Package 'flowMagic'

July 3, 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

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

Contents

`add_labels_column` *add_labels_column*

Description

function to add label association column.

Usage

```
add_labels_column(df, labels_assocation)
```

Arguments

`df`	Dataframe.
`labels_assocation`	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

gated_df	dataframe with labels (third column).
n_cores	Number of cores. Default to 1.
method_dist	Distance method calculation. Default to euclidean.
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

```
assign_events_to_nearest_centroids()
```

`check_polygons_intersection` *extract_polygon_gates*

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` *compute_gates*

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()
```

`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` *exports_plots*

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 = 7,
  h_val = 7,
  size_axis_text = 25,
  export_csv = 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.

x_lab	x-axis label.
y_lab	y-axis label.
size_title_x	Size x axis label.
size_title_y	Size y axis label.
aspect_ratio	Set aspect ratio. Default to NULL> If = 1, y and x axis ticks have same distance.
w_val	width value. Default to 7.
h_val	height value. Default to 7.
size_axis_text	Size of ticks labels.
export_csv	Export plot data as csv files. Default to False.

Examples

```
exports_plots()
```

`export_raw_gs_plots` *export_raw_gs_plots*

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

gs	GatingSet
path_output	Path to the directory where to export the plots.
n_cores	Number of cores to use. Default to 1.

x_lab	x-axis label.
y_lab	y-axis label.
w_val	width value. Default to 7 inches.
h_val	height value. Default to 7 inches.
size_points	Size points scatter plot.
return_data	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

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

Value

List of dataframes.

Examples

```
extract_polygon_gates()
```

get_centroids *get_centroids*

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

dens_df	Dataframe of density values.
original_df	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

df_dens	Dataframe of density estimates for both markers.
min_height	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

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

Value

List of p-values.

Examples

```
get_distance_loc_vs_test()
```

---

get_dist_template *get_dist_template*

Description

function to get distance between template and test data.

Usage

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

Arguments

matrix_scores	Matrix of density features generated by get_density_scores function.
dist_method	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()
```

get_hull_all_gates *get_hull_all_gates*

Description

function to get the convex hull of all gates.

Usage

```
get_hull_all_gates(gated_df, concavity_val = 1)
```

Arguments

gated_df	dataframe with labels (third column).
concavity_val	Values of concavity. Default to 1.

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

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

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` *get_slot_hierarchy_list*

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 *get_test_sets*

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 *get_train_data*

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 *import_sample_gated*

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 *import_reference_csv*

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 *import_test_set_csv*

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 *import_test_set*

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 = T,
  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.
show_legend	Show legend if type="ML". Default to True.
size_axis_text	Size of axis ticks labels. Default to 18.
size_title_x	Size of x axis label title.
size_title_y	Size of y axis label title.
treat_0_as_gate	Treat 0 label as gate. Defaul to False (0 label is background)
x_lab	Label of x axis.
y_lab	Label off y axis.
gates_to_plot	Select labels to plot.
apply_manual_scale	Apply predifined scale of colors. Default to True.
size_points	Size of points in scatter plot.
concavity_val	Concavity value. Default to 5. Higher value, less jagged boundaries
aspect_ratio	Aspect ratio value. If = 1, y and x axis have equal distance between the ticks labels. Default to NULL.
x_lim1	Minimum limit x axis. Default to NULL.
x_lim2	Max limit x axis. Default to NULL.
y_lim1	Minimum limit y axis. Default to NULL.
y_lim2	Max limit y axis. Default to NULL.

add_labels	add polygon labels. Default to FALSE.
map_label_polygon	map of polygon labels (to assign custom labels). Default to NULL.
size_pol_name	size polygon labels. Default to 6.
show_marginals	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

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

Value

plotly plot

Examples

```
magicplot_3D()
```

`magicPred` *magicPred*

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` *magicPred_all*

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` *magicPred_hierarchy*

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` *magicTrain*

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 = "oob",
  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. Default to oob.
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` *magicTrain_dt*

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 *magicTrain_knn*

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` *magicTrain_nb*

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 *magicTrain_nnet*

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 *magicTrain_rf*

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()
```

magic_label_poly *magic_label_poly*

Description

function to label points of dataframe based on polygon coordinates

Usage

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

Arguments

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

Value

Dataframe

Examples

```
magic_label_poly()
```

magic_label_rectangle *magic_label_rectangle*

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

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

Value

Dataframe

Examples

```
magic_label_rectangle()
```

`magic_plot_wrap` *magic_plot_wrap*

Description

function to wrap all plots in one plot

Usage

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

Arguments

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

Value

wrapped plot

Examples

```
magic_plot_wrap()
```

map_to_parent *map_to_parent*

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 *map_to_root*

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` *name_pop_gating*

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` *post_process_gates*

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 *process_test_data*

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` *range01*

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()
```

`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