Package ‘spade’

April 10, 2015

Title SPADE -- An analysis and visualization tool for Flow Cytometry

Version 1.14.0

Author M. Linderman, P. Qiu, E. Simonds, Z. Bjornson

Description SPADE, or Spanning tree Progression of Density normalized Events, is an analysis and visualization tool for high dimensional flow cytometry data that organizes cells into hierarchies of related phenotypes.

Maintainer Zach Bjornson <`bjornson@stanford.edu`>

Imports Biobase, flowCore

Suggests flowViz

Depends R (>= 2.11), igraph, Rclusterpp

License GPL-2

URL `http://cytospade.org`

biocViews FlowCytometry, GraphAndNetwork, GUI, Visualization, Clustering

R topics documented:

`SPADE.addClusterToFCS` *Annotate observations in FCS file with cluster assignment*

Description

Annotate observations in a FCS file with cluster assignment

Usage

```
SPADE.addClusterToFCS(infilename, outfilename, clusterfilename, cols = NULL, arcsinh_cofactor=NULL, tr
```

Arguments

`infilename`	Name of input FCS file
`outfilename`	Name of output FCS file
`clusterfilename`	Name of FCS file with subset of cells used in clustering
`cols`	Usually a vector of strings specifying the columns to be used in the density calculation, e.g., c("(Cd110)D","(Cs111)D"). Strings will be matched against the parameter names extracted from the FCS file. The default=NULL will use all parameters.
`arcsinh_cofactor`	DEPRECATED. Cofactor used in arcsinh transform `asinh(data/arcsinh_cofactor)` of data.
`transforms`	Transform object to apply to data. A single transform object will be applied to all channels. To apply different transforms to specific channels use a named vector of transform objects (where names are parameter names).
`comp`	Apply compensation matrix if present in SPILL or SPILLOVER keywords

Value

The name of the written file is returned.

Note

Underlying implementations have been parallelized with OpenMP. Set `OMP_NUM_THREADS` in environment to control the number of threads used.

Author(s)

Michael Linderman

See Also

`SPADE.FCSToTree`

Examples

```
# Not run
## Load two-parameters sample data included in package
#data_file_path = paste(installed.packages()["spade","LibPath"],"spade","extdata","SimulatedRawData.fcs",sep=.P

#output_dir <- tempdir()
#
## Compute and annotate FCS file with density
#density_file_path <- paste(output_dir,.Platform$file.sep,basename(data_file_path),".density.fcs",sep="")
#SPADE.addDensityToFCS(data_file_path, density_file_path, cols=c("marker1","marker2"))

## Downsample FCS file based on density
#downsample_file_path <- paste(output_dir,.Platform$file.sep,basename(data_file_path),".density.fcs",sep="")
#SPADE.downsampleFCS(density_file_path, downsample_file_path)

## Create tree from downsampled FCS file
#cells_file_path <- paste(output_dir,"clusters.fcs",sep="")
#clust_file_path <- paste(output_dir,"clusters.table",sep="")
#graph_file_path <- paste(output_dir,"mst.gml",sep="")
#SPADE.FCSToTree(downsample_file_path, cells_file_path, graph_file_path, clust_file_path, cols=c("marker1","mark

## Add cluster to FCS files (known as "upsampling")
#upsample_file_path <- paste(density_file_path,"cluster.fcs",sep=".")
#SPADE.addClusterToFCS(density_file_path, upsample_file_path, cells_file_path, cols = c("marker1","marker2"))
```

`SPADE.addDensityToFCS` *Annotate FCS file with local density of each observation*

Description

Compute the local density of observation and incorporate the result as a new parameter to the FCS file. The local density is modeled as an integer count of the number of other observations within a specified distance of the observation.

Usage

```
SPADE.addDensityToFCS(infilename, outfilename, cols = NULL, arcsinh_cofactor=NULL, transforms=flowCor
```

Arguments

`infilename`	Name of the input FCS file
`outfilename`	Name of the output FCS file
`cols`	Usually a vector of strings specifying the columns to be used in the density calculation, e.g., c("(Cd110)D","(Cs111)D"). Strings will be matched against the parameter names extracted from the FCS file. The default=NULL will use all parameters.

`arcsinh_cofactor`	DEPRECATED. Cofactor used in arcsinh transform `asinh(data/arcsinh_cofactor)` of data.
`transforms`	Transform object to apply to data. A single transform object will be applied to all channels. To apply different transforms to specific channels use a named vector of transform objects (where names are parameter names).
`kernel_mult`	Multiplier of the minimum median distance within which other observations are counted towards the density
`apprx_mult`	Multiplier of the minimum median distance within which observations are approximated to have the same density
`med_samples`	Number of observations used to estimate the minimum median distance
`comp`	Apply compensation matrix if present in SPILL or SPILLOVER keywords

Value

The name of the written file is returned

Author(s)

Michael Linderman

Examples

```
# Not run
## Load two-parameters sample data included in package
#data_file_path = paste(installed.packages()["spade","LibPath"],"spade","extdata","SimulatedRawData.fcs",sep=.P
#
#output_dir <- tempdir()
#
## Compute and annotate FCS file with density
#density_file_path <- paste(output_dir,.Platform$file.sep,basename(data_file_path),".density.fcs",sep="")
#SPADE.addDensityToFCS(data_file_path, density_file_path, cols=c("marker1","marker2"))
```

`SPADE.annotateGraph` *Add attributes to graph*

Description

Add specific and arbitrary attributes to a graph

Usage

```
SPADE.annotateGraph(graph, layout = NULL, anno)
```

Arguments

`graph`	The graph object to work on. Note that the original graph is never modified, a new graph object is returned instead; if you don't assign it to a variable your modifications will be lost!
`layout`	Optional numeric matrix with vertex x,y positions with the same number of rows as vertices and at least two columns, the x and y positions.
`anno`	List of annotations to add to the graph. Each entry in list must have a name and must be a matrix. All matrices must have the same number of rows as vertices. List entry name plus column names are used as attribute names (unless they match, then just the column name is used).

Details

Add specific arbitrary attributes to a graph.

Value

A new graph object with the attributes added.

Author(s)

Michael Linderman

See Also

`set.graph.attribute`, `set.vertex.attribute`, `set.edge.attribute`

Examples

```
## Load two-parameters sample data included in package
#data_file_path = paste(installed.packages()["spade","LibPath"],"spade","extdata","SimulatedRawData.fcs",sep=.P

## Run basic SPADE analyses, clustering on two parameters. Annotated graphs will be
## in output_dir. See SPADE.plot.trees to generate PDFs of annotated graphs.
#output_dir <- tempdir()
#SPADE.driver(data_file_path, out_dir=output_dir, cluster_cols=c("marker1","marker2"))

## Add additional parameters to output graphs using SPADE.annotateGraph
#old_graph <- igraph:::read.graph(paste(output_dir,"SimulatedRawData.fcs.density.fcs.cluster.fcs.medians.gml",s
#new_graph <- SPADE.annotateGraph(old_graph, layout=igraph:::layout.kamada.kawai(old_graph), anno=list(demo=matr
```

`SPADE.downsampleFCS` *Downsample observcations in a FCS file according to density parameter*

Description

Downsample the observations in a FCS file according to a previously computed density parameter. The goal is to produce a smaller set of observations with similar density. Downsampling is independent of how the density is modeled.

Usage

```
SPADE.downsampleFCS(infilename, outfilename,
      exclude_pctile = 0.01, target_pctile = 0.05,
      desired_samples = NULL)
```

Arguments

`infilename`	Name of the input FCS file. Must have a parameter named "density".
`outfilename`	Name of the output FCS file
`exclude_pctile`	Numeric value in [0,1]. Densities below this percentile will be excluded.
`target_pctile`	Numeric value in [0,1]. Densities below this percentile, but above '`exclude_pctile`' will be retained. Only meaningful if '`desired_samples`' is `NULL`.
`desired_samples`	Desired number of samples. If set to integer value, the target percentile will be set internally to downsample to approximately the desired number of samples.

Value

The name of the written file is returned

Note

Underlying implementations have been parallelized with OpenMP. Set `OMP_NUM_THREADS` in environment to control the number of threads used.

Author(s)

Michael Linderman

See Also

`SPADE.addDensityToFCS`

Examples

```
# Not run
## Load two-parameters sample data included in package
#data_file_path = paste(installed.packages()["spade","LibPath"],"spade","extdata","SimulatedRawData.fcs",sep=.P

#output_dir <- tempdir()
#
## Compute and annotate FCS file with density
#density_file_path <- paste(output_dir,.Platform$file.sep,basename(data_file_path),".density.fcs",sep="")
#SPADE.addDensityToFCS(data_file_path, density_file_path, cols=c("marker1","marker2"))

## Downsample FCS file based on density
#downsample_file_path <- paste(output_dir,.Platform$file.sep,basename(data_file_path),".density.fcs",sep="")
#SPADE.downsampleFCS(density_file_path, downsample_file_path)
```

`SPADE.driver` *SPADE workflow driver*

Description

A function to drive the SPADE workflow. Produces graphs annoated with parameter medians and fold change.

Usage

```
SPADE.driver(files, file_pattern="*.fcs", out_dir=".", cluster_cols=NULL, panels=NULL, comp=TRUE, arc
```

Arguments

`files`	Either a vector of FCS files, or a directory. If a directory, all of the *.fcs files in the directory are processed.
`file_pattern`	Wildcard pattern to match file if `files` is a director
`out_dir`	Directory where output files are written. Will be created if it does not exist.
`cluster_cols`	Usually a vector of strings specifying the columns to be used in the clustering, e.g., c("(Cd110)D","(Cs111)D"). Strings will be matched against the parameter names extracted from the FCS file. The default=NULL will use all parameters.
`panels`	List of panels for median and fold change calculations. See details for specific structure. If NULL, medians are computed for all markers in all files.
`comp`	Apply compensation matrix if present in SPILL or SPILLOVER keywords
`arcsinh_cofactor`	DEPRECATED. Cofactor used in arcsinh transform `asinh(data/arcsinh_cofactor)` of data.
`transforms`	Transform object to apply to data. A single transform object will be applied to all channels. To apply different transforms to specific channels use a named vector of transform objects (where names are parameter names).

`downsampling_samples`	Desired number of samples remaining after downsampling files
`downsampling_exclude_pctile`	Numeric value in [0,1]. Densities below this percentile will be excluded during downsampling.
`downsampling_target_pctile`	Numeric value in [0,1]. Densities below this percentile, but above 'exclude_pctile' will be retained during downsampling. Only meaningful if 'downsampling_samples' is 'NULL'.
`k`	Desirec number of clusters. Algorithm might create between [k/2,3k/2] clusters.
`clustering_samples`	Desired number of samples to be used in clustering.
`layout`	Layout function
`pctile_color`	A two element vector specifying lower and upper percentiles that should be used to set the color scale. Values below and above these percentiles will be forced to the 'smallest' and 'largest' color respectively. Not in effect if 'scale' is specified. Relevant for downstream tools that used global value ranges produced by driver.

Details

The `panels` argument must be null or a list of panel descriptors, which are themselves lists containing at minimum a vector of panel files and median cols. An example minimum panels argument would be `list( list(panel_files="basal.fcs", median_cols=NULL))` . `panel_files` is a single file name or vector of file names in the experiment. `median_cols` is similar to the `cluster_cols` argument. Each panel descriptor can optionally specifiy reference_files and columns for fold change analysis. An example full panel descriptor would be `list( list(panel_files=c("basal.fcs", "stim.f` `fold_cols` is similar to the `cluster_cols` argument. `reference_files` is a single file name or vector of file names in the experiment and in the `panel_files` for this experiment. `median_cols` and `fold_cols` are only interpreted in the context on their panel files, and so partially overlapping panels are possible. However, all the files specified within a panel must have the cluster, median and fold change parameters specified.

Value

NULL

Author(s)

Michael Linderman

Examples

```
# Load two-parameters sample data included in package
data_file_path = system.file(file.path("extdata","SimulatedRawData.fcs"), package = "spade")

# Run basic SPADE analyses, clustering on two parameters. Annotated graphs will be
# in output_dir. See SPADE.plot.trees to generate PDFs of annotated graphs.
output_dir <- tempdir()
SPADE.driver(data_file_path, out_dir=output_dir, cluster_cols=c("marker1","marker2"))
```

`SPADE.FCSToTree` *Cluster and build minimum spanning tree from data in FCS files*

Description

Hierarchically cluster observations in a set of FCS files and build a minimum spanning tree connecting those clusters.

Usage

```
SPADE.FCSToTree(infilenames, outfilename, graphfilename, clusterfilename,
  cols = NULL, k = 200, arcsinh_cofactor=NULL, transforms=flowCore::arcsinhTransform(a=0, b=0.2),
   desired_samples = 50000, comp=TRUE)
```

Arguments

`infilenames`	Vector of FCS file names that should be used as input
`outfilename`	Name of FCS file to write subset of cells used for clustering along with their cluster assignment
`graphfilename`	Name of file to write gml graph description
`clusterfilename`	Name of file to write table of cluster centers
`cols`	Usually a vector of strings specifying the columns to be used in the density calculation, e.g., c("(Cd110)D","(Cs111)D"). Strings will be matched against the parameter names extracted from the FCS file. The default=NULL will use all parameters.
`k`	Desired number of clusters. Algorithm might create between [k/2,3k/2] clusters.
`arcsinh_cofactor`	DEPRECATED. Cofactor used in arcsinh transform `asinh(data/arcsinh_cofactor)` of data.
`transforms`	Transform object to apply to data. A single transform object will be applied to all channels. To apply different transforms to specific channels use a named vector of transform objects (where names are parameter names).
`desired_samples`	Desired number of samples to be used in clustering. Usually leave at default.
`comp`	Apply compensation matrix if present in SPILL or SPILLOVER keywords

Value

None.

Note

Underlying implementations have been parallelized with OpenMP. Set `OMP_NUM_THREADS` in environment to control the number of threads used. Implementation can be very memory intensive.

Author(s)

Michael Linderman

See Also

`SPADE.downsampleFCS`

Examples

```
## Load two-parameters sample data included in package
#data_file_path = paste(installed.packages()["spade","LibPath"],"spade","extdata","SimulatedRawData.fcs",sep=.P

#output_dir <- tempdir()
#
## Compute and annotate FCS file with density
#density_file_path <- paste(output_dir,.Platform$file.sep,basename(data_file_path),".density.fcs",sep="")
#SPADE.addDensityToFCS(data_file_path, density_file_path, cols=c("marker1","marker2"))

## Downsample FCS file based on density
#downsample_file_path <- paste(output_dir,.Platform$file.sep,basename(data_file_path),".density.fcs",sep="")
#SPADE.downsampleFCS(density_file_path, downsample_file_path)

## Create tree from downsampled FCS file
#cells_file_path <- paste(output_dir,"clusters.fcs",sep="")
#clust_file_path <- paste(output_dir,"clusters.table",sep="")
#graph_file_path <- paste(output_dir,"mst.gml",sep="")
#SPADE.FCSToTree(downsample_file_path, cells_file_path, graph_file_path, clust_file_path, cols=c("marker1","mark
```

`SPADE.flattenAnnotations` *Flatten list of annotations to matrix*

Description

Helper function for flattening list of annotations

Usage

```
SPADE.flattenAnnotations(annotations)
```

Arguments

`annotations`	A list of annotation matrices. All matrices must have the same number of rows.

Value

Single matrix of annotations

Author(s)

Michael Linderman

Examples

```
# Not run
## Load two-parameters sample data included in package
#data_file_path = paste(installed.packages()["spade","LibPath"],"spade","extdata","SimulatedRawData.fcs",sep=.P

## Run basic SPADE analyses, clustering on two parameters.
#output_dir <- tempdir()
#SPADE.driver(data_file_path, out_dir=output_dir, cluster_cols=c("marker1","marker2"))

## Compute medians, counts and other parameters from processed files
#upsampled_file_path <- paste(output_dir,.Platform$file.sep,basename(data_file_path),".density.fcs.cluster.fcs"
#mst_graph <- igraph:::read.graph(paste(output_dir,"mst.gml",sep=.Platform$file.sep),format="gml")
#anno <- SPADE.markerMedians(upsampled_file_path, igraph:::vcount(mst_graph), cols = c("marker1","marker2"))

## Flatten annotations so they can easily be saved in table
#flat_anno <- SPADE.flattenAnnotations(anno)
```

`SPADE.installPlugin` *Install CytoSPADE Cytoscape plugin*

Description

Install, CytoSPADE, the Cytoscape plugin for working with SPADE that is distributed with the SPADE R package. CytSPADE provides a GUI for setting-up SPADE analyses and interactively visualizing the results.

Usage

```
SPADE.installPlugin(cytoscape_path)
```

Arguments

`cytoscape_path`	Path to your Cytoscape install, e.g., on OSX it is typically something like '/Applications/Cytoscape_v2.8.1'

Details

Copies the Cytoscape plugin file distributed with the SPADE R package to the Cytoscape plugin directory.

Value

Logical indicating success of the copy operation.

Author(s)

Michael Linderman

Examples

```
# On OSX:
# SPADE.installPlugin("/Applications/Cytoscape_v2.8.1/")
```

`SPADE.layout.arch` *Generate coordinates for plotting graphs*

Description

Performing "arch" layouts of graph vertices

Usage

```
SPADE.layout.arch(mst_graph)
```

Arguments

`mst_graph`	The graph to layout. Must be acyclic and undirected.

Details

These functions calculate the coordinates of the vertices for a graph.

`layout.arch` lays out the longest chain of the graph, the "backbone", on an arch, and the "side chains" as trees normal to that backbone.

Value

All these functions return a numeric matrix with at least two columns, x and y positions, and the same number of lines as the number of vertices.

Author(s)

Michael Linderman

See Also

`SPADE.annotateGraph`

Examples

```
# Not run
## Load two-parameters sample data included in package
#data_file_path = paste(installed.packages()[”spade”,”LibPath”],”spade”,”extdata”,”SimulatedRawData.fcs”,sep=.P

## Run basic SPADE analyses, clustering on two parameters.
#output_dir <- tempdir()
#SPADE.driver(data_file_path, out_dir=output_dir, cluster_cols=c(”marker1”,”marker2”))

## Generate PDFs of annotated graphs (into output_dir/pdf) using arch layout
#mst_graph <- igraph:::read.graph(paste(output_dir,”mst.gml”,sep=.Platform$file.sep),format=”gml”)
#SPADE.plot.trees(mst_graph, output_dir, out_dir=paste(output_dir,”pdf”,sep=.Platform$file.sep), layout=SPADE.l
```

`SPADE.markerMedians` *Compute marker medians, coefficient of variations and counts for clusters*

Description

Compute the marker medians, coefficients of variation and observations counts for cluster annoated FCS files.

Usage

```
SPADE.markerMedians(files, num.clusters, cols = NULL, arcsinh_cofactor=NULL, transforms=flowCore::arc
SPADE.annotateMarkers(files, cols = NULL, arcsinh_cofactor = 5)
```

Arguments

`files`	Name of input FCS file or vector of input FCS file names. FCS files must have "cluster" column.
`num.clusters`	Number of clusters. Note not all clusters need to be present in all files.
`cols`	Usually a vector of strings specifying the columns to be used in the density calculation, e.g., c("(Cd110)D","(Cs111)D"). Strings will be matched against the parameter names extracted from the FCS file. The default=NULL will use all parameters.
`arcsinh_cofactor`	DEPRECATED. Cofactor used in arcsinh transform `asinh(data/arcsinh_cofactor)` of data.
`transforms`	Transform object to apply to data. A single transform object will be applied to all channels. To apply different transforms to specific channels use a named vector of transform objects (where names are parameter names).
`cluster_cols`	A vector of strings specifying columns that should be marked as having been used in clustering
`comp`	Apply compensation matrix if present in SPILL or SPILLOVER keywords

Details

SPADE.annotateMarkers is deprecated.

Value

List with:

`count`	Matrix of observation count for clusters
`percenttotal`	Matrix of percent of total number of cells [0-100] in each cluster
`medians`	Matrix of medians for specified columns
`cvs`	Matrix of coefficient of variation (CV), 100*sd(data)/abs(mean(data)), for specified columns

Author(s)

Michael Linderman

See Also

`SPADE.addClusterToFCS`, `SPADE.annotateGraph`

Examples

```
# Not run
## Load two-parameters sample data included in package
#data_file_path = paste(installed.packages()["spade","LibPath"],"spade","extdata","SimulatedRawData.fcs",sep=.P

## Run basic SPADE analyses, clustering on two parameters.
#output_dir <- tempdir()
#SPADE.driver(data_file_path, out_dir=output_dir, cluster_cols=c("marker1","marker2"))

## Compute medians, counts and other parameters from processed files
#upsampled_file_path <- paste(output_dir,.Platform$file.sep,basename(data_file_path),".density.fcs.cluster.fcs"
#mst_graph <- igraph:::read.graph(paste(output_dir,"mst.gml",sep=.Platform$file.sep),format="gml")
#anno <- SPADE.markerMedians(upsampled_file_path, igraph:::vcount(mst_graph), cols = c("marker1","marker2"))
```

`SPADE.plot.trees` *Plot trees with annotated vertices*

Description

Plot trees for each vertex annotation setting vertex size and color based on the particular annotation.

Usage

```
SPADE.plot.trees(graph, files, file_pattern = "*anno.Rsave", out_dir = ".", layout = SPADE.layout.arch
```

Arguments

`graph`	iGraph graph object
`files`	Either a vector of save annotation files or a directory. If a directory, all of the files matching the `pattern` wildcard pattern are processesd.
`file_pattern`	Wildcard pattern to match files if `files` is a directory.
`out_dir`	Directory where output files are written. Will be created if it does not exist.
`layout`	Either a function or a numeric matrix specifying how vertices are placed on plot. If it is a matrix, the matrix must have two columns, x and y position, and as many rows as vertices. If `layout` is a function, it will be called with an igraph graph as the single parameter.
`attr_pattern`	A regular expression that matches the attributes that should be plotted for each graph. Parameter names matching regex "median|fraction|cvs" will be plotted with a scale range set to [min, max] for that attribute, while all other parameters will be plotted on a centered scale with the range [abs(min(parameter values),max(parameter values)), abs(min(parameter values),max(parameter values))]
`scale`	A two element vector, e.g. c(-1,1), specifying low and upper bound for color scale. Values below and above these bounds will be forced to the 'smallest' and 'largest' color respectively. If specified, overrides 'pctile_color'.
`pctile_color`	A two element vector specifying lower and upper percentiles that should be used to set the color scale. Values below and above these percentiles will be forced to the 'smallest' and 'largest' color respectively. Not in effect if 'scale' is specified.
`normalize`	A string (either "global" or "local"), specifying color scale normalization. Setting to "global" will set the scale range to the global min/max of all GML files in the folder, while "local" will set the scale range to the min/max of the particular GML file being plotted.
`size_scale_factor`	A scale factor for node size in drawing. Current function for node size: `percenttotal[i]/(max(percent`
`edge.color`	Set the edge color. See `igraph.plotting` for more details.
`bare`	Boolean specifying whether to omit titles and gradient legend.
`palette`	A string (either "jet" or "bluered"), specifying color palette for nodes. "bluered" tends to show up better on LCD projectors.

Author(s)

Michael Linderman

See Also

`SPADE.driver`

Examples

```
## Not run:
# Load two-parameters sample data included in package
data_file_path = paste(installed.packages()["spade","LibPath"],"spade","extdata","SimulatedRawData.fcs",sep=.Pl

# Run basic SPADE analyses, clustering on two parameters.
output_dir <- tempdir()
SPADE.driver(data_file_path, out_dir=output_dir, cluster_cols=c("marker1","marker2"))

# Generate PDFs of annotated graphs (into output_dir/pdf)
mst_graph <- igraph:::read.graph(paste(output_dir,"mst.gml",sep=.Platform$file.sep),format="gml")
SPADE.plot.trees(mst_graph, output_dir, out_dir=paste(output_dir,"pdf",sep=.Platform$file.sep), layout=igraph::

## End(Not run)
```

`SPADE.read.FCS` *Read an FCS File*

Description

Read and FCS file, correcting problematic column names and applying compensation if requested. A wrapper around read.FCS.

Usage

```
SPADE.read.FCS(file, comp = TRUE, verbose = FALSE, ...)
```

Arguments

`file`	FCS filename
`comp`	boolean indicating whether to apply compensation
`verbose`	boolean indicating whether to not suppress warnings from upstream FCS file reader.
`...`	additional arguments are passed through to read.FCS

Details

A wrapper around read.FCS that copies FCS parameter "short name" to description when none is supplied, and applies compensation matrix stored in the SPILL or SPILLOVER keyword.

Value

a flowFrame object.

Author(s)

Michael Linderman <michael.linderman@mssm.edu>

See Also

`read.FCS`

Examples

```
fcs <- SPADE.read.FCS(system.file(file.path("extdata","SimulatedRawData.fcs"), package = "spade"))
```

`SPADE.write.graph` *Writing the graph to a file in some format*

Description

General function for exporting graphs to foreign file formats, however at present only the GML format is implemented.

Usage

```
SPADE.write.graph(graph, file = "", format = c("gml"))
```

Arguments

`graph`	The graph to export
`file`	A connection or a string giving the file name to write the graph to.
`format`	Character string giving the file format.

Details

GML is general textual format for graphs.

The vertex and edge attributes are written to the file if they are numeric or strings. Currently only the graphics struct is supported, and only for vertices; `graphics.x` indicates an `x` attribute in the `graphics` struct.

Value

A NULL, invisibly

Author(s)

Michael Linderman

See Also

`write.graph`

Examples

```
# Not run
## Load two-parameters sample data included in package
#data_file_path = paste(installed.packages()["spade","LibPath"],"spade","extdata","SimulatedRawData.fcs",sep=.P

## Run basic SPADE analyses, clustering on two parameters.
#output_dir <- tempdir()
#SPADE.driver(data_file_path, out_dir=output_dir, cluster_cols=c("marker1","marker2"))

## Read and write minimum spanning tree graph
#mst_graph <- igraph:::read.graph(paste(output_dir,"mst.gml",sep=.Platform$file.sep),format="gml")
#SPADE.write.graph(mst_graph, file = paste(output_dir,"new_mst.gml",sep=.Platform$file.sep), format = c("gml"))
```

Index