Package 'GatingTree'

March 24, 2025

Version 0.2.0

Date 2025-03-23

Title Pathfinding Analysis of Group-Specific Effects in Cytometry Data

Description Implements the Gating Tree methodology, a novel pathfinding approach for analyzing high-dimensional cytometry data, including both flow and mass cytometry. This package circumvents traditional dimensional reduction and clustering by utilizing novel pathfinding algorithms that involve enrichment scores and gating entropy. These algorithms effectively explore the multidimensional marker space and identify group-specific features. The package includes functions for data transformation, visualization, and the application of advanced machine learning techniques, such as Random Forest, to enhance analysis. The methodology is designed to produce outputs that are immediately usable as gating strategies for downstream experimental applications.

Depends R (>= 4.2.0)

Imports utils, stats, graphics, grDevices, methods, rlang, ggplot2, gridExtra, data.tree, dunn.test, DiagrammeR, randomForest, dplyr, gplots

Suggests knitr, rmarkdown, KernSmooth, flowCore, HDCytoData

License file LICENSE

URL https://github.com/MonoTockyLab/GatingTree, https://MonoTockyLab.github.io/GatingTree

BugReports https://github.com/MonoTockyLab/GatingTree/issues

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder knitr

Contents

`AnalyzeNodeOverlaps` *Analyze Cell Overlaps Between Nodes And Generate Heatmap*

Description

Computes and visualizes the percentage overlap between nodes in a 'FlowObject', producing a heatmap of overlap percentages. Optionally, it can also display bar plots of specified parameters such as Enrichment, Entropy, or Average Proportion for the selected nodes.

Usage

```
AnalyzeNodeOverlaps(
  x,
  select_nodes = FALSE,
  n = NULL,
  graphics = TRUE,
  margin = 8,
  parameter = NULL
)
```

Arguments

`x`	A 'FlowObject' after PruneGatingTree.
`select_nodes`	Logical value indicating whether to select nodes interactively. Default is 'FALSE', which selects the top 'n' nodes based on significance.
`n`	Integer specifying the number of top nodes to select if 'select_nodes' is 'FALSE'. Default is '25'.

graphics	Logical indicating whether to use graphical selection when ‘select_nodes‘ is ‘TRUE‘. Default is ‘TRUE‘.
margin	Numeric value specifying the margins for the heatmap plot. Default is ‘8‘.
parameter	Character string specifying the parameter to plot alongside the heatmap. Can be ‘"Enrichment"‘, ‘"Entropy"‘, or ‘"Average Proportion"‘. Default is ‘NULL‘, which does not plot additional parameters.

Value

A matrix containing the percentage overlap between the selected nodes.

See Also

Other Unsupervised GatingTree Analysis: `ExtractTopNodes()`

Examples

```
## Not run:
# Basic usage with default settings
AnalyzeNodeOverlaps(x)

# Select nodes interactively
AnalyzeNodeOverlaps(x, select_nodes = TRUE)

## End(Not run)
```

---

convertGatingTreeToGraph

Convert Gating Tree to Graphical Representation

Description

This function converts a gating tree from a ‘FlowObject‘ into a graph object which can then be visualized or exported. It supports different modes of operation, allowing users to work with either all nodes, pruned nodes, or extracted top nodes based on previous analyses.

Usage

```
convertGatingTreeToGraph(
  x,
  mode = "Pruned",
  size_factor = 1,
  fontsize = 20,
  all_labels = TRUE
)
```

Arguments

x	A 'FlowObject' containing the gating tree data.
mode	A character string specifying which part of the gating tree to use: - 'All': Uses all nodes from the gating tree. - 'Pruned': Uses nodes pruned by 'PruneGatingTree'. - 'Extracted': Uses top nodes extracted by 'ExtractTopNodes'. The default is 'Pruned'.
size_factor	A numeric value used to adjust the size of the nodes in the graph. Default is 1.
fontsize	An integer specifying the font size used for node labels in the graph. Default is 20.
all_labels	A logical indicating whether to label all nodes or not. Default is TRUE.

Value

A graph object that can be further processed with graph rendering or exporting functions.

See Also

Other GatingTree Visualization: `convertToDataTree()`, `convert_to_diagrammer()`

Examples

```
## Not run:
  data(FlowObjectExample)
  graph <- convertGatingTreeToGraph(FlowObjectExample)
  render_graph(graph, width = 600, height = 600)
  export_graph(graph, file_name= "PrunedGatingTree.pdf")

## End(Not run)
```

---

convertToDataTree *Convert Gating Tree Node to Data Tree Node*

Description

This function converts a node from a gating tree into a data tree node format, suitable for further processing or visualization.

Usage

```
convertToDataTree(node, pathName = "rootNode")
```

Arguments

node	A node from a gating tree, usually containing statistical information and child nodes.
pathName	A string representing the node path, defaults to "rootNode".

Value

A data tree node object structured for hierarchical data representations.

See Also

Other GatingTree Visualization: convertGatingTreeToGraph(), convert_to_diagrammer()

Examples

```
## Not run:
# Assuming `node` is an object from a gating analysis tree:
convertedNode <- convertToDataTree(node)

## End(Not run)
```

---

convert_to_diagrammer *Convert Gating Tree to DiagrammeR Graph*

Description

This function takes a gating tree object and constructs a graphical representation using the DiagrammeR package. It visually represents the enrichment, entropy, and optionally the average proportion of nodes in the gating tree.

Usage

```
convert_to_diagrammer(
  root,
  size_factor = 1,
  fontsize = 12,
  average_proportion = FALSE,
  all_labels = TRUE
)
```

Arguments

root	The root node of the gating tree object, which must have properties like 'isRoot', 'name', 'CurrentEnrichment', 'CurrentEntropy', and optionally 'AverageProportion'.
size_factor	A multiplier for node sizes in the graph, allowing customization based on enrichment or average proportion values.
fontsize	The size of font to be used in the graph.
average_proportion	A logical flag indicating whether to use the average proportion of nodes to adjust node sizes and color gradient based on enrichment values.
all_labels	Logical. If TRUE, all parameters are included in each node within the output graph.

Details

The function recursively traverses the gating tree, starting from the root, adding nodes to the graph with labels that include relevant metrics. Node size and color are determined by either the enrichment or entropy values, depending on the 'average_proportion' flag.

Value

Returns a DiagrammeR graph object representing the gating tree with nodes colored and sized according to specified metrics.

See Also

Other GatingTree Visualization: `convertGatingTreeToGraph()`, `convertToDataTree()`

Examples

```
## Not run:
  # Assuming 'root' is your gating tree root node
  graph <- convert_to_diagrammer(root, size_factor = 1, average_proportion = TRUE)
  # To view the graph
  DiagrammeR::render_graph(graph)

## End(Not run)
```

`CreateFlowObject` *Create FlowObject by importing flow cytometric data*

Description

This function constructs a new FlowObject using either data imported from CSV files or data provided directly as arguments.

Usage

```
CreateFlowObject(
  path = ".",
  select = TRUE,
  sample_file = NULL,
  Data = NULL,
  sampledef = NULL,
  experiment_name = NULL
)
```

Arguments

`path`	Path to CSV files.
`select`	Whether to select files via an interactive window.
`sample_file`	When select is FALSE, a vector of sample file names.
`Data`	A data frame containing flow cytometric expression data. If provided, file reading is skipped.
`sampledef`	An object of class SampleDef or a list containing sample definitions. If provided, sampledef population is skipped.
`experiment_name`	Name of the experiment.

Value

A FlowObject

See Also

Other Initialization: SampleDef

Examples

```
## Not run:
x  <-  CreateFlowObject()

## End(Not run)
```

---

createGatingTreeObject

Create a Gating Tree Object

Description

Initializes and populates a gating tree based on user-defined criteria, handling decision-making through multiple layers of a gating hierarchy. This function integrates gating rules, applies negative gating definitions, and assesses cell population statistics to manage the flow cytometry data analysis process.

Usage

```
createGatingTreeObject(
  x,
  select_markers = FALSE,
  graphics = FALSE,
  markers = NULL,
  maxDepth = 3,
  min_cell_num = 25,
  expr_group = NULL,
  ctrl_group = NULL,
  verbose = TRUE
)
```

Arguments

x	An object, expected to be of class 'FlowObject', containing the initial data and parameters for gating.
select_markers	Logical; if TRUE, allows the user to select markers interactively.
graphics	Logical; if TRUE, enables graphical selection of markers.
markers	Optional; a vector of markers to be included if not choosing interactively.
maxDepth	Integer; the maximum depth of the gating tree, controlling how many levels of decision nodes can be created.
min_cell_num	The minimal number of cells allowed in nodes.

expr_group	Optional; a character to specify the experimental group. If NULL, this is determined interactively.
ctrl_group	Optional; a character to specify the control group. If NULL, this is determined interactively.
verbose	Logical indicating whether to print progress messages and outputs. Default is TRUE.

Details

The function first checks for the type of the input object to ensure it matches expected classes. It then extracts necessary data and parameters from the object, such as negative gating thresholds and sample definitions. Depending on the options, it may allow interactive selection of markers. The function constructs a hierarchical tree where each node represents a gating decision based on statistical calculations like entropy and enrichment, which are used to determine the next steps in the gating process or to terminate the process.

The gating process involves: - Merging data with sample definitions. - Calculating initial gating statistics. - Recursively creating child nodes based on gating outcomes and thresholds. - Dynamically managing markers and gating paths based on user-defined depth and available data.

Value

Modifies the input object by adding a 'GatingTreeObject' that contains the entire structure of gating decisions and nodes.

See Also

Other GatingTree: ExtractGatingTree(), GatingTreeToDF(), PlotDeltaEnrichment(), PlotDeltaEnrichmentPru
PruneGatingTree(), findNodeByPath(), getNode()

Examples

```
## Not run:
# Assuming 'x' is properly instantiated and contains necessary gating setups:
x <- createGatingTreeObject(x, select_markers = TRUE, graphics = FALSE, maxDepth = 3)

## End(Not run)
```

DefineNegatives *Interactively determine gates to define negative (and positive) regions for each marker expression*

Description

This function allows interactive setting of thresholds on plots to define negative and positive gates for markers. Users can choose to plot data as 2D scatter plots or as density plots, and may utilize pseudocolor to enhance visualization.

Usage

```
DefineNegatives(
  x,
  select = TRUE,
  max_cells_displayed = 30000,
  y_axis_var = NULL,
  pseudocolour = TRUE
)
```

Arguments

x	A FlowObject containing the data.
select	Logical, whether to interactively select markers to be processed. If FALSE, all logged expression variables will be used. Default is TRUE.
max_cells_displayed	Maximum number of cells displayed in plots for determining a negative threshold.
y_axis_var	The variable to use for the y-axis in the interactive plot, or 'Density' to use a density plot. If NULL (default), the user will be prompted to select a variable from x@Data.
pseudocolour	Logical, whether to use pseudocolor based on density in scatter plots. Default is TRUE. The option FALSE will use monochrome.

Value

A modified FlowObject containing updated threshold values for autofluorescence in the selected variables in the slot QCdata. You can repeat DefineNegatives to renew or adjust some or all of the negative thresholds for variables.

See Also

Other Data Transformation: LogData(), NormAF(), PlotDefineNegatives(), PlotNormAF()

Examples

```
## Not run:
x <- DefineNegatives(x)

## End(Not run)
```

---

export_negative_gate_def

Export Negative Gate Definitions from FlowObject

Description

This function exports the negative gate definitions from a FlowObject to a CSV file. It will stop if the negative gate definitions are not found within the FlowObject.

Usage

```
export_negative_gate_def(x, filename = "negative_gate_def.csv")
```

Arguments

x	A FlowObject containing negative gate definitions.
filename	The name of the file to which the negative gate definitions will be written.

Value

The FlowObject, unchanged, with the side effect of writing to a file.

Examples

```
## Not run:
export_negative_gate_def(x, filename = "negative_gate_def.csv")

## End(Not run)
```

ExtractGatingTree *Extract and Finalize the Gating Tree Object*

Description

This function performs a tree extraction by specifying node paths. The resulting tree will be stored in the ExtractedGatingTreeObject slot of the FlowObject.

Usage

```
ExtractGatingTree(x, node_paths)
```

Arguments

x	A FlowObject that has been previously processed with GatingTreeToDF and contains gating information in the Gating slot, including GatingTreeDF and PrunedGatingTreeObject.
node_paths	A character vector of node paths to be used in the extraction process.

Details

Node paths should use the format in GatingTreeDF's markers_up_to_max slot.

Value

The updated FlowObject with a new slot Gating$ExtractedGatingTreeObject containing the final, pruned gating tree.

See Also

Other GatingTree: GatingTreeToDF(), PlotDeltaEnrichment(), PlotDeltaEnrichmentPrunedTree(), PruneGatingTree(), createGatingTreeObject(), findNodeByPath(), getNode()

Examples

```
## Not run:
# Assume 'x' is a FlowObject that has been processed with GatingTreeToDF and PruneGatingTree:
x <- ExtractGatingTree(x, node_paths = c("CD4.logdata.pos_CD8.logdata.neg",
"CD4.logdata.pos_CD19.logdata.pos"))

## End(Not run)
```

---

extractNodes *Extract All Nodes from GatingTree*

Description

This function recursively extracts all nodes from the specified GatingTree object within a FlowObject, either the full GatingTree or the pruned version.

Usage

```
extractNodes(x, pruned = FALSE)
```

Arguments

x	A FlowObject containing a GatingTree.
pruned	Logical indicating whether to extract nodes from the pruned GatingTree. Defaults to FALSE.

Value

A data frame containing details of all nodes in the specified GatingTree.

Examples

```
## Not run:
  full_tree_nodes <- extractNodes(x)
  pruned_tree_nodes <- extractNodes(x, pruned = TRUE)

## End(Not run)
```

ExtractTopNodes *Extract Top Nodes from Node Overlaps in a FlowObject*

Description

This function analyzes node overlaps stored within a 'FlowObject' and extracts top nodes based on clustering and composite scores derived from entropy, enrichment, and average proportion. It performs hierarchical clustering on the dendrogram of node overlaps, selects the best node from each cluster based on a composite score, and updates the gating tree with these nodes.

Usage

```
ExtractTopNodes(x, cluster_num = 6)
```

Arguments

x	A 'FlowObject' containing the gating tree and node overlaps data.
cluster_num	An integer specifying the number of clusters to cut the dendrogram into. Defaults to 6.

Value

The modified 'FlowObject' with updated 'ExtractPrunedGatingTree' and 'ExtractPrunedGatingTreePaths' slots indicating the pruned gating tree and the paths of selected top nodes.

See Also

Other Unsupervised GatingTree Analysis: AnalyzeNodeOverlaps()

Examples

```
## Not run:
x <- ExtractTopNodes(x, cluster_num = 6)

## End(Not run)
```

---

findNodeByPath *Find a Node by Path in a Gating Tree*

Description

Traverses a gating tree starting from a specified root node to find and return a node located at a given path. The path should be a sequence of node names indicating the traversal route from the root to the target node.

Usage

```
findNodeByPath(rootNode, path)
```

Arguments

rootNode	The root node of the gating tree from which to start the search.
path	A character vector representing the path to the desired node. Each element of the vector should correspond to a node name at each level of the tree.

Value

Returns the node at the specified path if found.

See Also

Other GatingTree: ExtractGatingTree(), GatingTreeToDF(), PlotDeltaEnrichment(), PlotDeltaEnrichmentPru
PruneGatingTree(), createGatingTreeObject(), getNode()

Examples

```
## Not run:
rootNode <- createGatingTreeObject(...) # setup initial node, assumes function definition
path <- c("rootNode", "CD4pos", "CD8neg") # example path to find a specific node
tryCatch({
  targetNode <- findNodeByPath(rootNode, path)
  print(targetNode)
}, error = function(e) {
  cat("Error in findNodeByPath: ", e$message, "\n")
})

## End(Not run)
```

`flowObject` *Create a new FlowObject*

Description

This function constructs a new FlowObject using various inputs related to flow cytometry analysis, including data, sample definitions, metadata, and data processing configurations.

Usage

```
flowObject(
  Data = data.frame(),
  sampledef = new("SampleDef"),
  metadata = list(),
  prep = list(),
  Clustering = list(),
  Gating = list(),
  QCdata = list(),
  Transformation = list(),
  Model = list()
)
```

Arguments

`Data`	A data frame containing flow cytometric expression data.
`sampledef`	An object of class SampleDef, specifying sample definitions.
`metadata`	A list containing annotation data for the flow cytometry experiment.
`prep`	A list of character string vectors defining the samples and controls.
`Clustering`	A list containing cell identities and clustering data.
`Gating`	A list containing outputs of gating, including GatingTree objects.
`QCdata`	A list containing the quality control information for the flow cytometry data.
`Transformation`	A list containing settings for data transformation, including the log data settings.
`Model`	A list containing Data Models and Machine Learning Models

Value

An object of class FlowObject, which encapsulates all provided data and settings for flow cytometry analysis.

`FlowObject-class` *A class representing a FlowObject object*

Description

This class provides a representation of a FlowObject object.

Slots

`Data` A data frame containing flow cytometric expression data

`sampledef` A SampleDef object containing sample file definitions.

`metadata` A list containing annotation data

`prep` A list containing sample file definitions.

`Clustering` A list containing cell identities and clustering data

`Gating` A list containing gating information.

`QCdata` A list containing quality control information.

`Transformation` A list containing settings for data transformation, including the log data settings.

`Model` A list containing data models and machine learning models.

`GatingTreeRandomForest` *Perform Random Forest Analysis Using GatingTree Node Data*

Description

This function trains a Random Forest model using node-level percentage data from a FlowObject that includes a GatingTree. The fitted Random Forest model and associated metadata are then stored in the '@Model' slot of the same FlowObject.

Usage

```
GatingTreeRandomForest(
  train_x,
  node_paths = NULL,
  ntree = 100,
  mtry = NULL,
  expr_group = NULL,
  ctrl_group = NULL,
  class_weight = FALSE,
  filter = "all",
  q = 0.75
)
```

Arguments

train_x	A FlowObject containing a GatingTree and node-level percentage data in 'train_x@Gating$PrunedGa
node_paths	A character vector of node paths to include as predictors. If 'NULL', all columns containing "logdata" are used.
ntree	Number of trees to grow in the Random Forest. Defaults to 100.
mtry	Number of variables randomly sampled at each split in the Random Forest. The defaul is NULL and uses the automatic selection implemented in randomForest.
expr_group	Optional. A character vector to specify the experimental group when class weight is considered.
ctrl_group	Optional. A character vector to specify the control group when class weight is considered.
class_weight	Logical. Whether to consider class weight. The default is FALSE.
filter	A character to spercify how to filter nodes. The option 'all' will consider all enrichment score, entropy, and average proportion of node with equal weight. The option 'enrichment' will use enrichment score only.
q	Number between 0 and 1. Quantile value for the filter parameter to select top nodes. For example, 0.95 will select nodes above the 95th percentile.

Value

The updated FlowObject with a new Random Forest model stored in the '@Model' slot under the name '"RandomForestGatingTree"'.

See Also

Other GatingTree Random Forest Analysis: predictGatingTreeRandomForest()

Examples

```
## Not run:
  train_x <- GatingTreeRandomForest(train_x)

## End(Not run)
```

GatingTreeToDF *Convert Gating Tree Object to Data Frames*

Description

This function processes a Gating Tree object from a flow cytometry analysis package, extracting and calculating various statistics such as enrichment, entropy, and delta values for each gate. It returns the object with additional data frames attached that contain detailed analysis results.

Usage

```
GatingTreeToDF(x)
```

Arguments

x	FlowObject.

Details

The function traverses the gating tree recursively to collect enrichment and entropy values from each node. It calculates deltas of enrichment values as differences between consecutive gates. Additionally, it computes a maximum enrichment statistic for each path through the tree, sorting these values to identify the most significant gates. Interaction effects (IE) and a detailed breakdown of changes in enrichment values are also calculated and stored in the returned object.

Value

FlowObject with a GatingTreeDF.

See Also

Other GatingTree: `ExtractGatingTree()`, `PlotDeltaEnrichment()`, `PlotDeltaEnrichmentPrunedTree()`, `PruneGatingTree()`, `createGatingTreeObject()`, `findNodeByPath()`, `getNode()`

Examples

```
## Not run:
x <- GatingTreeToDF(x)

## End(Not run)
```

---

getNode *Retrieve a Node from a Gating Tree*

Description

This function traverses a gating tree structure to retrieve a node at a specified path.

Usage

```
getNode(gatingTreeObject, path)
```

Arguments

`gatingTreeObject`	The root object of the gating tree containing child nodes.
`path`	A vector of node names specifying the path to the desired node.

Value

The node object at the specified path.

See Also

Other GatingTree: `ExtractGatingTree()`, `GatingTreeToDF()`, `PlotDeltaEnrichment()`, `PlotDeltaEnrichmentPru`
`PruneGatingTree()`, `createGatingTreeObject()`, `findNodeByPath()`

Examples

```
## Not run:
gatingTreeObject <- createGatingTreeObject(...) # assuming this function exists
path <- c("rootNode", "CD4pos")
getNode(gatingTreeObject, path)

## End(Not run)
```

`import_negative_gate_def` *Import Negative Gate Definitions into FlowObject*

Description

This function imports a specified negative gate definition data frame into a FlowObject. The data frame must contain exactly two columns: 'variable' and 'negative.gate'.

Usage

```
import_negative_gate_def(x, negative_gate_def)
```

Arguments

`x`	A FlowObject.
`negative_gate_def`	A data frame containing the negative gate definitions.

Value

The modified FlowObject with updated negative gate definitions.

Examples

```
## Not run:
x <- import_negative_gate_def(x, negative_gate_def = my_gate_definitions)

## End(Not run)
```

`LogData` *Log fluorescence data*

Description

Log fluorescence data

Usage

```
LogData(x, graphics = TRUE, variables = NULL, prompt = FALSE)
```

Arguments

x	A FlowObject or TockyPrepData containing non-logged, raw fluorescence data
graphics	Whether to use a graphic device to choose variables for log transformation.
variables	A character vector for defining the variables to be log-transformed.
prompt	Whether to invoke a prompt for asking optional questions.

Value

logged data. FlowObject will include the slot logdata_parameters in the slot Transformation, which includes a list object of logged channel names.

See Also

Other Data Transformation: DefineNegatives(), NormAF(), PlotDefineNegatives(), PlotNormAF()

Examples

```
## Not run:
x  <-  LogData(x) #x is a FlowObject

## End(Not run)
```

NormAF *Moderate Extreme Negative Outliers with Random Values within Noise*

Description

This function moderates extreme negative outliers in the specified variables of a FlowObject by replacing these values with random numbers. These random numbers are drawn from a normal distribution determined by the data lying within the 'negative_gate_def' thresholds in the FlowObject. It is typically used to handle outliers in flow cytometry data after defining negative gates using the DefineNegative function.

Usage

```
NormAF(x, var = NULL, plot = FALSE)
```

Arguments

x	A FlowObject that has already been processed using DefineNegative.
var	A character vector specifying the variables (markers) in the FlowObject for which the moderation of extreme negative values should be performed. If NULL, the user is prompted to select variables interactively.
plot	Logical, whether to produce diagnotic plots.

Value

A modified FlowObject in which extreme negative values in the specified variables are replaced with random numbers based on the distribution of values within the defined negative gates. This modification is intended to reduce the impact of extreme outliers on subsequent analyses.

See Also

Other Data Transformation: DefineNegatives(), LogData(), PlotDefineNegatives(), PlotNormAF()

Examples

```
## Not run:
  # Assuming 'x' is a valid FlowObject with required preprocessing:
  x <- NormAF(x, var = c("marker1", "marker2"))

## End(Not run)
```

PlotDefineNegatives *Plot the threshold for for each marker expression as per determined by DefineNegatives*

Description

Plot the threshold for for each marker expression as per determined by DefineNegatives

Usage

```
PlotDefineNegatives(
  x,
  y_axis_var = NULL,
  output = FALSE,
  outputFile = "DefineNegativePlot.pdf",
  panel = NULL,
  variables = NULL
)
```

Arguments

x	A FlowObject.
y_axis_var	The variable to use for the y-axis in the generated plots, or 'Density' to use a density plot.
output	If TRUE, plots are generated as a file. The default is FALSE and shows plots in X window.
outputFile	When output is TRUE, this defines the filename of the output file.
panel	The number of panels to be included. If NULL, all panels are included in the output plot.
variables	The variables to be plotted. The default is NULL, showing all variable data.

Value

The same FlowObject is returned for safety.

See Also

Other Data Transformation: DefineNegatives(), LogData(), NormAF(), PlotNormAF()

Examples

```
## Not run:
PlotDefineNegatives(x)

## End(Not run)
```

`PlotDeltaEnrichment` *Plot Delta Enrichment or Interaction Effects for Each Marker*

Description

This function takes a complex object containing gating tree data and performs statistical tests to evaluate the significance of delta enrichment or interaction effects across markers. It plots the results as boxplots and returns the modified object with additional annotations based on the analysis.

Usage

```
PlotDeltaEnrichment(x)
```

Arguments

x	FlowObject.

Details

The function conducts Kruskal-Wallis tests to determine the significance of differences across markers, followed by Dunn's test for post-hoc analysis with Bonferroni correction if significant.

Value

Returns the object 'x' with statistical values

See Also

Other GatingTree: `ExtractGatingTree()`, `GatingTreeToDF()`, `PlotDeltaEnrichmentPrunedTree()`, `PruneGatingTree()`, `createGatingTreeObject()`, `findNodeByPath()`, `getNode()`

Examples

```
## Not run:
  x <- PlotDeltaEnrichment(x)

## End(Not run)
```

PlotDeltaEnrichmentPrunedTree *Plot Delta Enrichment or Interaction Effects for Each Marker Using Pruned GatingTree*

Description

Plot Delta Enrichment or Interaction Effects for Each Marker Using Pruned GatingTree

Usage

```
PlotDeltaEnrichmentPrunedTree(x, q = 0)
```

Arguments

x	FlowObject after applying PruneGatingTree.
q	A numeric value between 0 and 1 as a quantile threshold for extracting top markers.

Details

The function conducts Kruskal-Wallis tests to determine the significance of differences across markers, followed by Dunn's test for post-hoc analysis with Bonferroni correction if significant.

Value

Returns the object 'x' with statistical values

See Also

Other GatingTree: ExtractGatingTree(), GatingTreeToDF(), PlotDeltaEnrichment(), PruneGatingTree(), createGatingTreeObject(), findNodeByPath(), getNode()

Examples

```
## Not run:
  x <- PlotDeltaEnrichmentPrunedTree(x)

## End(Not run)
```

plotFlow2D *Two-Dimensional Plot with Gating and Quadrant Statistics*

Description

This function plots a two-dimensional flow cytometry data highlighting the gated regions and calculates quadrant statistics. It handles data preprocessing to manage outliers by replacing extreme negative values with values from a normal distribution within specified gates. The function is ideal for analyzing and visualizing flow cytometry data, providing insights into the distribution of cell populations across specified markers.

Usage

```
plotFlow2D(
  x,
  graphics = FALSE,
  output = "output",
  markers = NULL,
  states = NULL,
  gating = TRUE,
  split_group = TRUE,
  max_cells_displayed = 30000
)
```

Arguments

x	A FlowObject that has already been processed using DefineNegative and NormAF.
graphics	Logical, if TRUE, enables graphical selection of markers and gating thresholds via a GUI; otherwise, selections must be input manually. Defaults to FALSE.
output	The directory path where the output plots and data summaries will be saved.
markers	Optionally, a vector of marker names to be used for gating; if NULL, the function prompts for selection.
states	A vector indicating the gating state ('positive' or 'negative') for each marker; if NULL, the function prompts for selection.
gating	Logical, if TRUE, applies gating based on the markers and states provided; defaults to TRUE.
split_group	Logical, if TRUE, splits the data by 'group' variable within the dataset for separate analysis and plotting; defaults to TRUE.
max_cells_displayed	The maximum number of cells to display in the plots, which can help manage performance and clarity in visualizing dense datasets.

Value

Returns the same FlowObject for safety.

Examples

```
## Not run:
  plotFlow2D(x)

## End(Not run)
```

`PlotNodeScatterPlot` *Plot Node Scatter Plot*

Description

This function creates a scatter plot visualization of node percentages based on the given path through a gating tree. It computes and displays statistical summaries including means and standard deviations by group, adds error bars to the plot, and marks significant differences with asterisks.

Usage

```
PlotNodeScatterPlot(x, path)
```

Arguments

`x`	A FlowObject with GatingTreeObject
`path`	A character vector specifying the path through the gating hierarchy.

Value

A ggplot object representing the node percentage scatter plot with error bars and optionally asterisks denoting statistical significance.

Examples

```
## Not run:
PlotNodeScatterPlot(x, c("path", "to", "node"))

## End(Not run)
```

`PlotNormAF` *Plot Effects of Autofluorescnce Modelling*

Description

This function moderates extreme negative outliers in the specified variables of a FlowObject by replacing these values with random numbers. These random numbers are drawn from a normal distribution determined by the data lying within the 'negative_gate_def' thresholds in the FlowObject. It is typically used to handle outliers in flow cytometry data after defining negative gates using the DefineNegative function.

Usage

```
PlotNormAF(x, graphics = FALSE, max_cells_displayed = 30000, output = "output")
```

Arguments

x	A FlowObject that has already been processed using DefineNegative and NormAF.
graphics	Logical. The default is FALSE, showing variable options in console.
max_cells_displayed	The number of cells displayed in plots for determining a negative therehold.
output	A character for the name of output directory.

Value

The same FlowObject is returned for safety.

See Also

Other Data Transformation: DefineNegatives(), LogData(), NormAF(), PlotDefineNegatives()

Examples

```
## Not run:
  x <- PlotNormAF(x)

## End(Not run)
```

---

predictGatingTreeRandomForest

Predict Using GatingTreeRandomForest Model

Description

This function uses the GatingTreeRandomForest model stored in a 'FlowObject' to make predictions on a new dataset. It prepares the test dataset, applies the gating conditions specified in the model, and then uses the Random Forest model to predict outcomes.

Usage

```
predictGatingTreeRandomForest(train_x, test_y)
```

Arguments

train_x	A 'FlowObject' that contains a previously fitted 'GatingTreeRandomForest' model.
test_y	A 'FlowObject' containing the test dataset for prediction.

Value

A list containing:

- predicted_scores: The probabilities predicted by Random Forest.
- test_data: The test data used for Random Forest.
- results_pred: A data frame for predicted scores.

See Also

Other GatingTree Random Forest Analysis: `GatingTreeRandomForest()`

Examples

```
## Not run:
  train_x <- GatingTreeRandomForest(train_x)
  predictions <- predictGatingTreeRandomForest(train_x, test_y)
  head(predictions$predicted_scores)

## End(Not run)
```

PruneGatingTree *Prune a Gating Tree Based on Statistical Criteria*

Description

This function prunes a gating tree by applying various statistical thresholds to the nodes based on entropy, enrichment, and average proportion metrics. Nodes that do not meet the specified criteria are pruned from the tree. Additionally, p-values are adjusted for multiple comparisons. In the current implementation the output PrunedGatingTreeNodePercentages will have nodes filtered only through a threshold for average proportion.

Usage

```
PruneGatingTree(
  x,
  max_entropy = 0.9,
  min_enrichment = 0.1,
  min_average_proportion = 0.001,
  p_adjust_method = "BY",
  theta = NULL
)
```

Arguments

`x`	An object, expected to be of class 'FlowObject', containing gating tree data and metadata.
`max_entropy`	Maximum allowable entropy for a node to remain in the gating tree.
`min_enrichment`	Minimum enrichment required for a node to remain in the gating tree.
`min_average_proportion`	Minimum average proportion of cells required for a node to remain in the gating tree.
`p_adjust_method`	A character string indicating the method to be used for adjusting p-values for multiple comparisons. Defaults to 'BY' (Benjamini-Yekutieli).
`theta`	A numeric threshold added to the enrichment values for each node. This threshold is used to enforce a criterion where only nodes showing a steady increase in enrichment, greater than this threshold, can be considered for retention in the pruned gating tree. The default is zero.

Details

The function first identifies nodes that meet specified entropy and enrichment criteria, computes statistical metrics for these nodes, and then prunes the gating tree based on these metrics and average proportion criteria. Adjusted p-values are calculated to account for multiple testing.

Value

Returns the modified object 'x' with the gating tree pruned according to the specified parameters. The function also attaches the pruned gating tree and a data frame containing node statistics to the object.

See Also

Other GatingTree: `ExtractGatingTree()`, `GatingTreeToDF()`, `PlotDeltaEnrichment()`, `PlotDeltaEnrichmentPru`
`createGatingTreeObject()`, `findNodeByPath()`, `getNode()`

Examples

```
## Not run:
  updated_object <- PruneGatingTree(x, min_enrichment = 0.5,max_entropy =0.5)

## End(Not run)
```

SampleDef *SampleDef*

Description

A class to represent a sample definition object

This function updates a FlowObject with a SampleDef object created from a CSV file or a provided data frame.

Usage

```
SampleDef(
  x,
  sample_df = NULL,
  path = "sampledef",
  file = "sample.csv",
  group_column = "group",
  confirm = TRUE
)
```

Arguments

x	A FlowObject.
sample_df	Optional data frame containing the sample definitions with columns 'file' and 'group'.
path	A character string specifying the path to the directory containing the CSV file. Default is './sampledef'.

file	A character string specifying the name of the CSV file. Default is 'sample.csv'.
group_column	A character string specifying the column name for sample grouping in the CSV file.
confirm	If TRUE, prompts confirmation of editing the CSV file.

Value

A modified FlowObject containing a SampleDef object.

See Also

Other Initialization: `CreateFlowObject()`

Examples

```
## Not run:
x <- SampleDef(x)

## End(Not run)
```

`showSampleDef` *Show SampleDef*

Description

Show SampleDef

Usage

```
showSampleDef(x)
```

Arguments

x	A FlowObject.

Examples

```
## Not run:
showSampleDef(x)

## End(Not run)
```

Index