

Package ‘flowMagic’

July 18, 2025

Type Package

Title Automated gating of flow cytometry data

Version 0.99

Description tool to perform the automated gating of bivariate flow cytometry data.

License Apache License 2.0 | file LICENSE + file NOTICE

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports caret,
ggplot2,
stringr,
parallel,
randomForest,
concaveman,
sp,
sf,
stats,
grDevices,
doParallel,
nnet,
sm,
pracma,
patchwork,
ggExtra,
RColorBrewer,
dplyr

Suggests flowCore,
flowWorkspace,
Biobase,
data.tree,
visNetwork,
klaR(<= 0.6-12),

remotes,
CytoML,
plotly

Contents

add_labels_column	3
assign_events_to_nearest_centroids	4
check_polygons_intersection	5
compute_gates	5
convert_to_integers_chr	6
csv_to_dens	6
exports_plots	7
export_raw_gs_plots	8
extract_polygon_gates	9
flowmagic_pred_to_gs	10
flowmagic_pred_to_poly_gates	10
get_centroids	11
get_classes_expr_df	12
get_density_features	12
get_density_scores	13
get_distance_loc_vs_test	13
get_dist_template	14
get_flowframe_from_gs	15
get_hierarchy_all_pops	15
get_hull_all_gates	16
get_indices_cross_val	17
get_list_df_gated_plots	18
get_local_train_sets	18
get_pops_hierarchy_list	19
get_pop_multiclass	19
get_slot_hierarchy_list	20
get_test_sets	21
get_train_data	21
get_weights_density_features	22
import_gating_info	23
import_reference_csv	23
import_test_set_csv	24
import_test_set_fcs	24
magicPlot	25
magicplot_3D	27
magicPlot_template	28
magicPred	28
magicPred_all	29
magicPred_hierarchy	31
magicTrain	31
magicTrain_dt	32
magicTrain_hierarchy	33

<i>add_labels_column</i>	3
--------------------------	---

magicTrain_knn	34
magicTrain_nb	35
magicTrain_nnet	36
magicTrain_rf	37
magic_label_poly	38
magic_label_rectangle	38
magic_plot_wrap	39
map_to_parent	40
map_to_root	40
name_pop_gating	41
post_process_gates	41
process_test_data	42
range01	43
smooth_hull	43
update_label_association	44
Index	45

<code>add_labels_column</code>	<i>add_labels_column</i>
--------------------------------	--------------------------

Description

function to add label association column.

Usage

`add_labels_column(df, labels_association)`

Arguments

`df` Dataframe.
`labels_association` Vector of label association.

Value

Dataframe.

Examples

`add_labels_column()`

```
assign_events_to_nearest_centroids  
    assign_events_to_nearest_centroids
```

Description

function to assign events to class with nearest centroid.

Usage

```
assign_events_to_nearest_centroids(  
    gated_df,  
    n_cores = 1,  
    method_dist = "euclidean",  
    thr_dist = 0.15,  
    include_zero = F,  
    remove_centroids = T  
)
```

Arguments

<code>gated_df</code>	dataframe with labels (third column).
<code>n_cores</code>	Number of cores. Default to 1.
<code>method_dist</code>	Distance method calculation. Default to euclidean.
<code>thr_dist</code>	Distance threshold for centroids calculation. Default to 0.15.
<code>include_zero</code>	Consider centroid of label 0. Default to False.
<code>remove_centroids</code>	Remove centroids too near each other based on <code>thr_dist</code> value.

Value

Dataframe.

Examples

```
assign_events_to_nearest_centroids()
```

check_polygons_intersection	
	<i>extract_polygon_gates</i>

Description

function to check polygons intersection.

Usage

```
check_polygons_intersection(list_df_hull)
```

Arguments

list_df_hull List of polygons coordinates

Value

float

Examples

```
check_polygons_intersection()
```

compute_gates	<i>compute_gates</i>
---------------	----------------------

Description

function to assign events based on polygon gates.

Usage

```
compute_gates(gated_df, list_final_polygons_coords, no_classes = F)
```

Arguments

gated_df dataframe with labels (third column).
list_final_polygons_coords List of dataframes containing polygon coordinates.
no_classes Generate third column of labels. Default to False.

Value

Dataframe.

Examples

```
compute_gates()
```

```
convert_to_integers_chr
      convert_to_integers_chr
```

Description

function to text classes to integers-like classes if necessary.

Usage

```
convert_to_integers_chr(df)
```

Arguments

df Dataframe with classes (third column) to check and eventually modify

Value

Dataframe

Examples

```
convert_to_integers_chr()
```

```
csv_to_dens                      csv_to_dens
```

Description

function to get density of events (with classes associated if present).

Usage

```
csv_to_dens(df, with_classes = T, n_coord = "df", normalize_data = T)
```

Arguments

df Dataframe of marker expression values.
with_classes Consider classes. Default to True.
n_coord Grid size. Default to df.
normalize_data If True, data is normalized to 0-1 range. Default to True.

Value

Dataframe.

Examples

```
csv_to_dens()
```

exports_plots	<i>exports_plots</i>
---------------	----------------------

Description

function to generate plots (no hierarchy) from list of labelled dataframes.

Usage

```
exports_plots(  
  list_gated_data,  
  path_output,  
  n_cores = 1,  
  type_plot = "dens",  
  show_legend = T,  
  x_lab = "x",  
  y_lab = "y",  
  size_title_x = 23,  
  size_title_y = 23,  
  aspect_ratio = NULL,  
  w_val = 16,  
  h_val = 10,  
  size_axis_text = 25,  
  export_csv = F,  
  side_by_side = F,  
  ...  
)
```

Arguments

list_gated_data	list of dataframes. Each dataframe has 3 columns: marker 1 values, marker 2 values and label column.
path_output	Path to the directory where to export the plots.
n_cores	Number of cores to use. Default to 1.
type_plot	the user can choose between density (= "dens") or label assignment visualization (= "ML")
show_legend	If True it shows the legend for the label assignment visualization. Default to True.

<code>x_lab</code>	x-axis label.
<code>y_lab</code>	y-axis label.
<code>size_title_x</code>	Size x axis label.
<code>size_title_y</code>	Size y axis label.
<code>aspect_ratio</code>	Set aspect ratio. Default to NULL> If = 1, y and x axis ticks have same distance.
<code>w_val</code>	width value. Default to 16.
<code>h_val</code>	height value. Default to 10.
<code>size_axis_text</code>	Size of ticks labels.
<code>export_csv</code>	Export plot data as csv files. Default to False.
<code>side_by_side</code>	Arrange dens plot and ML plot side-by-side. Default to False.

Examples

```
exports_plots()
```

<code>export_raw_gs_plots</code>	<i>export_raw_gs_plots</i>
----------------------------------	----------------------------

Description

function to generate ungated plots from selected gs node

Usage

```
export_raw_gs_plots(
  gs,
  node_name,
  channel_x,
  channel_y,
  path_output,
  n_cores = 1,
  x_lab = "x",
  y_lab = "y",
  w_val = 7,
  h_val = 7,
  size_points = 1,
  return_data = F,
  ...
)
```


Arguments

<code>gs</code>	GatingSet
<code>path_output</code>	Path to the directory where to export the plots.
<code>n_cores</code>	Number of cores to use. Default to 1.
<code>x_lab</code>	x-axis label.
<code>y_lab</code>	y-axis label.
<code>w_val</code>	width value. Default to 7 inches.
<code>h_val</code>	height value. Default to 7 inches.
<code>size_points</code>	Size points scatter plot.
<code>return_data</code>	If TRUE, return the list of dataframes used to generate the plots. Default to FALSE.

Examples

```
export_raw_gs_plots()
```

```
extract_polygon_gates extract_polygon_gates
```

Description

function to extract the polygon gates objects based on the convex hull and classes.

Usage

```
extract_polygon_gates(gated_df, concavity_val = 1, ...)
```

Arguments

<code>gated_df</code>	dataframe with labels (third column).
<code>concavity_val</code>	Concavity of polygons. Default to 1.

Value

List of dataframes.

Examples

```
extract_polygon_gates()
```

```
flowmagic_pred_to_gs  flowmagic_pred_to_gs
```

Description

function to label points of dataframe based on polygon coordinates

Usage

```
flowmagic_pred_to_gs(list_poly_gates, gs, parent_node)
```

Arguments

list_poly_gates	List of polygonGate objects generated using the flowmagic_pred_to_poly_gates
gs	GatingSet to update with the flowMagic polygons.
parent_node	Parent population name of the new gate.

Value

GatingSet

Examples

```
flowmagic_pred_to_gs()
```

```
flowmagic_pred_to_poly_gates
  flowmagic_pred_to_poly_gates
```

Description

function to label points of dataframe based on polygon coordinates

Usage

```
flowmagic_pred_to_poly_gates(
  list_df,
  pred_label,
  gate_label,
  n_cores = 1,
  concavity_val = 10
)
```

Arguments

list_df	List of dataframes with three columns: marker 1, marker 2, class. Can get input generated by the magicPred function.
pred_label	Select prediction label to extract polygon.
gate_label	Label for the selected polygon within the GatingSet framework.
n_cores	Number of cores to use. Default to 1.
concavity_val	Concavity value for drawing polygon. Default to 10.

Value

list

Examples

```
flowmagic_pred_to_poly_gates()
```

get_centroids	<i>get_centroids</i>
---------------	----------------------

Description

function to get centroids for each label

Usage

```
get_centroids(
  df,
  low_thr = 0.1,
  up_thr = 0.9,
  thr_dist = 0.15,
  include_zero = F,
  remove_centroids = T
)
```

Arguments

df	dataframe with labels (third column).
low_thr	Lower threshold for quantile calculation.
up_thr	Upper threshold for quantile calculation.
thr_dist	Distance threshold for centroids calculation. Default to 0.15.
include_zero	Consider centroid of label 0. Default to False.
remove_centroids	Remove centroids too near each other based on thr_dist value.

Value

Dataframe.

Examples

```
get_centroids()
```

```
get_classes_expr_df      get_classes_expr_df
```

Description

function to get classes of original expression df based on density df predictions.

Usage

```
get_classes_expr_df(dens_df, original_df)
```

Arguments

<code>dens_df</code>	Dataframe of density values.
<code>original_df</code>	Original dataframe.

Value

Dataframe.

Examples

```
get_classes_expr_df()
```

```
get_density_features      get_density_features
```

Description

function to get density features only given a bivarite density csv.

Usage

```
get_density_features(df_dens, min_height = 0.06)
```

Arguments

<code>df_dens</code>	Dataframe of density estimates for both markers.
<code>min_height</code>	Minimum height of the peaks to consider.

Value

Vector of numbers.

Examples

```
get_density_features()
```

```
get_density_scores      get_density_scores
```

Description

function to get scores for distance template calculation.

Usage

```
get_density_scores(df_template, df_test, select_density_features = NULL)
```

Arguments

`df_template` Dataframe of template markers expression values.
`df_test` Dataframe of test markers expression values.
`select_density_features`
 Select features to use. Default to NULL.

Value

Matrix of numbers.

Examples

```
get_density_scores()
```

```
get_distance_loc_vs_test  
                          get_distance_loc_vs_test
```

Description

function to compare the similiraty between current test data and local training set.

Usage

```
get_distance_loc_vs_test(test_df, loc_df, show_plot = "none", nboot = 50)
```

Arguments

<code>test_df</code>	Dataframe with bivariate marker expression.
<code>loc_df</code>	Dataframe with bivariate marker expression.
<code>show_plot</code>	show density comparison. Default to none.
<code>nboot</code>	Number of permutations. Default to 50.

Value

List of p-values.

Examples

```
get_distance_loc_vs_test()
```

<code>get_dist_template</code>	<i>get_dist_template</i>
--------------------------------	--------------------------

Description

function to get distance between template and test data.

Usage

```
get_dist_template(matrix_scores, dist_method = "euclidean")
```

Arguments

<code>matrix_scores</code>	Matrix of density features generated by <code>get_density_scores</code> function.
<code>dist_method</code>	Type of distance method calculation.

Value

Number.

Examples

```
get_dist_template()
```

```
get_flowframe_from_gs  get_flowframe_from_gs
```

Description

function to get flowframe from gs (converting cytoframe to flowframe)

Usage

```
get_flowframe_from_gs(gs, node_name, sample_id)
```

Arguments

gs	GatingSet
node_name	Name of the Gating tree node whose gating data needs to be extracted.
sample_id	Name or index of the sample to extract.

Value

List.

Examples

```
get_flowframe_from_gs()
```

```
get_hierarchy_all_pops
      get_hierarchy_all_pops
```

Description

function to get the hierarchy of all pops from the sample manually gated (the input gating hierarchy). Based on its output the function magicTrain will perform the training step using the sample manually gated (local training set) and the project discovery data (global training set).

Usage

```
get_hierarchy_all_pops(gh, export_visnet = F, path.output = "None")
```

Arguments

gh	GatingHierarchy.
export_visnet	If true, it export visnetwork object.
path.output	Path to save visnetwork object. Default to None.

Value

List of Dataframes.

Examples

`get_hierarchy_all_pops()`

<code>get_hull_all_gates</code>	<i>get_hull_all_gates</i>
---------------------------------	---------------------------

Description

function to get the convex hull of all gates.

Usage

`get_hull_all_gates(gated_df, concavity_val = 1, spar_val = 0.7, smoothing = F)`

Arguments

- `gated_df` dataframe with labels (third column).
- `concavity_val` Values of concavity. Default to 1.
- `spar_val` Value of spar. Default to 0.7.
- `smoothing` Apply smoothing of lines? Default to F.

Value

List of dataframes.

Examples

`get_hull_all_gates()`

`get_indices_cross_val` *get_indices_cross_val*

Description

function to get indices for cross val.

Usage

```
get_indices_cross_val(  
    df_train,  
    n_cores = 1,  
    train_inds = "plot_num",  
    val_inds = "none",  
    n_train_plots = 5,  
    n_folds = 5,  
    seed = 40,  
    n_val_plots = 5  
)
```

Arguments

<code>df_train</code>	Dataframe of training features generated by the <code>get_train_data</code> function.
<code>n_cores</code>	Number of cores to use. Default to 1.
<code>train_inds</code>	Type of method to extract training indices: <code>plot_num</code> , <code>rand_set_num</code> , <code>rand_set_n_gates_info</code> .
<code>val_inds</code>	Type of method to extract validation indices: <code>plot_num</code> , <code>rand_set_num</code> , <code>rand_set_n_gates_info</code> .
<code>n_train_plots</code>	Number of training plots to include in training data for each iteration.
<code>n_folds</code>	Number of training iterations.
<code>seed</code>	Seed to randomly extract data.
<code>n_val_plots</code>	Number of training plots to include in validation data for each iteration.

Value

List of integers.

Examples

```
get_indices_cross_val()
```

```
get_list_df_gated_plots
      get_list_df_gated_plots
```

Description

function to get labelled dataframe based on selected node for each sample in gs.

Usage

```
get_list_df_gated_plots(gs, gate_name)
```

Arguments

gs	GatingSet
gate_name	Name of the Gating tree node whose gating data needs to be extracted.

Value

List.

Examples

```
get_list_df_gated_plots()
```

```
get_local_train_sets  get_local_train_sets
```

Description

Based on the hierarchy calculated using the get_hierarchy function, we generate the local training sets gated by the biologists.

Usage

```
get_local_train_sets(gh, hierarchical_tree, info_hierarchy)
```

Arguments

gh	GatingHierarchy.
hierarchical_tree	Dataframe of hierarchy information generated by the get_hierarchy function.
info_hierarchy	List of hierarchy information generated by the get_hierarchy function.

Value

List of Dataframes.

Examples

```
get_local_train_sets()
```

```
get_pops_hierarchy_list  
get_pops_hierarchy_list
```

Description

function to get info from a list of hierarchical dataset (it reports all the pops for each level in a vector).

Usage

```
get_pops_hierarchy_list(hierarchical_list)
```

Arguments

`hierarchical_list`
list of populations names for each level.

Value

Vector of characters.

Examples

```
get_pops_hierarchy_list()
```

```
get_pop_multiclass    get_pop_multiclass
```

Description

function to get the children of the selected population, useful for multiclass classification. The output is only the pops with same dimensions

Usage

```
get_pop_multiclass(gh, pop)
```

Arguments

gh	GatingHierarchy.
pop	Name of population to get events assignments.

Value

Vector of characters.

Examples

```
get_pop_multiclass()
```

get_slot_hierarchy_list
<i>get_slot_hierarchy_list</i>

Description

function to access a slot of results from the hierarchy list based on the pop selected.

Usage

```
get_slot_hierarchy_list(hierarchical_list, pop_selected)
```

Arguments

hierarchical_list	list of populations names for each level.
pop_selected	Population name.

Value

Element of a list.

Examples

```
get_slot_hierarchy_list()
```

get_test_sets	<i>get_test_sets</i>
---------------	----------------------

Description

Function to obtain the data ready to be gated (validation set or test set). It takes as input a flowSet and generates a list of dataframes with a ML structure for each ungated fcs file. Each dataframe is the expression matrix of the root.

Usage

```
get_test_sets(fs, gh)
```

Arguments

fs	flowSet to gate.
gh	Gating hierarchy.

Value

List of dataframes.

Examples

```
get_test_sets()
```

get_train_data	<i>get_train_data</i>
----------------	-----------------------

Description

function to import training data based on paths to files.

Usage

```
get_train_data(  
  paths_file = NULL,  
  df_paths = NULL,  
  n_cores = 1,  
  prop_down = NULL,  
  remove_class = NULL,  
  n_points_per_plot = NULL,  
  normalize_data = T,  
  vec_col = NULL  
)
```

Arguments

paths_file	Vector of paths. Each path points toward a single csv file containin training info (labels and bivariate expression). paths_file can be also directly the list of dataframes containing labels and bivariate expression.
df_paths	Dataframe containing the paths of file to read. The paths to data must be in the first column. The associated paths to classes are in second column.
n_cores	Number of cores. Default to 1.
prop_down	Proportion of events (downsampling). Default to NULL (downsampling using number of points).
remove_class	Vector of classes to ignore. Default to NULL.
n_points_per_plot	Number of points for downsampling.
normalize_data	If True, data is normalized to 0-1 range. Default to True.
vec_col	vector of columns names if the input dataframes have more than 3 columns. The third column name must always refer to the column with the gate label of each event. Default to NULL.

Value

Dataframe.

Examples

```
get_train_data()
```

```
get_weights_density_features
      get_weights_density_features
```

Description

function to calculate final score based on density features.

Usage

```
get_weights_density_features(df_scores)
```

Arguments

df_scores	Dataframe of distance scores.
-----------	-------------------------------

Value

Vector of numbers.

Examples

```
get_weights_density_features()
```

import_gating_info	<i>import_sample_gated</i>
--------------------	----------------------------

Description

convert a flowWorkspace or a GatingML file of the train data folder into a gated gh (gh_train)

Usage

```
import_gating_info(path, type = "gs", group_wsp = NULL)
```

Arguments

path	path to gs or GatingML object.
type	type of object.
group_wsp	Group of wsp to import.

Value

GatingSet object

Examples

```
import_sample_gated()
```

import_reference_csv	<i>import_reference_csv</i>
----------------------	-----------------------------

Description

function to import plain gold standards data (no hierarchy)

Usage

```
import_reference_csv(path_results, n_cores = 1)
```

Arguments

path_results	path to directory containing the csv files to read (with third column of labels).
n_cores	Number of cores to use. Default to 1.

Value

list of dataframes

Examples

```
import_reference_csv()
```

import_test_set_csv	<i>import_test_set_csv</i>
---------------------	----------------------------

Description

function to import test set in csv format.

Usage

```
import_test_set_csv(path_data, n_cores = 1, xy_col = T)
```

Arguments

path_data	path to directory containing csv files to read (third column is ignored).
n_cores	Number cores. Default to 1.
xy_col	Colnames equal to x and y. Default to True.

Value

List of dataframes.

Examples

```
import_test_set_csv()
```

import_test_set_fcs	<i>import_test_set</i>
---------------------	------------------------

Description

read the ungated fcs files into a flowSet. The ungated fcs are assumed to be already cleaned,compensated,and transformed.

Usage

```
import_test_set_fcs(path, n_samples = "All", ref_f_n = 1)
```

Arguments

path	path of directory containig the fcs files.
n_samples	Number of samples. Default to All.
ref_f_n	Set reference flowFrame to match channel names. Default to 1(first flowFrame).

Value

flowSet.

Examples

```
import_test_set()
```

magicPlot

magicPlot

Description

function to generate the scatter plot with colored density of the events.

Usage

```
magicPlot(
  df,
  type = "dens",
  polygons_coords_list = NULL,
  show_legend = T,
  size_axis_text = 18,
  size_title_x = 20,
  size_title_y = 20,
  treat_0_as_gate = F,
  x_lab = "x",
  y_lab = "y",
  gates_to_plot = NULL,
  apply_manual_scale = F,
  hull_only = F,
  size_points = 1,
  concavity_val = 20,
  aspect_ratio = NULL,
  x_lim1 = NULL,
  x_lim2 = NULL,
  y_lim1 = NULL,
  y_lim2 = NULL,
  add_labels = F,
  map_label_polygon = NULL,
  size_pol_name = 6,
  show_marginals = F,
  ...
)
```

Arguments

df	Dataframe of bivariate markers expression (with labels if gates to plot).
type	Type of plot to generated. "dens"=bivariate density plot. "ML"=events assignments plot
polygons_coords_list	list of gates coordinates. Needed if labels not included in df. Default to NULL.

<code>show_legend</code>	Show legend if type="ML". Default to True.
<code>size_axis_text</code>	Size of axis ticks labels. Default to 18.
<code>size_title_x</code>	Size of x axis label title.
<code>size_title_y</code>	Size of y axis label title.
<code>treat_0_as_gate</code>	Treat 0 label as gate. Default to False (0 label is background)
<code>x_lab</code>	Label of x axis.
<code>y_lab</code>	Label off y axis.
<code>gates_to_plot</code>	Select labels to plot.
<code>apply_manual_scale</code>	Apply predefined scale of colors. Default to True.
<code>size_points</code>	Size of points in scatter plot.
<code>concavity_val</code>	Concavity value. Default to 5. Higher value, less jagged boundaries
<code>aspect_ratio</code>	Aspect ratio value. If = 1, y and x axis have equal distance between the ticks labels. Default to NULL.
<code>x_lim1</code>	Minimum limit x axis. Default to NULL.
<code>x_lim2</code>	Max limit x axis. Default to NULL.
<code>y_lim1</code>	Minimum limit y axis. Default to NULL.
<code>y_lim2</code>	Max limit y axis. Default to NULL.
<code>add_labels</code>	add polygon labels. Default to FALSE.
<code>map_label_polygon</code>	map of polygon labels (to assign custom labels). Default to NULL.
<code>size_pol_name</code>	size polygon labels. Default to 6.
<code>show_marginals</code>	show 1D density next to axis. Default to False.

Value

ggplot.

Examples

```
magicPlot()
```

`magicplot_3D`*magicplot_3D*

Description

function to make a 3D plot of data.

Usage

```
magicplot_3D(  
  df,  
  class_col = F,  
  x_lab = "x",  
  y_lab = "y",  
  z_lab = "z",  
  type = "ML",  
  size_p = 1  
)
```

Arguments

<code>df</code>	input dataframe composed of at least three columns. Four columns if gate needs to be plot.
<code>class_col</code>	Gate to plot? Default to False.
<code>x_lab</code>	label of the x axis
<code>y_lab</code>	Label of the y axis
<code>z_lab</code>	Label of the z axis
<code>type</code>	Assuming <code>class_col==T</code> , If <code>type==ML</code> , Generate a plot colored based on the gate assignment. If <code>type=="mesh"</code> , generate a 3d polygon gate.
<code>size_p</code>	size of scatter plot points. Default to 1.

Value

plotly plot

Examples

```
magicplot_3D()
```

magicPlot_template	<i>magicPlot_template</i>
--------------------	---------------------------

Description

function to interactively generate a template based on bivariate ungated data.

Usage

```
magicPlot_template(df, size_points = 1)
```

Arguments

df	input dataframe composed of two columns reporting markers expression (first column = marker 1, second column = marker 2).
size_points	Size of points in scatter plot. Default to 1.

Value

Dataframe

Examples

```
magicPlot_template()
```

magicPred	<i>magicPred</i>
-----------	------------------

Description

function to predict on plain test data (no hierarchy)

Usage

```
magicPred(
  test_data,
  magic_model = NULL,
  magic_model_n_gates = NULL,
  ref_model_info = NULL,
  n_cores = 1,
  ref_data_train = NULL,
  prop_down = NULL,
  thr_dist = 0.05,
  n_points_per_plot = NULL,
  normalize_data = T,
  include_zero_val = T,
  ...
)
```

Arguments

test_data	Dataframe of test data to gate. It has only the two columns of marker expression.
magic_model	Global trained model to predict gates. It can be a single model or list of named models (each model trained on selected number of gates).
magic_model_n_gates	Global trained model to predict number of gates. If different from NULL, magic_model is expected to be a list of models to predict certain gates (e.g., 5 models for 2,3,4,5 or 6 gates).
ref_model_info	Template model to predic gates.
n_cores	Number of cores to use. Default to 1.
ref_data_train	Template data used to generate ref_model_info. Needed to calculate target-template distance.
prop_down	Proportion for downsampling. Default to NULL (automatic downsampling using n_points_per_plot).
n_points_per_plot	Number of points to consider for downsampling. Default to 500.
normalize_data	If True, data is normalized to 0-1 range. Default to True.
include_zero_val	considering events labeled as 0 as an additional gate when there is only one gate. Default to True.

Value

List of Dataframes.

Examples

```
magicPred()
```

magicPred_all	<i>magicPred_all</i>
---------------	----------------------

Description

function to predict on plain test data (no hierarchy)

Usage

```
magicPred_all(
  list_test_data,
  magic_model = NULL,
  ref_model_info = NULL,
  magic_model_n_gates = NULL,
  ref_data_train = NULL,
```

```

prop_down = NULL,
n_points_per_plot = NULL,
thr_dist = 0.05,
n_cores = 1,
normalize_data = T,
include_zero_val = T,
n_cores_all = 1,
verbose = F,
...
)

```

Arguments

list_test_data List of unlabeled dataframes. It has only the two columns of marker expression.

magic_model Global trained model to predict gates. It can be a single model or list of named models (each model trained on selected number of gates).

ref_model_info Template model to predic gates.

magic_model_n_gates
Global trained model to predict number of gates. If different from NULL, magic_model is expected to be a list of models to predict certain gates (e.g., 5 models for 2,3,4,5 or 6 gates).

ref_data_train Template data used to generate ref_model_info. Needed to calculate target-template distance.

n_points_per_plot
Number of points to consider for downsampling. Default to 500.

n_cores Number of cores to use to process one sample. Default to 1.

normalize_data If True, data is normalized to 0-1 range. Default to True.

include_zero_val
considering events labeled as 0 as an additional gate when there is only one gate. Default to True.

n_cores_all Number of cores to use across all samples. Default to 1.

verbose If True, print all message and disable tryCatch (any error will stop the execution). Default to False.

Value

List of Dataframes.

Examples

```
magicPred_all()
```

magicPred_hierarchy	<i>magicPred_hierarchy</i>
---------------------	----------------------------

Description

function to predict the gates on the ungated .fcs samples.

Usage

```
magicPred_hierarchy(list_test_sets, list_models_local, df_tree, n_cores = 1)
```

Arguments

`list_test_sets` contains the list of root dataframe for each ungated fcs file imported.

`list_models_local` contains the optimized local models pre-generated using the `magicTrain_local` function.

`df_tree` contains the info related to the populations hierarchy.

`n_cores` Number of cores to use. Default to 1.

Value

List of Dataframes.

Examples

```
magicPred_hierarchy()
```

magicTrain	<i>magicTrain</i>
------------	-------------------

Description

function to generate one training model based on a list of training sets (no hierarchy).

Usage

```
magicTrain(
  df_train,
  n_cores = 1,
  train_model = "rf",
  k_cv = 10,
  list_index_train = NULL,
  list_index_val = NULL,
  n_tree = 10,
```

```

    tune_lenght = 3,
    size_nnet_units = 100,
    decay_nnet = 0.1,
    method_control = "auto",
    type_y = "classes",
    seed_n = 40
)

```

Arguments

df_train	training dataframe generated by the get_train_data function.
n_cores	Number of cores to use. Default to 1.
train_model	Type of training model. Default to rf.
k_cv	Number of k for cross-validation (if method control=cv)
list_index_train	List of vector of indices to use in training for each fold.
list_index_val	List of vector of indices to use as held out data for each fold.
n_tree	Number of trees for random forest.
tune_lenght	Number of parameters values trained during cross-validation.
size_nnet_units	Number of units in hidden layer (if train_model=nnet).
decay_nnet	Decay parameter value for nnet model.
method_control	Type of training control: oob or cv or auto. Default to auto.
type_y	Type of response variable: classes (train to predict gates boundaries) or n_gates_info(train to predict number of gates).
seed_n	Set seed. Default to 40.

Value

model object.

Examples

```
magicTrain()
```

magicTrain_dt	<i>magicTrain_dt</i>
---------------	----------------------

Description

function to generate a random forest training model.

Usage

```
magicTrain_dt(
  Xtrain,
  Ytrain,
  k_cv = 10,
  list_index_train = NULL,
  list_index_val = NULL,
  tune_lenght = 5
)
```

Arguments

Xtrain	Dataframe of training features.
Ytrain	Dataframe of labels (one column).
k_cv	Number of k for cross-validation.
list_index_train	List of vector of indices to use in training for each fold.
list_index_val	List of vector of indices to use as held out data for each fold.
tune_lenght	Number of hyper parameters to test. Default to 5.

Value

model object.

Examples

```
magicTrain_dt()
```

magicTrain_hierarchy *magicTrain_local*

Description

function to generate the local training models using the list of hierarchical training set.

Usage

```
magicTrain_hierarchy(
  list_train_sets,
  n_tree = 10,
  train_model = "rf",
  method_control = "oob",
  n_cores = 1
)
```

Arguments

list_train_sets	List of labeled dataframe to train generated by the get_local_train function.
n_tree	Number of tree for random forest model.
train_model	Type of training model. Default to random forest ("rf").
method_control	Cross-validation method. Default to out-of-the-bag method (oob).
n_cores	Number of cores to use. Default to 1.

Value

List of models objects.

Examples

```
magicTrain_local()
```

magicTrain_knn	<i>magicTrain_knn</i>
----------------	-----------------------

Description

function to generate a random forest training model.

Usage

```
magicTrain_knn(
  Xtrain,
  Ytrain,
  k_cv = 10,
  list_index_train = NULL,
  list_index_val = NULL,
  tune_lenght = 5
)
```

Arguments

Xtrain	Dataframe of training features.
Ytrain	Dataframe of labels (one column).
k_cv	Number of k for cross-validation.
list_index_train	List of vector of indices to use in training for each fold.
list_index_val	List of vector of indices to use as held out data for each fold.
tune_lenght	Number of hyper parameters to test. Default to 5.

Value

model object.

Examples

```
magicTrain_knn()
```

magicTrain_nb	<i>magicTrain_nb</i>
---------------	----------------------

Description

function to generate a random forest training model.

Usage

```
magicTrain_nb(  
  Xtrain,  
  Ytrain,  
  k_cv = 10,  
  list_index_train = NULL,  
  list_index_val = NULL,  
  tune_lenght = 5  
)
```

Arguments

Xtrain	Dataframe of training features.
Ytrain	Dataframe of labels (one column).
k_cv	Number of k for cross-validation.
list_index_train	List of vector of indices to use in training for each fold.
list_index_val	List of vector of indices to use as held out data for each fold.
tune_lenght	Number of hyper parameters to test. Default to 5.

Value

model object.

Examples

```
magicTrain_nb()
```

magicTrain_nnet	<i>magicTrain_nnet</i>
-----------------	------------------------

Description

function to generate a neural net training model.

Usage

```
magicTrain_nnet(  
  Xtrain,  
  Ytrain,  
  k_cv = 10,  
  list_index_train = NULL,  
  list_index_val = NULL,  
  size = 100,  
  decay = 0.1,  
  tune_lenght = 5  
)
```

Arguments

Xtrain	Dataframe of training features.
Ytrain	Dataframe of labels (one column).
k_cv	Number of k for cross-validation.
list_index_train	List of vector of indices to use in training for each fold.
list_index_val	List of vector of indices to use as held out data for each fold.
size	Number of units in hidden layer (if train_model=nnet).
decay	Decay parameter value for nnet model.

Value

model object.

Examples

```
magicTrain_nnet()
```

magicTrain_rf	<i>magicTrain_rf</i>
---------------	----------------------

Description

function to generate a random forest training model.

Usage

```
magicTrain_rf(  
  Xtrain,  
  Ytrain,  
  list_index_train = NULL,  
  list_index_val = NULL,  
  n_tree = 10,  
  method_control = "oob",  
  k_cv = 10  
)
```

Arguments

Xtrain	Dataframe of training features.
Ytrain	Dataframe of labels (one column).
list_index_train	List of vector of indices to use in training for each fold.
list_index_val	List of vector of indices to use as held out data for each fold.
n_tree	Number of trees for random forest.
method_control	Type of training control: oob or cv. Default to oob.
k_cv	Number of k for cross-validation (if method control=cv).

Value

model object.

Examples

```
magicTrain_rf()
```

<code>magic_label_poly</code>	<i>magic_label_poly</i>
-------------------------------	-------------------------

Description

function to label points of dataframe based on polygon coordinates

Usage

```
magic_label_poly(df, polygon_df, label_pol = "1")
```

Arguments

<code>df</code>	Dataframe composed of two columns for marker expression of first (x axis = first column) and second marker (y axis = second column)
<code>polygon_df</code>	Dataframe containing the x coordinates (first column) and y coordinates (second column) of the current polygon to label.
<code>label_pol</code>	Label for current polygon.

Value

Dataframe

Examples

```
magic_label_poly()
```

<code>magic_label_rectangle</code>	<i>magic_label_rectangle</i>
------------------------------------	------------------------------

Description

function to label points of dataframe based on rectangle coordinates

Usage

```
magic_label_rectangle(df, x_min, x_max, y_min, y_max, label_pol = "1")
```

Arguments

<code>df</code>	Dataframe composed of two columns for marker expression of first (x axis = first column) and second marker (y axis = second column)
<code>x_min</code>	x coordinate minimum
<code>x_max</code>	x coordinate maximum
<code>y_min</code>	y coordinate minimum
<code>y_max</code>	y coordinate maximum
<code>label_pol</code>	Label for current polygon.

Value

Dataframe

Examples

`magic_label_rectangle()`

<code>magic_plot_wrap</code>	<i>magic_plot_wrap</i>
------------------------------	------------------------

Description

function to wrap all plots in one plot

Usage

`magic_plot_wrap(list_gated_data, n_col_wrap = 3, size_title = 10, ...)`

Arguments

- | | |
|------------------------------|--------------------------------------------------------------------------------------------------------------------------|
| <code>list_gated_data</code> | List of dataframes composed by three columns (could be generated by the <code>get_list_df_gated_plots()</code> function) |
| <code>n_col_wrap</code> | number of columns in the wrapped plot. Default to 3. |
| <code>size_title</code> | size of title of each plot. Default to 10. |

Value

wrapped plot

Examples

`magic_plot_wrap()`

map_to_parent	<i>map_to_parent</i>
---------------	----------------------

Description

function to get binary dataset with gate assignation based on the mother population of the selected pop (instead of the root).

Usage

```
map_to_parent(gh, binary_df)
```

Arguments

gh	GatingHierarchy.
binary_df	Dataframe generated by the map_to_root function.

Value

Dataframe.

Examples

```
map_to_parent()
```

map_to_root	<i>map_to_root</i>
-------------	--------------------

Description

it generates a dataset indicating what cells belong to the selected pop (1) 0 otherwise. Old name: pre_process_manual_binary() The dataset is always the dataset of the Root pop. gh and pop are mandatory arguments, dim and mode have a default value.

Usage

```
map_to_root(gh, pop)
```

Arguments

gh	GatingHierarchy.
pop	Name of the population to get events assignments.

Value

Dataframe.

Examples

```
map_to_root()
```

name_pop_gating	<i>name_pop_gating</i>
-----------------	------------------------

Description

function to get the name of all the pops of the gating hierarchy.

Usage

```
name_pop_gating(gh)
```

Arguments

gh GatingHierarchy.

Value

Vector of characters.

Examples

```
name_pop_gating()
```

post_process_gates	<i>post_process_gates</i>
--------------------	---------------------------

Description

function to post process the events after model prediction.

Usage

```
post_process_gates(
  gated_df,
  n_cores = 1,
  thr_dist = 0.15,
  include_zero = F,
  remove_centroids = T,
  type = "dist",
  concavity_val = 5,
  normalize_data = T,
  ...
)
```

Arguments

gated_df	dataframe with labels (third column).
n_cores	Number of cores. Default to 1.
thr_dist	Distance threshold for centroids calculation. Default to 0.15.
include_zero	Consider centroid of label 0. Default to False.
remove_centroids	Remove centroids too near each other based on thr_dist value.
type	Type of post-processing.
concavity_val	Concavity of polygons for the "polygon" type of post-processing

Value

Dataframe.

Examples

```
post_process_gates()
```

process_test_data	<i>process_test_data</i>
-------------------	--------------------------

Description

function to get test data correctly formatted.

Usage

```
process_test_data(
  test_data,
  prop_down = NULL,
  n_points_per_plot = 500,
  normalize_data = T
)
```

Arguments

test_data	Dataframe of bivariate markers expression.
prop_down	Proportion of events (downsampling). Default to NULL (downsampling using number of points).
n_points_per_plot	Number of points for downsampling.
normalize_data	If True, data is normalized to 0-1 range. Default to True.

Value

Dataframe.

Examples

```
process_test_data()
```

range01	<i>range01</i>
---------	----------------

Description

function to put data in range 0-1.

Usage

```
range01(x)
```

Arguments

x Vector of numbers to scale.

Value

Vector of numbers.

Examples

```
range01()
```

smooth_hull	<i>smooth_hull</i>
-------------	--------------------

Description

function to smooth concave polygons

Usage

```
smooth_hull(hull_df, spar = 0.7)
```

Arguments

hull_df	Dataframe generate by concaveman functions inside get_huget_hull_all_gates function
spar	Spar value to regulate smoothing process: higher value (max 1) higher smoothing.
buffer_dist	Regulate buffer distance before smoothing. Default to 500.

Value

Dataframe.

Examples

```
smooth_hull()
```

```
update_label_association
```

```
update_label_association
```

Description

function to update label association.

Usage

```
update_label_association(df)
```

Arguments

df Dataframe.

Value

Vector.

Examples

```
update_label_association()
```

Index

[add_labels_column](#), 3
[assign_events_to_nearest_centroids](#), 4

[check_polygons_intersection](#), 5
[compute_gates](#), 5
[convert_to_integers_chr](#), 6
[csv_to_dens](#), 6

[export_raw_gs_plots](#), 8
[exports_plots](#), 7
[extract_polygon_gates](#), 9

[flowmagic_pred_to_gs](#), 10
[flowmagic_pred_to_poly_gates](#), 10

[get_centroids](#), 11
[get_classes_expr_df](#), 12
[get_density_features](#), 12
[get_density_scores](#), 13
[get_dist_template](#), 14
[get_distance_loc_vs_test](#), 13
[get_flowframe_from_gs](#), 15
[get_hierarchy_all_pops](#), 15
[get_hull_all_gates](#), 16
[get_indices_cross_val](#), 17
[get_list_df_gated_plots](#), 18
[get_local_train_sets](#), 18
[get_pop_multiclass](#), 19
[get_pops_hierarchy_list](#), 19
[get_slot_hierarchy_list](#), 20
[get_test_sets](#), 21
[get_train_data](#), 21
[get_weights_density_features](#), 22

[import_gating_info](#), 23
[import_reference_csv](#), 23
[import_test_set_csv](#), 24
[import_test_set_fcs](#), 24

[magic_label_poly](#), 38
[magic_label_rectangle](#), 38

[magic_plot_wrap](#), 39
[magicPlot](#), 25
[magicplot_3D](#), 27
[magicPlot_template](#), 28
[magicPred](#), 28
[magicPred_all](#), 29
[magicPred_hierarchy](#), 31
[magicTrain](#), 31
[magicTrain_dt](#), 32
[magicTrain_hierarchy](#), 33
[magicTrain_knn](#), 34
[magicTrain_nb](#), 35
[magicTrain_nnet](#), 36
[magicTrain_rf](#), 37
[map_to_parent](#), 40
[map_to_root](#), 40

[name_pop_gating](#), 41

[post_process_gates](#), 41
[process_test_data](#), 42

[range01](#), 43

[smooth_hull](#), 43

[update_label_association](#), 44