m <- fg_get_feature(fg, type="node", feature="prop")
m1_ <- m[fg_data_pos30$meta$class=="control",,drop=FALSE]
m2_ <- m[fg_data_pos30$meta$class=="exp",,drop=FALSE]

# define test or summary function to conduct comparison
test_custom <- function(x,y)
  tryCatch(stats::t.test(x,y)$p.value, error=function(e) 1)
values_p <- sapply(seq_len(ncol(m)), function(j)
  test_custom(m1_[,j], m2_[,j]) )
```

```

values_p <- p.adjust(values_p , method="BY")

# the user can choose to fill either parameter "p" or "summ_fun",
# the latter of which must output a list with the same elements as "p".
# see documentation for ?flowGraph-class, slot "summary" for
# details on what should be in "p".
p <- list(values=values_p, test_fun=test_custom, adjust_fun="BY")
fg <- fg_add_summary(fg, type="node", summary_meta=list(
  feature="prop", test_name="wilcox_BY",
  class="class", label1="control", label2="exp"), p=p)

fg_get_summary_desc(fg)

```

---

fg\_clean\_phen

*Reformats phenotype*


---

## Description

Reformats cell population phenotypes into flowGraph format

## Usage

```
fg_clean_phen(phen, markers = NULL)
```

## Arguments

phen	Vector of cell population phenotype names as character strings.
markers	markers extracted from phen.

## Value

Vector with the same length as phen containing reformatted and not necessarily changed cell population phenotype names.

## See Also

[str\\_extract](#), [str\\_split](#)

## Examples

```
# fg_clean_phen(c("A+_B+", "B+_notC", "A-_C"))
```



---

fg_clear_features	<i>Clears all features in a flowGraph object.</i>
-------------------	---

---

**Description**

Returns a flowGraph object with only the count feature.

**Usage**

```
fg_clear_features(fg)
```

**Arguments**

fg                      flowGraph object.

**Value**

flowGraph object with only the count node feature.

**See Also**

[flowGraph-class](#)

**Examples**

```
no_cores <- 1
data(fg_data_pos30)
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
               no_cores=no_cores)

fg <- fg_clear_features(fg)
fg_get_summary_desc(fg)
```

---

fg_clear_summary	<i>Removes all summary statistics.</i>
------------------	--

---

**Description**

Removes all summary statistics in a flowGraph object; we recommend doing this to save space.

**Usage**

```
fg_clear_summary(fg)
```

**Arguments**

fg flowGraph object.

**Value**

flowGraph object with an empty summary slot.

**See Also**

[flowGraph-class fg\\_summary](#)

**Examples**

```
no_cores <- 1
data(fg_data_pos30)
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
                prop=FALSE, specenr=FALSE,
                no_cores=no_cores, node_features="count")
fg_get_summary_desc(fg)

fg <- fg_clear_summary(fg)
fg_get_summary_desc(fg)
```

---

fg\_data\_fca

*fg\_data\_fca*


---

**Description**

fg\_data\_fca

**Usage**

fg\_data\_fca

**Format**

A list containing the following elements derived from the flowCAP-II AML data set for cell populations up to layer 3.

- count: A numeric sample x cell population node matrix with cell count values.
- meta: A data frame containing meta information on samples in count; it contains columns:
  - class: a string indicating whether a sample is from a "control" or "aml" subject.
  - id: a string containing sample id's.
  - train: a logical variable indicating whether a sample is from the train or test set.
  - subject: a numeric variable containing the id of the subject from whom the sample came from.
  - tube: the tube or panel number; all samples in this data set is analyzed under the 6th panel.

**Source**

Aghaeepour N, Finak G, Hoos H, Mosmann TR, Brinkman R, Gottardo R, Scheuermann RH, Consortium F, Consortium DREAM, others (2013). “Critical assessment of automated flow cytometry data analysis techniques.” *Nature methods*, **10**(3), 228–238.

---

fg\_data\_pos2

fg\_data\_pos2

---

**Description**

fg\_data\_pos2

**Usage**

fg\_data\_pos2

**Format**

A list containing the following elements for a positive control data set with markers A, B, C, D. This is a positive control data set where node A+B+C+ increased by 50

- count: A numeric sample x cell population node matrix with cell count values
- meta: A data frame containing meta information on samples in count; it contains columns:
  - id: a string containing sample id’s.
  - class: a string indicating whether a sample is from a "control" or "exp" (experiment) subject.

---

fg\_data\_pos30

fg\_data\_pos30

---

**Description**

fg\_data\_pos30

**Usage**

fg\_data\_pos30

**Format**

A list containing the following elements for a positive control data set with markers A, B, C, D; note it was made with two and three thresholds for markers A and B to test functions with multiple thresholds (this is a positive control data set where nodes A+..B+..C+ increased by 50

- count: A numeric sample x cell population node matrix with cell count values
- meta: A data frame containing meta information on samples in count; it contains columns:
  - id: a string containing sample id's.
  - class: a string indicating whether a sample is from a "control" or "exp" (experiment) subject.

---

`fg_extract_phenotypes` *Extracts a set of phenotypes from a flowGraph object.*

---

**Description**

Extracts or removes a specified set of phenotypes from a flowGraph object.

**Usage**

```
fg_extract_phenotypes(fg, phenotypes)
```

**Arguments**

<code>fg</code>	flowGraph object.
<code>phenotypes</code>	A string vector of phenotype or cell population name labels.

**Details**

The summary in fg will not be modified; we recommend users recalculate them.

**Value**

flowGraph object.

**See Also**

[flowGraph-class](#) [fg\\_get\\_feature\\_desc](#) [fg\\_merge](#) [fg\\_extract\\_samples](#) [fg\\_merge\\_samples](#)

**Examples**

```

no_cores <- 1
data(fg_data_pos30)
fg0 <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
                 prop=FALSE, specenr=FALSE,
                 no_cores=no_cores)
fg_get_feature_desc(fg0)

fg <- fg_extract_phenotypes(fg0, fg_get_graph(fg0)$v$phenotype[1:10])
fg_get_feature_desc(fg)

```

---

fg_extract_raw	<i>Clears all features and feature summaries in a flowGraph object.</i>
----------------	---

---

**Description**

Returns a flowGraph object with only the count feature and meta data. This function clears all other features and feature summaries to save space.

**Usage**

```
fg_extract_raw(fg)
```

**Arguments**

fg                      flowGraph object.

**Value**

flowGraph object with all summary statistics and feature values removed except for the node count feature.

**See Also**

[flowGraph-class](#)

**Examples**

```

no_cores <- 1
data(fg_data_pos30)
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
                 no_cores=no_cores)

fg <- fg_extract_raw(fg)
show(fg)

```

---

fg_extract_samples	<i>Extracts a set of samples from a flowGraph object.</i>
--------------------	---

---

## Description

Extracts or removes a specified set of samples from a flowGraph object.

## Usage

```
fg_extract_samples(fg, sample_ids, rm_summary = TRUE)
```

## Arguments

fg	flowGraph object.
sample_ids	A string vector of sample id's that the user wants to keep in fg.
rm_summary	A logical indicating whether or not to clear summary.

## Details

The summaries in fg will not be modified; we recommend the user recalculates them.

## Value

flowGraph object.

## See Also

[flowGraph-class](#) [fg\\_get\\_feature\\_desc](#) [fg\\_merge](#) [fg\\_extract\\_phenotypes](#)

## Examples

```
no_cores <- 1
data(fg_data_pos30)
fg0 <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
                 prop=FALSE, specenr=FALSE,
                 no_cores=no_cores)
fg_get_feature_desc(fg0)

fg <- fg_extract_samples(fg0, fg_get_meta(fg0)$id[1:5])
fg_get_feature_desc(fg)
```

---

fg_feat_cumsum	<i>Converts cell counts into cumulated cell counts.</i>
----------------	---

---

## Description

Converts the cell counts in a flowGraph object into cumulated cell counts; this is optional and can be done only if there is more than one threshold for one or more markers. This should also only be run when initializing a flowGraph object as converting back and forth is computationally expensive. If the user is interested in seeing non- and cumulated counts, we recommend keeping two flowGraph objects, one for each version. This function simply converts e.g. the count of A+ or A++ into the sum of count of A+, A++, and A+++ or A++, and A+++.

## Usage

```
fg_feat_cumsum(fg, no_cores)
```

## Arguments

fg	flowGraph object.
no_cores	An integer indicating how many cores to parallelize on.

## Details

fg\_feat\_cumsum returns the given flowGraph object with an adjusted count feature. As in our example,

## Value

flowGraph object with cumulated counts.

## See Also

[flowGraph-class Matrix](#)

## Examples

```
no_cores <- 1
data(fg_data_pos30)
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
                prop=FALSE, specenr=FALSE,
                no_cores=no_cores)

fg <- flowGraph::fg_feat_cumsum(fg, no_cores=no_cores)
```

---

fg_feat_edge_prop	<i>Generates the proportion edge feature.</i>
-------------------	---

---

## Description

Generates the proportion edge feature and returns it inside the flowGraph object.

## Usage

```
fg_feat_edge_prop(fg, no_cores = 1, overwrite = FALSE)
```

## Arguments

fg	flowGraph object.
no_cores	An integer indicating how many cores to parallelize on.
overwrite	A logical variable indicating whether to overwrite the existing proportion edge feature if it exists.

## Details

Given a flowGraph object, fg\_feat\_edge\_prop returns the same flowGraph object with an additional proportions prop edge feature and its meta data. The proportions feature is made using the node count feature and is the cell count of each cell population (e.g. A+B+) over the cell count of its parent (e.g. A+); each edge then corresponds with such a relationship. The edge feature matrix has column names <from>\_<to> e.g. A+\_A+B+.

## Value

flowGraph object containing the proportion edge feature.

## See Also

[flowGraph-class](#) [fg\\_feat\\_node\\_prop](#) [fg\\_feat\\_node\\_specnr](#) [fg\\_add\\_feature](#) [fg\\_get\\_feature](#)  
[fg\\_rm\\_feature](#) [fg\\_get\\_feature\\_desc](#)

## Examples

```
no_cores <- 1
data(fg_data_pos30)
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
               prop=FALSE, specnr=FALSE,
               no_cores=no_cores)

fg <- fg_feat_edge_prop(fg)
```



---

fg\_feat\_edge\_specenr    *Generates the SpecEnr edge feature.*

---

## Description

Generates the SpecEnr edge feature and returns it inside the flowGraph object.

## Usage

```
fg_feat_edge_specenr(fg, no_cores = 1, overwrite = FALSE)
```

## Arguments

fg	flowGraph object.
no_cores	An integer indicating how many cores to parallelize on.
overwrite	A logical variable indicating whether to overwrite the existing proportion edge feature if it exists.

## Details

Given a flowGraph object, fg\_feat\_edge\_SpecEnr returns the same flowGraph object with an additional SpecEnr and expected proportions expect\_prop edge feature and its meta data. The expected proportions edge feature is calculated by taking the ratio of the child nodes' (e.g. A+B+) expected proportion value over its parent nodes' (e.g. A+) actual proportion value. The SpecEnr feature is the actual over expected proportion ratio, logged. The edge feature matrix has column names <from>\_<to> e.g. A+\_A+B+.

## Value

flowGraph object containing the proportion edge feature.

## See Also

[flowGraph-class](#) [fg\\_feat\\_node\\_prop](#) [fg\\_feat\\_node\\_specenr](#) [fg\\_add\\_feature](#) [fg\\_get\\_feature](#) [fg\\_rm\\_feature](#) [fg\\_get\\_feature\\_desc](#)

## Examples

```
no_cores <- 1
data(fg_data_pos30)
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
               prop=FALSE, specenr=FALSE,
               no_cores=no_cores)

fg <- fg_feat_edge_specenr(fg)
```

---

fg_feat_mean_class	<i>Normalizes all features for class.</i>
--------------------	---

---

### Description

For each class label in column class of meta, fg\_feat\_mean\_class takes the column mean of the rows in the given feature matrices (as specified in node\_features and edge\_features) associated with that class; it then takes the difference point by point between these means and the original rows for that class.

FUNCTION\_DESCRIPTION

### Usage

```
fg_feat_mean_class(
  fg,
  class,
  no_cores = 1,
  node_features = NULL,
  edge_features = NULL
)
```

### Arguments

fg	PARAM_DESCRIPTION
class	a column name in fg_get_meta(fg) indicating the meta data that should be used as the class label of each sample while conducting normalization.
no_cores	An integer indicating how many cores to parallelize on.
node_features	A string vector indicating the node features to perform normalization on; set as NULL to normalize all.
edge_features	A string vector indicating the edge features to perform normalization on; set as NULL to normalize all.

### Details

For all features in the given flowGraph object and for each class label in column class of meta, fg\_feat\_mean\_class. It takes the column mean of the rows in the given feature matrices (as specified in node\_features and edge\_features) associated with that class; it then takes the difference point by point between these means and the original rows for that class. fg\_feat\_mean\_class

### Value

A numeric matrix whose dimensions equate to that of the input and whose values are normalized per class.

flowGraph object with normalized features.

**See Also**[flowGraph-class](#)**Examples**

```

no_cores <- 1
data(fg_data_pos30)
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
               prop=FALSE, specenr=FALSE,
               no_cores=no_cores)

fg <- fg_feat_mean_class(fg, class="class", node_features="count",
                       no_cores=no_cores)

```

---

fg_feat_node_prop	<i>Generates the proportion node feature.</i>
-------------------	---

---

**Description**

Generates the proportion node feature and returns it inside the returned flowGraph object.

**Usage**

```
fg_feat_node_prop(fg, overwrite = FALSE)
```

**Arguments**

fg	flowGraph object.
overwrite	A logical variable indicating whether to overwrite the existing proportion node feature if it exists.

**Details**

Given a flowGraph object, fg\_feat\_node\_prop returns the same flowGraph object, inside of which is an additional proportions prop node feature and its meta data. The proportions feature is made using the node count feature and is the cell count of each cell population over the total cell count.

**Value**

flowGraph object containing the proportion node feature.

**See Also**

[flowGraph-class](#) [fg\\_feat\\_node\\_specenr](#) [fg\\_add\\_feature](#) [fg\\_get\\_feature](#) [fg\\_rm\\_feature](#)  
[fg\\_get\\_feature\\_desc](#)

**Examples**

```
no_cores <- 1
data(fg_data_pos30)
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
                prop=FALSE, specenr=FALSE,
                no_cores=no_cores)

fg <- fg_feat_node_prop(fg)
```

---

`fg_feat_node_specenr`    *Generates the SpecEnr node feature.*

---

**Description**

Generates the SpecEnr node feature and returns it inside the returned flowGraph object.

**Usage**

```
fg_feat_node_specenr(fg, no_cores = 1, feature = "prop", overwrite = FALSE)
```

**Arguments**

<code>fg</code>	flowGraph object
<code>no_cores</code>	An integer indicating how many cores to parallelize on.
<code>feature</code>	A string indicating feature name; this is the feature SpecEnr will be calculated on.
<code>overwrite</code>	A logical variable indicating whether to overwrite the existing SpecEnr node feature if it exists.

**Details**

Given a flowGraph object, `fg_feat_node_specenr` returns the same flowGraph object with an additional SpecEnr and expect\_prop node feature and its meta data. The expected proportions feature is made using the prop node and edge features; therefore, the returned flowGraph will also contain these two features. For details on how these feature is calculated.

**Value**

flowGraph object containing the SpecEnr node feature.

**References**

Yue A, Chauve C, Libbrecht M, Brinkman R (2019). "Identifying differential cell populations in flow cytometry data accounting for marker frequency." *BioRxiv*, 837765.

**See Also**

[flowGraph-class](#) [fg\\_feat\\_node\\_prop](#) [fg\\_add\\_feature](#) [fg\\_get\\_feature](#) [fg\\_rm\\_feature](#) [fg\\_get\\_feature\\_desc](#)

**Examples**

```
no_cores <- 1
data(fg_data_pos30)
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
               prop=FALSE, specenr=FALSE,
               no_cores=no_cores)

# SpecEnr is by default calculated based on proportions
fg <- fg_feat_node_specenr(fg, no_cores=no_cores)

# SpecEnr can be calculated for other feature values too
fg <- fg_feat_node_specenr(fg, feature="count")

show(fg)
```

---

fg_get_feature	<i>Retrieves a feature matrix.</i>
----------------	------------------------------------

---

**Description**

Retrieves a feature matrix from a given flowGraph object, the feature type, and feature name.

**Usage**

```
fg_get_feature(fg, type = "node", feature = "count")
```

**Arguments**

fg	flowGraph object.
type	A string indicating feature type 'node' or 'edge'.
feature	A string indicating feature name;

**Details**

Returns NULL if the requested feature does not exist.

**Value**

A numeric matrix of the specified feature values.

**See Also**

[flowGraph-class](#) [fg\\_get\\_feature\\_desc](#) [fg\\_add\\_feature](#) [fg\\_rm\\_feature](#) [fg\\_get\\_summary](#)

**Examples**

```
data(fg_data_pos30)
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
               prop=FALSE, specenr=FALSE,
               no_cores=1)

feature_matrix <- fg_get_feature(fg, type='node', feature='count')
```

---

fg_get_feature_desc	<i>Retrieves and/or recalculates a feature description table.</i>
---------------------	---

---

**Description**

Retrieves and/or recalculates a feature description table for a given flowGraph object.

**Usage**

```
fg_get_feature_desc(fg, re_calc = FALSE)
```

**Arguments**

fg	flowGraph object.
re_calc	A logical variable specifying whether or not a feature summary should be recalculated or directly retrieved from fg.

**Value**

A data frame where each row contains information on a feature from the given flowGraph object; its columns is as in the feat\_desc slot of [flowGraph-class](#).

**See Also**

[flowGraph-class](#) [fg\\_get\\_feature](#) [fg\\_add\\_feature](#) [fg\\_rm\\_feature](#) [fg\\_get\\_summary\\_desc](#)

**Examples**

```
no_cores <- 1
data(fg_data_pos30)
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
               no_cores=no_cores)

fg_get_feature_desc(fg, re_calc=TRUE)
```

---

fg\_get\_feature\_means     *Retrieves feature summaries.*

---

### Description

Retrieves a feature summary (e.g. colMeans) for samples specified by sample id's id OR class label label for class class given a feature specified by type and feat.

### Usage

```
fg_get_feature_means(  
  fg,  
  type = c("node", "edge"),  
  feature = "count",  
  class = NULL,  
  label = NULL,  
  id = NULL,  
  summary_fun = colMeans  
)
```

### Arguments

fg	flowGraph object.
type	A string indicating feature type the summary was created for 'node' or 'edge'.
feature	A string indicating feature name the summary was created for;
class	A string corresponding to a column name of the meta slot of fg whose values represent the class label of each sample on which the summary was created to compare or analyze;
label	A string indicating a class label.
id	A string vector containing the sample id's corresponding to the id column of the meta slot of fg.
summary_fun	A function that takes in a matrix and outputs a vector the same length as the number of columns this matrix has.

### Value

A list containing two numeric vectors calculated using the summary\_fun function on the subset of samples specified by sample id's id OR class label label for class class from a feature matrix specified by type and feat.

### See Also

[flowGraph-class](#) [fg\\_get\\_summary\\_desc](#) [fg\\_add\\_summary](#) [fg\\_rm\\_summary](#) [fg\\_get\\_summary](#)

**Examples**

```
no_cores <- 1
data(fg_data_pos30)
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
               no_cores=no_cores)
fg <- fg_summary(fg, no_cores=no_cores, class="class", label1="control",
               overwrite=FALSE, test_name="t", diminish=FALSE)
show(fg)
feat_mean <- fg_get_feature_means(fg, type="node", feature="count",
                                class="class", label="control")
```

fg\_get\_graph

*Retrieves a graph list from a given flowGraph object.***Description**

Retrieves a graph list from a given flowGraph object.

**Usage**

```
fg_get_graph(fg)
```

**Arguments**

fg                      flowGraph object.

**Value**

A list containing two data frames (v and lcodee) from the graph slot of the given flowGraph object containing information on the cell populations phenotype nodes and edges representing relation between cell populations.

**See Also**

[flowGraph-class](#) [fg\\_plot](#) [ggdf](#) [plot\\_gr](#)

**Examples**

```
no_cores <- 1
data(fg_data_pos30)
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
               prop=FALSE, specenr=FALSE,
               no_cores=no_cores)
gr <- fg_get_graph(fg)
head(gr$v)
head(gr$e)
```



---

fg_get_markers	<i>Retrieves the markers from a given flowGraph object.</i>
----------------	---

---

**Description**

Retrieves the markers from a given flowGraph object.

**Usage**

```
fg_get_markers(fg)
```

**Arguments**

fg                      flowGraph object.

**Value**

A character vector containing the markers used in a flowGraph object.

**See Also**

[flowGraph-class](#)

**Examples**

```
no_cores <- 1
data(fg_data_pos30)
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
                prop=FALSE, specenr=FALSE,
                no_cores=no_cores)
fg_get_markers(fg)
```

---

fg_get_meta	<i>Retrieves sample meta.</i>
-------------	-------------------------------

---

**Description**

Retrieves sample meta from a given flowGraph object.

**Usage**

```
fg_get_meta(fg)
```

**Arguments**

fg                      flowGraph object.

**Value**

A data frame containing sample meta data.

**See Also**

[flowGraph-class fg\\_replace\\_meta](#)

**Examples**

```
no_cores <- 1
data(fg_data_pos30)
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
                prop=FALSE, specenr=FALSE,
                no_cores=no_cores)
head(fg_get_meta(fg))
```

---

fg_get_summary	<i>Retrieves a summary statistic.</i>
----------------	---------------------------------------

---

**Description**

Retrieves a summary statistic from a given flowGraph object; while fg is required, the user can choose to input parameters summary\_meta, index, or all of type, feat, test\_name, class, label1, and label2. See [fg\\_get\\_summary\\_desc](#) for details.

**Usage**

```
fg_get_summary(
  fg,
  type = "node",
  index = NULL,
  summary_meta = NULL,
  adjust_custom = "byLayer",
  SpecEnr_filt = TRUE,
  summary_fun = colMeans,
  adjust0_lim = c(-0.1, 0.1),
  filter_adjust0 = 1,
  filter_es = 0,
  filter_btwn_tpthres = 0.05,
  filter_btwn_es = 0.5,
  default_p_thres = 1
)
```

**Arguments**

fg	flowGraph object.
type	A string indicating feature type the summary was created for 'node' or 'edge'.
index	The user must provide type and additionally, one of summary_meta or index. index is an integer indicating the row in fg_get_summary_desc(<flowGraph>) of the corresponding type and summary the user would like to retrieve.
summary_meta	The user must provide type and additionally, one of summary_meta or index. summary_meta is a list containing feat (feature name), test_name (summary statistic name), class (class), label1, and label2 (class labels compared). See <a href="#">fg_get_summary_desc</a> for details.
adjust_custom	A function or a string indicating the test adjustment method to use. If a string is provided, it should be one of c("holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none") (see p.adjust.methods). If a function is provided, it should take as input a numeric vector and output the same vector adjusted.
SpecEnr_filt	A logicle indicating whether or not to filter p-values for SpecEnr.
summary_fun	A function that takes in a matrix and outputs a vector the same length as the number of columns this matrix has. Set to NULL to not calculate this summary (i.e. returned list will not contain m1 and m2). See <a href="#">fg_get_feature_means</a> .
adjust0_lim	A vector of two numeric values indicating a range around 0, default set to -0.1 and 0.1.
filter_adjust0	A numeric variable indicating what percentage of SpecEnr values compared (minimum) should be not close to 0. Set to 1 to not conduct filtering. Original p-values stored in values_original.
filter_es	A numeric variable between 0 and 1 indicating what the Cohen's D value of the nodes/edges in question must be greater or equal to, to be significant.
filter_btwn_tpthres	A numeric variable between 0 and 1 indicating the unadjusted T-test p-value threshold used to test whether the actual and expected feature values used to calculate the specified SpecEnr feature are significantly different for each sample class. Note this only needs to be specified for SpecEnr features. Combined with filter_btwn_es, we conduct three tests to understand if there is an actual large difference between actual and expected features: (1,2) T-test of significance between the actual and expected raw feature value (e.g. proportion) for samples in each of the compared classes, (3) and the T-test of significance between the differences of actual and expected feature values of the two classes. If any two of the three tests come out as insignificant, we set the p-value for the associated node/edge to 1.
filter_btwn_es	A numeric variable between 0 and 1 indicating what the Cohen's D value of the nodes/edges in question must be greater or equal to, to be significant – see filter_btwn_tpthres.
default_p_thres	A numeric variable indicating the p-value threshold user is using. Currently, all nodes/edges not passing the filter criterion will be defaulted to 1; if this parameter is set, then all of these nodes/edges will be set to a minimum of default_p_thres.

**Value**

A list containing elements on feature summary retrieved by the user as in the summary slot of `flowGraph-class`. If `summary_fun` is not NULL, this list also includes:

- `m1`: a numeric vector the same length as `values`; this is a summary of the samples compared e.g. mean.
- `m2`: a numeric vector the same length as `values`; this is a summary of the samples compared e.g. mean.
- `cohensd`: a numeric vector indicating cohen's d values considering effect size.
- `cohensd_size`: a factor vector interpreting cohen's d values.
- `adjust0`: a numeric vector indicating the percentage of samples that have a `SpecEnr` value in the range of `adjust0_lim` around 0; if there are two classes of samples being compared, we output the smaller percentage between the two classes.
- `btwn`: a data frame containing columns:
  - `tpv1`: unadjusted p-value calculated between the actual and expected raw feature values of class 1.
  - `tpv2`: unadjusted p-value calculated between the actual and expected raw feature values of class 2.
  - `cd1`: Cohen's D between the actual and expected raw feature values of class 1.
  - `cd2`: Cohen's D between the actual and expected raw feature values of class 2.
  - `btp`: unadjusted p-value calculated between the difference between actual and expected raw feature of the two classes.
  - `bcd`: Cohen's D calculated between the difference between actual and expected raw feature of the two classes.
  - `btp_`: unadjusted p-value calculated between the log ratio between actual and expected raw feature of the two classes.
  - `bcd_`: Cohen's D calculated between the log ratio between actual and expected raw feature of the two classes.

**See Also**

`flowGraph-class` `fg_get_feature_means` `fg_get_summary_desc` `fg_add_summary` `fg_rm_summary` `fg_get_feature`

**Examples**

```
no_cores <- 1
data(fg_data_pos30)
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
               no_cores=no_cores)

# set features to NULL to apply summary statistic to all features.
fg <- fg_summary(fg, no_cores=no_cores, class="class", label1="control",
               overwrite=FALSE, test_name="t", diminish=FALSE,
               node_features=NULL, edge_features=NULL)

show(fg)
```

```
feat_summ <- fg_get_summary(fg, type="node", summary_meta=list(
  feature="SpecEnr", test_name="t", class="class",
  label1="control", label2="exp"))
```

---

`fg_get_summary_desc`     *Retrieves a feature summary description table.*

---

### Description

Retrieves a feature summary description table for a given flowGraph object.

### Usage

```
fg_get_summary_desc(fg)
```

### Arguments

`fg`                      flowGraph object.

### Value

A data frame where each row contains information on a feature summary from fg:

- `type`: feature type (i.e. 'node' or 'edge').
- `feat`: feature name.
- `test_name`: summary name.
- `class`: class or the column name of `fg_get_meta(fg)` whose values represent the class label of each sample on which the summary was created for.
- `label1`: A string from the class column of the meta slot indicating the label of samples compared.
- `label2`: A string from the class column of the meta slot indicating the label of samples compared.

### See Also

[flowGraph-class](#) [fg\\_get\\_summary](#) [fg\\_add\\_summary](#) [fg\\_rm\\_summary](#) [fg\\_get\\_feature\\_desc](#)

### Examples

```
no_cores <- 1
data(fg_data_pos30)
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
  no_cores=no_cores)

fg_get_summary_desc(fg)
```

---

fg\_get\_summary\_index    *Retrieves the index of the requested summary.*

---

## Description

Retrieves the index of the requested summary from a given flowGraph object.

## Usage

```
fg_get_summary_index(fg, type = "node", index = NULL, summary_meta = NULL)
```

## Arguments

fg	flowGraph object.
type	A string indicating feature type the summary was created for 'node' or 'edge'.
index	The user must provide type and additionally, one of summary_meta or index. index is an integer indicating the row in fg_get_summary_desc(<flowGraph>) of the corresponding type and summary the user would like to retrieve.
summary_meta	The user must provide type and additionally, one of summary_meta or index. summary_meta is a list containing type (feature type: node or edge), feature (feature name), test_name (summary statistic name), class (class), label1, and label2 (class labels compared). See <a href="#">fg_get_summary_desc</a> for details.

## Value

An integer analagous to index. If both index and summary\_meta are NULL, returns 1.

## See Also

[flowGraph-class](#) [fg\\_get\\_summary\\_desc](#) [fg\\_add\\_summary](#) [fg\\_rm\\_summary](#) [fg\\_plot](#)

## Examples

```
no_cores <- 1
data(fg_data_pos30)
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
               no_cores=no_cores)

# set features to NULL to apply summary statistic to all features.
fg <- fg_summary(fg, no_cores=no_cores, class="class", label1="control",
               overwrite=FALSE, test_name="t", diminish=FALSE,
               node_features=NULL, edge_features=NULL)

show(fg)

index <- flowGraph::fg_get_summary_index(
  fg, type="node", summary_meta=list(
    feature="SpecEnr", test_name="t", class="class",
    label1="control", label2="exp"))
```

---

`fg_get_summary_tables` *Retrieves a table containing all node or edge summary statistics.*

---

## Description

Retrieves a table containing all node or edge summary statistics given a flowGraph object.

## Usage

```
fg_get_summary_tables(fg, type = "node")
```

## Arguments

<code>fg</code>	flowGraph object.
<code>type</code>	A string indicating feature type the summaries the user wants to retrieve were created for, 'node' or 'edge'.

## Value

A list; this output is the same as that of function `fg_get_graph` with additional columns. These columns contain summary statistics from the summary slot of the flowGraph object. These columns are named: <feature type: node/edge>.<feature>.<summary name>.<class>.<class labels>.

## See Also

[flowGraph-class](#) [fg\\_get\\_feature\\_means](#) [fg\\_get\\_summary\\_desc](#) [fg\\_add\\_summary](#) [fg\\_rm\\_summary](#)  
[fg\\_get\\_summary](#) [fg\\_get\\_feature](#)

## Examples

```
no_cores <- 1
data(fg_data_pos30)
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
               no_cores=no_cores)

fg <- fg_summary(fg, no_cores=no_cores, class="class", label1="control",
               overwrite=FALSE, test_name="t", diminish=FALSE)
show(fg)

feat_summ_table_node <- fg_get_summary_tables(fg, type="node")
head(feat_summ_table_node)
```

---

fg_gsub_ids	<i>Replace sample id's.</i>
-------------	-----------------------------

---

### Description

Replace sample id's in a flowGraph object.

### Usage

```
fg_gsub_ids(fg, ids_new, ids_old = NULL)
```

### Arguments

fg	flowGraph object.
ids_new	A string vector of new sample id's; if ids_old is set to NULL, each id in ids_new should correspond to each id in fg_get_meta(fg)\$id.
ids_old	A string vector of old sample id's the user wants to replace; these marker names corresponding to those in fg_get_meta(fg)\$id with the same length as ids_new. If ids_old=NULL, ids_new should be the same length as fg_get_meta(fg)\$id.

### Value

flowGraph object with sample id's replaced.

### See Also

[flowGraph-class fg\\_get\\_feature\\_desc fg\\_gsub\\_markers](#)

### Examples

```
no_cores <- 1
data(fg_data_pos30)
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
                prop=FALSE, specenr=FALSE,
                no_cores=no_cores)

fg <- fg_gsub_ids(fg, ids_new=paste0(fg_get_meta(fg)$id, "_new"))
```



---

fg_gsub_markers	<i>Replace marker names.</i>
-----------------	------------------------------

---

## Description

Replace marker names in a flowGraph object.

## Usage

```
fg_gsub_markers(fg, markers_new, markers_old = NULL)
```

## Arguments

fg	flowGraph object.
markers_new	A string vector of new marker names; if markers_old is set to NULL, each marker in markers_new should correspond to each marker in the markers slot of the flowGraph object.
markers_old	A string vector of old marker names user wants to replace; these marker names corresponding to those in fg_get_markers(fg) with the same length as markers_new. If markers_old=NULL, markers_new should be the same length as fg_get_markers(fg).

## Value

flowGraph object with marker names replaced.

## See Also

[flowGraph-class fg\\_gsub\\_ids](#)

## Examples

```
no_cores <- 1
data(fg_data_pos30)
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
               prop=FALSE, specenr=FALSE,
               no_cores=no_cores)

fg <- fg_gsub_markers(fg, c("Anew", "Bnew", "Cnew", "Dnew"))
fg_get_feature_desc(fg)
```

---

fg_load	<i>Load a flowGraph object from a specified folder path.</i>
---------	--

---

## Description

Load a flowGraph object from a specified folder path.

## Usage

```
fg_load(folder_path)
```

## Arguments

folder_path	A string indicating the folder path to where a flowGraph object was saved using the fg_save function.
-------------	---

## Details

see function fg\_save

## Value

flowGraph object

## See Also

[fg\\_save](#)

## Examples

```
no_cores <- 1
data(fg_data_pos2)
fg <- flowGraph(fg_data_pos2$count, class=fg_data_pos2$meta$class,
               no_cores=no_cores)

fg_save(fg, "tmp")
fg <- fg_load("tmp")
```

---

fg_merge	<i>Merges two flowGraph objects together.</i>
----------	---

---

## Description

Merges two flowGraph objects together.

## Usage

```
fg_merge(  
  fg1,  
  fg2,  
  method_sample = c("union", "intersect", "setdiff", "none"),  
  method_phenotype = c("intersect", "setdiff", "none")  
)
```

## Arguments

fg1                    flowGraph object.

fg2                    flowGraph object.

method\_sample    A string indicating how samples from flowGraph objects should be merged:

- union: keep all samples from both flowGraph objects; in this case method\_phenotype must be intersect.
- intersect: keep only samples that exist in both fg1 and fg2.
- setdiff: keep only samples that exist in fg1 and not in fg2.
- none: keep all samples in fg1.

method\_phenotype    A string indicating how phenotypes from flowGraph objects should be merged:

- intersect: keep only phenotypes that exist in both fg1 and fg2.
- setdiff: keep only phenotypes that exist in fg1 and not in fg2.
- none: keep all phenotypes in fg1.

## Details

fg\_merge is a generic function that merges the samples and phenotypes of two flowGraph objects. Note that if method\_sample="union" then method\_phenotype must be set to "intersect".

## Value

flowGraph object.

## See Also

[flowGraph-class](#) [fg\\_extract\\_samples](#) [fg\\_extract\\_phenotypes](#) [fg\\_merge\\_samples](#)

**Examples**

```

no_cores <- 1
data(fg_data_pos30)
fg0 <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
                 prop=FALSE, specenr=FALSE,
                 no_cores=no_cores)

fg1 <- fg_extract_samples(fg0, fg_get_meta(fg0)$id[1:5])
fg2 <- fg_extract_samples(fg0, fg_get_meta(fg0)$id[4:7])
fg <- fg_merge(fg1, fg2, method_sample="intersect",
               method_phenotype="intersect")
fg_get_feature_desc(fg)

```

---

fg_merge_samples	<i>Merges the samples from two flowGraph objects.</i>
------------------	---

---

**Description**

Merges the samples from two flowGraph objects together; we recommend removing all summary statistics from the new flowGraph object as those won't be adjusted: [fg\\_clear\\_summary](#).

**Usage**

```
fg_merge_samples(fg1, fg2)
```

**Arguments**

fg1	flowGraph object.
fg2	flowGraph object.

**Details**

Appends the samples from fg2 onto those in fg1. This function requires that the two flowGraph objects must have the same phenotypes. Therefore, we recommend users to use, instead, [fg\\_merge](#).

**Value**

flowGraph object.

**See Also**

[flowGraph-class](#) [fg\\_get\\_feature\\_desc](#) [fg\\_merge](#) [fg\\_extract\\_samples](#)

**Examples**

```

no_cores <- 1
data(fg_data_pos30)
fg0 <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
                 prop=FALSE, specenr=FALSE,
                 no_cores=no_cores)

fg1 <- fg_extract_samples(fg0, fg_get_meta(fg0)$id[1:5])
fg2 <- fg_extract_samples(fg0, fg_get_meta(fg0)$id[4:7])
fg <- fg_merge_samples(fg1, fg2)
fg_get_feature_desc(fg)

```

---

fg_plot	<i>Creates a cell hierarchy plot.</i>
---------	---------------------------------------

---

**Description**

Creates a cell hierarchy plot given a flowGraph object. If a path is not provided for fg\_plot to save the plot, please use plot\_gr to view plot given the output of fg\_plot.

**Usage**

```

fg_plot(
  fg,
  type = "node",
  index = 1,
  summary_meta = NULL,
  adjust_custom = "byLayer",
  show_nodes_edges = NULL,
  label_max = 30,
  p_thres = 0.05,
  filter_adjust0 = 1,
  filter_es = 0,
  filter_btwn_tpthres = 1,
  filter_btwn_es = 0,
  node_labels = c("prop", "expect_prop"),
  summary_fun = colMeans,
  layout_fun = NULL,
  show_bgedges = TRUE,
  main = NULL,
  interactive = FALSE,
  visNet_plot = TRUE,
  path = NULL,
  width = 9,
  height = 9
)

```

**Arguments**

fg	flowGraph object.
type	A string indicating feature type the summary was created for 'node' or 'edge'.
index	The user must provide type and additionally, one of summary_meta or index. index is an integer indicating the row in fg_get_summary_desc(<flowGraph>) of the corresponding type and summary the user would like to retrieve.
summary_meta	The user must provide type and additionally, one of summary_meta or index. summary_meta is a list containing feature (feature name), test_name (summary statistic name), class (class), label1, and label2 (class labels compared). See fg_get_summary_desc for details.
adjust_custom	A function or a string indicating the test adjustment method to use. If a string is provided, it should be one of c("holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none") (see p.adjust.methods). If a function is provided, it should take as input a numeric vector and output the same vector adjusted.
show_nodes_edges	A logical vector indicating which nodes/edges (type) to show in the plot; if this is not specified, only nodes/edges with significant summary statistics will be shown.
label_max	An integer specifying the maximum number of nodes to label.
p_thres	A double indicating a summary statistic threshold e.g. if we are plotting a T test summary statistic, we can set the threshold to .05; nodes with a p-value greater than .05 will not be plotted.
filter_adjust0	A numeric variable indicating what percentage of SpecEnr values compared (minimum) should be not close to 0. Set to 1 to not conduct filtering.
filter_es	A numeric variable between 0 and 1 indicating what the Cohen's D value of the nodes/edges in question must be greater or equal to, to be significant.
filter_btwn_tpthres	A numeric variable between 0 and 1 indicating the unadjusted T-test p-value threshold used to test whether the actual and expected feature values used to calculate the specified SpecEnr feature are significantly different for each sample class. Note this only needs to be specified for SpecEnr features. Combined with filter_btwn_es, we conduct three tests to understand if there is an actual large difference between actual and expected features: (1,2) T-test of significance between the actual and expected raw feature value (e.g. proportion) for samples in each of the compared classes, (3) and the T-test of significance between the differences of actual and expected feature values of the two classes. If any two of the three tests come out as insignificant, we set the p-value for the associated node/edge to 1.
filter_btwn_es	A numeric variable between 0 and 1 indicating what the Cohen's D value of the nodes/edges in question must be greater or equal to, to be significant – see filter_btwn_tpthres.
node_labels	A string vector indicating which node feature(s) should be used to label a node. We recommend keeping the length of this vector to below 2. Set to "NONE" if no p-value labels are needed.

summary_fun	A function that takes in a matrix and outputs a vector the same length as the number of columns this matrix has; see <a href="#">fg_summary</a> .
layout_fun	A string representing a function from the <code>igraph</code> package that indicates what layout should be used if a cell hierarchy is to be plotted; all such functions have prefix <code>layout_</code> . Only specify if different from the default one already calculated in the <code>fg</code> <code>flowGraph</code> object given.
show_bgedges	A logical variable indicating whether or not edges not specified for plotting should be plotted as light grey in the background.
main	A string or the title of the plot; if left as <code>NULL</code> , a default title will be applied.
interactive	A logical variable indicating whether the plot should be an interactive plot; see package <code>ggiraph</code> .
visNet_plot	A logical variable indicating if an interactive plot is chosen, if function should output a <code>visNetwork</code> plot; if set to <code>FALSE</code> , <code>ggplot</code> 's <code>girafe</code> will be used instead.
path	A string indicating the path to where the function should save the plot; leave as <code>NULL</code> to not save the plot. Static plots are saved as <code>PNG</code> , interactive plots are saved as <code>HTML</code> .
width	A numeric variable specifying, in inches, what the plot width should be.
height	A numeric variable specifying, in inches, what the plot height should be.

### Details

`fg_plot` takes a `flowGraph` object as input and returns the graph slot of the given object with additional columns to serve as input into `plot_gr` for plotting using functions in the `ggplot2` package. Users can choose to save a `PNG` version of the plot by filling out the `path` parameter with a full path to the `PNG` plot. In addition to specifying columns added from `ggdf`, `fg_plot` also adds label column(s) whose values serve as labels in the interactive version of the plot.

### Value

A list of nodes and edges for plotting with the `plot_gr` function. Other elements in this list include `show_bgedges`, which has the same value as parameter `show_bgedges`, and `main`, the title of the plot.

### See Also

[flowGraph-class](#) [get\\_phen\\_meta](#) [ggdf](#) [plot\\_gr](#) [fg\\_get\\_feature](#) [fg\\_get\\_summary](#)

### Examples

```
no_cores <- 1
data(fg_data_pos2)
fg <- flowGraph(fg_data_pos2$count, class=fg_data_pos2$meta$class,
               no_cores=no_cores)

gr <- fg_plot(fg, type="node", index=1, label_max=30,
             show_nodes_edges=NULL, p_thres=.01, node_labels=c("prop", "expect_prop"),
             path=NULL) # set path to a full path to save plot as a PNG
# plot_gr(gr)
```

---

fg_plot_box	<i>Creates a boxplot of the values of one node/edge</i>
-------------	---

---

## Description

Creates a boxplot comparing the features of samples belonging to different classes corresponding to an existing summary statistic using ggplot2.

## Usage

```
fg_plot_box(
  fg,
  type = "node",
  index = 1,
  summary_meta = NULL,
  node_edge = 1,
  adjust_custom = "byLayer",
  p_thres = 0.05,
  filter_adjust0 = 0.5,
  filter_es = 0.5,
  filter_btwn_tpthres = 0.05,
  filter_btwn_es = 0.5,
  paired = FALSE,
  dotplot = TRUE,
  outlier = TRUE,
  all_labels = FALSE,
  show_mean = TRUE,
  main = NULL,
  path = NULL
)
```

## Arguments

fg	flowGraph object.
type	A string indicating feature type the summary was created for 'node' or 'edge'.
index	The user must provide type and additionally, one of summary_meta or index. index is an integer indicating the row in fg_get_summary_desc(<flowGraph>) of the corresponding type and summary the user would like to retrieve.
summary_meta	The user must provide type and additionally, one of summary_meta or index. summary_meta is a list containing feature (feature name), test_name (summary statistic name), class (class), label1, and label2 (class labels compared). See <a href="#">fg_get_summary_desc</a> for details.
node_edge	An integer/index of or a string of the cell population (node) / edge name (edge) the user wants to plot.



adjust_custom	A function or a string indicating the test adjustment method to use. If a string is provided, it should be one of c("holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none") (see p.adjust.methods). If a function is provided, it should take as input a numeric vector and output the same vector adjusted.
p_thres	A numeric variable indicating a p-value threshold
filter_adjust0	A numeric variable indicating what percentage of SpecEnr values compared (minimum) should be not close to 0. Set to 1 to not conduct filtering.
filter_es	A numeric variable between 0 and 1 indicating what the Cohen's D value of the nodes/edges in question must be greater or equal to, to be significant.
filter_btwn_tpthres	A numeric variable between 0 and 1 indicating the unadjusted T-test p-value threshold used to test whether the actual and expected feature values used to calculate the specified SpecEnr feature are significantly different for each sample class. Note this only needs to be specified for SpecEnr features. Combined with filter_btwn_es, we conduct three tests to understand if there is an actual large difference between actual and expected features: (1,2) T-test of significance between the actual and expected raw feature value (e.g. proportion) for samples in each of the compared classes, (3) and the T-test of significance between the differences of actual and expected feature values of the two classes. If any two of the three tests come out as insignificant, we set the p-value for the associated node/edge to 1.
filter_btwn_es	A numeric variable between 0 and 1 indicating what the Cohen's D value of the nodes/edges in question must be greater or equal to, to be significant – see filter_btwn_tpthres.
paired	A logical indicating whether the summary is paired.
dotplot	A logical indicating whether or not to plot sample points.
outlier	A logical indicating whether or not outliers should be plotted.
all_labels	A logical indicating whether or not to plot samples of all classes outside of just those used in the summary statistic test.
show_mean	A logical indicating whether or not to label the mean.
main	A string or the title of the plot; if left as NULL, a default title will be applied.
path	A string indicating the path to where the function should save the plot; leave as NULL to not save the plot. Static plots are saved as PNG.

## Details

The plot is made using the ggplot2 package. The interactive version is the same as the static version, it is only here to support the shiny app.

## Value

A static boxplot.

## See Also

[flowGraph-class](#) [fg\\_plot](#) [plot\\_gr](#) [fg\\_get\\_feature](#) [fg\\_get\\_summary](#) [fg\\_plot\\_qq](#)

## Examples

```
no_cores <- 1
data(fg_data_pos2)
fg <- flowGraph(fg_data_pos2$count, class=fg_data_pos2$meta$class,
               no_cores=no_cores)

fg_plot_box(fg, type="node", summary_meta=NULL, adjust_custom="byLayer", index=1, node_edge=10)
```

---

fg_plot_pVSdiff	<i>Creates a p value vs feature difference plot</i>
-----------------	---

---

## Description

Creates a p value vs feature difference plot where the difference is that of the features of samples belonging to different classes corresponding to an existing summary statistic.

## Usage

```
fg_plot_pVSdiff(
  fg,
  type = "node",
  index = 1,
  summary_meta = NULL,
  adjust_custom = "byLayer",
  logged = TRUE,
  label_max = 5,
  p_thres = 0.05,
  filter_adjust0 = 1,
  filter_es = 0,
  filter_btwn_tpthres = 1,
  filter_btwn_es = 0,
  shiny_plot = FALSE,
  nodes_max = 30,
  main = NULL,
  interactive = FALSE,
  path = NULL
)
```

## Arguments

fg	flowGraph object.
type	A string indicating feature type the summary was created for 'node' or 'edge'.
index	The user must provide type and additionally, one of summary_meta or index. index is an integer indicating the row in fg_get_summary_desc(<flowGraph>) of the corresponding type and summary the user would like to retrieve.

summary_meta	The user must provide type and additionally, one of summary_meta or index. summary_meta is a list containing feature (feature name), test_name (summary statistic name), class (class), label1, and label2 (class labels compared). See <a href="#">fg_get_summary_desc</a> for details.
adjust_custom	A function or a string indicating the test adjustment method to use. If a string is provided, it should be one of c("holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none") (see p.adjust.methods). If a function is provided, it should take as input a numeric vector and output the same vector adjusted.
logged	A logical indicating whether or not to log the summary statistic p value.
label_max	An integer indicating the maximum number of max difference and/or min p value nodes/edges that should be labelled.
p_thres	A numeric variable indicating a p-value threshold; a line will be plotted at this threshold.
filter_adjust0	A numeric variable indicating what percentage of SpecEnr values compared (minimum) should be not close to 0. Set to 1 to not conduct filtering.
filter_es	A numeric variable between 0 and 1 indicating what the Cohen's D value of the nodes/edges in question must be greater or equal to, to be significant.
filter_btwn_tpthres	A numeric variable between 0 and 1 indicating the unadjusted T-test p-value threshold used to test whether the actual and expected feature values used to calculate the specified SpecEnr feature are significantly different for each sample class. Note this only needs to be specified for SpecEnr features. Combined with filter_btwn_es, we conduct three tests to understand if there is an actual large difference between actual and expected features: (1,2) T-test of significance between the actual and expected raw feature value (e.g. proportion) for samples in each of the compared classes, (3) and the T-test of significance between the differences of actual and expected feature values of the two classes. If any two of the three tests come out as insignificant, we set the p-value for the associated node/edge to 1.
filter_btwn_es	A numeric variable between 0 and 1 indicating what the Cohen's D value of the nodes/edges in question must be greater or equal to, to be significant – see filter_btwn_tpthres.
shiny_plot	A logical indicating whether this plot is made for shiny; users don't need to change this.
nodes_max	An integer indicating maximum number of nodes to plot; this limit is set for interactive plots only.
main	A string or the title of the plot; if left as NULL, a default title will be applied.
interactive	A logical variable indicating whether the plot should be an interactive plot; see package ggiraph.
path	A string indicating the path to where the function should save the plot; leave as NULL to not save the plot. Static plots are saved as PNG.

## Details

The interactive plot is made using the ggiraph package.

**Value**

A static or interactive p value vs difference plot.

**See Also**

[flowGraph-class](#) [fg\\_plot](#) [plot\\_gr](#) [fg\\_get\\_feature](#) [fg\\_get\\_summary](#) [fg\\_plot\\_qq](#)

**Examples**

```
no_cores <- 1
data(fg_data_pos2)
fg <- flowGraph(fg_data_pos2$count, class=fg_data_pos2$meta$class,
               no_cores=no_cores)

gp <- fg_plot_pVSDiff(fg, type="node", summary_meta=NULL,
                    adjust_custom="byLayer", index=1, label_max=10)
```

---

fg\_plot\_qq

---

*Creates a QQ plot of a summary statistic.*


---

**Description**

Creates a QQ plot of a summary statistic.

**Usage**

```
fg_plot_qq(
  fg,
  type = "node",
  index = 1,
  summary_meta = NULL,
  adjust_custom = "byLayer",
  logged = TRUE,
  p_thres = 0.05,
  filter_adjust0 = 1,
  filter_es = 0,
  filter_btwn_tpthres = 1,
  filter_btwn_es = 0,
  shiny_plot = FALSE,
  main = NULL,
  interactive = FALSE,
  path = NULL
)
```

**Arguments**

fg	flowGraph object.
type	A string indicating feature type the summary was created for 'node' or 'edge'.
index	The user must provide type and additionally, one of summary_meta or index. index is an integer indicating the row in fg_get_summary_desc(<flowGraph>) of the corresponding type and summary the user would like to retrieve.
summary_meta	The user must provide type and additionally, one of summary_meta or index. summary_meta is a list containing feature (feature name), test_name (summary statistic name), class (class), label1, and label2 (class labels compared). See <a href="#">fg_get_summary_desc</a> for details.
adjust_custom	A function or a string indicating the test adjustment method to use. If a string is provided, it should be one of c("holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none") (see p.adjust.methods). If a function is provided, it should take as input a numeric vector and output the same vector adjusted.
logged	A logical indicating whether or not to log the summary statistic p value.
p_thres	A double indicating a summary statistic threshold e.g. if we are plotting a T-test summary statistic, we can set the threshold to .05; nodes with a p-value greater than .05 will not be plotted.
filter_adjust0	A numeric variable indicating what percentage of SpecEnr values compared (minimum) should be not close to 0. Set to 1 to not conduct filtering.
filter_es	A numeric variable between 0 and 1 indicating what the Cohen's D value of the nodes/edges in question must be greater or equal to, to be significant.
filter_btwn_tpthres	A numeric variable between 0 and 1 indicating the unadjusted T-test p-value threshold used to test whether the actual and expected feature values used to calculate the specified SpecEnr feature are significantly different for each sample class. Note this only needs to be specified for SpecEnr features. Combined with filter_btwn_es, we conduct three tests to understand if there is an actual large difference between actual and expected features: (1,2) T-test of significance between the actual and expected raw feature value (e.g. proportion) for samples in each of the compared classes, (3) and the T-test of significance between the differences of actual and expected feature values of the two classes. If any two of the three tests come out as insignificant, we set the p-value for the associated node/edge to 1.
filter_btwn_es	A numeric variable between 0 and 1 indicating what the Cohen's D value of the nodes/edges in question must be greater or equal to, to be significant – see filter_btwn_tpthres.
shiny_plot	A logical indicating whether this plot is made for shiny; users don't need to change this.
main	A string or the title of the plot; if left as NULL, a default title will be applied.
interactive	A logical indicating whether or not plot should be an interactive ggiraph plot as opposed to a static plot.

**path** A string indicating the path to where the function should save the plot; leave as NULL to not save the plot. Static plots are saved as PNG, interactive plots are saved as HTML.

### Details

The interactive plot is made using the ggiraph package.

### Value

A static or interactive qq plot.

### See Also

[flowGraph-class](#) [fg\\_plot](#) [plot\\_gr](#) [fg\\_get\\_feature](#) [fg\\_get\\_summary](#)

### Examples

```
no_cores <- 1
data(fg_data_pos2)
fg <- flowGraph(fg_data_pos2$count, class=fg_data_pos2$meta$class,
                no_cores=no_cores)

fg_plot_qq(fg, type="node", summary_meta=NULL, adjust_custom="byLayer", index=1,
           interactive=TRUE, logged=FALSE)

fg_plot_qq(fg, type="node", summary_meta=NULL, adjust_custom="byLayer", index=1,
           interactive=FALSE, logged=FALSE)
```

---

fg_replace_meta	<i>Replaces sample meta.</i>
-----------------	------------------------------

---

### Description

Replaces sample meta in a given flowGraph object.

### Usage

```
fg_replace_meta(fg, meta)
```

### Arguments

**fg** flowGraph object.

**meta** A data frame containing meta data; see details in [flowGraph-class](#).

### Value

A flowGraph object with an updated sample meta.

See Also

[flowGraph-class fg\\_get\\_meta](#)

Examples

```
no_cores <- 1
data(fg_data_pos30)
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
               prop=FALSE, specenr=FALSE,
               no_cores=no_cores)
head(fg_get_meta(fg))

new_df <- fg_data_pos30$meta
new_df$id[1] <- "newID"

fg <- fg_replace_meta(fg, new_df)
head(fg_get_meta(fg))
```

---

fg_rm_feature	<i>Removes a feature.</i>
---------------	---------------------------

---

Description

Removes a feature from a given flowGraph object.

Usage

```
fg_rm_feature(fg, type = "node", feature = NULL)
```

Arguments

- fg                    flowGraph object.
- type                A string specifying the type of the feature being removed i.e. 'node' or 'edge'.
- feature            A string indicating the unique name of the feature removed; note we cannot remove the 'node' 'count' feature type.

Details

fg\_rm\_feature removes a specified feature matrix from the given flowGraph object fg updating slots feat and feat\_desc. See [flowGraph-class](#) slot feat and feat\_desc for what should be in these slots.

Value

flowGraph object with specified feature removed.

**See Also**

[flowGraph-class](#) [fg\\_add\\_feature](#) [fg\\_get\\_feature](#) [fg\\_get\\_feature\\_desc](#) [fg\\_rm\\_summary](#)

**Examples**

```
no_cores <- 1
data(fg_data_pos30)
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
               no_cores=no_cores)
fg_get_feature_desc(fg)

fg <- fg_rm_feature(fg, type="node", feature="prop")
fg_get_feature_desc(fg)
```

---

fg_rm_summary	<i>Removes a feature summary.</i>
---------------	-----------------------------------

---

**Description**

Removes a feature summary from a given flowGraph object; while fg is required, the user can choose to input parameters summary\_meta, index, or all of type, feat, test\_name, class, label1, and label2. See [fg\\_get\\_summary\\_desc](#) for details.

**Usage**

```
fg_rm_summary(fg, type = "node", index = NULL, summary_meta = NULL)
```

**Arguments**

fg	flowGraph object.
type	A string indicating feature type the summary was created for; 'node' or 'edge'.
index	The user must provide type and additionally, one of summary_meta or index. index is an integer indicating the row in fg_get_summary_desc(<flowGraph>) of the corresponding type and summary the user would like to retrieve.
summary_meta	The user must provide type and additionally, one of summary_meta or index. summary_meta is a list containing feat (feature name), test_name (summary statistic name), class (class), label1, and label2 (class labels compared). See <a href="#">fg_get_summary_desc</a> for details.

**Value**

flowGraph object.

**See Also**

[flowGraph-class](#) [fg\\_get\\_summary](#) [fg\\_add\\_summary](#) [fg\\_get\\_summary\\_desc](#) [fg\\_rm\\_feature](#)



## Examples

```
no_cores <- 1
data(fg_data_pos30)
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
               prop=FALSE, specenr=FALSE,
               no_cores=no_cores)

fg <- fg_summary(fg, no_cores=no_cores, class="class", label1="control",
               overwrite=FALSE, test_name="wilcox_byLayer", diminish=FALSE,
               node_features=NULL, edge_features=NULL)
fg_get_summary_desc(fg)

fg <- fg_rm_summary(fg, summary_meta=c(
  feature="count", test_name="wilcox_byLayer",
  class="class", label1="control", label2="exp"))
fg_get_summary_desc(fg)
```

---

fg_save	<i>Saves flowGraph object to a specified path.</i>
---------	--

---

## Description

Saves flowGraph object to a specified path.

## Usage

```
fg_save(fg, folder_path = NULL, save_plots = TRUE, paired = FALSE, ...)
```

## Arguments

fg	flowGraph object to save.
folder_path	A string indicating the folder path to where the flowGraph object should save its elements; if this is the first time the object is being saved, this folder should be empty or if it is not yet created, the function will create it. If the object has previously been saved before and this parameter is set to NULL, the function will save the object into the save folder it was previously saved in.
save_plots	A logical indicating whether or not to save plots.
paired	A logical indicating whether the summary is paired; used in function fg_plot_box.
...	Other parameters for the fg_save_plots function.

## Details

See generated README.md file.

## Value

TRUE if flowGraph object successfully saved.

**See Also**

length,c("nrow", "nrow"),NULL [map](#)

**Examples**

```
no_cores <- 1
data(fg_data_pos2)
fg <- flowGraph(fg_data_pos2$count, class=fg_data_pos2$meta$class,
               no_cores=no_cores)

fg_save(fg, "tmp")
```

---

fg\_save\_plots

---

*Saves numerous plots for all summary statistics to a folder.*


---

**Description**

Saves numerous plots for all summary statistics in a given flowGraph object to a user specified folder.

**Usage**

```
fg_save_plots(
  fg,
  plot_path,
  plot_types = "node",
  interactive = FALSE,
  adjust_custom = "byLayer",
  label_max = 10,
  box_no = 20,
  paired = FALSE,
  logged = FALSE,
  filter_adjust0 = 1,
  filter_es = 0,
  filter_btwn_tpthres = 1,
  filter_btwn_es = 0,
  overwrite = TRUE,
  node_labels = "NONE",
  ...
)
```

**Arguments**

fg                      flowGraph object.

plot\_path              A string indicating the folder path to where the function should save the plots.

plot_types	A string or a vector of strings indicating what feature types and their summaries the function should plot for: 'node' or 'edge'.
interactive	A logical indicating whether the QQ plot, p-value vs difference plot, and the cell hierarchy plots should be interactive; see functions fg_plot and fg_plot_qq.
adjust_custom	A function or a string indicating the test adjustment method to use. If a string is provided, it should be one of c("holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none") (see p.adjust.methods). If a function is provided, it should take as input a numeric vector and output the same vector adjusted.
label_max	An integer indicating how many labels should be shown in the functions fg_plot_pVSdiff and fg_plot.
box_no	An integer indicating the maximum number of boxplots to save; used in function fg_plot_box.
paired	A logical indicating whether the summary is paired; used in function fg_plot_box.
logged	A logical indicating whether or not to log the summary statistic p value in the qq plots.
filter_adjust0	A numeric variable indicating what percentage of SpecEnr values compared (minimum) should be not close to 0. Set to 1 to not conduct filtering. This parameter is used for the QQ and the pVSdifference plots.
filter_es	A numeric variable between 0 and 1 indicating what the Cohen's D value of the nodes/edges in question must be greater or equal to, to be significant.
filter_btwn_tpthres	A numeric variable between 0 and 1 indicating the unadjusted T-test p-value threshold used to test whether the actual and expected feature values used to calculate the specified SpecEnr feature are significantly different for each sample class. Note this only needs to be specified for SpecEnr features. Combined with filter_btwn_es, we conduct three tests to understand if there is an actual large difference between actual and expected features: (1,2) T-test of significance between the actual and expected raw feature value (e.g. proportion) for samples in each of the compared classes, (3) and the T-test of significance between the differences of actual and expected feature values of the two classes. If any two of the three tests come out as insignificant, we set the p-value for the associated node/edge to 1.
filter_btwn_es	A numeric variable between 0 and 1 indicating what the Cohen's D value of the nodes/edges in question must be greater or equal to, to be significant – see filter_btwn_tpthres.
overwrite	A logical variable indicating whether or not to replace old plots if they exist under the same folder name.
node_labels	Parameter for the fg_plot function.
...	Other parameters for the fg_plot function.

## Details

The interactive plots are made using the ggiraph package.

**Value**

No return; plots are saved to file.

**See Also**

[flowGraph-class](#) [fg\\_plot](#) [plot\\_gr](#) [fg\\_get\\_feature](#) [fg\\_get\\_summary](#) [fg\\_plot\\_qq](#) [fg\\_plot\\_pVSdiff](#)  
[fg\\_plot\\_box](#)

**Examples**

```
no_cores <- 1
data(fg_data_pos2)
fg <- flowGraph(fg_data_pos2$count,
                class=fg_data_pos2$meta$class,
                no_cores=no_cores)

fg_save_plots(fg, "temp")
```

---

fg\_set\_layout

*Determines cell hierarchy layout.*


---

**Description**

Determines cell hierarchy layout and returns the X, Y coordinate of each cell population. This function is a wrapper for [set\\_layout\\_graph](#).

**Usage**

```
fg_set_layout(fg, layout_fun = "layout.reingold.tilford")
```

**Arguments**

fg	flowGraph object.
layout_fun	A string version of a function name from the igraph package that indicates what layout should be used if a cell hierarchy is to be plotted; all such functions have prefix layout_ e.g. layout_fun="layout.reingold.tilford".

**Details**

Given a flowGraph object, modifies the graph slot such that it contains X, Y axes for each node in accordance to a user specified layout.

**Value**

flowGraph object with coordinate meta data on cell populations and edges for plotting use.

## Examples

```
no_cores <- 1
data(fg_data_pos30)
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
                prop=FALSE, specenr=FALSE,
                no_cores=no_cores)

fg <- fg_set_layout(fg)
head(fg_get_graph(fg)$v)
```

---

fg\_summary

*Calculates feature summary statistics.*


---

## Description

Calculates feature summary statistics for flowGraph features; users can choose from a list of statistical significance tests/adjustments or define custom summary functions. For special cases, see example in function [fg\\_add\\_summary](#) on how to manually calculate summary statistics without using this function.

## Usage

```
fg_summary(
  fg,
  no_cores = 1,
  class = "class",
  label1 = NULL,
  label2 = NULL,
  class_labels = NULL,
  node_features = "SpecEnr",
  edge_features = "NONE",
  test_name = "t_diminish",
  diminish = TRUE,
  p_thres = 0.05,
  p_rate = 2,
  test_custom = "t",
  effect_size = TRUE,
  adjust0 = TRUE,
  adjust0_lim = c(-0.1, 0.1),
  btwn = TRUE,
  btwn_test_custom = "t",
  save_functions = FALSE,
  overwrite = FALSE
)
```

**Arguments**

fg	flowGraph object.
no_cores	An integer indicating how many cores to parallelize on.
class	A string corresponding to the column name or index of fg_get_meta(fg) whose values represent the class label of each sample.
label1	A string from the class column of the meta slot indicating one of the labels compared to create the summary statistic. If you would like to compare all other class labels against one label, set label2 to NULL. If you would like to compare all labels against all labels, set label1 and label2 to NULL.
label2	A string from the class column of the meta slot indicating one of the labels compared to create the summary statistic.
class_labels	A list of vectors, each containing two strings representing labels to compare; this parameter is an alternative to parameters label1 and label2 that supports multiple label pairings.
node_features	A string vector indicating which node feature(s) to perform summary statistics on; set to NULL or "NONE" and the function will perform summary statistics on all or no node features.
edge_features	A string vector indicating which edge feature(s) to perform summary statistics on; set to NULL or "NONE" and the function will perform summary statistics on all or no edge features.
test_name	A string with the name of the test you are performing.
diminish	A logical variable indicating whether to use diminishing summary statistics; if TRUE, a summary statistic for a node or edge will only be done if at least one of its parent node or edge is significant. Otherwise, the test will be performed on all nodes or edges.
p_thres	A double indicating the summary statistic threshold; if the result of a statistical test is greater than p_thres, then it is insignificant.
p_rate	A double; if diminish=TRUE, then p_rate needs to be specified. to determine whether or not a node or edge's parent is significant, we use p_thres. However, the higher the layer on which a node resides or to which an edge points to, the less stringent this p_thres should be. Therefore, we set p_thres as the threshold for the parent node or edge of the last layer and multiply p_thres by p_rate for each increasing layer e.g. given default values and 4 layers, the thresholds for layers 1 through 4 would be .4, .2, .1, and .05.
test_custom	A function or a string indicating the statistical test to use. If a string is provided, it should be one of c("t", "wilcox", "ks", "var", "chisq"); these correspond to statistical tests stats::t.test, stats::wilcox.test, and so on. If a function is provided, it should take as input two numeric vectors and output a numeric variable.
effect_size	A logical variable indicating whether or not to calculate effect size statistic (cohen's d) for this set of class labels; later used for plotting.
adjust0	A logical variable indicating whether or not to calculate the minimum percentage of values from samples of each class label that falls within the range of adjust0_lim. This is only done for SpecEnr values as p-values become unstable when comparing near 0 values.

adjust0_lim	A vector of two numeric values indicating a range around 0, default set to -0.1 and 0.1.
btwn	A logical variable indicating whether or not to calculate the btwn data frame given in the fg_get_summary function.
btwn_test_custom	Same as test_custom but for btwn.
save_functions	A logical variable indicating whether to save test and adjust functions.
overwrite	A logical variable indicating whether to overwrite the existing summary statistics if it exists.

### Details

fg\_summary calculates a summary statistic as specified by the user in parameters test\_name, diminish (p\_thres, p\_rate), and test\_custom. The test is done for a node or edge feature of interest within a given flowGraph object as specified by parameters node\_features, edge\_features. It then returns information on the summary statistic inside the same flowGraph object and returns it to the user. See [flowGraph-class](#) slot summary for details on the contents.

### Value

flowGraph object containing calculated summary statistics.

### See Also

[flowGraph-class fg\\_clear\\_summary](#)

### Examples

```
no_cores <- 1
data(fg_data_pos30)
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
               prop=FALSE, specenr=FALSE,
               no_cores=no_cores)
fg_get_summary_desc(fg)

fg <- fg_summary(fg, no_cores=no_cores, class="class", label1="control",
               overwrite=FALSE, test_name="t", diminish=FALSE,
               node_features="count", edge_features="NONE")
fg_get_summary_desc(fg)
```

## Description

Initializes a flowGraph object given the cell counts for one or more flow cytometry sample(s). The flowGraph object returned holds meta data for each sample, each cell population node, edges representing how each cell population node relate to one another, and features for these nodes and edges.

## Usage

```
flowGraph(
  input_,
  meta = NULL,
  class = "class",
  no_cores = 1,
  markers = NULL,
  layout_fun = "layout.reingold.tilford",
  max_layer = NULL,
  cumsumpos = FALSE,
  prop = TRUE,
  specenr = TRUE,
  path = NULL,
  calculate_summary = TRUE,
  node_features = "SpecEnr",
  edge_features = "NONE",
  test_name = "t_diminish",
  test_custom = "t",
  diminish = TRUE,
  label1 = NULL,
  label2 = NULL,
  save_plots = FALSE
)
```

## Arguments

input_	Any of the following: <ul style="list-style-type: none"> <li>a numeric matrix or vector of the cell counts; its column/names must be the phenotype names and its rownames must be sample ID's.</li> </ul> All input samples should have the same markers and partitionsPerMarker.
meta	A data frame with meta data for each Phenotypes or sample; One of its column names should be "id" whose values correspond to the name of each Phenotypes object. We also recommend for it to have a column named "class" where one of its unique values is "control".
class	A string corresponding to the column name or index of meta whose values represent the class label of each sample; OR a vector the same length as the the number of samples in input_ specifying the class of each given sample — this vector will be appended to meta under column name class.
no_cores	An integer indicating how many cores to parallelize on.
markers	A string vector of marker names used in input_.



layout_fun	A string of a function from the igraph package that indicates what layout should be used if a cell hierarchy is to be plotted; all such functions have prefix layout_. This is defaulted to e.g. layout_fun="layout.reingold.tilford".
max_layer	And integer indicating the maximum layer in the cell hierarchy to analyze; set to 'NULL' to analyze all layers.
cumsumpos	A logical variable indicating whether or not to cumulate cell counts; this applies only when partitionsPerMarker > 3 and will convert e.g. the count of A+ or A++ into the sum of the counts of A+, A++, A+++, ..., or A++, A+++, ... .
prop	A logical variable indicating whether or not to calculate the proportion feature; this can be done later on with flowGraph_prop.
specenr	logical variable: whether or not to calculate the SpecEnr feature, Default: T
path	A string indicating the folder path to where the flowGraph object should save its elements, Default = NULL (don't save).
calculate_summary	A logical variable indicating whether or not to calculate the summary statistics for SpecEnr based on default parameters using the fg_summary summary function on class specified in parameter class.
node_features	A string vector indicating which node feature(s) to perform summary statistics on; set to NULL or "NONE" and the function will perform summary statistics on all or no node features.
edge_features	A string vector indicating which edge feature(s) to perform summary statistics on; set to NULL or "NONE" and the function will perform summary statistics on all or no edge features.
test_name	A string with the name of the test you are performing.
test_custom	See <a href="#">fg_summary</a> .
diminish	A logical variable; applicable if calculate_summary is TRUE; see <a href="#">fg_summary</a> .
label1	A string indicating a class label in fg_get_meta(fg)[,class]; set to NULL if you would like to compare all classes against all classes; applicable if calculate_summary is TRUE.
label2	A string indicating a class label in fg_get_meta(fg)[,class]; applicable if calculate_summary is TRUE.
save_plots	A logical indicating whether or not to save plots.

## Details

flowGraph is the constructor for the flowGraph object. The user can choose to input as input\_ a vector, a Phenotypes object (meaning there is only one sample), a matrix, or a Phenotypes object list. If the user is also inputting a sample meta data frame, it must contain a id column corresponding to sample names.

## Value

flowGraph object

**See Also**

flowGraph-class fg\_get\_feature fg\_get\_feature\_desc fg\_get\_summary fg\_get\_summary\_desc  
 fg\_add\_feature fg\_rm\_feature fg\_add\_summary fg\_rm\_summary fg\_gsub\_markers fg\_gsub\_ids  
 fg\_merge\_samples fg\_extract\_samples fg\_extract\_phenotypes fg\_merge registerDoParallel  
 Matrix

**Examples**

```
no_cores <- 1
samplen <- 10
meta_file <- data.frame(
  id=1:samplen,
  class=append(rep("control", samplen/2), rep("exp", samplen/2)),
  stringsAsFactors=FALSE
)

## using the constructor -----

data(fg_data_pos30)

# input: vector of load-able Phenotypes paths
fg <- flowGraph(fg_data_pos30$count[1,], no_cores=no_cores)

# input: matrix + vector of class corresponding to samples
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
  no_cores=no_cores)
# - save to file directly
# fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
#   no_cores=no_cores, path="path_to_folder")

# input: matrix + meta data frame
# fg <- flowGraph(fg_data_pos30$count, meta=fg_data_pos30$meta,
#   no_cores=no_cores)
```

---

flowGraph-class	<i>'flowGraph': A class for storing cell count feature values for the Phenotype class.</i>
-----------------	--

---

**Description**

'flowGraph': A class for storing cell count feature values for the Phenotype class.

**Usage**

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

**Arguments**

object                    A flowGraph object.

**Value**

a flowGraph object.

**Methods (by generic)**

- show: show method

**Slots**

**feat** A list containing elements node and edge, each containing a list with feature values; each element in this list is named by the feature name and contains a numeric matrix with the sample id's as row names and cell populations phenotype labels or edge labels as column names. Column names for edge features are labelled as <from>\_<to> e.g. A+\_A+B+.

**feat\_desc** A list containing elements node and edge, each containing a data frame describing the features in the feat slot with columns:

- feat: feature name.
- nrow: number of samples.
- ncol: number of nodes or edges.
- inf: number of infinite values in the matrix.
- neginf: number of negative infinite values in the matrix.
- na: number of NA values in the matrix.
- nan: number of NaN values in the matrix.
- neg: number of negative values in the matrix.
- pos: number of positive values in the matrix.
- zero: number of 0's in the matrix.
- max: The maximum value in the matrix.
- min: The minimum value in the matrix.

**summary** A list containing elements node and edge, each containing a list with a feature summary list; each feature summary in this list contains elements:

- values: a numeric vector the same length as the number of nodes or edges.
- test\_custom: a function or a string name of the summary test method used.

**summary\_desc** A list containing elements node and edge, each containing a data frame describing the features in feat with columns:

- feat: A string indicating feature name the summary was created for.
- test\_name: A string containing the name of the summary.
- class: A string corresponding to the column name of the meta slot whose values represent the class label of each sample on which the summary was created to compare or analyze.
- label1: A string from the class column of the meta slot indicating one of the labels compared to create the summary statistic.

- **label2:** A string from the class column of the meta slot indicating one of the labels compared to create the summary statistic.
- meta** A data frame containing the column(s) **id** (sample id's corresponding to row names of features in the **feat** slot) and any other meta data pertaining to samples being analyzed.
- markers** A character vector containing markers used.
- edge\_list** A list containing elements **child** and **parent**. These elements contain an edge list from child to parent and vice versa.
- graph** A list containing data frames **v** and **e** with information on cell population nodes and edges. **v** contains columns:
- **phenotype:** The cell population node label names e.g. A+B+C+.
  - **phenocode:** A string of "0", "1", "2", ... indicating the whether each marker is expressed on a cell population.
  - **phenolayer:** The layer on which a cell population resides i.e. the number of markers in its phenotype label.
  - **phenogroup:** The markers used to make up the phenotype.
- plot\_layout** A string indicating the name of the **igraph** layout function used to layout the cell population nodes for plotting.
- etc** A list containing other information (see [fg\\_get\\_summary](#) for other things stored in this slot):
- **cumsumpos:** A logical indicating whether cell counts in flowGraph object contains cumulated cell counts; this is optional and can be done only for there is more than one threshold for one or more markers. This should also only be ran when initializing a flowGraph object as converting back and forth is computationally expensive. If the user is interested in seeing non- and cumulated counts, we recommend keeping two flowGraph objects, one for each version. This function simply converts e.g. the count of A+ or A++ into the sum of count of A+, A++, and A+++ or A++, and A+++.
  - **class\_mean\_normalized:** A logical indicating whether the features in the flowGraph object has been normalized according to some sample meta e.g. subject.
  - **save:** A list containing a string indicating the save ID of the object and a string indicating path where the object is saved – used in function **save\_fg** to identify whether or not to save to the same folder.

## Creating Objects

Objects can be created using `new("flowFrame")` or the constructor `flowGraph`, with mandatory argument `input_`. Creating objects using `new` is discouraged.

## Methods

'object' represents a flowGraph object.

- **show(fg):** Shows a description of the flowGraph object.
- **fg\_get\_meta:** Retrieves the sample meta data from a given flowGraph object. See [fg\\_get\\_meta](#).
- **fg\_get\_graph:** Retrieves the cell population (**v**) and edge (**e**) meta data from a given flowGraph object. See [fg\\_get\\_graph](#).
- **fg\_get\_feature:** Retrieves the numeric feature matrix requested by the user from a given flowGraph object. See [fg\\_get\\_feature](#).

- `fg_get_summary`: Retrieves the feature summary list requested by the user from a given flowGraph object. See [fg\\_get\\_summary](#).
- `fg_get_feature_desc`: Retrieves the data frame from the `feat_desc` slot of a given flowGraph object. See [fg\\_get\\_feature\\_desc](#).
- `fg_get_summary_desc`: Retrieves the data frame from the `summary_desc` slot of a given flowGraph object. See [fg\\_get\\_summary\\_desc](#).
- `fg_add_feature`: Adds a feature to a given flowGraph object; we do not recommend users directly use this method, instead please use wrapper functions e.g. [fg\\_feat\\_node\\_prop](#), [fg\\_feat\\_node\\_specnr](#), See [fg\\_add\\_feature](#).
- `fg_rm_feature`: Removes a user specified feature from a given flowGraph object. See [fg\\_rm\\_feature](#).
- `fg_add_summary`: Adds a feature to a given flowGraph object; we do not recommend users directly use this method, instead please use wrapper function [fg\\_summary](#).
- `fg_clear_summary`: Removes all feature summaries from a given flowGraph object. See [fg\\_clear\\_summary](#).
- `fg_rm_summary`: Removes a user specified feature summaries from a given flowGraph object. See [fg\\_rm\\_summary](#).
- `fg_gsub_markers`: Substitutes marker names in a given flowGraph object. See [fg\\_gsub\\_markers](#).
- `fg_gsub_ids`: substitutes sample id's in a flowGraph object See [fg\\_gsub\\_ids](#).
- `fg_merge_samples`: Merges the samples of two flowGraph objects; we recommend users use the wrapper function [fg\\_merge](#) instead. See [fg\\_merge\\_samples](#).
- `fg_extract_samples`: Extract data for specific samples from a flowGraph object. See [fg\\_extract\\_samples](#).
- `fg_extract_phenotypes`: Extract data for specific cell population nodes from a flowGraph object. See [fg\\_extract\\_phenotypes](#).
- `fg_merge`: Merges two given flowGraph objects. See [fg\\_merge](#).
- `fg_set_layout`: Sets layout for cell population nodes for the purpose of plotting. See [fg\\_set\\_layout](#).
- `fg_plot`: Plots cell hierarchies in the flowGraph object. See [fg\\_plot](#).

## Examples

```
showClass("flowGraph")
```

---

flowGraphSubset	<i>flowGraph object constructor.</i>
-----------------	--------------------------------------

---

## Description

Initializes a flowGraph object given the cell counts for one or more flow cytometry sample(s). The flowGraph object returned holds meta data for each sample, each cell population node, edges representing how each cell population node relate to one another, and features for these nodes and edges.

**Usage**

```

flowGraphSubset(
  input_,
  meta = NULL,
  class = "class",
  no_cores = 1,
  markers = NULL,
  layout_fun = "layout.reingold.tilford",
  max_layer = NULL,
  cumsumpos = FALSE,
  path = NULL,
  summary_pars = flowGraphSubset_summary_pars(),
  summary_adjust = flowGraphSubset_summary_adjust(),
  save_plots = TRUE
)

```

**Arguments**

input_	a numeric matrix of the cell counts; its column/names must be the phenotype names and its rownames must be sample ID's.
meta	A data frame with meta data for each Phenotypes or sample; One of its column names should be "id" whose values correspond to the name of each Phenotypes object. We also recommend for it to have a column named "class" where one of its unique values is "control".
class	A string corresponding to the column name or index of meta whose values represent the class label of each sample, Default: 'class'
no_cores	An integer indicating how many cores to parallelize on, Default: 1
markers	A string vector of marker names used in input_, Default: NULL
layout_fun	A string of a function from the igraph package that indicates what layout should be used if a cell hierarchy is to be plotted; all such functions have prefix layout_. This is defaulted to e.g. layout_fun="layout.reingold.tilford".
max_layer	And integer indicating the maximum layer in the cell hierarchy to analyze; set to 'NULL' to analyze all layers.
cumsumpos	A logical variable indicating whether or not to cumulate cell counts; this applies only when partitionsPerMarker > 3 and will convert e.g. the count of A+ or A++ into the sum of the counts of A+, A++, A+++, ..., or A++, A+++, ... , Default: FALSE
path	A string indicating the folder path to where the flowGraph object should save its elements, Default = NULL (don't save).
summary_pars	A list containing parameters for calculating the statistical significance summary significance that will determine whether to trim out phenotypes for this fast version of flowGraph. The lists' elements are: <ul style="list-style-type: none"> <li>• node_feature: "SpecEnr"; this is the feature we will be testing, don't change this.</li> <li>• edge_feature: "NONE"; this unneeded for now.</li> </ul>

- `test_name`: "t\_diminish"; this unneeded for now.
  - `test_custom`: "t"; a string or a function indicating the statistical test desires. These tests can be `c("t", "wilcox", "ks", "var", "chisq")` corresponding to functions `t.test`, `wilcox.test`, `ks.test`, `var.test`, `chisq.test`
  - `diminish`: TRUE; whether or not to continue testing phenotypes whos parent phenotypes are all insignificant.
  - `class`: "class"; the column name in meta that contains class labels you want to test.
  - `labels`: `c("aml", "control")` for the flowcap data set; SET THIS!! to the class labels you want to test using `test_custom`.
- `summary_adjust` A list of parameters on how to adjust the p-values; this also affects which phenotypes are tested. The elements in the list are:
- `adjust_custom`: "byLayer"; this is a string (corresponding to an option in `p.adjust`) or a function used to adjust p-values.
  - `btwn_test_custom`: "t"; see `test_custom` in `summary_pars`; this statistical significance test is used in the filters.
  - `adjust0_lim`: see `fg_get_summary`.
  - `filter_adjust0`: see `fg_get_summary`.
  - `filter_es`: see `fg_get_summary`.
  - `filter_btwn_tpthres`: see `fg_get_summary`.
  - `filter_btwn_es`: see `fg_get_summary`.
- `save_plots` A logical indicating whether or not to save plots.

## Details

All node and edge features are trimmed such that only the significant phenotypes are left; the original input is stored in the slot `etc$original_count` of the returned `flowGraph` object.

## Value

`flowGraph` object

## Examples

```
## Not run:
if(interactive()){
  data(fg_data_pos2)
  fg <- flowGraph(fg_data_pos2$count, meta=fg_data_pos2$meta, no_cores=1)
}

## End(Not run)
```

flowGraphSubset\_summary\_adjust

*Default for flowGraphSubset's summary\_adjust*

---

### Description

Default input for flowGraphSubset's summary\_adjust parameter. ONLY USE THIS OVER flow-Graph IF: 1) your data set has more than 10,000 cell populations and you want to speed up your calculation time AND 2) you only have one set of classes you want to test on the SAME SET OF SAMPLES (e.g. control vs experiment). As flowGraphSubset does not calculate the SpecEnr for all cell populations, so if you want to test other sets of classes on the same samples, you will not be able to test all possible cell populations on the new set of classes.

### Usage

```
flowGraphSubset_summary_adjust()
```

### Value

Default list parameter flowGraphSubset's summary\_adjust parameter.

### Examples

```
flowGraphSubset_summary_adjust()
```

---

flowGraphSubset\_summary\_pars

*Default for flowGraphSubset's summary\_pars*

---

### Description

Default input for flowGraphSubset's summary\_pars parameter.

### Usage

```
flowGraphSubset_summary_pars()
```

### Value

Default list parameter flowGraphSubset's summary\_pars parameter.

### Examples

```
flowGraphSubset_summary_pars()
```



---

fpurrr_map	<i>Wrapper for map</i>
------------	------------------------

---

**Description**

Wrapper for `purrr::map` and `furrr::future_map` to handle parallel-ization

**Usage**

```
fpurrr_map(x, f, no_cores = 1, prll = TRUE, ...)
```

**Arguments**

<code>x</code>	Variable to recurse over; must be indices!
<code>f</code>	Function to recurse over.
<code>no_cores</code>	Number of cores to use; future must have been ran already.
<code>prll</code>	If set to FALSE, forces use of <code>purrr::map</code> instead of <code>furrr::future_map</code> , Default: TRUE
<code>...</code>	Other parameters used by <code>f</code> .

**Details**

Wrapper for `purrr::map` and `furrr::future_map` to handle parallel-ization easily; note that future must have been ran already outside of the function and outputs will always be a list.

**Value**

Unnested named list.

**See Also**

[map future\\_map](#)

---

get_child	<i>Gets child populations of given cell populations</i>
-----------	---

---

**Description**

Gets the child populations of a vector of given cell populations `parens` and updates `pchild` the edge list if edge list doesn't contain the requested information.

**Usage**

```
get_child(parens, pchild, pc_i, ac__, meta_cell__)
```

Arguments

parens	Character vector of cell population phenotypes.
pchild	Edge list where the name of the list is the cell population and the vector in each element contains the child cell populations of the named cell population.
pc_i	A cell population x marker matrix where the values are 0/1/2/... corresponding to marker conditions -/+/-... for possible PARENT populations.
ac__	A list where the elements are marker index > "0"/"1"/"2"/... > a logical vector the same length as the number of cell population phenotypes indicating whether or not the marker condition exists in them; this is for the possible CHILD cell populations
meta_cell__	data frame with meta data for cell population phenotypes from the flowGraph object; this is for the possible CHILD cell populations.

Value

A list containing child populations of parens; also globally updates pchild.

See Also

[map,keep](#)

---

get_eprop	<i>Gets edge proportions of a given edge matrix</i>
-----------	---

---

Description

Gets the edge proportions of the edges in edge matrix edf\_ and updates ep edge proportion matrix if it didn't contain the requested information.

Usage

```
get_eprop(edf_, ep, mp_, no_cores = 1)
```

Arguments

edf_	edge x from&to data frame containing edges and their from and to cell population phenotypes.
ep	sample x edge (parent_child) matrix with edge proportions.
mp_	sample x phenotype matrix with proportions.
no_cores	Number of cores to use, Default: 1

Value

ep with only the specific columns (edges) requested; also updates ep globally.

---

get\_paren

*Gets parent populations of given cell populations*


---

### Description

Gets the parent populations of a vector of given cell childs and updates pparen the edge list if edge list doesn't contain the requested information.

### Usage

```
get_paren(childs, pparen, pc__i, ac_, meta_cell_)
```

### Arguments

childs	Character vector of cell population phenotypes.
pparen	Edge list where the name of the list is the cell population and the vector in each element contains the parent cell populations of the named cell population.
pc__i	A cell population x marker matrix where the values are 0/1/2/... corresponding to marker conditions +/-/...; this is for the possible CHILD cell populations.
ac_	A list where the elements are marker index > "0"/"1"/"2"/... > a logical vector the same length as the number of cell population phenotypes indicating whether or not the marker condition exists in them; this is for the possible PARENT cell populations.
meta_cell_	data frame with meta data for cell population phenotypes from the flowGraph object; this is for the possible PARENT cell populations.

### Value

A list containing parent populations of childs; also globally updates pparen.

---

get\_phen\_list

*Creates edge lists.*


---

### Description

Creates edge lists indicating relationships between cell populations given meta data on these cell populations produced by the get\_phen\_meta function.

### Usage

```
get_phen_list(meta_cell = NULL, phen = NULL, no_cores = 1)
```

**Arguments**

meta_cell	A data frame containing meta data on cell populations as produced by the get_phen_meta function.
phen	A string vector of phenotype or cell population name labels. Cannot be set to NULL if meta_cell is set to NULL.
no_cores	An integer indicating how many cores to parallelize on.

**Value**

A list containing 'pchild', an edge list indicating where edges point to, 'pparen', an edge list indicating where edges point from, and 'edf', a data frame where each row contains the nodes an edge points 'from' and 'to'.

**See Also**

[get\\_phen\\_meta cell\\_type\\_layers](#)

**Examples**

```
phen <- c('A+B-C+', 'A+B-', 'A+')
get_phen_list(phen=phen)
```

---

get_phen_meta	<i>Genrates phenotype meta data.</i>
---------------	--------------------------------------

---

**Description**

Generates phenotype meta data given a vector of phenotypes and optionally phenocodes.

**Usage**

```
get_phen_meta(phen, phenocode = NULL)
```

**Arguments**

phen	A string vector of phenotype or cell population name labels.
phenocode	A string vector of phenocodes corresponding to the phenotypes in phen.

**Value**

A data frame with columns containing meta data on cell poulation nodes with columns:

- phenotype: cell population node label e.g. "A+B+".
- phenocode: a string penocode containing a numeric corresponding to the phenotype column e.g. "2200".
- phenolayer: a numeric layer on which a cell population resides in e.g. 2.

**See Also**

[get\\_phen\\_list](#) [cell\\_type\\_layers](#)

**Examples**

```
phen <- c('A+B+C-D++', 'A+B-', '', 'B++D-E+')
phenc <- c('22130', '21000', '00000', '03012')
get_phen_meta(phen, phenc)
```

---

ggdf

Prepares a given node and edge graph list for plotting.

---

**Description**

Prepares a given node and edge graph list for plotting by function `plot_gr`; do not use this function on its own.

**Usage**

```
ggdf(gr0)
```

**Arguments**

`gr0`                      A list containing data frames `e` and `v`.

**Details**

`codeggdf` adds to the data frames `v` and `e` in slot `graph` from a `flowGraph` object specifying plotting options as required by [plot\\_gr](#):

- `v`
  - `size`: a numeric indicating node size.
  - `colour`: a numeric or string indicating node colour.
  - `label`: a string indicating the label of a node.
  - `label_long`: a string indicating the long label of a node; used in interactive plots in [plot\\_gr](#).
  - `label_ind`: a vector of logical variables indicating which nodes to add a label to in a static plot.
  - `v_ind`: a vector of logical variables indicating which nodes to plot.
- `e`
  - `colour`: a numeric or string indicating edge colour.
  - `e_ind`: a vector of logical variables indicating which edges to plot.

**Value**

A list containing data frames `e` and `v`, each with additional meta data column.

**See Also**

[flowGraph-class get\\_phen\\_meta plot\\_gr](#)

**Examples**

```
no_cores <- 1
data(fg_data_pos30)
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
               prop=FALSE, specenr=FALSE,
               no_cores=no_cores)

gr_ <- ggdf(fg_get_graph(fg))
head(gr_$v)
head(gr_$e)
```

---

loop_ind_f	<i>Prepares parallel loop indices.</i>
------------	--

---

**Description**

loop\_ind\_f is a helper function that splits a vector of loop indices into a list of multiple loop indices for use in parallel processes within the flowGraph package.

**Usage**

```
loop_ind_f(x, n)
```

**Arguments**

x	A vector of loop indices.
n	An integer, or the number of vectors to split x into.

**Value**

list of n vectors with elements from x.

**Examples**

```
old_loop_inds <- 1:10
no_cores <- 5

new_loop_inds <- flowGraph::loop_ind_f(old_loop_inds, no_cores)
# future::plan(future::multisession)
# example_indices <- furrr::future_map(new_loop_inds, function(ii) {
#   purrr::map(ii, function(i) i )
# s})
```

---

mean_diff	<i>Normalizes matrix values by class.</i>
-----------	---

---

**Description**

Used only in the [fg\\_feat\\_mean\\_class](#) function; for each class in the classes vector, meandiff takes the column mean of the rows in the given matrix associated with that class; it then takes the difference point by point between these means and the original rows for that class.

**Usage**

```
mean_diff(m0, classes)
```

**Arguments**

- m0                   A numeric matrix.
- classes             A vector whose length is equal to the number of rows in the given matrix.

**Value**

A numeric matrix whose dimensions equate to that of the input and whose values are normalized per class.

**See Also**

[fg\\_feat\\_mean\\_class](#)

**Examples**

```
classes <- append(rep('apples',4), rep('oranges',3))
m0 <- matrix(rnorm(35), nrow=7)
m <- flowGraph::mean_diff(m0, classes)
```

---

ms_create	<i>Calcuat SpecEnr from proportion and expected proportion</i>
-----------	--

---

**Description**

FUNCTION\_DESCRIPTION

**Usage**

```
ms_create(mp_, me_)
```

**Arguments**

mp\_                      Numerical sample x cell population matrix w/ proportions.  
 me\_                      Numerical sample x cell population matrix w/ expected proportions.

**Value**

Numerical sample x cell population matrix w/ SpecEnr.

---

ms_psig	<i>Determines which phenotypes are statistically significant</i>
---------	--

---

**Description**

Determines which phenotypes are statistically significant based on SpecEnr.

**Usage**

```
ms_psig(  
  ms_,  
  summary_pars,  
  summary_adjust,  
  test_cust,  
  test_custom,  
  lyrno,  
  mp_,  
  me_  
)
```

**Arguments**

ms\_                      sample x phenotype SpecEnr matrix  
 summary\_pars          See flowGraphSubset.  
 summary\_adjust        See flowGraphSubset.  
 test\_cust              Final significance test function.  
 test\_custom            Raw significance test function.  
 lyrno                  An integer indicating total number of layers in the cell hierarchy including layer 0.  
 mp\_                    sample x phenotype proportion matrix.  
 me\_                    sample x phenotype expected proportion matrix.

**Value**

A logical vector the same length as the number of columns in ms\_ indicating whether or not each phenotype is significant; used only for the fast version of flowGraph to determine whether or not to keep testing the phenotypes' children.



---

plot_gr	<i>Plots a cell hierarchy.</i>
---------	--------------------------------

---

## Description

Plots a cell hierarchy given the output from fg\_plot, a list of nodes and edges.

## Usage

```
plot_gr(
  gr,
  main = NULL,
  show_bgedges = TRUE,
  colour_palette = NULL,
  label_coloured = TRUE,
  shiny_plot = FALSE,
  interactive = FALSE,
  visNet_plot = TRUE,
  colour_edges = FALSE,
  ...
)
```

## Arguments

gr	A list containing data frames e and v.
main	A string containing the plot title. If this is set to NULL, the function will look for a plot title in the main slot of gr; otherwise, this defaults to "".
show_bgedges	A logical variable indicating whether or not edges not specified for plotting should be plotted as light grey in the background. If this is NULL, the function will look for a show_bgedges in the show_bgedges slot of gr; otherwise, this defaults to TRUE.
colour_palette	A colour palette e.g. the default palette if the user sets this to NULL is c('blue', 'cyan', 'yellow', 'red')
label_coloured	A logical indicating whether to colour the node labels using the same colours as the nodes in the non-interactive plot.
shiny_plot	A logical indicating whether this plot is made for shiny; users don't need to change this.
interactive	A logical variable indicating whether the plot should be an interactive plot; see package ggiraph.
visNet_plot	A logical variable indicating if an interactive plot is chosen, if function should output a visNetwork plot; if set to FALSE, ggplot's girafe will be used instead.
colour_edges	A logical variable indicating whether to colour edges if plotting a node feature summary.
...	Other parameters for ggplot if interactive is set to FALSE; other parameters for plot_ly if interactive is set to TRUE.

**Value**

A ggplot object if interactive is set to FALSE; a ggiraph object if interactive is set to TRUE.

**See Also**

[flowGraph-class](#) [fg\\_plot](#) [get\\_phen\\_meta](#) [ggdf](#) [fg\\_get\\_feature](#) [fg\\_get\\_summary](#)

**Examples**

```
no_cores <- 1
data(fg_data_pos2)
fg <- flowGraph(fg_data_pos2$count, class=fg_data_pos2$meta$class,
               no_cores=no_cores)

# fg <- fg_summary(fg, no_cores=no_cores, class="class", control="control",
#                 overwrite=FALSE, test_name="t_byLayer", diminish=FALSE)

gr_summary <- fg_plot(
  fg, type="node", p_thres=.05, show_bgedges=TRUE,
  path=NULL) # set path to a full path to save plot as a PNG

plot_gr(gr_summary, main=gr_summary$main, show_bgedges=TRUE)

plot_gr(gr_summary, main=gr_summary$main, show_bgedges=TRUE, interactive=TRUE)
```

---

set_layout_graph	<i>Determines cell hierarchy layout.</i>
------------------	--

---

**Description**

Determines cell hierarchy layout and returns the X, Y coordinate of each cell population.

**Usage**

```
set_layout_graph(gr, layout_fun = "layout.reingold.tilford")
```

**Arguments**

<code>gr</code>	A list containing data frames <code>e</code> and <code>v</code> .
<code>layout_fun</code>	A string of a function from the <code>igraph</code> package that indicates what layout should be used if a cell hierarchy is to be plotted; all such functions have prefix <code>layout_</code> e.g. <code>layout_fun="layout.reingold.tilford"</code> .

**Value**

A list containing data frames `e` and `v`; each data frame contains an X, Y column or coordinate for each node and edge.

Examples

```
no_cores <- 1
data(fg_data_pos30)
fg <- flowGraph(fg_data_pos30$count, class=fg_data_pos30$meta$class,
                prop=FALSE, specenr=FALSE,
                no_cores=no_cores)

head(set_layout_graph(fg_get_graph(fg)))
```

---

summary_table	Summarizes a numeric matrix.
---------------	------------------------------

---

Description

Summarizes a numeric matrix.

Usage

```
summary_table(m, feat_type = "")
```

Arguments

- m                   A numeric matrix.
- feat\_type           Name of the matrix m.

Value

A data frame containing one row summarizing m; see [fg\\_get\\_feature\\_desc](#).

Examples

```
summary_table(matrix(rnorm(12),nrow=3), feat_type='random')
```

---

test_c	Converts input into a significance test function
--------	--

---

Description

Converts input into a significance test function

Usage

```
test_c(test_custom)
```

**Arguments**

test\_custom      a string c("t", "wilcox", "ks", "var", "chisq") or a function.

**Value**

a statistical significance test function.

**See Also**

[t.test](#), [wilcox.test](#), [ks.test](#), [var.test](#), [chisq.test](#)

---

time_output	<i>Outputs elapsed time.</i>
-------------	------------------------------

---

**Description**

Given a time, prints the time elapsed from that time until now.

**Usage**

```
time_output(start, msg = "")
```

**Arguments**

start              A time variable of class POSIXct, POSIXt.  
msg                A string with a message to print out after the elapsed time.

**Value**

Prints to console, the time from which process started start - ended, and > time elapsed from start until now.

**Examples**

```
start <- Sys.time()
flowGraph::time_output(start, 'start - now > time elapsed')
```

---

tstr	<i>Formats time into string.</i>
------	----------------------------------

---

**Description**

Formats time into a string HH:MM:SS given time zone.

**Usage**

```
tstr(time)
```

**Arguments**

time	A time variable of class POSIXct, POSIXt.
------	---

**Value**

Time formatted as a string; used in `time_output` function.

**Examples**

```
# NOT EXPORTED
flowGraph::tstr(Sys.time())
```

# Index

## \* datasets

- [fg\\_data\\_fca](#), [10](#)
  - [fg\\_data\\_pos2](#), [11](#)
  - [fg\\_data\\_pos30](#), [11](#)
- [cell\\_type\\_layers](#), [3](#), [68](#), [69](#)
- [chisq.test](#), [63](#), [76](#)
- [extract\\_markers](#), [4](#)
- [fg\\_add\\_feature](#), [5](#), [7](#), [16](#), [17](#), [19](#), [21](#), [22](#), [48](#), [58](#), [61](#)
- [fg\\_add\\_summary](#), [6](#), [23](#), [28–31](#), [48](#), [53](#), [58](#)
- [fg\\_clean\\_phen](#), [8](#)
- [fg\\_clear\\_features](#), [9](#)
- [fg\\_clear\\_summary](#), [9](#), [36](#), [55](#), [61](#)
- [fg\\_data\\_fca](#), [10](#)
- [fg\\_data\\_pos2](#), [11](#)
- [fg\\_data\\_pos30](#), [11](#)
- [fg\\_extract\\_phenotypes](#), [12](#), [14](#), [35](#), [58](#), [61](#)
- [fg\\_extract\\_raw](#), [13](#)
- [fg\\_extract\\_samples](#), [12](#), [14](#), [35](#), [36](#), [58](#), [61](#)
- [fg\\_feat\\_cumsum](#), [15](#)
- [fg\\_feat\\_edge\\_prop](#), [16](#)
- [fg\\_feat\\_edge\\_specnr](#), [17](#)
- [fg\\_feat\\_mean\\_class](#), [18](#), [71](#)
- [fg\\_feat\\_node\\_prop](#), [6](#), [16](#), [17](#), [19](#), [21](#), [61](#)
- [fg\\_feat\\_node\\_specnr](#), [6](#), [16](#), [17](#), [19](#), [20](#), [61](#)
- [fg\\_get\\_feature](#), [6](#), [16](#), [17](#), [19](#), [21](#), [22](#), [28](#), [31](#), [39](#), [41](#), [44](#), [46](#), [48](#), [52](#), [58](#), [60](#), [74](#)
- [fg\\_get\\_feature\\_desc](#), [6](#), [12](#), [14](#), [16](#), [17](#), [19](#), [21](#), [22](#), [29](#), [32](#), [36](#), [48](#), [58](#), [61](#), [75](#)
- [fg\\_get\\_feature\\_means](#), [23](#), [27](#), [28](#), [31](#)
- [fg\\_get\\_graph](#), [24](#), [60](#)
- [fg\\_get\\_markers](#), [25](#)
- [fg\\_get\\_meta](#), [25](#), [47](#), [60](#)
- [fg\\_get\\_summary](#), [7](#), [21](#), [23](#), [26](#), [29](#), [31](#), [39](#), [41](#), [44](#), [46](#), [48](#), [52](#), [58](#), [60](#), [61](#), [74](#)
- [fg\\_get\\_summary\\_desc](#), [7](#), [22](#), [23](#), [26–28](#), [29](#), [30](#), [31](#), [38](#), [40](#), [43](#), [45](#), [48](#), [58](#), [61](#)
- [fg\\_get\\_summary\\_index](#), [30](#)
- [fg\\_get\\_summary\\_tables](#), [31](#)
- [fg\\_gsub\\_ids](#), [32](#), [33](#), [58](#), [61](#)
- [fg\\_gsub\\_markers](#), [32](#), [33](#), [58](#), [61](#)
- [fg\\_load](#), [34](#)
- [fg\\_merge](#), [12](#), [14](#), [35](#), [36](#), [58](#), [61](#)
- [fg\\_merge\\_samples](#), [12](#), [35](#), [36](#), [58](#), [61](#)
- [fg\\_plot](#), [7](#), [24](#), [30](#), [37](#), [41](#), [44](#), [46](#), [52](#), [61](#), [74](#)
- [fg\\_plot\\_box](#), [40](#), [52](#)
- [fg\\_plot\\_pVSDiff](#), [42](#), [52](#)
- [fg\\_plot\\_qq](#), [41](#), [44](#), [44](#), [52](#)
- [fg\\_replace\\_meta](#), [26](#), [46](#)
- [fg\\_rm\\_feature](#), [6](#), [16](#), [17](#), [19](#), [21](#), [22](#), [47](#), [48](#), [58](#), [61](#)
- [fg\\_rm\\_summary](#), [7](#), [23](#), [28–31](#), [48](#), [48](#), [58](#), [61](#)
- [fg\\_save](#), [34](#), [49](#)
- [fg\\_save\\_plots](#), [50](#)
- [fg\\_set\\_layout](#), [52](#), [61](#)
- [fg\\_summary](#), [6](#), [7](#), [10](#), [39](#), [53](#), [57](#), [61](#)
- [flowGraph](#), [55](#)
- [flowGraph-class](#), [58](#)
- [flowGraphSubset](#), [61](#)
- [flowGraphSubset\\_summary\\_adjust](#), [64](#)
- [flowGraphSubset\\_summary\\_pars](#), [64](#)
- [fpurrr\\_map](#), [65](#)
- [future\\_map](#), [65](#)
- [get\\_child](#), [65](#)
- [get\\_eprop](#), [66](#)
- [get\\_paren](#), [67](#)
- [get\\_phen\\_list](#), [4](#), [67](#), [69](#)
- [get\\_phen\\_meta](#), [4](#), [39](#), [68](#), [68](#), [70](#), [74](#)
- [ggdf](#), [24](#), [39](#), [69](#), [74](#)
- [keep](#), [66](#)
- [ks.test](#), [63](#), [76](#)
- [loop\\_ind\\_f](#), [70](#)
- [map](#), [50](#), [65](#), [66](#)

Matrix, [15](#), [58](#)  
mean\_diff, [71](#)  
ms\_create, [71](#)  
ms\_psig, [72](#)  
  
p.adjust, [63](#)  
plot\_gr, [24](#), [39](#), [41](#), [44](#), [46](#), [52](#), [69](#), [70](#), [73](#)  
  
registerDoParallel, [58](#)  
  
set\_layout\_graph, [52](#), [74](#)  
show, flowGraph-method  
    (flowGraph-class), [58](#)  
str\_extract, [8](#)  
str\_split, [4](#), [8](#)  
summary\_table, [75](#)  
  
t.test, [63](#), [76](#)  
test\_c, [75](#)  
time\_output, [76](#)  
tstr, [77](#)  
  
var.test, [63](#), [76](#)  
  
wilcox.test, [63](#), [76](#)